Self-Organized Backpressure Routing for the Wireless Mesh Backhaul of Small Cells

Ph.D. Dissertation by

José Núñez-Martínez

Advisor: Dr Josep Mangues Bafalluy
Tutor: Prof. Jordi Domingo-Pascual

Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya (UPC)

Barcelona, 2014

A mi familia

“Complexity is your enemy. Any fool can make something complicated. It is hard to make something simple.”

-Richard Brason

Abstract

The ever increasing demand for wireless data services has given a starring role to dense small cell (SC) deployments for mobile networks, as increasing frequency re-use by reducing cell size has historically been the most effective and simple way to increase capacity. Such densification entails challenges at the Transport Network Layer (TNL), which carries packets throughout the network, since hard-wired deployments of small cells prove to be cost-unfeasible and inflexible in some scenarios.

The goal of this thesis is, precisely, to provide cost-effective and dynamic solutions for the TNL that drastically improve the performance of dense and semi-planned SC deployments. One approach to decrease costs and augment the dynamicity at the TNL is the creation of a wireless mesh backhaul amongst SCs to carry control and data plane traffic towards/from the core network. Unfortunately, these low-cost SC deployments preclude the use of current TNL routing approaches such as Multiprotocol Label Switching Traffic Profile (MPLS-TP), which was originally designed for hard-wired SC deployments. In particular, one of the main problems is that these schemes are unable to provide an even network resource consumption, which in wireless environments can lead to a substantial degradation of key network performance metrics for Mobile Network Operators. The equivalent of distributing load across resources in SC deployments is making better use of available paths, and so exploiting the capacity offered by the wireless mesh backhaul formed amongst SCs.

To tackle such uneven consumption of network resources, this thesis presents the design, implementation, and extensive evaluation of a self-organized backpressure routing protocol explicitly designed for the wireless mesh backhaul formed amongst the wireless links of SCs. Whilst backpressure routing in theory promises throughput optimality, its implementation complexity introduces several concerns, such as scalability, large end-to-end latencies, and centralization of all the network state. To address these issues, we present a throughput suboptimal yet scalable, decentralized, low-overhead, and low-complexity backpressure routing scheme. More specifically, the contributions in this thesis can be summarized as follows:

- We formulate the routing problem for the wireless mesh backhaul from a stochastic network optimization perspective, and solve the network optimization problem using the Lyapunov-drift-plus-penalty method. The Lyapunov drift refers to the difference of queue backlogs in the network between different time instants, whereas the penalty refers to the routing cost incurred by some network utility parameter to optimize. In our case, this parameter is based on minimizing the length of the path taken by packets to reach their intended destination.

- Rather than building routing tables, we leverage geolocation information as a key component to complement the minimization of the Lyapunov drift in a decentralized way. In fact, we observed that the combination of both components helps to mitigate backpressure limitations (e.g., scalability, centralization, and large end-to-end latencies).

- The drift-plus-penalty method uses a tunable optimization parameter that weight the relative importance of queue drift and routing cost. We find evidence that, in fact, this optimization parameter impacts the overall network performance. In light of this observation, we propose a self-organized controller based on locally available information and in the current packet being routed to tune such an optimization parameter under dynamic traffic demands. Thus, the goal of this heuristically built controller is to maintain the best trade-off between the Lyapunov drift and the penalty function to take into account the dynamic nature of semi-planned SC deployments.

- We propose low complexity heuristics to address problems that appear under different wireless mesh backhaul scenarios and conditions. The resulting decentralized scheme attains an even network resource consumption under a wide variety of SC deployments and conditions. Among others, we highlight the following:

  - The proposed decentralized routing scheme efficiently manages any-to-any dynamic traffic patterns. Instead of maintaining one data queue per traffic flow and routing tables, our scheme routes traffic between any pair of nodes in the network by managing a single queue per each node and 1-hop geolocation information. We, thus, propose to handle the management of traffic with low complexity.

  - We demonstrate that the routing scheme adapts to wireless backhaul topology dynamics. In particular, we consider regular and sparse deployments with homogeneous and heterogeneous wireless link rates. Note that the correct operation of the protocol under sparse deployments is of primal importance not only for showing resiliency under node failures, but for considering an energy efficient SC deployment, in which SCs are switch on and off at the management level.

- In an anycast network environment, the proposed solution offers inherent properties to perform gateway load balancing in the wireless mesh backhaul. In fact, the proposed solution maximizes opportunistically the use of the deployed gateways. The distinguishing feature of the combination of backpressure routing and anycast is scalability with the number of gateways, and an even use of gateways, despite being unaware of their location.

- In terms of performance comparison, our backpressure routing scheme performs better than SoA routing approaches. We conducted extensive and accurate simulations to compare the solutions proposed in this thesis against various SoA TNL routing approaches.

- Last but not least, we implemented and evaluated the backpressure routing strategy in a proof-of-concept. The prototype is based on an indoor wireless mesh backhaul formed amongst 12 SCs endowed with 3G and WiFi interfaces. Thus, we experimentally validated the contributions of the work conducted in this thesis under real-world conditions. In particular, we evaluated our proposed scheme under static and dynamic wireless mesh backhaul conditions.

Resumen

El aumento de la demanda de datos en servicios inalámbricos ha otorgado un rol de gran importancia al despliegue masivo de celdas pequeñas para redes mobiles, dado que decrementar el tamaño de las celdas para reusar las frecuencias ha sido históricamente la manera mas simple y efectiva de incrementar la capacidad disponible. Este aumento de la densidad conlleva ciertos retos a nivel de red de transporte, encargada de transportar los paquetes por la red, ya que despliegues cableados de celdas pequeñas tienen graves problemas para proporcionar un servicio flexible y de bajo coste.

El objetivo de esta tesis es, precisamente, aportar soluciones dinámicas y efectivas a nivel de coste para mejorar el rendimiento de despliegues masivos y de bajo grado de planificación de celdas pequeñas. Una aproximación para reducir costes y aumentar la dinamizad es mediante la creación de una red mallada inalámbrica entre las celdas pequeñas, las cuales pueden transportar tráfico tanto del plano de datos como el de control originado/destinado a/en la red principal. Desgraciadamente, estos despliegues excluyen los algoritmos actuales de enrutamiento a nivel de transporte, como por ejemplo MPLS-TP diseñado originalmente para despliegues cableados, son incapaces de gestionar eficientemente los recursos inalámbricos de red a nivel de transporte debido a la naturaleza dinámica y semi-planeada de estos despliegues. Consecuentemente, esto conlleva a una degradación substancial de las métricas clave en la evaluación del rendimiento de la red debido al mal uso de los recursos de red. Una de las causas principales de esta degradación es el consumo consumo desnivelado de los recursos de red. En este caso, el equivalente a distribuir entre los recursos implica hacer un uso eficiente de los caminos disponibles, y por lo tanto explotar la capacidad ofrecida por la red mallada inalámbrica formada entre las celdas pequeñas.

Para un consumo de recursos de red equilibrado y, por tanto, una máxima explotación de la red esta tesis presenta un algoritmo de auto-organización basado en backpressure, explícitamente diseñado para el entorno de red mallada inalámbrica formado por cada uno de los enlaces radio de transporte en las celdas pequeñas. Pese a que backpressure en teoría promete un caudal óptimo de red, su complejidad introduce varios problemas, tales como la escalabilidad y el manejo de toda la información de red en una

entidad central. Además, los protocolos de enrutamiento basados en backpressure pueden introducir un incremento del retardo innecesario debido al uso de caminos de una gran longitud de saltos. Para abordar estos problemas, presentamos un algoritmo de enrutamiento escalable y descentralizado también basado en backpressure, pero en este caso asistido por información adicional usada para mitigar las limitaciones de esta aproximación. Entre otras técnicas, esta tesis demuestra que principalmente la geolocalización combinada con un esquema basado en backpressure puede mitigar las limitaciones de este en términos de complejidad a la hora de implementarlo, así como el excesivo incremento de retardos sin perder las propiedades presentadas a nivel de obtención de caudal de red. Mas específicamente, las contribuciones que presenta esta tesis son las siguientes:

- La formulación del problema de enrutamiento desde un punto de vista de optimización de redes estocásticas, y la solución del problema de optimización usando el método de la desviación-mas-castigo de Lyapunov. La desviación de Lyapunov se refiere al diferencial de colas de paquetes entre las celdas pequeñas, mientras que el castigo se refiere a una función de coste incurrida por una parámetro útil de red a minimizar. En nuestro caso, este parámetro esta basado en la distancia en numero de saltos sufrida por los paquetes para llegar a su destino correspondiente.

- En lugar de construir tablas de encaminamiento, hacemos uso de información geográfica como un elemento clave para complementar la minimización de la desviación de Lyapunov (minimización los diferenciales de longitud de cola) de una manera descentralizada. De hecho, la combinación de ambos componentes ayuda a mitigar las limitaciones de backpressure (i.e.como escalabilidad, descentralización y grandes latencias).

- El método de la desviación-mas-castigo usa un parámetro configurable de optimización. Hallamos evidencias que, de hecho, este parámetro de optimización afecta el rendimiento de la red. A la luz de esta observación, se propone un controlador de este parámetro distribuido y auto-organizado basado puramente en información local y del paquete a encaminar bajo demandas de tráfico dinámicas. El objetivo de este controlador, construido de forma heurística es el calculo del compromiso mas adecuado entre la desviación de Lyapunov y el castigo teniendo en cuenta la naturaleza dinámica de estos despliegues semi-planeados de pequeñas celdas.

- Se extiende la solución propuesta con diferentes heurísticas de baja complejidad para atacar diferentes escenarios de red. Además de el uso de información de geolocalización, se usan capacidades anycast, e información contenida en el paquete a encaminar para permitir la adaptación a una amplia gama de despliegues de celdas pequeñas. La solución descentralizada resultante realiza un uso equilibrado de los recursos de red bajo una amplía gama de despliegues dinámicos de celdas

pequeñas. Entre otros, se resaltan los siguientes:

- El esquema resultante gestiona eficientemente patrones de comunicación dinámicas entre cualquier par de nodos de la red. En lugar de mantener una cola por cada flujo, nuestro esquema encamina tráfico entre cualquier par de nodes manteniendo una sola cola por node e informacion geographica local. Nuestra solucion es ,por tanto, de baja complejidad.

- Se demuestra que la solución propuesta se adapta a la dinamizad de un entorno de red inalámbrico. En particular, consideramos despliegues regulares y redes dispersas compuestas por enlaces inalámbricos de igual o diferentes velocidades. Es importante remarcar que un funcionamiento apropiado del protocolo en entornos dispersos de red es de gran importancia no solo para mostrar tolerancia a fallos en los nodos, sino para considerar entornos de celdas pequeñas eficientes a nivel de energía.

- Con el uso de capacidades anycast en la red de transporte, demostramos que el enrutamiento backpressure ofrece propiedades para conseguir balanceo de cargas en despliegues con múltiples puertas de enlace a nivel de transporte hacia la red central. De hecho, la solución propuesta maximiza de manera oportunista el uso de las diferente puertas de enlace sin conocer su ubicación concreta. La característica distintivas que posibilitan estas propiedades son la combinación de backpressure y capacidades anycast. Esto, a su vez, habilita la escalabilidad y una alta explotación de estas puertas de enlace a nivel de transporte, a pesar de no conocer su ubicación.

- A nivel comparativo, los esquemas concebidos en esta tesis mejoran drásticamente el rendimiento de las redes malladas inalámbricas de retorno debido a la maximización del uso de de sus recursos con respecto a otras aproximaciones de enrutamiento provistas por el estado del arte. En particular, se han realizado simulaciones extensivas y precisas para comparar las soluciones propuestas con los esquemas de enrutamiento provistos por el estado del arte que confirman las contribuciones anteriormente expuestas.

- Por ultimo, pero no menos importante, hemos realizado la implementación y evaluación empírica del protocolo de enrutamiento backpressure en un prototipo de red mallada inalámbrica de retorno para celdas pequeñas. El prototipo en cuestión consta de 12 celdas pequeñas desplegadas en un entorno interior. Esto valida a nivel experimental las contribuciones anteriormente expuestas.

Acknowledgements

The development and writing of this thesis could not have been possible without the help of a large number of people and some organizations that directly or indirectly supported the different stages of my work.

First and foremost, I would like to thank my advisor Dr. Josep Mangues-Bafalluy for his exquisite guidance, providing many insightful suggestions during the research process. Josep is someone that does not easily give up. Proof of that is his infinite patience during all the meetings and discussions we have shared through all these years. He has given me freedom to pursue research projects I thought were worth the effort, giving me full support no matter how crazy the project may seem. It has been a great pleasure and an enriching experience to work under his supervision.

All my enthusiasm for research comes from my time as undergraduate student. My initial research steps started at Centro de Comunicaciones Avanzadas de Banda Ancha (CCABA) within the Universidat Politecnica de Catalunya (UPC). I will forever be thankful to my BsC advisor and PhD tutor Prof. Jordi Domingo. I still think fondly of my time as undergraduate student in CCABA's lab. The enthusiasm of the whole CCABA team for research is contagious. Thank you Jordi, it was really enjoyable to form part of your lab.

I would like to convey my gratitude to the Centre Tecnologic de Telecomunicacions de Catalunya (CTTC). Doing the PhD here has been an unforgettable experience. Here I have found an once-in-a-lifetime environment for learning, overcoming my own limitations, and gaining persistence. I do not want to lose the chance of showing my gratitude to my colleague and dear friend Marc Portoles, who always had the door open for me. Our never-ending discussions in the office, in the coffee machine, or even in a bar were priceless. Again, I would like to personalize my gratitude also in Dr. Josep Mangues for hiring me as research engineer for his research group in CTTC. Josep gave me the opportunity of doing what I wanted to do. I want also to acknowledge the great help provided by other researchers and technical staff from CTTC as well. In particular, part of the experimentation conducted in this disserta-

tion are the result of the effort and the capacity of other people. In particular, I would like to show my gratitude to Jorge Baranda for its help in the last stages of this PhD. Thanks Jorge. I would also like to show my gratitude to other (or former) researchers from the Mobile Networks department for their continuous help. Thanks Jaime Ferragut, Marco Miozzo, Dr Paolo Dini, Dr Nicola Baldo, Dr Andrey Krendzel, and Manuel Requena. And special thanks to David Gregoratti for providing latex edition support. Thanks David.

Last but not least, I wish to thank CTTC director Prof. Miguel Angel Lagunas, former CTTC General Administrator Simo Aliana, and CTTC General Administrator Edgar Ainer, who are main responsible of the proper functioning of CTTC. Thank you Miguel Angel, Simo, and Edgar for building and manage such a good research facility.

During the PhD I have worked in several projects and met outstanding researchers. In particular, I would like to thank to Prof. Albert Banchs for the great collaboration within the DISTORSION project. I would also like to thank Dr. Frank Zdarsky, and Dr. Thierry Lestable for their insightful comments and collaboration with the BeFEMTO research project.

Over the last year and a half, a sector of the industry believed on the work described in this dissertation. I am extremely thankful to the company AVIAT Networks and specially to Alain Hourtane for supporting the continuation of this research beyond this PhD. Thanks Alain for making it happen, you have played a key role on the evolution of this work.

I end this never-ending acknowledgments thanking both the members of the internal PhD committee within the computer architecture department Dr. Albert Cabellos, Dr. José Maria Barceló, and Dr. Pere Barlet, and the members of the final PhD committee Dr. Albert Cabellos, Dr. Xavier Perez-Costa, and Dr. Pablo Serrano as well. They have provided priceless feedback to improve the quality of this dissertation.

Finally, but most of all, I thank my family for their endless love and unwavering support throughout the years.

Contents

List of Figures

Glossary

Tabulated below are the most important acronyms used in this dissertation.

3GPP	Third Generation Partnership Project
3G	Third Generation
4G	Fourth Generation
APS	Automatic Protection Switching
BCP	Backpressure Collection Protocol
CAPEX	Capital Expenditures
CN	Correspondent Node
COPE	Coding Opportunistically
DCAR	Distributed Coding-Aware Routing
DOLSR	Directional Optimized Link State Routing
DRPC	Dynamic Routing and Power Control
DSCP	Differentiated Services Code Point
DSR	Dynamic Source Routing
eNodeB	Evolved Node-B
EATT	Expected Anypath Transmission Time
EAX	Expected Any-path Transmissions
ECMP	Equal Cost Multipath
EECA	Energy-Efficient Control Algorithm
EMT	Expected Medium Time
EPC	Evolved Packet Core
EPS	Evolved Packet System
ETT	Expected Transmission Time
ETX	Expected Transmission Count
HeNodeB	Home evolved Node-B

EAR	Expected Advancement Rate
ECMP	Equal Cost MultiPath
EPC	Evolved
EPS	Evolved Packet System
ERPS	Ethernet Ring Protection Switching
ETSI	European Telecommunications Standards Institute
ExOR	Extremely Opportunistic Routing
EWMA	Exponentially Weighted Moving Average
E-UTRAN	Evolved Universal Terrestrial Radio Access Networks
GGR	Greedy Geographic Routing
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GTP	GPRS Tunneling Protocol
HeMS	HeNB Management System
HSS	Home Subscriber Server
iAWARE	Interference-Aware Metric
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISG	Industry Standardization Group
IS-IS	Intermediate System to Intermediate System
ITU	International Telecommunication Union
ITU-T	ITU-Telecommunication Standardization Sector
LACP	Link Aggregation Control Protocol
LAG	Link Aggregation Group
LHS	Left Hand Side
LIFO	Last In First Out
LOS	Line-Of-Sight
LSP	Label Switch Path
LTE	Long Term Evolution
LTE-A	Long Term Evolution-Advanced
SON	Self-organizing network
MaLB	Mac-aware and Load Balanced Routing
MCC	Multipath Code Casting

MGOR	Multi-Rate Opportunistic Routing
MIC	Metric of Interference and Channel-switching
MPLS	Multi-Protocol Label Switching
MPLS-TP	MPLS Traffic Profile
MME	Mobile Management Entity
MNL	Mobile Network Layer
MNO	Mobile Network Operator
MORE	MAC-independent Opportunistic Routing & Encoding
M-LAG	Multi-Chassis/Multi-System Link Aggregation Group
MR-AODV	Multi-Radio Adhoc On-Demand Distance Vector
MR-LQSR	Multi-Radio Link Quality Source Routing
NFV	Network Function Virtualization
NGMN	Next Generation Mobile Networks
NLOS	Non-Line-Of-Sight
OPEX	Operational Expenditures
P-GW	Packet Data Network GateWay
QoS	Quality of Service
OAM	Operation, Administration, and Management
ORRP	Orthogonal Rendezvous Routing Protocol
OSPF	Open Shortest Path First
PMIP	Proxy Mobile IP
P-MME	Proxy Mobile Management Entity
P-SGW	Proxy Serving GateWay
PDN-GW	Packet Data Network GateWay
RAN	Radio Access Network
RAT	Radio Access Technology
RHS	Right Hand Side
RNC	Radio Network Controller
ROMA	Routing over Multi-Radio Access Networks
ROMER	Resilient Opportunistic Mesh Routing
RPC	Remaining Path Cost
RRC	Radio Resource Control
RSE	RAN Sharing Enhancements
SDN	Software Defined Networks

SC	Small Cell
SMAF	Shortest Multirate Anypath Forwarding
SoA	State of the Art
SOAR	Simple and Opportunistic Adaptive Routing
SON	Self-Organized Networks
SPB	Shortest Path Bridging
STP	Spanning Tree Protocol
S-GW	Serving GateWay
TDMA	Time Division Multiplexing Access
TIC	Topology and Interference aware Channel assignment
TNL	Transport Network Layer
TRILL	TRansparent Interconnect of Lots of Links
UE	User Equipment
VRR	Virtual Ring Routing
WCETT	Weighted Cumulative Expected Transmission Time
WiFi	Wireless Fidelity
WLAN	Wireless LAN
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

This chapter introduces in section 1.1 the context of this thesis and summarizes its motivations. This section also gives a general introduction of what is the research problem tackled, as well as the main applications that a solution to this problem has for industry and academia. Section 1.2 provides a brief description of the structure and the content of this thesis. And in section 1.3, we provide the main contributions coming out from this dissertation.

1.1 Motivation

The focus of this thesis lays in the context of dense and high-capacity small cell (SC) backhaul deployments. In order to cope with ever-increasing wireless data services, capacity-oriented mobile network deployments are needed. Capacity-oriented mobile networks mandate for dense SC deployments, since reducing cell radii has traditionally been an effective way to increase capacity, given the limited spectrum availability. Not only the ever-increasing data demands but also due to the non uniform mobile data traffic distribution, Mobile Network Operators (MNO) actually need to increase their capacity density (i.e., Mbps per km^2). It is important to note that these increasing capacity density requirements

require not only dense SC deployments with access capacity, but the corresponding backhaul capacity to transport traffic from/to the core network. Since it is unlikely that fiber reaches every SC (e.g., those deployed in lampposts), a wireless mesh network acting as backhaul is expected to become popular.

At a high-level, the research problem that this thesis addresses is: *how can an operator make the most out of the wireless mesh backhaul resources deployed?* In addition to this efficient use of resources, the solution provided must be practical enough to ease the roll-out of small cells. Furthermore, MNOs aim at increasing the scalability and dynamicity of these networks, while decreasing planning, configuration, and maintenance costs for the optimization of these deployments.

This thesis answers this question by designing and evaluating a self-organized backpressure routing protocol for the Transport Network Layer (TNL) that achieves scalability, adaptability, implementability, and improvement of key performance metrics against SoA routing approaches. The resulting protocol makes the most out of the wireless resources precisely because of two reasons. First, our self-organized backpressure routing scheme dynamically increases and shrinks the wireless network resource consumption according to traffic demands. Second, this functionality is achieved by using a decentralized method that solely requires HELLO based communication between neighboring SCs. Thus, practically all wireless backhaul resources are used to transport traffic generated by the Mobile Network Layer (MNL). Another remarkable aspect of this thesis is its research methodology. This thesis builds a practical solution on a proof-of-concept mesh backhaul prototype starting out from a theoretical perspective based on stochastic network optimization and extensive simulations.

The research question motivated and introduced above represents a big impact for the research community. A routing protocol that makes the most out of the network resources, no matter the wireless backhaul dynamicity, remains so far unexplored. On the other hand, wireless vendors and operators represent the most common entities that can benefit from the research aimed in this dissertation. Indeed, the outcomes of the work conducted in this dissertation have attracted the interest of AVIAT Networks [1], a leading vendor of wireless backhaul equipment. Having said that, it is also worth mentioning that the applications go beyond economical gains, including also social gains.

1.2 Outline of the dissertation

We organized the work conducted in this dissertation into eleven chapters. Seven of these chapters entail the main research work, which is in turn structured into three main parts. The first part encloses chapters 2, 3, 4, and 5, focused on providing background context, reviewing related research work, and

stating the specific research problem tackled in this thesis. Part II includes the approach taken to solve the research problem in chapter 7, and its refinement and validation through simulation throughout a wide variety of conditions in chapter 8. Finally, Part III includes the proof-of-concept implementation in chapter 9, and its evaluation in chapter 10. A more detailed explanation of all the chapters follows.

Chapter 2 provides some basic background on the main concepts around the mobile backhaul, defining the context in which we developed the work presented in this thesis.

Chapter 3 details current transport schemes used for the mobile backhaul that could better suit a wireless mesh backhaul.

Chapter 4 details the related research work on wireless routing from a data network perspective. We provide a classification of related work, and identify the main building blocks composing a routing protocol.

Chapter 5 describes the research problem, and uses the work presented in chapters 3 and 4 to demonstrate that the stated research problem is not solved yet. Besides, its validity and main applications of interest are discussed.

Chapter 6 provides the main changes required in the current 3GPP architecture for building a mesh backhaul amongst small cells.

Chapter 7 presents our solution to the question stated in chapter 5. We formulate the routing problem, and propose a solution to that problem using the Lyapunov drift-plus-penalty method.

Chapter 8 is devoted to the adaptation of the resulting self-organized backpressure routing policy to the dynamics present in wireless mesh backhauls under variable traffic demands. We characterize the performance of the routing protocol in wireless mesh backhauls under a wide variety of conditions. This evaluation is carried out through simulations, providing a comparison with SoA routing approaches.

Chapter 9 provides the framework required to implement the backpressure routing policy in a real mesh backhaul testbed. An introduction to the main building blocks of the implementation is provided first, so as to discuss in the following sections the modifications required to roll out the building blocks in a real testbed.

Chapter 10 provides a description of the wireless mesh backhaul testbed, and also the experimental results obtained with the backpressure routing policy.

Chapter 11 summarizes the work carried out in this thesis, states its main conclusions, and presents some of the future lines of work.

1.3 Contributions

In what follows, we provide a birds-eye view of the key contributions obtained from this thesis, which are ordered from most to least important. Note that a more precise description of the contributions of this thesis can be found in chapter 11.

1. The research problem stated in chapter 5 is solved using a practical and self-organized TNL routing approach, resulting in a distributed algorithm that makes the most out of the network resources. Chapter 7 formulates the theoretical roots in which we lay such TNL routing approach. These results have partially been published in [2]. And a survey on the related work in the tacked research problem has been published in [3].

2. One of the main agents enabling the practicality of our mechanism are the low complexity heuristics proposed in chapter 8. We demonstrated with an accurate simulator (see publication [4] for a demonstration of its accuracy) in chapter 8 that the resulting TNL routing policy improves SoA routing policies in scenarios with varying traffic demands and patterns [5], a variable number of gateways [6], and dynamic wireless backhaul deployments [7, 8].

3. As demonstrated in chapter 8, the solution maintains implementability, scalability, decentralization, and self-organization over all the aforementioned wireless mesh backhaul scenarios.

4. We solved the 3GPP architectural implications posed by the concept of a wireless mesh backhaul amongst small cells in chapter 6. The novel concept of network of small cells was presented in [9] and [10], whereas the contributions derived from the 3GPP architectural implications of a network of small cell were published in [11] and [12].

5. We implemented the resulting routing policy in a 12-node wireless mesh backhaul testbed facing the constraints of the practical backhaul. A description of the implementation and problems faced can be found in chapter 9. Such contributions in terms of implementation were also published in [13].

6. We demonstrated by conducting real experiments in a testbed [14–16] the performance of the resulting routing policy. Experimental results and consequent discussion detailed in in chapter 10 confirm the properties showed by our proposed mechanism through simulation in chapter 8.

Part I

Background and Problem Statement

Chapter 2

Mobile Backhaul Review

The transport infrastructure between a Radio Access Network (RAN) and a Core Network (CN) is called the mobile backhaul. In turn, the transport infrastructure is subdivided in two main entities: the access domain and the aggregation domain. This chapter emphasizes the mobile architectural entities that are crucial to understand the work conducted at the transport layer in this dissertation.

Section 2.1 introduces the Evolved Packet System (EPS) architecture. This section introduces the mobile architectural entities, which are defined by the 3GPP, related with the mobile backhaul. Section 2.2 provides a brief overview of the mobile network architecture and the main requirements of the mobile backhaul. Section 2.3 lays its focus on the access domain and emphasizes the advantages of a wireless mesh backhaul as access domain for emerging dense deployments of small cells (SCs).

2.1 The Evolved Packet System

The Evolved Packet System (EPS) includes the Evolved Packet Core (EPC) and Evolved Universal Terrestrial Radio Access Networks (E-UTRAN). An E-UTRAN can include two types of base stations, named as eNodeB (eNB) or macro cell and Home evolved Node B (HeNB) or small cells (SC). For

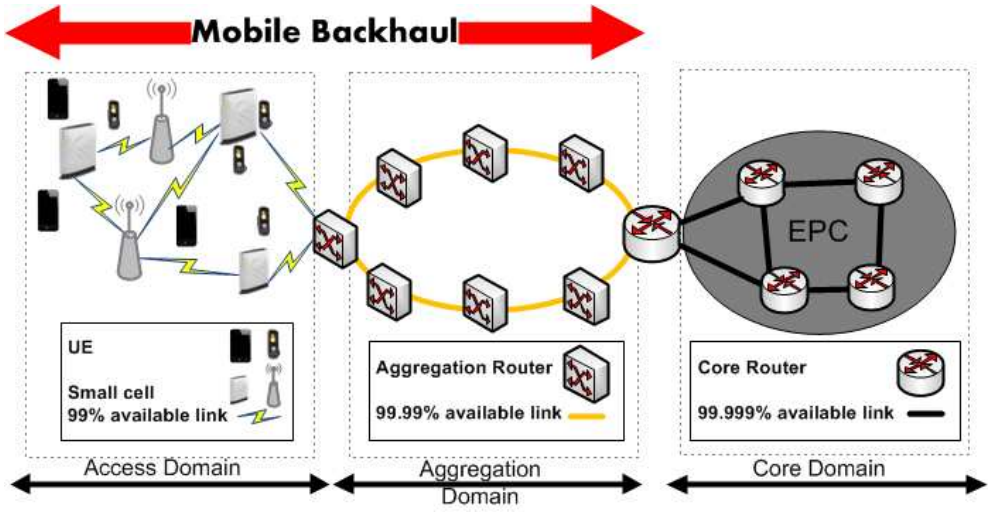


Figure 2.1: Mobile Backhaul Architecture

the sake of simplicity, we will use the term base station indistinctly to refer to eNB/macrocell and HeNB/SCs. The SC entity entails functionalities of both the Mobile Network Layer (MNL) and the Transport Network Layer (TNL). A HeNB refers to the Mobile Network Layer (MNL) component of the SC.

The EPC may contain many Mobility Management Entities (MME), Serving Gateways (S-GWs) and Packet Data Network Gateways (P-GWs) together with a Home Subscriber Server (HSS), which, located at the core of the EPC, is in charge of the storage and management of all user subscriber information. The MME is responsible for all the functions relevant to the users and the control plane session management. When an UE (User Equipment) connects to the EPC, the MME first contacts the HSS to obtain the corresponding authentication data and then represents the EPC to perform a mutual authentication with the UE. Different MMEs can also communicate with each other. A more detailed explanation of each of these entities follows:

- Packet Data Network Gateway (P-GW): The P-GW is the node that logically connects the User Equipment (UE) to the external packet data network; a UE may be connected to multiple P-GWs at the same time. Usually, each P-GW will provide the UE with one IP Address. The P-GW supports the establishment of data bearers between the S-GW and itself and between the UE and itself. This entity is responsible for providing IP connectivity to the UE (e.g., address allocation), Differentiated Services Code Point (DSCP) marking of packets, traffic filtering using traffic flow templates and rate enforcement.

- Serving Gateway (S-GW): The S-GW handles the user data plane functionality and is involved in the routing and forwarding of data packets from the EPC to the P-GW via the S5 interface. The S-

GW is connected to the eNB via an S1-U interface which provides user plane tunneling and inter-eNB handover (in coordination with the MME). The S-GW also performs mobility anchoring for intra-3GPP mobility; unlike with a P-GW, a UE is associated to only one S-GW at any point in time.

- Mobility Management Entity (MME): The MME is a control plane node in the EPC that handles mobility related signaling functionality. Specifically, the MME tracks and maintains the current location of UEs allowing the MME to easily page a mobile node. The MME is also responsible for managing UE identities and controls security both between the UE and the eNB (Access Stratum (AS) security) and between UE and MME (Non-Access Stratum (NAS) security). It is also responsible for handling mobility related signaling between UE and MME (NAS signaling).

- eNB: This functional entity is part of the E-UTRAN and terminates the radio interface from the UE (the Uu interface) on the mobile network side. It includes radio bearer control, radio admission control and scheduling and radio resource allocation for both the uplink and downlink. The eNB is also responsible for the transfer of paging messages to the UEs and header compression and encryption of the user data. eNBs are interconnected by the X2 interface and connected to the MME and the S-GW by the S1-MME and the S1-U interface, respectively.

- Home eNB (HeNB): A HeNB is a customer-premises equipment that connects a 3GPP UE over the E-UTRAN wireless air interface (Uu interface) and to an operator network using a broadband IP backhaul. Similar to the eNBs, radio resource management is the main functionality of a HeNB.

- User Equipment (UE): A UE is a device that connects to a cell of a HeNB over the E-UTRAN wireless air interface (Uu interface).

The EPS Architecture defines a wide variety of interfaces. An explanation of the more relevant interfaces regarding our work in the field follows:

- S1: The S1 interface is typically further distinguished by its user plane part (S1-U) and control plane part (S1-MME). This interface communicates the MME/S-GW/P-GW and the base stations (i.e., HeNBs/SCs and eNBs/macrocells).

- S5: This interface connects the S-GW and the P-GW in the case of non-roaming 3GPP access. To this aim, it uses the GPRS Tunneling protocol (GTP) protocol, where GPRS stands for General Packet Radio Service.

- X2: The X2 interface logically connects eNBs with eNBs and HeNBs with HeNBs. It is a point-to-point interface that supports seamless mobility, load, and interference management as defined in LTE Rel-10.
- Uu: The Uu interface connects the UE with the E-UTRAN over the air.

It is important to note that these interfaces first of all describe a communication relationship between functional entities. The details of the protocols used for this communication are described in the standards for existing interfaces and later on in this document for extended interfaces.

2.2 The Mobile Backhaul

The mobile backhaul serves as the transport medium for a mobile Radio Access Network (RAN) and connects the cells to the core network (Evolved Packet Core). As Figure 2.1 shows, at a high-level, a mobile network is composed of three domains [17], namely, the access, aggregation, and core. The access domain carries traffic generated by the base stations to an access gateway. Given the size that a dense base station deployment can have, the access domain requires of a high number of backhaul connections between base station sites and the access gateways. After the access domain, traffic is aggregated on the aggregation domain, which requires higher data rates than those of the access domain. The aggregation domain, which is composed of high capacity devices, connects the access domain with the core domain. The number of backhaul links in the aggregation domain decreases but the capacity per link increases, compared to the access domain. This domain could be a MPLS network. For the case of LTE network, the core domain connects controllers among them and with the mobile core network entities (i.e., Packet Data Network GateWay in EPC). In general, this domain is an MPLS network. The Evolved Packet Core (EPC) can be composed by the (RNC) in 3G, or Serving GateWay/Mobility Management Entity (S-GW, MME) in LTE. In the core domain, the situation is the opposite to the one exhibited by the access domain. Link and transport capacities are high, whereas the number of links is rather low.

As showed in Figure 2.1, the mobile backhaul entails two domains, namely, the access and the aggregation domain. Following the 3GPP terminology, notice that the mobile backhaul is part of the Transport Network Layer (TNL), which is in charge of carrying the mobile data traffic and the procedures defined by 3GPP, referred to as the Mobile Network Layer (MNL). Within the architecture depicted in Figure 2.1, our emphasis is in the mobile backhaul. In particular, we focus on dense SC deployments as access domain of the mobile backhaul.

2.2.1 Requirements of the Mobile Backhaul

There is a long list of requirements for transport networks acting as backhaul of cellular networks in general, and more specifically, of SCs deployments [18], such as synchronization, resiliency, availability, Quality of Service (QoS) and security. In particular, resiliency and availability, which define the service continuation characteristics of a network system, and the performance of the backhaul are tightly coupled with the research problem tackled in this thesis.

2.2.1.1 Resiliency

Resiliency refers to the ability of readily recover from network failures. It can be achieved with redundancy and proper control. Control can be in form of protection or restoration. Protected systems have a priori calculated secondary paths or routes which can be immediately activated in case of a link failure in a currently active link. Restoration in turn reacts to a link failure by finding another route after a convergence period. Thus, protection is a proactive procedure, while restoration is a reactive procedure.

2.2.1.2 Availability

Availability, understood as lack of network downtime, achieved by means of an Operation, Administration, and Management (OAM) scheme. For core transport this number is typically five nines 99.999% of availability, which allows merely a 5.26-minute downtime per year. With aggregation transport the number is usually four nines (99.99% availability) resulting in 52.56-minute downtime per year. In capacity-oriented spots where SCs are deployed in macro coverage areas, availability requirements can be relaxed from the typical to 99-99.9% (87.6 hours – 8.76 hours per year). In case there is some fault, an Automatic Protection Switching (APS) mechanism can provide fast recovery time. Availability in general is impacted by equipment failure, power outages, etc. and in wireless systems further reduced by weather conditions, temporary blocks, such as buses and trees, pole sway, and vibration.

2.2.1.3 QoS

Mobile clients should have the same experience whether accessing over SCs or over macrocells. Thus, it is crucial for the backhaul to support and provide a basic Quality of Service scheme for the RAN. The backhaul provides Quality of Service for aggregate traffic flows, requiring a handful of assignable traffic classes. The performance of a transport backhaul can refer to following aspects: throughput,

latency, jitter, packet delivery ratio, connection setup time, connection availability, connection drop rate, connection interruption times etc. The performance provided by the mobile backhaul is an important part in delivering the end-to-end service experience for mobile clients.

2.2.1.4 Other Requirements

A list of some relevant requirements for the SC backhaul that are out of the scope of this thesis follows:

- Synchronization: In order for base stations to work properly and with acceptable Quality of Service, proper synchronization is needed. Frequency synchronization is crucial in the radio interface to ensure stability of the transmitted radio frequency carrier.
- Security: As with any communication technology, security is an important aspect of mobile network and backhaul design. In the context of mobile networks, security is implemented by dividing the network into security domains. A security domain is a network portion controlled by a single operator generally with a similar security levels throughout the domain. Between different security domains there can be transit security domains, forwarding traffic between security domains. With outdoor SC backhauls, it is generally recognized that due to the easy-to-tamper locations the SC backhaul is considered to be more exposed to attacks than macro stations.

2.3 Mobile Backhaul Design Choices

There is no clear consensus on how to implement the transport network layer of the SC backhaul. New backhaul transport technologies, and topologies are emerging to ease the cost of deploying the backhaul and offer high capacity [19]. This section motivates the use of wireless mesh as backhaul for dense SC deployments. First, the section discusses over the backhaul transport technology choices (i.e., fiber vs wireless connectivity) pointing out their advantages and counterparts. Second, it describes the more appropriate backhaul topologies that suit the requirements explained in previous section. And third, it motivates the use of a mesh topology as a topology choice for the SC backhaul.

2.3.1 Mobile Backhaul Transport Technologies

2.3.1.1 Wired

Fiber is technologically the best backhaul solution for cellular networks since it can meet the required capacity, and most mobile operators have a strategic commitment to transition to fiber to backhaul traffic from cellular networks. If a fiber infrastructure is already deployed, it will be used for backhauling. This is because, in addition to offering the required capacity, they offer a high level reliability since with wired backhaul solutions there are no interference or NLOS/LOS issues. The introduction of dense deployments of SCs, however, make the transition to fiber a complex task. In the high-density areas where SCs will be deployed, fiber may be available but very expensive to bring to the nearby lampposts and other non-telecom assets where SCs will mostly be deployed.

2.3.1.2 Wireless

Wireless solutions can be either LOS (Line-of-Sight), NLOS (Non-line-of-Sight). LOS connections suggest the availability of a direct connection without obstacles between two wireless nodes, whereas NLOS admits a certain degree of path obstructions. In an urban environment, NLOS links generally can offer better adaptability to dynamic conditions but at the expenses of offering less capacity. NLOS links are feasible only with carrier frequencies under 6 GHz, due to the decrease of signal penetration capabilities. An interesting property of such links is that they can adapt to dynamic environments such as those posed by an outdoor SC deployment. With LOS links, the most common LOS frequencies are in the 6 to 38 GHz band (microwaves) and 60-80 GHz band (millimeter waves). They offer high capacity links as long as the link is not obstructed with some kind of obstacle.

Another consideration in wireless systems is the spectrum licensing. The frequency bands available between 6 GHz and around 60 GHz are largely license exempt and can offer low cost backhaul solutions, however, interference may become a problem. Access WLAN systems use 2.4, 5 and 60 GHz bands that might cause interference to backhaul systems on the same bands. Nevertheless, IEEE 802.11 is a promising candidate as backhaul technology due to its cheap availability. The transition of IEEE 802.11 towards higher frequency bands is evident through successive generations of 802.11 standards (2.4GHz, 5GHz, and 60GHz). In general, a licensed frequency band offers a more manageable and interference-free solution for the backhaul as well as more guaranteed capacity. Still, this advantage comes with the higher cost posed by a licensed frequency band.

As a final comment we can state that depending on the context and the necessities, the solution may

be composed by a heterogeneous wireless backhaul deployment with LOS as well as NLOS links using licensed and non licensed spectrum than can complement each other towards an appropriate solution.

2.3.2 Mobile Backhaul Topologies

One way to enhance resiliency is to choose a redundant topology. Given the cost of deploying multiple dedicated fiber links connecting the sites, the size a SC backhaul can attain prevents direct wireless connection between the SCs and the gateways of the access domain. This results in a need of enabling multihop communications between SCs in order to communicate with the gateways of the access domain. Current backhaul topologies deployed satisfying such properties are namely, trees, rings, and mesh topologies.

The connection of SCs via trees or rings may be an appropriate when it is sufficient that only one of the SCs is connected to the gateway and further connectivity to reach the gateway comes provided among the connection between the SCs. However, this is not usually the case. There might be cases where specific SCs cannot be directly connected to the gateway via a single wireless link because of physical obstructions, but can be reached via another SC. Under these circumstances, a redundant topology is necessary to provide alternative paths to reach the gateway. Trees are unable to offer redundancy, whereas ring topologies can provide a basic level of redundancy without any additional link. However, when there are link failures capacity in ring topologies is not preserved as in mesh topologies. Thus, mesh topologies show a higher level of availability compared to ring topologies, given that mesh topologies offer several levels of redundancy.

Another aspect of importance when choosing a wireless backhaul topology for SC deployments is the level of traffic demand fluctuations. With large and unexpected traffic demand fluctuations, the only solution in tree and ring topologies is to add higher capacity links. In turn, as the increase of traffic demands can happen anywhere in the network, all the links in the network would potentially require the addition of higher capacity links. On the other hand, meshed solutions allow traffic to be load balanced over the topology to mitigate congestion, or using alternative paths while suffering from link failures.

2.3.3 A Wireless Mesh Backhaul for Dense Deployments of Small Cells

From the wide variety of technologies showed in section 2.3.1, it is clear that an approach to reduce costs is to equip SCs with an additional wireless interface instead of laying fiber to every SC. Regarding the potential wireless backhaul topology used (see 2.3.2 for potential topology choices), note that this

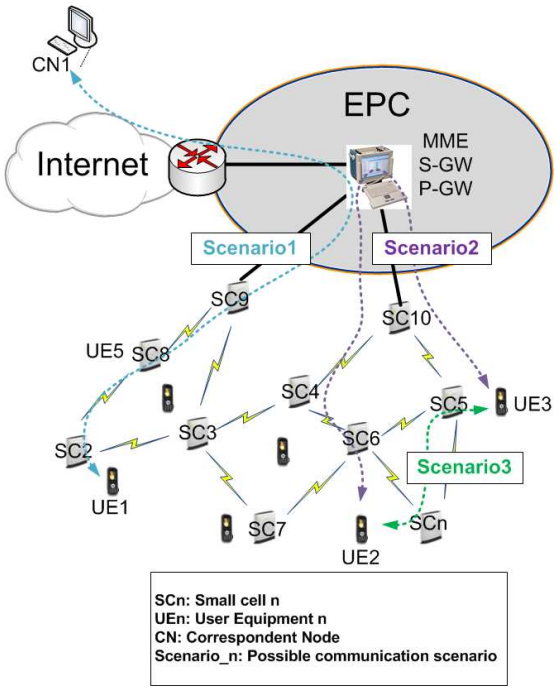


Figure 2.2: A Wireless Mesh Bachaul for SCs

wireless interface can be used to create a wireless mesh backhaul amongst SCs. A wireless mesh backhaul topology offers several advantages. In particular, such a topology can potentially satisfy the aimed requirements by NGMN, which are described in section 2.2. A wireless mesh backhaul offers inherent availability and resiliency features. Not only this, but it also offers remarkable traffic management capabilities due to its inherent multipath diversity, as noted in [19]. With a proper transport policy rolled out on top of this topology, several paths can be exploited amongst any pair of endpoints. Thus, the resulting all-wireless SC deployment yields improvements in terms of cost, coverage, ease of deployment, and capacity. They further represent a good choice under dynamic (indoor or outdoor) environments, since NLOS wireless links admit a certain dynamicity due to the environmental conditions.

Figure 2.2 illustrates the type of network scenario that results from the selection of a wireless mesh network amongst SCs as access domain for massive deployments of SCs. We also represent the different backhaul traffic patterns that can occur in such a network scenario. Scenario 1 entailing the communication pattern entailing a UE and a Correspondent Node (CN), which is a host external to the Mobile Network. Scenario 2 and 3 refer to the communication entailing two UEs attached to SCs belonging to the wireless mesh backhaul. The difference between these two scenarios follows. Whereas scenario 2 explicitly indicates the intervention of the EPC to direct traffic between both UEs (i.e., through the use of the S1 interface), in scenario 3 we consider that both UEs could establish communication without the intervention of the EPC through an interface that supports such communication pattern.

Chapter 3

Transport Schemes for the Mobile Backhaul

Routing data traffic throughout a mobile network involves two main building blocks, namely routing in the Mobile Network Layer (MNL), and in the Transport Network Layer (TNL). The former is out of the scope of this dissertation. Among other 3GPP procedures, the MNL is in charge of determining the endpoints of the GTP tunnels that carry the data for each bearer in all the relevant nodes. This may be understood as a high-level routing, which we will denote as GTP routing. Section 3.1 provides a brief description of routing in the MNL.

As for the later, the focus of this thesis, the routing problem refers to the required strategies needed to carry packets throughout the mobile backhaul GTP endpoints, which are determined by the MNL. Section 3.2 identifies the main TNL schemes that could better suit the wireless mesh backhaul topology proposed in the previous chapter. Besides, in section 3.3, we provide a discussion of the main limitations faced by the transport schemes usually deployed by Mobile Network Operators (MNO) that could exploit a wireless mesh network.

3.1 Routing in the Mobile Network Layer

The Mobile Network Layer (MNL) encompasses the Core Network (CN) and the Radio Access Network (RAN), hence including the macro and small cell (SC) sub-system. Among a wide variety of 3GPP procedures, one of the roles of the MNL is to determine the endpoints of the GTP tunnels that carry the data for each bearer in all the relevant nodes. This may be understood as a high-level routing, which we will denote as GTP routing.

In order to carry traffic to/from the core network to/from the UE, one relevant task is the establishment of S1 and S5 bearers. During bearer establishment, tunnel endpoints (i.e., HeNB and S-GW) have been determined using standard 3GPP procedures. For the S1 interface, these procedures involve the S-GW and the HeNBs. For the S5 interface the S-GW and the P-GW. Assuming that GTP is used in the S5 interface, two GTP tunnels with their corresponding endpoints are set up, namely one from the HeNB to the S-GW, and one from the S-GW to the P-GW. After applying the traffic flow template to the incoming packet, the GTP tunnel to the S-GW is determined. Once there, the S-GW performs GTP routing in order to send the incoming packet through the appropriate outgoing GTP tunnel. The other end of the tunnel is a HeNB. The S-GW functionality performs GTP routing, so that the packet reaches the HeNB.

3.2 Routing in the Transport Network Layer

As for the underlying transport network layer (TNL), as far as the mobile backhaul is concerned, it comes into play in the path between the S-GW and the HeNBs. The 3GPP EPS architecture defines a MNL that is designed to be more or less independent of a specific backhaul technology, apart from imposing certain requirements on its functionality and performance. It is also the case of the small cell backhaul. The architecture of the underlying transport layer was thus long considered out of scope of 3GPP.

The TNL is in charge of routing the packets between the endpoints determined by the MNL. The TNL determines the actual path that data packets have to follow to reach the destination endpoint. These paths traverse transport network nodes, such as routers or switches. Although the procedures handled at the TNL by these nodes are not specified by 3GPP, they are key components for an efficient operation of the mobile network. We refer to these procedures as low-level routing.

Previous chapter stressed the advantages of a wireless mesh network as mobile backhaul. One of the characteristics of mesh topologies is the availability of multiple paths to any destination. In what follows,

we provide a summary of the main features of commonly deployed TNL schemes that could suit a wireless mesh backhaul given their characteristics. Note that these schemes are not restricted to routing.

3.2.1 Transport Schemes that exploit multiple paths

The previous section motivated on the one hand the potential of the wireless backhaul technology over the wired backhaul in terms of cost; on the other hand, the use of a mesh as preferred topology due to its redundancy and potential adaption to traffic dynamicity. This section describes the currently available transport protocols that can be considered for wireless mesh backhaul scenarios.

3.2.1.1 Multiprotocol Label Switching Transport Profile

The ITU-T and IETF are currently working on a Transport Profile for MPLS (MPLS-TP) [20]. That is, the necessary and sufficient subset of MPLS functionalities that make it become a carrier-grade packet transport network plus some additional functionalities required for the transport access backhaul. Upgrading MPLS seemed the natural choice for the transition of legacy backhauls based on TDM to packet based backhauls given its efficient and connection-oriented way of handling packet services. In this direction, additional functionalities were added to MPLS-TP schemes, mainly revolving around Operation Administration, and Maintenance (OAM), and survivability (i.e., the capacity of the network to recover from failures):

- MPLS-TP OAM procedures: MPLS-TP provides mechanisms to support in-band OAM functions, such as continuity check, connectivity verification, loss measurement, delay measurement, remote defect indication and alarm indication signal. Like legacy transport networks, these mechanisms allows for fault localization, performance monitoring, remote indications and alarm suppression.

- MPLS-TP protection switching and restoration: An operator typically provisions the primary and backup Label Switch Paths (LSPs) of an MPLS-TP connection, statically. The role of OAM protocols is to monitor the status of primary and backup LSPs and detect the presence of faults. Usually, OAM protocols run at both ends of the MPLS-TP connection. Upon loss of connectivity or fault detection, both ends of the MPLS-TP connection switchover to the backup LSP (independently or coordinated by a per-hop-behavior scheduling class) and bi-directional traffic is exchanged on the backup LSP as soon as the switchover is complete. To attain this, the OAM messages need to be transmitted at a high rate (e.g., 3.3ms) to ensure failure detection within 10-15 ms and path switchover within 50 ms. Regarding the level of resources devoted to path recovery, MPLS-TP

defines several levels of protection. Dedicated protection reserves a whole transport path for the recovery. This is referred to as 1:1 protection. Shared protection maintains a backup path for several entities. This is referred to as 1:N protection.

3.2.1.2 ERPS

Ethernet Ring Protection Switching (ERPS) defined in ITU-T Recommendation G.8032 [21], provides a means of achieving carrier network requirements for ring (or collection of rings) network topologies. ERPS can be an alternative to Spanning Tree Protocol (STP) since its main focus lies on techniques for loop avoidance for rings. Although it is specifically rings, we find appropriate to include it in this classification given that somewhat similar of flavour may be appropriate for a wireless mesh network.

3.2.1.3 Shortest Path Bridging

Shortest Path Bridging (SPB) is specified as an amendment to IEEE 802.1Q [22] standard. Among other characteristics [23], one of the main properties indicating the convenience of SPB in a wireless mesh backhaul is the use of multiple equal cost paths. The use of shortest paths allows for inherent simplification and improvement in the utilization of a mesh network, because all paths may be used, and none need to be blocked for loop prevention. It is possible to get even greater utilization of resources by allowing the simultaneous use of multiple equal cost shortest paths. SPB allows for 16 relatively diverse tunable shortest paths between any pair of nodes. This is achieved by manipulation of tie breaking between multiple equal cost shortest paths. A more intelligent load balancing based on link utilization has been proposed in [24].

3.2.1.4 Transparent Interconnect of Lots of Links

TRILL(TRansparent Interconnect of Lots of Links) is a L2 routing protocol similar to SPB that operates within one IEEE 802.1-compliant Ethernet. RFC 6325 [25] specifies the base TRILL protocol. TRILL is a link-state based routing protocol in the sense that each node floods the network with the state of its links. It uses IS-IS (Intermediate System to Intermediate System) routing to distribute link state information and calculate shortest paths through the network. TRILL's main focus is (highly) meshed topologies, due to its multipath capabilities, allowing an unlimited number of equal cost paths.

3.2.1.5 Complementary Approaches

Equal Cost Multi Path: Both SPB and TRILL can use Equal Cost Multipath Routing (ECMP), a technique described in RFC 2991 [26], which combined with the most common routing protocols such as OSPF [27] or IS-IS [28] allows the splitting of a traffic flow in several different paths. Each of these paths has the same cost from the routing protocol point of view, thus, there is no higher or lower priority path when splitting the traffic between different routes. All paths are indifferently used as they cost the same. As a result of this splitting, the traffic flow is not affected by any additional delay, as there is no latency associated. Typically, a hash of IP header information modulo the number of next-hops is used to select the next hop among the possible alternatives. Because the traffic flow is spread through the network, the available bandwidth per link is higher. A fast and high level degree of protection against link failures is assured by the already existing alternative paths. On the other side, it becomes more complex for the operator to keep control of the traffic and the physical route a given packet is following.

3.2.1.6 Link Aggregation

Link aggregation is a technology that allows bonding multiple parallel links into a single virtual link. It assists Spanning Tree Protocol (STP) with parallel links being replaced by a single link, since STP detects no loops and all the physical links can be fully utilized.

- Link Aggregation Control Protocol (LACP): Link Aggregation Control Protocol is a protocol that provides redundancy at the link layer. LACP is the protocol in charge of creating Link Aggregation Groups (LAGs). LACP provides a method to bundle several physical interfaces from a network entity into one logical interface with the consequent capacity increase. The group of bundled physical interfaces forms a LAG.
- Multi-Chassis/Multi-System Link Aggregation Group (M-LAG): This link aggregation introduces the concept of bundling links of two different physical chassis rather than bundling interfaces from the same physical chassis.

Protocol	Layer	Distributed or Link-level	Standardization	Mesh Applicability	Comments
MPLS-TP	2.5	Distributed	IETF and ITU-T working jointly towards single standard	Resiliency	Connection oriented. Path reestablishment required after link failure. Various options may be used in control plane (e.g., no control plane, LDP, RSVP-TE).
ERPS	2	Distributed	ITU-T Recommendation	N/A	Control signaling flooding network to block failed link.
TRILL	2	Distributed	IETF	Resiliency	Link state, i.e., each node floods network with state of its links Need for path reestablishment in case of failures.
SPB	2	Distributed	IEEE 802.1aq	Resiliency and capacity increase through ECMP	Link state, i.e., each node floods network with state of its links. Need for path reestablishment in case of failures.
LAG/LACP	2	Link-level	IEEE 802.1ax	Link level redundancy and link level capacity increase	Not a routing or forwarding protocol, but a link aggregation protocol, i.e., it aggregates the capacity of multiple links. LACP is the control protocol used to create LAGs.
M-LAG	2	Link-level	Vendor Specific	Link level redundancy and link level capacity increase	802.1ax only provides link redundancy. M-LAG is the extension to also provide node redundancy through the aggregation of ports in multiple chassis. It needs an additional protocol inside the cluster of switches to create the single logical LAG.

Table 3.1: Transport Schemes exploiting multiple paths.

3.3 Limitations of Current Transport Schemes

The TNL routing schemes discussed above (MPLS-TP, SPB, and TRILL) represent the closest choices currently available when rolling out a wireless mesh backhaul. The preferred choice seems the use of a single MPLS domain comprising the aggregation and access networks [17]. Although the trend could be to move Multi Protocol Label Switching Traffic Profile (MPLS-TP) towards the edge to make it reach the access domain, we identify several problems that make MPLS-TP unsuitable for a dense deployment of SCs using a wireless mesh backhaul as transport network.

These TNL schemes were not designed for a dynamic and heterogeneous environment as we move towards the edge of the network due to the lower reliability of the equipment, the increasing density of wireless links, and the increase of traffic demands. This poses a series of challenges that mainly derive from the two main design assumptions of state-of-the-art TNL schemes: a high level of reliability and a low level of dynamicity.

Current TNL schemes on top of a highly dynamic and unrealiable wireless mesh backhaul can dramatically increase the frequency of recovery procedures. First, these TNL schemes follow a connection-oriented approach, which was also a distinguishing feature of circuit switched networks. In this case, dynamicity is handled by means of recovery schemes, which require sending a substantial amount of OAM and control plane traffic for maintenance and restoration of paths, which consume scarce wireless resources, hence reducing the capacity left for the data plane. In turn, this means that the path must be reestablished by Automated Protection Switching (APS) procedures before resuming regular operation, and so, recovery times are in the order of path reestablishment times. Second, state-of-the-art schemes also assume that the underlying network is mostly reliable, since they were initially conceived for wired mostly static core network environments, and hence, systems are designed to handle failures as rare exceptions. This may be true for wired networks with centralized redundant equipment, but it is less true for cheap equipment deployed in lampposts with wireless backhaul feeds. Procedures to handle such exceptions include continuity check and verification, or automated protection switching. Although such schemes are continuously being improved to make them become more lightweight and provide the required recovery times (e.g., MPLS-TP bidirectional forwarding detection), they still consume a slice of the available network resources.

As a consequence, the challenge is to design schemes that go one step beyond in handling the dynamicity and unreliability introduced by wireless mesh backhauls. In the same way, 3GPP is working on Self-Organized Networks (SON) at the mobile network layer due to the increasing dynamicity, heterogeneity, and complexity of networks. A wireless mesh backhaul acting as transport network requires

an equivalent effort for handling such dynamicity. First, the reaction times required go in the direction of putting more intelligence at the nodes, hence enabling fully distributed forwarding decisions taken locally with the least possible coordination with other nodes of the network. Second, in massive deployments, and given their lower reliability, a piece of equipment should not be treated as singular any more, but as part of a global pool of resources that can fulfill a certain forwarding functionality. In these deployments, what matters is node density, since planning is difficult. Therefore, if one node cannot do the job, the other in the pool with equivalent characteristics will. Finally, schemes should be put in place to allow packet-level reaction times (as opposed to path-level ones). In this sense, the goal is to design a new control plane able to fulfill the above challenges.

3.3.1 The Necessity of L3 Routing

Given the aforementioned context, it is clear that a L3 backhaul solution eases the management of all the dynamicity and low reliability levels posed by a semi-planned wireless mesh network acting as mobile backhaul. Despite a L2 backhaul solution can save some processing costs given the reduced complexity of the backhaul network, a L3 solution can offer a higher degree of flexibility to adapt to new traffic demands, scalability, and reliability. Given the lack of L3 TNL schemes that could fully suit a wireless mesh backhaul of the sort described above, we believe that L3 routing solutions specifically designed for wireless mesh networks, which come from the wireless data networking literature, can be a source of inspiration to design a L3 routing solution for the wireless mesh mobile backhaul.

Chapter 4

Related Work on Routing for the Wireless Mesh Backhaul

Chapter 3 described routing techniques used for Mobile Networks. This chapter explores alternative routing schemes that go beyond the cellular context, with the goal of finding novel routing techniques that can be applied to wireless mesh backhauls deployments for a dense network of small cells (SCs). Here, the reader can find an overview of routing schemes that could suit the wireless mesh backhaul scenario introduced in chapter 2. Thus, it is needed to understand if these routing protocols with its associated novel techniques can suit the requirements of a mobile backhaul (see chapter 2 for a list of mobile network operator requirements) to act as transport network layer in a context of dense deployments of small cells (SCs), or at least inspire the design of a routing protocol for such an environment.

As will be shown throughout the following sections, the body of related work of routing applicable to wireless mesh backhauls is very broad. In particular, Section 4.1 focuses on protocols that exploit the characteristics of wireless mesh networks (WMN), such as WiFi-based WMNs, which usually have data networking foundations . This family of routing protocols has been widely accepted as the philosophy to follow when designing a routing protocol for wireless mesh backhauls.

In sections 4.2 and 4.3, respectively, we lay our focus on providing an overview of two particular routing techniques that were designed for wireless networks (WMNs and ad-hoc wireless networks). These techniques include two orthogonal and desirable properties for the target routing protocol. In particular, Section 4.2 focuses on stateless routing strategies, whereas section 4.3 lays its focus on backpressure routing protocols, which theoretically promise throughput optimality [29]. The contributions of this chapter have been partially published in [3].

4.1 Routing protocols designed for the Wireless Mesh Backhaul

In general, the goal of an Mobile Network Operator (MNO) goes beyond achieving connectivity between any pair of nodes (i.e., obtaining a 100% of packet delivery ratio). This is also the case of a wireless mesh network acting as backhaul. The focus shifts from connectivity to more demanding network performance metrics, such as throughput and latency. The routing protocol plays a key role on attaining such a goal. In general, the backhaul requires to satisfy network performance metrics starting from high throughput and going towards low latency, low jitter, low energy consumption, fairness, and even providing service guarantees while assuring scalability. Thus, the reader should note that wireless routing protocols not tackling at least one of such network metrics are out of the scope of the routing protocols described in this section.

There is a vast research work in the literature applicable to routing for wireless mesh backhauls. From all this research work, we have identified those routing strategies exploiting any mesh backhaul property, and presenting a practical implementation suitable to be at least explored by a wireless mesh backhaul provider. For instance, some characteristics of wireless meshes with respect to other wireless networks is the lack of mobility of infrastructure nodes or the realization of a non-power constrained environment that can maximize the lifetime of the network. A routing protocol for wireless mesh backhauls that tackles network performance metrics such as throughput and latency is challenging, and often the research community can not go farther than conducting a theoretical study. In this taxonomy, we focus on practical routing strategies that rely on either some mathematical basis or heuristics. Therefore, for any of the schemes under consideration in this section the research study must have associated a practical distributed implementation.

Subsection 4.1.1 provides the taxonomy of these implementable routing protocols specifically designed for wireless mesh networks. In addition, the routing protocols presented in this section merely include those protocols that have been fully implemented in a distributed way in a real testbed or in a network simulator. Subsection 4.1.2 decomposes the routing architecture into building blocks. Subsection 4.1.3

summarizes and qualitatively compares the most relevant features of each the of routing protocols described in subsection 4.1.1. Subsection 4.1.4 identifies open research issues for each of the identified routing building blocks in subsection 4.1.2.

4.1.1 Taxonomy

The taxonomy is presented in Figure 4.1 and developed throughout this section. Specifically, we introduce two main routing categories and several related sub-categories. In particular, we include a set of routing protocols based on how the used routing strategy exploits the specific characteristics of the mesh backhaul. The set of mesh backhaul characteristics is, among others, the availability of multiples radios and channels, the shared medium nature that often comes along with wireless radios, and the type of antenna used in each of the radios. These characteristics determine the design of the routing strategies included in this taxonomy.

As a result of the analysis of the literature, we have classified routing protocols that exploit WMN-specific characteristics and are oriented to maximize throughput gains into two main approaches. The first one relies on the joint use of multiple radios and multiple channels. This approach may be further subdivided based on the type of antennas employed, namely directional or omnidirectional. Routing protocols using omnidirectional antennas have been further classified as coupled or decoupled, depending on how tight is the relationship and dependencies between routing and channel assignment. The second one is based on exploiting the broadcast nature of the wireless medium. In this approach, the subset of proposals referred to as opportunistic routing have been classified into two subgroups that differ in the relationship between the routing protocol, and the selection of the wireless link rate. In single-rate approaches, the routing protocol does not select the link rate, whilst it does in multi-rate approaches. Furthermore, there is another subset of proposals that also exploits the additional CPU and storage capabilities a node provides by using one of the two identified network coding strategies: intra-flow or inter-flow network coding. In what follows we will review the aforementioned strategies.

4.1.1.1 Exploiting Multiple Interfaces and Channels: Multi-Radio Multi-Channel Routing

The coordinated use of multiple radios and multiple channels per node may improve throughput in mesh backhauls. With an intelligent channel assignment scheme, radios can also work at the same frequency band, but tuned to orthogonal channels. The 802.11a, 802.11b/g, and 802.16 standards provide multiple frequency channels, which may provide an efficient use of the available spectrum when appropriately configured to orthogonal channels. As a result, throughput is expected to substantially increase, which is

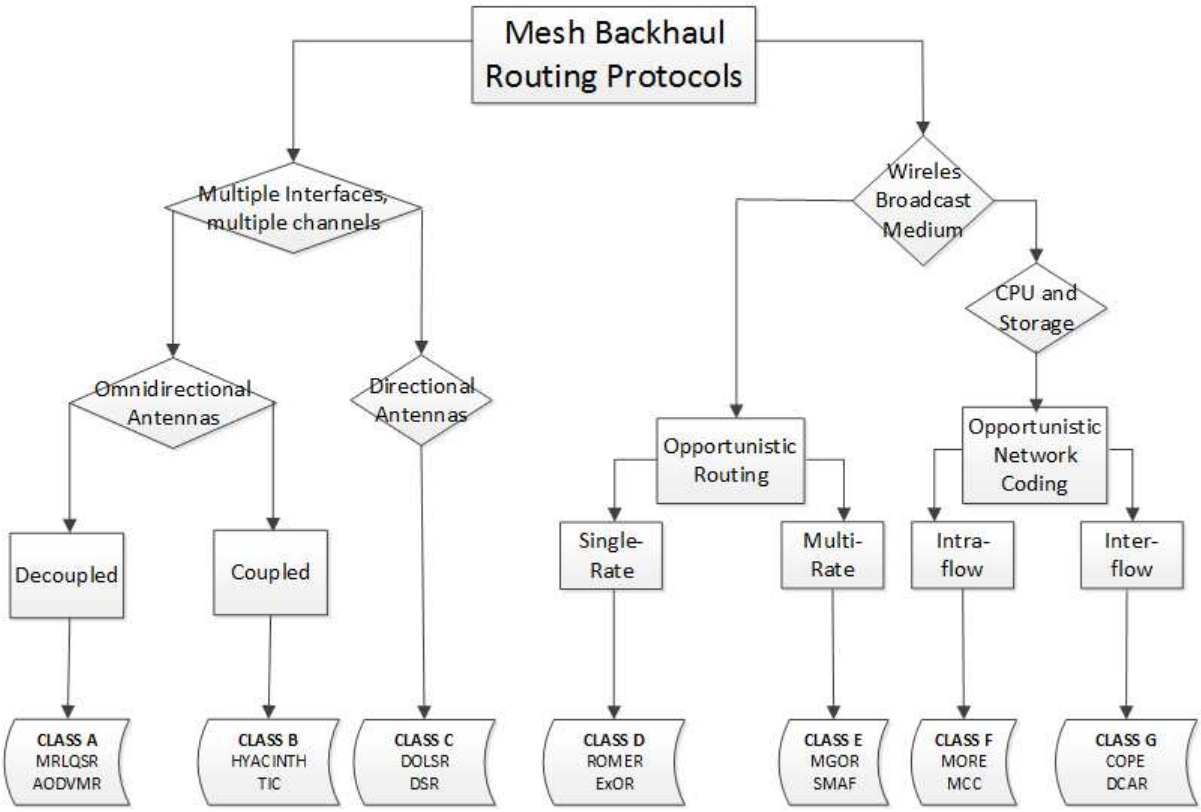


Figure 4.1: Taxonomy of Routing Protocols that exploit Wireless Mesh Backhaul properties.

mainly due to the feasibility of transmissions occurring in parallel in multi-radio nodes and minimization of interference. This is not feasible in a single radio node. Therefore, routing protocols should ideally work in cooperation with a channel assignment scheme. One of the main goals of a channel assignment strategy is the minimization of interference. On the other hand, the routing protocol determines the paths followed by data packets, and hence the traffic load distribution. In turn, the traffic load distribution determines the interference. Thus, a channel assignment strategy that cooperates with a routing protocol may provide substantial throughput gains.

A static and non-power constrained wireless mesh backhaul offers an ideal architecture for multi-radio multi-channel routing. First, the non-power constrained mesh backhaul allows adding multiple radio technologies per node. Second, when appropriately configured to orthogonal channels, the addition of radio technologies working even in the same frequency band is no longer an issue. And third, endowing with multiple wireless radios a node is economically feasible due to the availability of cheap off-the-shelf commodity hardware.

After a careful review of the literature, one may observe a conceptual difference between multi-radio multi-channel routing schemes that use omnidirectional antennas and those using directional antennas.

1 Multi-Channel Routing with Omnidirectional Antennas

This is the most common approach in the literature. First, omnidirectional antennas are cheaper. Second, due to their radiation pattern, in dense topologies, they may potentially offer increased successful reception probabilities, due to the number of potential receivers in transmission range. However, for the same reason, dealing with omnidirectional antennas can also lead to increased contention issues without proper medium access coordination. In turn, we have subdivided this group of routing proposals into those coupled and those decoupled with the channel assignment scheme. By routing coupled with channel assignment, we refer to proposals in which there is a tight relationship between routing and channel assignment. In other words, the routing protocol could determine the channel assignment strategy. On the other hand, with decoupled schemes the routing solution assumes that an independent multi-channel assignment strategy a priori precomputed is used.

- Decoupled Channel Assignment and Routing (Class A): In this case, the routing protocol is agnostic to the channel assignment strategy. In order to pursue throughput improvements, one simple channel assignment strategy independent of the routing protocol is to configure each radio in a node to different non-interfering channels. This assumption is made in Multi-Radio Link Quality Source Routing (MR-LQSR) [30] and Multi-Radio Ad-hoc On-Demand Distance Vector (MR-AODV) [31]. Multiple radios are used to increase the number of candidate paths, hence offering more paths with potentially more throughput. They are based on giving higher preference to the more channel diverse path. Giving higher preference to the more channel diverse path may result in a decrease of the contention level between data packets belonging to the same flow traversing a certain path. Moreover, in [32] and [31], addressing the interference from other flows also becomes a concern.
- Coupled Channel Assignment and Routing (Class B): Here, we refer to those routing protocols that determine the dynamics of the channel assignment strategy. In [33], Raniwala et al. present a routing protocol tightly coupled with a dynamic channel assignment scheme. At a routing level, nodes build a tree-based topology. Trees are rooted at each gateway of the mesh backhaul. All the nodes periodically compute a routing metric aimed at joining the less congested tree. The routing metrics evaluated are the hop count to the gateway, the gateway link capacity, and the gateway path capacity. As a consequence, nodes may periodically switch between multiple trees, hence providing load balancing among the gateways. On the other hand, and concerning the channel assignment strategy, each node periodically exchanges its channel load within its interference range. Channel usage is estimated by

using the number of interfering nodes and the aggregated traffic load contributed by each node. Then, a node selects the less loaded channel not currently assigned to a node in a higher hierarchy level in a routing tree. It is assumed that traffic load grows as a node is higher in the tree hierarchy. Therefore, when assigning channels, a high level node has priority over low level nodes. As the routing protocol dictates the traffic load distribution and traffic load dynamically changes the channel assignment strategy, the dynamic load-aware channel assignment scheme turns out to be driven by the routing protocol. Furthermore, the tree hierarchy imposed by the routing protocol constrains the channel assignment scheme by determining the channel assignment priorities of nodes.

Reference [34] presents a Topology and Interference aware Channel assignment architecture (TIC), which describes a hybrid channel assignment technique. The scheme works in cooperation with a routing protocol based on the Weighted Cumulative Expected Transmission Time (WCETT) [30] metric, and the Dijkstra algorithm for minimum path cost computation. It is assumed that each node has one radio tuned to the same channel in the same physical layer technology. Furthermore, all these radios have a unique and static channel assigned. The rest of the wireless radios are dynamically assigned. To assign a channel to each radio, each node builds a conflict graph in order to assign orthogonal channels. A conflict graph is used to represent interference between wireless links. Channel assignment and route quality evaluation is done in parallel by combining the use of Dijkstra with the conflict graph. For a source node, each non-interfering channel not present in the conflict graph is used to compute the minimum cost next-hop using WCETT. From all the possible channels and neighbor nodes, those which minimize the routing cost are chosen. Once the next-hop and channel are determined, the Dijkstra search advances one hop to the intended destination and computes the minimum cost and channel, and so on. Therefore, in this particular case, the channel assignment strategy is determined by the routing protocol.

Additionally, when a node changes the channel of one of its radios, data traffic transmitted through this channel switches to the common radio interface. Therefore, the common channel network is used as default network for routing flows until channel assignment in other channels ends. In this sense, the channel assignment scheme also influences routing operation.

ROMA [35] is a joint distributed channel assignment and routing scheme. The gateway is the network entity in charge of performing channel assignment based on feedback gathered from the network. The goal of such scheme is to mitigate intra-path and inter-path interference. Based on a modified version of Expected Transmission Count [36] (mETX), ROMA includes

a routing metric that takes into account link delivery ratio, channel fluctuations and external interference.

2 Multi-Channel Routing with Directional Antennas (Class C)

Directional antennas may significantly simplify the routing protocol. A wireless link composed of two directional antennas is similar in concept to a wired link. In a wired link, there is no interference and it may potentially offer long transmission ranges. A wireless link composed of directional antennas, especially when there are not many available channels, may offer spatial separation to handle contention. Therefore, interference between antennas may be minimized, or totally suppressed. Moreover, directional antennas offer increased transmission range, hence potentially decreasing the number of hops to reach the destination nodes. In turn, this decreases the number of medium access contentions. However, the widespread deployment of directional antennas poses several challenges. On the one hand, automatically-steerable directional antennas are expensive. On the other hand, if they are not steerable, the direction to which the antenna points must be changed manually, which may also be costly. But this problem is smaller in mesh backhauls, as they have a static mesh backhaul backbone. All nodes and their associated antennas are fixed; hence cheap directional antennas may be located by pointing to a known predefined neighbor. Routing and channel assignment with cheap directional antennas has been addressed in Directional Optimized Link State Routing (DOLSR) [37]. The mesh backhaul is composed of a tree rooted at the gateway. A radio with an omnidirectional antenna, configured to the same channel in every node in the network, is used to collect control information. The control information is used to decide which node could be the parent in the tree of the node. Information gathered from the omnidirectional antenna is used to select the more suitable directional parent. Thus, the directional antennas are used only for data packet transmission. They also evaluate different channel assignment strategies, decoupled from the routing protocol, which is similar to the approaches in [30–32]. Reference [38] studies the mesh backhaul routing problem when directional and omnidirectional antennas coexist. Essentially, Dynamic Source Routing (DSR) computation is used with a modified version of the Expected Transmission Count (ETX) [39] metric. When a node does not have a route to the intended destination, it floods the network by sending control packets that carry the accumulated ETX value. The ETX computation method is modified so that it not only discover neighbors in the omnidirectional range, but also in the directional range. Basically, different values of transmission power covering omnidirectional and directional ranges are tested.

4.1.1.2 Exploiting the Broadcast Medium: Opportunistic Routing

Opportunistic routing is based on the following principle: when a node wants to transmit a data packet, instead of transmitting it in unicast mode (i.e., to a single next-hop), it directly broadcasts the data packet. The broadcast transmission (common in 802.11 wireless technology) permits the senders to not necessarily know which node is the next-hop. Afterwards, the routing protocol decides on-the-fly which of the potential receivers of the broadcast packet may forward the data packet, and thus become the next-hop. The potential receivers of the packets need to work in a coordinated way in order to minimize forwarding of duplicated packets. In turn, the forwarding of data packets is also done in broadcast mode. The coordination process entails the need for specific opportunistic routing metrics and mechanisms to decide the best receivers. Therefore, in opportunistic routing, the next-hop is known after data packet transmission, which is contrary to classical unicast routing approaches. In classical unicast routing approaches, the next-hop is known before the data packet is forwarded. Wireless mesh backhauls are a suitable candidate for incorporating the opportunistic routing philosophy. This is because with a dense and static point-to-multipoint wireless mesh backhaul, the number of potential receivers of a broadcast packet increases. Thus, opportunistic routing may provide robustness in the transmission.

Specifically, we will focus on opportunistic routing protocols that effectively increase throughput measured at the destination node. The static wireless mesh backhaul offers inherent path diversity. Path diversity is provided due to the existence of a rich mesh topology that also offers point-to-multipoint links with WiFi technology. Thus, opportunistic routing protocols can exploit the point-to-multipoint transmissions a wireless mesh backhaul provides in order to maximize throughput gains. Opportunistic routing tackles a known problem in wireless mesh backhauls, namely short-term path quality variations. An issue in wireless mesh backhauls is to guarantee high throughput paths due to the high variability of the link quality. In general, classical routing protocols are not able to update link costs (and thus not able to update path costs) at the fine time scale wireless link variations occur. They usually recompute wireless link costs at the scale of various seconds. For instance, the unicast ETX metric is recomputed every 10 seconds in [39]. Path recalculation is done at an even coarser time scale.

In opportunistic routing, each packet may potentially follow a different path. Each data packet takes a single path but none of the successive packets are forced to follow this path. There are some possible candidate paths but none of them is chosen a priori. In fact, the path is chosen on-the-fly depending on the current, usually point-to-multipoint, link status. Furthermore, as transmission is done in a broadcast manner, the transmission rate is significantly improved. This is because in 802.11, for each data packet transmitted, there is the exponential backoff delay mechanism. For unicast transmissions, this

may lead to excessive delays when handling retransmissions. Moreover, even when the transmission is successfully received, the data packet needs to be acknowledged, not allowing the potential sender to transmit a new data packet until the ACK control packet is received. On the other hand, in broadcast mode, there are neither per-link layer 2 retransmissions nor acknowledgment procedures. Thus, nodes may potentially transmit at higher rates, being only limited by the physical carrier sensing. As a consequence, throughput may be increased. We classify opportunistic routing protocols into two different categories: single-rate and multi-rate. In single-rate opportunistic routing, it is assumed that the nodes are not able to manage the data rate at which packets are transmitted. These protocols assume that the wireless link rate is fixed in every node. On the other hand, in an opportunistic multi-rate environment, the routing protocol selects both the forwarding next-hop and the data rate for each radio in the node. Thus, these schemes introduce another dimension of choice in addition to the next-hop node. In turn, it is important to note that the selection of rate is highly correlated with the reliability of the packet forwarding mechanism.

1 Single-Rate Opportunistic Routing (Class D)

The main issue in these protocols is how to decide which neighbors forward data packets. The main challenge that arises is the cooperation between the potential next-hop set in order to select the best forwarding node, where best often means the node that maximizes throughput gains. Extremely Opportunistic Routing (ExOR) [40] computes at each potential forwarder the shortest path to the intended destination. The shortest path is estimated by summing up the link costs associated to the path calculated using the Dijkstra algorithm. The link costs are computed using the ETX [39] metric. Thus, each node has all the ETX values to reach the destination, and it selects the path that has the minimum sum of ETX values. For coordination purposes, a forwarder priority list is sent in each data packet to schedule the order of forwarding attempts by the next-hop set. As a result, a node only forwards a data packet if all higher priority nodes failed to do so.

To avoid the number of duplicated packets, SOAR (Simple Opportunistic Adaptive Routing) [41] restricts the selection of the candidate to the neighbors with lower ETX.

In Resilient Opportunistic Mesh Routing (ROMER) [42], the key idea is that each packet carries a credit which is initially set by the source and is reduced as the packet traverses the network. As in ExOR, each node also computes a path cost for forwarding a packet from itself to the intended destination. In ROMER, a data packet may be duplicated when traversing the wireless mesh backhaul. This may happen because potential next-hops may forward data packets, if the credit of the packet is high enough. The credit associated to each data packet is decremented at each

forwarding step according to the node credit cost, which basically means that more credits are consumed as the packet moves away from the shortest path to the destination; hence data packets are not forwarded through these paths.

2 Multi-Rate Opportunistic Routing (Class E)

Leveraging rate control to select the optimal rate in opportunistic routing may lead to throughput gains. Specifically, some broadcast links may be underutilized, hence losing throughput. A potential improvement driven by the routing protocol consists of increasing the data rate to increase throughput, and hence the optimal utilization. (Note that this is independent from MAC layer auto-rate algorithms based on unicast MAC layer procedures, not compatible with opportunistic routing, such as counting the number of retries per packet to determine the optimal rate.) On the other hand, as higher transmission rates entail shorter radio ranges, link loss rate may potentially be increased. Therefore, the network could eventually become disconnected. A solution for these links is to decrease the link rate of the node, thus, increasing the number of potential next-hops (i.e., increasing connectivity).

Nevertheless, achieving the optimal transmission rate poses several challenges. To date, there are some recent proposals addressing multi-rate. First, as different rates mean different transmission ranges, there is a trade-off between the rate selected and the number of hops. Choosing a high rate may decrease reliability, thus requiring more L3 data packet (re-)transmissions using L2 broadcast transmissions. A low bit rate may guarantee reliability, but it can also result in an unnecessary decrease of throughput. Therefore, depending on the bit rate selected, the set of potential next-hops of a node is variable.

Zeng et al. [43] propose Multi-rate Geographic Opportunistic Routing (MGOR), a heuristic for opportunistic multi-rate routing, which takes into account the constraints imposed by transmission conflicts. They argue that the problem is NP-hard, and so heuristics are used to find a solution. Specifically, two different heuristics are proposed. One is the Expected Advancement Rate (EAR), which addresses what next-hop is closer to the destination in distance by using location information. The other heuristic is the Expected Medium Time (EMT), which is based on a generalization of ETT [30].

On the other hand, Shortest Multi-rate Anypath Forwarding (SMAF) [44, 45] computes the opportunistic multi-rate path by modifying a generalized version of the Dijkstra algorithm. The main contribution of this work is to reduce the number of neighbor combinations to test for finding the optimal next-hop neighbor set at a given rate. Specifically, the optimization reduces the number of combinations to test to the number of neighbors, and not all the possible combinations

for each tested rate, which is an exponential number of combinations. This is the key to achieve a polynomial algorithm, hence reducing computational complexity. However, one of the issues from [44, 45] is that the protocol relies on accurate information about packet reception is known to every node. Recent work from [46] relaxes this assumption, while still providing opportunistic routing and rate selection.

4.1.1.3 Exploiting the Broadcast medium and CPU/storage capabilities: Opportunistic Network Coding Routing

In the reviewed routing schemes, network coding is an add-on to opportunistic routing. But it is classified as a different group in the taxonomy, due to the qualitative conceptual change that exploiting the CPU and storage of wireless wireless mesh backhauls may entail. Essentially, in opportunistic network coding routing, wireless mesh backhauls mix the content of data packets. Then, at each hop, they transmit in a broadcast manner the resulting coded packet over the point-to-multipoint wireless medium. Therefore, every coded data packet received at an intended destination contains information about different original packets. Different received data packets contain information of some original packets, thus providing, in general, useful incremental information to the receiver. Additionally, network coding may employ, if needed, original non-previously coded packets received at the destination.

Luckily, network coding poses specific requirements that may be easily fulfilled by small cell nodes forming a WMN. First, wireless mesh backhauls need to keep remarkable state information to store data packets. Second, for some network coding scenarios, each wireless mesh backhaul is recommended to be highly static in order to facilitate the buffering of data packets to be combined. Finally, wireless mesh backhauls require considerable CPU operations for mixing packets. As a consequence, wireless mesh backhauls should not be power-constrained. We have grouped network coding routing into two main groups: intra-flow network coding and inter-flow network coding.

1 Intra-flow Network Coding (Class F)

Intra-flow network coding is based on mixing packets belonging to the same data flow. This is, in fact, a specific case of single-rate opportunistic routing. When a source wants to send data packets to a destination, the source node breaks up the file into batches of packets and keeps transmitting packets in broadcast mode from the same batch until the batch is acknowledged by the destination. However, in intra-flow network coding, there is no coordination between the receivers of data packets. Prior to forwarding data packets, the forwarders store them in a buffer. When enough data packets are stored, the forwarder computes a random linear combination of the

packets. The mixed packets are headed to the same destination. The randomness in the mixing procedure assures with high probability that different nodes will not forward exactly the same packets. Thus, the number of packets received by the destination is increased. It is likely that nodes participating in the forwarding procedure send different combinations of packets. In MAC-independent Opportunistic Routing & Encoding (MORE) [47], every node sends probe packets to capture the link costs associated to its neighbors. Upon link calculation, the cost of each link is flooded to the whole network so that the Dijkstra algorithm can calculate the shortest paths. The cost associated to each link is calculated by means of the ETX [39] metric. The nodes check whether they are closer to the destination than the transmitter or not by using Dijkstra combined with ETX. If this is the case, they store the received coded packets in a buffer. When a forwarder has sufficient data packets, it makes a random linear combination of received data packets, thus generating new coded data packets, and it eventually forwards the coded packets. This process continues at each hop until enough data packets are received by the destination so that it is able to decode the original information. In order to decode the original data packets, a common constraint for the receiver is that the number of innovative coded data packets received must be greater or equal to the number of original data packets. Furthermore, to support reliability, the destination node sends and ACK (using unicast best path routing based on ETX) to the source when it has received enough coded data packets. Gkantsidis et al. present Multipath Code Casting (MCC) in [48], which also employs intra-flow network coding. In this scheme, link costs are collected and propagated by an overlay routing discovery module called Virtual Ring Routing (VRR) [49]. Moreover, a credit-based distributed algorithm is used for rate control.

2 Inter-flow Network Coding (Class G)

In inter-flow network coding, the coding operation is done over data packets belonging to different data flows. Coding Opportunistically (COPE) [40] is based on mixing packets generated by different flows, when a node detects an advantage for doing this operation. An advantage is usually detected when the number of coded packets transmitted in a single transmission may be maximized, and the destination has enough information to decode the packet. To detect an advantage, a node has to gather some information of the flows present in the network. In COPE, Dijkstra and ETX [39] are used for computing minimum cost paths. Distributed Coding-Aware Routing (DCAR) [50] goes beyond COPE and suggests combining the route discovery process with the detection of coding opportunities in order to maximize the inter-flow network coding opportunities. On-demand source computation combined with the ETT metric are used for calculating minimum cost paths. The basic idea of this scheme is to discover intersecting paths, instead of choosing disjoint paths for certain flows in the network. These flows are such that making them

coinciding in a node to code their packets is beneficial for achieving network throughput gains. Thus, end-to-end throughput of different flows is maximized.

4.1.2 Building Blocks

The functionality required by a routing protocol may be split into three building blocks, namely neighbor discovery, control message propagation, and route determination. These building blocks are common to the broad set of network protocols studied in the previous section. Essentially, every building block addresses one specific function, which is a part of the routing protocol. First, nodes have to gather link cost information about its neighbors (i.e., neighbor discovery). Second, information about link costs must be distributed throughout the network to the appropriate nodes, implying a certain propagation of route control messages over the mesh backhaul. This is handled by the control message propagation building block. Finally, once the necessary routing information is collected by all parties, the routing paths to the destination nodes are determined (i.e., route determination). Notice also that the characteristics of mesh backhauls highly influence the strategy adopted by each building block to carry out its function. Throughout this section, the different underlying strategies, and their inter-dependency with the exploited characteristics are studied for each identified building block. In brief, in neighbor discovery, the radiation pattern of the antennas; in control message propagation, the stable non-power constrained backbone; and in route determination, the forwarding approach employed and the possibility of employing multi-rate features.

4.1.2.1 Neighbor Discovery

The neighbor discovery building block groups the functionality related to the process of determining which nodes can be reached by means of direct communication (i.e., without having to cross any other intermediate node). Periodic or non-periodic broadcast packets are usually used to discover the nodes reachable by direct communication. This could be sufficient to maintain a neighbor table in each node if wireless links were as stable as wired links. However, in wireless links, the neighboring relationship is mainly determined by the quality of the link. As it is highly variable and unstable, the wireless link quality is not limited to the same two classical states as in wired networks. In classical wired networks, it is usually assumed that a link works well or does not work at all. Moreover, wireless link quality may vary depending on the direction of the link, which results in wireless link asymmetry. In practice, this means that the neighbor discovery building block is in charge not only of discovering nodes in the physical proximity, but also of estimating the link quality and stability towards each node within

transmission range. Depending on this latter estimation, a neighboring relationship will be established or not. Wireless mesh backhauls offer an environment that enables accurate wireless link estimation. A basic approach would consist of sending a control packet and then wait for an answer. However, the non-power constrained nature of mesh backhauls allows implementing more elaborate and complex procedures to increase the wireless link cost estimation accuracy. On the other hand, in mesh backhauls, the radiation pattern of the antennas equipping nodes may cause considerable changes in the issues tackled to measure link quality. Therefore, link quality measurement procedures and their associated link quality metrics, taking into account the antenna radiation pattern, become key issues of the neighbor discovery building block. Link quality metrics are explained in the first subsection. The second subsection summarizes the different procedures to carry out depending on the antenna radiation pattern.

Proposal	Antenna	Primary Metric	Measurement Technique	Link Quality Estimates
ETX [39]	Ommni/Dir	PDR	Probe Packet	Loss Rate
ETT [30]	Ommni/Dir	PDR	Packet Pair	Bandwidth
mETX [36]	Ommni/Dir	BER,PDR	Probe Packet	Loss Rate
EAR [51]	Ommni/Dir	PDR	Probe Packet Passive Cooperative	Bandwidth
Power-ETX [38]	Ommni/Dir	PDR	Active	Loss Rate
MIC [32]	Ommni	PDR	Packet Pair	Bandwidth Interference
iAWARE [31]	Ommni	SNR/SINR PDR	Packet Pair	Bandwidth Interference
ETP [52]	Ommni	PDR	Probe Packet	Bandwidth Interference

Table 4.1: Link Quality Estimators.

4.1.2.1.1 Link Quality Metrics. The cost associated to each link, which will be later used to calculate the routes in the route determination building block, requires the computation of link quality metrics in the neighbor discovery building block. Note that link metrics are conceptually different from routing metrics. Link quality metrics quantify the cost associated to a wireless link, and they are handled by the neighbor discovery process. On the other hand, routing metrics are handled by the route determination building block, as they measure the quality of paths, and not merely of single links. Thus, routing met-

rics are built by using link metrics as input to quantify the cost of an end-to-end route path. As shown in Table 4.1, for each of the proposals found in the literature, two main factors condition the design of wireless link metrics: the primary metric employed and the measurement technique utilized to calculate the parameters to estimate. Besides, the procedure may vary depending on the radiation pattern of the antenna. Finally, the link cost to be used by the route determination building block (referred to as metric in Table 4.1) is also presented.

1 Primary metric:

A primary metric is an indicator used to quantify the quality of a wireless link. There are four primary metrics [53] used in the literature,namely:

- Packet delivery ratio (PDR): The more common parameter chosen in the literature (e.g, [30–32, 34, 36, 39, 51]) quantifies wireless link reliability at a packet level. The PDR is the ratio of packets correctly received/captured to the total number of packets sent by the sender. The PDR is usually calculated in both directions of a wireless link in order to deal with link asymmetry, which is common in wireless links.
- Bit error rate (BER): This is the ratio of bits with errors to the total number of bits that have been received during a given time period. The BER primary metric defines the reliability at a bit level.
- Signal-to-interference plus noise ratio (SINR): The extent to which the power of the received signal exceeds the sum of noise plus interference at the receiver. SINR quantifies the quality of the received signal.
- Received signal strength indication (RSSI): This is the signal strength observed at the receivers' antenna during packet reception. RSSI defines the quality of the signal received.

2 Measurement technique

The most common measurement technique used to measure packet delivery ratio is based on the probe packet concept.

- Probe packet: It consists of periodically broadcasting or unicasting a packet of fixed size. The packet contains the number of probe packets received by the sender. Therefore, the receiver of the probe packet can calculate the delivery ratio of the link in the receiver-to-sender direction.
- Packet pair: Packet Pairs are a special case of probe packets. In ETT [30] , Metric of Interference and Channel-Switching (MIC) [32] , and Interference-Aware Metric (iAWARE) [31]

metrics, a node sends two unicast probe packets of different size. The receiver node measures the difference between the instants in which each packet is received, and it forwards this information to the sender. Then, it is used to estimate the available bandwidth of a link. Furthermore, based on the strategy employed to generate probing packets, the approaches followed may be categorized as active and passive.

Active: A node explicitly sends control packets to discover its neighbors. This is the default procedure in most proposals explored, either sending probe packets or packet pairs.

Passive: In Efficient and Accurate link-quality monitoR (EAR) [51], discovery can be made with the use of data packets. The real data traffic generated in the network is also used as probing packets without incurring extra overhead. In particular, passive strategies can also be cooperative. In [51], a node overhears data packets transmitted by each of its neighbors to estimate the link quality from its neighbors to itself.

3 Link Quality Estimates

The final goal of the measurement procedure is to quantify the link cost by means of one or more link quality estimates, which are obtained by appropriately combining one or more wireless link primary metrics. The metrics found in the literature reviewed follow:

- Loss Rate: Most of the proposals try to measure the loss rate, which is the percentage of packet/bit losses in the link. The loss rate is usually measured by means of probe packets, which are used to calculate the PDR primary metric. The values obtained for the PDR primary metric are used to compute the packet loss rate. For instance, these values may be averaged by means of an exponentially weighted moving average (EWMA). In [38], a modification of the ETX metric (Power-ETX) is proposed to deal with a mesh backhaul composed by nodes equipped with omnidirectional and directional antennas. It is based on alternating the transmission of broadcast probe packets at two different transmission power levels. Each of these transmission powers covers the omnidirectional and directional range. As a result, a node may discover neighbors that are beyond the omnidirectional range, which are neighbors in the directional range. On the other hand, modified ETX (mETX) [36] takes into account the average and standard deviation of the BER primary metric of the captured packets to calculate the loss rate. The standard deviation may potentially be useful in order to quantify wireless link variability.
- Bandwidth: There are proposals ([30–32, 36, 51], and [52]) focused on measuring the available bandwidth of the wireless link. Available bandwidth is usually captured through the use of packet pairs.

- Interference: Interference caused by neighbors of a node in transmission range may also be a parameter to estimate. Specifically, the interference measured is inter-flow interference. This is interference generated to each other by/to packets belonging to different flows. Essentially, to measure the inter-flow interference ([32], [31], and [52]), monitoring methods are employed to capture the number of interfering nodes at each wireless link. In general, this estimate is associated to the use of omnidirectional antennas. Measurement techniques introduced by omnidirectional antennas are based on sensing the medium and exchanging the captured information. For instance, in [32], a rough estimation is made to count the number of interfering neighbors of a node. On the other hand, [31] uses the measured SNR and SINR primary metrics to capture inter-flow interference variations.

4.1.2.1.2 Dependency of Neighbor Discovery on the radiation pattern of the antennas. The procedures followed to perform neighbor discovery vary depending on the radiation pattern of the antennas equipping nodes. According to their radiation pattern, antennas can be classified into directional and omnidirectional.

- Omnidirectional Antennas: An omnidirectional antenna has a uniform radiation pattern in all directions. The discovery of neighbors with omnidirectional antennas becomes straightforward with the use of broadcast probe packets [39]. On the other hand, interference requires special attention, as studied in [30–32, 52]. For instance, in high-density mesh backhauls, where nodes are equipped with omnidirectional antennas, contention should be carefully handled due to the potentially high number of neighboring nodes.
- Directional Antennas: Neighbor discovery with directional antennas is more challenging, since it introduces additional issues with respect to neighbor discovery with omnidirectional antennas. With directional antennas, nodes must appropriately manage the direction and beam width of the antennas in order to maintain their neighbors. On the other hand, non-steerable antennas must be manually installed. When direction and beam width of the antennas are correctly managed, the deafness problem may be minimized. Deafness occurs when a transmitter is unable to communicate with its intended receiver, because the antenna of the receiver is not pointing to the transmitter.

To discover the neighbors of a node, some approaches have been proposed:

- Probabilistic discovery: In [54], probe packets are sent in a random direction and beam width, to calculate the direction and beam width of the antenna.

- Omnidirectional neighbor discovery: As stated in [37], omnidirectional antennas could be used to handle control messages for directional antennas. Specifically, omnidirectional antennas are used to discover the neighbors of directional antennas and estimate their associated wireless link costs. Link quality metrics with directional antennas may be simplified with respect to the link quality metrics for omnidirectional antennas. Directional antennas increase spatial separation for contending transmissions compared to contention in omnidirectional antennas. Therefore, estimates that quantify interference may not be necessary, as presented in Table 4.1. An appropriate channel assignment scheme offering frequency separation may be sufficient to deal with contention in scenarios with directional antennas. In principle, proposals such as [39] may be directly used with directional antennas. However, due to their different physical layer properties link metrics should be slightly modified, as observed by [38].

Proposal	Scenario	Components	Stategy
Clustering [39]	Any-to-any	Dissemination	Efficient flooding
Fisheye [30]	Any-to-any	Dissemination	Efficient flooding
LOLS [36]	Any-to-any	Dissemination	Efficient flooding
OLSR [51]	Any-to-any	Dissemination	Efficient-flooding
Gossip [38]	Any-to-any	Dissemination	Efficient-flooding
ORRP [32]	Any-to-any	Dissemination Discovery	All-to-some
VRR [31]	Any-to-any	Dissemination Discovery	All-to-some
Hyacinth [52]	Any-to-gw	Dissemination Discovery	Tree
MaLB [52]	Any-to-gw	Dissemination	Tree

Table 4.2: Approaches to send routing control messages.

4.1.2.2 Control Message Propagation

The control message propagation building block is responsible for sending all the necessary routing control messages to the appropriate nodes. The cost incurred by the transmission of control messages is not as critical as in power-constrained ad-hoc networks. As a consequence, an appropriate control message propagation building block for mesh backhauls should aim at maximizing overall throughput by propagating accurate routing information of wireless link measurements, even at the expense of being

more costly. On the other hand, a secondary goal is to decrease the overall overhead incurred by the building block. In this sense, as mesh backhauls offer a stable backbone, no additional control messages due to the movement of nodes must be sent. In mesh backhauls, the literature distinguishes between two different traffic pattern scenarios: 1) traffic only exchanged between nodes and gateways, and 2) traffic exchanged between any pair of nodes. Throughout this section, we refer to the former as any-to-gateway and the latter as any-to-any.

Authors of [33, 37, 52] handle the any-to-gateway scenario. References [49, 55–60] handle the any-to-any scenario. Furthermore, in mesh backhauls, both traffic scenarios have in common the use of some strategy to perform the propagation of route control messages. Depending on the particular protocol, control message propagation could be carried out by one (or both) of two components: route dissemination and route discovery. Their common goal is to provide the necessary route control information to the route determination building block. And the potential coexistence of route discovery with route dissemination is facilitated by the stability of the mesh backhaul backbone. This is explained in the first subsection. On the other hand, depending on the combination between route dissemination and discovery, different techniques are presented in the literature to efficiently propagate control messages. They are discussed in the second subsection.

1. **Route Dissemination & Route Discovery.** The goal of route dissemination and route discovery approaches is the same, i.e., obtaining the necessary routing information from the network to compute the routes, but the way in which they obtain such information is different. Route dissemination refers to the process of propagating information about link state previously obtained by the neighbor discovery building block. And this information is periodically disseminated to the network in a proactive way, i.e., without any node asking for it. There are some key design decisions to make, such as the accuracy of the information to disseminate. For instance, in [55] the accuracy of the information disseminated is decreased as the distance in hops from the node disseminating the information to the recipient node increases.

   In addition, there is another method for obtaining routing information from nodes in a mesh backhaul, which is the route discovery process. It is triggered by a source node for obtaining the necessary routing information on demand. Therefore, it is done in a reactive way, that is, when the source node has data packets to send to a certain destination. In brief, this process usually works by sending control messages that asks for route information to the nodes they traverse. Once these control messages obtain the requested routing information, they are sent back to the requesting node. In mesh backhauls, the route dissemination and route discovery components may need to work in cooperation. In other words, both components may complement each other.

For instance, in large mesh backhauls, route discovery may help route dissemination to complete the propagation of required route control information, as it is not practical in this case that all nodes keep state about the rest of the nodes in the network. Therefore, the existence of route discovery may be highly dependent on the procedure followed for route dissemination and the other way around. The stability of the mesh backhaul facilitates the coexistence of both route dissemination and discovery components. For instance, one may employ a set of well known (i.e., by all nodes) static nodes to which all routing information is disseminated. As this set of nodes is not mobile, it facilitates any node requester to locate/access them. Nevertheless, in some cases, route dissemination may be sufficient to obtain the necessary routes. For instance, in small mesh backhauls, a flooding-based dissemination scheme may be appropriate. In this case, each node in the network has enough information to route packets to any destination without incurring into excessive overhead due to the small size of the mesh backhaul. Furthermore, depending on certain mesh backhaul requirements (e.g., delay), the route discovery process may be sufficient to obtain the desired routes.

2. **Techniques for Propagating Control Messages:** Route dissemination and route discovery require a massive transmission of control messages throughout the network. Therefore, it is fundamental that this is done as efficiently as possible. In this subsection, we present a brief review of representative methods for propagating routing control messages. Every routing protocol may have an associated technique for propagating useful control messages over the network. Table 4.2 presents a summary of this section. We have categorized control message propagation schemes as tree-based, efficient flooding, and all-to-some propagation. Moreover, the components (dissemination and/or discovery) used to gather routing information for each studied proposal are also presented. Finally, the traffic pattern scenario assumed by each proposal is also shown. A discussion of each of the propagation schemes follows.

   Tree-based: Several approaches in the literature are based on tree topologies ([33] and [52]). Such a tree structure is used in any-to-gateway scenarios. Essentially, the root of the tree is a gateway in the mesh backhaul. Thus, as many trees as gateways are built. These trees are usually built in an incremental way, i.e., they are expanded as nodes join the network. If there are multiple trees, a recently joined node must decide which tree to join. The construction and maintenance of a tree topology determines specific control message propagation strategies. In MAC-Aware Load Balancing (MaLB) [52], each node disseminates the accumulated routing information to its parent node. Specifically, each node propagates to its parent the cumulative routing information of all the nodes for which it is root of the subtree that includes all nodes from leaf nodes up to itself. On the other hand, in [33], the gateway disseminates its routing information to the rest of the nodes

in the tree following the tree-like structure. Moreover, there is a route discovery component that requests to the gateway, which is the root of the tree, if a new node is allowed to join the tree. Then, the gateway sends an answer to the new node.

In both above approaches, the control message propagation is influenced by the tree topology. Specifically, the propagation of control messages in tree based proposals is such that leaf nodes do not forward control messages.

Efficient flooding: Flooding is a well-known technique to propagate messages to all the nodes in the network. A node disseminates (i.e., route dissemination component) a message to all its neighbors and these neighbors, in turn, transmit to all of its neighbors, and so on, until all the nodes receive the message. However, this may incur in unnecessary duplicated transmission of packets. To avoid reception of duplicated packets, [57] proposes to use the multipoint relay scheme. This strategy is based on acquiring 2-hop neighbor information in order to select the minimum number of 1-hop neighbors that guarantee successful reception of all 2-hop neighbors. On the other hand, the scheme presented in [61], follows a gossip-based approach, in which each receiver decides with a certain probability if the control message is forwarded or not. Essentially, a source node sends a control messages with probability one. A node forwards a control message with probability p and discards the control message with probability 1-p. If a node receives a previously received control packet again, it is discarded. Although not specifically studied for mesh backhauls, clustering [60] could potentially be another strategy employed to reduce the overhead caused by route control messages. Clustering is based on partitioning the network into groups of nodes called clusters. The forwarding of control messages is limited to cluster heads and cluster gateways. A cluster head is chosen so that all nodes in the cluster receive control messages. A certain node is elected as cluster gateway to forward route control messages to other clusters. In this scheme, additional route control messages to elect the cluster head as well as the gateways must be sent to build the clusters. A specific case in efficient flooding approaches is partial flooding approaches. In [55], the flooding process only covers a certain area of the network close to the source node. Besides, the flooding may be done at different frequencies depending on the range covered. In fact, there are some strategies focused on reducing the frequency of flooded messages. One approach proposed in [55] is to define different frequencies of transmissions depending on the distance in number of hops from the node disseminating the information. Thus, the more distance, the less frequently routing information is disseminated. Additionally, the disseminated information may vary depending on the dissemination period. In Localized On-demand Link State (LOLS) [56], for short periods, each node sends route control information which quantifies average values of link costs. This route information is sent to the entire network. On the other

hand, for longer periods, nodes disseminate route control information which quantifies current link cost to nodes in the neighborhood.

All-to-some: In an all-to-some approach, all nodes maintain routing entries to some nodes. The routing information stored in each node differs. The goal is to keep routes to a sufficient number of nodes in the mesh backhaul so that it is guaranteed that any intended destination is reachable. These proposals pose several advantages. For instance, there is no flooding process involved. And opposed to flooding-based approaches, route state information stored at each node is substantially reduced. To obtain the routing information, some proposals based on sending the requests in some strategically predetermined directions have been conceived ([49, 58]). A representative example of all-to-some schemes is VRR [49]. In fact, VRR explores the idea of porting overlay routing concepts, usually used at the application layer, to sit directly above the MAC layer. An overlay is basically a routing structure that relies on an underlying network routing protocol. Specifically, VRR employs a ring-like structure for the overlay. Every node maintains paths only for its virtual neighbors, which are some predecessors and successors in the ring structure. Virtual neighbors may be far apart from each other in the physical network, hence requiring mesh paths to reach each other. Thus, a virtual hop may be composed of multiple physical hops. VRR exploits this dichotomy to assure that following the virtual paths is sufficient to reach any intended node. To build these virtual paths, a request (route discovery) is sent to find the nodes that are virtually closest to the requester in the ring structure. Furthermore, the request includes routing information about the requester (route dissemination) in order to update routing information at the requested nodes. In Orthogonal Rendezvous Routing Protocol (ORRP) [58], a request (route discovery) is sent in orthogonal directions until it finds a node with the route information requested. Furthermore, each node periodically disseminates (route dissemination) its routing information in two orthogonal directions. Thus, the number of control messages in the network is decreased compared to a flooding approach by sending control messages to only two orthogonal directions. This strategy is based on the idea that two pairs of orthogonal lines intersect in a plane.

4.1.2.3 Route Determination

Based on the routing information (e.g., link cost information) gathered by means of the control message propagation building block, the route determination building block is in charge of determining the most appropriate routing paths from a certain node to any other node (any-to-any scenarios), or from/to the gateway (any-to-gateway scenarios). Thus, the expected outcome of the route determination building block is the computation of routing tables that specify the next-hop for incoming data packets. Fur-

thermore, to compute routing tables in mesh backhauls, it is necessary to take into account the different methods incoming data packets may be forwarded. As a result, the route determination building block depends on two main components, namely forwarding approach and route computation. The particular properties of a mesh backhaul allow a node to forward data packets using different approaches (i.e., unicast and broadcast), which are explained in the first subsection. The different forwarding approaches have several implications in the route computation design. Specifically, the forwarding approach has several implications on the algorithms employed to compute the routing tables. These algorithms determine the path that has the minimum route cost metric to the intended destination. Additionally, the routing metric design depends on the utilized forwarding approach. (A routing metric is used to quantify the cost of the paths to the intended destination.)

1. **Forwarding Approach**

   In mesh backhauls, there are two methods for forwarding data traffic through the network to the next-hop. On the one side, there exists the option of deterministically unicasting the data packet from one node to one of its neighbors, which is selected by looking up the precomputed routing table. On the other side, one may broadcast from one node to all nodes in transmission range. Therefore, in a broadcast forwarding approach, various nodes may potentially be the simultaneous receivers of a data packet. The unicast approach handles the wireless link in the same way forwarding in wired networks does, i.e., as if it was a point-to-point wired link. A directional or omnidirectional antenna may potentially be used in nodes following the unicast approach. However, for environments where direction of data packets is known and unique, it may be more efficient to associate a directional antenna with the unicast forwarding approach. On the other hand, the broadcast approach, changes the classical concept of link. In a shared wireless medium, transmission matches a point-to-multipoint distribution, rather than a point-to-point one. Thus, omnidirectional antennas are specially suited to exploit such kind of links, where it may be useful to send data packet in all possible directions and/or received by multiple next-hops.

2. **Route Computation**

   A static and non-power-constrained node can perform costly route computations, which is not feasible for nodes belonging to power-constrained wireless networks. Therefore, a node may, in general, use shortest-path algorithms to calculate the most appropriate routing paths without taking into account battery or CPU-load issues. The algorithms to compute the minimum path cost to the intended destination in the schemes reviewed can be categorized as Dijkstra, Bellman-Ford, and local-based.

   Dijkstra: In link state routing approaches, the link costs of the entire network are disseminated

by using an adequate strategy to propagate control messages to the intended receiver. In this approach, the algorithm to compute the shortest path commonly used in mesh backhauls is based on modifications of the well-known Dijkstra algorithm.

Bellman-Ford: In Bellman-Ford-based routing protocols, routes are computed in a more distributed manner. In this case, a node receives information about the network after being processed by its neighbors. The distance-vector approach is used. There are various flavors of the Bellman-Ford algorithm. For instance, one of these flavors is used in on-demand source routing to update the path cost carried in the control packet at each hop [50].

Local-based: The calculation is done in a greedy manner, which selects the best next-hop closer in distance to the destination by only using local information ([42, 49, 62]). It is calculated on a hop-by-hop basis during data transmission.

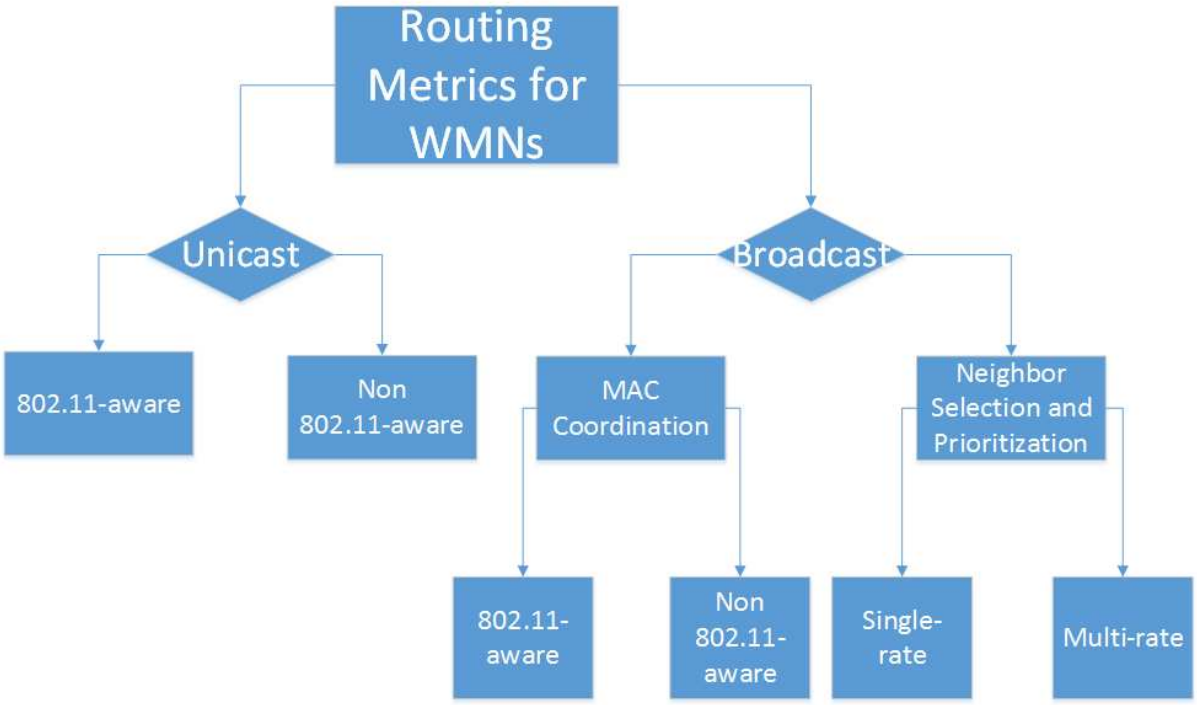


Figure 4.2: Taxonomy of Routing Metrics.

Depending on the size of the network, the Dijkstra and Bellman-Ford algorithms require costly CPU operations and considerable storage capabilities. Furthermore, the cost associated to the algorithms may change with the forwarding approach. Specifically, in a broadcast multi-rate forwarding approach, as there are various potential next-hop neighbor and rate choices, the number of operations to calculate the minimum path cost may increase [44]. On the other hand, as only local information is handled to compute the next-hop, local-based route computation tends to

consume less CPU.

The minimum path cost algorithm comes together with a routing metric. As mentioned in section 4.1.1, the path cost is quantified by means of a routing metric, as opposed to the link cost, which is quantified by a link quality metric. The computation of a routing metric takes as input parameter a set of link costs calculated during the neighbor discovery process. A subset of these link costs will form part of the resulting minimum cost path. The cost represented by the routing metric in use may be calculated by means of three different methods. First, it may only use local information to compute the cost. Second, it may be the sum of the weights of the cost of all links in the path. (Recall that each link cost was previously calculated by the neighbor discovery building block.) Finally, additional information may be required to compute a more elaborated function. There are two different philosophies to compute the routing metric depending on the strategy followed in the forwarding approach component. As illustrated in Figure 4.2, they are categorized as unicast and broadcast routing metrics.

Routing Metric	Interference
WCETT [30]	Intra-flow
iAWARE [31]	Intra/Inter-flow
MIC [32]	Intra-flow
MaLB [52]	Intra/inter-flow

Table 4.3: MAC-aware Unicast Routing Metrics.

(a) **Unicast Routing metrics:**

Although any kind of antenna may potentially be used with protocols employing unicast routing metrics, these metrics are probably more suited for directional antennas. One may group routing metrics into those not aware and those aware of the operation of the MAC protocol (see Figure 4.2). Focusing on 802.11 networks, some considerations follow for each of the groups.

Non 802.11-aware unicast routing metrics. In this case, the link metric does not directly take into account either link contention or channel usage or interference. Shortest path algorithms using as metric the sum of the weights of link metrics, like ETX [39] and ETT [30] are some representative proposals.

802.11-aware unicast routing metrics. Other metrics to calculate the optimal route are aware of the operation of the MAC layer. These routing metrics also take into account the variation of wireless link quality in the route determination building block.

Basically, the factors that are relevant to determine the quality of a path, and are captured by 802.11-aware routing metrics are the following ones:

Intra-flow interference: The interference due to packets belonging to the same flow. A common method to measure the intra-flow interference is to estimate how channel diverse are the links composing a path. As showed in Table 4.3 , intra-flow interference is captured in references ([30–32, 52].

Inter-flow interference: This type of interference is usually calculated during link quality estimation by the neighbor discovery building block. References [31] and [52], as showed in Table 4.3, capture inter-flow interference. Literature on routing metrics often presents weighted average functions of different measured components ([30–32]), such as intra-flow or inter-flow interference. The link cost metric is modified by modeling some interference level, which may be measured in different ways. A common feature in such approaches is that the calculated routing metric reflects the cost of the path from one single node to the intended destination. On the other hand, [52] evaluates how the overall network performance would be degraded if a new node joins one of the different trees. Eventually, the node joins the tree that minimizes the global delay associated to transmitting a bit for all the nodes in the forest, which is a union of trees rooted at the gateways, to its associated gateway. As a result, load balancing is explicitly provided. Therefore, the routing metric takes into account what would happen to the overall network quality when a new path is chosen.

Routing Metric	Rate	MAC coordination	Based on
EAX [56]	single	yes	ETX
RPC [63]	single	no	ETX
EAR [62]	multiple	yes	Location Information
EMT [43]	multiple	no	ETT
EATT [44]	multiple	no	ETT

Table 4.4: Broadcast Routing Metrics.

(b) **Broadcast Routing metrics:**

This operation requires at least one static broadcast wireless interface per node. Additionally, nodes are static and may be equipped with an omnidirectional antenna. Consequently, broadcast traffic distribution perfectly fits mesh backhauls. In fact, broadcast routing metrics are usually associated to opportunistic routing protocols. Though unicast routing metrics have been used with broadcast forwarding [40], these routing metrics are not totally appropriate for these environments because they do not take into account all the potential path

opportunities [63]. First, there is the issue of finding the optimal set of neighbors that guarantees maximum advancement to the intended destination, which is not present in unicast forwarding. This is similar in concept to the neighbor discovery building block. However, in this case, the discovery of the neighbors is dependent on the current intended destination. Second, another singular issue is that of prioritizing neighbors amongst those in the selected neighbor set. The optimal candidate neighbor set is the union of neighbors in transmission range for a node that maximizes progress to the destination. Additionally, there exists a trade-off between the number of neighbors available (to maximize reception probability), and the number of neighbors that truly add some progress as next-hop. Thus, broadcast routing metrics are based on finding the optimal candidate neighbor set that adds more progress towards the destination. As shown in Table 4.4, one may classify the neighbor set selection and prioritization into two groups: single-rate and multi-rate.

Single-rate neighbor set selection and prioritization. Expected Anypath Transmissions (EAX) [64] and Remaining Path Cost (RPC) [63] metrics try to select and prioritize the number of forwarding candidates from all those belonging to the neighbor set of a node. As showed in Table 4.4, these proposals are based on generalizing the well-known ETX [39] metric to account for the expected number of anypath transmissions. The expected number of anypath transmissions is the estimated number of broadcast transmissions so that the intended destination could eventually receive a data packet. In [63], all the possible neighbor node combinations at each potential next-hop in the path towards the intended destination are collected. After that, the optimal candidate forwarder is selected by using a generalization of the Bellman-Ford algorithm.

Multi-rate neighbor set selection and prioritization. As showed in Table 4.4, location information has also been used to decide which nodes of the potential neighbor set will forward the data packet [62]. And this selection is done by combining location information with an appropriate tuning of the underlying transmission rate, hence exploiting the underlying multi-rate transmission capabilities to provide a heuristic for maximizing advancement to the destination at each hop. In Expected Anypath Transmission Time (EATT) [44], wireless link quality is measured by checking the possible rates achievable. Furthermore, an algorithm to compute optimal routes based on Dijkstra is proposed. As for the metric, the ETT [39] (see Table 4.4) metric is generalized to account for the multiple rates in an anypath environment.

On the other hand, in [43], a generalization of the ETT metric is proposed as broadcast routing metric. The proposed candidate selection and prioritization may be computed us-

ing a distributed Bellman-Ford algorithm. Finally, in addition to guaranteeing that an optimal neighbor set selection choice is made, the forwarding approach used must minimize the number of duplicated packets as well as packet contention using some prioritization scheme. There are some common generic techniques to solve this issue. Some of them are based on introducing a scheduler algorithm aware of the wireless medium, while others exploit network coding, hence not needing additional coordination at all. As a result, the coordination strategy may be classified into two groups: 802.11-aware coordination and non-802.11-aware coordination (see Table 4.4).

802.11-aware coordination. References [40, 62, 64] note that sending data packets in broadcast mode requires a scheduler to avoid duplicated transmissions by the potential forwarders. Thus, a coordination strategy to avoid (or minimize) this situation must be in place. In general, such a strategy is based on using timers associated to the MAC layer [62]. And it works as follows. Every data packet carries the node forwarding priorities calculated in the sending node. And these priorities are calculated by exploiting the available location information to determine the distance to the destination. Furthermore, the MAC broadcast layer is modified to transmit an ACK packet when a data packet is received. Then, a node candidate forwarder with jth priority order may wait for the time needed for transmitting $j - 1$ ACK packets before deciding to send if it does not overhead any previous ACK. Furthermore, in [40, 64], control messages are exchanged between the forwarding nodes to schedule in order their forwarding attempts. Therefore, a node forwards a data packet only if higher priority nodes failed to do so.

Non 802.11-aware coordination. With network coding, each receiver mixes received packets before forwarding them. Random network coding assures each receiver will not forward the same packets. Packets belonging to the same ([47] and [48]) or different ([65] and [50]) flows may be combined. The main advantage is that no explicit coordination between nodes is needed because the probability that two nodes use the same linear combination is quite low. Therefore, random network coding exploits spatial diversity and increases throughput due to the absence of such an explicit coordination scheme.

4.1.3 Qualitative Comparison

This section summarizes and qualitatively compares the most relevant features of each the routing protocols considered in section 4.1.1. We highlight and summarize the most relevant design decisions

each routing protocol made for each of its building blocks (see Table 4.5). Each representative routing proposal is tagged with a letter (A, B, C, D, E, F, G) identifying the classes defined in section 4.1.1. Furthermore, for each building block, we identify the more important aspects out of those discussed in section 4.1.2.

Class	Proposal	Neighbor Discovery	Message Propagation	Route Determination		
				Forwarding	Route Computation	MAC
A	MR-LQSR [30]	ETT	Flooding	Unicast	Dijkstra	Yes
A	MR-AODV [30]	iAWARE	Flooding	Unicast	Dijkstra	Yes
B	Hyacinth [30]	Hellos	Tree	Unicast	Local	No
B	TIC [34]	ETT	Flooding	Unicast	Dijkstra	Yes
C	DOLSR [37]	ETT	Flooding	Unicast	Dijkstra	Yes
C	DSR [38]	ETT	Flooding	Unicast	Dijkstra	Yes
D	ExOR [40]	ETX	Flooding	Broadcast	Dijkstra	Yes
D	ROMER [42]	ETX	No	Broadcast	Local	No
E	MORE [47]	ETX	Flooding	Broadcast	Dijkstra	No
E	MCC [48]	ETT	All-to-some	Broadcast	Local	No
F	SMAF [44]	ETT	Flooding	Broadcast	Dijkstra	No
F	MGOR [62]	Hellos	No	Broadcast	Local	Yes
G	COPE [65]	ETX	Flooding	Broadcast	Dijkstra	No
G	DCAR [50]	ETX	Flooding	Broadcast	Bellman	Yes

Table 4.5: Comparison of the main Building Blocks for the most representative Routing protocols.

As for neighbor discovery, we focus on the link quality metric as the more relevant aspect. As shown in Table 4.5, the ETX [39] metric seems to be the most common approach used in the literature. Some other relevant proposals choose the ETT metric [30], or even, not to estimate the link quality at all and merely use HELLOs received to discover the neighbors. Another relevant link metric is iAWARE [31], which is also used by MR- AODV [30].

As for control message propagation, Table 4.5 presents the propagation technique implemented by each routing proposal. Flooding is the most common approach followed by the generic routing approaches explored. But other relevant alternatives exist. For instance, an all-to-some approach is implemented in MCC [48], a tree-based approach in Hyancinth [33], and an efficient-flooding approach in DOLSR [37]. Another interesting observation is that in ROMER [42] and MGOR [43], the control message propagation building block is not needed due to the specific operational characteristics of these protocols.

As for route determination, we take into account three main features. The first one is the interaction with the MAC layer when computing the routing metric (rightmost column in Table 4.5). Furthermore, in case such interaction is present it is based on two operational principles: 1) the MAC coordination (see the route determination building block), implemented by ExOR [40], MGOR [62], and DCAR [50], and 2) whether the routing metric takes into account the MAC layer, implemented by MR-LQSR [30], MR-AODV [31], and TIC [34]. The second feature examined for route determination is the algorithm used for minimum cost path computation, namely Dijkstra, Bellman-Ford, and local-based. As shown in Table 4.5, the most common strategy followed by routing protocols shown is the Dijkstra algorithm. This algorithm is used by MR-LQSR [30], MR-AODV [31], MORE [47], ExOR [40], SMAF [44], COPE [65], and TIC [34]. The Bellman-Ford algorithm is implemented by DSR [38] and DCAR [50]. And, the computation of the routes merely using local information is implemented by ROMER [42] and MGOR [62].

The third feature compared is the forwarding approach chosen for transmitting data packets, namely unicast or broadcast. The unicast forwarding approach is used by MR-LQSR [30], MR-AODV [31], Hyacinth [33], TIC [34], DOLSR [37], and DSR [38]. On the other hand, broadcast forwarding is implemented by MORE [47], ROMER [42], ExOR [40], SMAF [44], MGOR [62], COPE [65], and DCAR [50].

4.1.4 Open Research Issues

Here, we identify one of the main problems associated with studied routing protocols in section 4.1.1. In particular, section identifies one of the main research issues that challenges their direct use on high-scale

and unreliable wireless mesh backhauls generated by network of small cells. Moreover, section 4.1.4.2 lists specific open research issues that could be of interest to study for wireless mesh network other than the wireless mesh backhauls introduced by an all-wireless and high-scale network of small cells.

4.1.4.1 The Main Research Issue for Wireless Mesh Backhauls

In summary in section 4.1.1 we described a family of routing protocols that exploit diverse wireless mesh features. In general, all these protocols end up directing traffic through a relatively small number of usually pre-computed routing paths. In turn, the computation of these paths entails the use of a distributed shortest path algorithm that requires the consumption of a slice of the wireless resources. The consumption of resources increases with the size of the network. This poses significant problems in an unreliable and constrained wireless capacity environment. This is the main reason to study generic wireless routing techniques that minimize the consumption of wireless resources to compute the routes. These techniques are summarized in section 4.2.

But what is needed in the dense and unreliable wireless network environment posed by a wireless mesh backhaul is to make the most out of the network resources. To make the most out of the network resources, the routing protocol must distribute the load across network resources. Thus, rather than sending traffic across a small number of pre-computed routing paths, the principle of distributing load favors higher wireless mesh backhaul utilization and adaption to the network conditions, even if some of the paths are not the shortest ones. This is the main reason to study generic backpressure techniques in section 4.3.

4.1.4.2 Research Issues Specific from Studied Routing Protocols

A list of open research issues that may need further work in order to use the protocols described in section 4.1.1 into the scenario posed by a wireless mesh backhaul follows. In particular, we provide a summary of identified open research issues for each building block forming part of these protocols.

4.1.4.2.1 Neighbor Discovery A list of the identified open research issues related to the neighbor discovery building block follows:

Link quality. Vlavianos et al. [53] suggest that every single primary metric on its own may not be a good estimate of link quality. A proof of this fact in an indoor testbed may be found in [53]. These studies showed that although BER may be a good predictor of link quality, it requires a high number of

computations to make the appropriate measurements. On the other hand, RSSI cannot capture interference and SINR is quite complex to be measured. A starting point may be a deep review of the effects captured by each of the different primary metrics by studying appropriate combinations of the primary metrics in order to find an accurate link quality metric.

Active measurement strategies. Regarding active measurements, the schemes followed by recent literature are quite similar. Active measurement techniques are based on periodically sending broadcast or unicast probe packets that also use additional network resources. An effort should be made to study other measurement strategies. For instance, in general, the size of the probe packet and the inter-generation time of probe packets are fixed. A future research direction may consist of evaluating whether changes in the active measurement strategy may lead to more accurate link quality metrics.

Self-interference. Active probe packets have the disadvantage of affecting the wireless link quality they are measuring. Current wireless link metrics do not take into account the interference generated by the active measurements. A detailed study of the impact of interference caused by active measurements may be of interest.

Wireless link quality prediction. A parameter not sufficiently evaluated in current work on wireless link quality assessment is how to predict the variability of a wireless link. Keeping historical measurements or storing traffic patterns to predict the future state of a link may provide a starting point.

Link quality estimation with directional antennas. The spatial separation offered by directional antennas is able to decrease the complexity of the calculation of a link metric. However, directional antennas have their own physical layer properties. A subject of further study may be the definition of estimators specific to directional antennas rather than using those originally designed for omnidirectional antennas.

4.1.4.3 Control Message Propagation

A list of the identified open research issues related to the control message propagation building block follows:

Propagated information. Given the instability of wireless links, a challenge that arises is what exact information is going to be spread to other nodes in the mesh backhaul. For instance, one possible option is to associate a predicted lifetime to the propagated wireless link cost. This may be calculated by the neighbor discovery building block.

Intelligent dissemination. Future work should also target the minimization of the routing overhead.

For instance, one may consider an intelligent strategy that only disseminates routing control messages when relevant changes occur. A question that arises here is what is considered as a relevant change in a mesh backhaul. There is a proposal for generic networks (i.e., not only mesh backhauls) suggesting this possibility in [59].

Route dissemination vs. route discovery. There is no generic agreement in the procedure followed to propagate route control messages. Although route discovery may exist, it is not clear its relative importance with respect to route dissemination. The route dissemination component may yield lower delays but considerable overhead costs. On the other hand, the route discovery component may lead to higher delays but lower overhead costs. Probably, the trade-off between delay and overhead may depend on the mesh network requirements. As a result, the importance between route dissemination and discovery may ideally vary over time. Therefore, the introduction of mechanisms devoted to gather dynamic conditions of the network so that the relation between these components is optimal may be a subject of further study.

Paths with enough available bandwidth. In general, the studied routing protocols globally assume that there is a path with enough available bandwidth to the destination. However, when there is not enough bandwidth available to later send the data packets, path discovery should be avoided. In wireless mesh networks, it is usually assumed that there is a path between any pair of nodes. But even though there is a path, it is not guaranteed that there is sufficient available bandwidth to deliver a certain service or maintain a communication. A mechanism that can detect and relieve congestion by changing the default set of paths may be of interest.

Overlays. A promising approach for efficient control message propagation is the use of overlays for propagating routes. However, there is a primary challenge to face. In fact, porting P2P overlay routing systems to the network layer is not trivial, as there are some differences to take into account. First, pushing Distributed Hash Tables (DHTs) on top of the link layer makes connectivity between any couple of nodes become an issue. And second, the mapping of logical paths of P2P structures into physical paths does not take into account the underlying physical topology, which leads to path inefficiencies. In general, this is not a problem in wired networks, due to their higher link rates. However, this is no longer true in mesh backhauls where bandwidth is a scarce resource. Consequently, the study of strategies able to generate logical paths that are similar to physical paths may be of potential interest. For instance, this may potentially be done by setting up some rules to apply when a node joins a mesh backhaul, so that its assigned location maintains the logical structure without compromising the path stretch of the mesh backhaul.

4.1.4.4 Route Determination

A list of the identified research topics related to the route determination building block that may need further work follows:

Path Interference estimation. A known problem faced by unicast routing metrics is how to quantify the two types of interference in the form of a routing metric. For instance, MIC [32], iAWARE [31] and MaLB [52] capture intra-flow and inter-flow interference, but each proposal requires different methods for interference estimation. Therefore, there is no consensus in the research community on how to measure interference in a mesh backhaul. Specifically, it is somewhat unclear whether information from lower layers (i.e., Physical and MAC) may be necessary to obtain accurate interference estimations. Therefore, a research direction may consist of measuring interference merely using the network level without resorting to lower layers. Furthermore, an evaluation of its achieved accuracy to see the necessity of using cross-layer interactions may be required.

Integration of the routing metric with the rest of components. The routing metric designed may not work properly with any routing computation algorithm. The design of the routing metric is tightly coupled with the design of the rest of the components of a routing protocol. For instance, it is shown in [32] that the WCETT [30] metric combined with the Dijkstra algorithm does not provide isotonic, where isotonic means that a routing metric should ensure that the weighted order of two paths is preserved if they are appended or prefixed by a common third path. Thus, WCETT cannot be calculated locally for each node and then simply perform a summation to obtain the cost of the whole path. In other words, WCETT requires a single calculation with the presence of all the node components involved in the path quality calculated by WCETT, namely the ETT of each link and the channel assigned to each link. Otherwise, the calculated routing path may be non-optimal or even may generate routing loops. Therefore, the design of accurate isotonic routing metrics may be of interest.

Route recalculation timers. There is no consensus on appropriate values of the expiration timer that triggers the recalculation of the quality of a route. There is a trade-off between the optimal route choice and the stability of the route [34]. Frequent route path changes may lead to packets not received in order at the receiver. Thus, routing pathologies may occur at large scale. Some metrics may be needed to decide whether it is an advantage to recompute the routing path on a per-packet basis or keeping the same routing decisions.

Shortest Path Computation on a distributed routing protocol. The overall overhead required for a minimum path calculation using algorithms such as Dijkstra or Bellman Ford is not scalable as network size increases. In fact, with Dijkstra all nodes must be aware of the link costs of the whole network.

And this is not feasible in a large-scale mesh backhaul, even assuming that nodes may embed powerful processors. Approaches to minimize or restrict such requirements to certain areas should be investigated.

Broadcast transmission limitations. The broadcast forwarding approach introduces one major issue, namely the absence of a reliability mechanism similar to that present in unicast forwarding. This may imply shifting reliability mechanisms to the routing level, i.e., guaranteeing reliability hop-by-hop, by areas, or in an end-to-end basis.

4.2 Stateless Routing

Stateless routing refers to routing schemes able to take forwarding decisions without computing a routing table. Thus, to take forwarding decisions they merely rely on information about their 1-hop neighborhood. These strategies are characterized by its large scalability and low consumption of wireless resources due to signaling traffic. In particular, this section summarizes two generic stateless routing techniques for wireless mesh networks: geographic routing in subsection 4.2.1 and potential field-routing in 4.2.1. We focus on reviewing stateless routing strategies since we found them of primal importance in order to maximize the use of wireless resources.

4.2.1 Geographic Routing

Geographic routing [66] approaches tackle scalability by leveraging geographic positions of nodes to take routing decisions. Precisely, rather than establishing or maintaining complete routes from sources to destinations, the state stored at each node comes determined by the size of the 1-hop neighborhood. Consequently, there is also a substantial decrease with regards to the number control messages transmitted to maintain the routing state. Nodes merely exchange control messages with its 1-hop neighbors, regardless of the number of flows injected in the network, and/or the size of the network. However, position-based routing schemes require of a protocol able to map address to location. While highly scalable from the routing level point of view, geographic routing protocols assume location-to-address mapping techniques and require either node-localization equipment, such as GPS receivers, or node-localization techniques (i.e., a virtual coordinate system) to specify node positioning. Among the wide variety of geographic routing techniques, we highlight the following: **GGR (Greedy Geographic Routing)**: GGR assumes that every forwarding node has a neighbor closer to the destination, unless the forwarding node itself is the destination (in that case the packet is pulled from the network). To route to any destination, each node only relies on its position and that of both its local vicinity, and the coordinates of

the destination node carried on the packet. Thus, nodes take routing decisions based on local information, being the next hop the node that minimizes the geographic distance to the destination. In case there is no neighbor closer to the destination than the current node, greedy geographic routing would fail to deliver the packet. A node in this situation is referred to as local minimum or concave node. Therefore, GGR requires of additional routing strategies to overcome the network void.

Broadly, literature proposes a high variety of different strategies to overcome local minima. A survey on void handling techniques can be found on [67]. The main problem is that they end up breaking the initial properties of position-based routing (i.e., scalability) to end up into solutions based on flooding the network when there is a local minima. There are also proposals based on heuristics, hence not guaranteeing packet delivery. Finally, solutions making unrealistic assumptions. For instance, the network can be modeled as an unit disk graphs or unit ball graphs (UBG) for 3D networks. A brief description of GPSR [68], a widely used strategy including greedy and face routing, follows:

GPSR (Greedy Perimeter Stateless Routing): GPSR [68] belongs to the category of position-based routing and proposes two modes of operation to forward packets: greedy and recovery (or perimeter) mode. In greedy mode, each node forwards a packet to an immediate neighbor which is geographically closer to the destination node. Greedy mode is the default mode of operation until a packet reaches a local minima. A packet reaches a local minima where its distance to the destination is closer than its neighbors' distance to the destination. GPSR recovers from a local minima using recovery mode based on the right-hand rule. The rule states that when a node a first enters into the recovery mode, its next forwarding hop b is the node that is sequentially counterclockwise to the virtual edge formed by the current node a and destination D. Afterwards, the next hop is sequentially counterclockwise to the edge formed by b and its previous node a. However, if the edge formed by the current node b and the next hop crosses the virtual edge (a, D) and results in a point that is closer than the previous intersecting point a, recovery mode will perform a face change in that the next hop c is chosen sequentially counterclockwise to the edge (b, c), where the closer intersecting point was found. Note that if the graph is not planar, that is, there are cross-edges in the graph, routing loops may occur. To deal with routing loops the Relative Neighborhood Graph and Gabriel Graph are planar graphs that can be generated as long as the edges satisfy the unit disk graph assumption [68]. The packet resumes forwarding in greedy mode when it reaches a node whose distance to the destination is closer than the node at the local minima to the destination.

Solutions to handle local minima lead to an increased path stretch (i.e., the ratio between the path length and the shortest available path), and so, and increment of latency. Moreover, graph planarization algorithms require unrealistic assumptions with regards to the network. Graph planarization also implies

the removal of some links in the network, which could be unaffordable in a real mesh backhaul. On the other hand, note that solutions treated above consider voids in the network dealing with worst-case conditions, and not realistic conditions in the context of a static mesh backhaul. In favor of these techniques, we could state that voids static mesh backhaul might be present but their extension should not be huge, as in the environment posed by an mobile ad-hoc network. The explanation for this is simple: networks studied are static, dense, and we assume GPS like coordinate assignment schemes are hard to fail. Thus, wireless dynamics and node failures should be, in principle, the main reason of finding local minima or network voids.

4.2.2 Potential- or Field-based Routing

These schemes all share the following idea: the construction of a scalar field throughout the network, which assigns a value $P(n)$ (or potential) to every node n. The destinations have assigned the minimum values. Packets are always forwarded along the steepest gradient in order to reach the destination. All nodes require to keep track of the potentials of their neighbor nodes. Specifically, a packet is routed at each hop so that the node n_k forwards data packets to that of its neighbors n_{k+1} with the minimum potential, assuming this minimum is smaller than the potential of n_k. The concept of field-based routing provides a very versatile way of making routing decisions. The roots of these algorithms come from physics using adaptations of Newtons' Method of steepest gradient search algorithms. In other words, packets follow a discretized version of the path that a positive charge follows as it moves in an electrostatic field. Usually, the field component contains information of distance to the destination as well as congestion information (i.e., queue backlog).

PB-Routing

Basu et al. [69] present the Policy Based Traffic Aware (PBTA) routing algorithm that uses steepest gradient search methods to assign potentials to nodes that are a function of both cost to the destination and queue congestion. Their routing protocol iteratively converges on these potentials, which are then routed on. The authors prove that the queue sizes will remain bounded, and that looping cannot occur under the PBTA routing algorithm. The underlying traffic assumptions on which the proofs rely are rather strict, and require rapid node potential updates with respect to queue updates. Additionally, the queue sizes required for stability are still quite large, as the notion of congestion is path based, not single hop.

HEAT

The HEAT [70] routing algorithm was designed for large multi-hop wireless mesh networks. HEAT

uses anycast packets instead of unicast packets to make better use of the underlying wireless network, which uses anycast by design. HEAT relies on a temperature field to route data packets towards the Internet gateways. Every node is assigned a temperature value, and packets are routed along increasing temperature values until they reach any of the Internet gateways, which are modeled as heat sources. The protocol that establishes such temperature fields does not require flooding of control messages over the network. Rather, every node in the network determines its temperature considering only the temperature of its direct neighbors, which renders this protocol particularly scalable with the network size.

ALPHA

ALPHA [71, 72] is partially aligned with HEAT. In this case the scalar value is determined by the distance to the Gateway and a degree of congestion. ALPHA uses an analogy with physics to derive a distributed scheme. However, it requires manually setting up some of the key parameters (e.g., sensitivity to congestion) and 10 to 15 iterations affecting all nodes in the network to converge, hence making it less adaptable to realistic and varying traffic demands. Additionally, the scheme is conceived to only handle upstream traffic.

4.2.3 Open Research Issues

In a wireless backhaul, since nodes are static, there are several factors that can originate a local minimum: an unplanned deployment of nodes, a variable wireless link, a node failure, and a wrong coordinate assignment. In general, the strategy followed by geographic routing protocols is to use GGR and then switch to some alternative routing recovery method (e.g., face routing) once packets find a local minimum.

One major drawback of geographic routing protocols is the overhead incurred from switching to the routing recovery method and then eventually back to GGR, since most hybrid protocols will typically return to greedy forwarding once the local minimum has been recovered. For instance, face routing, one common routing recovery strategy to overcome dead ends, requires of graph planarization techniques needed to guarantee the avoidance of routing loops. In turn, these techniques may imply the removal of wireless links in the network. The removal of active wireless links of a wireless network is a total waste of air resources for a wireless mesh backhaul. Further, graph planarization techniques imply the activation of some resource consuming operations due to the generation of control overhead.

On the other hand, geographic routing protocols do not tackle congestion. Hence, geographic routing paths experience low performance under congestion, since routing paths are only calculated based on geographic distance. Given the lack of congestion awareness, the amount of traffic flowing through a

link can be very variable. While some links can get congested, other links could remain totally unusued. Therefore, geographic routing shall be accompanied by some dynamic load balancing strategy to deal with congestion. In practice, geographic routing approaches should only be used under light traffic conditions, since geographic routing on its own is unaware of congestion incurred when there are severe traffic demands.

We believe void avoidance is not the only problem position-based routing has to face but congestion. Hole avoidance algorithms opened an extensive research area in the context of Wireless Sensor Networks (WSNs). However in the context of the mesh backhaul, one should expect a dense deployment and a careful planning of nodes with GPS-like positioning systems. Though network voids are prone to occur they should not be big voids but small voids. This is not the case of unplanned network deployments such as the usual deployment of WSNs. Therefore, the hole avoidance techniques should be simpler than those entailing the planarization of a graph, which is a complex and non practical technique from the point of view of a mobile network operator. In conclusion, despite introducing some interesting features, geographic routing does not seem to be a complete routing strategy for the wireless backhaul. However, it provides one remarkable feature: awareness of proximity to the intended destination. This is a huge help to direct packets packets to the intended destination in a static wireless backhaul. This feature also comes at a low price (i.e., GPS at each node). However, it is prone to significantly reduce the performance of network performance metrics, whereas some techniques will be also required to circumvent small holes.

Regarding potential- of field-based routing, one main characteristic of these type of proposals is that the path followed by a packet is not defined by a routing table. Instead of this, packets follow the path with the steepest gradient towards the destination. In order to work properly, this routing strategy must guarantee that packets do not reach a local minimum on its way to the destination. Since packets belonging to the same flow do not have to follow a pre-computed and enforced path, the level of robustness increases. The problem with this type of proposals is that they are focused on the many-to-one traffic pattern, rather than the intended any-to-any traffic pattern in our scenario of interest. The major advantages of field-based routing are the robustness and simplicity. By design, a field comprises multiple routes to a destination. Thus, if the link to the neighbor with the highest field intensity breaks, an alternative can easily be determined.

4.3 Backpressure Routing

Backpressure algorithms [29] are recently receiving much attention due to their provable throughput performance guarantees, robustness to stochastic network conditions and, most importantly, their ability to achieve the desired performance without requiring any statistical knowledge of the underlying randomness in the network. In the following section we will give special focus to the work leading up to and extending the backpressure algorithm described by [29], as it forms the foundation of our work. Section 4.3.2 focus on practical adaptations and derivations of the work initiated by [29]. However, to date there has been no systems implementation of the dynamic backpressure routing component of these algorithms to route any-to-any traffic patterns. And there has been no practical implementation considering the wireless mesh backhaul posed by an all-wireless Network of Small Cells.

4.3.1 Theoretical Backpressure

Here we describe the initial work on backpressure by [29], which demonstrates the throughput optimality of the backpressure algorithm. Then, we introduce additional theoretical work derived from the initial backpressure proposal that has an impact in the solutions proposed in this thesis. Specifically, our interest lies on approaches based on the Lyapunov Drift-plus-penalty techniques.

4.3.1.1 The roots of backpressure

The intellectual roots of dynamic backpressure routing for multi-hop wireless networks lies in the seminal work by Tassiulas and Ephremides [29]. They considered a multi-channel downlink with ON/OFF channels, and proved that using the product of queue differential and link rates as weights for a centralized maximum-weight-matching algorithm allows any traffic workload capable of being served to be scheduled stably.

4.3.1.2 Lyapunov Drift plus-penalty

In [73], Neely et al. build upon the max-weight work of Tassiulas and Ephremides to support a general power control problem for multi-beam one-hop satellite links. In addition to this, Neely et al. make several novel contributions that lay the foundation for many future publications by providing joint power allocation and throughput optimality in multi-hop networks while supporting links having generalized inter-link interference and time varying channel capacity.

This generalizes the results of Tassiulas and Ephremides. Neely et al. define a concept of network capacity, different from the information-theoretic concept of capacity. They then bridge the existing gap between network capacity, throughput optimality, and network optimization. Their work applies to multi-hop wireless networks with general ergodic arrival and channel state processes, and need not know the parameters of these processes. The authors assume a rate-power curve is known for each link, possibly influenced by other transmission power decisions (e.g., an SINR model). They describe the Dynamic Routing and Power Control (DRPC) algorithm and its power allocation and routing/scheduling control decisions, which they prove are throughput optimal while obeying per-node power budgets. Finally, Neely et al. provide analytic bounds on the asymptotic time average delay experienced by packets traversing a network under the DRPC algorithm.

The power control work is subsequently extended by Neely in [74]. Here, through the introduction of a tuning parameter V, Neely is theoretically able to maintain throughput optimality while coming arbitrarily close to the optimal (minimum) time average power consumption per node. Increasing V results in the time average power consumption of node i approaching to the optimal objective parameter (in this case time average power consumption) p$*$like O(V), while the queue size bound grows like O(V) and therefore the queuing delay bound grows like O(V). This and another work by Neely et al. [75] are the first applications of Lyapunov drift for the joint purpose of utility optimization and throughput optimality. The authors call this energy efficient, throughput optimal algorithm the Energy-Efficient Control Algorithm (EECA). Also in [74], Neely introduces the concept of virtual queues within the Lyapunov drift minimization framework. Leveraging this novel concept, he is able to support time average penalty or utility constraints. Specifically, in [74] Neely notes that one might relax the power minimization objective and instead specify per-node time average power consumption constraints, then maximize network capacity subject to these time average constraints. These additional virtual queues are serviced at the constrained energy rate, while arrivals are equal to the per-timeslot power expenditures of the node. In order to maintain stability in these virtual queues, which are introduced into the Lyapunov network in [74], the virtual queues must also be strongly stable.

4.3.2 Practical Backpressure

This section categorizes practical backpressure in two main groups: practical backpressure approaches derived from the Lyapunov drift-plus-penalty algorithms, and a group practical backpressure approaches that derived from the initial work from Tassiulas.

4.3.2.1 Practical Backpressure Routing from a Lyapunov drift-plus-penalty perspective

The main contribution of [76] is the implementation and preliminary evaluation of a backpressure routing algorithm using a Lyapunov optimization approach [77]. Moeller et al. used Neely's Lyapunov drift-plus-penalty theoretical framework to implement the Backpressure Collection Protocol (BCP) [76], which is a practical backpressure routing approach for Wireless Sensor Networks (WSNs) dealing with many-to-one low-volume traffic scenarios. A many-to-one traffic scenario simplifies the management of queues at each node. In fact, since there is only a single destination there are no further readjustments with respect to the original backpressure algorithm. The original backpressure algorithm requires one queue per source-destination pair, and in this scenario there is only a single destination.

On the other hand, one of most remarkable features of BCP is the use of LIFO data queues. In particular, they empirically show that by using LIFO queues, the end-to-end delay experienced by packets decreases. However, this improvement has as byproduct that some data packets are trapped at queues in order to maintain decreasing queue length gradients towards the sink, and hence, they are never delivered to the destination. In particular, what is referred to as floating queue is implemented in each sensor node to maintain the queue gradient. Its size is incremented in one unit when there is a queue overflow, and decremented in one unit when there is a queue underflow. Additionally, a queue underflow triggers the transmission of a null packet to preserve decreasing queue length gradients towards the destination. And such null packets are forwarded until they reach the sink, where they are discarded. Furthermore, in case of queue overflow, the older packet is discarded.

4.3.2.2 Practical Backpressure derived from seminal work

Laufer et. al. [78] presented XPRESS, a cross-layer backpressure stack implementation. XPRESS uses backpressure to take scheduling and routing decisions. The resulting centralized implementation follows what is proposed in theory [29]. Despite showing the potential of backpressure scheduling, it has several problems that limits its implementation to small centralized wireless networks. First, it maintains centralized routing tables, and a queue per every flow per every node. In addition, it forces the wireless network to operate on a Time Division Multiple Access (TDMA) MAC, as it is originally proposed in theory.

Although work by Tassiulas and Ephremides [29] promises throughput optimality, there are two fundamental practical problems not tackled in this work: i) the high complexity of queue structure and ii) the high end-to-end latencies.

In the context of wireless networks and based on Tassiulas work, several modifications have been proposed to the backpressure algorithm focused on decreasing the complexity of queue structures and decreasing the attained latency: the shadow queue in [79] and per-hop queues in [80], which empirically result in lower end-to-end latency.

Other relevant work related to backpressure-based strategies for wireless mesh networks can be found in [81–84], in which backpressure is used for purposes other than routing. Information about queue lengths is used to regulate MAC scheduling in [81] and [84], congestion control in [81], load balancing in [82], and scheduling in [83].

4.3.3 Open Research Issues

Given the relevance of the open research issues with backpressure routing in this dissertation, we will a provide an extensive review in chapter 5, which describes the problem tackled in this dissertation.

Chapter 5

The Routing Problem

This chapter describes the research problem that this thesis addresses in a detailed manner. First, section 5.1 provides the generic context, stating the high-level research question to answer in this dissertation. From the generic research question, section 5.1 builds up the specific research question. In turn, section 5.1 formulates the main requirements to satisfy to positively answer the specific research question. Second, section 5.2 demonstrates that the research question is unanswered. For this purpose, the discussion in previous chapters on the behavior of the existing mechanisms and their drawbacks is taken up again jointly with other considerations that justify that the research question is unanswered by previous research. In this sense, this section clearly points out the separation between the work previously conducted by other authors summarized in chapter 4 and the research work carried out in this dissertation. In section 5.3, we discuss the resulting implications of providing a solution to positively answer the specific research question. Specifically, this section demonstrates that the research question this dissertation tackles is worthwhile from two different aspects. First, from a technical point of view, that is, detailing why the answer of this research question supposes a big step forward for the research community. Second, from an economical point of view, that is, explaining why is an unsolved problem of primal importance for the industry.

5.1 Research Question

Capacity-oriented deployments mandate for dense deployments, since reducing cell radii has traditionally been the most effective way to offer increased capacity densities. Current and future access capacities require dense SC deployments, as well as the corresponding backhaul capacity. Although it is unlikely that fiber reaches every SC (e.g., those deployed in lampposts), the creation of a wireless multi-hop backhaul amongst small cells to carry control and data plane traffic is expected to become popular. We consider that each SC with its associated transport devices is a resource of the transport network. With this in mind, and at a high-level, the research problem that this thesis tackles is: *how can a Mobile Network Operator (MNO) make the most out of the transport network resources deployed?*

Our hypothesis is that to answer this high-level research question the transport network layer (TNL) requires the inclusion of Self-Organizing Network (SON) capabilities. The TNL is in charge of carrying control and data plane traffic to/from the core network (e.g., EPC) to the SCs. According to the 3rd Generation Partnership Project (3GPP) [85] and Next Generation Mobile Networks (NGMN) [86], the degree of self-organization acquires primal importance when deploying future mobile networks [87]. The standardization status of self-organization regarding 3GPP release 12 is comprehensively summarized and discussed in [88]. However, the term self-organized merely refers to Radio Access Technologies (RATs) procedures, such as those defined for Long Term Evolution (LTE), and not precisely to the intrinsic procedures conducted at the TNL. Operators are looking at ways to minimize OPerational and CApital EXpenditures (OPEX and CAPEX), by minimizing the human intervention and optimizing the operation of the TNL. Our view is that introducing SON capabilities at the wireless multi-hop TNL could yield substantial OPEX/CAPEX improvements for the MNOs.

The concept of self-organization comprises self-configuration, self-optimization, and self-healing methods. The self-configuration method is triggered by incidental events of an intentional nature (e.g., when a new cell joins the network). Second, the operator needs to exploit wireless transport network resources (i.e., SCs) as much as possible in order to attain improved network performance. This procedure must be done without any external intervention, rather than the self-optimization procedures that the wireless TNL must have implemented. For instance, this will include a set of procedures oriented to keep the wireless transport network stable under sudden traffic changes. Third, the wireless TNL should include some self-healing methods triggered by incidental events of a non-intentional nature. For instance, an unexpected node/link failure in the wireless TNL.

For the purpose of this thesis, applying the above reasoning yields a more detailed research question than the high-level one formulated above. The specific research question follows: *Can we design and imple-*

ment a self-organized routing protocol capable of meeting wireless multi-hop backhaul requirements? Specifically, the routing protocol must cope with many requirements related to the intrinsic nature of the wireless multi-hop TNL. The set of routing requirements follows:

- Adaptability to the dynamicity of wireless backhaul deployments.
- Scalability with network parameters.
- Implementability in a real system.
- Improvement of performance against SoA routing approaches in key performance metrics.

5.1.1 Adaptability to the dynamicity of wireless backhaul deployments

Chapter 2 justified the importance of wireless multi-hop backhauls. We must clarify the specific wireless multi-hop backhauls studied in this thesis. That is the specific topologies and the level of planning considered when deploying a wireless multi-hop backhauls. From the several type of backhaul deployments we will focus on unplanned or semi-planned wireless mesh backhauls. Mesh topologies offer path redundancy and resiliency, hence decreasing the per-hop availability requirements between cell sites and/or backhaul nodes. Redundancy and resiliency are desirable properties present in wireless multi-hop backhauls. Equipment failures, and wireless link variability are some of the common drawbacks of wireless multi-hop backhauls that a redundant topology, like mesh topologies, can potentially mitigate. The semi-planned or unplanned dense deployments of small cells are prone to be highly variable leading, in some cases, to sparse deployments. On the one hand, the wireless backhaul may be subject to traffic dynamics [89]. Therefore, maintaining active all SCs when traffic conditions are light is unnecessarily resource consuming. A possibility is to power off SCs during light operation conditions (e.g., during the night), hence ending up with an appropriate percentage of nodes powered off. Despite these mechanisms can potentially suppose high energy efficiency gains, they also substantially alter the wireless backhaul topology. On the other hand, these SC deployments may suffer from node and link failures due to vandalism ambient conditions, or obstacles.

As a consequence, the TNL would require a high degree of adaptability to the dynamics posed by wireless backhaul deployments.*The key challenge, and precisely the most important of all requirements listed in 5.1, is to provide a routing protocol for the TNL able to adapt to varying wireless backhaul topologies. This implies the design of a routing protocol aiming at using all the resources of the wireless transport network efficiently.* Therefore, it is of primal importance to design mechanisms at the TNL that can leverage redundancy and resiliency to a variety of topological models. The range of path

redundancy may vary depending on the wireless backhaul deployment. For instance, a ring topology offers two paths, while a mesh offers several paths that can potentially be used to route traffic. Therefore, with such wireless backhaul topologies there are many ways of carrying traffic from the cell sites to the core network, and between cell sites. It is for this reason that cell sites require of optimization at the TNL in order to appropriately manage the different routes offered by redundant topologies.

5.1.2 Scalability with network parameters

The TNL has to provide massive scalability. In the access backhaul, dense deployments are expected to fulfill capacity requirements. The move towards capacity-oriented deployments has given a starring role to small cells, as increasing frequency re-use by decreasing cell radii has historically been the most effective way to increase capacity at the spectrum level. This may entail massive deployments of small cells, with a variable number of heterogeneous interfaces per small cell. To exploit such a benefit, the TNL running on top of these interfaces must scale with the size of the backhaul (i.e., mesh backhaul topology), the traffic volumes carried in the backhaul, the number of TNL aggregation gateways able to pull packets from the all-wireless backhaul, the number of interfaces per cell, and the heterogeneity of wireless interfaces located underneath.

5.1.3 Implementability in a real system

Usually, real equipment faces implementation constraints that make difficult the direct implementation of routing protocols derived from basic analytical research to the real world. Usually, there are assumptions (centralization, availability of technology not present in the market) that are not currently viable in the industry. This dissertation focuses on ending up with a *practical* approach to the routing problem for the TNL. Thus, practical considerations in terms of implementation are also a relevant matter in this dissertation. In fact, one of the main challenges of this thesis is to solve a very complex problem (i.e., any-to-any self-organized routing for the TNL) with a very simple solution, where simple means practical in terms of implementation. To do so, the routing protocol would require to operate in a decentralized manner, avoiding the excessive use of the wireless channel to transmit routing control messages. In this way, the minimization of the use of these wireless resources for sending control information would free more resources to be used for data traffic.

5.1.4 Improvement of performance against SoA routing approaches in key performance metrics

The TNL is expected to allow running high-throughput applications. Therefore, adding *high throughput* to the set of requirements of the intended routing protocol becomes of primal importance. As explained in chapter 2, whilst wireless mesh topologies can provide extensive connectivity and cost reduction, they may do so at the cost of losing capacity [90]. However, under this context, the goal has been shifted from merely maintaining connectivity to obtaining high-throughput. But throughput is an always-increasing demand, and so, a main requirement for wireless network operators. In light of these phenomena, one may think on high-throughput oriented wireless routing protocols in order to propose an appropriate TNL mechanism that satisfies the ever growing capacity demand. However, throughput is not the only requirement;*latency* is also a network metric of primal importance for mobile network operators. Therefore, the comparison of our solution with SoA TNL routing approaches in terms of the aforementioned key performance metrics will determine the success of the TNL routing strategy presented in this dissertation.

5.2 Validity of the question

The research problem stated in section 5.1 is a valid and non resolved research question. Specifically, it poses several problems that are currently being analyzed by the wireless community. Current routing protocols used in the TNL (summarized in chapter 2) were designed for scenarios that are quite different from what a redundant wireless mesh backhauls entails. Actually, current approaches derive from Internet core routing protocols, despite being deployed for wired and wireless backhauls as well.

5.2.1 Review of Current TNL Schemes

As summarized in chapter 2, there is currently a wide variety of TNL solutions such as: Multi Protocol Label Switch Transport Profile (MPLS-TP) [20], Equal Cost Multi Path (ECMP) [26], and Ethernet Shortest Path Bridging (ESPB) [91]. In addition to the routing component, some of these protocols are composed by many functionalities, like the management plane, in charge of monitoring the behavior of the protocol (e.g., OAM functions to detect and isolate faults among others). Note that the research problem tackled in this thesis focuses on the routing component present in these TNL solutions. The rest of the functionalities that these protocols contain are out of the scope of this dissertation.

These protocols are unable to satisfy the requirements of the wireless multi-hop backhaul. This is mainly because the design of these protocols assumes that they will be rolled out with a highly reliable infrastructure (e.g., wired) underneath. In fact, despite being rolled out as TNL solutions, these solutions derive from Internet core routing protocols, which also rely on highly stable wired links. Thus, their migration to wired backhaul infrastructures is, to some extent, straightforward. Still, this is not the case of low-cost wireless multi-hop backhaul infrastructures that are unstable, less reliable, and have scarce resources. In what follows, we assess whether current TNL routing approaches cope with each one of the requirements identified for a routing protocol for the wireless multi-hop backhaul:

5.2.1.1 Adaptability to the dynamicity of wireless backhaul deployments

As mentioned in section 5.1, adaptability is one main requirement that a routing protocol designed for the TNL of the wireless multi-hop backhaul should satisfy in order to exploit all available wireless resources. Dynamicity acquires importance to tackle the expected increase of unreliability of equipment devices when featuring low-cost wireless backhaul deployments. In this sense, some routing protocols, such as ECMP, MPLS-TP, Carrier Ethernet, and ESPB used in current backhaul deployments already include some multipath extensions. One problem is that the offered degree of flexibility in these multipath extensions limits the exploitation of all resources that a wireless multi-hop backhaul can offer. Besides, despite these protocols offer path recalculation mechanisms, they incur costs that 1) increase the latency due to the path recalculation time and 2) the introduction of a significant amount of routing control overhead due, for instance, to Operation and Maintenance (OAM) procedures.

Precisely, either the routing protocol keeps active one single-path and merely uses an alternative in case of path breakage, or other proposals like ESPB maintain a set of multiple-paths active. However, in general with these protocols, traffic follows one or a few pre-established and fixed paths, restricting the level of flexibility allowed in the network to the path granularity. These routing protocols end up restricting the set of links, and so, the set of transit nodes available for each data packet. Many alternative paths that could have been potentially used by any source of traffic are unknown. This low level degree of flexibility may be sufficient in deployments in which the operator uses high-capacity wired technologies, such as fiber, and equipment is also highly reliable. However, this is not the case of low-cost wireless multi-hop backhauls.

5.2.1.2 Key performance metrics in wireless backhaul deployments

The lack of dynamicity from current TNL routing approaches introduces one important limitation that causes the degradation of network performance metrics. There is a design assumption in current TNL routing protocols that causes such a network performance degradation. In fact, current TNL routing protocols do have the flawed notion of forcing traffic between two communication endpoints to follow a fixed set of routes. Typically, the use of the fixed set of shortest paths leads to minimize the end-to-end latency, while the throughput is limited to that of the bottleneck wireless link in this set. Despite this could be true under light traffic conditions, the use of previous fixed shortest-path-based routing approaches can lead to congestion, while the backhaul may have available resources in the network (i.e., spare SCs that can increase the number of routing paths) sufficient to efficiently serve the injected backhaul traffic. One of the most important limitations that current TNL approaches could experience is suffering from network congestion at a point in which there are sufficient network resources available to serve the workload.

Congestion leads to an increase of data buffers that consequently affects the performance of network performance metrics, such as throughput and latency. Buffering induces low throughput, high latency, and packet latency variation. Furthermore, the buffering problem worsens with the buffer size. For instance, once a buffer fills, congestion notification messages from TCP could lose their intended goal, since they are unable to arrive in time to their intended destination. This recognized problem of high buffers has been recently termed as bufferbloat [92] in the generic context of the Internet. Buffering is common over variable and scarce wireless channels carrying fatty traffic volumes. Retransmissions, and also the intrinsic capacity limitations of wireless links, increase the importance of exploiting all the over-the-air resources that a wireless multi-hop backhaul can offer.

5.2.1.3 Scalability

These approaches were not designed for the massive deployments envisioned for small cells. This is because these TNL solutions have an implicit burden of routing control overhead required to discover routing paths with long holding times. The required routing control overhead needs to grow with the size of the network, while maintaining fixed wireless link capacities. As a consequence, the capacity left to carry data traffic decreases with the size of the SC deployment. In addition to this, the burden of routing control overhead could also increase when reducing the holding time of discovered paths. Therefore, despite being continually improved, the fixed path-based philosophy does not adapt well to the relatively frequent link failures and sudden link rate changes of wireless links equipping SCs, since

they take routing decisions based on the previous construction of a path between two given end points.

Furthermore, these solutions increment their forwarding state with the number of ongoing traffic communication endpoints, and the number of TNL interfaces per SC. The number of forwarding entries should be fixed in order to scale with the size of the network.

5.2.1.4 Implementable and deployable in a real system

Despite these protocols have been implemented in wired backhaul deployments, the comparison with previous requirements indicate several difficulties to take advantage of such an implementation. In summary, current protocols deployed in the TNL 1) exhibit a lack of dynamicity that the wireless backhaul may require, which reflects in key performance metrics degradation, and 2) require a high amount of routing control overhead to pre-compute routes prior to starting sending packets, which limits their scalability. Thus, the cost of implementing these strategies comes at the expenses of facing all the aforementioned problems.

5.2.2 Review of Wireless Data Networking Routing Protocols

We review the routing literature within the context of the wireless data networking world taking into account the requirements listed in section 5.1: scalability, dynamicity, implementability, and optimization of key performance metrics. Indeed, we found a family of practical routing protocols that really intends to address capacity issues by exploiting wireless characteristics [3] making the most out of the available resources. These protocols were practical in terms of implementation, yet they present scalability, high overhead, throughput, delay, or packet delay variation issues as discussed in chapter 4. In fact, although the research on the literature gave us hints to solve the challenge of designing a self-organized routing algorithm for the wireless backhaul, no one of the proposals from chapter 4 is likely to solve the routing problem for the wireless mesh backhaul.

According to the specific review of the literature done in chapter 4, there is no practical wireless routing protocol able to maintain a network stable under such strict requirements (low latency, low complexity, scalability). We found, nevertheless, some interesting ideas that we used to design the routing protocol proposed in this thesis. In particular, there are two ideas found in the literature that were of primal importance for the design of the routing protocol: backpressure and geographic routing. In what follows, we contrast both approaches against the routing requirements to satisfy for the wireless multi-hop backhaul.

5.2.2.1 Backpressure Routing

The theoretical literature inevitably mentions the class of backpressure algorithms [29] when one discusses high-throughput routing algorithms. In theory, backpressure offers network stability, meaning that it is able to serve all the traffic within the network capacity region (i.e., throughput optimality). However, the seminal backpressure work makes several assumptions that do not hold for practical wireless multi-hop backhauls. One of the goals would be to relax these set of assumptions so that a practical self-organized routing protocol can be rolled out in the TNL of the mobile backhaul.

Within the context of stochastic network optimization, many techniques such as Lyapunov function or the descent lemma have been used to study the throughput optimality problem (see [93] for a survey). Out of the many proposed approaches, the class of algorithms built upon quadratic Lyapunov functions [77] (called Quadratic Lyapunov function based Algorithms (QLA)), also known as max-weight or backpressure, is receiving much attention to attain throughput optimality.

Actually, there have been some attempts to bridge the gap between this theoretical research work and the practical research field. In particular, within the field of wireless sensor networks, we found some important practical ideas. The authors of [76] proposed a **dynamic backpressure routing protocol** that takes decisions on a **per-packet basis** trying to optimize a network objective parameter while maintaining the network stable. The proposal was far from what a TNL routing solution needs, since it was designed to cover many-to-one unidirectional traffic patterns and low traffic volumes. On the other hand, a self-organized routing protocol for the TNL requires to support any-to-any traffic patterns and high traffic volumes. These limitations of [76] hampers its direct application to wireless multi-hop backhauls. The routing process at the TNL has been restricted to bidirectional communications between the cell sites and the core network because almost all the necessary intelligence was hosted in the core network. However, as mobile networks are upgraded, the range of communications patterns tends to be more flexible. Specifically, the X2 interface specified in 3GPP TS 36.423 [94] allows smarter base stations to interact with neighboring base stations instead of requiring to communicate via the core network using the S1 interface. Note that the X2 interface is merely a logical interface. Wireless multi-hop backhauls such as mesh topologies offer physical interfaces on top of which the logical X2 interfaces can be built.

A practical self-organized routing protocol for the wireless multi-hop backhaul represents an important research effort. Given the potential and novelty of the practical backpressure work presented by [76], we examined its theoretical roots. In particular, we found two relevant sources. The first is the work from Neely in [77], which details the theoretical framework based on the Lyapunov-drift-plus-penalty. The second is the seminal theoretical work of backpressure by Tassiulas [29]. Tassiulas et al. posited a

generic backpressure routing strategy able to handle any-to-any traffic patterns while being throughput optimal. However, these schemes present **practical implementation** constraints that make them unable to cope with the requirements presented in section 5.1. The design of a practical backpressure solution entails solving a certain number of important aspects of the original algorithm that makes feasible its development for real wireless multi-hop networks. Specifically, the set of assumptions posited by Tassiulas et al. [29] and Neely [77] that complicates their direct application to a self-organized routing protocol for wireless multi-hop backhauls follows:

High Queue complexity: Backpressure algorithms maintain per-flow or per-destination queues. The number of queues per node has to be bounded to scale with the number of traffic flows (i.e., source rates), and the size of the network. Some research efforts has to be done in this direction in order to limit the number of queues.

Infinite Queue Sizes: The original backpressure algorithm use techniques for analyzing queuing systems that require infinite queue sizes. Infinite queue sizes are not physically realizable in wireless multi-hop backhauls.

Flow scheduling: Since the original backpressure algorithm maintains a queue per each flow, there is a scheduling algorithm between the different queues to coordinate which flow can serve packets. There are certain implications derived from restricting the number of queues in a node, such as the flow scheduling algorithm.

Latency: One challenge for backpressure routing algorithms is to attain an appropriate trade-off between network performance metrics. In this sense, one identified problem is how to improve other network performance metrics that go beyond throughput optimality, which is demonstrated by theoretical backpressure, like latency. Note that potential improvements of latency could affect to a certain extent the demonstrated throughput optimality.

High Routing complexity: The original backpressure algorithm requires centralized information and computation to take routing/scheduling decisions on a centralized way. It relies on a central entity that is aware of all the networks state details. This framework is unrealizable in wireless multi-hop backhauls. First, to gather network state information, SCs would require a direct reliable link to this central entity. Second, the required computational complexity could be unaffordable in a wireless multi-hop backhaul. Thus, on the one side, routing control messages do not have to incur into high control overhead since wireless resources are scarce, and distributed routing decisions require low computation complexity.

5.2.2.2 Geographic Routing

An interesting proposal identified was the use of geo-location using some GPS, or virtual coordinate system [66]. Geographic routing enables any-to-any communication in wireless multi-hop networks without incurring either into high routing control overhead or computing routing tables. Therefore, no state space whatsoever is required by the nodes in the routing framework, aside from per-neighbor geographic locations. Though solving the scalability problem for any-to-any traffic patterns, geographic routing was far from optimizing target network performance metrics, such as network throughput and latency. On the other hand, note that SC deployments could highly decrease its level of planning. One key challenge of geographic routing techniques is the handling of network holes, in which a node may find itself not the destination of a packet, but also discover that no neighbors are geographically closer to the destination. Usually, to circumvent network connectivity holes, these routing protocols resort to alternative routing methods that incur into high routing control overhead, and in an increase of state kept per node.

5.3 Is it a worthwhile question?

This section answers whether the research question stated in section 5.3.1 is worthwhile from two different points of view. Subsection 5.3.1 provides the impact that answering this question supposes on the research community. Subsection 5.3.2 provides the main industrial applications.

5.3.1 Technical Impact for the Research Community

At a research level, the answer to the proposed research question in section 5.1 has two major technical impacts for the research community.

5.3.1.1 Impact of making the most out of the network resources

The first one comes from designing a self organized routing protocol satisfying all the requirements provided in section 5.1: adaptability, scalability, optimization of key performance metrics, and implementability. This is precisely, what enables the most significant value with respect to previous routing approaches: *A routing protocol that makes the most out of the network resources, represents a big impact for the research community. No matter the wireless backhaul topology and its reliability, given a set of source input rates and a set of network resources, such a routing protocol would make the best out*

of them in order to serve these input rates. This routing philosophy would change the notion of maintaining a fixed set of routes to conduct traffic management that has been widely adopted so far. More specifically, there would be a huge impact resulting from a routing protocol that offers what follows:

- High exploitation of available resources offered by the network for carrying data traffic. This is because the routing protocol would minimize the amount of routing control overhead sent to the network and uses all the resources, if they are needed.
- The set of resources offered by the network are used as they are needed.
- Even resource consumption in the network of all the wireless resources.
- High adaptability to the dynamics foreseen for available network resources. Although the wireless multi-hop backhaul could suffer from node or wireless link failures, the routing protocol would offer a fast convergence time in order to maintain an even network resource consumption.

In turn, a routing protocol with these characteristics would decrease the complexity of related research topics in the context of wireless multi-hop backhauls. For instance, a related research topic when considering wireless multi-hop backhauls is that of network planning. Network planning is one complex task wireless researchers are currently facing. A routing protocol that makes the most out of the network resources with the aforementioned properties would ease research on network planning. In fact, the success of capacity density deployments relies on self organized routing protocols in charge of finding optimized routes among distant nodes through intermediate ones without the help of fixed or centralized infrastructures.

5.3.1.2 Impact of the research methodology used

One remarkable aspect is the research methodology proposed in this thesis. This thesis aims to build a practical routing solution starting from a theoretical perspective. In particular, the starting point is the theoretical backpressure framework provided by [29, 77]. One concept defined in this work is that of the network capacity region (i.e., the set source rates that can be supported by the network). In practice, it is well known that in an unreliable wireless environment, the network capacity region is usually time-varying. Therefore, without being too conservative, it is impossible to guarantee that the source rates are always within the network capacity region. Most recent theoretical work ensures that the network is stable for any source rate within the capacity region, which is therefore assumed to be known. Though this throughput-optimality criterion is useful, it does not say anything about other network performance

metrics, such as latency. Therefore, we have to evaluate which is the cost (if any) of a high-throughput routing towards other network performance metrics such as latency. In any case, the previous criterion of throughput optimality theoretically demonstrated in [29, 77] is quite far from the current findings obtained by experimental wireless researchers.

Real deployments of wireless multi-hop networks already exist in academia. In general, these deployments use a routing protocol with low dynamicity on top of unlicensed standard technology (i.e., IEEE 802.11) that was not initially intended for mesh communications. Such deployments face surprising outcomes related to the wireless technology, such as "it does not make sense to handle more than three hops with unlicensed WiFi technology" [90]. Thus, practical researchers do not even discuss about the concept of network capacity region, since they are already experiencing significant performance issues with a small size wireless backhaul. Thus, it is not straightforward to map findings from theoretical to experimental work. In fact, there is a huge gap from the theoretical work in [29] claiming that there is throughput optimality as long as the injected traffic lies within the network capacity region, and the three-hop boundary from the experimental wireless researchers.

In fact, research on routing lacks the necessary transition from theoretical to experimental work. This thesis starts from the theoretical backpressure framework and adds the needed assumptions to bring theoretical research into the experimental level to cope with requirements listed in section 5.1. Such routing protocol strives to follow the theoretical framework from [29, 77], while identifying the main practical limitations in a real setup. These practical limitations are precisely what causes performance gaps between theoretical and experimental results. *It is clear that bridging this gap or understanding its main reasons would be of great interest for the research community.* The routing protocol proposes practical solutions to tackle these practical limitations. To do so, there is a clear need to understand how the throughput evolves under a dynamic wireless multi-hop environment: traffic demands, wireless link states, network topology, either within or even outside the capacity region. Besides, note that real deployments do not merely tackle throughput. Latency is a network performance metric as important as throughput. Furthermore, which are the factors causing this throughput gap between theory and experimental work are other issues to understand.

5.3.2 Economical Impact: Applications for the Industry

The research work in this dissertation has a wide range of applications. First, we detail from an economic perspective the benefits the outcomes of this dissertation entail for wireless vendors and operators. Precisely, the outcomes from the work carried out in this dissertation led to a contract agreement with

AVIAT Networks [1], leading vendor of wireless backhaul equipment. Having said that, it is also worth mentioning that the applications resulting from this investigation go beyond economical gains including also social gains. In particular, we detail applications that can also exploit the research done within this dissertation for the context of Public Protection and Disaster Relief (PPDR).

5.3.2.1 Economical Applications

The research question studied in this dissertation is part of a problem operators are currently facing with emerging 4G/5G networks. In fact, since the rise of 3G/3.5G networks and the rise of diverse bandwidth hungry mobile applications, operators gained interest in technologies for handling the ever-increasing data demands. However, new RAN technologies (i.e., LTE, LTE-A) may be insufficient to tackle the ever increasing demand of bandwidth. With such new RAN, technologies the bottleneck problem may be shifted to the backhaul. In fact, the challenge comprises achieving high-throughput in a very wide range of wireless backhaul deployments; from the deployment of a small size cell network up to the point of rolling out small cells in every street corner, and/or lamppost while keeping connected all the small cells wirelessly, among them and to the core network. As a result, it is clear that these deployments require a high degree of scalability.

While deploying large numbers of cell sites in a mesh fashion near the consumers helps solving the capacity problem at the RAN level, the challenge for the wireless backhaul is not just about increasing the capacity of over-the-air interfaces. Operators need to deploy low-cost routing schemes to provide connectivity with the core network and among cell sites with sufficient capacity and QoE level exploiting the path diversity offered by mesh topologies. Precisely, the increasing mobile data traffic volume and bandwidth required per user, jointly with the decreasing Average Revenue per User (ARPU), demand for highly-efficient, simple, easy-to-deploy network-level routing operations for intelligent transport of data packets. As a consequence, the design of an appropriate routing mechanism is crucial to solve the mobile backhaul challenge, whose OPEX is in general above 50% of the total OPEX of an operator. The use of low-cost equipment and minimization of cost of installation and planning also brings, in general, benefits in terms of CAPEX.

Additionally, mobile network operators are facing more resource-constrained setups; for instance, those being deployed in developing countries and/or rural areas. Despite both the RAN and backhaul technology will differ compared to their most advanced top-capacity counterparts, a self-organized routing solution has to be agnostic from these heterogeneity. As a consequence of this, the TNL routing algorithm has to adapt and function properly in a wide range of wireless backhaul deployments. There is

a wide range of choices with regards to the backhaul technology, and also RAN technologies. In fact, currently, operators claim there is not an unique solution for all the use cases. The goal is to attain all these requirements with a TNL able to connect these heterogeneous small cells in order to deliver an appropriate Quality of Experience (QoE) without leading to excessive CAPEX/OPEX. The main idea behind a routing solution able to satisfy the aforementioned conditions is to exploit all available backhaul resources, rather than continuously over-provisioning the wireless link capacity of their networks without really needing it.

The aforementioned context comes from one project that partially funded the work carried out in this dissertation. Precisely, the roots of the research efforts presented in this thesis are partially funded by the European project *Broadband Evolved FEMTO Networks* (BEFEMTO) [95], an IP project funded by the ICT-FP7 program. The BeFEMTO project aimed to develop evolved femtocell or small cell technologies based on LTE/LTE-A that enable a cost-efficient provisioning of ubiquitous broadband services and support novel usage scenarios, like networked femtocells (or small cells).

The research outcomes from this thesis (developed within the context of BeFEMTO) attracted the interest of wireless backhaul vendors. Wireless vendors need to position themselves in the market with innovative products to attract their customers: the operators. This is the case of *AVIAT Networks* [1] interested in incorporating in their equipment a self-organized network-level algorithm integrated in their heterogeneous equipment. The self-organized routing algorithm has to adapt to varying and heterogeneous topologies. It also has to cover from an access mesh backhaul for emerging rural scenarios using low-cost unlicensed technology (i.e., WiFi) to more dense backhauls with more expensive licensed wireless technology (i.e., LTE, microwave). In addition, self-organized routing has to cover low scale deployments in licensed microwave bands, such as those formed by the ring aggregation backhaul composed by aggregation nodes. One of AVIAT Networks' main challenge, as a wireless backhaul vendor, is to decrease backhaul deployment costs while still meeting capacity demands. Reducing the cost of deployment/management of a small cell becomes crucial to handle the envisioned small cell deployments. Otherwise, given the potential size these deployments may have, AVIAT's position is that there will be no market for that unless the cost per cell site (CAPEX and OPEX) is seriously decreased. One of the ways to reduce the OPEX is to increase the degree of self-organization with intelligent procedures in cell sites, hence minimizing the human intervention of a network administrator. The human intervention burden increases as the number of cell sites increase. Hence, self-organization is a key objective for AVIAT Networks to achieve wireless backhaul scalability.

5.3.2.2 Economic Applications: Public Protection and Disaster Relief (PPDR)

Public Protection and Disaster Relief (PPDR) situations can arise both in developing and developed countries. Indeed, different external factors could disrupt the smooth operation of the network. In the case of natural disasters, such as earthquakes or tsunamis, the wired infrastructure could be physically damaged thus preventing traditional communications from taking place. The absence of connectivity difficult the coordination of the rescue teams, which leads to dramatic consequences in emergency situations, during which every minute counts to save human lives. In such cases, a wireless mesh backhaul can provide a solution to these situations, as they allow the rapid deployment of an operational network without relying on any pre-existing infrastructure. Nevertheless, the success of this technology relies on the efficiency of the routing protocol satisfying the aforementioned requirements (in terms of throughput, latency, adaptability, implementability, and scalability).

Part II

Solution Approach and Simulation Results

Chapter 6

An all-wireless mesh backhaul for Small Cells

The contributions of this chapter have been published in [11], [9], [12], and submitted to [10]. In previous chapters, we stressed the advantages of deploying a wireless mesh network as backhaul to carry traffic from/to the HeNBs to the S-GW belonging to the core network. This chapter provides the 3GPP architectural implications of creating a wireless mesh backhaul among small cells. In particular, section 6.1 presents a novel deployment concept referred to as large-scale Network of Small Cells (NoS), studied in the context of the European project BeFEMTO [95]. NoSs are conceived as a complementary solution to existing macro cell deployments in order to improve network coverage and capacity, offload traffic from the EPC, and provide new services to mobile subscribers. Section 6.2 describes the main 3GPP architectural modifications proposed to provide a wireless mesh network as backhaul for the NoS. Section 6.3 describes the resulting steps in the life of a data packet to/from the S-GW from/to the HeNB, taking into account the proposed 3GPP architectural changes.

6.1 Overview of the Network of Small Cells architecture

An all-wireless NoS can be defined as a group of small cells that are able to form a partially autonomous network under the administration of a Local Network Operator (LNO), which could even be different from the MNO. Note that a small cell (SC) is a different entity than a HeNB. The HeNB, specified by the 3GPP, refers to the MNL of the SC. Thus, the SC includes a HeNB entity and additional entities such as TNL entities. In a NoS, small cells feature self-organizing (SON) capabilities, hence collaborating with each other to optimize the global operation of the cellular network. Examples of application scenarios are large-scale outdoor urban deployments, shopping malls, corporate environments, convention centers, or university campuses.

They perform functions like radio resource and mobility management cooperatively and mainly by means of local communication, i.e., minimizing the involvement of the mobile core network (EPC, in 3GPP terminology). As opposed to the traditional concept of standalone small cells, where each small cell acts in an uncoordinated fashion, NoSs aim at optimizing global network performance by allowing cooperation between small cells in a self-organizing (SON) fashion. In a NoS, some of the functions implemented by network elements in the EPC are now delegated to 'proxy' entities in the LSGW, namely the Proxy Mobility Management Entity (P-MME) and the Proxy Serving Gateway (P-SGW). This is done to reduce the volume of data and signalling traffic reaching the EPC. From the architectural point of view, one of the main challenges of a NoS is to feature a network architecture that allows LNO-deployed functional entities to run MNO-controlled procedures whilst complying with the standard interfaces and procedures defined in the 3GPP Technical Specifications.

The NoS architecture has been designed in the framework of the BeFEMTO System architecture [96]. At a high level, it is a superset of the functionalities of the 3GPP Release 10 Evolved Packet System as far as the mobile network layer is concerned and a superset of the NGN TISPAN transport architecture [97] as far as the transport network layer is concerned. The BeFEMTO Evolved Packet System Architecture encompasses the mobile network layer, which includes the core network and the radio access network, as well as the small cell subsystem. On the other hand, the BeFEMTO Transport Network Architecture describes the communication networks that transport the data between the elements of the BeFEMTO EPS Architecture, e.g. the local area network connecting a NoS or the fixed broadband backhaul. In this case, no modifications are required in the operator domain entities, but those of the customer side (in which the NoS is deployed) are extended with additional routing and mobility management functionalities. Figure 6.1 presents the most important entities of the BeFEMTO EPS that are relevant in the NoS scenario. A detailed presentation of all these building block can be found in [98]. As shown, there are

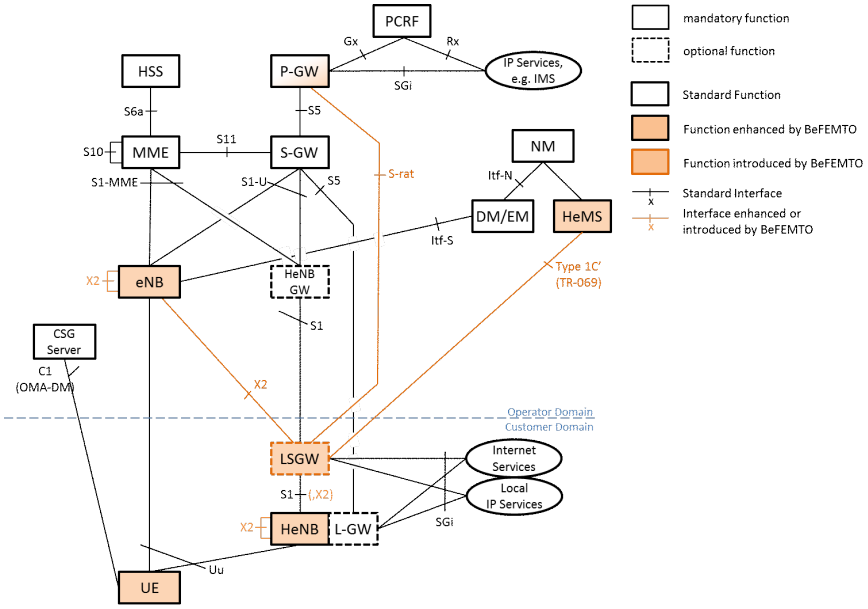


Figure 6.1: The NoS architecture.

no modifications to the core network of the MNO except for the HeNB Management System (HeMS) and the Packet Data Network Gateway (PDN-GW). The former needs enhancements for supporting the new LSGW element, while the latter is enhanced for handling the S-rat (Remote Access Tunnel) interface, which is conceived for offering Local and Remote IP access consistently in terms of session and mobility management.

The introduction of local Mobility Management Entity (MME) and Serving Gateway (S-GW) functional entities in the NoS create the need for additional network interfaces located between HeNBs and LSGWs in both control and user planes. In terms of protocol signaling, HeNBs in the NoS communicate with P-MMEs via S1-MME interfaces. Similarly, communication on the user plane between HeNBs and P-SGWs is provided over standard S1-U interfaces. The LSGW appears to the functional entities in the EPC as a standard eNB. This effectively means that all existing 3GPP interfaces originated towards/from a HeNB (S1-U, S1-MME, and X2 [94]) remain intact between the LSGW and the network elements in the EPC. However, two new network interfaces (S-rat, Type 1C) have been introduced, as shown in Figure 6.1. On one hand, the S-rat interface has been introduced between the LSGW and the PDN-GW to allow the tunneling of IP packets through to/from the PDN GW (e.g., based on GTP, PMIP, IP in IP or other), whenever data needs to be sent from/to the local network while the UE is currently in the macro network. On the other hand, the optional Type 1C interface between the HeMS and the LSGW is used to provide configuration, software updates, monitoring, and other network management functionality for the LSGW and the small cell network as a whole.

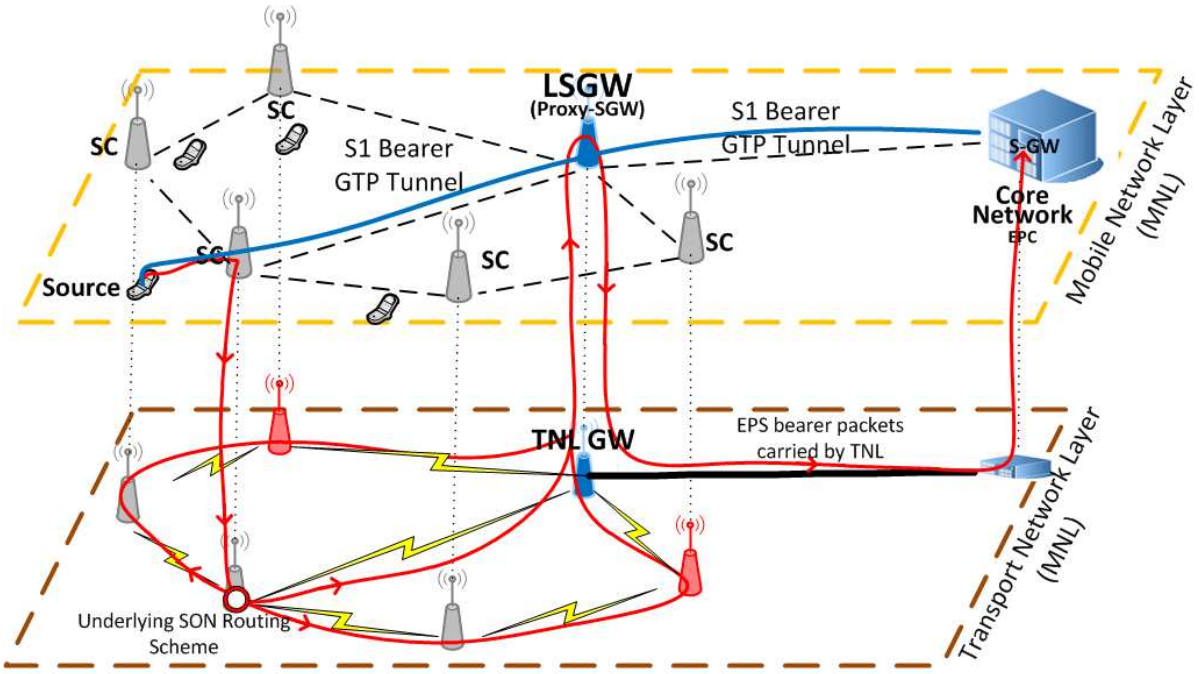


Figure 6.2: All-wireless Network of Small Cells.

6.2 Functional Entities supporting the Network of Small Cells

This section focuses on the architectural issues that need to be solved to be in compliance with the 3GPP Technical Specifications. The concept of an all-wireless NoS, along with the routing scheme described in this chapter, has implications in the 3GPP architecture. However, these implications are local to the NoS and, therefore, totally transparent to the EPC.

Steering traffic throughout a mobile network involves two main building blocks, namely traffic and mobility management. The classification of the two main routing problems to solve comes naturally with the division between the Mobile Network Layer (MNL), which handles 3GPP procedures, and the Transport Network Layer (TNL), which carries packets throughout the network. Figure 6.2 illustrates an all-wireless NoS emphasizing the division among the MNL and the TNL. With regards to traffic and mobility management, the MNL is in charge of determining the endpoints of the GTP tunnels that carry the data for each bearer in all the relevant nodes. This may be understood as a high-level routing, which we will denote as GTP routing. High-level routing is out of the scope of this dissertation. Once the endpoints have been determined, the TNL is in charge of routing the packets between those endpoints. This is a more fine-grained routing, as it deals with the actual path that packets follow to reach the destination endpoint. These paths traverse transport network nodes, such as routers or

switches. Although the procedures handled by these nodes are not specified by 3GPP, they are key components for an efficient operation of NoSs. One of these procedures to define is the underlying SON TNL routing, which is the main focus of this dissertation. In particular, the 3GGP architecture requires the addition of an entity called LSGW, which acts as gateway at the transport network level. Furthermore, the introduction of such low-level TNL routing procedures requires modifications in the transport network building blocks of small cells. The following sections describe in detail each one of these implications.

6.2.1 Local Small Cell Gateway

In order to provide EPC functionalities within the NoS and to keep data and signaling traffic within the local network, a new element is introduced in the network architecture: the Local Small cell Gateway (see Figure 6.1 and Figure 6.2). LSGWs are mandatory network elements for the wireless multihop backhaul considered in this dissertation. From the logical point of view, the LSGW is located in the edge of the NoS and acts as a network manager for local mobility, traffic routing, access control, authentication, power management, and fault management. In addition, the LSGW can also provide Local IP Access (LIPA) or Selected IP Traffic Offload (SIPTO) services, which are particularly relevant in small cell networks [98, 99].

In addition, the LSGW can also provide Local IP Access (LIPA) or Selected IP Traffic Offload (SIPTO) services, which are particularly relevant in small cell networks [99]. The LSGW allows the NoS to be managed by the mobile operator as a single, aggregate entity with respect to many of these functions, i.e., hiding network internals to the EPC whenever possible while exposing the features the mobile operator needs access to.

From the functional point of view, an LSGW encapsulates two main functional entities, namely, the Proxy Mobility Management Entity (P-MME) and the Proxy Serving Gateway (P-SGW). Both entities operate on behalf of the corresponding EPC entities (i.e., MME and S-GW) in order to perform mobility and data traffic management functions within the NoS, respectively. In particular, the P-SGW performs S1 bearer termination and mapping between the NoS and the EPC, as well as user-plane data routing from/to the NoS and the EPC. In case there is UE mobility between small cells, the P-MME manages the associated control plane signalling for switching the bearer towards the new endpoint.

In a multi-LSGW environment, packets may opportunistically traverse an LSGW different from the one that terminates the S1 bearer originated at the HeNB. In such situations, the P-SGW function in the source LSGW (i.e., the first LSGW reached by the packet) extracts the IP packet from the geo-

graphic packet, no matter what the bearer endpoint is, and forwards it to the destination LSGW (i.e., the one terminating the bearer) by means of traditional IP routing protocols running over the existing wired/wireless backbone amongst LSGWs. Once the packet has reached the intended LSGW, it is routed into the corresponding S1 bearer between the LSGW and the S-GW in the EPC.

6.2.2 Modifications to Small Cells

Standard 3GPP interfaces and functionalities supported by HeNBs have not been altered. However, some modifications to the transport network layer of the HeNB protocol stack have been made in order to provide additional functionalities within the NoS. In particular, a Routing Sublayer has been added to the protocol stack that faces the (all-wireless) NoS. This routing underlay in the transport network layer is the key enabler of the SON scheme conceived for transporting data traffic through the wireless mesh network. The regular relaying of messages from S1-AP to Radio Resource Control (RRC) has not been modified either. In terms of protocol signaling, the LSGW acts as a contention entity, hence preventing unnecessary control-plane traffic from reaching the EPC. It must also be noted that all routing and LLM operations have been designed to be consistent with 3GPP procedures.

Figure 6.3: GeoSublayer introduction in the User-plane protocol stack for S1-U interface in a SC below IP layer.

6.2.2.1 GeoSublayer

The GeoSublayer, a protocol layer between legacy IP and MAC layers in the HeNB protocol stack, has been added to the HeNB protocol stack (see Figure 6.3) that faces the (all-wireless) NoS. The GeoSublayer respects the standard interfaces in the HeNB protocol stack. Therefore, the HeNBs keep working in the same way, as standard interfaces are respected. This underlay in the transport network layer is the key enabler of the SON low-level routing scheme that will be described in the following chapters.

In addition, the GeoSublayer is also in charge of providing Local Location Management procedures. User location schemes in 3GPP networks rely on mobile subscriber identifiers, such as the Serving Temporary Mobile Subscriber Identity (S-TMSI). Since the routing protocol presented assumes that a HeNB can obtain the geographic coordinates of the intended destination within the local network, a Local Location Management scheme is needed in order to map a given 3GPP mobile subscriber identifier to the geographic coordinates of the HeNB where the subscriber's UE is currently camped on. As LLM schemes are tightly coupled to 3GPP control-plane procedures, such as Paging or Tracking Area Update (TAU), its detailed description is beyond the scope of this dissertation.

As the concept of NoS has not been standardized in 3GPP Technical Specifications yet, there is no 3GPP mechanism that allows HeNBs to establish S1 bearers directly between them (i.e., without having to traverse an S-GW or a P-SGW) [100]. In our scenario, IP packets are routed over the NoS in a completely transparent way to existing 3GPP control- and user-plane procedures. In order to solve the local routing issue, HeNBs in the NoS may implement LIPA mechanisms that are able to identify IP packets addressed to local UEs. Once detected, these packets are handed over to the GeoSublayer that performs the routing procedures that will be described in the following chapters.

6.3 Resulting Data Traffic Handling in the Network of Small Cells

Figure 6.2 depicts an example of the resulting NoS scenario. The MNL is in charge of determining the endpoints of the GTP tunnels that carry the data for each bearer in all the relevant nodes. During bearer establishment, tunnel endpoints have been determined using standard 3GPP procedures. For the S1 interface, these procedures are transparently handled by the LSGW to both the HeNBs and the core network entities. Assuming that GTP is used in the S5/S8 interface, three GTP tunnels with their corresponding endpoints are set up, namely one from the HeNB to the LSGW, one from the LSGW to the S-GW, and one from the S-GW to the P-GW. Therefore, in the downlink, when a packet reaches the

P-GW from the Internet, the standard procedure is followed [88]. After applying the traffic flow template to the incoming packet, the GTP tunnel to the S-GW is determined. Once there, the S-GW performs GTP routing in order to send the incoming packet through the appropriate outgoing GTP tunnel. In a conventional network, the other end of the tunnel is a (H)eNB. However, in the proposed architecture, the other end is the P-SGW of the LSGW.

As the LSGW is transparent to the core network for data traffic, the P-SGW behaves as a HeNB. Once the packet reaches the LSGW, the P-SGW functionality performs GTP routing, so that the packet reaches the HeNB. Furthermore, the LSGW is also transparent to the HeNBs, i.e., the LSGW appears to the HeNB as a S-GW. Note that it is important to maximize the amount of traffic managed by the LSGW.

In the case of local traffic, instead of enforcing bearers with the LSGW, an important enhancement for the all-wireless NoS is the introduction of direct bearers between HeNBs. This procedure is based requires the IP address of the destination HeNB, prior to the direct exchange of data traffic amongst SCs. The IP addresses of the HeNBs are known by the LSGW. Thus, it is required a procedure amongst the source HeNB and the LSGW to obtain the IP addresses of the HeNBs. Such a procedure is described in the work presented in [101]. In this way, data plane traffic will be directly routed amongst SCs without having to traverse the LSGW, thus, attaining traffic offloading of local traffic even from the LSGW. This architectural simplification is of primal importance since it can help to mitigate the generation of hotspots around the LSGW in the case of intensive local traffic scenarios.

As for the underlying transport, the transport network layer comes into play in the path between the LSGW and the HeNBs. We assume a large-scale, all-wireless network of small cell scenario, where packets are handled by an underlying routing mechanism to be described in the following chapters. Eventually, the GTP packet will arrive to the HeNB. To carry traffic from the P-SGW to the HeNB endpoint and vice versa through the deployed wireless mesh backhaul, the underlying TNL requires from a SON routing scheme. Note that a similar process is followed in the uplink. Designing such a scheme and making it efficient in this context is the main goal of this thesis.

Chapter 7

From Theory to Practice: Self-Organized Lyapunov Drift-plus-penalty routing

The contributions presented in this chapter were partly presented in [2]. Having seen in previous chapters that there still are some unanswered questions in the field of routing for the wireless mesh backhaul, this chapter intends to provide a contribution to the aforementioned points that remain unsolved. In particular, it introduces a new practical routing mechanism for wireless mesh backhauls called self-organized Lyapunov drift-plus-penalty routing.

In section 7.1, we present the routing problem from a stochastic network optimization [77] perspective. In particular,section 7.1 formulates the network optimization problem using the Lyapunov optimization framework, taking advantage of the previous work by Neely [77]. Algorithms designed in the context of Lyapunov optimization base their operation on the minimization of the Lyapunov drift. The minimization of the Lyapunov drift refers to the minimization of the difference of the queue backlogs in the network in two consecutive time slots. This approach was further expanded by adding a penalty function to the drift. In this way, the theoretical algorithm aims at providing both queue stability and optimization of a given objective function.

Aiming at a practical implementation, section 7.2 proposes a decentralized algorithm that satisfies some of the wireless backhaul requirements mentioned in chapter 5, such as adaptability, scalability, and implementability. In addition to the practical constraints posed by a distributed wireless mesh backhaul environment, note that our scheme need to satisfy some requirements (e.g., even resource consumption, adaptability, implementability), which are defined in chapter 5. This scheme goes beyond previous approaches [29] by proposing a practical distributed backpressure routing strategy that, assisted by 1-hop geographic information, only requires one finite queue at each node to deal with any-to-any communications. Thus, it is practical in the sense that, unlike theoretical centralized algorithms, our scheme enables a distributed and decentralized implementation with low queue complexity (i.e., one finite queue per node) to deal with any-to-any communications.

Section 7.3 provides a detailed study of the resulting practical routing policy. Specifically, using ns-3 [102] simulations under different wireless mesh backhaul configuration setups, this section evaluates how various network performance metrics (throughput, delay, and fairness) are affected by the value of the weight of the penalty function given certain traffic demands, which is a tunable optimization parameter. In particular, we observe how the practical constraints posed by a wireless a mesh backhaul (e.g., scalability, finite queues sizes, and any-to-any traffic patterns) affect the results claimed by the theoretical framework. Consequently, we characterized the strengths and weaknesses of the resulting self-organized drift-plus-penalty routing policy in our intended wireless mesh backhaul environment. Note that ns-3 was selected as the simulator to conduct our evaluations of the routing protocol given its accuracy, which was demonstrated in our work in [4].

This preliminary study is of primal importance to understand the existing gap between Lyapunov drift-plus-penalty strategies and the needs of wireless mesh backhauls. Indeed, the findings of the study in section 7.3 indicate the convenience of varying the optimization parameter of the proposed algorithm, instead of computing the weights using a fixed optimization parameter.

Finally, section 7.4 outlines the main points raised in this chapter. Note that this chapter provides the basics of the routing algorithm that will be evaluated and tuned in next chapter. The contributions in this chapter have been partially published in [2] and [4].

7.1 The Routing Problem

Here, we define the routing problem as a stochastic network optimization problem [77]. Subsection 7.1.1 provides the assumed network model to formulate the routing problem. Subsection 7.1.2 formulates the

routing problem using stochastic network optimization, and subsection 7.1.3 approaches the network optimization problem using the Lyapunov drift and the Lyapunov optimization method described in [77].

7.1.1 Network Model

This subsection specifies the general network model used to formulate the routing problem in a wireless mesh backhaul from a stochastic network optimization [77] perspective. In particular, the model considers a stochastic network that operates in discrete time with unit time slots, i.e., $t \in \{0, 1, 2, ...\}$, where each node in the network is seen as a queue. The goal is to design a network controller that takes routing decisions from the observation of the current network state and the queues.

At every time slot, the network controller performs some routing action chosen based on the observed network state. The chosen routing action may incur a cost, but may also serve some amount of traffic. This traffic causes congestion, and thus, leads to backlogs at nodes in the network. In what follows, we provide a generic description of the network states. Then, we define the system processes determining the evolution of node queues backlogs, and so, the evolution of network states.

7.1.1.1 Network State

The network state is defined by network variables denoting random events (e.g., channel state and arrivals). In our case, it is defined as the vector of wireless link rates, and the stochastic process defining the random number of arrivals. The vector of wireless link rates could change at every time slot due to the wireless link conditions, or node failures.

The processes characterize the random events happening in the network. We assume that there are a total of finite number M of different random network states, and define $S = \{s_1, s_2, ..., s_M\}$ as the set of possible states. Each particular state s_i indicates the current network state parameters, such as a vector of channel conditions for each link, and a collection of other relevant information about the current network arrivals.

Let $S(t)$ denote the random network state at time t. We assume that $S(t)$ evolves according to some general probability law, under which there exists a steady state distribution of transition probabilities between the different states. There is a certain steady state probability p_i of being in state s_i. We assume $S(t)$ is a finite state irreducible and aperiodic Markov chain. The network controller can therefore observe a Markov process $S(t)$ at the beginning of every slot t. But the statistics of the $S(t)$ process, including the p_i probabilities and transition probabilities, are not necessarily known. As shown in [103],

these set of assumptions on the network state are crucial to solve the network optimization problem formulated in 7.1.2 with the Lyapunov optimization method.

7.1.1.2 Queuing, Arrivals, and Departures

Each node in the network can be represented by a queue. The network is then described by a vector of queue backlogs written in vector form, $Q(t) = (Q_1(t), ..., Q_r(t)) \in \mathbb{Z}_+, t = 0, 1, 2, ...$, where $r \geq 1$ is the number of nodes in the network. Let R be the set of all nodes in the network.

Furthermore, we define the vectors determining the arrivals and departures to/from the set of nodes/queues. The first one is the set of arrivals $a(t) = (a_1(t), ..., a_r(t))$ at time slot t, in units of packets. Every $a_i(t)$ accounts for both the exogenous (i.e., packets received from other nodes/queues, and the endogenous arrivals (i.e., packets generated at node i arriving at queue $Q_i(t)$). The second vector determines the set of serviced packets by every node $b(t) = (b_1(t), ..., b_r(t))$, in units of packets at time slot t.

In every time slot t, a network must take a control action $x(t)$. This action determines the arrival and departures to/from the set of nodes. Each transmission is contained within one time slot, and all transmissions during a given slot start and finish at the same time. Assuming that at instant $t = 0$, $Q_i(0) = 0$, the queuing dynamics are reflected by the following equation:

$$Q_i(t+1) = max[Q_i(t) - b_i(t), 0] + a_i(t); 1 \leq i \leq r \tag{7.1}$$

Note that both $a(t)$ and $b(t)$ are random processes that are function of a random state $S(t)$, and a control action $x(t)$. Every time slot the network controller observes a state $S(t)$ and chooses an action $x(t)$ that will define both $a(t)$ and $b(t)$. Then, according to the queuing dynamics defined in Equation (7.1), the queue backlog process is updated for the next time slot.

7.1.2 Routing Problem Formulation

One important concept used throughout this dissertation is that of the network capacity region. The work from [29] defined the concept of network capacity region. The network capacity region $\wedge$ is the closure of the set of all input rate matrices (an input rate matrix defines the set of rates between each possible pair of nodes in the network) able to be stably supported by the network. Therefore, given an injected input rate matrix λ, if Equation 7.2 is satisfied, then λ is inside $\wedge$. Furthermore, before presenting the stochastic network optimization problem, we must first give a definition of stability in a wireless mesh backhaul.

Let Q_i denote the non-negative queue backlog of node i at time slot t. The set of queue backlogs $Q(t) = (Q_1(t), ..., Q_r(t))$ in a wireless mesh backhaul are strongly stable if they satisfy the following expression:

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\{Q_i(\tau)\} < \infty, \forall i \in R \tag{7.2}$$

In addition to maintaining the queue backlog vector process $Q(t)$ strongly stable according to the above criterion, the objective of the network controller is to deliver all data to their intended destinations, while minimizing the time to deliver each data packet. Note that if we assume that in the long term all wireless links would have on average the same level of reliability, this is also closely related to minimize the number of hops traversed by each data packet, and so, the network resources used per packet. In this case, by network resources we mean over-the-air wireless transmissions per packet between nodes in the network.

All data generated from any source node that is destined for a particular node $c \in 1, ..., r$ is classified as commodity c data. Let $A_c(t)$ represent the exogenous arrivals (i.e., packets received from other nodes/queues) of commodity c data at precisely node c. In other words, $A_c(t)$ represents the rate, in units of packets, at which the network delivers packets for a given destination c at time slot t. As in [77], we assume that $A_c(t)$ is a stationary process, and so $E\{A_c(t)\} = A_c$.

We define a non-negative cost function that restricts the set of wireless links traversed towards each destination c. In particular the cost function grows by selecting links that eventually result in an increase of the number of hops towards c. Therefore, a low cost is attained by choosing links that minimize the number of hops to the intended destination c.

Every destination $c \in R$ may pull out data from the network at a different rate, in units of packets. This variability will depend on the network routing controller taking decisions according to the state of the system $S(t)$. In addition, the number of hops to deliver commodity c data generated from node i can also vary. Let $A_c^{h_i^c(t)}(t)$, be the amount of commodity data c generated from node i delivered with paths of a number of hops $h_i^c(t)$. In turn, commodity c data generated from node i may be delivered using different paths with a different number of hops. Let $h_i^c(t)$ denote the number of hops utilized to deliver commodity c data generated from node i on time slot t. There is a minimum number of hops to deliver from node i to node c. This number of hops also depends on t since there may be node or wireless link failures. Let $hmin_i^c(t)$ be the minimum number of hops required to deliver commodity c data generated from node i. The use of wireless link resources increases with the number of hops. We assume that $hmin_c^c(t) = 0$ for all t, since in this case commodity data c does not consume wireless link resources. Note that $h_i^c(t) \geq hmin_i^c(t)$.

The routing algorithm serves amounts of commodity c data generated at node i in a number of hops $h_i^c(t)$. To some extent similar to [80], the objective is to bring this number of hops as close as possible to the minimum number of hops $hmin_i^c(t)$ in order to minimize the cost function. The objective is to minimize the path length for all the traffic load A_c arriving at destination c. The network objective function $y(t)$ to minimize is, therefore, defined as:

$$y(t) \triangleq \sum_{i=1}^{r} \sum_{c=1}^{r} \sum_{h_i^c(t) \geq hmin_i^c(t)} A_c^{h_i^c(t)}(t)(h_i^c(t) - hmin_i^c(t)) \tag{7.3}$$

When routing data from i to c, the cost function $y(t)$ grows with the transfer of commodity data c generated from node i that results in an average number of hops that exceeds the minimum number of hops $hmin_i^c(t)$. Such definition of the cost function in (7.3) can be interpreted as follows. First, trying to minimize the difference between the number of hops $h_i^c(t)$ utilized to deliver commodity c data and the minimum number of hops $hmin_i^c(t)$ has the goal of minimizing the over-the-air resources utilized to route data. Note that $h_i^c(t) \geq hmin_i^c(t)$. Second, the cost function $y(t)$ is highly related to the end-to-end delay. The lower the difference with the minimum number of hops $hmin_i^c(t)$ required to deliver commodity c data, the lower the time devoted for over-the-air transmissions. Thus, the component of delay denoting over-the-air transmission time for the data is minimized. Such network objective parameter also helps to reduce the number of routing loops, since its goal is to approximate as much as possible the number of hops traversed by data sent from i to c to the minimum $hmin_i^c(t)$. Note that given the use of a number of hops equivalent to $hmin_i^c(t)$ implies the use of the shortest path, and the shortest path cannot involve any routing loop. Third, the cost function aims at minimizing the number of hops for the maximum number of data packets, as it would imply a lower consumption of over-the-air resources. Therefore, under scenarios for which the workload requires of more wireless resources than those present in the shortest paths among the respective source destination pairs, our cost function decreases the more workload is served by the shortest paths. Thus, instead of, for instance, randomly deviating from the shortest path a half of the data traffic, our cost function prefers to minimize the rate of traffic deviated from the shortest path. principle use the same set of paths. Instead of this, given two different traffic flows of different rates our cost function decreases when the number of hops traversed by the more intense (i.e., the traffic flow with a higher rate) decreases, compared to the number of hops traversed by the less intense traffic flow.

On the other hand, the queuing delay is another critical component characterizing the end-to-end delay, which is not taken into account by the cost function. This is captured by the component in charge of the minimization of the Lyapunov drift. Our interest is to minimize the average queuing delay. The strong

stability constraint can be mapped to an average queuing delay constraint, since according to Little's Theorem [104] the average queue backlog is proportional to the average queuing delay.

Let $\overline{y}$ be the time average on the long run of the process $y(t)$. We can now formulate the stochastic network optimization problem in which routing control decisions are taken to minimize the time average of the network objective function $y(t)$ while maintaining strong stability in the node queues belonging to the wireless mesh backhaul. It is expressed as follows:

$$\text{Minimize: } \overline{y} \tag{7.4a}$$

$$\text{Subject to: } \limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\{Q_i(\tau)\} < \infty \tag{7.4b}$$

7.1.3 Drift-plus-penalty as a solution to the Routing Problem

We tackle the optimization problem defined in ((7.4a), (7.4b)) using the Lyapunov drift and Lyapunov optimization [77] method. One of the reasons to choose Lyapunov optimization and drift theory is that the resulting algorithm, built upon a quadratic Lyapunov function, does not require any statistical knowledge of the complicated underlying stochastic processes in the network. For instance, they do not require knowledge of the transition probabilities between network states associated with future random events such as variability of link rates or packet arrivals. In particular, the resulting approach takes routing decisions solely requiring knowledge of the current network state $S(t)$.

To fulfill constraint (7.4b), we define a quadratic Lyapunov function $L(t)$ with update Equation (7.1) as follows:

$$L(Q(t)) \triangleq \frac{1}{2} \sum_{j=1}^{r} Q_j^2(t) \tag{7.5}$$

Equation (7.5) defines a function that grows whenever at least one data queue in the queue backlog vector process, which defines the network, grows. Thus, it gives a scalar measure of the size of the queue backlog vector process. The function is non-negative, and it is equal to zero if and only if all the elements in the queue backlog vector are zero. Then, we define the one slot conditional Lyapunov drift as follows:

$$\Delta(Q(t)) \triangleq E\{L(Q(t+1)) - L(Q(t))|Q(t)\} \tag{7.6}$$

where $\Delta(Q(t))$ denotes the difference of Quadratic Lyapunov function from slot t to slot $t+1$.

Taking into account the queue dynamics defined in (7.1), we use the Lyapunov drift-plus-penalty algorithm as described in Lemma 4.6 in [77], which states the upper bounds for the drift-plus-penalty

expression. We obtain the following bounds for the Lyapunov drift, whose structure is exploited for approaching the routing problem:

$$\Delta(Q(t)) \leq B + \sum_{j=1}^{r} Q_j(t) E\{a_j(t) - b_j(t)|Q(t)\} \tag{7.7}$$

where B is a positive constant that satisfies the following for all t:

$$B \geq \frac{1}{2} \sum_{j=1}^{r} Q_j(t) E\{a_j^2(t) + b_j^2(t)|Q(t)\} - \sum_{j=1}^{r} E\{\tilde{b}_j(t) a_j(t)|Q(t)\} \tag{7.8}$$

where $\tilde{b}(t) = min[Q_j(t), b_j(t)]$. We add to both sides the term $VE\{y(t)|Q(t)\}$, where $V \geq 0$ is a scalar control parameter. Consequently, we obtain the following bounds:

$$\Delta(Q(t)) + VE\{y(t)|Q(t)\} \leq B + \sum_{j=1}^{r} Q_j(t) E\{a_j(t) - b_j(t)|Q(t)\} + VE\{y(t)|Q(t)\} \tag{7.9}$$

The minimization of the Lyapunov drift-plus-penalty expression, and so the problem formulated in Equations ((7.4a),(7.4b)), can be tackled using the expression in Equation (7.9). Rather than directly minimizing the Left Hand Side (LHS) of the equation, the strategy aims to minimize the provided bound in the Right Hand Side (RHS) of Equation (7.9). Given that B is a positive constant the resulting strategy seeks to minimize $\sum_{j=1}^{r} Q_j(t) E\{a_j(t) - b_j(t)|Q(t)\} + VE\{y(t)|Q(t)\}$ every time slot.

7.1.3.1 Algorithm

The implementation of a centralized strategy for solving ((7.4a),(7.4b)) requires to observe the network state $S(t)$ (i.e., the vector of arrivals, and the matrix of link rates for every link in the network) and the queue backlog vector process to take appropriate routing decisions every time slot. From the Lyapunov optimization theorem in [77], the optimization problem ((7.4a),(7.4b)) suggests a network routing controller that, every time slot, minimizes the right hand side of (7.9). This algorithm can be obtained by using the framework of opportunistically minimizing an expectation, which is described and proved in section 1.8 of [77]. Thus, the algorithm would work as follows. Let $x(t)$ denote a network routing control action that has an associated objective given by the objective function $y(t)$. And let $X_{S(t)}$ be the set of control actions that can be chosen give the network state $S(t)$. In every time slot t, given both the current state $S(t)$ and queue backlog state $Q(t)$, choose a routing control action $x(t) \in X_{S(t)}$ that solves the following problem:

$$\text{Minimize: } \sum_{j=1}^{r} Q_j(t) \hat{a}_j(x(t), S(t)) - \hat{b}_j(x(t), S(t)) + V\hat{y}(x(t), S(t)) \tag{7.10a}$$

$$\text{Subject to: } x(t) \in X_{S(t)} \tag{7.10b}$$

As a result of the decision taken at every time slot, the queues are updated according to Equation (7.1). Note that the resulting algorithm chooses routing control actions based on the instantaneous network state, and independently of the probabilities of staying at the different network states. After observing the current network state $S(t)$, it takes a decision $x(t)$ that intends to minimize functions of $x(t)$.

Regarding performance guarantees, as demonstrated in [77], the suggested routing strategy from Figures ((7.10a),(7.10b)) can guarantee a time average objective that is within O($1/V$) to the optimal network objective value denoted by y^*. Therefore, as V grows large, the time average objective $\overline{y}$ can be pushed arbitrarily close to the optimal y^*. However, pushing close to the optimal value y^* the network objective parameter $\overline{y}$ incurs into a large queuing delay. Specifically, when achieving the O($1/V$) close-to-optimal objective, one can only guarantee that the incurred network delay is O(V). The closeness to the minimum value denoted by y^* comes determined by the control V parameter. Interestingly, the resulting policy has near-optimal performance, which can be proved without the knowledge of optimal performance.

7.1.3.1.1 Implementation

In practice, the implementation of an optimal algorithm entails several challenges. First, the optimal implementation would require a centralized scheduling to calculate the set of non-interfering wireless links at each time slot that can concurrently transmit. In practice, this would require a fast communication link of every small cell with a centralized controller that will compute this set of non-interfering links, which would also require computing power. Second, the network objective parameter $\overline{y}$ focuses on minimizing the number of hops traversed by data to reach their intended destination. Given that the solution must handle any-to-any traffic patterns, a centralized controller would compute and maintain a routing table with all the possible paths between any pair of nodes. Note that under the dynamic environment posed by a wireless mesh backhaul, the computation of routes between any pair of nodes would imply the continuous exchange of high amounts of routing control overhead and computation towards the central entity. Thus, at each time slot for each node pair (i, c) the central entity requires to compute paths of minimum length $hmin_i^c(t)$, and all the paths $h_i^c(t) > hmin_i^c(t)$. The number of entries between each node pair will be bounded by the number of nodes directly connected to i. Third, the central controller should also have global knowledge of the node queue backlog and the network state (e.g., arrivals and channel state), which should be measured at the small cells and transferred to this central controller every time slot.

All the aforementioned issues constraint the optimal implementation of the backpressure routing policy. In light of these observations, section 7.2 seeks to relax these assumptions and approximate the

backpressure routing policy on a practical and decentralized way.

7.2 The quest for Practicality

There are many practical issues to take into account when porting this backpressure routing policy to a real wireless mesh backhaul. The requirements of the mesh backhaul (routing any-to-any traffic patterns, each node with finite queue sizes, adaptability to wireless backhaul dynamics) complicates the introduction of a backpressure routing controller as the size of the network grows. Furthermore, the assumption of a centralized controller, such as in [29], collecting all the statistics of all the nodes/cells is in general unfeasible in a wireless mesh backhaul scenario aimed to optimize the use of wireless resources.

Subsection 7.2.1 proposes a decentralized approximation using low-complexity heuristics to tackle each of these issues. Further, subsection 7.2.2 illustrates one of the key features of the resulting policy. Finally, subsection 7.2.3 lists the main characteristics of the resulting decentralized backpressure routing policy.

7.2.1 The Practical Solution

In this subsection, we consider a decentralized approximation of the solution to the problem proposed in Equations ((7.10a),(7.10b)). The routing policy is derived using the Lyapunov framework in [74, 77], computing the weights independently in every node in the network. In our solution, we aim at applying the same philosophy for minimizing Equation (7.10a) as in the centralized scheme. But this is done independently at each node considering the local information available. In this distributed backpressure routing policy, every node calculates a weight for every link (i, j) in every time slot. In particular, the weight denoted by w_{ij} of a link (i, j) is calculated as follows:

$$w_{ij}(t) = (\Delta Q_{ij}(t) - V p_{i,j}(t))\Upsilon_{ij}(t) \tag{7.11}$$

Here $\Delta Q_{ij}(t) = Q_i(t) - Q_j(t)$ is the queue differential with $Q_i(t)$ and $Q_j(t)$ representing the backlog of nodes i and j, respectively, $p_{i,j}(t)$ is the routing penalty function that depends on the cost function defined in Equation (7.3), and $\Upsilon_{ij}(t)$ is the estimated wireless link rate. The V parameter is a constant that trades the importance of $p_{i,j}(t)$ with respect to $\Delta Q_{ij}(t)$. Specifically, the selected neighboring node for transmitting the current packet being routed maximizes the link weight $w_{ij}(t)$ in (7.11) between the

local node i and all its neighbor nodes $j \in J$, where J represents the set of 1-hop neighbor of the local node i. Formally, the selected neighbor node j^* is such that:

$$j^* = \arg\max_{j \in J} w_{ij} \tag{7.12}$$

If there is no neighboring node $j \in J$ with a weight $w_{ij}(t) > 0$, the data packet is kept at the local node i. Furthermore, in this case, the routing control algorithm considers that it is even better for the network not to forward a data packet. Therefore, this routing policy also involves scheduling decisions in the sense that, in addition to deciding the next hop, it also regulates the arrival rate of data packets to lower layers.

Note also that while the centralized solution assumes that the network operates in slotted time, the real network operates in continuous time. This affects the calculation of the weights that is done on a per-packet basis rather than calculated on a time slot basis. To calculate the weight $w_{ij}(t)$, each node i exchanges with its neighbors j in the neighbor set J both the queue backlog and geographic information, which is shared between neighbor nodes by periodically exchanging HELLO messages.

A detailed description of each component defining the decentralized solution in (7.11), and their mapping with the centralized solution defined in Equations (7.10a) and (7.10b) follows. The data packets are stored in queues $Q_i(t)$ and $Q_j(t)$. The number of stored data packets in the queues depends on the difference between serviced packets and input arrivals at each node. Let ΔQ_{ij} denote the the difference of physical queue lengths between node i and node j. Therefore, $\Delta Q_{ij}(t)$ increases $w_{ij}(t)$ as the difference between $Q_i(t)$ and $Q_j(t)$ increases. Consequently,$\Delta Q_{ij}(t)$ approaches the minimization of the Lyapunov drift $\Delta(t)$ in (7.10a) in a decentralized way.

The state of the channel in the distributed solution is denoted by $\Upsilon_{ij}(t)$, which estimates the rate of link (i, j). This parameter can be locally estimated using information from the local network interface. Therefore, the resulting solution prioritizes high-capacity links over low capacity links for transmission, as the weight will be higher for those links.

The penalty function $p_{ij}(t)$ focuses on delivering data to their intended destinations. For decentralization purposes, this would require that each node would be aware of the number of hops to reach each potential destination. In turn, this would involve the knowledge of the topology in each node on a distributed way. Even though this is feasible, it will incur into high amounts of routing control overhead exchanged between all the nodes.

Our approach to implement such sense of proximity to the intended destination on a distributed and

practical way is to assume local geographic information is present in every node to have a sense of proximity to the intended destination d. In addition, geographic information of the intended destination d is encoded in data packets. Therefore, instead of flooding the wireless backhaul to compute end-to-end routes, we leverage geographic information to have a sense of proximity to the destination. Thus, let $p_{i,j}^{d}(t)$ denote our penalty function computed as the cost to traverse the link between node i and node j to reach destination d. The penalty function can be computed on many ways, as we will show in the following chapters. In its basic form, and assuming a regular network, we can locally compute the penalty of transmitting over link (i, j) to reach destination d as follows:

$$p_{i,j}^{d}(t) = \begin{cases} +1 & \text{i closer to d than j} \\ -1 & \text{i further to d than j} \end{cases} \tag{7.13}$$

Therefore, Equation (7.13) minimizes the Euclidean distance to the destination by giving a sense of relative proximity between two neighbor nodes (i, j) and the destination d. With an appropriate coordinate assignment, the minimization of the Euclidean distance can also be mapped to the minimization of the number of hops. Intuitively, this heuristic approach gives us an approximation of over-the-air transmission time reduction, as it avoids data packets to take paths that lead them farther away from their intended destinations. Geographic information enables the computation of the distributed penalty function (7.13) in a decentralized way, hence avoiding the computation of end-to-end routes. Furthermore, as in (7.10a) the penalty function is parametrized by a non-negative and constant control parameter $V \geq 0$ that trades queue occupancy for penalty minimization. Subsection 7.2.2 provides practical details on the role of the V parameter.

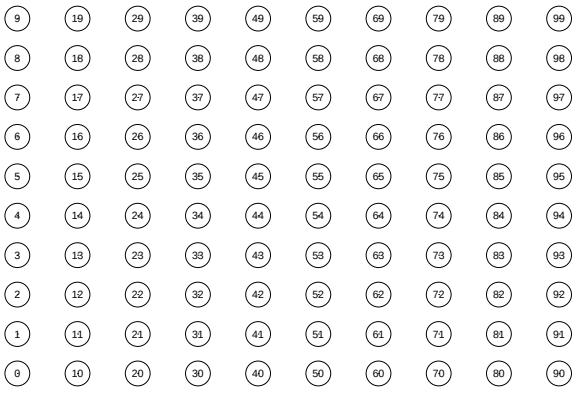


Figure 7.1: Grid Wireless Mesh Backhaul.

7.2.2 Illustration

This subsection illustrates the main features of the resulting distributed policy. The main feature of the resulting routing policy is that it distributes the wireless mesh backhaul resource consumption (e.g., node data buffers), and this is a very relevant characteristic nonexistent in other routing schemes. More specifically, as network load increases, it is in general interesting to increase path utilization by evenly distributing workload for a better aggregate network performance. However, despite this distribution, mechanisms are put into place so that packets eventually reach the destination and appropriate performance levels are attained. The heat map in Figure 7.2 illustrates the aforementioned concept by showing how traffic sent by a source node labeled with a 4 in Figure 7.1 of the 10x10 wireless backhaul grid is transmitted towards the destination (i.e., the node labeled with a 94 in Figure 7.1). Different colors correspond to a different number of packets handled by each node in the grid.

As shown in Figure 7.2, the resulting routing policy distributes resource consumption over the grid, as most nodes in the grid contribute to send data packets to the destination. As can be observed in Figures 7.3 and 7.4, the V parameter denotes the importance given to the penalty function. As the V parameter grows the emphasis given to the geographic component in the distributed backpressure routing policy increases. This translates into packets originated at source node of the grid to reach destination node in a lower number of hops.

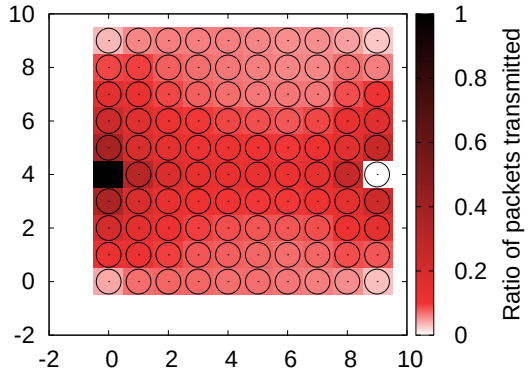


Figure 7.2: Use of the network resources with a low V parameter.

7.2.3 Properties of the resulting Solution

The properties of the resulting solution can be summarized as follows.

- *Dynamic*: The protocol takes routing decisions on a per-packet basis. Data are routed over wireless backhaul links that help them make progress towards the appropriate destination without

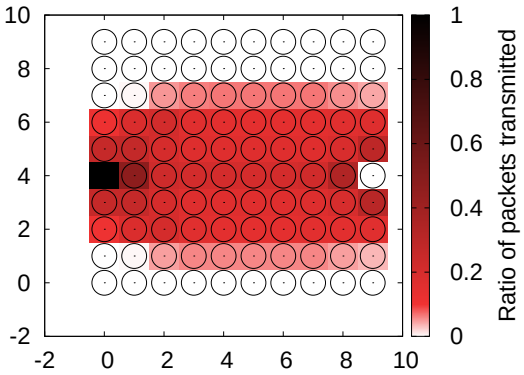


Figure 7.3: Use of network resources with a medium V parameter.

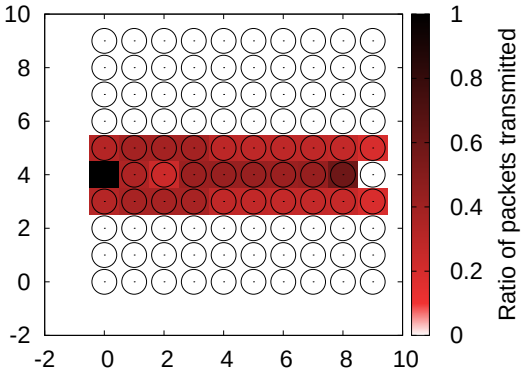


Figure 7.4: Use of network resources with a high V parameter.

restricting the set of links to cross through a pre-computed routing table.

- *Distributed*: The resulting routing policy is also distributed because it does not require a centralized entity for its implementation.

- *Scalability*: The resulting protocol merely requires one data queue per each node in the network, unlike previous theoretical solutions that required one data queue per traffic flow.

- *(Quasi)-stateless*: The protocol just requires state information of queue backlogs and geolocation of 1-hop neighboring nodes.

- *Low Control Overhead*: As a result, and apart from the common HELLO message exchange, the protocol does not add any messaging control overhead. Note that even the advertisement of queue lengths by means of HELLO messages can be avoided in loaded networks. This can be done by piggybacking queue lengths in data packets, which could be overheard by all neighboring nodes. Furthermore, it can be implemented in real wireless mesh backhauls with low control overhead.

- *Lower layer Agnostic*: The resulting distributed routing policy is agnostic from lower layers, no

matter the technology underneath. In any case, the routing policy regulates the flow rate of packets sent from the routing layer to the access layer.

- *Practical*: Eases the implementation for wireless mesh backhauls.
- *Distribution of resource consumption*: As illustrated in previous subsection, the protocol (when configured with a low V parameter) is able to route traffic using all the backhaul rather than utilizing a fixed set of paths computed a priori, hence distributing the resource consumption in the wireless backhaul.

The next subsection provides a study of the implications brought by these properties in terms of mobile backhaul metrics (throughput, delay).

7.3 Studying the resulting distributed solution

This section studies the routing policy determined in previous section against backhaul performance metrics, such as throughput and latency. By means of ns-3 simulations under different wireless mesh backhaul setups, we study the impact of the weight of the penalty function on the network performance metrics. In particular, we vary the weight of the penalty function V and the input rate matrices (i.e., source-destination pairs and input rates) to evaluate the wireless mesh backhaul response in terms throughput, delay, and fairness.

7.3.1 Evaluation Methodology

The network setup described next is common among all the experiments. Figure 7.1 shows the wireless mesh backhaul under evaluation. In particular, the wireless mesh backhaul model used is a grid of 100 SCs connected amongst them through a WiFi mesh backhaul. In particular, every node has a single WiFi 802.11a card configured to the same channel at a rate of 54Mbps. We used WiFi as backhaul technology because it has been shown a low-cost and flexible yet challenging technology choice for the small cell wireless backhaul [105]. Since small cells are often deployed in clutter, such a wireless backhaul possibly needs non-line-of-sight (NLOS) technology that has the ability to use both Point-to-Point (PTP) and Point-to-Multipoint (PTM) techniques. Among other reasons, the routing scheme laying on top determines the success of such a wireless backhaul configuration.

In terms of packet losses, the goal is to study the behavior of the distributed routing policy presented in section 7.2. To do so, we implement a simple physical layer model so that interference due to hidden

nodes and path propagation loss are avoided. The queuing discipline and maximum queue size are preset to FIFO and 400 packet respectively. Hence, the evaluation methodology considers a grid wireless mesh backhaul with no interference (hence few retransmissions), and bounded queue backlogs.

In the following evaluation, we carried out 24 different cases in which two different randomly chosen source nodes in the grid inject a UDP CBR flow towards two different random destination nodes. The UDP payload is configured to 1460 bytes. Each case is repeated 6 times incrementing the input rate of each flow (i.e., 500Kbps, 1Mbps, 1.5Mbps, 2Mbps, 2.5Mbps, and 3Mbps). Furthermore, all the resulting experiments are evaluated for 12 different V parameters ranging from 1 to 500. As a result, we performed a total of 1728 ns-3 simulations. The duration of every experiment was 300 seconds. The UDP flows are injected during a window interval of 150 seconds denoted by τ. Specifically, τ ranges from the second 5 to the second 155 in the ns-3 simulation. All the network metrics described next are calculated during the τ interval.

7.3.2 Network Performance Metrics

In order to characterize the proposed routing protocol explained in previous section, we have studied its behavior under the following network performance metrics:

Throughput and Delay: These are most likely the main performance metrics a routing protocol designed for wireless mesh backhaul must satisfy. The throughput attained by the network as well as the end-to-end delay of packets traversing the network during τ are calculated.

Furthermore, we calculate the end-to-end delay attained by data packets traversing the network during the injection period. Both metrics are evaluated for different input rates and V configuration parameters.Throughout this and the following, we use the terms delay and latency interchangeably.

Fairness: We define the wireless mesh backhaul to be fair if, over τ interval, the number of packets received from each source-destination pair in the wireless mesh backhaul is approximately the same, since both flows have the same characteristics. Hence, we see the network as fair with regards to the throughput attained by each flow. We use the Worst Case Fairness Index (WFI) in order to compare the throughput attained by the two flows injected to the network. The WFI is computed as follows:

$$WFI = \frac{\min\limits_{1 \leq i \leq n} \{x_i\}}{\max\limits_{1 \leq i \leq n} \{x_i\}} \tag{7.14}$$

where x_i is the ratio between the measured throughput and the expected throughput of i_{th} flow. The WFI is appropriate to detect discrepancies in the network when a small number of flows is injected to the network.

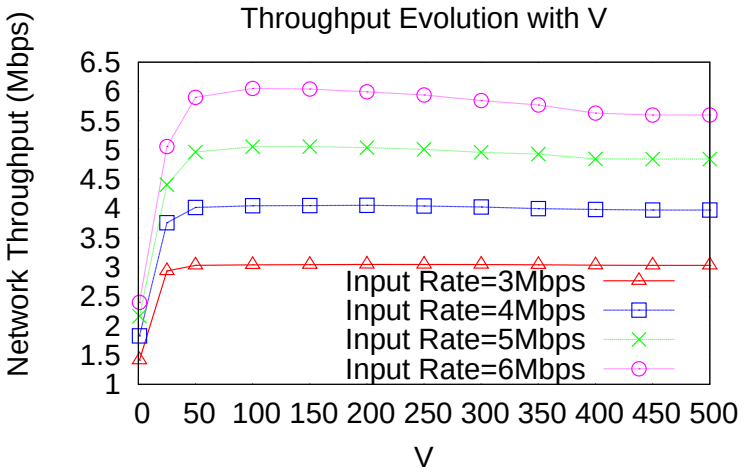


Figure 7.5: Average Network Throughput evolution with the V parameter.

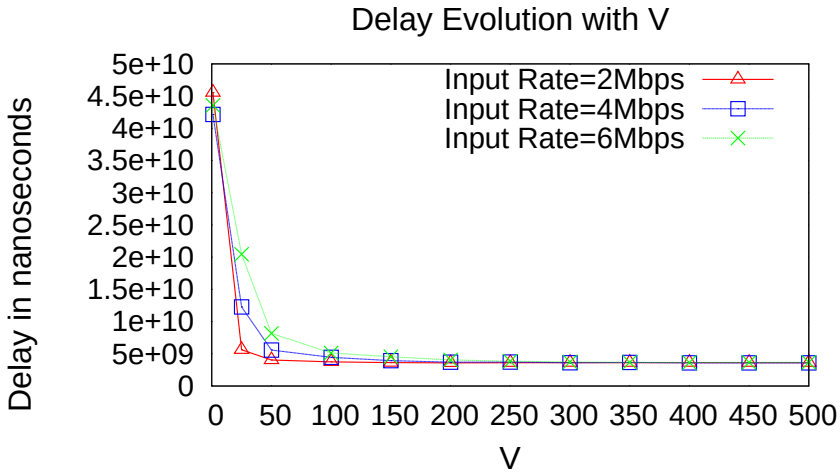


Figure 7.6: Average End-to-end Delay evolution with the V parameter.

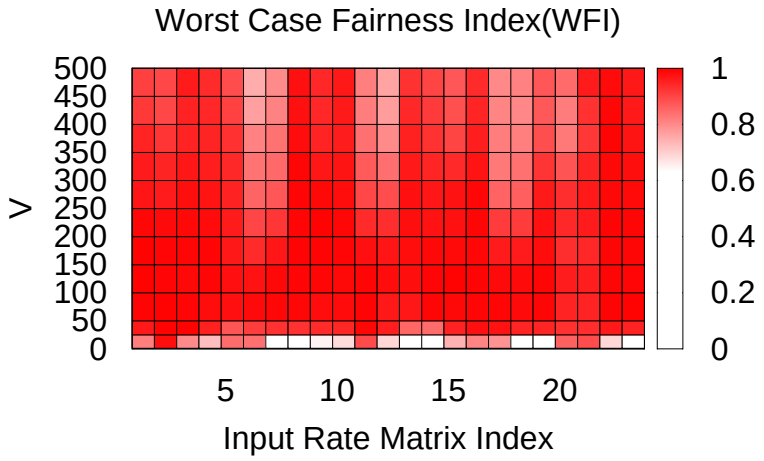


Figure 7.7: Worst Case Fairness Index evolution with the V parameter.

7.3.3 Degradation of network metrics when operating near the minimization of the Lyapunov drift

We study the routing algorithm when it is operating with a low V parameter. In this case, the algorithm merely aims at minimization of the Lyapunov drift (i.e., backpressure algorithm). Therefore, practically just information about queue backlogs is used to forward packets. As can be reflected in Figures 7.5, 7.6, and 7.7 the wireless mesh backhaul suffers from throughput, delay, and fairness degradation for Vs roughly ranging from 1 to 25.

The reason for such throughput degradation, low fairness, and the high end-to-end delay in Figures 7.5, 7.6, and 7.7 is the low importance given to the penalty function. Moreover, the fact of maintaining a single queue per node breaks one of the primary assumptions made by [29]. Recall that the seminal work by Tassiulas et al. considered one queue per flow instead of one queue per node. Both facts lead packets to take close to random walks to reach their intended destination during τ interval, yielding into high delays for delivered packets within τ. Because of this, after τ, a high percentage of data packets still remain at data queues of nodes in the wireless mesh backhaul. For instance, we observe that a quasi backpressure algorithm (i.e., V=1) obtains just a 43%, and 32% of packet delivery ratio for the 3Mbps, and 6Mbps case, respectively. Likewise, this randomness in the trajectories taken by packets leads to high variability in the throughput attained by each flow, which in turn, leads to a high WFI variability (e.g., for V=1 WFI ranges between 0 and 0.96).

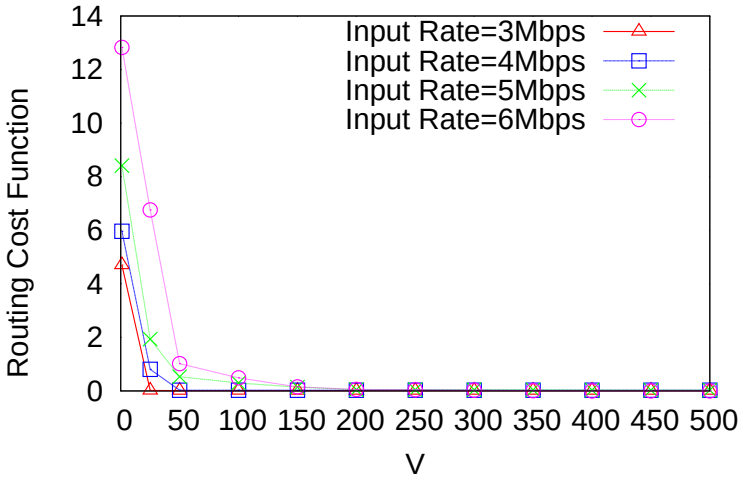


Figure 7.8: Routing Cost Function evolution with the V parameter.

7.3.4 Network objective within $O(1/V)$ of optimality

Recall that the routing control policy is designed so that the V parameter emphasizes the network objective function (see Equation 7.11). Roughly, the aim of the distributed penalty function is to reduce the path length followed by all the packets reaching their respective destinations. This can be seen as an approximation of minimizing over-the-air transmission time experienced by packets, and so the subsequent reduction of the resources utilized by each packet. Interestingly, Figure 7.8 illustrates that the average routing cost function evolves with $O(1/V)$. This confirms that our decentralized approach of the routing cost function calculated independently in every node in the network is a good approximation of the intended routing cost function. As V increases, the path followed in average by a packet makes more progress to the destination than random path selection. Once the V parameter assigned to each node in the network makes sufficient emphasis in the penalty function component, the deviation of data packets from the shortest path between any pair of nodes in the network decreases, thus, minimizing the wireless network resources. Further, the rate at which packets arrive at the destinations (see Figure 7.5) increases. As a result, this leads to a minimization of the routing cost function.

On the other hand, note that Figure 7.6 experiences a similar evolution with $O(1/V)$, as Figure 7.8. The reason for this is that the optimization of the routing cost function is highly related to the minimization of end-to-end delay, given the reduction of the number hops targeted by the routing cost function.

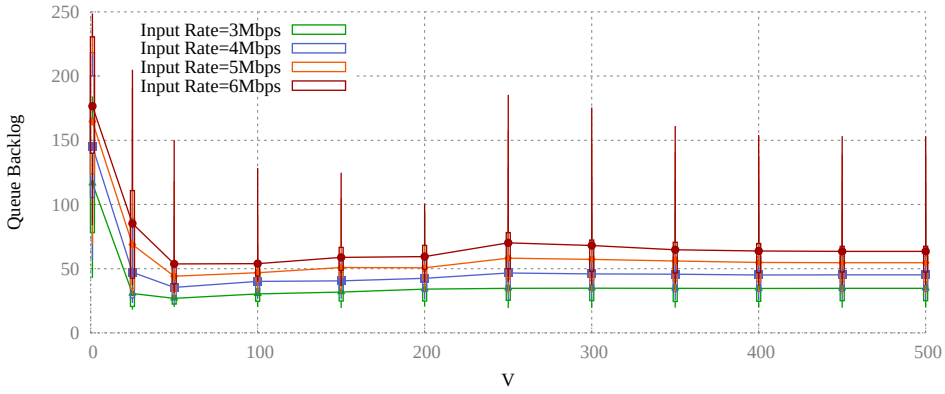


Figure 7.9: Average queue backlog evolution with the V parameter.

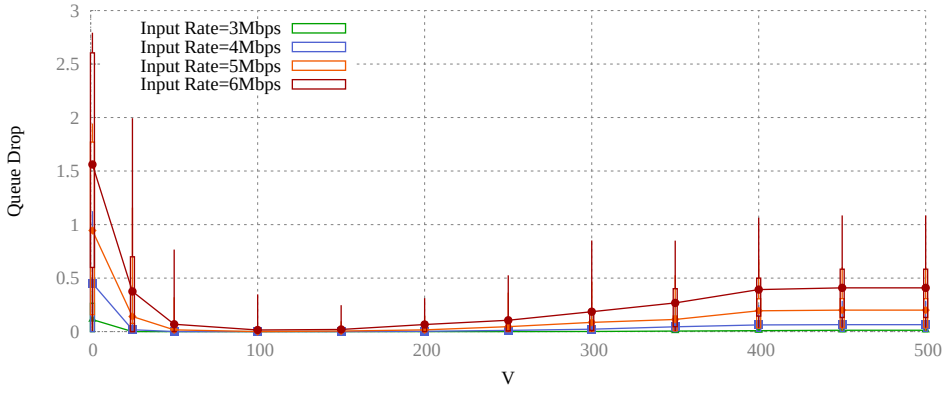


Figure 7.10: Average queue drops evolution with the V parameter.

7.3.5 Queue backlog increase with $O(V)$

In [77], Neely proofs that there is a objective-backlog trade-off of $[O(1/V), O(V)]$. Figures 7.9 and 7.10 illustrate both the average queue backlog and number of queue drops of a node in the grid mesh backhaul network, respectively. Recall that the maximum queue size is of 400 packets.

We highlight three main observations from these figures that affect that constraints the increase of the queue backlogs with parameter V in a network with finite queue sizes. The first observation is the high average queue backlog and queue drops when the algorithm is close to the the Lyapunov drift minimization. Yet, this might seem contradictory with theoretical bounds provided by $O(V)$, note that in this case we neither have established end-to-end routes nor a queue per destination. Because of this, all the nodes in the network are potential forwarder candidates of a packet. The second observation is that, when the penalty function acquires importance (i.e., increasing V), Neely's growth of the average queue backlogs (and so average time in queues) with $O(V)$ for used nodes is satisfied up to the bound determined by the network (i.e., 400 packets). After this bound, queue drops occur in the network, and so the growth with $O(V)$ is not satisfied. Third observation is that the routing policy experiences the same behavior once V=400, which is equivalent to the maximum queue backlog allowed for any node in the network. This is because increasing the V parameter beyond 400 does not yield into changes in the packet distribution in the network. Thus, the network experiences no changes in terms of queue backlogs or queue drops.

7.3.6 The dependence of V with the queue size

Figure 7.5 illustrates that for and 5Mbps, and specially for 6Mbps there is a decrease in the measured network throughput for high V parameters, to an extent to which the average input rate matrix cannot be served. The reason for this reduction is that as the V parameter increases, the routing policy restricts the set of candidate nodes used to forward a data packet (an illustrative example of such reduction can be found in Figures 7.2, 7.3, and 7.4), given the increasing importance of the penalty function with respect to the Lyapunov drift. Restricting the set of nodes used to forward packets incurs into higher congestion of used nodes given the higher number of data packets they have to handle. This can cause an increase of queue backlogs of used nodes, which may lead to queue overflows when the maximum size of a data queue is exceeded.

This dependence between the queue size and the V parameter can be showed in other network metrics such as fairness. Figure 7.7 clearly demonstrates that the network fairness depends on the V parameter.

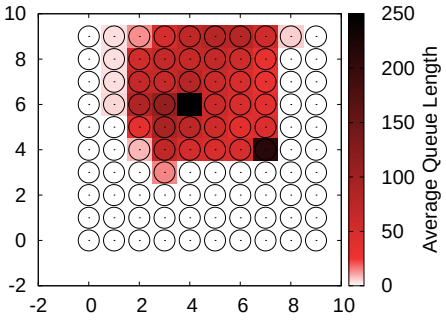


Figure 7.11: Average Queue Length per second during the measured windows interval. Fairness Case.

Precisely, only two of the twenty-four pair of flows under evaluation are able to get a fairness index over 0.85 no matter the V parameter. The rest of the flow pairs experience between 20% and 100% of variability with respect to the V parameter. Conversely, as can be shown in Figure 7.7, there are some cases ranging from $V = 100$ to $V = 150$ in which the network experiences a WFI bigger than 0.94, no matter the input rate matrix. Furthermore, within this configuration, network throughput and delay experience near-optimal attained values. It is important to note that the V parameter choice is lower than the maximum allowed queue backlog in nodes, hence attaining an appropriate trade-off between direction and congestion in used nodes.

On the other hand, Figure 7.7 depicts that for some input rate matrices the network experiences a decrease in its WFI for high V values, given the lower number of resources (i.e., nodes) the routing control policy is allocating. Indeed, Figure 7.7 shows that depending of the source-destination pairs sometimes high V values are unnecessary (e.g., see 21 and 22 input rate matrix index) to experience a lack of resources in one of the injected flows, and so unfairness. For instance, Figure 7.12 illustrates one of this cases. In this case, the number of nodes used to route the flow originated at $(7, 2)$ to $(1, 2)$ is reduced to an extent in which packets cannot be delivered, hence not providing the network resources required to serve the traffic flow. Thus, for the same V parameter, the allocation of resources provided by the algorithm (i.e., nodes) to route data packets is different between both flows.

7.3.7 The location of the source-destination pairs matters

Figure 7.7 clearly demonstrates the dependence of the WFI with the input rate matrix injected to the network. In particular, the magnitude of queue overflows depends on the input rate matrix injected to the mesh backhaul (e.g., up to 21% of queue drops in the 6Mbps case). We observe the location of the source-destination pairs representing the input rate matrix has a big impact on the response of the routing control policy in terms of backhaul performance metrics. Precisely, flows might experience degradation

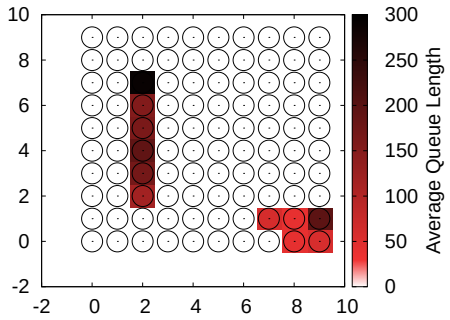


Figure 7.12: Average Queue Length per second during the measured windows interval. Unfairness due to lack of resources.

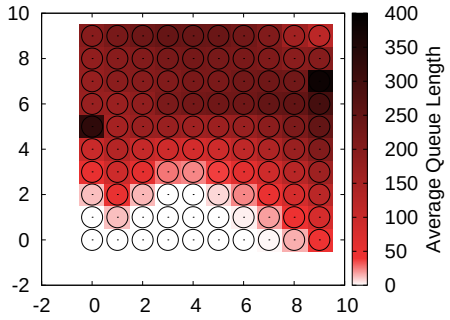


Figure 7.13: Average Queue Length per second during the measured windows interval. Unfairness due to flow dependencies.

when the input rate matrix is such that nodes queuing packets of one flow are also responsible of queuing packets from other flows. This degradation is higher when nodes forwarding a high amount of traffic is close to any of the destinations of the input rate matrix. For instance, in Figure 7.13) source nodes are close to the destinations (sources in $(0, 5)$, $(9, 7)$ and destinations in $(9, 9)$, $(1, 6)$), hence limiting the reception of packets. For instance, the throughput attained by a flow is a 20% lower with respect to the other flow, leading to an WFI of 0.80 for $V = 50$.

In contrast, there are input rate matrices for which for equivalent V parameter, the network experiences a high degree of fairness. We have observed that this corresponds to input rate matrices for which, though nodes can share traffic of both injected flows, the traffic is directed towards the equivalent portions of the network. In Figure 7.11, the sources are located in $(4, 6)$, $(7, 4)$ and the destinations in positions $(9, 3)$, $(9, 2)$ of the grid. The network experiences a WFI of 0.98 for $V = 50$.

7.4 Summary

This chapter provided the theoretical foundations of the basic form of the decentralized backpressure routing policy described in Section 7.2. In particular, the routing problem is tackled with the Lyapunov drift-plus-penalty approach [77], which adds a routing penalty to the queue backlog differential policy (i.e., the original backpressure policy). The drift-plus-penalty approach also incorporates a parameter that determines the importance of the routing penalty. This tunable parameter yields a routing cost within $O(1/V)$ from its minimum value to the detriment of increasing the average queuing delay of the network by $O(V)$.

In this case, we propose to use as routing cost the routing distance to the intended destination. For the sake of practicality and scalability, we introduced a routing penalty based on geolocation information. Thus, the distributed backpressure algorithm backpressure exploits both 1-hop geographic and queue backlog information. The resulting distributed routing policy, which is easy-to-implement and scalable, allows small cells to operate with a single data queue for all destinations, and to take forwarding decisions by only exchanging control information with their neighbors.

We have evaluated at a high level how various network performance metrics (throughput, delay, and fairness) are affected by the value of the weight of the penalty function and finite data queues at small cells, which constrains the results claimed by the theoretical framework. The findings presented in this chapter may help fill the existing gap between Lyapunov drift-plus-penalty function and practical routing based on backpressure for the mesh backhaul.

Note that in the following chapters, we will provide modifications of the proposed routing policy through the use of low complexity heuristics, not studied in this chapter, to adapt to the dynamics of wireless mesh backhauls. In particular, we are interested on common use cases predicted for an small cell wireless backhaul. Among others, we identified as important scenarios those considering the increase of the number of gateways, the increase of the wireless backhaul unreliability, and different topologies namely, regular and sparse deployments.

Chapter 8

Self-Organized Backpressure Routing for the Wireless Mesh Backhaul

The contributions showed in this chapter have been partially published in [15], [6],and [7], and submitted to [8]. This chapter focuses on Lyapunov drift-plus-penalty method, described in previous chapter, for comparison purposes with SoA TNL routing approaches. Further, this chapter describes low complexity heuristics that extend the basic method described in chapter 7. These heuristics are of primal importance to satisfy the practical requirements posed by a wireless mesh backhaul. We analyze the Lyapunov drift-plus-penalty method in a wide range of mesh SC deployments.

Section 8.1 focuses on a dense SC mesh deployment with many-to-one traffic patterns (i.e., traffic patterns are limited to the communication between SCs and a single TNL GW). However, there are some important issues that are not tackled in this analysis, and still are key requirements. First, and this is the most important one, the analysis in section 8.1 considers a many-to-one traffic scenario, however, any-to-any traffic patterns must also be tackled.

Section 8.2 evolves towards more complex dense mesh backhaul scenarios (e.g., any-to-any dynamic traffic patterns). This section contributes with a SON algorithm to calculate the V parameter on a per-

packet basis. Indeed, results shown in previous chapter already suggested the importance of a variable-V algorithm to tackle any-to-any variable traffic patterns in the wireless mesh backhaul. In an all-wireless Network of Small Cells (NoS), it is expected that the input rate matrix could be variable in time [89]. The SON variable-V controller tunes its values in function of the injected traffic rates. The resulting variable-V controller running in every SC finds the best trade-off between the drift and the penalty on decentralized and self-configured way. As a result, evaluation results confirm that the resulting routing algorithm makes the most out of the resources available in the backhaul and reduces the level of manual configuration in the NoS.

Section 8.3 tackles SC deployments with multiple gateways. Dense capacity-oriented deployments must be flexible enough to evolve with traffic demands and must have mechanisms to relieve hotspots. With one single gateway pulling packets from the network, the potential increase of traffic demands can lead to congestion in the NoS. Instead of laying fiber from each small cell to the core, a way to relive congestion is to increase the number of TNL gateways. In this case there is the need, for every given number of SCs, to augment a small cell with a high-capacity link to the core. Therefore, these singulars SCs act as a TNL aggregation gateways towards the core network. Deploying additional TNL gateways increases the global backhaul transport capacity in two ways, namely by introducing additional exit points with high-capacity links towards the core network and by reducing the average number of wireless hops traversed by traffic. However, for a full exploitation of these additional resources, balancing of traffic between TNL gateways must be possible. Therefore,the architecture should allow incrementally deploying and gracefully integrating such new network entities without introducing much additional management and/or operational burden. Evaluation results confirms that scalability can be provided at a low cost, showing a better aggregated throughput and latency compared to SoA TNL schemes. Note that we use both the terms delay and latency interchangeably.

Finally, section 8.4 tackles sparse SC deployments. Under such semi-planned and dense SC deployments, the necessity to provide energy efficiency solutions jointly with the unreliable nature of massively-deployed low-cost SC equipment introduces a high degree of dynamicity, hence requiring a TNL solution that adapts to network topology changes.

These dense deployments of SCs are prone to be highly variable leading, in some cases, to sparse SC deployments. On one hand, the wireless backhaul may be subject to traffic dynamics. Maintaining active all SCs when traffic conditions are light is unnecessarily resource consuming. A possibility is to power off SCs during light operation conditions (e.g., during the night), hence ending up with an appropriate percentage of nodes powered off. Despite these mechanisms can potentially suppose high energy efficiency gains, they also substantially alter the wireless backhaul topology. On the other hand,

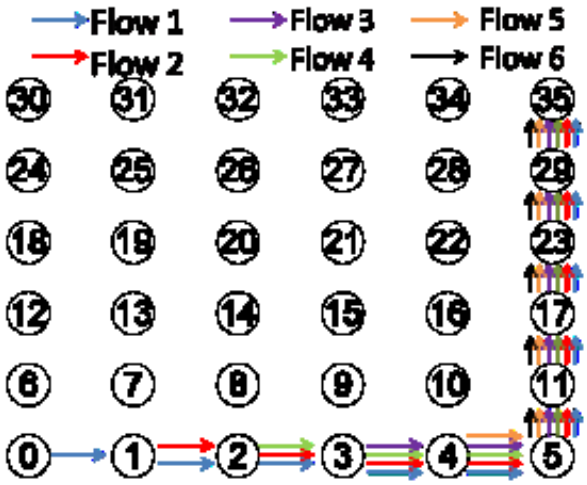


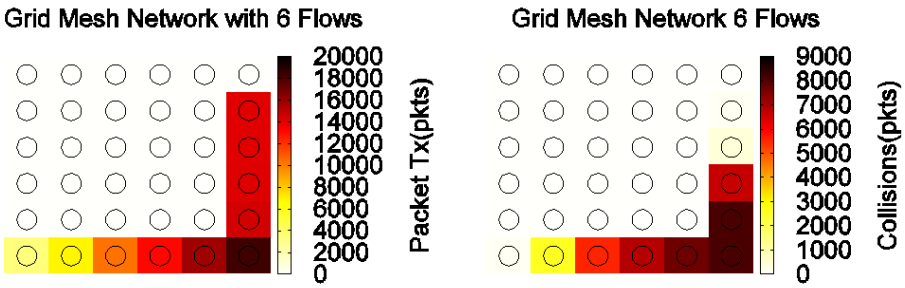
Figure 8.1: Grid Mesh backhaul.

these SC deployments may suffer from node and link failures due to vandalism ambient conditions, or obstacles. Therefore, the challenge is to evaluate whether the the proposed Lyapunov drift-plus-penalty method can provide adaptability to varying wireless backhaul topologies. To this end, section 8.4 evaluates distributed backpressure routing for sparsity deployments. Evaluation results reveal that the proposed solution adapts to dynamic wireless mesh backhaul environments, and significantly improves SoA routing solutions, merely based on geolocation information, hence providing another perspective to deal with backhaul deployments that include dead ends.

8.1 SON Fixed-V Backpressure Routing with a Single Gateway

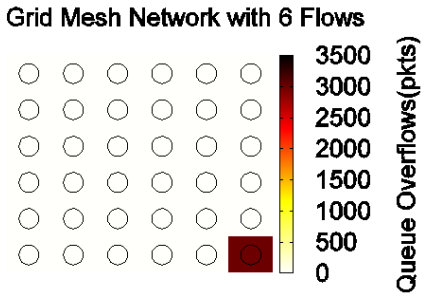
This section studies the routing protocol in a 6x6 grid backhaul with a single TNL GW, which represents a fully-connected dense SC mesh network. To this end, we compare the performance of the proposed method, implementing the Lyapunov drift-plus-penalty function against a SoA tree-based routing protocol using ns-3 [102] simulator. Each node is equipped with one 802.11a wireless card configured at a fixed rate of 54Mbps. All the nodes are assigned to the same channel. Transmission and Carrier Sense Ranges are set to the same value and the wireless channel does not suffer from path loss propagation errors. For instance, node 7 in Figure 8.1 only has direct communication with nodes 1, 6, 8, and 13, and hidden nodes for node 7 are nodes 0, 2, 9, 12, 14, and 19. The number of MAC retries is set to 3. The data queue is set to a fixed length of 400 packets in all the SCs. The SC with TNL GW functionalities is node 35 (i.e., the node in the top right border of the grid depicted in Figure 8.1).

We consider a many-to-one communication pattern. In particular, it consists of unidirectional CBR UDP



(a) Packets Transmsitted

(b) Collisions



(c) Queue Overflows

Figure 8.2: Nature of a tree-based SoA routing protocol.

flows sent from the SCs (i.e., node IDs between 0 and 34 in Figure 8.1) heading for the single TNL GW of the network. The size of the UDP payload is of 1440 bytes.

Prior to start such a comparison, we provide in subsection 8.1.1 a high-level discussion of the advantages and drawbacks of classical routing proposals. The goal is to give some insights on the usual wireless mesh pathologies (i.e., queue overflows and collisions) that cause starvation, and so degradation of network performance metrics. Subsection 8.1.2 presents the performance comparison of our distributed backpressure routing proposal against SoA tree-based protocol.

8.1.1 Sources of Degradation

The above network scenario is used in this section to explain the details of wireless pathologies that lead to node starvation. In a mesh backhaul, node starvation is caused by (unfair) distributed contention and hidden nodes. Starvation in small cell backhauls commonly appears when multiple flows are present in the network. This generates contention and interference, which, in turn, lead to collisions and queue overflows. A properly designed routing protocol should aim to minimize contention and hidden nodes, whilst providing traffic load balancing to avoid congestion. Figure 8.1 and Figure 8.2 illustrate typical network problems that arise when these issues are not taken into account. Figure 8.2(a) shows the path

followed by six UDP flows of 2Mbps originated by nodes 0, 1, 2, 3, 4, and 5 towards the TNL GW (i.e., node 35) using a tree SoA routing protocol, whereas Figure 8.2(b) and Figure 8.2(c) illustrate the number of collisions and queue overflows, respectively.

8.1.1.1 Pathology 1: Node Starvation due to Contention

One cause for starvation is the inability of a node to transmit its queued packets, hence causing queue overflows in its transmission queues. Routing decisions have a clear influence on the contention patterns of the network. For instance, in Figure 8.2(a), one part of the network has the responsibility to route most of the traffic injected to the wireless network (i.e., mainly the rightmost ones). As Figure 8.2(c) illustrates, the problem in this case is that the flows in nodes 1, 2, 3, 4, and 5 are routed towards the TNL GW through a sub-path of the path chosen by node 0. Thus, while advancing in the path from node 0 to the TNL GW, the number of data packets to forward, and so, contention, substantially increases. As a result, the queue length of the nodes involved in routing grows until exceeding the limit of the maximum queue length of node 5 as Figure 8.2(c) depicts.

8.1.1.2 Pathology 2: Node starvation due to Hidden Node

Figure 8.2(b) depicts the number of collisions experienced by each node sending data packets in the network setup of Figure 8.1. As the figure shows, collisions become more frequent as the number of data packets sent by the nodes grows. For instance, node 5 is one of the nodes experiencing more collisions given that it has two hidden nodes (i.e., nodes 3 and 17), and two contending nodes (i.e., nodes 4 and 11) sending a high number of packets. Fewer collisions occur in node 23 compared to node 5 given the lower number of hidden nodes sending data packets, as hidden node 35 is the TNL GW, which is not transmitting but consuming data packets. Consequently, not only the number of hidden and contending nodes influences the number of collisions, but also the amount of hidden and contending traffic present in the network.

8.1.2 Evaluation

This section presents the comparison of backpressure and a tree-based protocol. Since in this chapter we are not focusing on the control overhead required for routing (e.g., building trees), but on data-plane performance comparisons, any protocol that eventually builds a tree would be of use in terms of comparison. Therefore, it is more a comparison of both routing philosophies rather than of specific protocols.

As tree-based protocol, we ported to ns-3 the protocol AODV-ST [106], which stands for Ad hoc On-Demand Distance Vector-Spanning Tree. The first subsection describes the evaluation methodology used to compare both protocols, whilst the second subsection discusses the results obtained.

8.1.2.1 Methodology

Both protocols under evaluation (backpressure and AODV-ST) require the exchange of HELLO messages. The configured HELLO period is 100ms, which translates into a control overhead of 8.8Kbps per node. With regards to the configuration parameters of AODV-ST, note that we use hop count as routing metric for the evaluation of the protocol and not ETX or ETT, as in the original implementation [106], because 1) there are not path loss propagation errors in the network scenario under evaluation, and 2) as showed in [107], ETX suffers from the hidden node effect. Additionally, control packets generated by AODV-ST flooded in the mesh backhaul for proactive maintenance of the tree are not taken into account. Notice that this comparison is unfair to backpressure, which does not introduce any additional overhead besides HELLO packets. As for AODV-ST, there is an off-line process (hence out of the measurement time in the simulation) that computes its necessary end-to-end routes. The reason is that no topology needs to be maintained in the former and decisions are taken on a per-packet basis (not at the path/route level). This is expected to adapt much better to varying conditions. Therefore, unlike for backpressure, the values obtained for AODV-ST must be understood as upper bounds for throughput and lower bounds for delay. The actual value would depend on how often flooding is carried out. To compare both protocols in the 6x6 grid mesh backhaul described above, we simulated two cases, namely single and multi-flow cases. Our interest lied on measuring the throughput, and delay under steady state conditions.

1. **Single Flow Case:** This case compares both protocols when there is a single flow in the network heading for the TNL GW. We evaluated 35 different scenario setups. For each setup, we select a single node out of all the SCs in the backhaul as originator of the data flow. The source node send a UDB CBR flow headed towards the TNL GW. In particular, we tested twenty different rates ranging from 1Mbps to 20 Mbps in steps of 1Mbps. Thus, we generated a total of 700 different simulation scenarios for each of the routing protocols.

2. **Multiple Flows Case:** This case compares both protocols when there are several traffic flows headed for the TNL GW. In particular, each simulation offline selects a set of random source nodes from the backhaul, ranging from 2 up to 12 in steps of 2. For each number of flows injected, we randomly selected 40 different combinations of source nodes, making a total of 240 simulation per routing protocol. Each of these nodes generates a UDP CBR flow towards the TNL

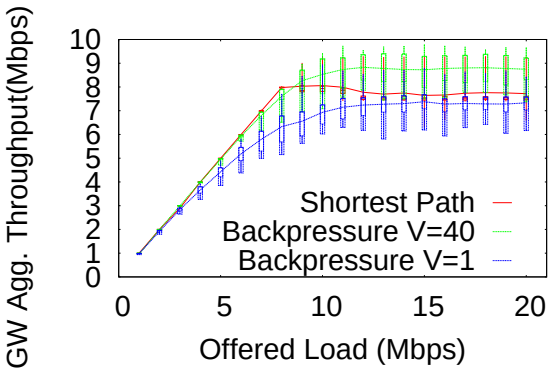


Figure 8.3: Grid Mesh backhaul.

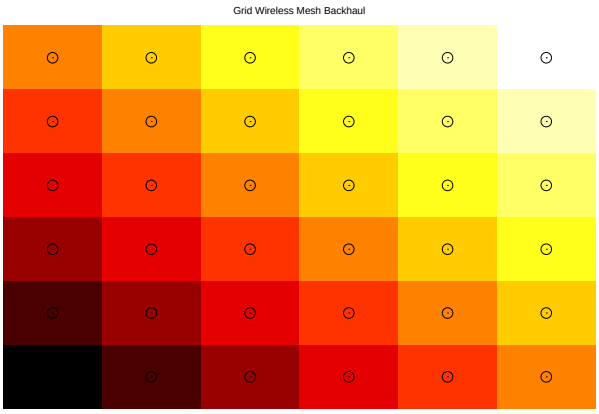


Figure 8.4: Default Gradient Generated by the Cost Function.

GW. Further, such a case considers two subcases regarding traffic rate intensity. In the first one, all the UDP flows injected have a fixed rate of 2Mbps, whilst in the second one each UDP flow has a random rate between 1 and 10Mbps. Thus, the chances of saturation in the later case with fewer flows increase compared to the former case. The goal is to study the performance of each routing protocol when the offered load is within and above the network capacity region under a different number traffic flows.

8.1.2.2 Results for the Single Flow Case

Figure 8.3 presents the aggregated throughput obtained with backpressure when using two different V parameters against the performance attained by the tree-based one. Aggregated throughput is the sum of all rates achieved by each of the individual flows when they reach the TNL GW. Specifically, we compare the backpressure method configures with two different V values, 40 and 1 being the values (V=1 the

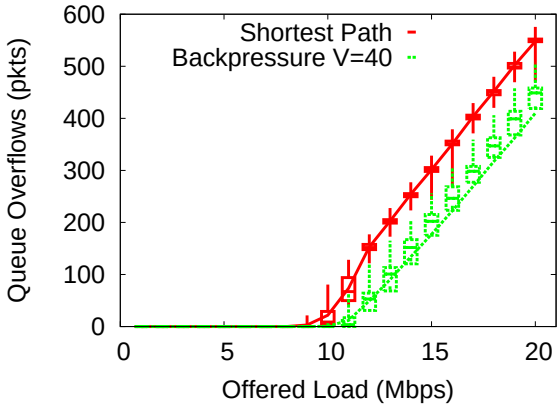


Figure 8.5: Single Flow Case. Queue Overflows.

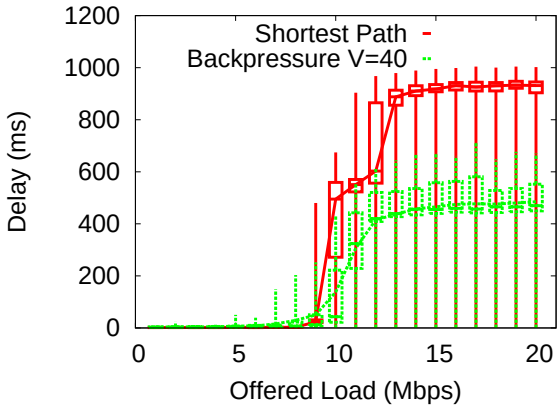


Figure 8.6: Single Flow Case. Delay.

value used in [76]) between neighbors that have different hop-count to the TNL GW. Figure 8.3 depicts how backpressure with a V parameter equal to 1 achieves an aggregated throughput that is substantially lower than that obtained with a V parameter equal to 40 as the offered load increases. The reason of this behavior is that the flow experiences a higher number of packet losses whose root cause is routing loops. The rest of this section configures the V parameter to 40, as it shows a better behavior in terms of throughput and latency. The key for this behavior is the default gradient generated towards the TNL GW. Figure 8.4 shows the resulting trend of pushing traffic towards the TNL GW, caused by the V parameter. The heatmap represents the gradients generated by the cost function, based on geolocation information, and assuming there is no traffic in the mesh backhaul. Note that the relative gradient between neighbor nodes increases as the V parameter increases, since the geolocation component of the protocol gains importance.

In terms of throughput, for low offered loads, both protocols show a similar behavior. Interestingly, at an offered load of around 6Mbps, AODV-ST is able to service all the offered load, while backpressure starts to experience a slight throughput degradation. Backpressure on top of a CSMA access layer increases the number of collisions due to the higher degree of traffic distribution, which implies more packets contending for the medium at the same time. Thus, backpressure experiences a higher number of packet losses, tagged as wireless losses. Recall that a data packet is discarded when it has exhausted the MAC retry limit, which is set to 3 in our evaluation. However, for medium and high (i.e., $\geq 8Mbps$) offered loads, backpressure presents substantial improvements in terms of aggregated throughput regarding AODV-ST. Figure 8.5 shows that AODV-ST begins to experience a high number of queue overflows. The use of a single shortest path jointly with the substantial increase of queue drops in AODV-ST yield fewer contention at the access layer. Despite AODV-ST suffers from fewer CSMA contention, the use of a reduced number of resources (i.e., nodes) in the backhaul substantially penalizes the performance in terms of throughput compared to backpressure.

Figure 8.6 shows that the performance in terms of delay of both backpressure and AODV-ST is similar for low loads. In fact, both queue backlogs and path length used by both protocols coincides under low loads. This is a remarkable aspect for backpressure since under non congested traffic conditions a shortest-path based routing protocol such as AODV-ST (shortest path based) can be considered an optimal routing policy. For an offered load of 8Mbps, AODV-ST experiences a substantial increase of delay, because of the increase of the average queue backlog (see Figure 8.5). In contrast, backpressure does not suffer such increase, since queue occupancy at nodes is on average lower than that of AODV-ST. Backpressure starts increasing the average queue backlog of the grid mesh backhaul when all resources in such backhaul are occupied. Figure 8.5 demonstrates that this happens when the workload is of

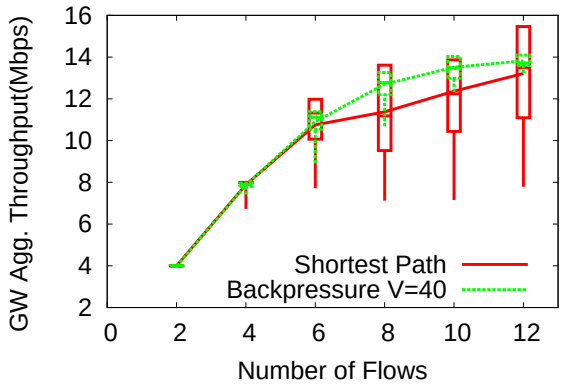


Figure 8.7: Throughput under Multiple Flows at a Fixed Rate.

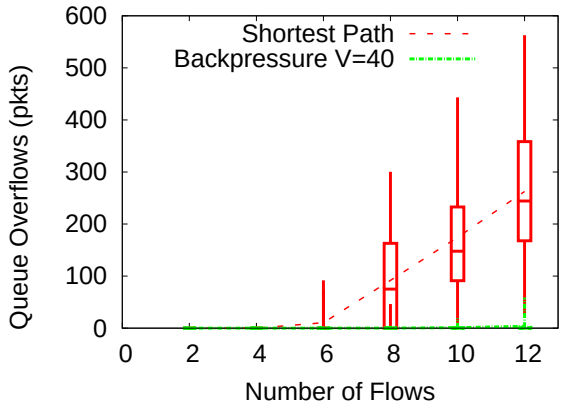


Figure 8.8: Queue Drops under Multiple Flows at a Fixed Rate.

10Mbps, whilst AODV-ST suffers from queue drops at lower offered loads (i.e., 8Mbps). Therefore, we can conclude that the load distribution of resource consumption over the backhaul leads to substantial improvements in terms of throughput, and delay.

8.1.2.3 Results for the Multiple Flow Case

In terms of throughput, for both cases backpressure obtains, in general, higher aggregate throughput at the TNL GW compared to AODV-ST, especially in the case that flows are generated at a random rate. The main observations obtained from both cases in terms of throughput follow.

- Figure 8.7 depicts the throughout obtained when the traffic flows are generated at a fixed rate. Backpressure average throughput values show a better performance than those of AODV-ST. AODV-ST solely takes routing decisions hop-by-hop based on minimizing the hop distance to

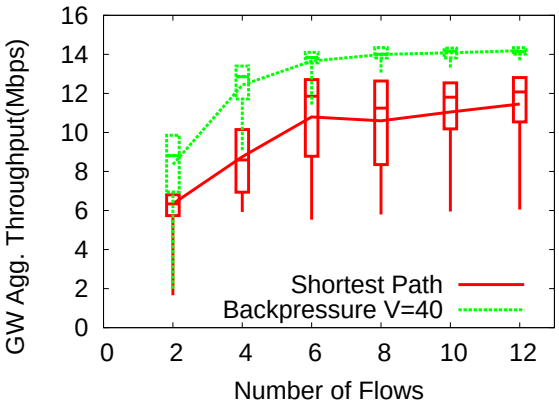


Figure 8.9: Throughput under Multiple Flows at a Random Rate.

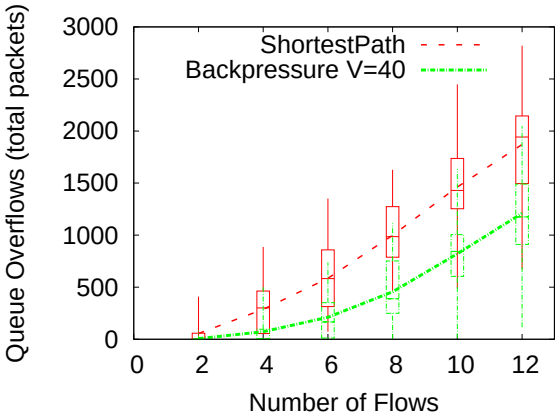


Figure 8.10: Queue Drops under Multiple Flows at a Random Rate.

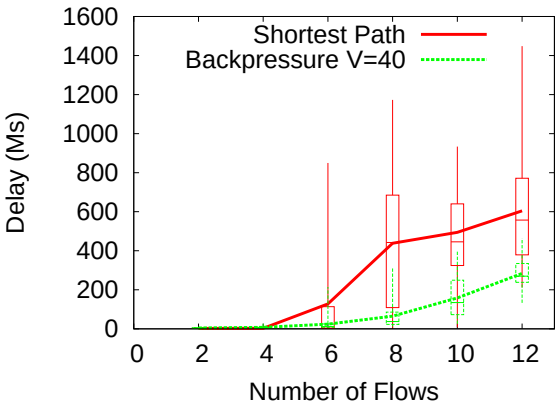


Figure 8.11: Delay under Multiple Flows at a Fixed Rate.

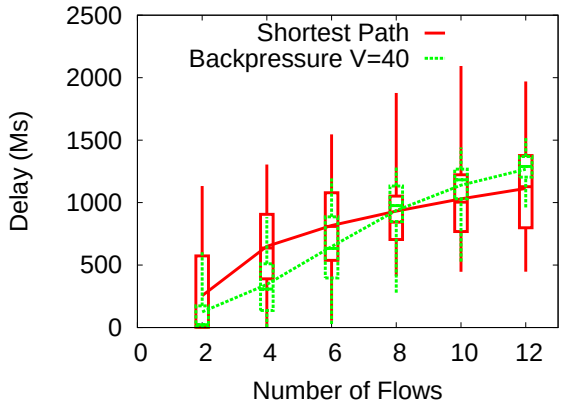


Figure 8.12: Delay. Multiple Flows at a Random Rate.

the TNL GW, hence it has a null degree of traffic distribution per se. Yet, for 12 flows, the 95th percentile in Figure 8.7 clearly depicts that AODV-ST can obtain better results than backpressure for some combinations of source SCs. The reason for this is that AODV-ST can find routing paths that achieve a certain degree of aggregated traffic distribution in terms of resource consumption because of the precise distribution of the random sources. Further, in such particular cases the AODV-ST trend is to serve traffic flows close in number of hops to the TNL GW, while practically dropping the traffic injected from the farther ones. Despite backpressure shows lower maximum throughput values for these precise cases, it also shows a stable trend keeping slight variations of throughput. Backpressure evenly distributes traffic, by providing load distribution of resource consumption at nodes, hence substantially decreasing the queue backlogs in the backhaul. Figure 8.8 demonstrates that backpressure keeps lower queues compared to AODV-ST. In fact, up to the injection of 12 traffic flows queue drops experienced by backpressure are not particularly significant.

- Figure 8.9 shows the throughput obtained in the case in which traffic flows are generated at a random rate. Backpressure outperforms AODV-ST in all the tested scenarios. We observe that with a lower number of traffic flows the backhaul arrives to saturation compared to the previous case, since each traffic flow can send up to 10Mbps. Therefore, the input rate is in average much higher than that tested in the previous case. For 2 traffic flows, ns-3 simulation results show that backpressure and AODV-ST behave similarly under light loads. Interestingly, as traffic rate increases with the number of flows we observe that backpressure shows a stable behavior, whereas AODV-ST show a highly variable and lower throughput. These significant since it is maintained even under limit circumstances (i.e., when the injected traffic rate is clearly out of the capacity region). Figure 8.10 demonstrates that backpressure keeps lower queues compared to AODV-ST,

yet incurring into queue overflows with a lower number of traffic flows compared to the previous case due to the higher input rates per flow in average.

We now investigate the properties of backpressure in terms of delay. The main observations obtained from both cases in terms of throughput follow.

- Figure 8.11 shows the evolution of delay with the increase of traffic flows generating at a fixed rate of 2Mbps. For up to 4 flows, making an aggregated offered load of 8Mbps, both protocols experience similar levels of delay. When the the number of traffic flows is of 6 flows, AODV-ST starts experiencing a huge increase of latency due to the congestion experienced in the backhaul. Figure 8.8 depicts that the increase of queue drops starts precisely when the offered load is of 6 flows. As the number of traffic flows increases, AODV-ST worsens given that it always uses the same paths to reach the TNL GW. Yet backpressure starts experiencing a slight increase of average queue backlogs when the traffic load is of six flows, it is still able to serve the traffic without experiencing queue drops. Backpressure starts distributing packets, hence causing a slight increase of delay due to the use of longer paths. When the traffic load is of 8,10, and 12 flows both routing protocols enter into saturation. In saturation, backpressure is able to serve more traffic, while still showing lower latencies values compared to AODV-ST. Interestingly, Figure 8.8 demonstrates that the saturation level attained by backpressure is lower than that of AODV-ST.

- Figure 8.12 shows the evolution of delay with the increase of traffic flows generating at a random rate ranging from 1Mbps up to 10Mbps. Under this network setup, both protocols have to deal with saturation conditions practically with 2 traffic flows since they can't handle such offered load. In this case, backpressure attains lower delays for 2,4,6 and 8 traffic flows, increasing the delay with the number of flows due to the increase of queuing latency and CSMA contention. As the number of flows increases to 10 and 12 traffic flows, network delay deteriorates even more than that of AODV-ST. This is due to the attempt of backpressure of handling more traffic by distributing traffic on a medium access layer suffering from CSMA contention.

A common aspect to emphasize in all the simulation results shown in Figures 8.7, 8.11, 8.9, and 8.12 is that backpressure is able to achieve similar values in terms of throughput and delay, independently of the set of source nodes chosen to send traffic to the TNL GW. This can be observed by focusing on the much smaller size of the boxplots for backpressure. In contrast, AODV-ST suffers from higher throughput and delay variability. Consequently, backpressure seems to be fairer than AODV-ST in terms of throughput at the TNL GW and end-to-end delay, as throughput and delay values obtained are similar

and independent of the set of sources chosen to send data traffic and their distance (i.e., number of hops) to the TNL GW.

8.2 SON Variable-V Backpressure Routing

In chapter 7, we proposed a scalable and distributed backpressure routing strategy for the TNL based on the Lyapunov drift-plus-penalty framework [77]. In this method, a tunable non-negative parameter, denoted as V, is used to trade off between the Lyapunov drift and the penalty function. In such a framework, minimizing the Lyapunov drift has as goal pushing queues to a lower congestion state, hence making the network stable. On the other hand, the goal of the penalty function is to make the network evolve towards the optimal values of the chosen network objective metric. For theoretical centralized schemes, it was proven [77] that one may get arbitrarily close to the optimal value of the metric as $O(1/V)$ at the cost of making queue backlogs increase as $O(V)$. Further, the main issue addressed in chapter 7 is the impact of a given V value on the target performance metrics, since the optimal V value to assign in each node is not known a priori. In light of the observations in 7, we inferred some statements regarding the assignment of the value of V as a function of certain stable network conditions (e.g., constant offered load) in the TNL. However, a TNL must usually cope with sudden changes in its injected traffic load, as well as in the wireless backhaul topology characteristics. Different nodes may experience different loads at the same instant in a wireless backhaul topology subject to dynamic workloads. This naturally leads to consider the adjustment of the V parameter independently at every node to meet the unpredictable traffic demands. In what follows, we address how to independently calculate the V parameter on each SC in the backhaul.

The goal of this section is to answer the following question: *Given a TNL with varying and uneven traffic demands, can each node adjust its V parameter to cope with high queuing delays and/or queue drops, and still get as close as possible to the optimal value of the target performance metrics?* Subsection 8.2.1.1 shows with a simple example how the design of an optimal variable-V algorithm emphasizing the penalty function while avoiding queue drops is hard.

Subsection 8.2.2 introduces an algorithm that computes the V value using the information received by the periodical HELLO messages periodically exchanged between nodes. This variable-V algorithm estimates the optimal trade-off between the Lyapunov drift and the penalty function at each node. The goal of the variable-V algorithm for taking routing decisions is to weight more the penalty function whenever queue congestion around the 1-hop neighborhood permits it. Further, the variable-V algorithm can be distributed, since it merely requires queue backlog and position information of 1-hop neighbors.

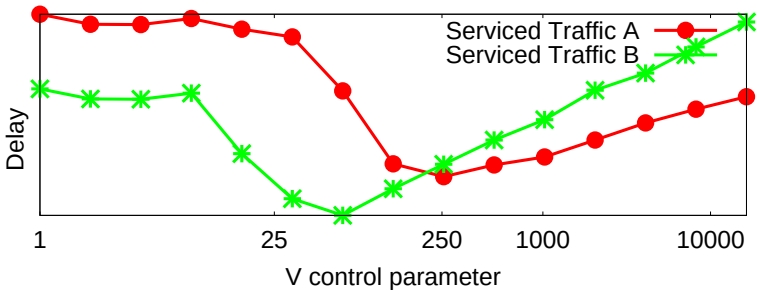


Figure 8.13: Impact of V on Delay.

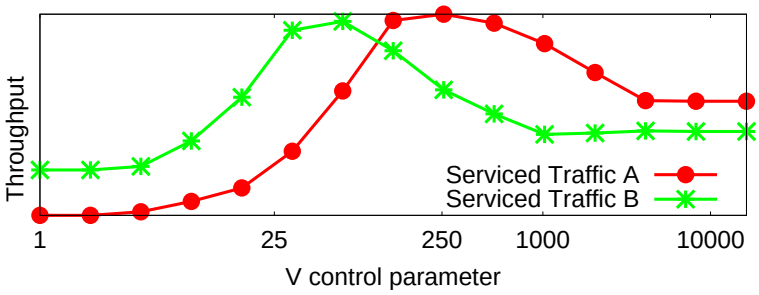


Figure 8.14: Impact of V on Throughput.

The former is used to either relieve congestion in the TNL by decreasing the V parameter or to quickly approach the target network performance metrics by increasing the V parameter.

Although the above variable-V algorithm brings performance benefits, we demonstrated that it has scalability issues with the number of traffic flows injected in the mesh backhaul. In particular, its performance deteriorates when there are several traffic flows in the TNL. Firstly, the level dynamicity may even be higher than that attained by V periodically computing its value. Secondly, the V value is equivalent no matter the specific data packet, and such data packets may have different characteristics (e.g, a different destination, and a different number of hops traversed). For these reasons, we compute the V parameter on a per-packet, rather than periodically on a per-HELLO basis. In particular, the per-packet V calculation introduces a characteristic of the data packet itself: the TTL field in the IP header of the data packet. We leverage on the TTL to estimate the urgency of the packet to the reach the destination. Thus, this variable-V algorithm has an inherent component of current routed packet.

In summary, this section extends the distributed Lyapunov drift-plus-penalty routing protocol presented in chapter 7 by implementing and evaluating the variable-V algorithm with the ns-3 [102] simulator. In general simulation results show that the variable-V routing policy avoids queue overflows while improving throughput and delay while scaling with the number of flows traversing the TNL.

8.2.1 The problem with Fixed-V routing policies

This subsection presents some of the issues that appear with fixed-V routing policies, justifying the need for a variable-V routing policy. To illustrate such issues, Figures 8.13 and 8.14 show the curves of delay and throughput for two different traffic flows injected in the network. This wireless mesh backhaul performance metric is measured when serving a constant traffic pattern able to saturate node queues. In this particular example, each node has an infinite buffer size for the sake of illustrating the effect of high queuing on wireless mesh backhaul performance metrics. Moreover, there is variable amount of background traffic injected in the network.

Throughput and delay degradation can be observed for small values of V, as decisions are strictly made on pushing queue backlogs to a lower congestion state (i.e., packets are not steered towards the intended destination). We also observe a range of values that yield the best possible performance metrics (i.e., low end-to-end delay and high throughput) when V is increased. Essentially, high values of V imply distance-to-destination minimization (instead of resource distribution) in the penalty function. However, once a certain value of V is reached, performance metrics start degrading due to high queuing backlogs, thus causing queuing delays. Figures 8.13 and 8.14 present a range of low values of V experiencing bad performance. For both traffic cases under evaluation, we also observe a range of values with the best possible performance metrics when increasing V (i.e., low end-to-end delay and maximum throughput). One interesting remark is that throughput and delay share a common range of V values. However, once a certain value of V is exceeded, performance metrics degrade again due to high queuing backlogs. Another relevant aspect illustrated by these figures is that the specific range of V values experiencing low delays and high throughput is different depending on the traffic vector injected in the network.

Finding a constant value of V that is valid for all nodes at the same time, whilst making the network operate in the optimal set of V values illustrated in Figure 8.14 and 8.13 is unfeasible, in general. On the one hand, wireless mesh backhaul traffic demands may be different in different areas of the network. Furthermore, the traffic demands can highly vary over time. Actually, certain nodes may need a very low V value in order to meet high local traffic demands whilst maintaining queue backlogs under control (e.g., nodes around the traffic generators). Other nodes may weight more the penalty function, as queue backlogs are not that critical (e.g., nodes far from the flow of traffic). This suggests that different values of V may be needed at each node. On the other hand, as shown in Figures 8.14 and 8.13, increasing the V parameter emphasizes the penalty function targeting network objective while not pushing packets to experience high queue occupancy. One may notice that effect caused by infinite queue backlogs will not be an issue in a practical wireless mesh backhaul, as nodes will have finite buffer sizes, in general.

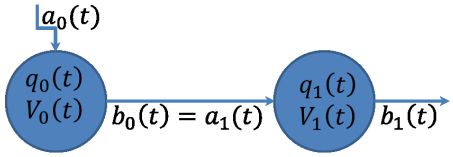


Figure 8.15: Simple two-node WiFi mesh network.

In this case, one may want to minimize queue drops. Therefore, *the value of V at every node should weight the penalty function as much as possible without generating queue overflows.*

8.2.1.1 An Illustrative Example

In this section, we consider the simple two-node mesh backhaul depicted in Figure 8.15 to illustrate the issues that appear when adjusting the V parameter in any given node. We focus our attention on the evolution of queue Q_1 as a function of $V_0(t)$. As explained in previous chapter, Equation (7.11) determines whether node 0 transmits packets to node 1 or not.

More specifically, node 0 transmits packets to node 1 when $w_{01} > 0$. We identify two cases for which $w_{01} > 0$ as a function of $p(0, 1, d)$. When node 1 is farther from destination d than node 0, $p(0, 1, d) = +1$. Then $-V_0(t)p(0, 1, d) < 0$, since we are focusing on values of $V_0(t) > 0$. This means that for the weight to be non-negative, $Q_0(t) - Q_1(t) > V_0 > 0$. Note that, in this case, the routing control policy is taking decisions targeting the minimization of the Lyapunov drift (i.e., minimization of the queue backlog differential), as opposed to approaching the destination. As a consequence, the scheme already takes the best possible routing decisions to minimize queue drops.

On the other hand, when node 1 is closer to destination d than node 0, $p(0, 1, d) = -1$. Then, $-V_0(t)p(0, 1, d) > 0$, since $V_0(t) > 0$. In this case, even if the queue differential is smaller than 0, the weight could still be positive, hence allowing routing decisions focusing on getting closer to the destination, as opposed to minimizing the Lyapunov drift. Furthermore, if $Q_0(t) - Q_1(t) < 0$ and $Q_0(t) - Q_1(t) + V_0(t) > 0$, queue overflows may occur, as queue backlog is not relevant in this case. These are the type of queue drops that our variable-V algorithm (see Section) tries to avoid. Therefore, the goal is to avoid queue overflows in node 1 by controlling the transmissions of node 0 under the above circumstances. To further illustrate this issue, let us assume $a_1(t) = b_0(t)$. The condition to be fulfilled in node 1 for avoiding queue drops follows:

$$Q_1(t) + b_0(t) - b_1(t) < Q_{MAX}. \tag{8.1}$$

Let H be the possible range of served packets $b_0(t)$ (i.e., the number of packets transmitted by node 0

to node 1 during the time slot interval $[t, t+1)$), hence

$$H(b_0(t)) = [0, min(Q_0(t), Q_0(t) - Q_1(t) + V_0(t))]. \tag{8.2}$$

In fact, in a mesh backhaul, $b_0(t)$ is affected by the network conditions of the medium and how CSMA/CA reacts to them. In turn, $b_0(t)$ is also affected by $b_1(t)$, since both nodes may contend when simultaneously accessing the medium for transmission. Therefore, under a variable-V scheme, $V_0(t)$ also depends on the network conditions of the medium. Let us further assume that CSMA/CA allows transmitting $Q_0(t)$ packets during time slot $[t, t+1)$. If $Q_0(t) > Q_0(t) - Q_1(t) + V_0(t)$, and according to Equation 8.2, $b_0(t) = Q_0(t) - Q_1(t) + V_0(t)$. This means that, in this case, $V_0(t)$ limits the number of packets transmitted from node 0 to node 1 while $w_{01}(t) > 0$. For avoiding queue drops (Equation 8.1), $V_0(t)$ is then constrained by the following expression:

$$V_0(t) < Q_{MAX} - Q_0(t) + b_1(t). \tag{8.3}$$

Thus, the maximum number of packets that can be transmitted from node 0 to node 1 is known. Even in this simple example, the resulting expression of $V_0(t)$ is a function of the number of served packets $b_1(t)$ in node 1. This means that the number of packets transmitted during the interval $[t, t+1)$ in node 1 should be known in node 0 at the beginning of the same time slot, which is in general not possible in such a dynamic environment where nodes take distributed forwarding decisions.

8.2.2 Periodical-V on a per-HELLO basis

The problem of the distributed calculation of the variable value of V at each node may be formulated as follows. Note that we assume a stochastic network such as that described in the previous chapter 7. The queue backlogs of the network are determined by the following dynamics:

$$Q_i(t+1) = Q_i(t) - b_i(t) + a_i(t); 1 \leq i \leq r \tag{8.4}$$

At time instant t, node i observes $Q_i(t)$ and all $Q_j(t)$, for $j \in J$ (where *J* is the neighbor set of node *i*), to calculate the value $V_i(t)$ so that the penalty function is emphasized while avoiding queue drops at any node $j \in J$. And this is done by taking into account the weights of all links between i and its neighbor set, i.e.,

$$w_{ij}(t) = \Delta Q_{ij}(t) - V_i(t)p(i, j, d), \tag{8.5}$$

where d is the destination node for the packet being forwarded. Therefore, $V_i(t)$ determines the number of serviced packets $b_i(t)$, which in turn determines $Q_i(t+1)$. Therefore, in our case, at every time slot t, every node i observes the 1-hop neighbor queues and selects a value for $V_i(t)$. The $V_i(t)$ value must

be such that the penalty function is emphasized while avoiding queue drops due to exogenous arrivals. Note that in this chapter we focus on avoiding queue drops due to exogenous arrivals. Queue drops due to endogenous arrivals are out of the scope of this chapter.

- The distributed Variable-V Algorithm Here we describe the distributed variable-V algorithm, and discuss some issues with regards to its practical implementation. The goal of the variable-V algorithm is to increase $V_i(t)$ as much as possible (i.e., to increase the importance of the penalty function) while not leading to queue drops in the queues of the nodes.

  On the other hand, we showed in Section 8.2.1.1 that there is a relation between the appropriate value of $V_i(t)$ and the number of packet transmissions of node i. The underlying idea behind our scheme lies in the fact that we consider $V_i(t)$ as the maximum number of packets that could potentially be greedily transmitted from node i to one of its neighbors j without exceeding Q_{MAX}. In this case, greedily means that Lyapunov drift minimization is not taken into account when sending traffic. To adjust $V_i(t)$, we estimate an upper bound of the expected maximum queue backlog in the 1-hop neighborhood at time slot $t+1$. This estimation is based on 1-hop queue backlog information at time slots t and $t-1$, being $t>0$. More specifically, we propose the following distributed algorithm:

  Distributed Variable-V Algorithm:

  If $t=0$, $V_i(t) = Q_{MAX}$. At time $t = 1, 2, 3...$, let

  $$V_i(t) = \max(0, \max(Q_{MAX}, Q_{MAX} - \max \Delta Q_j(t) - \max Q_k(t))); k, j \in J \tag{8.6}$$

  where $max(Q_k(t))$ is the maximum of all backlogs of nodes $\in J$ (i.e., the set of 1-hop neighbors of node i), and $max(\Delta Q_j(t))$ is the maximum differential $Q_j(t) - Q_j(t-1)$ experienced by 1-hop neighbor queues of J between time slots t and $t-1$. The sum of $max(Q_k(t))$ and $max(0, Q_j(t) - Q_j(t-1))$ provides an upper bound (i.e., worst case) of the maximum queue backlog a neighbor of i may experience in time slot $t+1$. The difference between Q_{MAX} and the estimated congestion at time slot $t+1$ in terms queue backlog determines the value of $V_i(t)$ for time slot t. Figure 8.16 is a representation of the worst case queue backlog of the neighbor set of i expected at time slot $t+1$. Since the value of $V_i(t)$ represents the maximum number of allowed greedy transmissions from node i during a given time slot, $V_i(t)$ is also a key component in the estimation of the future more congested queue backlog in Figure 8.16.

- Practical Considerations The specific duration of the time slot determines the efficiency of the proposed distributed routing algorithm. More specifically, the algorithm is assuming the knowledge of past queue backlog differential information (i.e., $Q_j(t) - Q_j(t-1)$) to estimate future queue

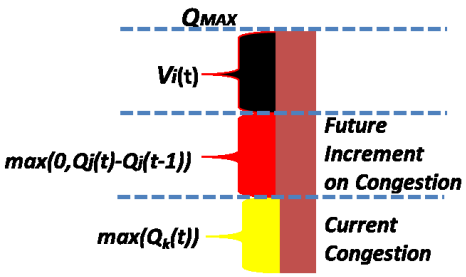


Figure 8.16: Maximum Queue Backlog estimation.

backlog differentials. In other words, it is implicitly assuming that there are no abrupt changes in the differential of queue backlogs with neighboring nodes, and so, in the offered load in the mesh backhaul during time slot $[t, t+1)$. Thus, the smaller the duration of the time slot $[t, t+1)$, the faster V would adapt to varying conditions.

8.2.3 Evaluation of the Periodical-V on a per-HELLO basis

The first subsection describes the evaluation methodology used with the ns-3 [102] simulator to evaluate the variable-V algorithm. The second subsection outlines main observations.

8.2.3.1 Methodology

We consider a single-radio single-channel mesh backhaul with ideal CSMA/CA. It is ideal in the sense that it does not generate losses due to hidden nodes or propagation errors. Each node transmits HELLO messages at a constant rate of 10 packets per second. Regarding the maximum allowed queue backlog Q_{MAX} configured in all nodes, and since there is a significant disparity in the buffer size used by commercial WiFi cards, we opt for the most common buffer size of the MadWiFi legacy drivers (i.e., 200 packets) [108]. On the other hand, the duration of the time slot, which determines how often the value of V at each node is re-calculated is equal to that specified for the transmission of HELLO messages (i.e., 100ms).

To evaluate the efficiency of the variable-V algorithm, we set up an experiment in which there are different changes in the offered load to the mesh backhaul. The first change in the offered load occurs at instant 5s, at which a new UDP CBR flow from R to node D is injected (see Figure 8.17). In fact, F_1 injects UDP CBR traffic up to instant 80s in a grid mesh network. In addition, we launch another unidirectional CBR data flow F_2 originated in C with destination D lasting from instant 20s to instant 60s. Therefore, at instant 20s, in the ns-3 simulation there is an additional change in the offered load

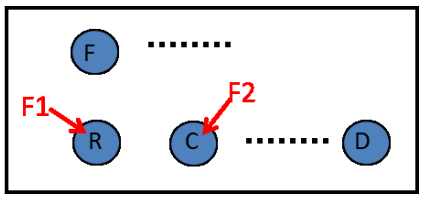


Figure 8.17: Network Scenario.

injected to the network. Furthermore, F_1 and F_2 saturate the queues of the nodes originating the flows (i.e., nodes R and C in Figure 8.17).

8.2.3.2 Observations on the V controller

Up to instant 5s, the offered load to the mesh backhaul is light (i.e., HELLO packets). The V controller, working individually in all the nodes, is adjusted to the maximum value of $V(t)$ (i.e., $V(t)$=Q_{MAX}). Therefore, the penalty function (and so geolocation information) prevails when taking routing decisions. Significant changes in the offered load occur at instants 5s, 20s, 60s, and 80s. We map them to four different states in the network: S_0, S_1, S_2, and S_3, respectively. These states result in different transitions of the V parameter in any node of the mesh network under evaluation. Figure 8.2.3.3 shows the evolution of the V parameter in node R.

S_0: At instant 5s, node R starts experiencing a substantial increase of its data queue leading to queue drops for routing policies with high fixed-V parameters (i.e., V=200, V=150, and V=100). However, as shown in Figure 8.2.3.3, the variable-V algorithm is able to react to this change in the offered load of the network by decreasing V in node R. The variable-V algorithm in R detects an abrupt change in queue backlog from node C, and so, it decreases its V parameter due to the accumulation of packets in the queue of node C. As a result of the initial decrease due to the previous change, and its subsequent traffic distribution to nodes F and C, the algorithm increases the V parameter in node R, hence giving more emphasis to the penalty function without causing queue drops in node C.

S_1: At instant 20s, node C injects an additional flow. As shown in Figure 8.2.3.3, node R reacts by adjusting the V parameter. Since C is advertising a full queue, V is decreased down to zero in neighboring nodes (see Figure 8.2.3.3 between instants 5s and 20s), hence operating at the maximum possible degree of queue load balancing for avoiding queue drops. We observe that packets are load balanced until they find a less congested zone in the network (i.e., with a higher value of V in the nodes), and so, they can be more greedily directed towards the destination.

S_2: At instant 60s, node C stops flow F_1. Consequently, node C starts advertising a non-full queue. Neighbors of C react by increasing the V parameter so that the penalty function is made more relevant when taking routing decisions (see Figure 8.2.3.3 between instants 60s and 80s).

S_3: Finally, at instant 80s, node R stops flow F_0. In this case, all the nodes in the mesh backhaul adjust V to Q_{MAX}.

8.2.3.3 Impact on Network Performance Metrics

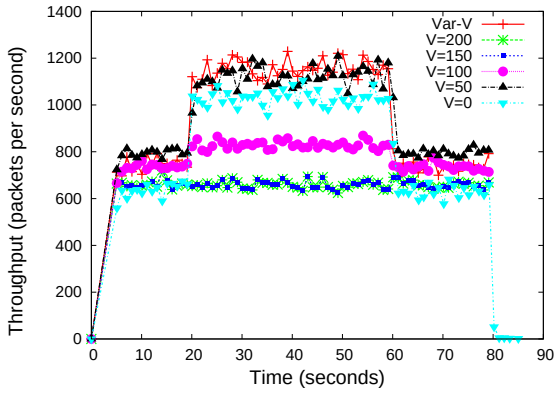


Figure 8.18: Achieved Throughput for fixed and variable-V SON routing schemes.

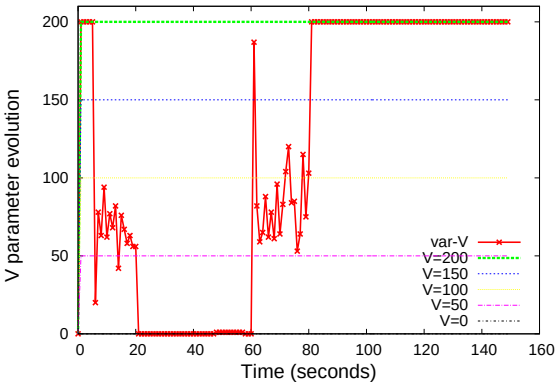


Figure 8.19: V Parameter Evolution on node R.

Figure 8.18 shows the throughput achieved by both fixed-V and variable-V routing policies. In this graph, we can observe that the proposed variable-V algorithm obtains similar results to those obtained by the best instance of the fixed-V policy (i.e., V=50). Therefore, without an a priori knowledge of the optimal V configuration, the V parameter is appropriately adjusted in each node for this network setup. Figure 8.20 presents boxplots of the packet delay of both fixed-V and variable-V routing policies. For each configuration setup in the X-axis, we also provide the Packet Delivery Ratio (PDR). We first observe that the variable-V scheme has a lower maximum delay compared to the rest of setups and

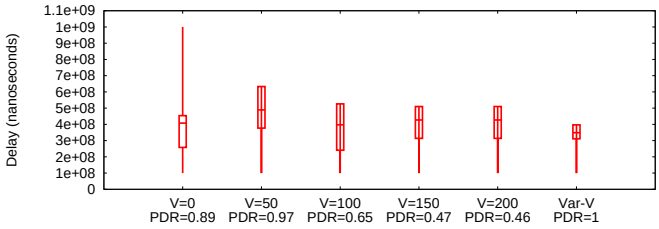


Figure 8.20: Delay.

that it also reduces the variability of the end-to-end delay experienced by data packets. Moreover, all data packets arrive at the intended destination (i.e., PDR=1). Thus, there are zero queue drops due to exogenous arrivals in nodes. In contrast, high delays appear in fixed-V setups due to high queuing delays when V is high, as well as due to the lack of greediness towards the destination when forwarding packets when V is low. Additionally, packet losses due to queue drops grow as we increase the V parameter, or when there is a lack of greediness when forwarding for low values of V.

This section describes the distributed variable-V algorithm, and discusses some issues with regards to its practical implementation.

8.2.4 Variable-V controller on a per-packet basis

The previously shown variable-V algorithm recomputed the V parameter for every HELLO message received from 1-hop SC neighbors (i.e., on a per HELLO message basis). This approach presents certain limitations. First, the V parameter denoting the trade-off between minimization and proximity and destination is equivalent, no matter the data packet being currently routed. This can lead to drastic performance degradation results as we will show in next subsection. Second, congestion conditions may change faster than the periodic HELLO window interval due to sudden traffic changes (e.g., injection or more traffic flows in a given instant). Consequently, the calculation of the V parameter on a per HELLO basis could be biased regarding the actual congestion conditions.

We propose two major changes in the calculation of the variable-V algorithm. On the one hand, there is an additional algorithm that corrects the algorithm periodically calculated on a per HELLO message basis per each data packet. On the other hand, there are some changes in the computation of the variable-V algorithm on a per HELLO message basis that take into account the particularities of the currently routed data packet. Recall that, so far, the recalculation of the V value at time slot t in each SC was conducted periodically, at every timeslot. In practice, the duration of a timeslot corresponds to the time required to receive HELLO messages from all the neighbors of a SC. Precisely, every time a SC receives

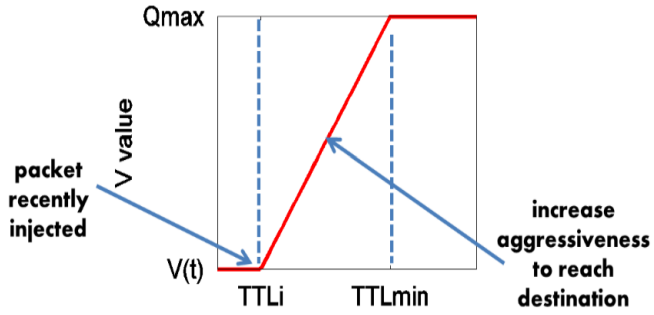


Figure 8.21: The lower the TTL, the lower the degree of load balancing allowed for the packet being routed.

a set of HELLO messages from all its neighbors with their associated queue backlog information, the SC exploiting the variable-V mechanism presented in previous section, self-regulates the V parameter. Actually, the new V value at a SC remains valid until it receives a new set of HELLO messages from all its neighbors. Therefore, all the weights calculated for taking routing decisions within (t, t+1) use the same V value. This means that a considerable bunch of packets intended to be sent by every SC during interval (t, t+1) are routed using the same trade-off between the Lyapunov drift and the penalty function, even though they could have a different "necessity" to reach the destination. Note that a SC can keep in its queue data packets corresponding to different flows. And these packets may have different needs in terms of end-to-end delay. Further congestion conditions could considerably change within the given HELLO period interval, leading to a biased estimation of local congestion conditions.

The main intuition behind the increase of the V parameter using the TTL field in the IP header can be illustrated with an example. For instance, consider two packets a and b accumulated in a data queue in a SC i. Assume that the number of hops traversed by packets a and b is quite different. While packet a traversed n>0 hops till SC i receives it, SC i generates packet b. Our mechanism proposes to forward data packets a and b with a different V parameter. Specifically, the V parameter should be higher in the case of packet a than in the case of packet b. In particular, the increase of the V parameter depends on the number of hops traversed by packet a.

We propose to calculate the V parameter on a per-packet basis. First, we propose to calculate local congestion conditions on a per-packet basis. Local congestion conditions require 1-hop queue backlogs, which are directly unaccessible. Even though each SC cannot have instantaneous information of 1-hop neighbors queues, they have access to the local queue backlog. Thus, accessing this parameter on a per-packet basis can give a more accurate estimation of current 1-hop congestion conditions. Further, this parameter in a shared WiFi medium can be used as an estimation of 1-hop neighbor congestion

conditions. Second, we propose to increment the V parameter already calculated using local congestion information to forward data packet b. To distinguish between the different packets, we use the IP "time-to-live" (TTL) field, which specifies the maximum number of hops a packet can traverse within the transport network. The TTL is by default decreased by one at every hop traversed in the transport network. Thus, the TTL provides an idea of the time spent over the air by a packet in the NoS. Specifically, the lower the TTL field the higher will be the increment experienced by V(t) for that packet. Figure 8.21 how we propose to ramp up the V parameter with the observed TTL value for each data packet.

Thus, the parameter V is calculated on a per-packet basis and decentralized way based on two different components, namely, one component tackling 1-hop congestion information and a second component taking into account packet deadline characteristics:

$$V_i(t) = V_{i1}(t) + V_{i2}(t); 1 \leq i \leq r \tag{8.7}$$

In our specific case, the penalty function is tightly related with the number of hops traversed by data packets. Additionally, the air time packets have been in the NoS is directly related to the TTL field in the IP header. The lower the TTL, the higher the time spent over the air in the NoS. As a result of this, we propose to bias V(t) as a function of the time the packet has been routed over the NoS. Specifically, we propose to increment the V parameter calculated for every packet, and so the emphasis in the penalty function as the TTL field in the header of the packet decreases. In particular $V_{i2}(t)$ is calculated as follows:

$$max(0, (\frac{TTL_i - TTL_p}{TTL_i - TTL_{min}}) \times (Q_{max} - V_{i1}(t)) \tag{8.8}$$

where TTL_i corresponds to the higher TTL a packet has to experience in order to apply the proposed algorithm to the packet. This is a parameter that can be configured by the routing protocol. The parameter TTL_{min} corresponds to the minimum TTL value accepted to include a queue load balancing in the routing decision. This is a parameter to be configured by the routing protocol. This parameter would come determined by the delay requirements of the NoS. Specifically, if the NoS specifies the maximum delay a data packet may experience. The parameter TTL_p corresponds to the TTL value for the packet. And $V_i(t)$ is the V value used to forward data packet p. The calculation of V on a per-packet basis is especially important for improving network performance metrics in a NoS loaded with multiples flows, and hence with a high degree of TTL variability at SCs. Finally, there are two observations we want to highlight, which may pose limitations on the proposed $V_i(t)$ algorithm:

8.2.5 Evaluation of the Variable-V on a per-packet basis

The first subsection describes the evaluation methodology used with the ns-3 [102] simulator to evaluate the per-packet variable-V algorithm. The second subsection outlines main observations from the results obtained.

8.2.5.1 Methodology

Here, the goal is to compare the Periodical V algorithm described in section 8.2.2 with the Per-packet V algorithm detailed in section 8.2.4. In all the experiments, the injected traffic consists of unidirectional UDP with maximum packet size (i.e., 1472 bytes) at a CBR rate of 6Mbps generated by the ns-3 OnOff application. The traffic patterns are such that spatial traffic variations in the network decrease with the increase of the number of traffic flows. We vary the set of source-destination SC pairs twenty times.

The duration of each simulation is of 120 seconds. The data queue size limit assigned to the SC is of 200 packets. Therefore, in this case, using a V parameter greater than or equal to 200 means using the shortest path in terms of Euclidean distance to the destination. On the other hand, the lower the V parameter (i.e., from 200 up to 0), the bigger the degree of load balancing offered by the backpressure routing protocol. The wireless link data rate used by all SCs in the backhaul is a fixed rate of 54Mbps. We label the variable-V algorithm computing periodically the V parameter as Periodical-V, whereas the variable-V algorithm computing the V on a per-packet basis is labeled as Per-packet V. With regards to the configuration of the variable-V algorithms, the Periodical-V algorithm calculates V every 100ms, and the Per-packet V policy computes the V parameter for each data packet being routed. We configured the TTL_i to 60 in Per-packet V so that the data packets can traverse at least 4 hops in the NoS without giving importance to the calculation of V on a per-packet basis. Additionally, the TTL_{min} is configured to 50, meaning that if a packet traverses 14 hops the queue backlogs are not taken into account to compute forwarding decisions. In this case we consider the data packet requires to be received by the destination even at the expense of being dropped by a data queue.

8.2.5.2 Results

Figure 8.22 confirms the convenience of computing the V parameter on a per-packet basis. Indeed, throughput results confirm the advantages of calculating V on a per-packet basis for scalability with the number of traffic flows injected in the network. The main reason for this gains are due to the more dynamic adaption of V, especially important under saturation conditions. With a Periodical-V

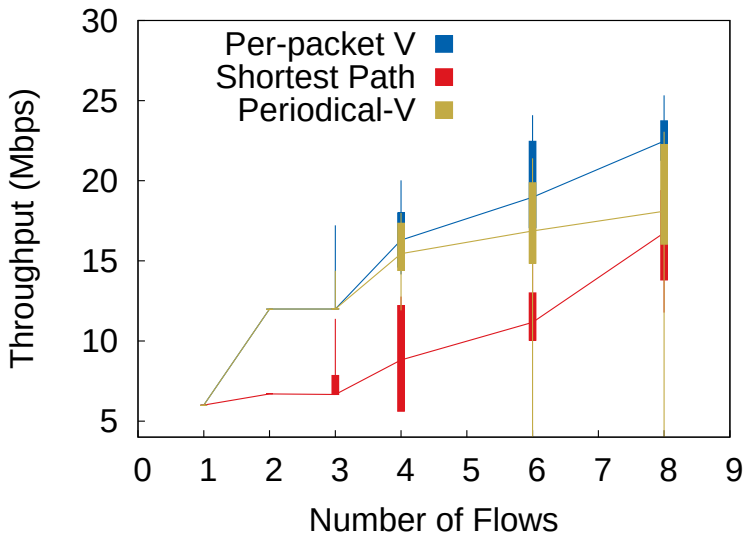


Figure 8.22: Throughput evolution under increasing saturation conditions.

calculation, the protocol cannot react to sudden changes on the traffic patterns, and under saturation conditions a periodical-V calculation tend to excessively decrease its V value in a high percentage of SCs in the network. It is under these circumstances where the increase of V with the decrease of TTL acquires relevance over a periodical computation of V. In fact, this is one the most critical caveats of computing periodically the V parameter independently in each node.

Figure 8.22 shows an important increase of throughput as the number of traffic flows increases from six to eight with the shortest path routing policy. This is because with such a workload the traffic load starts to be already evenly distributed over the network by using a single shortest path between each pair of SCs. All the SC observe an equivalent traffic load, no matter the portion of the network in which the SC is located. Thus, dynamically increasing the path utilization under such traffic conditions experience lower gains with respect to using a static shortest path.

Keeping a finer granularity of queue backlog information denoting 1-hop congestion conditions on a per-packet basis and taking into account the packet TTL to compute V yields a better response under saturating traffic demands, while it preserves the decentralized and independent calculation of V at every SC. In this case, under some traffic patterns three intense traffic flows are sufficient to experience a dramatic throughput degradation with a V parameter periodically calculated. Figure 8.23 reveals packet delivery ratio gains obtained with the computation of V on a per-packet basis with respect to a shortest path based routing algorithm, and the importance of the TTL parameter to the calculation of V with respect to SoA shortest path routing algorithm.

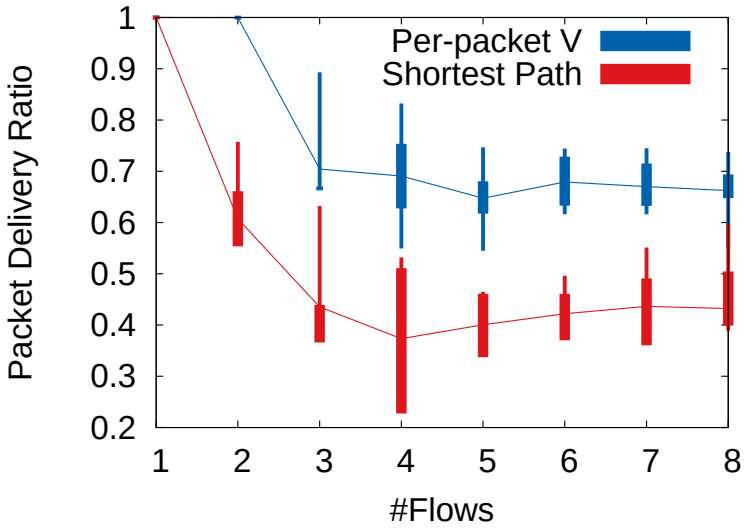


Figure 8.23: Packet delivery Ratio evolution under increasing saturation conditions.

8.3 SON Backpressure Routing with Multiple Gateways

The move towards capacity-oriented deployments has given a starring role to small cells (SC), as increasing frequency re-use by decreasing cell radii has historically been the most effective way to increase capacity. Such densification (e.g., achieved by deploying SCs in lampposts) entails several challenges, both at the mobile network layer (MNL), specified by 3GPP, and at the underlying transport network layer (TNL). As for the former, the idea of network of small cells (NoS) has been proposed to confine control plane and data plane traffic to the local environment [12]. In this way, the core network (e.g., the Evolved Packet Core, or EPC, for LTE networks) only receives a small percentage of traffic, which results in more scalable deployments. The architectural entity at the MNL allowing such confinement of traffic is the local small cell gateway (LSGW), which embeds the corresponding control and data plane entities (e.g., a Proxy-MME and a Proxy-SLSGW for LTE), and attains its goals whilst being transparent to user equipment (UE), SCs, and core network entities.

As for the TNL, the required densification makes economically unfeasible to lay fiber to each small cell. In this case there is the need, for every given number of SCs, to augment a small cell with a high-capacity link to the core. Therefore, this singular SC acts as a TNL aggregation gateway towards the core network. The rest of the SCs would then connect through a multi-hop wireless network to this gateway and among them. Hence, the SCs form a sort of local all-wireless mobile backhaul at the TNL level.

Additionally, the wireless nature of links jointly with the lower reliability of massively-deployed (hence,

low cost) equipment introduce dynamicity, hence requiring a resilient TNL solution. Finally, dense capacity-oriented deployments must be flexible enough to evolve with traffic demand and must have mechanisms to relieve hotspots. Deploying additional TNL gateways increases the global backhaul transport capacity in two ways, namely by introducing additional exit points with high-capacity links towards the core network and by reducing the average number of wireless hops traversed by traffic. However, for a full exploitation of these additional resources, balancing of traffic between TNL gateways must be possible. Therefore, the architecture should allow incrementally deploying and gracefully integrating such new network entities without introducing much additional management and/or operational burden.

This section exploits our TNL scheme that defines a Lyapunov drift-plus-penalty function [77], which is optimized in a distributed way, to deal with the aforementioned flexibility and resiliency requirements in a multi-gw environment. More specifically, our self-organized routing scheme (see chapter 7) exploits geographic and queue backlog information of neighboring nodes to take forwarding decisions by dynamically finding the appropriate balance between reaching the destination through the shortest possible path and avoiding congested spots. In fact, our scheme inherently evolves towards network states that minimize bufferbloat problems [92]. Furthermore, the combination of the above packet forwarding principles and the creation of an anycast group composed by the deployed TNL gateways allows opportunistically pulling packets out of the all-wireless NoS at any gateway, hence increasing overall capacity without introducing additional control overhead. In summary, our scheme is able to dynamically exploit both path and gateway diversity.

The ns-3 simulations (see Section 8.3.4) confirm that our TNL mechanism inherently integrates gateways and benefits from gateway diversity. In this respect, some previous work either assumes some previous knowledge of the topology [109] and/or reactively generates control overhead to establish paths to the destination [110]. Contrarily, our scheme does not require knowledge either of the topology or of the set of all gateways deployed in the NoS (only of the one taken as reference), and it does not establish paths either, despite it is able to benefit from all gateways deployed. Furthermore, it also outperforms single-path and multi-path TNL approaches that associate each SC to a given aggregation gateway. Besides, our scheme dynamically selects forwarding neighbors on a per-packet basis based on their congestion state, hence avoiding hotspots. Partially aligned with this same approach, an analogy with physics is used to derive a distributed scheme ([71], [72]). However, unlike our scheme, it requires manually setting up some of the key parameters (e.g., sensitivity to congestion) and 10 to 15 iterations affecting all nodes in the network to converge, hence making it less adaptable to realistic and varying traffic demands. Additionally, both [109] and [71] were conceived to only handle upstream traffic and

our scheme also handles downlink and local traffic without any modification to the algorithm.

In summary, no transport planning is needed in our scheme, and no re-planning or route re-calculation is needed when new gateways are deployed to fulfill increased traffic demands. Our scheme also has inherent self-healing capabilities, as the failure of a TNL gateway just means that the remaining ones will opportunistically be used instead without any additional re-configuration overhead, hence making our scheme more scalable. Some of these qualitative differences also render our scheme more advantageous when compared to other anycast (e.g., Plasma [109]) or unicast (e.g., HWMP [110]) approaches in which there is overhead for the path route construction and/or to decide what neighboring node forwards the packet.

8.3.1 Anycast Backpressure Routing

Anycast Backpressure Routing assumes there is a reference TNL gateway for a group of SCs. Additionally, there may be more TNL gateways able to pull packets out of the NoS. All the gateways form an anycast group identified by an IP anycast address. When SCs send outward traffic (e.g., when carrying traffic towards the EPC), they map the IP anycast address to the geographic address of the reference gateway, which is then used to make forwarding decisions according to the above procedures. On their way to the reference gateway, and depending on network congestion conditions, packets may traverse a non-reference gateway. Since this gateway shares the same IP anycast address with the reference one, it will opportunistically pull packets out of the multi-hop wireless network, hence freeing scarce wireless resources for other packets and consequently increasing the total network capacity.

8.3.2 Flexible gateway deployment with Anycast Backpressure

AB allows incrementally deploying gateways and smoothly integrating the additional capacity they bring without any additional configuration. That is, the operator may initially deploy a single gateway, and evenly deploy more gateways throughout the network as traffic volume increases. Therefore, instead of increasing the OPEX by requiring manual/static configuration of nodes, our scheme finds both the path and the exit gateway dynamically on a per-packet basis without requiring prior associations or route computation. As a consequence, low over-the-air overhead and low state information are introduced at nodes and global network capacity is increased because of the new (potentially wired) links behind the gateways, but also because of the reduced number of wireless hops packets traverse. As in any multi-gateway approach, these advantages may come at the cost of potential reordering of packets, which should be appropriately handled (for instance, at the TNL gateway co-located with the LSGW).

8.3.3 3GPP data plane architectural considerations

As explained in chapter 6, LSGWs embed a MNL entity acting as Proxy Gateway (e.g., Proxy S-LSGW for LTE/EPC). That is, from the point of view of the UE and the SC, the LSGW acts as a regular gateway (e.g., S-LSGW for LTE) with which bearers can be established. But in fact, it performs bearer termination inside the NoS and mapping to other bearers towards the core network (e.g., EPC for LTE networks), as well as user-plane data routing from/to the NoS and the core network. In the multi-gateway setup and focusing now on the TNL, packets may traverse a TNL gateway different from the TNL gateway co-located with the LSGW terminating the bearer. In this case, this TNL gateway pulls the packet out of the all-wireless local network and forwards the IP packet to the LSGW through a different subnetwork (e.g., a wired high-capacity network). This latter forwarding may be done by using conventional IP routing deployed in the subnetwork interconnecting the gateways. Once the packet has reached the intended LSGW, it is routed into the corresponding bearer between the LSGW and the core network gateway (e.g., S-LSGW for LTE/EPC).

8.3.4 Evaluation

Here, subsection 8.3.4.1 describes the approached methodology, whereas subsection 8.3.4.2 and subsection 8.3.4.3 discusses main results obtained with the proposed solution to deal with wireless backhaul of multiple gateways.

8.3.4.1 Methodology

In this section, we use the ns-3 [102] network simulator to evaluate Anycast-Backpressure (AB) and two state-of-the-art unicast routing strategies in a 5x5 grid representing the local multi-hop wireless backhaul of the NoS. For illustration purposes, in this particular setup, each node in the grid mounts a single 802.11a Wi-Fi radio in the 5GHz band, though AB could be used on top of any layer two technology. SCs transmit traffic at a constant bit rate (CBR). The packet size of each CBR packet is 1500 bytes. We measure throughput and latency, and use average values and boxplots to represent the distribution of the forty repetitions of 50 seconds each. The box stretches from the 25th to the 75th percentiles, and the whiskers represent the 5th and 95th percentiles. Note that for ease of visualization, we omit boxplots in some cases.

The two unicast routing strategies are an idealized abstraction of multi-gateway Single-Path (SP) and Multi-Path (MP) approaches. They are idealized in the sense that an offline centralized process selects

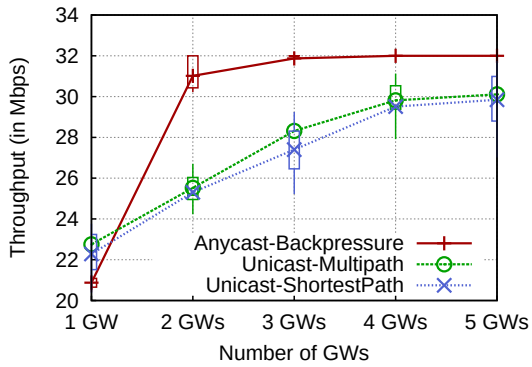


Figure 8.24: Homogeneous link rates. Agg. Throughput vs. number of gateways.

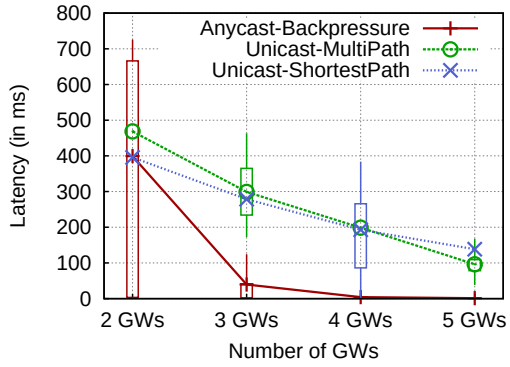


Figure 8.25: Homogeneous link rates. Average latency vs. number of gateways.

the closest gateway to each source SC. The difference is that SP establishes a single path to reach the gateway, whereas MP uses multiple paths as long as they have the same cost to the destination. Notice that a distributed implementation of such protocols would require over-the-air overhead for route construction and maintenance, which is not required in AB. The consequent reduction of throughput for SP and MP is not taken into account in the results presented.

We consider two different backhaul settings in our simulations featuring homogeneous and heterogeneous transmission rates, respectively. The goal is to evaluate how AB behaves in front of various sources of dynamicity and how it integrates the additional capacity offered by new gateways.

8.3.4.2 Homogeneous link rate scenario

This simulation scenario evaluates the behavior of AB, SP, and MP when uneven traffic loads are injected from different regions of the NoS. SP and MP represent the best possible path (or paths), i.e., the one actually used, among all paths available to an anycast protocol. Each SC injects 1Mbps of traffic, except a group of SCs in a region of the grid that inject more, for a total offered load of 32Mbps. The number

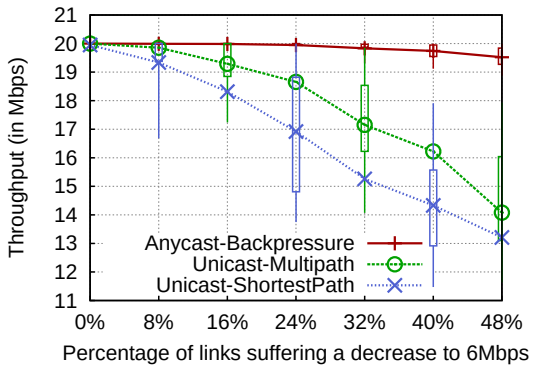


Figure 8.26: Heterogeneous link rates. Agg. throughput vs. % of low-rate links.

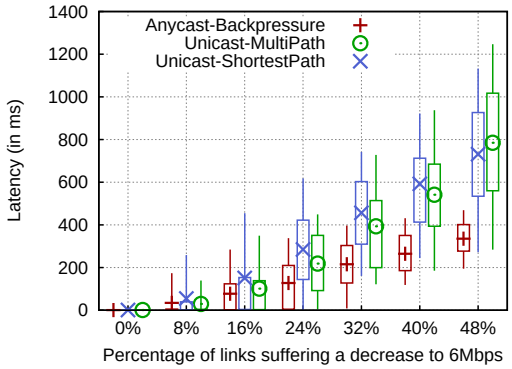


Figure 8.27: Heterogeneous link rates. Average latency vs. % of low-rate links.

of gateways is varied between 1 and 5. For each number of gateways, the set of nodes acting as gateway is randomly selected for each of the forty repetitions.

Figure 8.24 presents the achieved throughput as the number of gateways increases. Remarkably, AB is able to serve all the injected traffic with three gateways disregarding the randomly selected position of gateways, as the null (or almost null) variability of the boxplots show. On the other hand, SP and MP are not able to serve all the traffic even with a 20% of the nodes acting as gateways (i.e., 5 gateways). In this case, there is also a variability of the throughput achieved depending on the location of gateways (see boxplots). This throughput difference between AB and SP/MP can be explained by noticing that, in the latter, SCs attach to a single gateway (the closest one), while in AB any gateway can pull packets out of the network from any SC.

Additionally, the one-gateway case is also presented despite not being a realistic deployment scenario due to the highly saturated setup it represents. However, it offers an interesting insight on the operation of AB. In fact, an interesting remark from Figure 8.24 is that the average throughput for AB in this case is slightly lower compared to SP and MP. The key to understand this behavior is the observed

packet delivery ratio, which is much higher in AB, which combined with its inherent load balancing functionality generates more CSMA contention. However, under less congested setups (e.g., when deploying more gateways) aggregated throughput gains of more than 20% are observed depending on the case.

As for the latency, Figure 8.25 presents its dependency on the number of gateways. AB with 2 gateways shows high fluctuations of latency (from 2ms up to 774ms), which highly depend on the randomly selected location of the gateways. In fact, a more detailed analysis allows concluding that the lowest latencies are obtained for the smallest distances between the opportunistic/non-reference gateway and the TNL aggregation gateway that is co-located with the LSGW (i.e., the reference gateway). Additionally, these fluctuations are also due in part to the much higher contention experienced by AB, given the much higher throughputs handled by the network (see Figure 8.24). As a consequence of these observations, one may conclude that for best exploiting all AB features, an even deployment of gateways (higher than two) is needed. This is confirmed by Figure 8.25, in which AB experiences an abrupt decrease of latency with the increase of the number of gateways, starting with an average latency of 400ms with 2 gateways down to 1.4ms with 5 gateways, while SP and MP exhibit an approximately linear decrease of latency with the number of gateways, achieving 100ms in average with 5 gateways. The reader should also notice the abrupt decrease in latency variability just with one additional gateway, which is not the case for SP and MP. Overall, this confirms that AB handles heterogeneous traffic loads and exploits multiple gateways better than approaches fixing the attachment between the SC and a gateway. In fact, since SP and MP associate each SC to a single gateway, even if it is the closest one, buffers fill up and eventually overflow in congested areas, even with five gateways, hence increasing the average latency observed. As a lateral remark, the reader should also notice the crossing of the SP and MP curves, which is due to the queue drops experienced by SP, hence degrading latency measurements.

8.3.4.3 Heterogeneous link rate scenario

This simulation scenario evaluates how AB, SP, and MP exploit the aggregated capacity offered by gateways in the presence of wireless link dynamicity. More specifically, we study the dependency of the aggregated throughput and latency on the percentage of links using the lowest 802.11a rate (i.e., 6 Mbps). For each such percentage, forty replications are run, each having a different randomly selected set of low-rate links. The rest of links are configured to the highest rate, i.e., 54Mbps. There are five fixed gateways, and twenty SCs send 1Mbps of traffic towards the core network, hence making a total of 20Mbps.

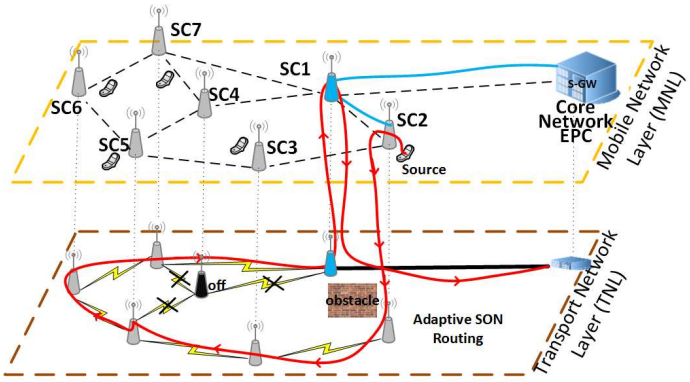


Figure 8.28: Sparse NoS deployment with obstacles and an SC powered off.

As shown in Figure 8.26, and unlike MP and SP, AB is able to serve all the offered load injected in the NoS even when a higher percentage of low-rate links are present. Specifically, AB is able to serve almost all the traffic (i.e., 99%) even when there is a 32% of low-rate links, while SP and MP are suffering a 23% and 14% of throughput degradation, respectively. This relative throughput degradation increases with the number of gateways up to a point in which AB achieves 40% and 30% throughput gains with respect to SP and MP. Besides, throughput variability in AB is much smaller than that of SP and MP. This indicates a high independence with respect to wireless link dynamicity due to its load balancing capabilities. Notice also that the variability is smaller despite handling more packets and experiencing more contention.

As for latency, Figure 8.25 shows increasing latency gains with the increase of percentage of low-rate links. (Notice that, for the sake of readability, points are shifted so that boxplots do not overlap.) Specifically, for a 48% of low-rate links, AB attains twice as low latency compared to SP and MP (i.e., 340ms for AB versus 750ms for SP and 800ms for MP). Besides, the variability of latency with AB is smaller in all cases, and the difference in variability between AB and SP/MP increases with the percentage of low-rate links. This comes as a consequence of SP and MP making an excessive use of low-rate links. In summary, Figures 8.26 and 8.27 confirm the ability of AB of relieving network congestion by exploiting all available wireless link resources.

8.4 SON Backpressure Routing in Sparse Deployments

The semi-planned and low-cost nature of all-wireless NoS deployments will inevitably lead to non-regular topologies, SC failures, or disconnection due to obstacles (e.g., wireless link amongst SC_1 and SC_2 in Figure 8.28), wireless link variability (e.g., due to adverse weather conditions), or vandalism.

On the other hand, a wireless mesh backhaul is subject to traffic dynamics. The study in [89] shows that a large fraction of mobile subscribers generate traffic only a few days a week and a few hours during the day. The activation of all SCs when a low fraction of mobile subscribers are using the network results in unnecessary resource consumption. A possibility is to power off some SCs (e.g., SC4 in Figure 8.28) selectively during low load conditions (e.g., at night), whilst still being able to serve all the traffic. Despite energy efficiency gains, these mechanisms may also substantially alter the wireless backhaul topology, hence contributing to non-regular sparse deployments.

The dynamicity of the above context may render transport protocols such as MPLS-TP [20], traditionally used in wired TNLs, unsuitable for an all-wireless NoS. As shown in chapter 4, a strategy to tackle large-scale multi-hop wireless topologies is geographic routing. However, these strategies entail a substantial increase in control overhead as well as an increment of the per-node routing state, required to build alternative routes, hence compromising their scalability in sparse deployments. Further, despite eventually circumventing network voids, geographic routing can lead to network congestion due to a misuse of network resources and a high routing stretch (i.e., the ratio of the hop count of the selected paths to that of the shortest path) due to the lack of flexibility of the route recovery method.

Our previous work in this chapter helped us to evaluate the potential of combining geographic and backpressure routing when applied to regular (i.e., grid-like) multi-hop wireless networks. However, practical deployments are far from regular due to the reasons explained above. We tackle this fundamental aspect to deploy BS in realistic non-regular topologies whilst retaining the beneficial features of our previous schemes. Instead of using a complex and resource consuming geographic recovery method to deal with dead ends, we propose a self-organized, low-overhead, scalable, and decentralized routing approach that makes the most out of the network resources while maintaining a low routing stretch. Unlike geographic routing schemes, the resulting approach neither requires routing recovery methods or incrementing the per-node state to dynamically adapt to the current wireless backhaul topology.

Extensive ns-3 [102] simulations results validate the robustness of BS under a wide variety of sparse wireless mesh deployments and workloads. Under uncongested traffic demands, BS showed a latency and routing stretch close to an idealized single path routing protocol (ISPA), which is aware of the global current network topology without consuming air resources. ISPA is taken as an abstraction of traditional TNL protocols of core networks and wired mobile backhauls (e.g., MPLS-TP [20]). In turn, BS improved the latency results obtained by GPSR [68], taken in general as benchmark for comparison against geographic routing featuring void circumvention mechanisms. In the case of more severe traffic conditions, BS outperforms both GPSR and ISPA showing a reduction in terms of average latency of up to a 85% and 70%, respectively, due to its inherent load balancing capabilities while serving the offered

Figure 8.29: Sparse wireless mesh backhaul scenario.

load and maintaining a low routing stretch.

In subsection 8.4.1, we list the main problems to tackle when using backpressure routing in sparse deployments. Details on the operation of BS are provided in subsection 8.4.2. Finally, subsection 8.4.3 discusses the simulation results. Some preliminary results in this section appeared in [7]. However, the required details for the proper operation of the protocol as well as an extensive evaluation are novel in this section.

8.4.1 Limitations of Backpressure in Sparse Deployments

Despite the potential of the above framework, a few problems remain to be solved. Here, using network simulation with ns-3 [102], we analyze how our scheme reacts under sparse deployments. Figure 8.29 depicts a 5x5 grid mesh backhaul, and assume that a percentage of the SCs (shaded nodes) have been powered off at a certain instant. Each SC maintains a single FIFO queue with Q_{MAX} equal to 200 packets. We assume that for each SC, horizontal and vertical neighbors are 1-hop neighbors whereas diagonal neighbors are considered 2-hop neighbors. Within this scenario setup, SC_6 sends a 2Mbps UDP CBR flow to SC_8. Thus, given that the SC_7 is unavailable, the shortest path under this mesh backhaul configuration has 4-hops through SCs (11, 12, 13, and 8) and (1, 2, 3, and 8).

Figure 8.30 plots the time evolution of the queue backlog in SC_6, the one facing a dead end due to SC_7 being switched off at time $t = 5s$. The routing scheme is configured with different fixed V values and using the variable-V algorithm, which is described above. Interestingly, Figure 8.30 reveals that packets remain trapped up to a certain extent. *The first aspect to point out is that such a scheme requires to fill the queue of the SC being the local minimum up to a limit in which routing decisions emphasize more the reduction of queue backlog differentials rather than geographic proximity to the destination. Nonetheless, packets remain trapped in SC_6 once the queue backlog is below this limit (i.e., at instant*

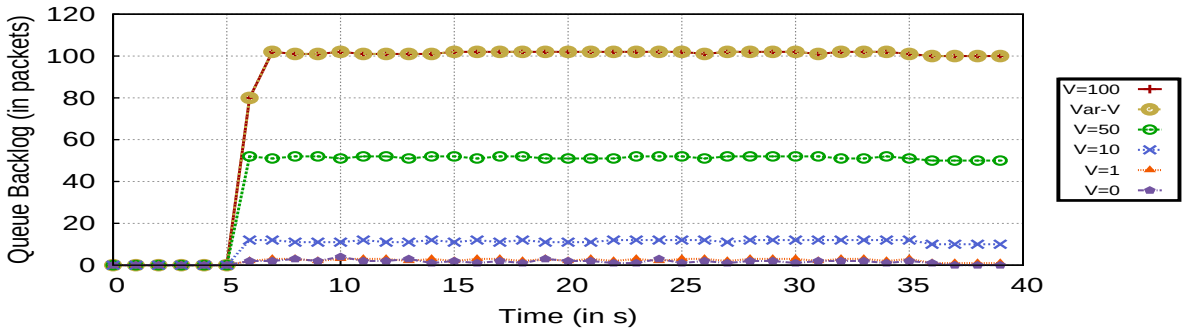


Figure 8.30: Impact of the value of the V parameter in the queue backlog to circumvent network voids. The void is circumvented once the queue backlog of the local minimum is stabilized, but some packets may get trapped.

$t = 35s$ *when the flow terminates).*

We observed how different configurations of the V parameter yield different queue backlogs to enable the use of queue backlog differentials, hence allowing packet to escape from the network void. With the decrease of the V parameter, the queue threshold required to start taking routing decisions based on queue backlog differentials also decreases, hence causing a decrease of queuing latencies. *The second point to remark is that the configuration of the V parameter is of primal importance to determine the extent of queue backlogs to escape from network voids, and so, has a significant influence in the attained latency.*

Table 8.1: Impact of the value of the V parameter in latency.

Value of V parameter	Average Latency (ms)
V=0	143.81 ms
V=1	19.67 ms
V=10	126.79 ms
V=50	603.00 ms
V=100	1198.21 ms
Variable-V	1198.21 ms

Table 8.1 exhibits the consequent decrease on end-to-end latency with the decrease of the V parameter. Note that the case of the Variable-V, initialized to Q_{MAX}, requires to decrease its value up to half the value of Q_{MAX} to enable routing decisions based on the minimization of queue backlog differentials. The trend towards lower latencies has a turning point at V=0. Indeed, we observe a latency increase

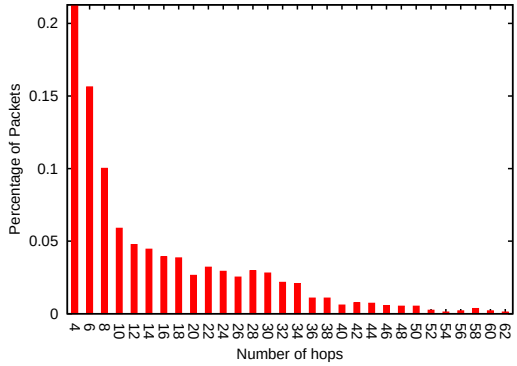


Figure 8.31: Histogram of the hop distribution when exclusively making forwarding decisions based on queue backlogs ($V = 0$).

with respect to V=1 because the forwarding decisions are exclusively based on the minimization of queue backlogs, without taking into account where the proximity to the destination. This results in long paths to reach the destination.

The third main aspect to highlight is that an excessive use of the minimization of queue backlogs to take routing decisions could result in excessive path lengths to escape from network voids. Figure 8.31 shows the hop distribution of all the packets carried in the backhaul with V=0. Note that the minimum path length in the considered case is 4 hops. Instead of merely using 4 hop paths, data packets traverse a number of hops that increases up to more than 60 hops. In particular, there is only a 20% of data packets following paths of a minimum number of hops. In this case, a fixed V value parameter set to 1 may solve the aforementioned problems for one single traffic flow. Nevertheless, as argued in [2] and [5], this is not a feasible solution, since it does not appropriately handle the traffic and network dynamics that are the norm in wireless networks, hence resulting in routing loops and increased latencies.

Nonetheless, given its potential, the solution presented in this chapter still relies on the Lyapunov drift-plus-penalty approach with a single queue per node to handle any-to-any traffic communication patterns. In this way, we benefit from its advantages, namely scalability, self-organization, statelessness, decentralization, and low control overhead.

8.4.2 Backpressure Solution for Sparse Deployments

Given the above context, this section describes the proposed scheme to void handling without incurring into additional routing recovery procedures, whose objectives are 1) to minimize queue backlogs (and associated latencies), 2) to avoid excessive path lengths causing potential routing loops, and 3) to avoid packets to get trapped at data queues while maintaining the advantages of our original scheme [5].

Both the minimization of the Lyapunov drift and the Variable-V algorithm are key components to make the most out of the network resources. An appropriate cost function is fundamental to avoid the inefficiencies observed in section 8.4.1 for sparse deployments and achieve the aforementioned objectives. As revealed by the results in section 8.4.1, this is particularly noticeable when a substantial decrease of latency is observed when reducing the importance of the geographic-based cost function when taking routing decisions (low values of V). Subsections 8.4.2.1, 8.4.2.2, and 8.4.2.3 explain the main intuition, as well as the details of the operation and implementation of the cost function.

8.4.2.1 The Intuition behind the Cost Function

The cost function $c_{i,j}^{d}(t)$ conceived for sparse (and uniform) deployments follows the same trend of rewarding the selection of SCs closer to (and penalizing SCs farther from) the destination when there is uniform connectivity. However, the proposed cost function differs from the previous one designed for uniform SC deployments in two key ways in order to avoid queuing latencies under dead ends.

First, the cost function includes the possibility of rewarding routing decisions that select SCs located farther from the destination in the presence of dead-ends, rather than allowing packets to get trapped in data queues. The second key point is to penalize decisions generating 1-hop loops, which occur when a packet is routed back to the node from which the packet was just received. In this way, a 1-hop loop would occur when there is only one neighbor available and the Lyapunov drift minimization gains in importance regarding the cost function.

The results presented in section 8.2.5 show that, with these two simple features in the cost function plus the geographic and backpressure components, the sparse deployments under consideration can serve the offered load appropriately.

8.4.2.2 The Cost Function

Before delving into the details of the cost function, let us first define some auxiliary functions. Let the loop function $L_{i,j,d}(t)$ be equal to 1 when the current node i forwarding packet p received this packet from node j (that is, there is a 1-hop loop), and 0 otherwise. Additionally, let $NC_{i,d}(t)$ denote the set of 1-hop neighbors of node i closer to the destination d and $dist(n1, n2)$ be the function that calculates the Euclidean distance between nodes $n1$ and $n2$. Finally, let $OC_{i,d}(t)$ be a binary function that, when forwarding a packet from i headed to d, is equal to 1 if there is a single neighbor closer to d, and 0 otherwise. According this notation, the cost function is defined as:

$$c_{i,j}^{d}(t) = \begin{cases} L_{i,j,d}(t) - 1 & dist(j,d) < dist(i,d) \\ & \text{or } (dist(j,d) > dist(i,d) \\ & \text{and } |NC_{i,d}(t)| = 0) \\ 1 - 2L_{i,k,d}(t)OC_{i,d}(t) & dist(j,d) > dist(i,d) \text{ and} \\ & dist(k,d) < dist(i,d) \\ & \text{and } |NC_{i,d}(t)| \geq 1 \end{cases} \tag{8.9}$$

For easing the description of the cost function, we use the sparse network illustrated in Figure 8.32. The first case in equation (8.9) represents how the cost function treats neighbors i) closer to the destination, or ii) farther from the destination when there are not neighbors closer to the destination. For nodes closer to the destination, the loop function determines whether the cost is -1 (rewarding the closer neighbor if the loop function is 0), or is equivalent to 0 (base the decision on the queue drift to reward such neighbor if the loop is 1). In this sense, forwarding decisions approaching packets to the destination d are rewarded, unless it supposes a 1-hop loop. When packets reach dead-ends such as SC_{11} in Figure 8.32 (i.e., $|NC_{i,d}(t)| = 0$) our approach circumvents the void by rewarding forwarding decisions towards a node j, which is farther from d than the local node i. However, to avoid never-ending 1-hop loops, we check if the packet arrived from that same node j. This is controlled by the loop function, as when it is equal to 1, it makes the cost function towards node j equal to 0, and so, other nodes farther from d different from j are preferred. In terms of weights, these farther nodes are preferred over closer node j because i obtains a more negative cost function when forwarding this packet, and so, a higher weight is obtained. Since the packet is eventually forwarded to the neighbor with the highest weight at the time node i takes the forwarding decision, these nodes are selected first. On the other hand, if there is no other node except j for circumventing the void (i.e., going backwards through the same path), $L_{i,k,d}(t)$ is equal to 1, hence resulting in node i taking the forwarding decision exclusively based on the queue drift (e.g., backward path from SC_{24} to SC_{23} in Figure 8.32). Since, queue backlogs in dead-ends are high, taking such decisions allows packets to go backwards to circumvent the void.

The second case of equation (8.9) is devoted to handle the cost calculation for nodes that are farther from the destination d when node i (the local node forwarding the packet) has neighbors closer to d. In the normal case, this will result in the cost being equal to 1. When combined with a positive value of V in equation 7.11, it results in lower weights than for those nodes close to d. Therefore, closer nodes are preferred. However, to avoid packets being trapped in the queues of dead-end nodes (or nodes close to dead-ends) another case must be handled.

For instance, the depicted network in Figure 8.32 forms a multi-hop line sub-topology surrounded by a network void. If a packet arrives to the dead-end node (e.g., SC_{11} in Figure 8.32), it will be handled

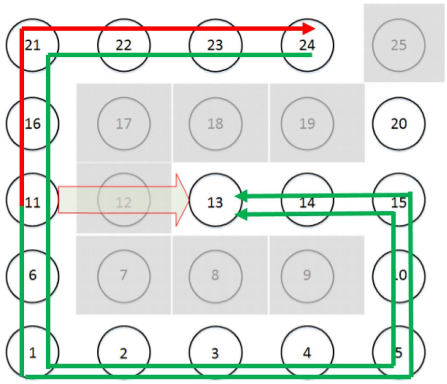


Figure 8.32: Topology surrounded by network void.

by the first case, as explained above. However, once this packet reached the node just before the dead-end (e.g., SC_{16} or SC_6 in Figure 8.32), instead of sending it again towards the dead-end, it must send it backwards (see red arrow in Figure 8.32) until all the hops in the line sub-topology are traversed to be able to circumvent the whole. This is handled in the second case of the equation with the term $2L_{i,k,d}(t)OC_{i,d}(t)$. In fact, this term is different from 0 only when node i (the local node) receives a packet from node k (the only one closer to d) from which the packet arrived to i. In this case, both $L_{i,k,d}(t)$ and $OC_{i,d}(t)$ are equal to 1, which makes the cost become negative, which in turn, results in a high positive weight and the packet is sent to node j (farther from d) instead of k (closer to d). Thus, the packet traverses the line sub-topology in the backward direction. In this way, packet do not get trapped in the queues of such kind of sub-topologies.

8.4.2.3 Implementation details of the Cost Function

To implement the binary function $L_{i,j,d}(t)$, knowledge of an identifier of previous hop that forwarded the data packet is required. In the IP header of a data packet, there is neither information identifying the previous node/SC that transmitted a data packet nor the coordinates (or the IP address) of that node. Rather than adding new headers with the source IP address of the previous hop, in our implementation we use MAC addresses for that purpose.

In terms of state information, each SC maintains a table with information related to its available 1-hop neighbors. Furthermore, each entry of the table contains the queue backlog, the geographic coordinates, the IP address, and the MAC address of the neighbor. Additionally, we store the source MAC address of each incoming packet to be forwarded in the data queue. For each data packet being forwarded, the SC checks whether the MAC address of the target next hop matches the MAC address stored with the packet. If there is a match, $L_{i,j,d}(t)$ becomes 1 for packet p, and in this way 1-hop routing loops are

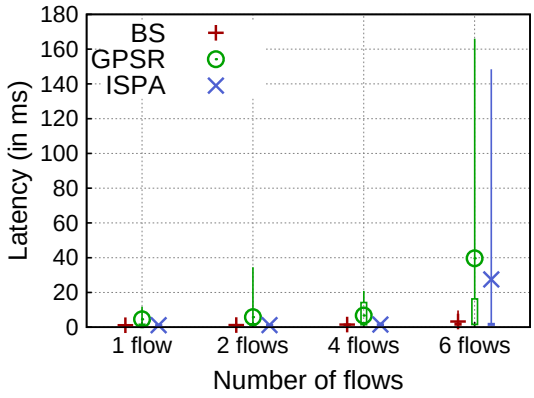


Figure 8.33: Latency results are independent of the set of SCs switched off as long as there is a path between each pair of source-destination SCs.

detected.

8.4.3 Evaluation

Subsection 8.4.3.1 describes the methodology followed to evaluate the resulting mechanism. Our evaluation focuses on studying the impact of traffic demands in subsection8.4.3.2, and that of the wireless backhaul topology in subsection 8.4.3.3

8.4.3.1 Methodology

We conduct all the simulations with the ns-3 [102] network simulator, with a duration per simulation run of 50 seconds. The simulated network is a 5x5 square grid backhaul of SCs, where the distance between neighboring nodes is of 100 meters. The set of neighbors of a given SC are the nodes within a range of 100 meters. To carry backhaul traffic, every SCs has a single IEEE 802.11a WiFi interface configured to the same channel, and at a link rate of 54Mbps. In particular, we use a simple WiFi channel model with a 2-hop interference pattern that does not generate losses due to hidden nodes or propagation errors.
The goal of all the experiments is to show the robustness of backpressure for sparse deployments (BS) when some SCs of the modeled wireless mesh backhaul are unavailable. The set of unavailable SCs is selected as follows: an SC may be unavailable with a certain probability $p > 0$ when there is not any other SC already unavailable within wireless transmission range, else $p = 0$ (i.e., SC continues active). This methodology ensures that a path can be constructed between any possible combination of source-destination SCs. We fixed the set of powered off SCs to the 20% of the total number of SCs. Figure 8.29 illustrates an example of the strategy followed to switch off nodes in the 5x5 grid.

To evaluate BS robustness, simulation results compare the performance of BS with that of GPSR, and an idealized version of the shortest path routing algorithm (ISPA) under different traffic demands and different backhaul topologies. We use GPSR as a benchmark because it is the reference protocol in the literature [66] for comparison with geographic routing protocols, given its robustness and low control routing overhead. In turn, we use ISPA because it follows, in terms of data plane, the philosophy of current protocols deployed in the mobile backhaul such as MPLS-TP [20]. ISPA is 'ideal' in the sense of having a complete knowledge of the global network topology without exchanging any control information message. Therefore, ISPA always knows a priori the shortest path in terms of number of hops between any pair of nodes, hence building routes that do not use nodes that are switched off. That is, network voids are not a problem in ISPA. On the other hand, GPSR must change its operation from greedy forwarding mode to perimeter routing, using the right hand rule to guarantee that it will find a path to the destination. BS, on the contrary, merely uses the distributed computation of weights to circumvent network voids, as described in the previous section.

In addition to this, note that in all the ns-3 simulations, BS sends HELLO broadcast messages of 110 bytes every 100ms, whereas ISPA routing does not transmit any control messages. In turn, GPSR [68] sends HELLO messages of 135 bytes every 100ms, and includes an additional header in the data traffic, adding 50 bytes to the packet size (1488 bytes). Every SC maintains a single first-in first-out (FIFO) data queue of a maximum of 400 packets.

We characterized the performance of each protocol by measuring the throughput, latency, number of hops, and routing stretch (i.e., ratio of the hop count of the select paths to that of the shortest path) in every simulation in steady state (i.e., transient periods in each simulation run are discarded). These results are obtained using our implementation of BS, the GPSR implementation provided by the authors of [111], and the ISPA implementation found in [112]. Note that most of the results regarding throughput are omitted, since, unless explicitly mentioned, the offered load is fully served by the three routing protocols. For each of these network performance metrics, we generally used average values and boxplots to represent their statistical distribution. In particular, the box stretches from the 25th to the 75th percentiles, and the whiskers represent the 5th and 95th percentiles.

8.4.3.2 Impact of Traffic Demands

This subsection provides the comparison of the performance of BS, GPSR, and ISPA while keeping a fixed set of SCs unavailable (see Figure 8.29) and considering different traffic workloads. In all the simulations, the same set of source-destination pairs are considered for all the routing protocols under comparison. The number of traffic flows injected to the network varies from 1 to 6, out of the set

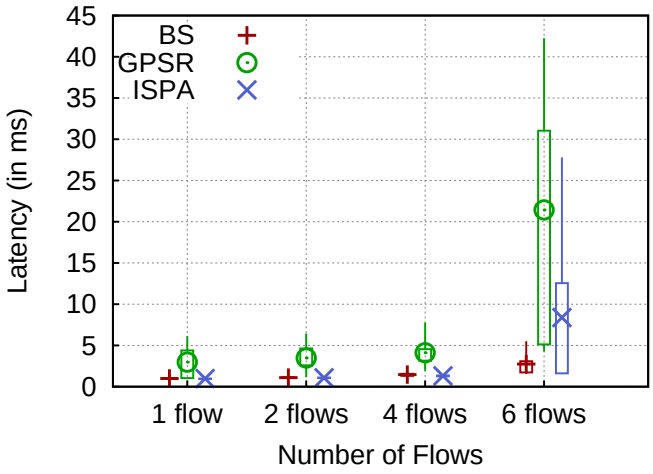


Figure 8.34: Latency distribution for 20 different wireless backhaul scenarios.

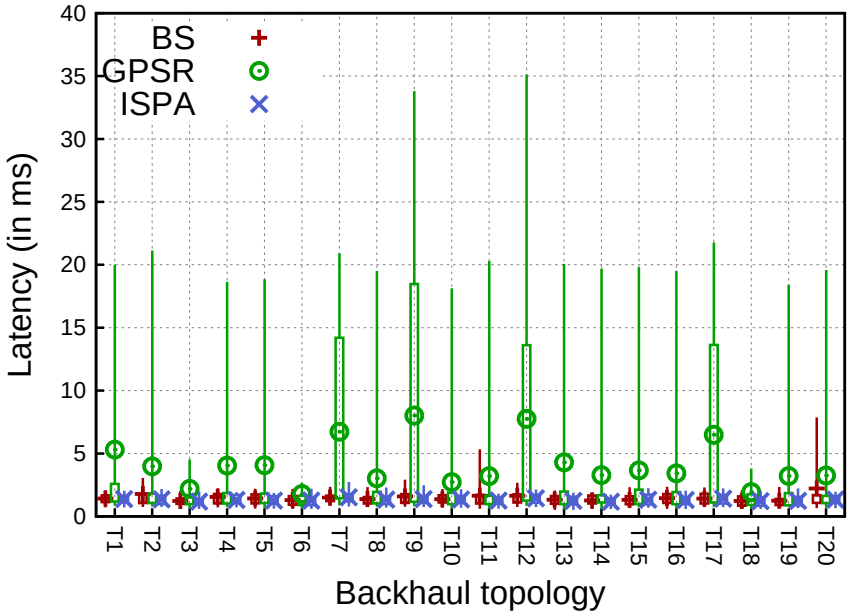


Figure 8.35: Latency distribution for 40 random source/destination pairs along 20 backhaul scenarios.

{1,2,4,6}. The generation of the set of traffic flows followed an incremental approach. That is, when moving from 2 to 4 flows, two new flows are added to the previous ones (i.e., 2 out of the 4 flows are the same ones as in the 2-flow case). Each of the four different traffic workload configurations were simulated 40 times with different seeds. The use of different seeds caused a random generation of the required source/destination SC pairs. Each selected source SC injected 2Mbps of UDP CBR traffic directed towards a destination SC, for a total offered load in the backhaul of $2Mbps \cdot Number\_of\_Flows$, where $Number\_of\_Flows \in \{1, 2, 4, 6\}$. Thus, we execute 160 different simulations for each protocol, hence resulting in 480 simulations in total. Figure 8.33 compares the average latency of BS, GPSR, and

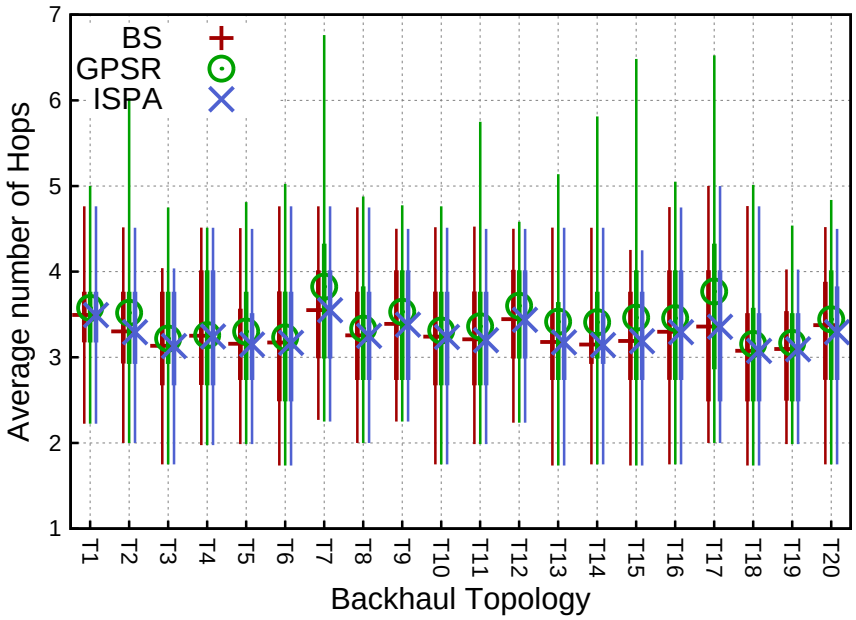


Figure 8.36: Hop Distribution for 40 random source/destination pairs along 20 backhaul scenarios.

ISPA as the number of traffic flows varies from one to six flows.

With one, two and four traffic flows, the performance of BS is on average close to that of ISPA, as both protocols route data packets following paths of a minimum number of hops. Whilst ISPA builds offline end-to-end shortest path routing tables in every SC and requires topology information of the whole network to do that, BS only requires neighbor information. However, despite this remarkable qualitative difference, BS neither traverses paths with an excessive number of hops nor increases average queue backlogs. In turn, BS outperforms the latency values of GPSR given that the defined cost function in section 8.4.2 allows overcoming local minima in a more efficient way than GPSR. Actually, GPSR starts suffering from highly variable latencies when the number of flows is equal or bigger than two. The backhaul topology showed in Figure 8.29 and the randomly chosen source-destination SC pairs provoked GPSR to use perimeter mode for some combinations of source-destination pairs. When GPSR enters in recovery mode, the use of the right-hand rule to overcome a dead-end can often lead to suboptimal paths in terms of number of hops, increasing the end-to-end latency, even under light traffic conditions. Additionally, switching from greedy forwarding to recovery mode already increases the latency. Data packets experiencing recovery mode at the source node are queued and periodically served on bursts once the requested route, calculated using the right hand rule, is known [111]. In this way, when such packets are being served, the FIFO service policy is altered, which derives into additional latency perturbations to the rest of the traffic flows traversing the node.

When the number of traffic flows is equal to 6, the backhaul starts suffering from congestion and the latency increases with GPSR and ISPA because these protocols do not take into account the available

resource to take routing decisions. Thus, while certain SCs get congested, other SCs are not used at all. With six traffic flows, BS shows lower average latencies than GPSR and ISPA due to its inherent traffic distribution capabilities. By exploiting the minimization of the Lyapunov drift, BS attains an even resource consumption in the mesh backhaul. The resulting distribution of traffic is such that the resulting deviation from shortest path aims to relieve congestion, hence resulting in lower latencies despite traversing longer paths.

8.4.3.3 Impact of the Backhaul Topology

The goal of this subsection is to evaluate the robustness of BS compared to that of GPSR and ISPA when varying both the backhaul topology and the traffic demands. To this aim, this subsection extends the previous one to include in the evaluation twenty different mesh backhaul topologies. We generated each of the twenty topologies by varying the set of SCs switched off from the 5x5 grid illustrated in Figure 8.29. The set of powered off SCs have been selected following the strategy explained in subsection 8.4.3.1.

Figure 8.37 plots the average latency distribution exhibited by BS, GPSR, and ISPA accounting the twenty sparse mesh setups as the workload increases (from one flow to six concurrent flows). In turn, the latency values have been calculated over forty independent repetitions. Each of these forty simulations runs used a random set of source-destination pairs. As a result, we run 3200 different simulations for each protocol, making a total of 9600 different simulations for the three routing variants.
The most remarkable observation is that simulation results confirm the robustness of BS in different sparse mesh backhaul topologies. BS outperforms both GPSR and ISPA showing a reduction in terms of average latency of up to a 85% and 70% and maintaining its inherent load balancing capabilities while overcoming voids with a low routing stretch. Despite varying twenty times the backhaul topology, the latency trend showed by the three protocols in Figure 8.37 is similar to that showed in the previous subsection with a fixed backhaul topology (see Figure 8.33). The workload is almost always served, yet there are some throughput inefficiencies under heavy traffic conditions, i.e. with six traffic flows. Figure 8.38 shows the cumulative distributed function of the attained throughput for six traffic flows. Results show that GPSR and ISPA fail more frequently than BS, but for a minority (less than 1%) of the chosen source-destination pairs, for which the workload is above the rates that the network can handle, BS exhibits a remarkable degradation of throughput. This is due to the fact that the distributed variable-V algorithm of BS excessively decreases the V values in most of the SCs to zero. Thus, routing decisions are merely focused on minimizing the Lyapunov drift rather than maximizing the rate of data arrivals at the destinations. The other protocols stick to a given route, handling such saturation conditions by losing

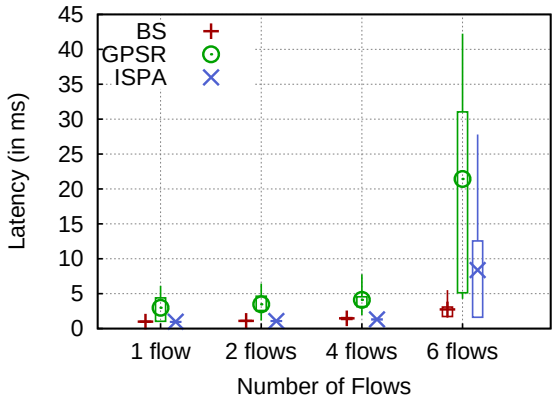


Figure 8.37: Average latency for each topology and a fixed workload.

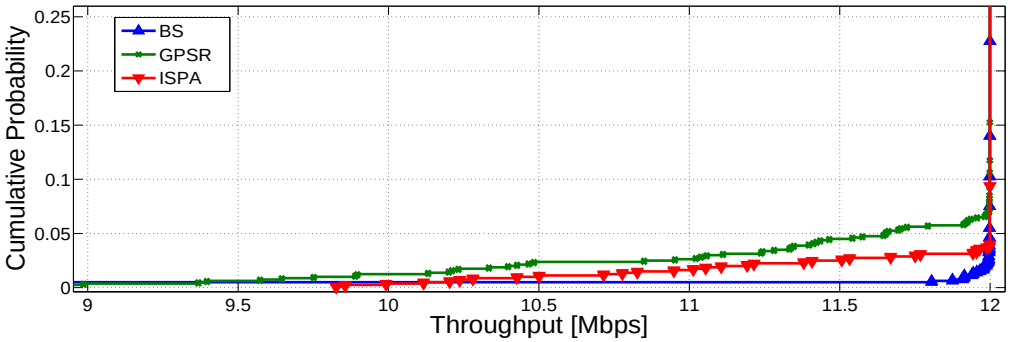


Figure 8.38: Aggregated throughput for six traffic flows.

packets due to buffer overflows at nodes. This behavior results in less packets being transmitted over the network, and so, the remaining packets can be more appropriately served. However, the reader should recall that these are saturation conditions that the operator will avoid by other means. One potential solution is the design of a distributed flow rate controller that shapes the injected traffic to prevent the network from reaching saturation as in [78].

Figure 8.39 depicts the average path length distribution of the three routing variants for the twenty sparse mesh setups and a workload of four traffic flows. Note that ISPA represents the lower bound in terms of path length distribution, and can be considered the optimal routing solution in terms of maximizing throughput and minimizing latency under light traffic loads. Interestingly, Figure 8.39 confirms that BS exhibits a path length distribution close to that attained by ISPA for the twenty sparse mesh setups and different combinations of four traffic flows injected in the network.

Figure 8.40 illustrates the specific distribution of latency of the three routing variants for each of the twenty backhaul mesh topologies and a workload of four traffic flows. We can highlight two main observations. First, BS clearly outperforms GPSR for the twenty mesh backhaul topologies given the

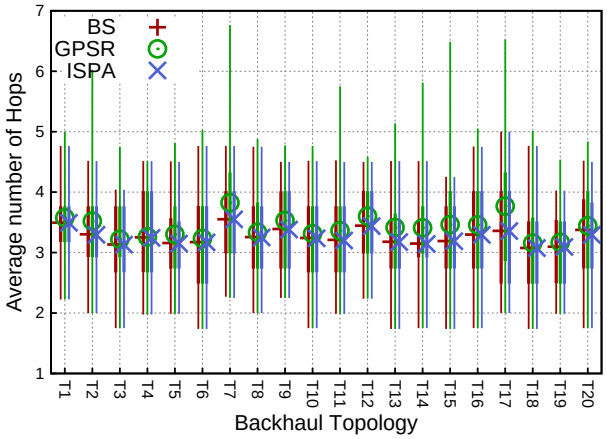


Figure 8.39: Average number of hops experienced in each one of the 20 mesh backhaul topologies with a workload of 4 flows.

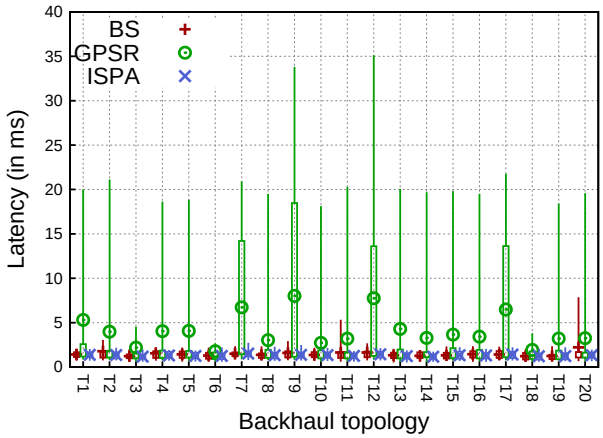


Figure 8.40: Latency distribution in each one of the 20 mesh backhaul topologies with a workload of 4 flows.

potential inefficiency of the routing recovery mode of GPSR. Second, BS presented similar latencies in most of the mesh backhaul topologies under evaluation compared to ISPA. We only found two backhaul topologies, labeled as T11 and T20, of slightly higher average latencies in BS compared to ISPA. For these specific setups, BS experiences a higher latency in the 5% of the forty repetitions.

Figure 8.41 presents the attained latency distribution attained by BS, GPSR, and ISPA for the considered twenty backhaul topologies and a workload of six traffic flows. BS clearly outperforms GPSR both on the average values and on their variance. Additionally, latency values of BS present a better trend on average than those attained by ISPA, as showed in Figure 8.37. The single shortest path selected by ISPA is not sufficient to serve efficiently the traffic, causing congestion. Load balancing is required to avoid packets stay longer periods at SC queues. Indeed, the problem with GPSR as well as ISPA is

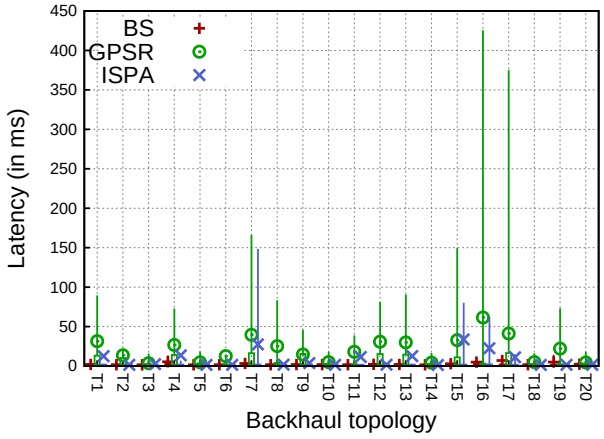


Figure 8.41: Latency distribution in each one of the 20 mesh backhaul topologies with a workload of 6 flows.

that they are insensitive to congestion, being not able to adapt dynamically their routes to relieve traffic congestion. The improvement in terms of latency experimented by BS is explained in part by its ability to distribute traffic, exploiting in a more efficient way the backhaul resources.

Despite the increase of the offered load to six traffic flows, BS still shows a path length distribution similar to that of ISPA. This indicates that BS leverages multiple non congested paths of a low number of hops, whereas ISPA merely uses one single shortest path. This explains why the difference in number of hops is so small between ISPA and BS. It is also important to note that the path length distribution of BS proves that routing loops are rare for most of the twenty topologies under consideration. When balancing traffic, the aim of BS is to prioritize short rather than long paths. Figure 8.42 presents the CDF of the routing stretch metric for the BS and GPSR protocols with respect to ISPA (the routing protocol providing the optimal route in terms of hops) when injecting a workload of 6 flows. As we can observe, GPSR experiences path lengths up to 4.5 times bigger than ISPA, whereas BS path lengths do not exceed 1.7 times this value. BS incurs in more hops in order to exploit the resources of the network when overcoming a communication void, while GPSR uses the right-hand rule, which is not always effective. Simulation results indicate that the right-hand rule chooses on average a non optimal route in 50% of the cases, which derives in a high routing stretch value.

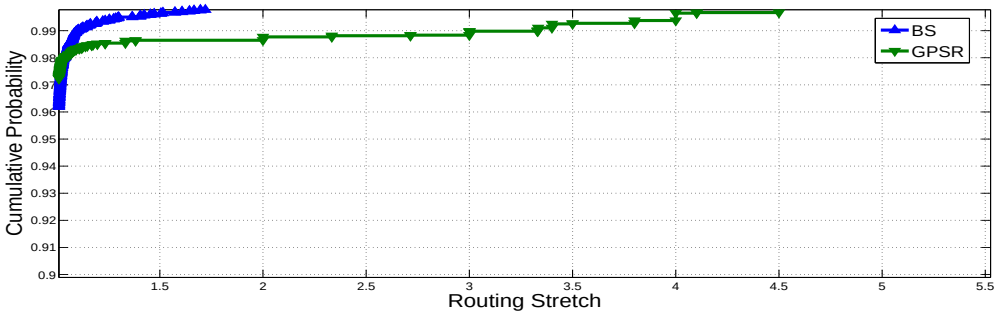


Figure 8.42: Routing stretch CDF of the flows considered in each one of the 20 mesh backhaul topologies with a workload of 6 flows.

Part III

Prototype and Experimental Results

Chapter 9

Proof-of-concept Implementation

This chapter describes the implementation of the Lyapunov drift-plus-penalty routing scheme described in previous chapters to obtain real measurements in a real wireless mesh backhaul testbed. The contributions of this chapter have been published in [13] and [15]. The goal is to minimize the time devoted to deploy the protocol in the testbed. In this way, it is also a challenge in this thesis to provide rapid prototyping of wireless network protocols in real testbeds. Usually, a wireless networking researcher has to choose between either a network simulator or a proof-of-concept testbed to evaluate a designed wireless network protocol. As for the former, the main issue is the accuracy of the obtained results. As for the latter, one main concern is the evaluation of its scalability due the low network size that real deployments exhibit.

An approach to evaluate the degree of accuracy of a network simulator is to compare the simulation results obtained with the experimental results. Once the researcher determines simulation accuracy, scalability tests can be conducted with the validated network simulator. In our case, we used the ns-3 network simulator. Despite the accuracy demonstrated by the ns-3 simulator in our work in [4], both methods should be approached to tackle scalability and accuracy requirements. In general, the aforementioned approach requires to start from scratch twice the implementation of the network protocol, that is, one for simulation and one for the testbed. Thus, this approach significantly increases the time in-

vested on development, prior to claim evaluation results of the network protocol have been appropriately validated for scalability and accuracy requirements.

The ns-3 [102] network simulator provides one feature precisely conceived to decrease development time. This feature, referred to as emulation mode, is devoted to avoid costly re-implementations in real testbeds. Basically, it allows the ns-3 simulator to send ns-3 generated packets to real physical devices, and to receive real (or ns-3) packets from real physical devices (see Section 9.1 for background on emulation). On the one hand, as it is designed in a way that follows the Linux implementation stack, packets are created with the same form as in the Linux case. On the other hand, ns-3 is able to send/receive packets from a device which can be attached to a real physical device. However, the ns-3 emulation framework has not been used to develop routing protocols for wireless testbeds.

The main points tackled in this chapter are (i) how we do design and develop the Lyapunov drift-plus-penalty routing algorithm conceived in chapter 7 for ns-3 so that it can be run by the ns-3 emulation framework without a complete re-design; (ii) the integration necessary to run the Lyapunov drift-plus-penalty algorithm ns-3 implementation in the real testbed.

In summary, our routing protocol, conceived in chapter 7 and adapted for the mesh backhaul in chapter 8, calculates wireless link weights on a per-packet basis by means of the computation of the distributed Lyapunov drift-plus-penalty routing algorithm (also referred to as distributed backpressure routing), picking for transmission those links that maximize such weight. Section 9.2 provides a summary of how we designed from scratch the routing protocol by decomposing it on building blocks so that it can be easily ported to the ns-3 emulation framework. Prior to this, we provide a brief summary on emulation in section 9.1 highlighting its advantages for experimental researchers. Section 9.3 describes the resulting implementation in ns-3 of the designed routing protocol.

One of the particularities of the developed routing protocol in ns-3 is its continuous use of data queues to determine the next-hop for every packet being routed. In a real node, the management of data queues becomes challenging since, in addition to ns-3 data queues running in user space, there are data queues managed at the kernel space. In particular, a real node contains data queue located underneath the ns-3 routing protocol, which is running at user space. Thus, in addition to data queues in user space level (i.e., ns-3), there are data queues in kernel space. We describe the design of a mechanism based on Netlink sockets and the /sys subsystem for the appropriate interaction between the queues located in kernel space and in the ns-3 application at user space. Such integration entails an additional design and development effort not provided by the ns-3 emulation framework, however, of primal importance for any wireless routing protocol managing data queues.

9.1 Background on Emulation

Emulation capabilities have been identified as one of the techniques with a variety of applications in networking experimentation. Reference [113] discusses about it in the context of cluster testbeds, of which the EXTREME Testbed ® [114] used in this chapter is an example. The goal in this case is to emulate the behavior of wide area links in a lab, and to enable the interaction between real, emulated, and simulated networking entities in a given experiment, respectively.

In network emulation, simulated components interact with real-world protocol implementations. The focus in the *nse* [115] emulation framework of the *ns* simulator has been on network emulation. In this case our goal is to exploit the ns-3 emulation framework [102], which has been completely re-designed with respect to previous versions of the *ns* simulator. Fall [115] classifies emulation into two types. In *network emulation*, simulated components interact with real-world protocol implementations, whilst in *environment emulation*, an implementation environment is built so that real protocol implementations can be executed in a simulator. As for the former, recent work in [116] used the ns-3 emulation for evaluating network protocols for wired networks. The goal in this chapter is to use the ns-3 emulation framework, which has been completely re-designed with respect to previous versions of the *ns* simulator for the evaluation of wireless routing protocols. In contrast, in this work, initiated in [15], we want to allow for rapid prototyping of protocols for wireless networks so that the same (or almost the same) code could be run in a real wireless mesh backhaul testbed, and in the simulator. As for the latter, Network Simulation Cradle (NSC) [117] is a pioneer of introducing real world stacks (e.g., Linux stacks) into the network simulators. In particular, a framework for executing Linux kernel code in the ns-3 simulator was presented in [118].

The ns-3 emulation framework allows executing ns-3 IP stacks over real physical devices implementing layer 2 (L2) functionalities. To enable these features, the ns-3 emulation framework provides ns-3 emulated network devices that look like a usual ns-3 simulated device from the point of view of the ns-3 IP stack. As illustrated in Figure 9.1, the EmuNetDevice provides the interface to the real physical network underneath using a packet (raw) socket, hence sending/receiving packet to/from the real physical device is feasible. The EmuNetDevice uses MAC spoofing to avoid conflicts between the virtual ns-3 IP stack and real native IP stack. Therefore, packets generated with the EmuNetDevice and sent over the physical device can have assigned MAC addresses different from the those belonging to the real physical device. Additionally, as packets are received by the EmuNetDevice, which has to be configured in promiscuous mode, they are sent to the ns-3 IP stack whenever the MAC destination address corresponds to the spoofed (i.e., virtual) MAC address specified by the ns-3 EmuNetDevice. Therefore,

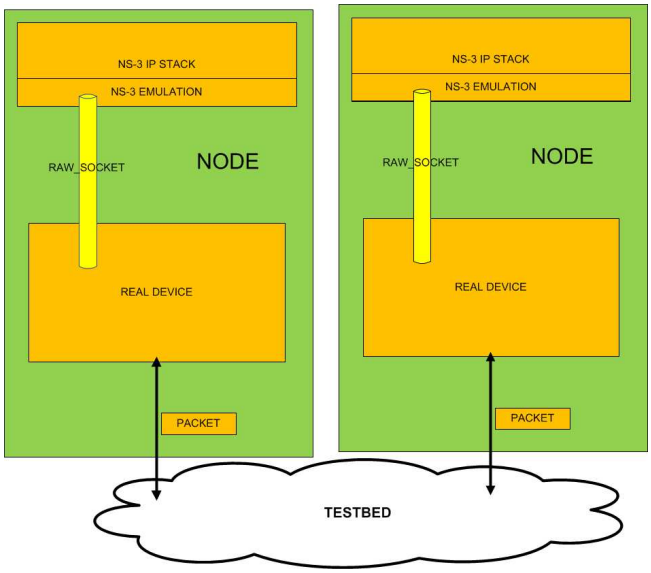


Figure 9.1: Ns-3 Emulation Framework.

to work properly, the ns-3 emulation framework requires a real L2 implementation with the associated interfaces configured in promiscuous mode. With the functionalities implemented by the EmuNetDevice and such configuration in the real physical cards, any functionality above L2 implemented in the ns-3 simulator can be tested in real networks.

9.2 Routing Protocol Building Blocks

As shown in Figure 9.2, the routing protocol, described in chapter 7 and chapter 8, implements three main building blocks, namely, the *neighbor management*, the *data queue management*, and the *next-hop determination* building blocks. The *neighbor management* building block is in charge of determining which nodes can be reached without having to cross any intermediate node. To do so, each node i exchanges HELLO packets containing queue backlog and local geographic information with its neighbors j, which belong to the 1-hop neighbor set J. Furthermore, the routing protocol implements a *next-hop determination* building block with the information provided in HELLO packets. The *next-hop determination* building block is in charge of computing the forwarding paths on a per-packet basis. The protocol is distributed and quasi-stateless, since all decisions taken by the *next-hop determination* building block use the information gathered by the *neighbor management* building block and the current packet being routed.

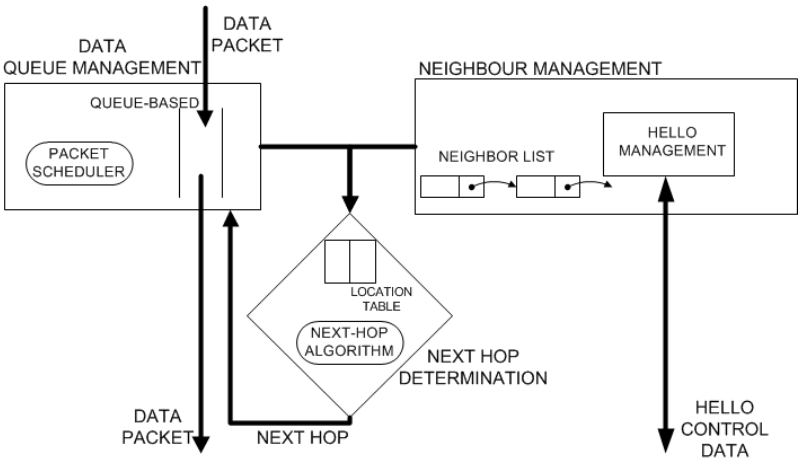


Figure 9.2: Building Blocks of the distributed Backpressure Routing Protocol.

9.2.1 Neighbor Management Building Block

The *neighbor management* building block groups the functionality related to the process of determining which nodes can be reached by means of direct communication (i.e., without having to cross any intermediate node). It is composed of two components, namely a neighbor list, and a HELLO management entity. The HELLO management entity launches the transmission of a HELLO packet, which is generated by the ns-3 protocol using the same structure as if would be generated by the Linux kernel. The HELLO packet is used to announce node state information such as congestion level, and the geolocation of the node.

Upon the reception of HELLO packets, the neighbor list component can be filled up with the current state of the neighbor nodes. Every neighbor entry is a data structure that, among others fields, contains, its IP address, geolocation information, and queue length. The reception rate of HELLO packets from a neighbor determines the validity of the corresponding neighbor state information stored for that node. A low reception rate can turn into a neighbor entry that is no longer valid. As a result of this, that particular neighbor is not considered in the *next-hop determination* building block.

9.2.2 Data Queue Management Building Block

The *data queue management* building block determines the scheduling carried out for incoming data packets. It is subdivided into two components namely, a packet scheduler policy and a queue-based structure. The queue-based structure is the entity in which the protocol store data packets waiting till a routing decision is made for them.

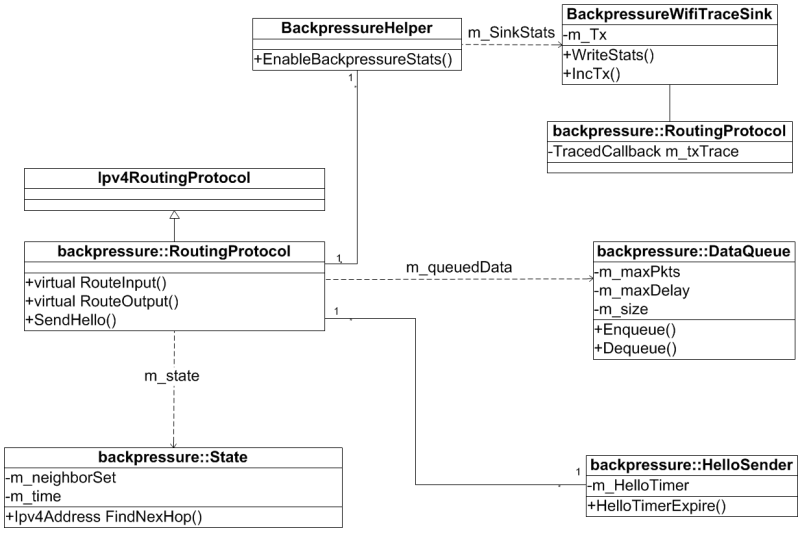


Figure 9.3: UML Diagram of the Routing Implementation.

The protocol pushes incoming packets to the queue-based structure. Once the routing decision is made, it is sent either to lower layers if the packet obtained a next-hop, or dropped when the queue-based structure is full. It is worth mentioning that one possible decision contemplated for a packet being routed can be to keep it in the local queue. Note that due to increasing congestion conditions, the protocol could prefer to take a decision that entails a null wireless resource consumption, thus, delaying its transmission.

On the other hand, we used a packet scheduler policy to assign priorities to packets stored in the queue-based structure. Thus, this priority determines the order of data packets in the queue-based structure. Whenever the *next hop determination* building block decides to forward a data packet, it calls the *data queue management* building block indicating the next hop to the first packet in the queue. Then, this data packet is extracted from the queue-based structure. The selected packet out of stored in the queue will depend on the packet scheduler policy chosen, such as, FIFO (First In First out), LIFO (Last In First Out), or WFQ (Weighted Fair Queuing). Note that, we choose FIFO as defaults queuing policy in all the experiments conducted in this dissertation.

9.2.3 Next-Hop Determination Building Block

The *next-hop determination* building block is in charge of computing the most appropriate next-hop. This building block acts on a per-packet basis, hence the expected outcome is the decision that specifies the next-hop for every data packet.

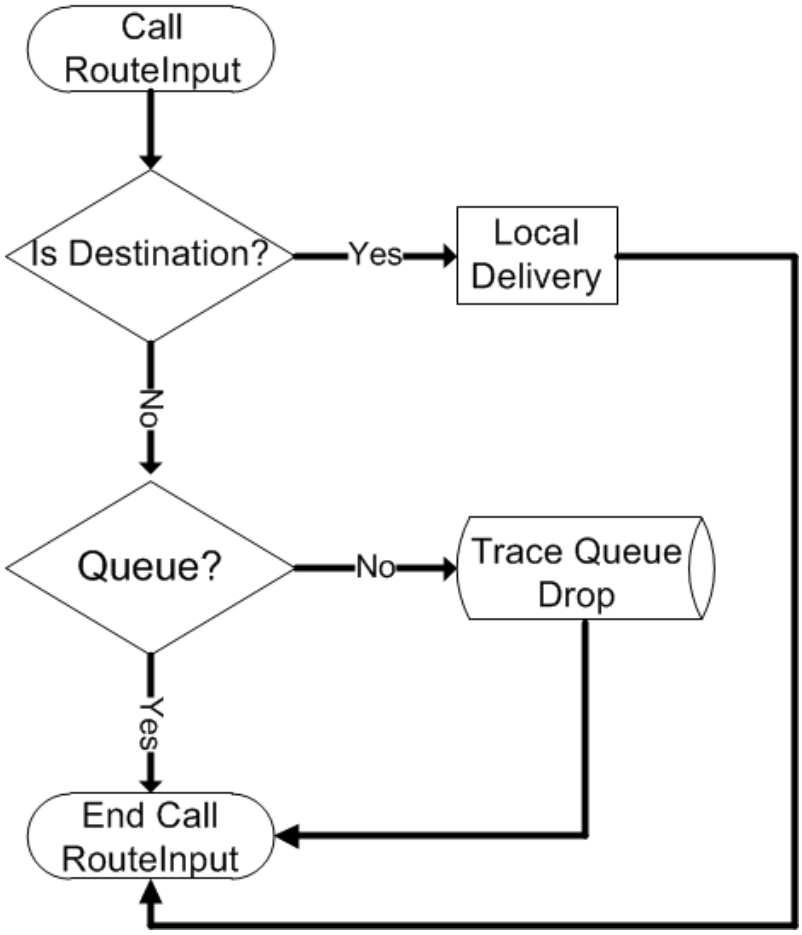


Figure 9.4: Flow Chart Route Input.

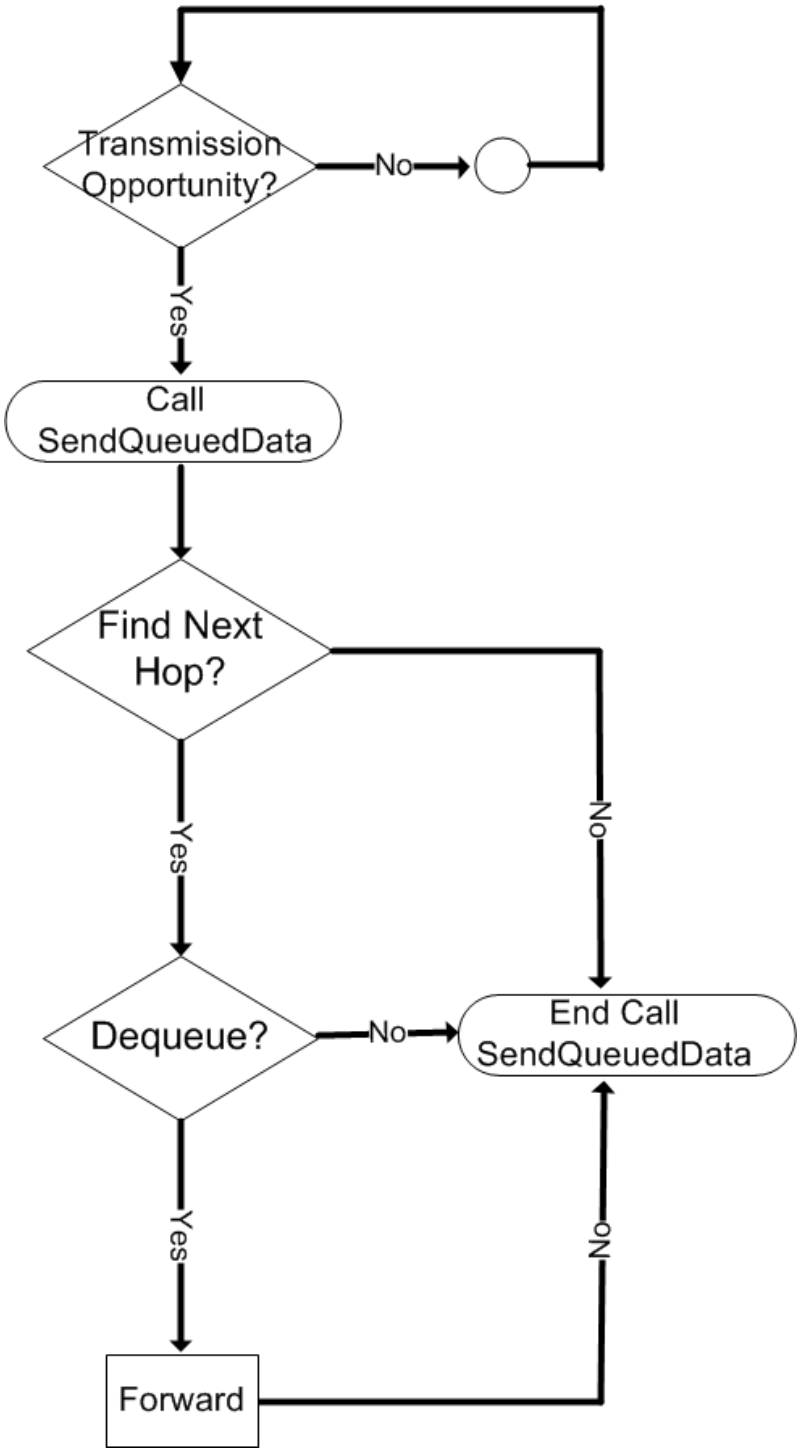


Figure 9.5: Flow Chart Tx Queued Data.

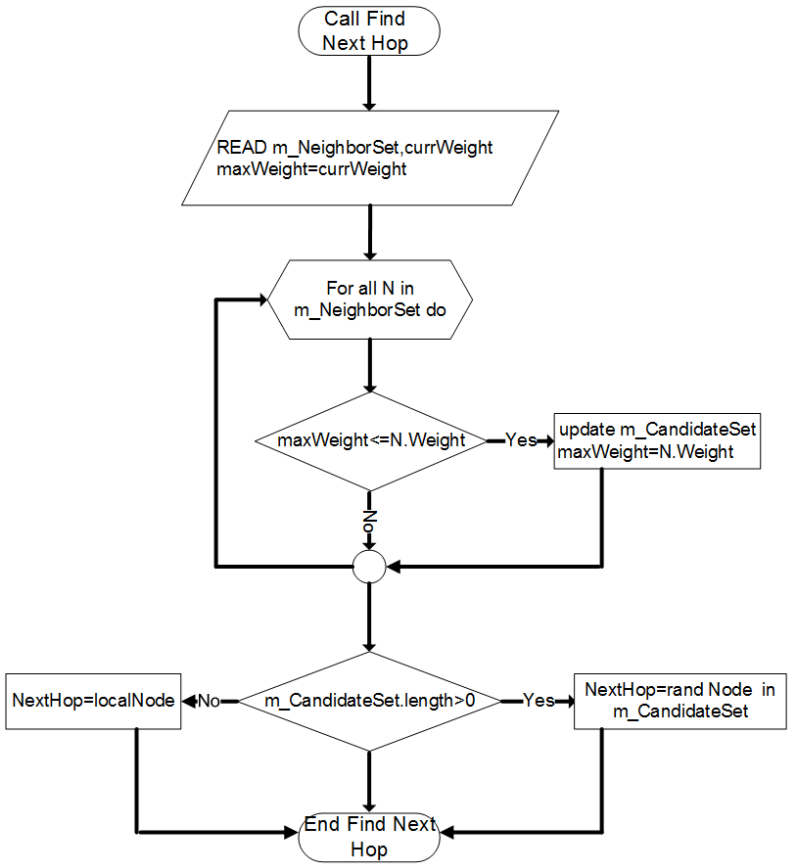
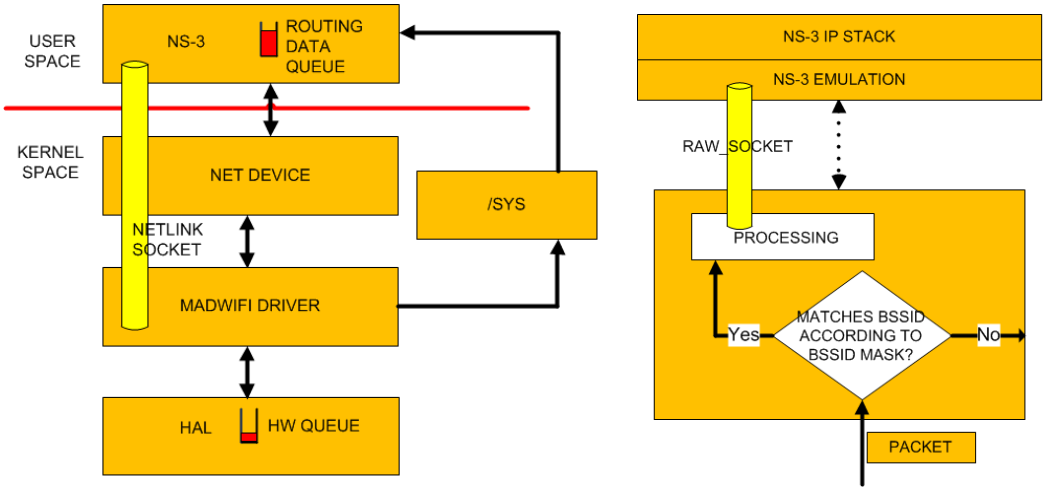


Figure 9.6: Flow Chart NextHop.

To compute forwarding decisions in every node, a weight is computed for every wireless link. The selected neighboring node j to which a packet is forwarded is the node that maximizes the link weight between the local node i and all its neighboring nodes $j \in J$. If there is no neighboring node $j \in J$ with a positive weight, the data packet is stored at the local node i waiting for better network conditions for forwarding it.

This building block requires as input, for every packet being routed, data gathered by the *data queue management* and the *neighbor management* building blocks. Precisely, it requires to compute a weight against every entry in the neighbor list maintained by the *neighbor management* building block. And it requires information of its current congestion level, information gathered from the *data queue management* building block and the location table, which contains the mapping between the IP address of a node and its geolocation in the wireless mesh backhaul. The specific method in which the algorithm computes weights w_{ij} of a link between two wireless nodes i and j is detailed in chapter 7.



(a) User/Kernel Interaction.

(b) MAC Address Reception.

Figure 9.7: Connection between ns-3 user space and kernel space.

9.3 Routing Protocol Implementation

The first subsection summarizes the main classes, implemented in ns-3 to conceive the conceptual design described in the previous section, is divided into four classes. The second subsection details the additional development efforts required for the integration of these ns-3 building blocks with the real nodes in the wireless mesh backhaul for the correct operation of the routing protocol in the testbed not implemented by the ns-3 emulation framework. In particular, the integration requires additional development in ns-3 as well as in the kernel.

9.3.1 Ns-3 Implementation

The first subsection, describes the RoutingProtocol class, which is in charge of handling inbound and outbound data packets. Additionally, its task is to initialize the routing protocol, and to inter-connect the other classes. Second subsection describes the implementation of the DataQueue class which implements the *data queue management* building block. The third subsection describes the State class which implements the *next hop determination* building block and the neighbor list component of the *neighbor management* building block. The fourth subsection describes the HelloSender class, which implements the HELLO management component of the *neighbor management* building block. Finally, fifth subsection details a block needed to gather data statistics of the routing protocol, that is, the TracingHelper class.

9.3.1.1 RoutingProtocol Class

The RouteOutput function is called every time upper layers have an outbound packet to route. Specifically, this function is merely used to send each data packet to the RouteInput function. To do this, the route returned is the local loopback route (i.e., 127.0.0.1), which triggers the reception of the packet in the RouteInput function.

The flowchart of the RouteInput operation is depicted in Figure 9.4. The RouteInput function is called in two cases. The first case occurs when a packet is received from the lower layers. The second case corresponds to those packets coming from the RouteOutput function. A node executing the RouteInput function checks whether it is the intended receiver of the packet. In this case, the packet is delivered to upper layers. Otherwise, a different treatment is required for the data packet. In this case, the RouteInput function is not examining routing tables and taking forwarding decisions. Instead of this, the protocol adds the current data packet in the data queue located at the routing layer. In the case the data queue is full, the data packet is dropped. Therefore, the RouteInput operation does not forward packets immediately. In this case, data packets wait for transmission until the SendQueuedData operation is called.

As illustrated by the flowchart in Figure 9.5, the function SendQueuedData is in charge of dequeuing data packets queued by the RouteInput function. A flowchart of this operation In addition, this function also computes the nex-hop for the packet being forwarded. This function is called whenever the MAC layer detects a new transmission opportunity for the Routing layer. This occurs when the MAC layer has transmitted a new data packet. Thus, whenever, the MAC senses an idle medium and has no packet transmits, it triggers the routing protocol operation SendQueuedData. In this way, the protocol maxi-

mizes the number of packet forwarding opportunities. As the protocol is also able to decide whether to transmit or not (i.e., scheduling), the number of forwarding opportunities per each packets cannot be upper bounded by one, since it would potentially imply packets unnecessarily trapped at data queues. Finally, the FindNextHop, called within SendQueuedData, (defined in State class) is in charge of deciding whether a data packet is transmitted or not, and the specific next hop. A flowchart of FindNextHop is provided in Figure 9.6.

9.3.1.2 DataQueue Class

The DataQueue class currently implements packet scheduling policy of the data queue at the IP layer. More specifically, the packet scheduling policy currently implemented is FIFO policy. The class is mainly composed of two main operations, namely the Enqueue and the Dequeue functions. Finally, the Cleanup function, common to both Enqueue and Dequeue functions, is also required. The Enqueue function is called within the RouteInput function (defined in the RoutingProtocol class). It is in charge of queueing and assigning a given priority to incoming data packets. The Enqueue function first checks whether the FIFO queue is full or not by checking if its queue size m_size is lower than $maxPkts$. As flowchart shown in Figure 9.4 illustrates, if the queue is full the packet is dropped. In particular, given its relevance, this event is captured by the tracing subsystem which is described in next section. Otherwise, the packet is queued at the last position in the data queue, hence following the FIFO policy. Additionally, and jointly with the data packet, the FIFO queue also stores the timestamp corresponding to the queuing event in order to be used in the Cleanup function. The Dequeue function is called within the SendQueuedData function (see flowchart in Figure 9.5). It is in charge of dequeuing data packets queued by the Enqueue function.

This function is called whenever there is a dequeuing opportunity. The dequeued packet is the first one stored in the queue, hence following the FIFO policy. Whenever the Enqueue and the Dequeue functions are called, the Cleanup function is also called in order to drop data packets that have been waiting for the FIFO data queue more than maxDelay time. Precisely, to obtain this queuing delay, the instant the packet was queued is subtracted from Simulator::Now (i.e., the current instant). Consequently, data packets can potentially be dropped, even when the data queue is not full. These queue drops would be indicators of low transmission opportunities, but not necessarily of excessive data queue lengths.

9.3.1.3 State Class

The State class is in charge of taking local routing decisions for every data packet traversing this node, and storing the necessary routing state to take these decisions. The information required to take routing decisions is locally generated. Figure 9.6 describes the FindNextHop function which decides the next hop to forward a data packet. The function FindNextHop requires as input $m\_NeighborSet$ and $m\_currWeight$. The first parameter, $m\_NeighborSet$, is the list of current neighbors of the node. Every neighbor entry is a structure that, among other fields, contains the IP and MAC address and queue length of a certain neighbor. Current neighbors are those for which recent information about queue lengths has been received. More precisely, they are those neighbors from which a HELLO packet has been received during the last interval of duration m_time. The second parameter, $m\_currWeight$, is the current weight. The variable $m\_currWeight$ keeps the maximum weight calculated in the current the forwarding decision.

The current implemented policy for taking local routing decisions consists of going through all the entries in $m\_NeighborSet$, and selecting the more appropriate neighbors as candidates (i.e., $m\_CandidateSet$ in flowchart in Figure 9.6). The candidate neighbor list is filled in with those nodes that form wireless links with the current node of maximum weight. All neighbors with a weight lower than $maxWeight$ are discarded for acting as next hop of the current packet being routed. However, if the weight of the current link is strictly bigger than $maxWeight$, all the previous neighbors entries in $m\_CandidateSet$ are deleted, and the current neighbor is included in $m\_CandidateSet$. And if the weight of the current computed link is equal to $maxWeight$, $m\_CandidateSet$ is updated by adding the current neighbor to the list of candidates. Thus, note that once all neighbors in NeighborList are evaluated, more than one link could have an equivalent maximum weight. This implies that there is more than one neighbor being an appropriate candidate to forward the packet. This explains the maintenance of a list of candidates, rather than maintaining only one candidate.

Therefore, to distribute traffic amongst all the different candidates, we use a random selection policy in case there is more than on neighbor having the lowest queue length. On the other hand, if there is no neighbor having a lower queue length, there is no data packet transmission, and so, the local loopback IP address (i.e., 127.0.0.1) is returned. Otherwise, the IP address in the selected neighbor entry is returned. In any case, the routing protocol is also doing some kind of distributed scheduling of the transmission of packets throughout the network by taking local decisions.

9.3.1.4 HelloSender Class

The HelloTimerExpires function triggers the transmission of HELLO This section provides a description of the tracing framework to do packets. When the HelloTimerExpires function is called, the the debugging and evaluation of the routing protocol. The first SendHello function within the RoutingProtocol class associated to subsection describes BackpressureHelper, a module able to write the HelloSender class is invoked. In addition, when a HELLO packet tracing information for each node running the protocol and for is received by the RoutingProtocol the HELLO packet is also passed each data packet injected. The second subsection summarizes the to the HelloSender class. use of existing AthStatsHelper module provided by the official ns-3 code release. Finally, the third subsection describes how some global routing metrics are reported by the main simulation program.

The final goal of handling the sending and reception of HELLO packets by means of the HelloSender class from the RoutingProtocol class is to allow the maintenance of wireless link quality metrics. Wireless link quality metrics can be used as penalty functions for the computation of the backpressure weights [76]. For instance, ETX [39] is a wireless link quality metric based on calculating packet delivery ratio during a window interval. Then, when the HELLO timer expires, apart from calling the SendHello of the RoutingProtocol class, the packet delivery ratio calculated is sent as a parameter to the RoutingProtocol. Then, the link quality metric information (in this case, the packet delivery ratio) is added to the payload of the HELLO packet in the RoutingProtocol class.

9.3.1.5 TracingHelper Class

The BackpressureHelper module enables the activation of specific statistics (e.g., queue lengths, packets transmitted, and queue drops) at the node level. These metrics are useful to understand the behavior of the dynamic backpressure routing protocol. Therefore, they are helpful to debug the routing protocol in order to detect the causes of performance degradation of typical routing metrics (e.g., throughput, delay). Precisely, for each node in the network we collect the following metrics: the queue length, the number of queue overflows, the number of data packets transmitted, and the number of times the TTL of packets sent through the network has expired. On the other hand, the module also provides per-packet level tracing. It can be useful to capture information about the meta-data associated to a packet (e.g., the packet sequence number) traversing a node. Furthermore, the source and destination addresses can be used as parameters to filter the IP packets that can be traced. For instance, this can be used to obtain the list of nodes traversed by every packet belonging to a specific flow. Specifically, the packet sequence number, the timestamp, the source address, and destination address is some of the information which

can be reported. Additionally, the per-node as well as per-packet level metrics are collected by means of TracedCallbacks.

9.3.2 Integration of the Ns-3 Routing Protocol in the Wireless Mesh Backhaul Testbed

The ns-3 distributed backpressure routing protocol requires additional interactions with the kernel than that proposed by the ns-3 emulation framework for its experimental evaluation. Specifically, we introduce two additional communication channels between ns-3 at user space and kernel space to make the ns-3 backpressure routing implementation aware of the data queue present at kernel space.

Synchronous read of the WiFi MAC Queue Size: In a node running the ns-3 routing code through the ns-3 emulation functionality, the location of data packets is shared between the WiFi hardware queue in kernel space (see Figure 9.7(a)), and a data queue in user space (i.e., in the ns-3 IP stack). Two building blocks require the synchronous read of the total (i.e., ns-3 plus WiFi MAC data queue sizes) data queue size in a node. The *neighbor management* building block requires the advertisement of the number of data packets accumulated in a node whenever it sends a HELLO message. On the other hand, the *next-hop determination* building block must also be aware of the current queue length since the protocol computes forwarding decisions using queue sizes.

To determine the number of data packets accumulated in the WiFi hardware queue, the *neighbor management* and *next-hop determination* building block perform a synchronous read of a file entry available via the Linux /sys file system whenever the *neighbor management* sends a HELLO a message, or the *next-hop determination* requires to re-compute the weight for another packet. The MadWiFi driver maintains accurate information of the WiFi hardware queue size. We extend the MadWiFi driver so that an entry in the /sys file system maintains the current WiFi hardware queue size so that it could be read by the ns-3 routing protocol at user space.

Asynchronous interaction to regulate of Forwarding Opportunities: The WiFi MAC queue contains those data packets with an already selected next-hop by the *next-hop determination* building block. The ns-3 routing data queue accumulates all the data packets which still do not have a next-hop. However under high congestion conditions WiFi hardware queues can become full, leading to queue overflows at the kernel level. The ns-3 routing protocol incorporates the possibility of tracing queue drops occurring at the ns-3 application. However, the ns-3 application cannot trace queue drops occurring at the kernel level. In order to reuse ns-3 tracing capabilities, we introduce a mechanism to regulate the forwarding opportunities at the ns-3 user space so that queue drops can never occur at the kernel level. As Figure 9.7(a) depicts, this is attained with the use of an asynchronous communication method based on

Netlink sockets.

To keep potential queue drops at the ns-3 space, the routing protocol stops the operation of the *next-hop determination* building block whenever the WiFi hardware queue is full, hence accumulating data packets without a next-hop in the ns-3 routing data queue (see Figure 9.7(a)). The kernel sends an asynchronous message through Netlink sockets to the ns-3 routing protocol running in user space whenever the hardware queue is full. On the other hand, it also provides a message through Netlink sockets to the ns-3 space indicating the ns-3 routing queue that the hardware queue is ready to restart the injection of data packets. The ns-3 routing protocol implements a separate thread that keeps track of the queue state in the WiFi card indicating whether the hardware queue is full or not. This event happens after detecting a non-full WiFi hardware queue. The specific WiFi hardware queue threshold to restart the injection of packets between user and kernel space is a configuration parameter of the routing protocol. Additionally, the ns-3 routing data queue is periodically polled to serve packets to the hardware queue. This avoids keeping data packets trapped at routing data queues since packets might not be serviced to the hardware queue at exactly the the same pace they arrive to the routing data queue.

Switch ON/OFF the Wireless Backhaul interface: The evaluation of this concept requires some changes in the way the ns-3 emulator works. Specifically, the sockets created from the ns-3 application and associated to the active interfaces should be maintained even when the interface is switched off. Thus, the socket can send/receive data packet once the interface is switched on. We modified the management of the physical interfaces within the ns-3 emulation framework so that when an interface composing the WiFi backhaul is switched off it continues working. However, since the interface is turned down no data packets can be transmitter nor received over the air. Once the interface is switched the behavior of the ns-3 is restored. Therefore, the binding between the EmuNetDevice and the real interface is appropriately maintained in spite of the real interface status (up or down).

Chapter 10

Experimental Results

This chapter provides the experimental evaluation of our distributed backpressure routing protocol in an all-wireless Network of Small Cells prototype. The contributions in this chapter have been published in [15] and submitted to [16]. Previous chapter detailed the implementation in ns-3, as well as the integration of the protocol in the ns-3 emulation framework. Section 10.1 describes the experimental setup of the 12-node indoor all-wireless NoS, and how we perform the process of experimentation and data gathering. Section 10.2 provides the results obtained with the 12-node all-wireless NoS that validate the proper operation and functionality claimed in the design of the routing protocol in Chapter 7, and demonstrated by simulation in chapter 8. In summary, evaluation results characterize the 12-node all-wireless NoS and demonstrate the correct operation of the routing protocol in a wide variety of circumstances with regards to the mesh backhaul configuration. In particular, we conduct some initial experiments without the intervention of the routing protocol to characterize the all-wireless NoS testbed environment by evaluating the quality of its WiFi backhaul links. The first set of experiments evaluating our routing scheme, showed in subsection 10.2.3, validates the correct operation of the distributed backpressure routing protocol, and studies whether the ns-3 emulation framework may introduce throughput degradation or not in the testbed. We also illustrate the impact of the V configuration parameter in the load balancing capabilities of the routing protocol. The second set of experiments in subsection 10.2.4,

demonstrates the operation of our scheme under varying wireless backhaul topology conditions (i.e., a SC being periodically being periodically switched on and off). This set of experiments validates the adaptability claimed by the routing protocol by simulation in chapter 8 under dynamic wireless mesh backhaul deployments. We finally conclude this chapter in section 11.1.

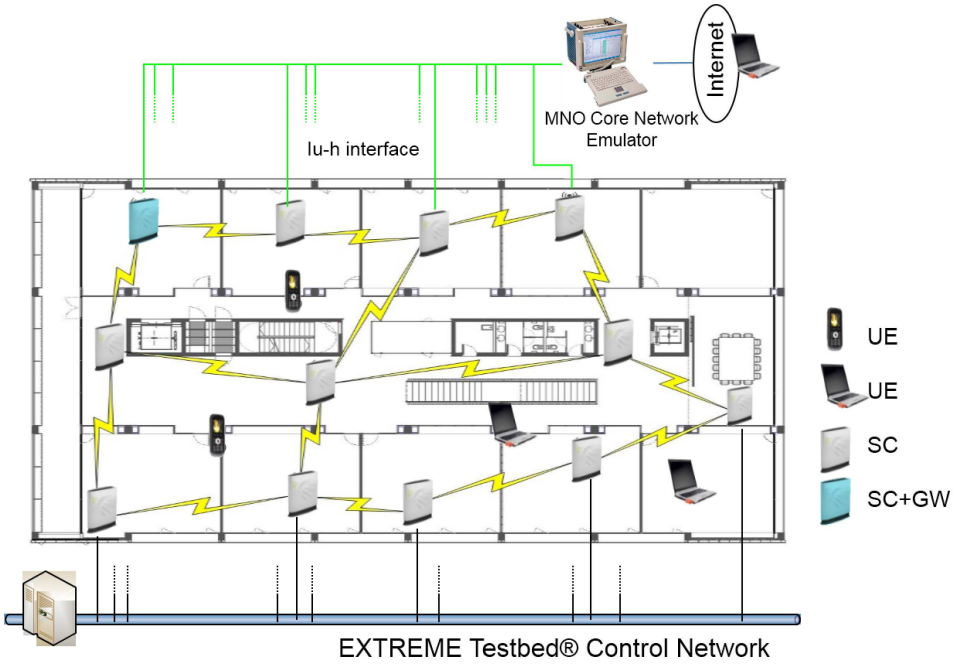


Figure 10.1: Architecture of the NoS testbed.

10.1 All-wireless NoS testbed

This section first describes the indoor all-wireless NoS [12] developed in the first floor of the CTTC building. Then, we detail the configuration and lessons learned during the deployment of the testbed. Finally, we describe the experiment setup and how to gather results of our prototype as well.

10.1.1 Description

Our main goal is the evaluation of the TNL in the NoS testbed illustrated in Figure 10.1. In particular, we aim to test the functionalities of our distributed backpressure routing protocol, presented in [6, 7]. With regards to the setup of experiments within the TNL, we highlight the following features. A key aspect in the TNL of our proof-of-concept all-wireless NoS showed in Figure 10.1 is to evaluate the functionalities of the TNL routing protocol designed for the wireless mesh backhaul, disregarding its configuration.

Note that performance goals are out of the scope in this evaluation. Thus, we opt for a default single-radio singe-channel WiFi mesh backhaul without any additional specific tuning or calibration (e.g., power calibration, or addition of several channels to increase capacity). We demonstrate that the targeted TNL setup is sufficient to demonstrate the functionalities and properties demonstrated by simulation in [6, 7]. The main entities, illustrated in Figure 10.2, enabling the indoor all-wireless NoS proof-of-concept follow:

- **Small Cell**: The Small Cells (SC) are based on two components. One component is the 3G Sagemcom SC implementing an Iuh compliant interface to enable interoperability with the interface implemented by the emulated core network. The 3G SC is connected via Ethernet to a WiFi mesh router to build the WiFi mesh backhaul. This a mesh router merely entailing TNL functionalities based on a mini-ITX board (based on Pentium M 1.6GHz) that endows up to four CM9 WiFi cards (802.11abg) for building the WiFi mesh backhaul. For a more proper description of the WiFi mesh routers, we refer the reader to [14]. Thus the mesh routers build the Transport Network Layer, whereas the SCs include both MNL and TNL functionalities. These two components are connected through an Ethernet interface.
- **Core Network**: The G35 Core Network Emulator implements the Iuh interface to which the 3G SC, which also implement the Iuh interface, can connect. It includes the MNL functionalities of the core network.
- **TNL GW**: One of the SCs acts also as a TNL GW [6]. This component is in charge of pulling packets from the NoS, and direct them towards the G35 core network emulator.
- **User Equipment**: The User Equipment (UE) are based on laptops equipped with an UMTS PCMCIA card so that they can be attached to the 3G Sagemcom SCs.

(a) Core Network Emulator (b) Small Cell (c) User Equipment

Figure 10.2: Main entities of the all-wireless NoS testbed.

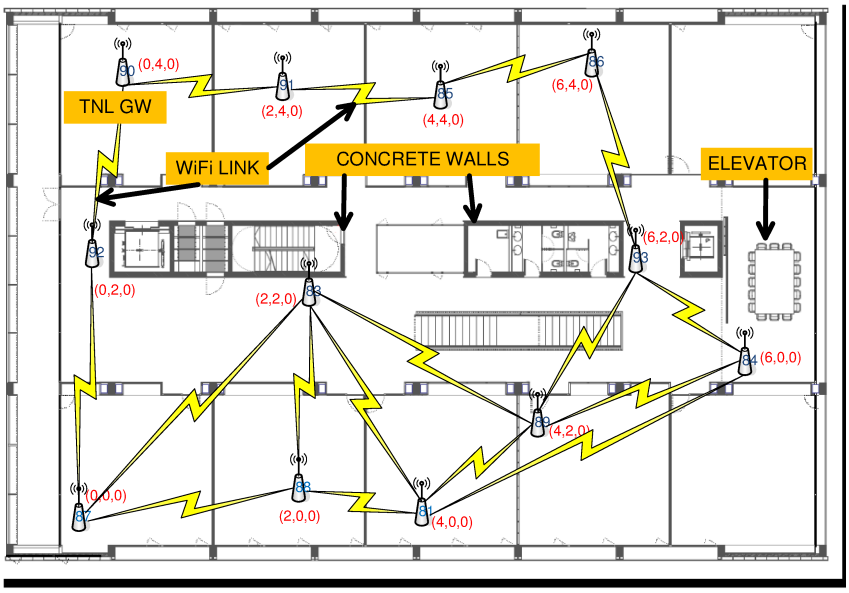


Figure 10.3: WiFi-based Mesh Backhaul Testbed.

The most relevant part for the evaluation conducted in this chapter is the wireless TNL, composed by the mesh routers described above. In particular, we build a WiFi mesh backhaul amongst 12 SCs over an approximate indoor area of 1200 square meters (see Figure 10.3) in the first floor of the CTTC building. Each of the SCs composing the testbed are static and non-power constrained. The SCs are connected to a central server for management and maintenance purposes by means of wired Ethernet within the EXTREME Testbed ® [114]. Through an Ethernet connection, we extend each SC so that it also entails a mini-ITX (Pentium M 1.6 GHz) small form-factor PC. These PC can mount up to four CM9 wireless cards (802.11abg) with the last version of MadWiFi driver (i.e., v0.9.4), though, for the lack of simplicity and rapid generation of results evaluating the protocol, in our wireless mesh backhaul setup we use one WiFi card per SC.

10.1.2 Configuration of the WiFi Mesh Backhaul Testbed

For the validation of the distributed routing policy, we configure one WiFi card per mesh router to act as wireless link belonging to the backhaul network. On top of the Ethernet (directly connected to an SC) and WiFi interfaces, we associate an instance of an ns-3 stack, by means of the EmuNetDevice [102]. We associate to every interface a different MAC destination address from that corresponding in the physical device. All the WiFi interfaces composing the mesh backhaul are configured to the band of 5Ghz, and they have assigned the same channel. We configure the wireless mesh backhaul links in the 5Ghz band to avoid external interference from the usual WiFi LAN service of the CTTC building, and neighboring

buildings so that the experiments can be carried out without taking into account external interference sources, which usually use the 2.4Ghz frequency band for its production WLAN. On the other hand, the Ethernet interface in each of the different nodes (UEs, SC, and Wireless mesh router) has management purposes. Besides, it also acts as a point to point connection between SC and Wireless mesh router as well. We highlight the following features with regards to the configuration of the WiFi-based cards.

On the spoofed MAC address: The MadWiFi drivers allow the creation of multiple virtual wireless interfaces that can be configured independently. One lesson learned from using the ns-3 emulation framework running on top of a WiFi-based testbed is the importance in the specification of the MAC address associated to the ns-3 EmuNetDevice class. In the MadWiFi driver, the Basic Service Set Identifier (BSSID) mask specifies the common bits that a MAC address must match in order to process a receiving packet. Therefore, the spoofed MAC address associated to the EmuNetDevice object in the ns-3 emulator must comply the restrictions imposed by the BSSID mask specified in the MadWiFi driver, whereas an Ethernet card does not pose such restrictions. Otherwise, as illustrated in Figure 9.7(b) packets received by the MadWiFi driver with a MAC address not following BSSID mask restrictions are discarded by the WiFi card. Thus, they cannot be captured by the packet (raw) socket in the ns-3 class defined by the EmuNetDevice. In our case, the mask allows specifying a different MAC address to the ns-3 emulated device by changing the most significant bits of the usual MAC identifier.

On the WiFi cards: Every ns-3 node runs on top of a virtual interface (in our Linux setup labeled as ath0 interface) associated to one physical WiFi card, labeled as Wifi0 interface. A second WiFi virtual interface (in our specific Linux setup labeled as ath1 interface) associated to the same physical WiFi card (e.g., Wifi0) is configured in each node in monitor mode for tracing purposes. As the 2.4GHz frequency band is crowded in the CTTC building, we use channels in the 5GHz band. In addition to building obstacles (e.g., brick walls, plants, elevator), there are mobile obstacles during working hours (e.g., workers moving inside the building). Additionally, they have assigned the same channel and are configured to the maximum transmission power. This configuration allows most nodes to be in carrier sense range of each other, hence minimizing interference due to hidden nodes which is beyond the scope of this evaluation.

On guaranteeing multiple hops in a confined space: To evaluate the proposed routing protocol, a primary aspect is its behavior in multihop layer 3 topologies. However, in our testbed setup practically all the nodes configured to the same channel and maximum transmission power are in carrier sense range since the testbed is deployed in a confined space of around 1200 square meters. In fact, the layer 3 (L3) topology is determined by the set of nodes receiving HELLO messages. After an initial set of experiments, we observe that, in a so confined space, HELLO broadcast messages sent by every node are

received by almost all the nodes in the testbed. The reason for this is that HELLO broadcast messages are usually transmitted at the minimum data rate allowed by the WiFi card (in this case, 6Mbps).

Using the lowest rate for HELLO messages leads to a high decoding probability at all nodes in such a confined space. To establish L3 multihop topologies under this configuration setup, we extend the MadWiFi driver so that one can increase the physical rate at which broadcast HELLO messages can be transmitted. After a set of empirical tests, we found that increasing the physical data rate of broadcast packets to 54Mbps is an appropriate choice to have multihop L3 topologies given the defined testbed deployment and power setup. Another solution would be to decrease the transmission power of every node. However, though reducing the transmission power would create an actual multihop network, it would also lead to an increase of interference due to hidden nodes which is beyond of the scope of this evaluation. In Figure 10.3, we show the resulting L3 connectivity patterns when sending HELLO broadcast messages at a physical rate of 54Mbps. With this HELLO broadcast rate configuration, every node has at least two direct neighbors and no more than four, with a high probability. Thus, we can evaluate forwarding paths that are up to 4 hops long.

On the coordinate assignment: The routing protocol needs location information in order to compute the weights of every link. To do so, the testbed requires an assignment of coordinates. Geographic coordinates are statically assigned so as to form a 4x3 grid (see Figure 10.3) with a step of size 2.

10.1.3 Experiments and Gathering of Results

In this subsection we outline the launch of an experiment, and also the collection of data in order to get network performance metrics over the testbed.

Launch of an experiment: Experiment automation benefits from the capabilities of the EXTREME Testbed ® [114]. Experiments are launched from a central server, which is directly connected to the testbed via Ethernet through several switches. This central server is in charge of executing EMMA [114], the tool of the framework that we use to describes the execution of an experiment in the testbed. The configuration of the tool is based on XML. For the experiments carried out in this evaluation, we use EMMA features to configure the ns-3 application in every mesh node. Moreover, EMMA allows running the same experiment with a different input parameter (e.g., the V parameter, the experiment duration, or the traffic volume generated in case the node is a sender), or to repeat the same experiment in order to get statistically significant results.

Gathering of results: Exploiting the EMMA features, we remotely specify a WiFi virtual device in monitor mode associated to the physical device used to send/receive packets in every node. In this way,

a pcap trace file is generated at every node in the testbed. Furthermore, the central server gathers the results generated by the ns-3 application running in every mesh node (e.g., data queue length, queue overflows, and packets sent to the WiFi MAC queue).

10.2 Testbed Results

This section is devoted to evaluate the implementation of the routing protocol in the WiFi mesh testbed. The first set of experiments is devoted to validate the correct operation of the routing protocol under static wireless backhaul conditions, and study whether the ns-3 emulation framework may introduce throughput degradation or not in the testbed. The second set of experiments illustrates the advantages of our variable-V algorithm presented in [6] under dynamic wireless backhaul conditions. Subsection 10.2.1 characterizes the WiFi mesh backhaul testbed under evaluation, and subsection 10.2.2 describes the followed methodology. In subsections 10.2.3 and 10.2.4, we provide the results and main observations for static and dynamic wireless mesh backhaul conditions, respectively.

10.2.1 WiFi Mesh Backhaul Characterization

Prior to evaluate our routing scheme, we need to characterize the testbed environment by evaluating the quality of the WiFi links under different real conditions. To assess the quality of the WiFi links in the testbed, we use HELLO messages generated by the ns-3 distributed routing protocol every 100ms. We measure the quality of WiFi links by calculating the Packet Delivery Ratio (PDR) of HELLO messages sent at a fixed rate of 54Mbps between each pair of SCs. HELLO messages determine which SCs are within direct communication, and so the neighbor SCs at which one SC can directly transmit a data packet.

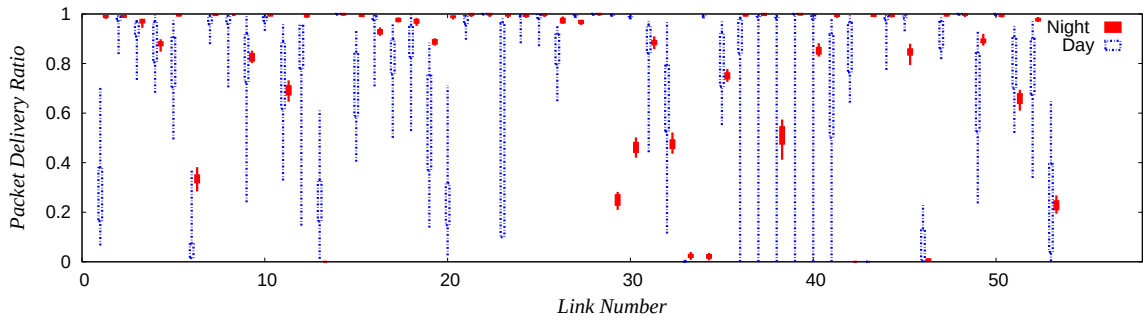


Figure 10.4: Wireless Link Quality: Day (i.e., Working) Hours vs. Night Hours.

HELLO messages are sent at the maximum rate (i.e., 54Mbps) not only to maximize the number of hops in a confined space, but because they also maximize the rate at which neighboring SCs can decode received data packets (i.e., those SC neighbors usually reachable at a HELLO data rate of 54Mbps). For data packets we use the SampleRate autorate algorithm. In the SampleRate autorate algorithm, the rate is selected as a function of the frame losses experienced by the WiFi link at every probing rate the algorithm tries. Hence, the physical data rate is decreased if losses are experienced, and increased if no losses are experienced at a given rate.

The set of experiments that characterize our WiFi mesh testbed evaluate the PDR of WiFi links in two different time frames. The first time frame starts at working hours (i.e., 11AM), whilst the second time frame starts at night hours (i.e., at 10PM without workers in the building). For each time frame under evaluation, the testbed calculates the average PDR of HELLO during 100 seconds. The testbed repeats this experiment 15 times, with a pause time of 100 seconds between every replication of the experiment.

One observation characterizing WiFi links in the testbed is the impact of the time of day in the wireless link quality. We observe that WiFi links during day hours are more unstable than during night hours. Precisely, 57 links during the night, and up to 69 lossy links with a higher degree of dynamics were observed during the day for the 15 replications. The more variable environmental conditions during working can even help some packets reach more distant SCs through direct communication. Such dynamics lead to the generation of more intermittent links since just a 36% of the wireless links are present in all the 15 replications carried out during working hours. Figure 10.4 plots the subset of WiFi links with a PDR bigger than zero during the 15 replications of the experiment in at least one of the two times frames (i.e., day and night) under evaluation. Thus, we compare the PDR of those selected WiFi links in both time frames. During working hours the reported PDR of WiFi links shows a high degree of variability, whilst during night hours WiFi links show more stability as shown by the size boxplots in Figure 10.4.

As expected from previous work [119, 120], since during night hours there are no people in the building, the environmental conditions can be considered constant during the 15 replications. As a result of this, we can conclude that no matter the PDR shown by a WiFi link (i.e., good or bad), link quality shows a high degree of stability.

10.2.2 Methodology

In both set static and dynamic set of experiments, the traffic injected consists of unidirectional UDP with maximum packet size (i.e., 1472 bytes) generated by the ns-3 OnOff application. We execute the

experiments at night hours, and the duration of each experiment is of 80 seconds if not said differently. The data queue size limit denoted by Q_{max} in each SC is of 200 packets. With such conditions, a V parameter bigger or equal than 200 in each SC is equivalent to always use the shortest path in terms of Euclidean distance. On the other hand, the lower the V parameter (i.e., from 200 up to 0), the bigger the degree of load balancing offered by the backpressure routing protocol. If not said differently, the link data rate all the WiFi cards use is the SampleRate autorate algorithm, and the variable-V self-organized controller detailed in [7] is used. This self-organized controller is in charge of determining the more appropriate trade-off between load balancing and proximity to the destination.

10.2.3 Static Wireless Mesh Backhaul Results

This subsection evaluates the distributed backpressure routing protocol running on top the ns-3 emulation framework in the testbed. The first subset of experiments injects a single-flow in a 1-hop and in a 2-hop path, showing whether the ns-3 emulation framework introduces any performance degradation during packet generation, forwarding, and reception. The second subset of experiments evaluates the routing protocol with different flows injected in the testbed at a different number of hops from the destination.

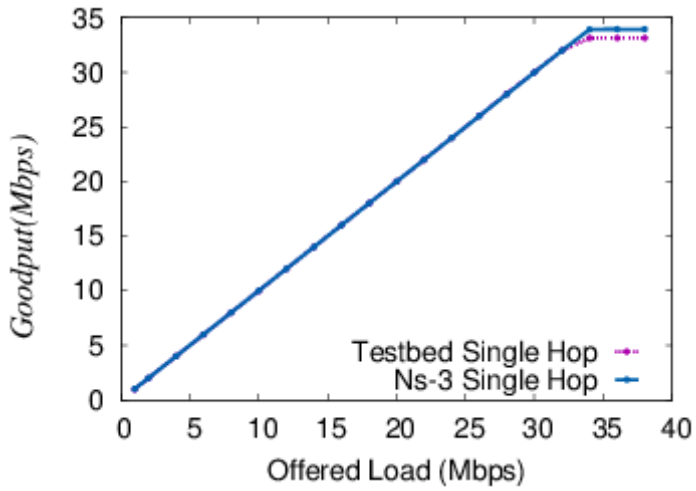


Figure 10.5: Reference Scenario to evaluate the performance of the ns-3 emulation framework: 1-hop case.

Single Flow: In this specific case, we inject one single traffic flow in the NoS. Regarding the configuration of the routing protocol, the V parameter is bigger than the data queue size limit of nodes (i.e., 200 packets). Therefore, the protocol is configured to take forwarding decisions prioritizing the penalty function, based on geolocation information.

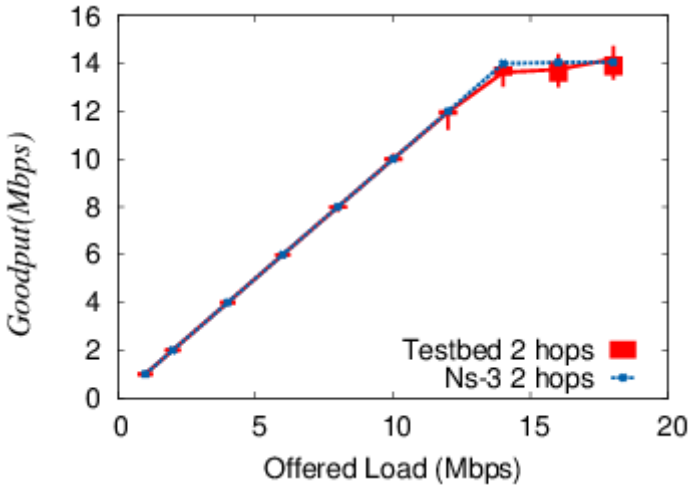


Figure 10.6: Reference Scenario to evaluate the performance of the ns-3 emulation framework: 2-hop case.

The testbed runs two different experiments 20 times. The first experiment measures the achieved goodput in the receiver at a 1-hop distance from the source. In Figure 10.5, goodput results with the ns-3 emulator running in the testbed match those ones obtained with the ns-3 network simulator using the realistic Friis physical layer propagation model (i.e., around 33Mbps). Therefore, in terms of packet generation and packet reception, the ns-3 emulation framework does not introduce throughput degradation when even saturating WiFi cards at the generation level.

The second experiment measures the achieved goodput in the receiver at a distance of 2-hops from the source. The source injects various traffic volumes ranging from from 1 to 18Mbps. As can be shown in Figure 10.6, though there is higher variability in the testbed (up to 1Mbps), goodput results are also similar to those obtained with the ns-3 network simulator. These results validate the operation of several routing protocol functionalities such as HELLO transmission/reception, neighbor management, and forwarding in a real wireless environment using ns-3 emulation. Additionally, we observe that the ns-3 emulation framework does not cause degradation in terms of throughput since results practically coincide with those obtained with the ns-3 simulator.

Multiple Flows: We inject 2,4, and 6 UDP traffic flows of 1Mbps using random SCs sources, which are different from the unique destination of all the flows, that is, the TNL GW. Since our interest lays on measuring impact of the WiFi link rate configured in the backhaul, we compare the goodput results obtained with different WiFi link rate configurations in the testbed, namely a fixed rate of 36Mbps, 54Mbps, and the SampleRate [121] autorate algorithm. Furthermore, for every WiFi link rate under evaluation, we repeat the experiment 16 times. As depicted in Figure 10.7, we observe that, no matter

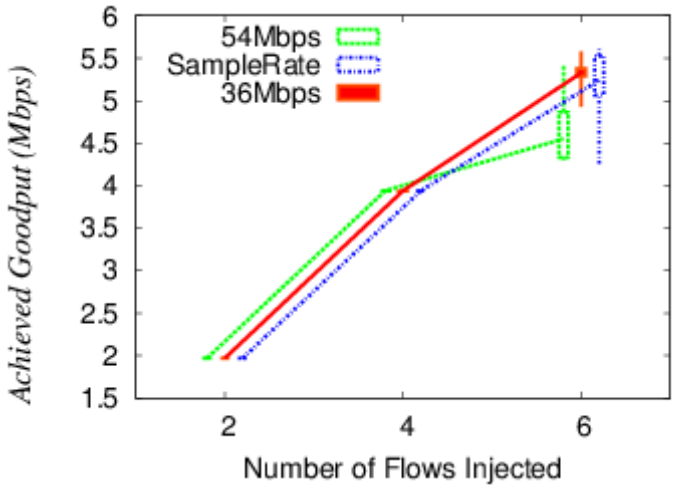


Figure 10.7: Achieved Goodput at different WiFi link rates.

the WiFi link rate configuration, the injection of two or four flows does not influence the achieved goodput at the destination. In both cases, the offered load is equivalent to the achieved goodput at the destination. However, the backhaul is unable to serve 6 traffic flows, since there is a significant mismatch between the workload (i.e., 6Mbps) and the achieved goodput measured at the receiver. Thus, with the current setup, the backhaul already reaches an upper bound in terms of goodput. We analyzed the results both with the variable-V and also a fixed V configuration parameter, and in this case results showed that with a V parameter configured to 100 in every SC, results obtained with the variable-V were equivalent.

Figure 10.7 reveals that, for the a workload of 6Mbps, the specific WiFi link rate configuration has a significant impact on the achieved goodput. In this case, significant goodput results with all the backhaul links configured to a data rate of 36Mbps. Indeed, Figure 10.8 confirms that the predominant WiFi link rate chosen by the SampleRate autorate algorithm is 36Mbps. On the other hand, Figure 10.7 also shows that the autorate algorithm experiences the higher variability due to the autorate configuration experienced by all WiFi cards. This is because of the way the SampleRate uses probe packets to calculate the rate. Precisely, it uses every tenth packet sent as a probe packet, and chooses the maximum rates depending on the losses experienced by these probe packets. Therefore, the Sample rate intends to estimate the maximum link rate experiencing a high reception probability in the backhaul testbed. Interestingly, the attained goodput is lower at a fixed rate of 54Mbps than that attained with a fixed rate of 36Mbps. As Figure 10.7 depicts, this is due to the decreasing packet decoding probability with the increase in the physical data rate. In the case in which all WiFi cards are configured to 36Mbps, results show the maximum goodput results since the more appropriate rate reported by the Sample autorate algorithm is used in the experiment.

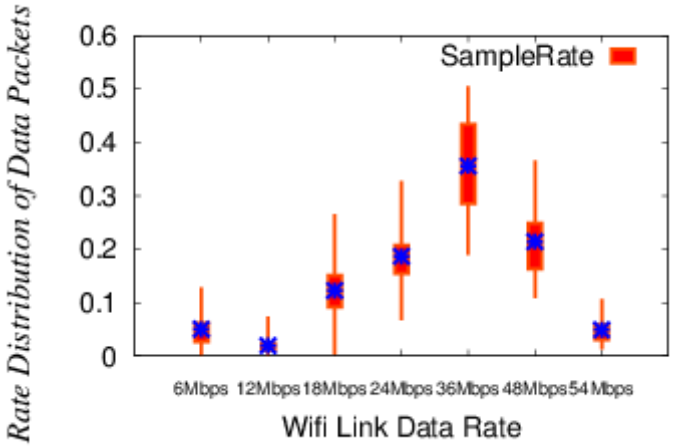


Figure 10.8: Data Rate Distribution generated by the SampleRate autorate algorithm.

We repeat the experiments conducted with WiFi link rates configured to 36Mbps, and activating the Ambient Noise Immunity algorithm, which is a MadWiFi proprietary algorithm. Figure 10.9 shows the achieved goodput enabling the ANI (Ambient Noise Immunity) proprietary algorithm. The ANI algorithm manages the sensitivity of wireless cards to discard potential external noise. The algorithm resides on the Hardware Abstraction Layer (HAL) of the MadWiFi driver. (Recall that the HAL acts like a wrapper with the card hardware registers and is distributed in its binary form.) The goal of the algorithm is to reduce the impact of the interfering sources in a given environment. The algorithm is recommended to be activated in an indoor environment such as ours. The algorithm is based on modifying the receiver sensitivity thresholds of the wireless cards in an adaptive way to deal with the surrounding interference environment. As Figure 10.9 depicts the achieved goodput with the ANI algorithm enabled is equivalent to the one attained for the case of 2, or 4 flows with the ANI algorithm disabled.

However, in the case in which the backhaul has a workload of 6 traffic flows of 1Mbps, ANI achieves a higher goodput than that achieved with the ANI algorithm disabled. Therefore, the algorithm is recommended to be activated in an indoor environment such as ours. The algorithm is based on modifying the receiver sensitivity thresholds of the wireless cards in an adaptive way to deal with the surrounding interference environment. As Figure 10.9 depicts the achieved goodput with the ANI algorithm enabled is equivalent to the one attained for the case of 2, or 4 flows. However, in the case in which the backhaul is loaded with 6 UDP CBR traffic flows, the ANI algorithm achieves a higher goodput than that with the ANI algorithm disabled. Thus, results suggest that ns-3 emulation framework can handle injecting and forwarding, and receiving traffic multiple flows. On the other hand, the limit seems to be defined by the injection of 6Mbps due to the deactivation of MadWiFi proprietary algorithms and probably to the

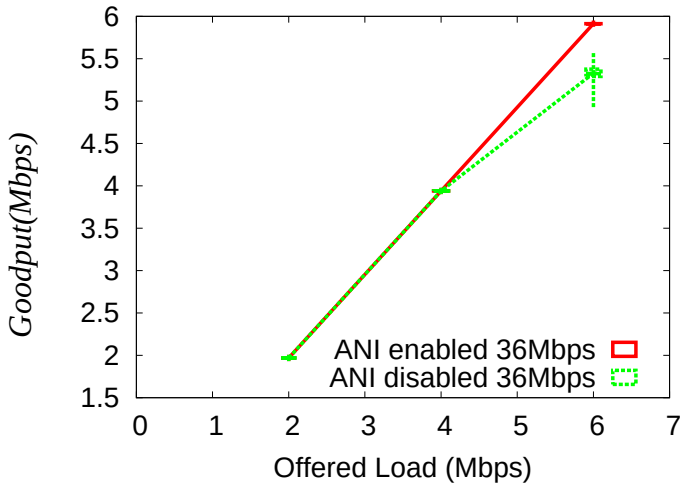
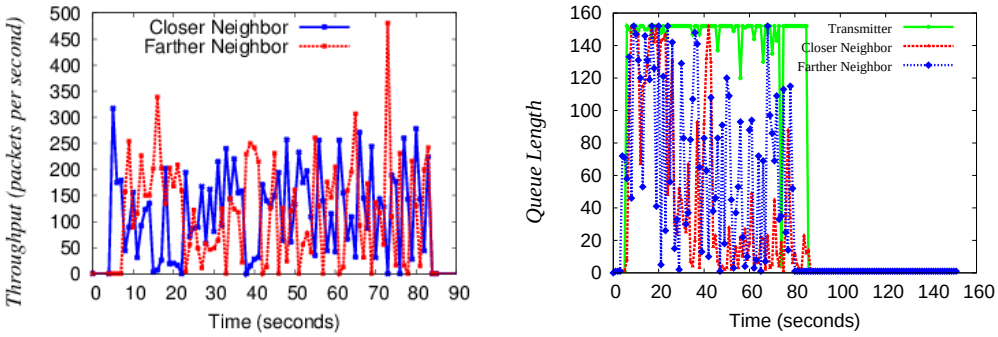


Figure 10.9: Impact of Ambient Noise Immunity (ANI) in Goodput.

contention as maximum transmission power is configured in every WiFi card of the testbed.

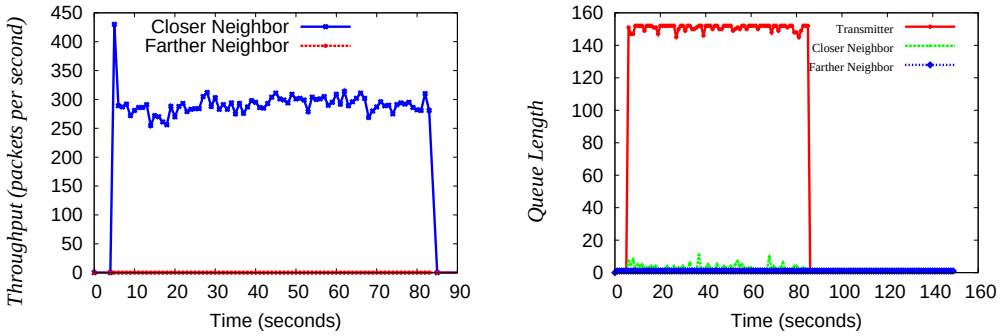


(a) Transmitted Packets

(b) Queue Length

Figure 10.10: Load balancing behavior of the routing protocol for V=0.

Load Balancing Results: Here the goal is to validate the role of the V parameter for enabling load balancing capabilities in the routing protocol. As explained in section 9.2, one of the key aspects characterizing the implementation of the routing protocol is its capacity to do load balancing with the V parameter. In particular, we carry out two experiments in which we evaluate the impact of the V parameter on the next-hop selection algorithm proposed by the routing protocol. Specifically, there is one source node transmitting packets with precisely two neighbors. One neighbor is closer to the destination than the source node, and the second neighbor is farther from the destination than the source node. We fix our attention in the selection of the next hop with the specific choice of the parameter V. The testbed measures the throughput in packets per second received by 1-hop neighbors, and also the queue length evolution of the transmitter and its two 1-hop neighbors.



(a) Transmitted Packets

(b) Queue Length

Figure 10.11: Load balancing behavior of the routing protocol for V=100.

In the first case illustrated in Figure 10.10(a) and Figure 10.10(b)), we evaluated the routing protocol configured with V=0. This is equivalent to a routing policy that strives to minimize the queue backlog differentials between neighboring nodes. As can be shown in Figure 10.10(a), traffic sent by the source node is forwarded through both available neighbors. On the other side, as Figure 10.10(b) we observe high queue backlogs in both the farthest and the closest neighbors from a geographic perspective. This confirms the proper operation of the routing protocol with V=0, because with V=0 the network requires the generation of decreasing queue backlog gradient towards the destination. In this way, the protocol steers packets to the destination since geolocation information is not taken into account with V=0.

In the second case, we repeat the previous experiment but decreasing the degree of load balancing by means of increasing the V parameter to 100. This means that a high weight is assigned to approaching the destination when taking forwarding decisions. In practice, this means that to send data packets to the farthest neighbor, the difference in terms of data queue length between the closest and farthest neighbor neighbor must be greater than 100. In fact, as shown in Figure 10.11(a), all the packets are transmitted to the closest neighbor, and no packets are transmitted to the farthest neighbor. Figure 10.11(b) reveals that, with this degree of load balancing (i.e., V=100) and the injected workload (i.e., 6Mbps of traffic), the node closest to the destination does not experience sufficient queuing so that packets are transmitted directly to the farthest neighbor. In fact, because of this increased steering of packets towards the destination, the network experiences lower data queuing, as the network does not build a decreasing queue backlog gradient to the destination.

10.2.4 Dynamic Wireless Mesh Backhaul Results

In this experiment, a SC in position (6,4,0) in Figure 10.3 injects traffic directed towards the TNL GW, that is, the SC in position (0,4,0) in Figure 10.3. The traffic injected consists of UDP CBR of 1Mbps

with maximum packet size. In particular, this supposes around 85 packets per second in the network. This is an offered load totally feasible for the WiFi mesh backhaul with the use of one single path without causing queuing in the NoS. The traffic volume is light such that there is no necessity to do load balancing over multiple paths, hence facilitating the visibility of the functionalities aimed to be demonstrated in the testbed.

In this case, the source SC has two neighbors available. One neighbor is closer to the destination than the source SC, whereas the other is father from the destination than the source SC. According to Figure 10.3, choosing the closer neighbor supposes reaching the TNL GW in exactly three hops, whereas using the farther neighbor as next-hop can introduce path diversity to reach the TNL GW (i.e., 4, 5, or even 6 hops). The testbed is configured so that one can select the distributed routing policy rolled out in the testbed, either the fixed-V routing policy, based on fixing the V parameter, or the variable-V algorithm, based in the SON algorithm described in chapter 8.

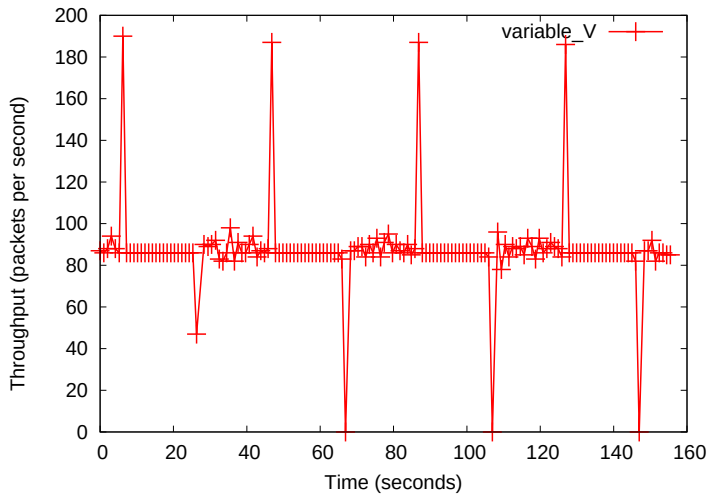


Figure 10.12: Projection over time of the attained Throughput at the TNL GW with distributed back-pressure routing with the variable-V algorithm.

The goal of this experiment is to show how our self-organized backpressure routing protocol with the variable-V algorithm adapts to the dynamic network conditions an all-wireless NoS may pose without zero-configuration by the Mobile Network Operator (MNO). To emulate dynamic network conditions in the wireless mesh backhaul testbed, we periodically switch off and of the real interfaces composing the WiFi backhaul. To have awareness of the real WiFi card status (i.e., up or down) at the ns-3 stack, we extended the implementation of the EmuNetDevice class of the ns-3 emulator, as the current implementation relies on a default up status for real cards underneath. In the experiment, we switch on and off SCs belonging to the shortest path to reach the TNL GW from SC in location (6,4,0) in Figure 10.3.

In particular, the switch off period duration is of 20 seconds, hence disabling the shortest path to reach the intended destination. With such a configuration, our scheme has to quickly react and periodically switch for alternative paths to reach the TNL GW. The duration of the experiment is of 160 seconds. We repeat the experiments for both fixed-V and variable-V routing policies.

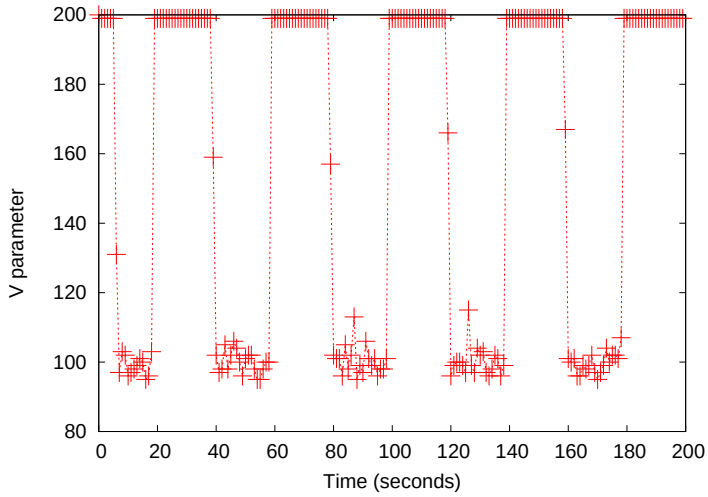


Figure 10.13: Evolution of the V parameter during time with the periodical switch of nodes belonging to the shortest path with the distributed backpressure routing protocol enhanced with the variable-V algorithm.

Figure 10.12 shows the projection over time of the attained throughput at the TNL GW when using the variable-V algorithm. This figure reveals some remarkable aspects to point out:
First, we observe a state in which the served traffic is equivalent to the input rate injected to the WiFi mesh backhaul. In this case, the routing protocol uses either the shortest path or multiple longer paths to serve the workload. When it uses the shortest path, the number of packets received by the destination is constant in time. However, when it uses the multiple longer paths the number of packets received every second experiences a slight variation due to the increase of path utilization diversity.

Second, we observe there are very short instants in which the attained throughput experiences a dramatic degradation. This corresponds to the time required for the source SC to notice that neighbors belonging to the shortest path are switched off. This period is highly related to the HELLO emission interval (i.e., 100ms). To change to another path the source SC requires to auto-configure its V parameter to a lower value (see Figure 10.2.4) so that geographic distance is no more taken into account as a priority in routing decisions. Since the source SC is unaware of finding the next-hop of the packets being routed, it starts accumulating this data packets in its queue. Due to the queuing experienced at the source SC, the V parameter decreases its value, according to the variable-V algorithm described in [6]. Once the V

parameter decreases to an extent in which minimizing the queue backlog differentials in the network is more important than geographic distance to the intended destination, the source SC starts transmitting packets to the farther neighbors. Then, the served throughput at the TNL GW returns to serve the rate injected at the network.

Third, Figure 10.12 periodically shows throughput peaks. These throughput peaks correspond to the instant in which the closer SCs to the TNL GW joins the wireless mesh backhaul upon being switched on. When these SCs are marked as valid, the source SC starts immediately using the closer path to the destination. This happens not only for newly generated traffic but for all the data packets accumulated at the data queue source SC are transmitted, since the capacity of the WiFi link amongst the source SC and the closer SC has enough available capacity. Note that this causes subsequently a throughput peak in the destination, since the WiFi backhaul is able to serve, in addition to the CBR traffic flow of the source SC, the data packets accumulated in the data queue of the source SC. Once the source SC has been dequeued, it returns to serve the 1Mbps CBR traffic flow. As showed by Figure note how the V parameter evolves by increasing its value with the decrease of the queue backlog in source SC.

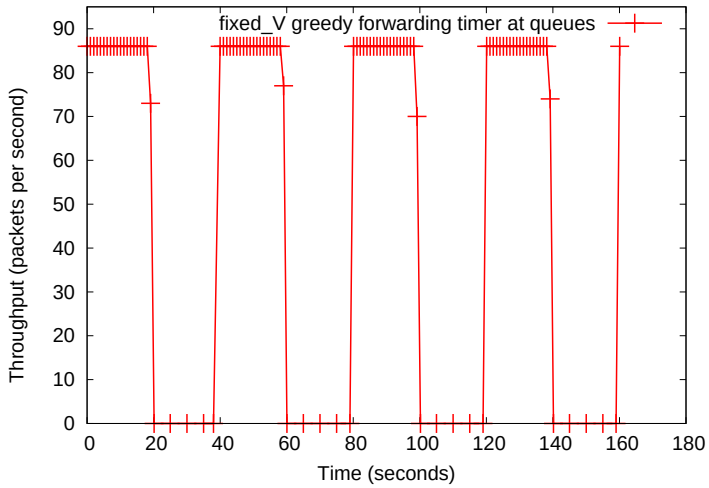


Figure 10.14: Projection over time of the attained TNL GW Throughput with distributed backpressure routing with a fixed-V algorithm based on giving all the importance to the geographic component with a timer implemented at data queues from dropping data packets.

Figure 10.14 shows the projection over time of the attained throughput at the TNL GW with a fixed-V routing policy. In this case note that whenever the SCs belonging to the shortest path are turned down the source SC configured with a fixed-V routing policy is unable to transmit packets towards the intended destination. This is because the V parameter is configured to a fixed value equal to the maximum queue size (i.e., 200 packets) so that merely geographic distance towards the intended destination is taken into

to take forwarding decisions. Therefore, during a time interval of 20 seconds, the GW does not receive data packets. On the other hand, the GW starts receiving packets when SCs closer to the destination are available by the source (around 80 packets per second).

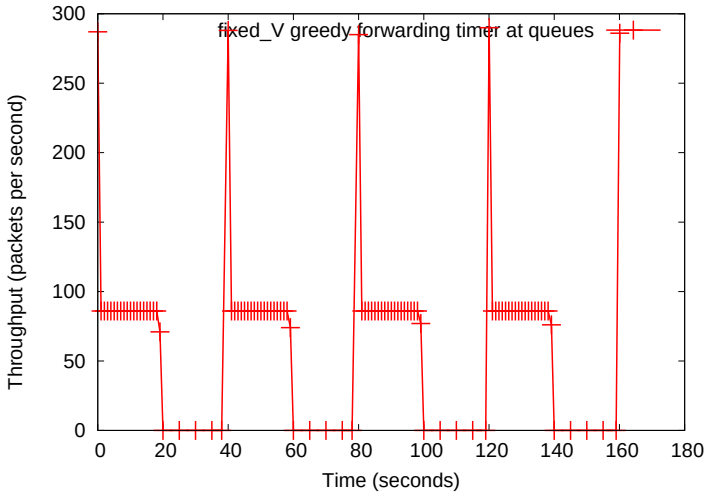


Figure 10.15: Projection over time of the attained TNL GW Throughput with distributed backpressure routing with a fixed-V algorithm based on giving all the importance to the geographic component without a timer implemented in the queue for dropping data packets.

Note that in Figure 10.14 periodic throughput peaks do not appear, as in Figure 10.12 with the variable-V algorithm. This is because the data queue in the source SC drops data packets that experience excessive queuing delays. In these experiments the maximum queuing delay allowed is configured to 5 seconds. Thus, if a packet stays longer than 5 seconds in a queue, the data queue management building blocks considers it can be dropped, and so not transmitted through the backhaul. Since the switch off period of the nodes closer to the destination is of 20 seconds, and the queues implement a FIFO scheduler, when the SCs belonging to the path closer to the destination are switched on, the data packets accumulated at the data queue in the source SC are dropped.

Figure 10.15 shows the same experiment, but with the queuing delay expiration timer disabled. In this case we observe how the throughput at the TNL GW experiences peaks every 20 seconds, since it is transmitting the data packets accumulated at data queues during the switch off period. Since Q_{max} is of 200 packets, and the source node is injecting CBR traffic at a data rate of 85 packets per second, there is enough room to fill in the source node data queue during the time nodes closer to the destination are not available (i.e., 20 seconds). Therefore, as soon as the nodes are switched on, the source node is able to transmit all data packets accumulated at data queues (i.e., 200 packets) jointly with the constant rate offered to the network (i.e., 85 packets per second). This explains the throughput peaks of around 285

packets per second observed in Figure 10.15. Note that again, we observe the degradation of throughput experienced by the TNL GW during the time intervals in which SCs closer to the destination are switched off.

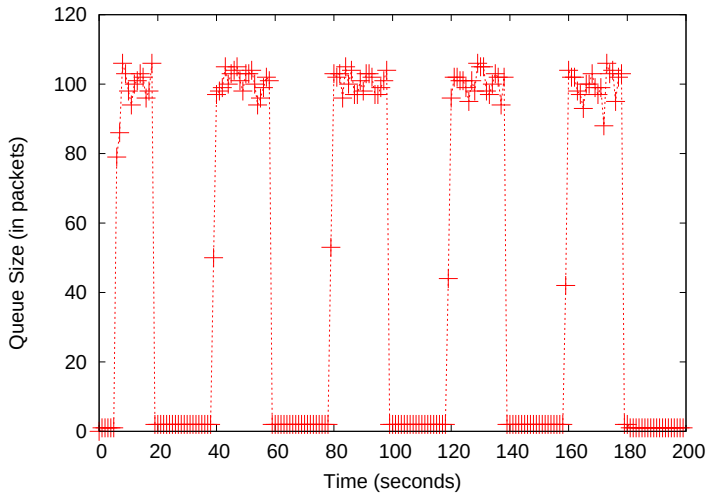


Figure 10.16: Evolution of the Queue Size parameter during time with the periodical switch of nodes belonging to the shortest path with the routing protocol enhanced with the variable-V algorithm.

As Figure 10.16 illustrates, the data queue in the source SC never overflows, since the variable-V algorithm leverages the multiple available paths offered by the wireless mesh backhaul testbed. The source node directly forwards packets to the geographically closer when it is available, whereas it requires to accumulate some packets in its queue to use longer paths. In contrast, using a fixed-V routing policy leads eventually to queue drops in the source SC, as depicted by Figure 10.17. This occurs during the 20 second interval when the closer neighbor is switched off, because the queue size limit of nodes is not large enough to accumulate all the data packets sent during 20 seconds. Thus, even though there are available paths to reach the destination, data packets are dropped by data queues in the source nodes, as showed by Figure 10.18.

Thus, the variable-V algorithm fosters the adaptation of the routing protocol to dynamic wireless backhaul environments. First, it selects the optimal shortest path when it is available, whereas it is able to autonomously select paths different from the shortest ones under unavailability conditions. Figure 10.16 shows that the variable-V algorithm experiences null queue drops during the total duration of the experiment, since queue backlog are below Q_{max}.

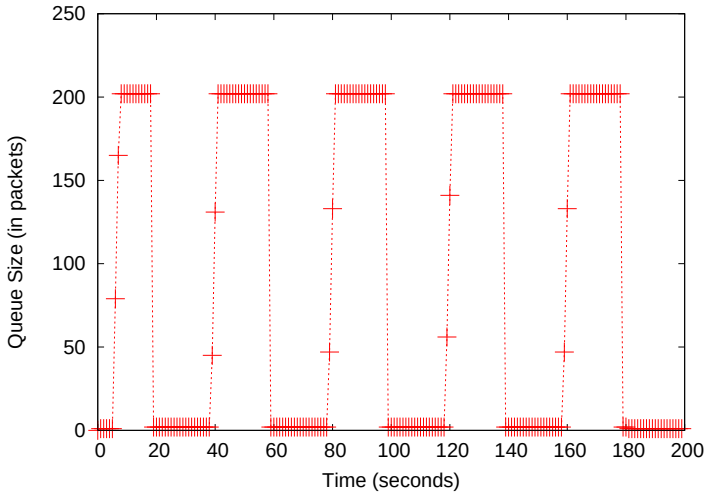


Figure 10.17: Evolution of the Queue Size parameter during time with the periodical switch of nodes belonging to the shortest path with the routing protocol enhanced with the fixed-V algorithm configuring V to the maximum queue size (close to a greedy forwarding routing policy).

10.3 Summary

This chapter experimentally demonstrates that the proposed self-organized backpressure routing scheme satisfies the requirements claimed in our previous work based on simulations. By using the ns-3 emulation framework, we could run our ns-3 implementation practically unmodified in a WiFi-based mesh backhaul testbed of small cells. We evaluated the routing protocol under a wide range of realistic wireless backhaul setups, validating the proper operation of the routing protocol under static and dynamic network conditions. In particular, due to the inherent dynamic path diversity fostered by the resulting algorithm, we can state that the wireless mesh backhaul resources are also fully exploited at the experimental level.

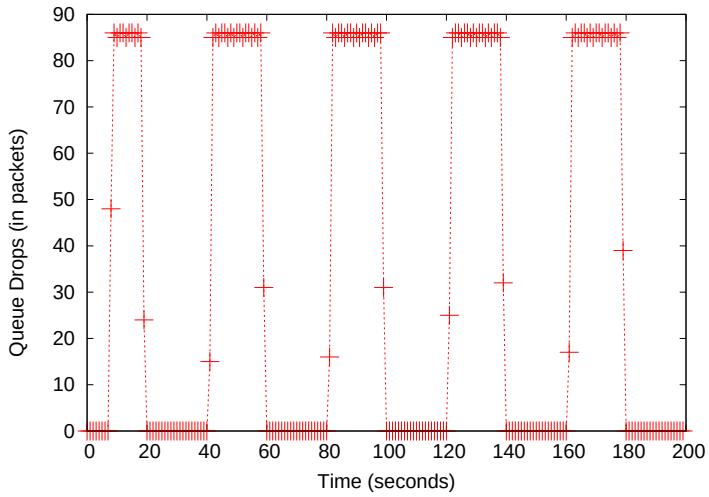


Figure 10.18: Queue Drops over time with the periodical switch of nodes belonging to the shortest path with the routing protocol enhanced with the fixed-V algorithm configuring V to the maximum queue size (equivalent to greedy forwarding). In this case, the routing protocol has the queuing delay timer disabled.

Chapter 11

Conclusions

This chapter covers the main statements of inferences made in this thesis. Section 11.1 presents a set of concise statements ordered from most to least important. Section 11.2 summarizes the contributions of this thesis, which are also ordered from most to least important. Finally, section 11.3 summarizes future research work that remains unsolved.

11.1 Conclusions

1. The main research problem stated in chapter 5 was solved, thus, tackling the requirements posed by wireless mesh backhaul requirements. As showed in chapters 6 to 10, we designed an self-organized algorithm capable of making the most out of the network resources based on the Lyapunov drift-plus-penalty routing approach.

2. Computing routing tables can be counter-productive in a dynamic environment. Rather than following the usual principles of routing protocols designed for wireless mesh backhauls, which are oriented to compute routing tables, our scheme relies on a max-weight algorithm calculated on a per-packet basis.

3. Due to properties such as the no need for statistical knowledge of the underlying network stochastic processes, theoretical algorithms based on the Lyapunov drift are a good candidate to derive dynamic and practical algorithms exhibiting good performance. Chapter 7 provides insights on the theoretical roots in which we based the proposed scheme, formulating the routing problem as a network stochastic optimization problem, and approximated using the Lyapunov drift-plus-penalty method. In particular, we describe the steps taken to convert the routing problem formulation into a practical and self-organized routing scheme for the TNL using the Lyapunov drift-plus-penalty method.

4. The key for attaining scalability and decentralization in a wireless mesh backhaul dealing with any-to-any communication traffic patterns is the inclusion of locally available information such as geolocation information. Chapter 7 provides the main ideas behind the use of locally available information in the computation of weights.

5. A key aspect for the improved network performance metrics and adaptability to wireless mesh backhaul dynamics is the dynamic configuration of the optimization parameter, V, present in the computation of weights to determine the importance of the penalty function, compared to queue backlog differentials. We extended the Lyapunov drift-plus-penalty with a self-organized algorithm presented in chapter 8 in charge of trading the importance between the Lyapunov drift and the penalty function for every packet being routed. The parameter V is calculated independently at every node in the network.

6. The resulting distributed routing algorithm assisted by the dynamic optimization of the configuration parameter V, geolocation, and information carried in the packet is practical. We implemented and evaluated the resulting distributed backpressure routing policy in a testbed addressing the constraints of wireless mesh backhauls. A description of the implementation and problems faced as well as the experimental evaluation of the solution can be found in chapters 9 and 10, respectively.

7. As shown in chapter 8, the resulting distributed backpressure routing solution outperforms SoA routing approaches for the TNL in throughput and delay for a wide variety of wireless mesh backhaul deployments. Indeed, the solution adapts to both regular and non-regular SC deployments, a variable number of gateways, and heterogeneous wireless link rates.

8. The 3GPP architectural issues faced by a wireless mesh backhaul formed amongst SCs should not suppose a limit for dense small cell deployments. In particular, chapter 6 defines the concept of Network of SC (NoS) proposing a solution at the architectural level to confine control and data plane traffic to the local environment.

9. Chapters 8 and 10 confirm that the proposed self-organized TNL routing scheme satisfies the aimed requirements, yet being simple to implement. Further, it merely requires low-complexity operations to take routing decisions on a per-packet basis. Actually, its implementation does not incur into additional expenses than potentially a GPS per SC.

11.2 Contributions

The list of contributions of this thesis are as follows:

1. We brought the Lyapunov drift-plus-penalty theoretical method into practice for wireless mesh backhauls. Moeller et. al. [76] implemented the drift-plus-penalty model to the context of many-to-one WSN light traffic patterns. We extended this work by bringing this model in the context of unicast any-to-any traffic patterns to handle the ever-increasing traffic demands experienced by Mobile Network Operators (MNO). The resulting self-organized backpressure routing protocol satisfies the requirements defined in chapter 5, unlike SoA TNL approaches partly surveyed in [3].

2. We showed in [2] that the inclusion of geolocation information in the penalty function alleviates several theory-to-practice barriers to implement a self-organized backpressure routing protocol for wireless mesh backhauls. Practical limitations such as scalability, finite queue sizes, the necessity of a central entity with a global view of the network are addressed with the inclusion of geolocation information in the penalty function, present in the computation of weights. In addition to this, aiming for a low-overhead approach, instead of computing and maintaining routing tables, we leverage geolocation information to allow any-to-any traffic pattern communications, and to avoid the management of a data queue per flow being routed in the network. The state kept at every node in the network grows with the number of 1-hop neighbors, which makes it also scalable with the network size.

3. The drift-plus-penalty method uses a tunable optimization parameter. We find evidence that, in fact, this optimization parameter impacts the overall network performance under traffic dynamics. We addressed the potential limitations introduced by the drift-plus-penalty method in [5], which uses a fixed optimization parameter. We designed a self-organized algorithm that calculates the best trade-off between the Lyapunov drift and the penalty function based on geolocation information for every packet being routed. The self-organized backpressure routing algorithm, which is calculated using locally available information and information carried in the packet header, attains significant latency and throughput gains with respect to routing policies with a fixed con-

figuration parameter. Selecting dynamically the proper importance between queue awareness and geographic important results in a policy that uses short paths as long as there is no congestion in the network. With the rise of congestion and so queuing latencies, the routing policy uses higher non congested paths without limiting the set of nodes that a packet can traverse.

4. As showed in [2], and [5], the resulting protocol makes the most out of the available wireless backhaul resources, hence distributing the workload among available network resources. This brings benefits regarding the performance of network metrics compared with other SoA routing approaches. Instead of computing routing tables and using a fixed number of equal cost paths, the proposed routing policy computes a weight on a per-packet basis, which depends on a combination of factors extracted from the locally observed network state. As a result, data packets can be dynamically sprayed across a large number of non-equal cost paths (e.g., different path lengths), when the traffic load requires the use of more resources to avoid congested paths.

5. We implemented [13] and evaluated the self-organized backpressure routing algorithm in a 12-node proof-of-concept testbed [14, 15], located in the first floor of the CTTC building. This prototype allowed the validation with real measurements of the schemes discussed in this thesis for wireless mesh backhauls. The use of the ns-3 emulation framework eased the development process, avoiding the development of the protocol from scratch. As a matter of fact, we used most of the developed ns-3 code in a real testbed by means of the ns-3 emulation mode. Evaluation results demonstrate the proper operation of our scheme, confirming findings observed with ns-3 simulation results.

6. We demonstrated that the algorithm can scale with the number of TNL gateways deployed in the wireless backhaul. Moreover, the resulting scheme reduces complexity and costs, since most of the gateways require null planning costs and merely entail TNL functionalities. Our scheme maximizes the exploitation of such gateways in terms of throughput and latency by up to 40% and 99%, respectively, without awareness of the precise location in the network of most gateways. These results were published in [6].

7. We showed in [7] that with our solution the effort devoted to SC location planning will decrease. This is because the protocol circumvent dead-ends present in sparse wireless mesh backhaul topologies while maintaining low overhead and low state properties, unlike geographic-only routing approaches. Another interesting contribution here is that wireless backhaul topology dynamics do not cause an impact in the main goal of the protocol, which is making the most out of the network resources. Thus, per-packet routing decisions taken by our backpressure routing scheme

appear well suited for unreliable and dynamic wireless mesh backhauls showing a good compromise between throughput and latency. In particular, our scheme outperforms SoA approaches in terms of throughput and latency by up to 85% and 70%, respectively.

8. In [11], [9], and [12], we addressed the architectural challenges posed by an all-wireless mesh backhaul formed by SCs, defining an architecture that can overcome the initial 3GPP constraints. The resulting concept, coined as Network of Small Cells (NoS), proposes an architecture capable of supporting a wireless mesh backhaul amongst SCs to carry traffic from/to the core network to/from the SCs. The resulting architecture allows packets to be routed over the NoS in a completely transparent way to existing 3GPP control- and user-plane procedures.

9. For comparison against other SoA TNL approaches and scalability validation purposes, we conducted simulations. In particular, we used an accurate network simulator, the ns-3 [102] network simulator, whose accuracy was demonstrated in [4].

11.3 Future Work

Our investigation has sparked many ideas that can form the basis for future research, which we will now discuss in some detail.

1. Wireless Backhaul Sharing. A scenario not tackled in this dissertation is that of RAN-sharing [122], in which several tenants can share the networks resources deployed by a Mobile Network Operator (MNO). It is important to note that sharing backhaul resources can be a cost effective solution against over-provisioning. In this way, Mobile Network Operators (MNO) would reduce CAPEX and OPEX by reusing both the access and backhaul mobile wireless infrastructure.

   Currently, the 3GPP RAN Sharing Enhancements (RSE) study item is defining new scenarios of multiple mobile carrier operators sharing radio network resources. In such a scenario, a key goal is to make the most out of the resources deployed, which coincides with the goal of this thesis. A challenge to do this is clearly researching solutions for implementing Wireless Backhaul sharing that can achieve load distribution of virtual network functions within the shared mobile infrastructure. Attaining load distribution on a virtual environment is a topic subject of further work. We believe that Software Defined Networks [123] and Network Function Virtualization (NFV) [124] jointly with the techniques studied in this thesis can be key players to make the most out of the network resources in such an environment.

2. Scheduling. Our work has been hitherto focused on backpressure routing assuming a distributed CSMA/CA link layer scheduling. In the theoretical work, Backpressure assumes a globally synchronized time-slotted MAC protocol as well as a central controller that computes and disseminates a schedule (i.e., a set of links allowed to transmit) for each time slot. In our case, we relaxed such assumptions by implementing the backpressure routing algorithm on top of the 802.11 MAC protocol. Although we showed high performance gains compared to SoA routing solutions, the addition of a TDMA access layer to carry joint backpressure routing/scheduling decisions would obtain significant performance gains. This is demonstrated in [125], and some practical work has already been carried out in this direction in [78].

3. Study of multi-interface scenarios. Throughout this work, we assumed that SCs are only endowed with a single wireless radio and potentially a high-capacity link to the TNL GW (see chapter 8). However, to increase capacity per SC the addition of multiple wireless backhaul interfaces can be promising to fulfill increasing traffic demands. The evaluation of the routing protocol in small cell backhaul endowed with multiple wireless backhaul interfaces is subject of further study.

4. Study on Packet Delay Variation. Throughout this dissertation, we mainly studied the performance of throughput and latency. However, packet delay variation, though apparently reduced as can be noted during the evaluations conducted in this dissertation, is a metric that requires deeper attention.

5. Traffic Classes. One important aspect not tackled in this dissertation is the allocation of resources for different traffic classes. Despite of providing improvements at the aggregated network performance metric level, the resulting routing protocol only deals with a single traffic class. The challenge here is to provide different routing treatment per each traffic class, whilst maintaining the simplicity in terms of the implementation of our TNL scheme.

Bibliography

[1] "AVIAT networks," `http://www.aviatnetworks.com`.

[2] José Núñez Martínez, Josep Mangues-Bafalluy, and Marc. Portoles-Comeras, "Studying practical any-to-any backpressure routing in wi-fi mesh networks from a lyapunov optimization perspective," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, 2011.

[3] José Núñez-Martínez and Josep Mangues-Bafalluy, "A survey on routing protocols that really exploit wireless mesh network features," *Journal of Communications*, vol. 5, no. 3, pp. 211–231, 2010.

[4] Nicola Baldo, Manuel Requena-Esteso, José Núñez Martínez, Marc Portolès-Comeras, Jaume Nin-Guerrero, Paolo Dini, and Josep Mangues-Bafalluy, "Validation of the ieee 802.11 mac model in the ns3 simulator using the extreme testbed," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010, SIMUTools '10.

[5] José Núñez Martínez and Josep Mangues-Bafalluy, "Distributed lyapunov drift-plus-penalty routing for wifi mesh networks with adaptive penalty weight," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, 2012, pp. 1–6.

[6] José Núñez Martínez, Josep Mangues-Bafalluy, and Jorge Baranda, "Anycast backpressure routing: Scalable mobile backhaul for dense small cell deployments," *Communications Letters, IEEE*, December 2013.

[7] Jose Núñez-Martínez, Jorge Baranda, and Josep Mangues-Bafalluy, "Backpressure routing for the backhaul in sparse small cell deployments," in *International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2013.

[8] José Núñez-Martínez, Jorge Baranda, and Josep Mangues-Bafalluy, "A self-organized backpressure routing scheme for dynamic small cell deployments," *Submitted to Elsevier Adhoc Networks*, 2014.

[9] Nicola Baldo, Paolo Dini, Josep Mangues-Bafalluy, Marco Miozzo, and José Núñez Martínez, "Small cells, wireless backhaul and renewable energy: A solution for disaster aftermath communications," in *Proceedings of the 4th International Conference on Cognitive Radio and Advanced Spectrum Management*, New York, NY, USA, 2011, CogART '11, pp. 52:1–52:7, ACM.

[10] Marco Miozzo, José Núñez-Martínez, Nicola Baldo, Josep Mangues-Bafalluy, and Paolo Dini, "Zero grid energy small cells with wireless backhaul:a rapidly deployable network for disaster relief communications," *Submitted to IEEE Communication Magazine*, 2014.

[11] José Núñez Martínez, Jaime Ferragut, and Josep Mangues-Bafalluy, "On stateless routing for an all-wireless network of femtocells: Implications in the 3gpp architecture," in *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, 2011, pp. 1–5.

[12] Jaime Ferragut, Josep Mangues-Bafalluy, José Núñez Martínez, and Frank Zdarsky, "Traffic and mobility management in networks of femtocells," *Mob. Netw. Appl.*, 2012.

[13] José Núñez-Martínez and Josep Mangues-Bafalluy, "Design, implementation, and tracing of dynamic backpressure routing for ns-3," in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 462–469.

[14] Manuel Requena-Esteso, José Núñez Martínez, and Josep Mangues-Bafalluy, "Wireless mesh networking framework for fast development and testing of network-level protocols," in *Proceedings of the ICT Mobile Summit*, 2009.

[15] José Núñez-Martínez and Josep Mangues-Bafalluy, "A case for evaluating backpressure routing using ns-3 emulation in a wifi mesh testbed," in *Proceedings of the seventh ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*. ACM, 2012, pp. 27–34.

[16] José Núñez-Martínez and Josep Mangues-Bafalluy, "Experimental evaluation of self-organized backpressure routing in a wireless mesh backhaul of small cells," *Submitted to Elsevier Adhoc Networks: Special Issue on performance evaluation for wireless networks*, 2014.

[17] Next Generation Mobile Networks Alliance, "LTE backhauling deployment scenarios," *NGMN White paper*, 2011.

[18] Next Generation Mobile Networks Alliance, "Small Cell Backhaul Requirements," *NGMN White paper*, 2012.

[19] S. Chia, M. Gasparroni, and P. Brick, "The next challenge for cellular networks: backhaul," *Microwave Magazine, IEEE*, vol. 10, no. 5, pp. 54–66, 2009.

[20] M. Bocci, "A framework for mpls in transport networks," *Internet RFC 5921*, 2010.

[21] "ITU-T rec. g.8032/y.1344, Ethernet Ring Protection Switching," 2008.

[22] "IEEE standard for local and metropolitan area networks virtual bridged local area networks," *IEEE Std 802.1Q-2005*, 2006.

[23] D. Allan, P. Ashwood-Smith, N. Bragg, J. Farkas, D. Fedyk, M. Ouellete, M. Seaman, and P. Unbehagen, "Shortest path bridging: Efficient control of larger ethernet networks," *Communications Magazine, IEEE*, vol. 48, no. 10, pp. 128–135, 2010.

[24] D. Allan, J. Farkas, and S. Mansfield, "Intelligent load balancing for shortest path bridging," *Communications Magazine, IEEE*, vol. 50, no. 7, pp. 163–167, 2012.

[25] R. Perlman, D. Eastlake 3rd, D. Dutt, S. Gai, and A. Ghanwani, "Routing bridges (RBridges): Base protocol specification," 2011.

[26] D. Thaler and C. Hopps, "Multipath issues in unicast and multicast next-hop selection," 2000.

[27] J. Moy, "RFC 2328: OSPF version 2," 1998.

[28] D. Oran, "OSI IS-IS intra-domain routing protocol," 1990.

[29] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, Nov. 1992.

[30] Richard Draves, Jitendra Padhye, and Brian Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*. ACM, 2004, pp. 114–128.

[31] Anand Prabhu Subramanian, Milind M Buddhikot, and Scott Miller, "Interference aware routing in multi-radio wireless mesh networks," in *Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on*. IEEE, 2006, pp. 55–63.

[32] Yaling Yang, Jun Wang, and Robin Kravets, "Designing routing metrics for mesh networks," in *IEEE Workshop on Wireless Mesh Networks (WiMesh)*, 2005.

[33] Ashish Raniwala and Tzi-cker Chiueh, "Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. IEEE, 2005, vol. 3, pp. 2223–2234.

[34] Krishna Ramachandran, Irfan Sheriff, Elizabeth M Belding, and Kevin C Almeroth, "A multi-radio 802.11 mesh network architecture," *Mobile Networks and Applications*, vol. 13, no. 1-2, pp. 132–146, 2008.

[35] Aditya Dhananjay, Hui Zhang, Jinyang Li, and Lakshminarayanan Subramanian, "Practical, distributed channel assignment and routing in dual-radio mesh networks," in *ACM SIGCOMM Computer Communication Review*. ACM, 2009, vol. 39, pp. 99–110.

[36] Can Emre Koksal and Hari Balakrishnan, "Quality-aware routing metrics for time-varying wireless mesh networks," 2006, vol. 24, pp. 1984–1994, IEEE.

[37] Saumitra M Das, Himabindu Pucha, Dimitrios Koutsonikolas, Y Charlie Hu, and Dimitrios Peroulis, "DMesh: incorporating practical directional antennas in multichannel wireless mesh networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 11, pp. 2028–2039, 2006.

[38] Sivaram Cheekiralla and Daniel W Engels, "Routing in heterogeneous wireless ad hoc networks," in *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*. IEEE, 2007, pp. 951–956.

[39] Douglas SJ De Couto, Daniel Aguayo, John Bicket, and Robert Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[40] Sanjit Biswas and Robert Morris, "ExOR: opportunistic multi-hop routing for wireless networks," in *ACM SIGCOMM Computer Communication Review*, 2005, vol. 35, pp. 133–144.

[41] Eric Rozner, Jayesh Seshadri, Yogita Mehta, and Lili Qiu, "SOAR: Simple opportunistic adaptive routing protocol for wireless mesh networks," *Mobile Computing, IEEE Transactions on*, vol. 8, no. 12, pp. 1622–1635, 2009.

[42] Yuan Yuan, Hao Yang, Starsky HY Wong, Songwu Lu, and William Arbaugh, "ROMER: resilient opportunistic mesh routing for wireless mesh networks," in *IEEE workshop on wireless mesh networks (WiMesh)*, 2005, vol. 12.

[43] Kai Zeng, Wenjing Lou, and Hongqiang Zhai, "Capacity of opportunistic routing in multi-rate and multi-hop wireless networks," 2008, vol. 7, pp. 5118–5128, IEEE.

[44] Rafael Laufer, Henri Dubois-Ferriere, and Kleinrock Leonard, "Multirate anypath routing in wireless mesh networks," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 37–45.

[45] Rafael Laufer, Henri Dubois-Ferrière, and Leonard Kleinrock, "Polynomial-time algorithms for multirate anypath routing in wireless multihop networks," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 3, pp. 742–755, 2012.

[46] Wei Hu, Jin Xie, and Zhenghao Zhang, "Practical opportunistic routing in high-speed multi-rate wireless mesh networks," in *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2013, pp. 127–136.

[47] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi, "Trading structure for randomness in wireless opportunistic routing," in *ACM SIGCOMM Computer Communication Review*. 2007, vol. 37, ACM.

[48] Christos Gkantsidis, Wenjun Hu, Peter Key, Bozidar Radunovic, Pablo Rodriguez, and Steluta Gheorghiu, "Multipath code casting for wireless mesh networks," in *Proceedings of the 2007 ACM CoNEXT conference*. ACM, 2007, p. 10.

[49] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron, "Virtual ring routing: network routing inspired by DHTs," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2006, SIGCOMM '06, pp. 351–362, ACM.

[50] Jilin Le, John CS Lui, and Dah-Ming Chiu, "DCAR: Distributed coding-aware routing in wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 4, pp. 596–608, 2010.

[51] Kyu-Han Kim and Kang G Shin, "On accurate measurement of link quality in multi-hop wireless mesh networks," in *Proceedings of the 12th annual international conference on Mobile computing and networking*. ACM, 2006, pp. 38–49.

[52] Vivek Mhatre, Henrik Lundgren, Francois Baccelli, and Christophe Diot, "Joint mac-aware routing and load balancing in mesh networks," p. 19, 2007.

[53] Aggelos Vlavianos, Lap Kong Law, Ioannis Broustis, Srikanth V Krishnamurthy, and Michalis Faloutsos, "Assessing link quality in ieee 802.11 wireless networks: Which is the right metric?," in *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*. IEEE, 2008, pp. 1–6.

[54] Sudarshan Vasudevan, Jim Kurose, and Don Towsley, "On neighbor discovery in wireless networks with directional antennas," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*. IEEE, 2005, vol. 4, pp. 2502–2512.

[55] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen, "Fisheye state routing: A routing scheme for ad hoc wireless networks," in *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*. IEEE, 2000, vol. 1, pp. 70–74.

[56] Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Junling Wang, Zifei Zhong, Guor-Huar Lu, and Zhi-Li Zhang, "Blacklist-aided forwarding in static multihop wireless networks.," in *SECON*, 2005, pp. 252–262.

[57] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot, "Optimized link state routing protocol (OLSR)," 2003.

[58] Bow-Nan Cheng, Murat Yuksel, and Shivkumar Kalyanaraman, "Orthogonal rendezvous routing protocol for wireless mesh networks," 2009, vol. 17, pp. 542–555, IEEE Press.

[59] Kirill Levchenko, Geoffrey M Voelker, Ramamohan Paturi, and Stefan Savage, "XL: An efficient network routing algorithm," in *ACM SIGCOMM Computer Communication Review*. ACM, 2008, vol. 38, pp. 15–26.

[60] Taek Jin Kwon and Mario Gerla, "Efficient flooding with passive clustering (PC) in ad hoc networks," 2002, vol. 32, pp. 44–56, ACM.

[61] Zygmunt J Haas, Joseph Y Halpern, and Li Li, "Gossip-based ad hoc routing," 2006, vol. 14, pp. 479–491, IEEE Press.

[62] Kai Zeng, Wenjing Lou, and Yanchao Zhang, "Multi-rate geographic opportunistic routing in wireless ad hoc networks," in *Military Communications Conference, 2007. MILCOM 2007. IEEE*. IEEE, 2007, pp. 1–7.

[63] Henri Dubois-Ferriere, Matthias Grossglauser, and Martin Vetterli, "Least-cost opportunistic routing," in *Proceedings of 2007 Allerton Conference on Communication, Control, and Computing*, 2007, vol. 11, pp. 13–39.

[64] Zifei Zhong and Srihari Nelakuditi, "On the efficacy of opportunistic routing," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON'07. 4th Annual IEEE Communications Society Conference on*. IEEE, 2007, pp. 441–450.

[65] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Médard, and Jon Crowcroft, "XORs in the air: practical wireless network coding," in *ACM SIGCOMM Computer Communication Review*. ACM, 2006, vol. 36, pp. 243–254.

[66] F. Cadger, K. Curran, J. Santos, and S. Moffett, "A survey of geographical routing in wireless ad-hoc networks," *Communications Surveys Tutorials, IEEE*, 2013.

[67] Dazhi Chen and P.K. Varshney, "A survey of void handling techniques for geographic routing in wireless networks," *Communications Surveys Tutorials, IEEE*, vol. 9, no. 1, pp. 50–67, 2007.

[68] Brad Karp and Hsiang-Tsung Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000, pp. 243–254.

[69] Anindya Basu, Alvin Lin, and Sharad Ramanathan, "Routing using potentials: a dynamic traffic-aware routing algorithm," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, SIGCOMM '03.

[70] R. Baumann, S. Heimlicher, Vincent Lenders, and M. May, "HEAT: Scalable routing in wireless mesh networks using temperature fields," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, 2007, pp. 1–9.

[71] Sangsu Jung, M. Kserawi, Dujeong Lee, and J.-K.K. Rhee, "Distributed potential field based routing and autonomous load balancing for wireless mesh networks," *Communications Letters, IEEE*, 2009.

[72] Sangsu Jung, Jihoon Sung, Yonghwan Bang, M. Kserawi, Hyeji Kim, and J.-K.K. Rhee, "Greedy local routing strategy for autonomous global load balancing based on three-dimensional potential field," *Communications Letters, IEEE*, 2010.

[73] Michael J Neely, Eytan Modiano, and Charles E Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 1, pp. 89–103, 2005.

[74] Michael J Neely, "Energy optimal control for time-varying wireless networks," *Information Theory, IEEE Transactions on*, vol. 52, no. 7, pp. 2915–2934, 2006.

[75] Michael J Neely, Eytan Modiano, and Chih-Ping Li, "Fairness and optimal stochastic control for heterogeneous networks," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 2, pp. 396–409, 2008.

[76] Scott Moeller, Avinash Sridharan, Bhaskar Krishnamachari, and Omprakash Gnawali, "Routing without routes: the backpressure collection protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, New York, NY, USA, 2010, IPSN '10, pp. 279–290, ACM.

[77] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queuing Systems*, Synthesis Lectures on Communication Networks, in Morgan & Claypool Publishers, 2010.

[78] Rafael Laufer, Theodoros Salonidis, Henrik Lundgren, and Pascal Le Guyadec, "XPRESS: a cross-layer backpressure architecture for wireless multi-hop networks," in *Proceedings of the 17th annual international conference on Mobile computing and networking*. ACM, 2011, pp. 49–60.

[79] L.X. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1597–1609, 2011.

[80] Lei Ying, S. Shakkottai, A. Reddy, and Shihuan Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *Networking, IEEE/ACM Transactions on*, 2011.

[81] A. Warrier, S. Janakiraman, Sangtae Ha, and I. Rhee, "Diffq: Practical differential backlog congestion control for wireless networks," in *INFOCOM 2009, IEEE*, april 2009, pp. 262 –270.

[82] Bozidar Radunovic, Christos Gkantsidis, Dinan Gunawardena, and Peter Key, "Horizon: balancing tcp over multiple paths in wireless mesh network," New York, NY, USA, 2008, MobiCom, ACM.

[83] B. Radunovic, C. Gkantsidis, P. Key, and P. Rodriguez, "Toward practical opportunistic routing with intra-session network coding for mesh networks," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 2, pp. 420 –433, april 2010.

[84] Adel Aziz, David Starobinski, Patrick Thiran, and Alaeddine El Fawal, "EZ-Flow: removing turbulence in ieee 802.11 wireless mesh networks without message passing," 2009, CoNEXT '09, pp. 73–84, ACM.

[85] "The mobile broadband standard," available at: `http://www.3gpp.org`.

[86] Next Generation Mobile Networks, ," `Online:http://www.ngmn.org`.

[87] "Evolved universal terrestrial radio access network (E-UTRAN); self-configuring and self-optimizing network (son) use cases and solutions," *3GPP TS 36.902*, 2011.

[88] 3GPP Release 12, ," `http://www.3gpp.org/Release-12`.

[89] Utpal Paul, Anand Prabhu Subramanian, Milind M Buddhikot, and Samir R Das, "Understanding traffic dynamics in cellular data networks," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 882–890.

[90] "The LUNAR project," `Online:http://cn.cs.unibas.ch/projects/lunar`.

[91] "Standard for local and metropolitan area networks: Virtual bridged local area networks - amendment 8: Shortest path bridging," 2012.

[92] J. Gettys, "Bufferbloat: Dark buffers in the internet," *Internet Computing, IEEE*, vol. 15, no. 3, 2011.

[93] Yi. Y. and Chiang M., "A tribute to kelly's paper published in this journal a decade ago," *European Transactions on Telecommunications*, vol. 19, no. 4, 2008.

[94] "3rd generation partnership project; technical specification group radio access network; evolved universal terrestrial radio access network (e-utran); x2 application protocol (x2ap) (release 11)," *3GPP TS 36.423*, 2013.

[95] "The BeFEMTO project," `http://www.ict-befemto.eu`.

[96] Broadband evolved FEMTO networks (BeFEMTO) project, "The BeFEMTO system architecture, deliverable d2.2. december 2011.," .

[97] ETSI ES 282 001, "Telecommunications and internet converged services and protocols for advanced networking (tispan); ngn functional architecture v3.3.0, 2009.," .

[98] "General packet radio service (gprs) enhancements for evolved universal terrestrial radio access network (e-utran) access, v.8.14.0 (release 8)," *3GPP TS 23.401*, 2011.

[99] Broadband evolved FEMTO networks (BeFEMTO) project, "Femtocell access control, networking, mobility, and management concepts, deliverable d5.1. december 2010.," .

[100] "Evolved universal terrestrial radio access (E-UTRAN);overall description, (release 12)," *3GPP TS 36.300*, 2014.

[101] Jaime Ferragut, "Traffic and mobility management in large-scale networks of small cells," *Phd Thesis to be published*, 2014.

[102] "The ns-3 network simulator," available at: `http://www.nsam.org`.

[103] M. J. Neely L. Georgiadis and L. Tassiulas, *Resource Allocation and Cross-Layer Control in Wireless Networks*, Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.

[104] R. G. Gallager D. P. Bertsekas, *Data Networks*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1987; Edition 2, Dec., 1991.

[105] "The LTE generation: Smaller and cloudier," Mobile Europe: Insight Report.

[106] Krishna Ramachandran, M Buddhikot, Girish Chandranmenon, Scott Miller, Elizabeth Belding-Royer, and K Almeroth, "On the design and implementation of infrastructure mesh networks," in *IEEE workshop on wireless mesh networks (WiMesh)*, 2005.

[107] Yi Li, Lili Qiu, Yin Zhang, Ratul Mahajan, Zifei Zhong, Gaurav Deshpande, and Eric Rozner, "Effects of interference on wireless mesh networks: Pathologies and a preliminary solution," 2007, Citeseer.

[108] "Madwifi: Multiband atheros driver for wifi," `http://www.madwifi-project.org`.

[109] R. Laufer, P.B. Velloso, L.F.M. Vieira, and L. Kleinrock, "PLASMA: A new routing paradigm for wireless multihop networks," in *Proc of INFOCOM*. 2012, IEEE.

[110] "802.11s-2011: Wireless lan medium access control (MAC) and physical layer (phy) specifications amendment 10: Mesh networking," 2011.

[111] Antonio Fonseca, Andre Camoes, and Teresa Vazao, "Geographical routing implementation in ns3," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*. ACM, 2012, pp. 353–358.

[112] Junseok Kim, "Shortest Path Routing Protocol for ns3," 2011, [Online; accessed 28-November-2013].

[113] B. Aiken et al., "Report of NSF workshop on network research testbeds," 2002.

[114] "EXTREME Testbed ®," a generic description of the testbed is available in `http://networks.cttc.es/mobile-networks/extreme_testbed/`.

[115] K. Fall, "A network emulation in the vint/NS simulator," in *Proceedings of the IEEE Symposium on Computers and Communications*. 1999, IEEE.

[116] Manuel Ricardo Gustavo Carneiro, Helder Fontes, "Fast prototyping of network protocols through ns-3 simulation model reuse, simulation modeling practice and theory," 2011, Elsevier.

[117] Sam Jansen and Anthony McGregor, "Simulation with real world network stacks," 2005, Winter Simulation Conference.

[118] Hajime Tazaki, Frédéric Uarbani, Emilio Mancini, Mathieu Lacage, Daniel Camara, Thierry Turletti, and Walid Dabbous, "Direct code execution: Revisiting library os architecture for reproducible network experiments," in *Proceedings of the Ninth ACM CoNEXT*, 2013.

[119] Pablo Serrano, Carlos J Bernardos, Antonio De La Oliva, Albert Banchs, Ignacio Soto, and Michael Zink, "Floornet: Deployment and evaluation of a multihop wireless 802.11 testbed," *EURASIP Journal on Wireless Communications and Networking*, 2010.

[120] Konstantina Papagiannaki, Mark D Yarvis, and W Steven Conner, "Experimental characterization of home wireless networks and design implications.," in *INFOCOM*, 2006.

[121] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris, "Architecture and evaluation of an unplanned 802.11 b mesh network," in *Proceedings of the 11th annual international conference on Mobile computing and networking*. ACM, 2005, pp. 31–42.

[122] Xavier Costa-Perez, Joerg Swetina, Tao Guo, Rajesh Mahindra, and Sampath Rangarajan, "Radio access network virtualization for future mobile carrier networks," *Communications Magazine, IEEE*, vol. 51, no. 7, 2013.

[123] J. Rexford N. Feamster and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *Queue ACM*, 2013.

[124] "ETSI Network Function Virtualization Industry Standardization Group," available at: `http://www.etsi.org/technologies-clusters/technologies/nfv`.

[125] Jae-Yong Yoo, Cigdem Sengul, Ruben Merz, and JongWon Kim, "Backpressure scheduling in IEEE 802.11 wireless mesh networks: Gap between theory and practice," 2012, vol. 56, pp. 2934–2948, Elsevier.