UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Cooperation in Open, Decentralized, and Heterogeneous Computer Networks

Axel Neumann

PhD Thesis by the
Universitat Politècnica de Catalunya

Advisors:
Leandro Navarro
Llorenç Cerdà-Alabern

Barcelona, Spain 2017

Cooperation in Open, Decentralized, and Heterogeneous Computer Networks.
September 2017.

Axel Neumann
axel@ac.upc.edu

Computer Networks and Distributed Systems Group
Universitat Politècnica de Catalunya
Jordi Girona 1-3 C6, D6
08034 - Barcelona, Spain

Acknowledgments

I am so much grateful to my advisors, Leandro Navarro and Llorenç Cerdà-Alabern. Without their endless patience, guidance and support, this work would simply have been impossible.

I would like to thank all the people and communities that stimulated and inspired this work Roger Baig (Guifi.net), Pau Escrich (Guifi.net, Libremesh.org), Roger P. Centelles (Guifi.net), Agusti Moll (Guifi.net), Thank you for all the dedication brought up for the idea of community networks.

I would also like to thank my co-authors Ester Lopez (UPC) and Leonardo Maccari (UniTn).

Lastly, I am thankful to my family for their tremendous support, hand holding, and grounding me.

This work was partially supported by the following research projects: European Community Framework Program 7 FIRE Initiative project Community Networks Testbed for the Future Internet (CONFINE) FP7-288535, the H2020 Program netCommons H2020-688768, the Catalan government 2009-SGR-1167, and the Spanish government TIN2013-47245-C2-1-R and TIN2016-77836-C2-2-R.

Summary

Community Networks (CNs) are naturally open and decentralized structures, that grow organically with the addition of heterogeneous network devices, contributed and configured as needed by their participants. The continuous growth in popularity and dissemination of CNs in recent years has raised the perception of a mature and sustainable model for the provisioning of networking services that does not depend on the investment and willingness of profit-oriented traditional network service providers.

However, because such infrastructures include uncontrolled entities with non delimited responsibilities, every single network entity does indeed represent a potential single-point of failure that can stop the entire network from working, and that no other entity can prevent or even circumvent. Given the open and decentralized nature of CNs, that brings together individuals and organizations with different and even conflicting economic, political, and technical interests, the achievement of no more than basic consensus on the correctness of all network nodes is challenging. In such environment, the lack of self-determination for CN participants in terms of control and security of routing can be regarded as an obstacle for growth or even as a risk of collapse.

To address this problem we first consider deployments of existing Wireless Community Networks (WCNs) and we analyze their technology, characteristics, and performance. We start this investigation with the experimental evaluation of a production 802.11an WCN, and compare to studies of other WCNs deployments in the literature. In an effort to better assess the achievable and achieved path performance of existing deployments and its underlying routing protocol, we compare experimentally obtained throughput traces with path-capacity calculations based on well-known conflict graph models. We observe that in the majority of cases the path chosen by the employed BMX6 routing protocol corresponds with the best identified path in our model. We analyze monitoring and interaction shortcomings of CNs and address these with Network Characterization Tool (NCT), a novel tool that allows users to assess network state and performance, and improve their quality of experience by individually modifying the routing parameters of their devices. We also evaluate performance outcomes when different routing policies are in use.

Routing protocols provide self-management mechanisms that allow the continuous operation of a Community Mesh Network (CMN) without requiring the users detailed technological knowledge, permanent monitoring and eventual manual intervention that would be otherwise be required. We focus on three widely used proactive mesh routing protocols and their implementations: BMX6, OLSR, and Babel. We describe the core idea behind these protocols and study the implications of these in terms of scalability, performance, and stability by exposing them to typical but challenging network topologies and scenarios. We evaluated the protocols experimentally using emulation, and the W-ILab.T testbed. That provided us with a realistic wireless medium and respective measurements, especially in terms of interference and CPU consumption. Our results show the relative merits, costs, and limitations of the three protocols.

Built upon the studied characteristics of typical CN deployments, their requirements on open and decentralized cooperation, and the potential controversy on the trustiness of particular components of a network infrastructure, we propose and evaluate SEMTOR, a novel routing-protocol that can satisfy these demands. SEMTOR allows the verifiable and undeniable definition and distributed application of individually trusted topologies for routing traffic towards each node. One unique advantage of SEMTOR is that it does not require a global consensus on the trustiness of any node and thus preserves cooperation among nodes with even oppositional defined trust specification. This gives each node admin the freedom to individually define the subset, and the resulting sub-topology, from the whole set of participating nodes that he considers sufficiently trustworthy to meet their security, data-delivery objectives and concerns.

The proposed mechanisms have been realized as a usable and open-source implementation called BMX7, as successor of BMX6. We have evaluated its scalability, contributed robustness, and security. Security achievements have been analyzed regarding the correctness of specified guarantees and the performance that our implementation achieves, when encountering an attack of a wrongly trusted adversary, by propagating and applying respectively-updated trust specifications. Scalability has been analyzed with respect to network characteristics that are typical for existing network deployments such as size, density, dynamics, and the limited processing capabilities of nodes, but also with respect to new parameters such as the strength of crypto parameters or the frequency of reconfigured trustiness. These results show that the usage of SEMTOR for securing trusted routing topologies is feasible, even when executed on real and very cheap (10 Euro, Linux SoC) routers as commonly used in CMNs.

Resumen

Las Redes Comunitarias (CNs) son estructuras de naturaleza abierta y descentralizada, que crecen orgánicamente con la adición de dispositivos de red heterogéneos que aportan y configuran sus participantes según sea necesario. El crecimiento continuo en la popularidad y difusión de las CNs en los últimos años ha ampliado la percepción como un modelo maduro y sostenible para la provisión de servicios de rede que no dependan de la inversión y disposición de proveedores comerciales de servicios de red tradicionales.

Sin embargo, debido a que estas infraestructuras incluyen entidades con responsabilidades poco delimitadas, cada entidad en la red puede representar un punto de fallo que puede impedir que la red funcione y que ninguna otra entidad pueda prevenir o eludir. Dada la naturaleza abierta y descentralizada de las CNs, que agrupa individuos y organizaciones con diferentes e incluso contrapuestos intereses económicos, políticos y técnicos, conseguir poco más que un consenso básico sobre los nodos correctos en la red puede ser un reto. En este entorno, la falta de autodeterminación para los participantes de una CN en cuanto a control y seguridad del encaminamiento puede considerarse un obstáculo para el crecimiento o incluso un riesgo de colapso.

Para abordar este problema consideramos primero las implementaciones de redes comunitarias inalámbricas (WCN) y se analiza su tecnología, características y desempeño. Comenzamos esta investigación con la evaluación experimental de una WCN 802.11an establecida y se compara con estudios de otros despliegues similares en la literatura. Para evaluar mejor el rendimiento alcanzable y alcanzado por las implementaciones existentes y el protocolo de encaminamiento utilizado, comparamos las trazas de rendimiento experimentales con cálculos de la capacidad de los caminos basados en modelos bien conocidos del grafo. Se observa que en la mayoría de los casos el camino elegido por el protocolo de encaminamiento BMX6 corresponde con el mejor camino identificado en nuestro modelo. Analizamos las limitaciones de monitorización e interacción en CNs y los tratamos con NCT, una nueva herramienta que permite a los usuarios evaluar el estado y rendimiento de la red, y mejorar la calidad de experiencia modificando los parámetros de sus dispositivos individuales. También evaluamos el rendimiento resultante para diferentes políticas de encaminamiento.

Los protocolos de encaminamiento proporcionan mecanismos de autogestión que hacen posible el funcionamiento continuo de una red comunitaria “mesh” (CMN) sin obligar a sus usuarios a tener un conocimiento tecnológico detallado, hacer un seguimiento permanente y las eventuales intervenciones manuales que de otro modo serían necesarias. Nos centramos en tres protocolos de encaminamiento proactivos para redes “mesh” ampliamente utilizados y sus implementaciones: BMX6, OLSR y Babel. Se describe la idea central de estos protocolos y se estudian la implicaciones de éstos en términos de escalabilidad, rendimiento y estabilidad al exponerlos a topologías y escenarios de red típicos pero exigentes. Los protocolos se evalúan de forma experimental en una red emulada y con la red experimental W-ILab.T. Esto ha proporcionado un entorno inalámbrico realista, así como las mediciones correspondientes, especialmente en términos de interferencia y consumo de CPU. Nuestros resultados muestran los méritos, costes y limitaciones de los tres protocolos.

A partir de las características analizadas en despliegues típicos de redes comunitarias, y de las necesidades en cuanto a cooperación abierta y descentralizada, y la esperable divergencia sobre la confiabilidad en ciertos componentes de la infraestructura de red, proponemos y evaluamos SEMTOR, un nuevo protocolo de encaminamiento que puede satisfacer estas necesidades. SEMTOR permite definir de forma verificable e innegable, así como aplicar de forma distribuida, topologías de confianza individualizadas para encaminar tráfico hacia cada nodo. Una ventaja única de SEMTOR es que no precisa de consenso global sobre la confianza en cualquier nodo y por tanto preserva la cooperación entre los nodos, incluso con especificaciones de confianza definidas por oposición. Esto proporciona a cada administrador de nodo la libertad para definir el subconjunto, y la sub-topología resultante, entre el conjunto de todos los nodos participantes que considere dignos de suficiente confianza para cumplir con su objetivo y criterio de seguridad y entrega de datos.

Los mecanismos propuestos se han realizado en forma de una implementación utilizable de código abierto llamada BMX7, como sucesor de BMX6. Se ha evaluado su escalabilidad, así como la robustez y seguridad que aporta. Se han analizado los logros en seguridad en cuanto a la corrección en las garantías especificadas y el rendimiento que consigue nuestra implementación frente a un ataque de un adversario erróneamente considerado de confianza, al propagar y aplicar las especificaciones de confianza actualizadas. Se ha analizado la escalabilidad con respecto a las características de red típicas en implementaciones de red existentes, en cuanto a tamaño, densidad, dinamismo, capacidades de procesamiento limitado en los nodos, así como respecto a nuevos parámetros tales como fortaleza de los parámetros criptograficos o frecuencia de reconfiguración de la confianza. Estos resultados demuestran que el uso de SEMTOR para asegurar topologías de encaminamiento de confianza es factible, incluso cuando se ejecuta en routers reales y muy baratos (10 Euro, Linux SoC), utilizados de forma habitual en redes comunitarias inalámbricas.

List of Publications

[1] Neumann, A., Lopez, E. & Navarro, L. *An evaluation of BMX6 for community wireless networks* in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on* (Oct. 2012), 651–658

[2] Cerdà-Alabern, L., Neumann, A. & Escrich, P. *Experimental Evaluation of a Wireless Community Mesh Network* in *Proceedings of the 16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (ACM, Barcelona, Spain, 2013), 23–30

[3] Cerdà-Alabern, L., Neumann, A. & Escrich, P. *Experimental Evaluation of Wireless Mesh Networks: A Case Study and Comparison* in *NICST'2013, New and smart Information Communication Science and Technology to support Sustainable Development* (Clermont Ferrand, France, Sept. 2013)

[4] Neumann, A., Navarro, L., Baig, R. & Escrich, P. *Receiver-driven routing for community mesh networks* in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a* (June 2013), 1–7

[5] Cerdà-Alabern, L., Neumann, A. & Maccari, L. *Experimental Evaluation of BMX6 Routing Metrics in a 802.11an Wireless-Community Mesh Network* in *2015 3rd International Conference on Future Internet of Things and Cloud* (Aug. 2015), 770–775

[6] Centelles, R. P., Oncins, V. & Neumann, A. Enhancing reflection and self-determination in a real-life community mesh network. *Computer Networks* **93, Part 2.** Community Networks, 297–307 (2015)

[7] Neumann, A., Baig, R. & Oncins, V. *Mobility performance Evaluation of Mesh rOuting protocols (MEMO)* tech. rep. (Fed4FIRE, Dec. 2014)

[8] Neumann, A., López, E. & Navarro, L. Evaluation of mesh routing protocols for wireless community networks. *Computer Networks* **93, Part 2.** Community Networks, 308–323 (2015)

[9] Neumann, A., Lopez, E., Cerdà-Alabern, L. & Navarro, L. *Securely-entrusted multi-topology routing for community networks* in *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)* (Jan. 2016), 1–8

[10] Neumann, A., Navarro, L. & Cerdà-Alabern, L. Enabling Individually Entrusted Routing Security for Open and Decentralized Community Networks. *Under review for Ad Hoc Networks* (2017)

Other Publications

[11] Neumann, A., Vilata, I., León, X., Escrich Garcia, P., Navarro, L. & López, E. *Community-Lab: Architecture of a Community Networking Testbed for the Future Internet* in *1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB'2012)* (Barcelona, Spain, Oct. 2012), 628–635

[12] Navarro, L., Escrich, P., Baig, R. & Neumann, A. *Community-Lab: Overview and Invitation to the Research Community* in *IEEE International Conference on Peer-to-Peer Computing* (IEEE Press, Tarragona, Sept. 2012)

[13] Escrich, P., Baig, R., Vilata, I., Neumann, A., Aymerich, M., Lopez, E., Vega, D., Meseguer, R., Freitag, F. & Navarro, L. *Community home gateways for P2P clouds* in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (Sept. 2013), 1–2

[14] Escrich, P., Baig, R., Dimogerontakis, E., Carbo, E., Neumann, A., Fonseca, A., Freitag, F. & Navarro, L. *WiBed, a platform for commodity wireless testbeds* in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on* (Oct. 2014), 85–91

Community Events

[15] Neumann, A. *"BMX6 Mesh Routing Protocol, Reclaim Your Mesh – Self-paced Networking in Multilateral Environments"* International Summit for Community Wireless Networks (IS4CWN). Berlin/Germany, 2013

[16] Neumann, A. *"Trusted Multi-Topology Routing with BMX6"* Wireless Battle of the Mesh version 7 (WBMv7). Leipzig/Germany, May 2014

[17] Neumann, A. *"Individually-Secured Mesh Routing with BMX7: Advances and Integration"* Wireless Battle of the Mesh version 9 (WBMv9). Porto/Portugal, May 2016

[18] Neumann, A. *"Secure and Decentralized Distance-Vector Routing with BMX7"* Seminar at Berlin Freifunk Community. Berlin/Germany, July 2016

Contents

List of Acronyms

AHC	Anchored Hash Chain
AP	Access Point
AS	Autonomous System
BGP	Border Gateway Protocol
BMX6	BatMan-eXperimental version 6
CDF	Cumulative Distribution Function
CGA	Cryptographically Generated Address
CN	Community Network
CMN	Community Mesh Network
CNML	Community Network Mark Up Language
DAT	Direct Air Time
e2e	end to end
EB	Expected Bandwidth
EQ	Expected Quality
ETT	Expected Transmission Time
ETX	Expected Transmission Count
GSF	Gràcia Sense Fils
GW	Gateway
HC	Hop Count
ISP	Internet Service Provider
MAN	Metropolitan Area Network
MANET	Mobile Ad-hoc NETwork
MB	Multiply Bandwidth
MIMO	Multiple input, multiple output

MP	Multiply Path
MQ	Multiply Quality
MTR	My Trace Route
NCT	Network Characterization Tool
NCd	NCT daemon
NCui	NCT user interface
OS	Operating System
OSPF	Open Shortest Path First
qMp	Quick Mesh Project
RIP	Routing Information Protocol
RP	Routing Protocol
RQA	Research Question A
RQB	Research Question B
RTT	Round-Trip Time
SNMP	Simple Network Management Protocol
UPC	Universitat Politècnica de Catalunya
VB	Vector Bandwidth
WCN	Wireless Community Network
WMN	Wireless Mesh Network

Chapter 1

Introduction

Computer networks have become a vital line of today's society and provide the key enabler for many omnipresent tools such as the Internet and all the various applications built on it.

Unfortunately, yet only a fraction of the world's population has adequate digital access. This includes not only the majority of third-world countries but, looking at regional scale, also many rural areas in industrial countries such as Germany or Spain, where the lack of profit prospects discourages Internet Service Providers (ISPs) to invest in the infrastructure needed to meet even the basic requirements.

To overcome this divide, and stimulated by the continuously increasing pressure for digital access plus the popularity and availability of inexpensive IP-network and licence-free WIFI equipment, inhabitants and organizations of disconnected areas have joined efforts and started creating their own networks. The so called WCNs have experienced an exponential growth in recent years, causing the arise of network infrastructures covering thousands of square kilometres and interconnecting tenths of thousands of nodes [19]. Due to their unplanned evolution, they are naturally open and decentralized structures, growing organically with the addition of heterogeneous network devices that are contributed and configured as needed by individual participants.

Technically, like in any IP-based network, the abstraction from topology-detection, routing, and forwarding related tasks results in a transparent end-to-end connectivity service for distributed application processes that is provided by the network layer. This layer, although distributed and relying on cooperation by nature, is not necessary decentralized or guaranteeing the proper functioning of its entities. That, combined with a routing algorithm that determines routes based on link parameters, restrict the freedom of community users to express preference or avoid certain nodes in paths.

In the traditional case, where Internet backbone and last-mile networks are maintained by large and profit-oriented ISPs, centrality and single-point of failures concentrate in higher hierarchical layers of the involved organizations, which, following a clear customer-relation model and interest for providing the promised quality of service, can relatively easily set up responsibilities and measures to prevent, or at least react on, outages.

However, in the case of open CN, where the network infrastructure is, by design, composed also of uncontrolled entities with unclear responsibilities, every single network entity represents a single-point of failure that can prevent the entire network from working and that no other entity can avoid or even circumvent.

With the objective to narrow down the wide scope addressed by the title of this thesis into a concrete, relevant, and practical scenario we decided to apply its key subjects to the case of community networks and formulate the following overall research question:

How to cooperate decentralized, efficiently and reliably on the collaborative provisioning of routing services in open and heterogeneous community networks?

In the following, the basic components related with this question, being the case of CNs in general, the routing protocols which provide the core function of the underlying network layer, and the aspect of decentralization and robustness of this layer, are elaborated in more detail.

1.1 Background

The context of this research is crowdsourced computer networks: network infrastructures built by citizens and organizations who pool their resources and coordinate their efforts [20]. The goal of most of these networks is to provide non-discriminatory and open access to the shared network and be open for the participation of any member of the community in the governance of the network. Therefore we first define what is a community network. One of the popular technological choices in CNs, especially in urban areas, is wireless mesh networks. Therefore we look at the characteristics and requirements of such networks and the used mesh routing protocols. Finally, we focus on the limitations of current mesh routing protocols in handling decentralization and resilience, as these protocols assume that all routing processes in nodes operate correctly and nodes and their users have no influence in the choice of paths of forwarding nodes.

1.1.1 Community Networks

CNs [21–24] are supposedly open and decentralized structures, growing and evolving organically as network infrastructure is contributed, deployed, and configured by individual participants.

The operation of such networks is based on the principle of cooperation among its members. These communities usually have participation rules as a membership license or peering agreement [19, 25, 26], that define their freedom, openness and neutrality.

However, for cost reasons and due to the open participation rules, infrastructure contributions often consist of inexpensive wireless routers [23] with usually limited storage, bandwidth, and computation capabilities that can be mounted at low cost on rooftops or even isolated rural locations powered by solar energy. Links, manually or automatically established between neighboring nodes, then create a restricted-route network without universal direct connectivity between all mesh nodes [27, 28] and with very heterogeneous and dynamic characteristics. In addition, because of potential poor hardware quality, neglected maintenance, or remote location, they may be technically impaired or even tampered by an adversary to launch an attack from inside the network.

Typically, the deployment of such networks, such as Guifi.net [24] with more than 30,000 total active nodes, is structured in clouds, groups consisting of up to hundreds of nodes, that constitute an Autonomous System (AS), operate its cloud-specific internal routing protocol (e.g. OSPF, OLSR), and peer with neighboring clouds via an exterior gateway protocol such as Border Gateway Protocol (BGP). Many CNs also integrate into the global Internet as one or serveral Internet ASs. Then exterior gateway protocols are typically used on two levels. On a global level to exchange routing information between the CN AS and other Internet ASs. And inside the CN, on a macroscopic level, to interconnect different CN clouds that, on a microscopic level, operate internal routing protocols for routing within the cloud.

CNs make technology choices and combine them to build, expand, and maintain their networks. Therefore, we have to decouple the overall concept of community networks from a particular technology choice or combination. The integration of wireless equipment is a common choice for the facility of deployment, therefore we can find the term Wireless Community Network (WCN) to refer to that choice and the challenges related with wireless channel characteristics. However, also WCN do not restrict itself to rely only on wireless links as CN often also integrate optical fiber and optical wireless links in their networks. Further, networks can be build based on dedicated and manually configured point-to-point links between nodes or with nodes that autonomously detect and interact with all other nodes that are in communication range of each other. This latter, so called mesh topology, typical preferred for CN clouds in urban areas, is referred to as Community Mesh Network (CMN), The combination of both is referred as Wireless Mesh Networks (WMNs). Other times the focus is on the changing conditions (mobility) of nodes plus the self-configuration (mesh) capabilities in the network, and the term used is Mobile Ad-hoc NETwork (MANET).

1.1.2 Mesh Routing Protocols

Among the basic network-layer functions of neighbor discovery, routing, and forwarding, the second can be regarded as the most critical and challenging. This is because in contrast to the first, the routing shall coordinate decisions beyond the local neighborhood and is responsible for the distributed establishment of consistent end-to-end paths between nodes. On the other hand, the forwarding function can then already rely on the local availability of rules (established by the routing function) and only needs to forward each incoming packet respectively towards its destination. In addition, and especially in the case of organically growing CNs that include links with dynamically changing characteristics and potentially several hops along the path from the source to the destination, a number of further characteristics are desired:

- Compatibility and smooth integration with existing IP networks such as the Internet. Therefore IP compatible addresses are desired and the inter operation with traditional routing protocols must be supported.

- Provisioning of connectivity quality similar to that experienced from traditional ISPs. Therefore instant route availability as provided by proactive routing protocols should be supported. In contrast to this, reactive routing protocols do not start discovering routes before they are actually needed.

- Support for autonomous neighbor discovery and flat (virtually random) addressing schemes is important to consider new nodes, links, and paths without manual interaction and avoid the need to coordinate address allocations with the topological or physical location of a node.
- Self-healing and optimizing capabilities to continuously adapt and optimize routing decisions to link outages and changes.
- Support the topological characteristics of CN deployments such as the order of expected links and nodes.
- Meet resource constraints of community participants. For example, the processing requirements of a protocol should not exceed the computation capabilities of inexpensive wireless equipment that stimulated the emergence of these networks in the first place.
- Satisfy ethical and usability requirements of network communities. For example, to increase transparency and control, an actively maintained and open-source protocol implementation that smoothly integrates with OS distributions typically used for low-cost routing hardware might be favored.

1.1.3 Network Layer Decentralization and Resilience

The previous introduction to mesh routing protocols actually assumed only correctly operating routing processes. However, in an open network, where impaired or even malicious nodes can be added at any time, this is not necessarily the case.

Thinking further, in an open and decentralized CN that brings together individuals and organizations with different and even conflicting economic, political, and technical interests [21, 29, 30] the achievement of no more than basic consensus on the (in-)correctness of all existing nodes is at least hard. This could go as far that some users may insist to disapprove particular nodes because of suspected passive traffic inspection while other less-critical users heavily rely on exactly these. Thus, openness and decentralization is expected to enrich controversy while full consensus may only be given among certain subsets of the overall user group.

In summary, although the community may be united by the common ideal to share networking resources such as routing hardware and links, different interests may hinder the cooperation among them and raise the risk of exclusion and partitioning, affecting the resilience of the network.

1.2 Problem Statement

The continuously, if not exponentially, growth of popularity and dissemination of CN in recent years has raised the perception of a mature and sustainable concept for the provisioning of fast networking services that do not depend on the investment and willingness of profit-oriented traditional ISPs.

In fact, the growth rate in a CN reflects the diverse issues that appear over time. Figure 1.1 shows the example of the Guifi.net CN, where the monthly growth rate peaked in 2013 due

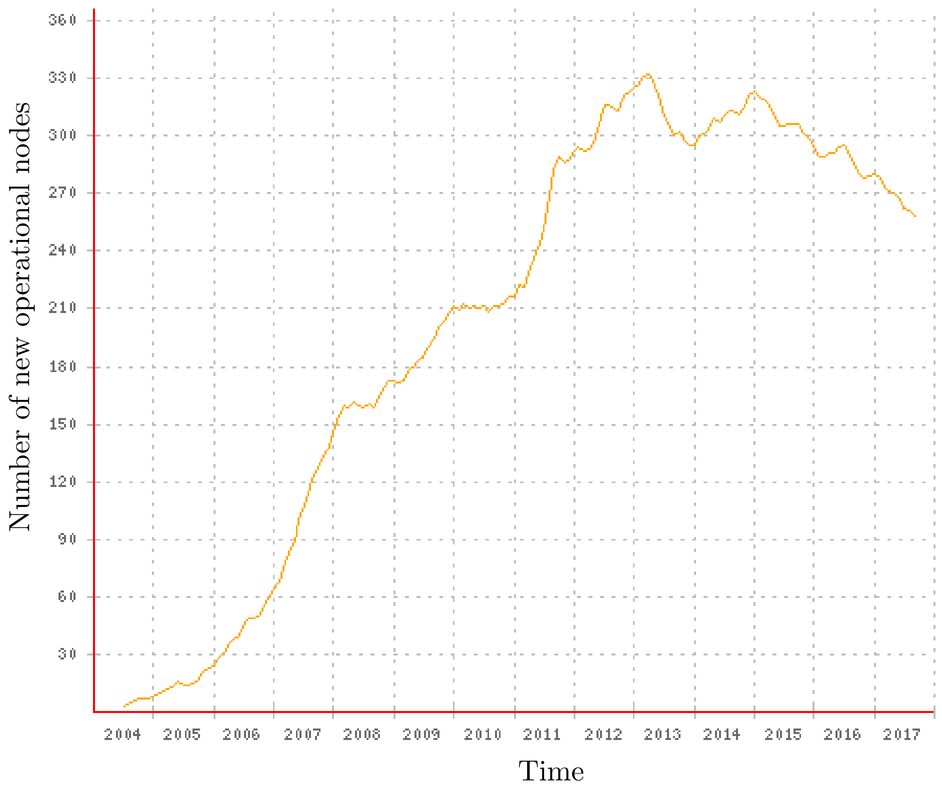


Figure 1.1: Rate of monthly growth in number of nodes for the Guifi.net CN (Source: `http://guifi.net/guifi/menu/stats/nodes`)

to diverse reasons that, as discussed with and among the community, are mainly seen in scalability issues with the management of the network and the governance of the community due to its growing size and diversity in aspects such as topology (point-to-point vs. mesh), diversity of technologies (wireless vs. optical), attitudes (active, passive, opportunistic, or even abusive participants), or levels of participation (active vs. silent participants).

In such environment, the lack of self-determination for CN participants in terms of control and security, can be regarded as a potential obstacle for their further growth or even as a risk for their eventual collapse.

With the objective to target this problem, the general research question of how to cooperate decentralized, efficiently and reliably on the collaborative provisioning of routing services in open and heterogeneous community networks, can be decomposed into one major and one necessary research question and several respective aspects to be considered.

The major question, is formulated as *Research Question B (RQB):*
How to increase the user freedom in terms of self-determination and security while preserving openness, decentralization, and cooperation and satisfying the constraints imposed by nowadays existing CN ?

We look for solutions to this question that respect the technologies prevailing in nowadays CN deployments and facilitate a smooth integration into the concept and architecture of such networks. For that reason, we consider pragmatic and obvious constraints, together with advanced constraints that demand further clarification.

Regarding the obvious constraints, the following further assumptions and consequential restrictions were imposed:

- IP-based routing is the most widely accepted standard for the provisioning of networking-layer services in computer networks in general and also for CNs. Adoption of the IP-protocol architecture and its respective APIs facilitates easy integration in a wide range of already existing network environments, Operating Systems (OSs), and tool chains.
- CN clouds represent the lowest organizational level for networking within a (potentially larger) CN deployment where the overall network is structured in different clouds, each constituting an AS that interconnect with others via an exterior gateway protocols such as BGP.
- Topology independent (flat) addressing and autonomous neighbor and topology detection to support organic evolution and growth of CN clouds. As a consequence we concentrate on approaches suitable for mesh network clouds. In the following, the term mesh network will be used synonymously for MANET (a term that is often used in the literature to characterize unplanned and dynamic wireless networks) and the term CMN for a CN cloud.
- Proactive route establishment, healing, and optimization support to reflect the end-to-end connection latencies known from traditional ISPs.

The development of practical solutions for RQB in this work takes place under the constrains just defined: within the scope of mesh network deployments, and focusing on the routing protocols. In this scope, the mesh topology typically provides alternative links to nodes, and the routing protocol has the choice to select the forwarding paths across network nodes. Therefore, to understand the characteristics and challenges faced, a second research question is formulated within this scope as *Research Question A (RQA):*
What are the constraints in terms of properties, requirements, and accepted limitations of existing mesh-network deployments and its used routing protocols?

Aiming for solutions that do not burden the community with structural or conceptual changes, new expenditures or drawbacks in the experienced service quality, the following aspects of advanced constraints shall be assessed and satisfied respectively:

- Topology: Current and forthcoming topological properties, such as the number of nodes (size), the average and maximum number of interfaces and links per node (density), the frequency and impacts of changes such as the emergence of new nodes and links (dynamics), and the diversity of these components, which may range from unstable and failure-prone devices or links (e.g. wireless) to dedicated and high-performance equipment.
- Openness: Participation should not be hindered by the enforcement of controversial policies. Therefore solutions should be technologically neutral, not depend on any central component, and provide user-centric configuration flexibility without disturbing the distributed collaboration of components controlled by different users.
- Performance: Convergence, and route optimization capabilities should not degrade. Convergence describes how quickly the routing can adapt to and heal from topological

changes such as link or node outages or the emergence of new links or nodes. Path optimization describes how well a selected route reflects the best possible selection of existing end-to-end paths and how quickly this can be achieved.

- Resources: The imposed processing and communication overhead in terms of memory, CPU usage, transmitted protocol data, should range in the order of what currently existing solutions consume and should meet with the available resources provided by even the weakest components of the overall cloud infrastructure to which these requirements apply. For example if a potential solution implies cryptographic calculations that must be performed by all nodes then it should be ensured either that nodes not being able to satisfy this are so exceptional that they can be replaced with little effort or that such operations are sufficiently lightweight that even the weakest nodes can perform them.

Coming back to the previously formulated research questions, the problem addressed by RQA is to assess and if possible quantify the above summarized aspects based on the evaluation of protocols and deployments used in the scope of community networks with mesh topology, which implies wireless links (WCNs. The challenge addressed by RQB is then to develop solutions in the selected scope that mitigate the lack of control and security and validate that previously assessed constraints can still be satisfied.

1.3 Contributions

Characteristics and Performance of Wireless Community Mesh Network Deployments: We consider deployments of existing WCNs and analyze their technology, motivations, and achievements. As a first step we focus on characteristics of WCNs such as network topological structure, usage, and evolution. Therefore we perform an experimental evaluation (in Section 2.1) of a production 802.11an WCN deployed by a local community in a quarter of the city of Barcelona that is connected with a testbed at Universitat Politècnica de Catalunya (UPC) and compare this with deployment studies of other WCNs found in the literature.

One of the main metrics of CN users for valuing their expenditures into CN infrastructure is given by the experienced end-to-end throughput between two nodes of the network (typically the users own and his most favored gateway node). Such performance measure is directly related to the path metric responsible for optimizing end-to-end routing based on the availability of existing infrastructure components. In an effort to better assess the achievable and achieved path performance of existing deployments and its underlying routing protocol, we compare in Section 2.2 the experimentally obtained throughput traces with path-capacity calculations based on a well-known conflict graph model. We observe that in the majority of cases the path, chosen by the employed BatMan-eXperimental version 6 (BMX6) routing protocol, corresponds with the best identified path from our model.

In Section 2.3 we analyze monitoring and interaction shortcomings of CN and address these with a novel tool called NCT that allows users to assess network state and performance and improve their quality of experience by individually modifying their devices' routing parameters. We evaluate performance outcomes when different routing policies are in use with a set of experiments using the NCT in a real-life CMN built with the open-source router system Quick Mesh Project (qMp) and the BMX6 routing protocol.

The results related to this contribution are presented in Chapter 2 and were originally reported in [2, 3, 5, 6].

Evaluation of Mesh Routing Protocols: Routing protocols provide the self-management mechanisms that allow the continuous operation of CMN without imposing the detailed technological knowledge, permanent monitoring and respective manual intervention that would be otherwise be required.

In Chapter 3 we focus on the mechanisms of three widely used, proactive, mesh routing protocols and their implementations: BMX6, OLSR, and Babel. We describe the core idea behind each of these protocols and study the implications of these in terms of scalability, performance, and stability by exposing them to typical but challenging network topologies and scenarios. Two different evaluation approaches were used: emulation, that provides comprehensive high-level control and repeatability, and allows to easily apply topological characteristics obtained from real-world deployments; and the W-ILab.T testbed at iMinds, that provided us with a realistic wireless medium and respective measurements, especially in terms of interference and CPU consumption. Our results show the relative merits, costs, and limitations of the three protocols.

The results related to this contribution are presented in Chapter 3 and were originally reported in [1, 7, 8].

Individually Entrusted Routing: Building upon the studied characteristics of typical CN deployments, their requirements on open and decentralized cooperation, and the expected controversy on the trustiness of particular components of an overall network infrastructure, we propose and evaluate a novel routing-protocol called SEMTOR that can satisfy these demands.

SEMTOR allows the verifiable and undeniable definition and distributed application of individually trusted topologies for routing traffic towards each node. One unique advantage of SEMTOR is that it does not require a global consensus on the trustiness of any node and thus conserves cooperation among nodes with even oppositional defined trust specification. This gives each node admin the freedom to individually define the subset (and resulting sub topology) from the whole set of participating nodes that he considers sufficiently trustworthy to meet his security and data-delivery objectives and concerns.

The proposed mechanisms have been realized as an usable and open-source implementation called BMX7 (as the successor of the BMX6) and evaluated for their scalability and contributed robustness and security. Security achievements have been analyzed regarding the correctness of specified guarantees and the observed performance with which our implementation achieves to encounter an attack of a wrongly trusted adversary by propagating and applying respectively-updated trust specifications. Scalability has been analyzed with respect to network characteristics that are typical for existing WMN deployments such as size, density, and dynamics, and limited processing capabilities of nodes, but also with respect to new parameters, such as the strength of crypto parameters used or the frequency of reconfigured trustiness. These results also show that the usage of asymmetric cryptography for securing trusted routing-topologies is possible, even when executed on real and very cheap (10 Euro, Linux SoC) embedded routers as commonly used in CMN.

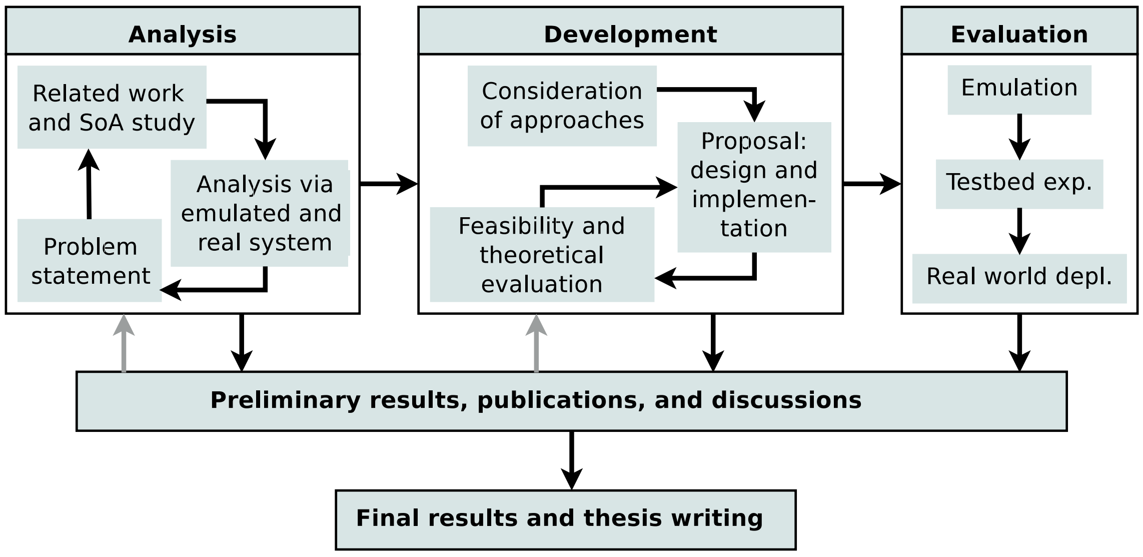

Analysis
Related work and SoA study
Analysis via emulated and real system
Problem statement
Development
Consideration of approaches
Proposal: design and implemen-tation
Feasibility and theoretical evaluation
Evaluation
Emulation
Testbed exp.
Real world depl.
Preliminary results, publications, and discussions
Final results and thesis writing

Figure 1.2: Research guidelines

The results related to this contribution are presented in Chapter 4 and were originally reported in [4, 9, 10].

1.3.0.1 Integrations in Community Projects

- The Network monitoring tool developed during the work on Section 2.1 is available as open source and actively used by several qMp based CMNs in Barcelona. Their current status can be queried online at `http://tomir.ac.upc.edu/qmpmon`
- NCT from Section 2.3 integrated in experimental branch of qMp[31].
- Capacity aware routing metric based on path-capacity model from Section 2.2 integrated as default metric algorithm in BMX7.
- SEMTOR protocol from Chapter 4 integrated in BMX7.
- BMX7 integrated in Lede/OpenWrt [32] routing packages, Libremesh[33] and qMp[31] CN distributions, and current Battlemesh [34] experimentation repositories.

1.4 Methodology

With the objective to answer the two main research questions, we followed an iterative and experimentation driven approach that includes four main phases, being the analysis, design, evaluation, and discussion phase. An overview of the this work flow is given in Figure 1.2 and described together with considered data sources, design and experimentation guidelines in the following.

1.4.1 Analysis

To understand the state-of-the art on the research topics addressed by this PhD we combined the study of related work with opportunities and resources from the field that go in the classical literature and of which the most relevant shall be briefly summarized:

- Our close interaction and collaboration with existing network communities such as qMp [31, 35], Guifi.net [36] and Freifunk [37] and our active participation in related community events such as the annual Wireless Battlemesh Conference [34] provided valuable insights and awareness for the prevailing approaches and most bothering challenges from the field, yielding continuous feedback on the findings that we could achieve.

- The mutual trust established with the communities allowed us to get full access to several productively used CN deployments and enabled us to perform experiments, and even test some of our solutions, in real environments.

- Publicly available CN meta-data, such as the Community Network Mark Up Language (CNML), that describes status, or the topology of existing CN deployments, has been used to understand the structure of these networks and to feed experiments with representative configurations.

The collected knowledge helped to substantiate the addressed scope and draft the assumptions, restrictions, and challenges to be considered. Selected aspects of this study are analyzed more comprehensively by experimenting with real, and if necessary emulated, systems to understand their characteristics and benefits. Relevant findings obtained during this phase are discussed with the research as well as the networking community which we are considering as our main target group. The resulting feedback eventually stimulated further input and iterations and helped us to define and refine the objectives of our work.

1.4.2 Development

During the design and development phase, the knowledge gained from the analysis was used for the development of ideas towards possible approaches to the problem which were then studied regarding their applicability and adaptability on the case of community networks. The design of concepts and mechanisms was addressed iteratively through modeling and refinement. Intermediate results from this approach were checked for feasibility and consistency with the overall architecture. Consequential implications have been estimated based on practical and theoretical consideration and refined or discarded respectively. Once having reached a convenient stage, proposed mechanisms were implemented exploiting prior work as much as possible, for example by extending the functionality of existing routing protocol implementations.

Here, the familiarity with open-source routing systems in general and the implementations of routing protocols in particular, especially our own involvement in the development of the BATMAN and BMX6 [38] routing-protocols, facilitated the opting for pragmatic approaches and the effective alignment of ideas with feasibility and usability considerations.

1.4.3 Evaluation

The evaluation and validation of our work has accompanied the development process with the objective to provide continuous and experimentation-based feedback on the validity and performance of the proposed solutions.

To allow for a relatively fast feedback cycle, an emulation based evaluation approach has been followed. In addition to the observation of our prototype solution under selected and well controllable scenarios, as would be provided also by simulation (including measuring the consequences from its exposure to relatively large virtual deployments), this allowed us the possibility for a later direct reuse of our implementation for real-world experiments. The network emulation tool Mesh Linux Containers (MLC) [39], enabled us the creation of such virtual experimentation scenarios with hundreds of nodes and links while providing an environment that is almost fully consistent with the Linux firmware systems employed by today's open-source router projects.

Having refined the proposed solution with the knowledge gained via emulation, an enhanced implementation has been tested against physical testbeds. Therefore, laboratory testbeds (such as the W-ILab.T testbed at iMinds), community testbeds integrated in real-world deployments (as given by the CONFINE [40] project), or temporary deployed testbeds (as given by the annual Battlemesh [34] conference or the Wibed [14] system deployed at UPC) have been used. The exploitation of such infrastructures did not only facilitate the further optimization of the proposed solution and hardening of its implementation but also yielded discussion with community network activists and the research community. Contacts and confidence gained during these and related opportunities have been used for the dissemination of our contributions. Eventually, existing community-network groups could have been convinced to consider and integrate our achievements which provided further feedback on the quality of experience, usability, and performance from the user's perspective.

1.5 Thesis Structure

The rest of this thesis is organized as follows.

Our efforts to answer the question given by RQA and its related aspects are described in Chapter 2 and Chapter 3. Here the aspects of topology, performance, and openness are in the focus of Sections 2.1, 2.2, and 2.3 respectively while Chapter 3 then builds on our topological findings and focuses primarily on the aspects of performance and resource consumption from a routing-protocol point of view.

In Chapter 4 we address RQB by proposing and validating routing-protocol mechanisms that enhance the self-determination and security of CN users. Here our design, validation and evaluation is driven by the constraints identified in Chapters 2 and 3.

Existing technologies and work from the literature, that address findings and solutions related to the issues we have introduced and worked on, are discussed throughout the Chapters of this thesis.

Chapter 5 concludes and summarizes the findings of this thesis and draws future directions that deserve further work.

Chapter 2

Evaluation of Community Network Deployments

In this chapter we focus on the evaluation of constraints and properties of existing CN deployments.

We start in Section 2.1 with the experimental evaluation of a production 802.11an Wireless Community Network (WCN), deployed by a local community in a quarter of the city of Barcelona that is connected with a testbed at the Universitat Politècnica de Catalunya (UPC), and compare our findings with deployment studies of other commercial and non-commercial wireless networks found in the literature.

In Section 2.2 we complement these studies by investigating the achievable and achieved path performance of this deployment and its underlying routing protocol. Therefore, we compare throughput traces obtained experimentally with path-capacity calculations based on a well-known conflict graph model. We observe that in the majority of cases the path, chosen by the employed BMX6 routing protocol, corresponds with the best identified path from our model.

We discuss in Section 2.3 about monitoring and interaction shortcomings of CN. For that, we describe the development, integration, and usage of a tool that allows users to assess topology and performance characteristics of their network and improve their quality of experience by interacting with parameters of their used routing-protocol.

2.1 Experimental Case Study and Comparison

Community networks are deployed and maintained by their own users. Unlike the model used by the traditional telecommunication companies (which are business-focused), each user is owner of a part of the total infrastructure which builds the mesh network. Using an organization system (i.e. web site) they are able to understand each other and connect with neighbors, neighbors of neighbors and so on. These networks are normally open and free. Most of them use WiFi technology because it is the easiest and cheapest way to deploy an outdoor network.

A relevant example is Guifi.net [26, 36, 41, 42], the largest currently existing community network, having more than $30,000$ operational nodes. Guifi.net has been deployed in urban and rural areas of Spain (mostly Catalonia). Probably one of the main reasons of Guifi.net's success is the fact that it operates as an umbrella for many other small communities. Each community uses its own kind of hardware, software and organization methods (meetings, mailing lists, etc.). But all of them share probably the most important part of the Guifi.net community, the web page. It is used mainly to distribute the IPs and federate the small networks using a common system.

Guifi.net has become a rather complex and heterogeneous network, merging wired and wireless links. Most of Guifi.net's infrastructure is wireless and the OSPF and BGP routing protocols predominate in the network.

Inside of Guifi.net the *Quick Mesh Project* (QMP) [31], an OpenWRT [32] distribution, is being developed using BMX6 as routing protocol [1]. This distribution has been adopted by one of the network communities which operate under the Guifi.net umbrella in the quarter of Sants in Barcelona, Spain. For the sake of brevity we shall refer to the quarter of Sants simply as Sants in the rest of the paper. Additionally, the EU CONFINE project [43] deployed a research testbed at *Universitat Politècnica de Catalunya* (UPC) using QMP. The CONFINE testbed has been linked to Sants, creating a network referred to as *QMPSU* (from Quick Mesh Project at Sants + UPC) and as illustrated in figure 2.1[1]. The QMPSU network model is expanding among other districts of Barcelona. It grows very fast and other Guifi.net groups in the city have become interested, and started the migration.

In this chapter we present an experimental evaluation of QMPSU and later compare it with other existing community networks. To do so, live measurements have been taken hourly over the last 5 months. We use this data to analyze main aspects of QMPSU. These include topology, usage and network performance and characterization. To our best knowledge, this is the first academic work where a production community wireless network using 802.11an is analyzed.

The rest of section 2.1 is organized as follows: More details about QMP project and QMPSU are given in sections 2.1.1 and 2.1.2. The measurements methodology is explained in section 2.1.3, followed by the analysis of the topology, Internet access and Usage in sections 2.1.4, 2.1.5 and 2.1.6. Then, wireless links are characterized in section 2.1.7. Related work is discussed in section 2.1.8 together with a comparison in section 2.1.9 highlighting differences between the QMPSU and other existing networks. Some concluding remarks are given in section 2.1.10.

2.1.1 Quick Mesh Project

QMP [31] is an operating system for embedded network devices based on OpenWRT/Linux. It was started by some Guifi.net activists in 2011 with the objective to provide a fully open-source solution to easily and quickly deploy a mesh network and share Internet uplinks between it's users.

As main routing protocol QMP uses BMX6 [1], a destination-sequenced distance-vector protocol using UDP messages to discover other nodes and disseminate node and routing information. Some extra features have been specially developed for QMP, such as a

[1]QMPSU web site: `http://Guifisants.net`.

smart gateway selection using IPIP tunnels or a short message plugin which permits to send arbitrary information to other nodes, piggybacked by the protocol packages. BMX6 obtained very good results compared with other mesh routing protocols tested in various Wireless Battle Mesh iterations such as v6 in Aalborg (Denmark) 2013 and v7 in Leipzig (Germany) in 2014.[2] Other important characteristics of QMP are the native IPv6 and full auto-configuration support. Each node auto-configures its own IPv6 address based on a ULA[3] prefix. IPv4 connections are enabled via tunnels over the ULA-based IPv6 network.

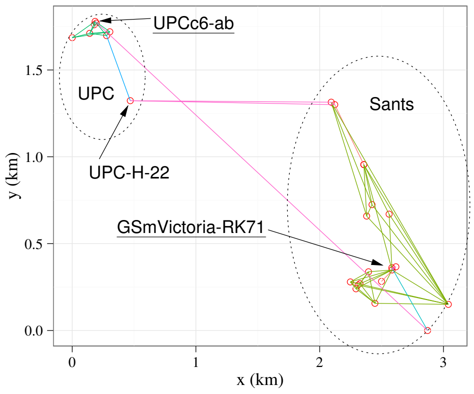


Figure 2.1: QMPSU network. Two main gateways are underlined.

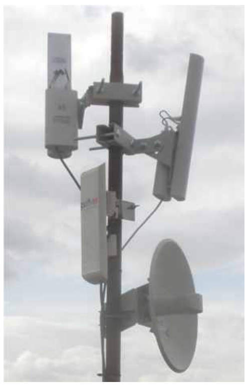
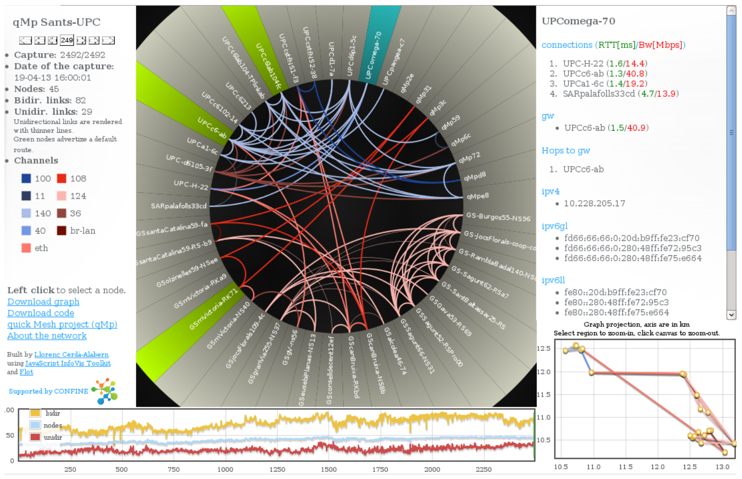


Figure 2.2: Supernode.



qMp Sants-UPC

- **Capture:** 2492/2492
- **Date of the capture:** 19-04-13 16:00:01
- **Nodes:** 45
- **Bidir. links:** 82
- **Unidir. links:** 29
  Unidirectional links are rendered with thinner lines.
  Green nodes advertize a default route.
- **Channels**

🟦	100	🟥	108
🟦	11	🟥	124
🟦	140	🟥	36
🟦	40	🟥	br-lan
🟥	eth		

Left click to select a node.
Download graph
Download code
quick Mesh project (qMp)
About the network

Built by Llorenç Cerdà-Alabern using JavaScript InfoVis Toolkit and Flot

Supported by CONFINE

UPComega-70

connections (RTT[ms]/Bw[Mbps])

1. UPC-H-22 (1.6/14.4)
2. UPCc6-ab (1.3/40.8)
3. UPCa1-6c (1.4/19.2)
4. SARpalafolls33cd (4.7/13.9)

gw

- UPCc6-ab (1.5/40.9)

Hops to gw

1. UPCc6-ab

ipv4

- 10.228.205.17

ipv6gl

- fd66:66:66:0:20d:b9ff:fe23:cf70
- fd66:66:66:0:280:48ff:fe72:95c3
- fd66:66:66:0:280:48ff:fe75:e664

ipv6ll

- fe80::20d:b9ff:fe23:cf70
- fe80::280:48ff:fe72:95c3
- fe80::280:48ff:fe75:e664

Graph projection, axis are in km
Select region to zoom-in, click canvas to zoom-out.

Figure 2.3: QMPSU network web page.

2.1.2 QMPSU Architecture

QMPSU is a wireless, 802.11-based, mulit-hop, semi-static, and productively used mesh network where (topologically) all nodes provide the same routing functionality for relaying other node's traffic. Most of the network users only have the mesh network for reaching the Internet, so they depend on the community. In consequence, stability and good performance of the network are mandatory points. QMPSU is not a pure MANET network, it is being

[2]Experiments description and raw measurements data online at `http://battlemesh.org/BattleMeshV6/ActualTests`, `http://battlemesh.org/BattleMeshV7/Tests`, and `https://github.com/axn/wbm2pdf/tree/wbmv7`

[3]RFC4193 Unique Local Address

deployed following a model named by the participants: *SmartMesh*. The concept tries to explain a model placed between the complete unmanaged model used in many of the known Mesh/MANET networks and the complete organized model used in the AP/Infrastructure networks. The main points which define the *SmartMesh* deployment idea are:

- Use patch antennas with $< 90°$ as main equipment in AD-HOC mode.
- Use parabolic antennas ($< 6°$) for long shots (point-to-point links).
- Use two or more devices for some strategic locations (super-nodes).
- Use different channels to multiplex the spectrum (super-nodes make the interconnection).
- Use only 802.11an to reduce the noise and interference (spectrum space is bigger than 802.11bgn).
- Use only stable links and devices placed in the outdoor (roof or balcony).

These requirements imply the inconvenience of planning in advance the deployments and links and requires to manually orientate antennas during the installation. But it has the advantage of reduced interference (one of the biggest problems in ISM-spectrum based networks) and fewer neighboring nodes but with better links. Following this model, reasonable end-to-end performance could be achieved even over 7 ore more hops.

The most common hardware used in QMPSU is NanoStation M5[4], which has the following characteristics: Antenna 5GHz 16dBi, Processor Atheros MIPS 400MHz, Flash Memory 8MB, SDRAM Memory 32MB, two Ethernet ports 10/100, Radio Atheros 9k 802.11AN MiMo 2T2R. For point-to-point long shots the typically used hardware is NanoBridge M5[5], a variant of the NanoStation with parabolic antenna. Finally for super-nodes, the most common equipment is a couple of Rocket M5[6] with 120° sector antennas (see figure 2.2).

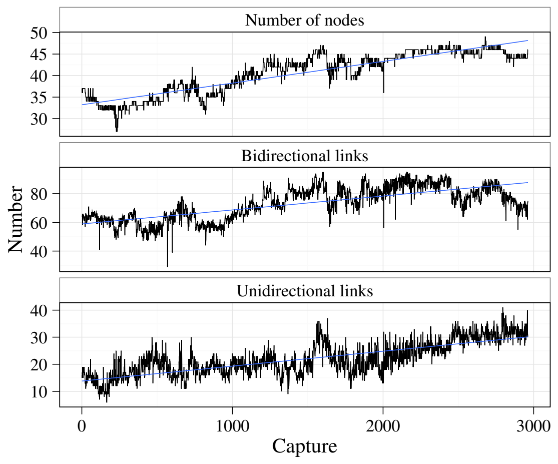


Figure 2.4: Number of nodes, bidirectional and unidirectional links in each capture.

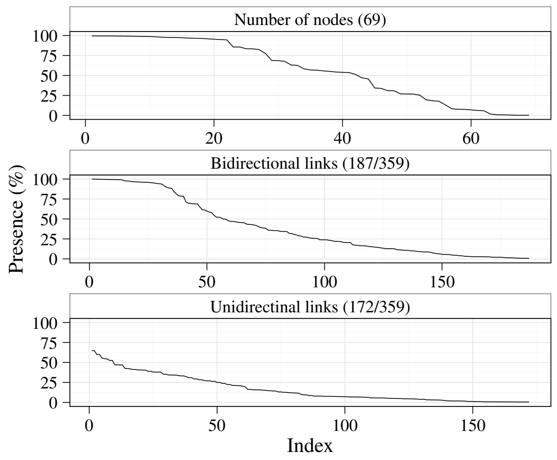


Figure 2.5: Nodes and links presence.

[4]http://www.ubnt.com/downloads/datasheets/nanostationm/nsm_ds_web.pdf
[5]http://www.ubnt.com/downloads/datasheets/nanobridgem/nbm_ds_web.pdf
[6]http://www.ubnt.com/downloads/datasheets/rocketmgps/Rocket_M_GPS_Datasheet.pdf

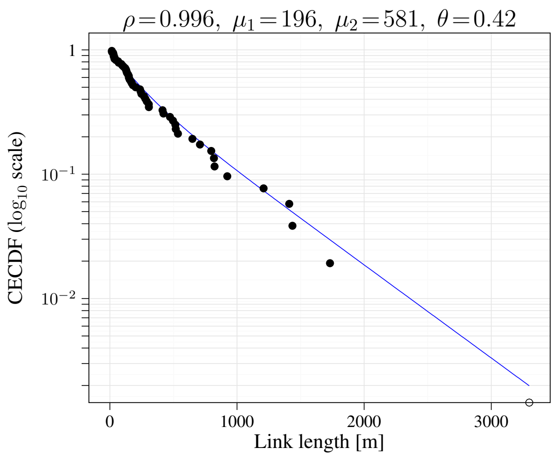


Figure 2.6: Link length distribution.

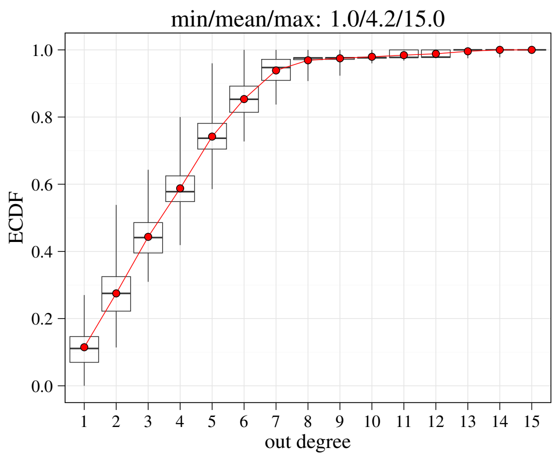


Figure 2.7: Average out degree ECDF.

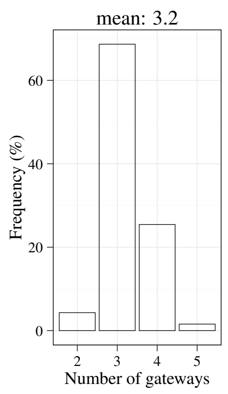


Figure 2.8: Number of gateways.

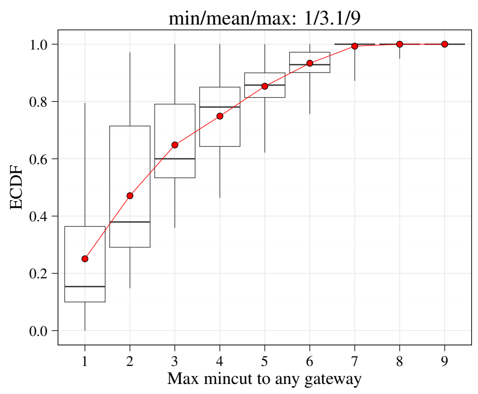


Figure 2.9: Average ECDF of the max-min-cut from a node to any of its gateways.

2.1.3 Methodology

Measurements were obtained using ssh to connect to each QMPSU node and gathering information with basic system commands available in the QMP distribution. This method has the advantage that no changes or additional software had to be installed in the nodes. This is an important point, since being a community network, the users are the owner of

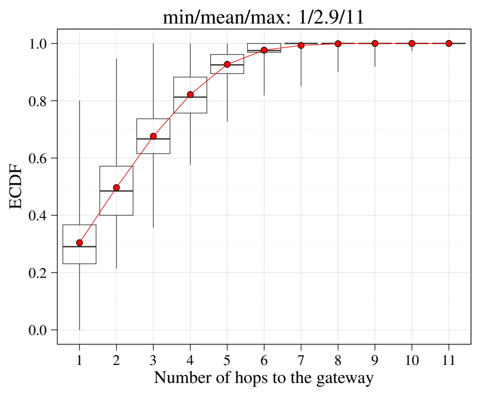


Figure 2.10: Average ECDF of the number of hops to the gateway.

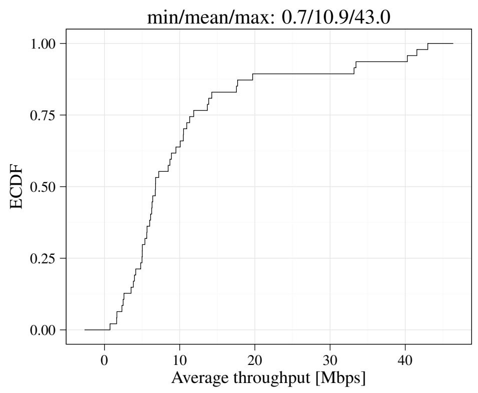


Figure 2.11: ECDF of the average throughput to the gateway.

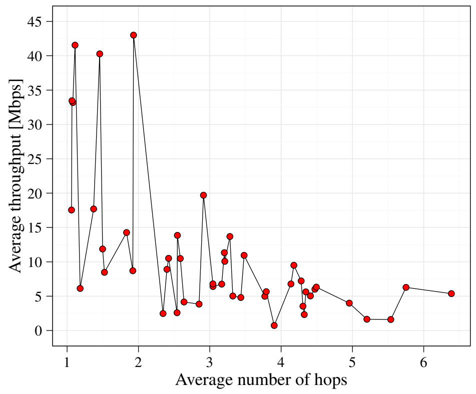


Figure 2.12: Average throughput vs. average number of hops to the gateway.

their nodes, and so, a minimum intrusion was desirable. The data collection was done hourly from December the 29th 2012 to June the 13th 2013. A simple monitoring web page was developed, which is publicly available at [44] (see figure 2.3). The web page allows navigating through the graphs obtained in the captures.

QMPSU is rather dynamic. Several reasons contribute to this fact: Being a community network the growth is essentially unplanned. In Sants, nodes are added by community members using their home roofs, which are often at non optimal locations. This fact produce a high diversity on the quality of the links, making some of them to flip-flop time to time, and even some nodes to be sporadically unreachable. Other reasons of unreachability have been electricity cuts, nodes that have been upgraded, reconfigured, hanged, etc. Additionally, community members some times have re-tuned the radios of their devices, trying to achieve better performance (transmission power, channel and other parameters), thus, changing the characteristics of the links. Furthermore, during the measuring period not only new nodes have been added, but other were removed or changed their position. E.g. testing nodes only used temporarily at UPC, or users that changed residence in Sants.

Characterizing such a dynamic network is challenging. That is why in the following sections we have chosen different statistical representations depending on the parameter under study. For instance, in order to reflect the dispersion of the measured parameters, in some cases we have used standard boxplots (showing the range, median and quartiles).

2.1.4 QMPSU Topology

Figure 2.4 shows the number of nodes and bidirectional and unidirectional links found in each capture. Regarding the links, we have considered those reported by BMX6 (with the command `bmx6 -c show=links`). For bidirectional links, in figure 2.4 we count both links in opposite direction as a single link. Figure 2.5 shows the nodes and links presence. We define presence as the percentage of times a given node or link is observed over the captures.

Figure 2.4 shows that QMPSU is growing. The solid line in the figure shows a linear regression fit, which increases from around 33 to 48 nodes. On the average 40.6 nodes have been found, 21.4 in Sants and 19.2 in UPC. Even if UPC and Sants have a similar number of nodes, at UPC the nodes are distributed in the Campus, which covers a rather smaller area than Sants (see figure 2.1). Overall, 69 different nodes where detected. From those, figure 2.5 shows that only 42 had a presence larger than 50%. Indeed, figure 2.5 shows that only 27 nodes were alive almost all measuring period, having a presence higher than 80%. The reason for this are temporary installations (that got removed after a while of testing) and poorly maintained installations with weak wireless link connections to other nodes of the mesh.

Figure 2.6 shows the link length Complementary Empirical Cumulative Distribution Function (CECDF). We have found that the link length distribution can be fitted by a mixture of 2 exponentials (solid line in figure 2.6). The distribution is fitted with correlation coefficient of $\rho = 0.996$. This is in line with the results reported for Guifi.net in [42, 45]. Let L be the complementary CDF of the link length, X, then: $L(x|\mu_1, \mu_2, \theta) = P(X > x) = \theta\,\mathrm{e}^{-x/\mu_1} + (1-\theta)\,\mathrm{e}^{-x/\mu_2}$ This result shows that links can be grouped in two sets: 42% of shorter links with mean $\mu_1 = 196$ m and 58% of longer distance links with mean $\mu_2 = 581$ m. Thus, an overall mean link length of 419.3 m.

Figure 2.7 shows the out degree distribution. To derive this figure we have proceeded as follows: We have first built the graph of each capture and its out degree ECDF. Note that the graphs may change in every capture for the reasons formerly described. Then, for each out degree we have taken the average of its ECDF value obtained over all captures. In order to show the variability of the out degree, in figure 2.7 we have added a boxplot of the out degree ECDF values obtained over the captures.

Figure 2.7 shows that, on the average, around 90% of the nodes have more than 1 link, and around 40% of the nodes have at least 4 links, with an overall average degree of 4.2. This is in contrast with Guifi.net[42], were the average is around 2, and only 20% of the nodes have degree higher than 3. This fact can be explained by the higher number of links that are automatically discovered and established by the nodes in the ad-hoc configuration used in QMPSU, than the static links manually configured in Guifi.net. We note that no standard distribution (including a power law) has been found to fit the average out degree. Therefore, it cannot be stated that the scale-free pattern found in the Internet applies to QMPSU. Nevertheless, the size of QMPSU (around 45 nodes) is too small to do an accurate topology fitting.

```
# iw dev wlan0 station dump
Station dc:9f:db:26:6a:40
 inactive time: 10 ms
 rx bytes:   3568971019
 rx packets: 135222757
 tx bytes:   3147225400
 tx packets: 167098650
 tx retries: 457952780
 tx failed:  104548
 signal:     -72 dBm
 signal avg: -73 dBm
 tx bitrate: 162.0 Mbps MCS 12 40Mhz
 rx bitrate: 120.0 Mpbs MCS 11 40Mhz
Station dc:9f:db:08:8d:a9
 ...
```

Figure 2.13: Output of the command `iw dump`.

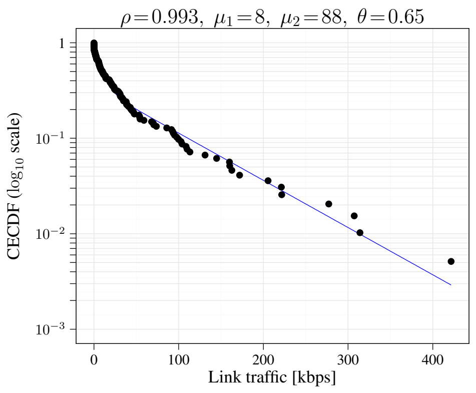


Figure 2.14: Link traffic CECDF.

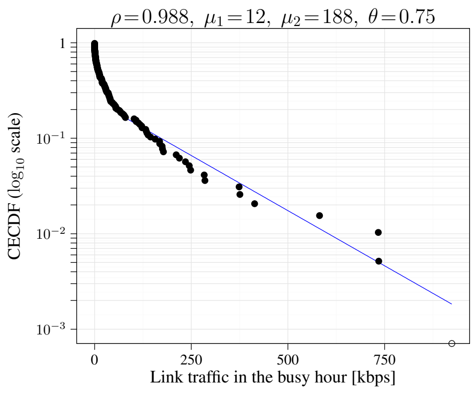


Figure 2.15: Link traffic in the busy hour CECDF.

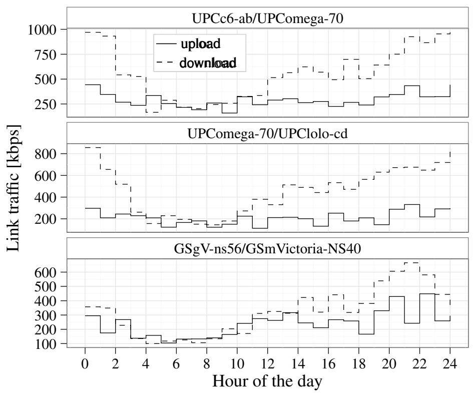


Figure 2.16: Traffic in the 3 busiest links.

2.1.5 Internet access

Internet access is provided by some nodes of QMPSU having links with other nodes of Guifi.net. These nodes disseminate a default route, and we shall refer to them as *gateways*. The number of gateways has been variable during the measuring period: there were found between 2 and 5 gateways, 3.3 on the average. Figure 2.8 shows the frequency plot of the number of gateways. BMX6 estimates a metric to each gateway, and chooses the best one.

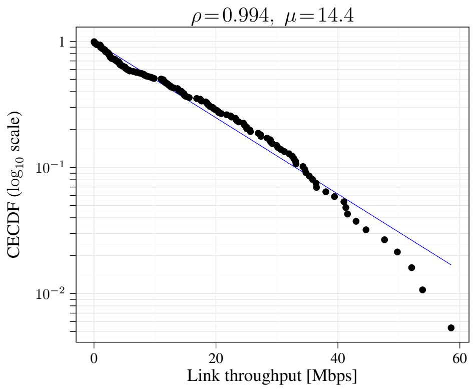


Figure 2.17: Throughput CECDF.

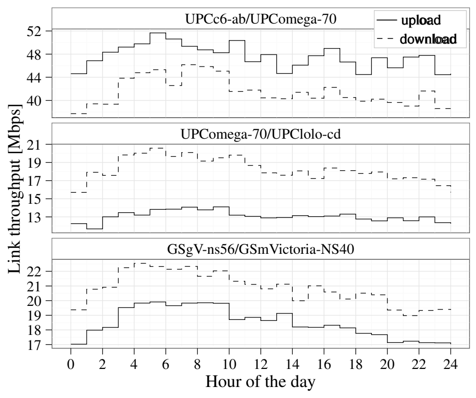


Figure 2.18: Throughput in the 3 busiest links.

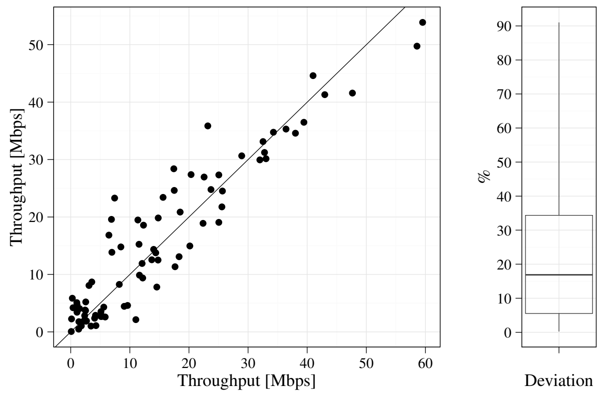


Figure 2.19: Throughput asymmetry.

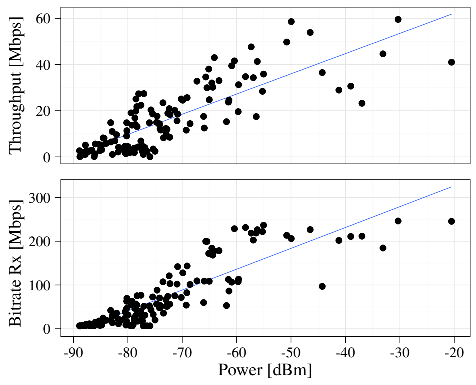


Figure 2.20: Throughput (top) and average bitrate of received packets (bottom) vs signal power.

Note that when a gateway becomes unreachable by a node, it will stop receiving its default route announcements, and BMX6 will switch to another one. Thus, in order for a node to be disconnected from the Internet, is necessary a failure in all links that allows the node to reach any of the gateways. Therefore, as an estimation of robustness of the Internet availability we define the *max-min-cut to any gateway* as the maximum min-cut between a node and any gateway of the network. Recall that the min-cut between two nodes s and t is the minimum number of edges such that, after removal, there is not directed path

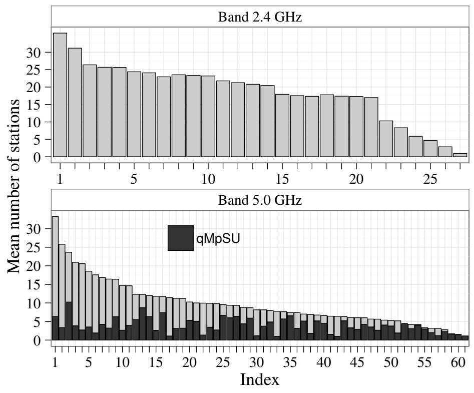


Figure 2.21: Number of stations per scan.

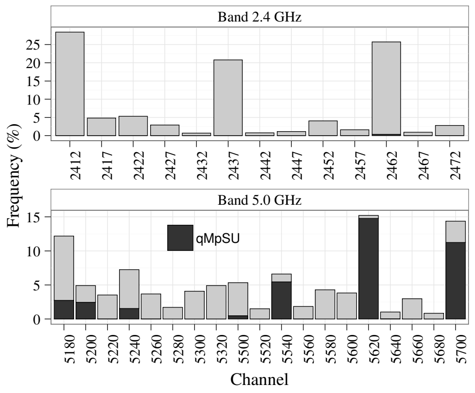


Figure 2.22: Channel occupancy.

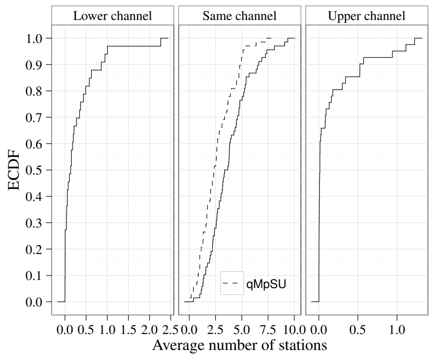


Figure 2.23: ECDF of the number of interfering stations.

from s to t.[7] Figure 2.9 shows the average ECDF of the max-min-cuts obtained in each capture. The distribution has been obtained following the same procedure as in figure 2.7, i.e. taking the average of the max-min-cut ECDF obtained for each graph, and adding a boxplot in order to show the dispersion of each value. Figure 2.9 shows that around 75% of the nodes have a max-min-cut of at least 2 links to a gateway. We note that no standard distribution have been found to fit the average max-min-cut. Figure 2.8 shows that the dispersion of the max-min-cut equal to 2 is rather high: the interquartile of the ECDF values for this point is around 0.42. This is motivated by the fact that one of the nodes at UPC (called UPC-H-22, see figure 2.1) had a wireless link not very stable with the Sants. The failure of this link decreases the max-min-cut of many nodes.

Figure 2.10 depicts the average ECDF of the number of hops to the gateway of each node. This has been derived from the routing tables, which were also recorded in every capture. The plot has been obtained similarly to figures 2.7 and 2.9, i.e. taking the average of the ECDF obtained for each capture, and adding a boxplot in order to show the dispersion. Figure 2.10 shows that around 67% of the nodes have 3 or less hops to the gateway, with an average of 2.9 hops. We can see from the boxplots of the figure that the ECDF measured over the captures does not show strong deviations.

We have estimated links throughput using the TCP_STREAM test of netperf [46]. As before, measurements were taken hourly. The command was run from every node to its gateway. In order to limit the disturbance to the users we tried to reduce the test to the minimum time. After some trials, we observed that running netperf tests of only 3 seconds yield a good estimation. The throughput of every wireless link was also computed (link throughputs are discussed in section 2.1.7). For the link measurements IPv6 link local addresses were used, thus, assuring that no other links would be used. To avoid interferences, throughputs tests were done in serial (only one test at a time). Figure 2.11 depicts the ECDF of the average throughput of the nodes to their gateway measured over all captures. The figure shows that the throughputs are rather high, with an average of 10.9 Mbps. This is due to the high performance that can be achieved with Multiple input, multiple output (MIMO) 802.11an cards of most equipment. Finally, figure 2.12 shows the average throughput vs. average number of hops to the gateway. As expected, throughput tend to reduce as the number of hops increases. However, the line is rather irregular, due to the diversity of the links.

[7] There are efficient algorithms to compute the s-t min-cut available in common numerical tools.

2.1.6 QMPSU Usage

We have gathered the usage of the network using the linux `iw` command [47] (see figure 2.13). Recall that captures were done hourly. Thus, taking the difference between the transmitted bytes counter (`tx bytes` in figure 2.13) of two consecutive runs of `iw`, it is possible to estimate the average traffic sent each hour in every link.

Measurements where done using directional links, i.e. traffic sent in opposite directions between the same nodes is counted as two different links. Only wireless links, and having a presence higher than 25% (in order to avoid outliers) where considered in the statistics: 196 links in total.

Figure 2.14 shows the CECDF of the average traffic sent in each of these links. Interestingly, it was found that the traffic is well fitted by a mixture of 2 exponentials (solid line in figure 2.14): 65% with mean $\mu_1 = 8$ kbps and 35% with mean $\mu_2 = 88$ kbps (overall mean of 36 kbps). An explanation of this result is the presence of two groups of links: Those where most of the traffic belongs to a single user, and *backbone links* carrying the aggregate traffic from a number of users.

We found that the hour having the highest average traffic (*busy hour*) was between 22h and 23h. Figure 2.15 shows the CECDF of the average traffic sent in the busy hour. The figure shows that a mixture of 2 exponentials still gives a good fit. The overall average is now 56 kbps, almost 55% higher than before. Figure 2.16 shows the average traffic in both directions of the three busiest links over each hour of the day.

2.1.7 QMPSU Wireless Links

In this section we try to characterize the wireless links of the network. We start by studying their throughput, measured using netperf as described in section 2.1.5. In order to avoid outliers, only those of section 2.1.6 are considered. From the 196 links, the throughput was able to be measured in 169 (86%). Failures were due to unidirectional links (were netperf cannot run the test), or too weak links for netperf to succeed. Figure 2.17 shows the CECDF of the throughput of the links were netperf succeeded. The figure shows that the link throughput can be fitted with an exponential distribution with mean 14.4 Mbps. In order to see the variability of the throughput, figure 2.18 shows the throughput averages in both directions of the three busiest links (same links as in figure 2.16) over each hour of the day. Comparing figures 2.18 and 2.16 we observe that the throughput is only slightly affected by the traffic in the links. Additionally, measurements on each direction of the links are identified with the same solid or dashed lines in figures 2.15 and 2.16. Comparing the figures it is apparent that the asymmetry of the throughputs measured in both directions it is not due to the asymmetry of the users' traffic. For instance, around 5am, when the user traffic is the lowest and equal in both directions, the asymmetry of the links throughputs observed in figure 2.15 remains the same. We thus conclude that this asymmetry must be links characteristics, as level of interferences present at each end, or different transmission powers.

In order to measure the links' asymmetry, figure 2.19 depicts the throughput measured in each direction. A boxplot of the absolute value of the deviation over the mean is also depicted on the right. The figure shows that around 25% of the links have a deviation higher than 30%. Thus, we can conclude that the symmetry of the links, an assumption often used in the literature of in wireless mesh networks, is not very realistic.

Figure 2.20 shows the average throughput of each link (top of the figure) versus the average signal power of the received packets (measured with iw dump). The figure also show the average bitrate reported by iw dump for unicast received packets (bottom of the figure). We have assumed that unicast transmissions correspond to packets with bitrates higher than the lowest basic rate (6 Mbps in the 5 GHz band). As expected, the figure shows the clear dependency of both measures with respect of the signal power.

Finally, we have estimated the interference level by doing a scan in every node at each hourly capture. All nodes were configured in ad-hoc mode in the 5GHz band. However, some radios were dual band, and reported scans in both 2.4 and 5 GHz. We discriminated QMPSU stations because they share the same BSSID address, which is reported in the scans. Figure 2.21 shows the average number of stations detected in every scan of the nodes, in decreasing order. The figure shows the stations detected in both bands. The average of QMPSU stations is shown in darker color. Figure 2.21 shows that the 2.4 GHz band is much more crowded: only 3 non-overlapping channels of 20 MHz are available, while a much higher number of stations are detected. Figure 2.22 shows the number of stations detected on every channel. As expected, it can be observed as most of the stations detected in the 2.4 GHz band use the recommended 3 non-overlapping channels. In the 5 GHz it can be observed the channel where most of the QMPSU stations are located. Note that it is not possible to have high frequency diversity in QMPSU since most of the stations have only a single radio. Even if channel assignment is a key issue, it was fixed manually by QMPSU users based on trial and error. Figure 2.23 shows the ECDF of the average number of stations detected in the same and adjacent channels used by QMPSU nodes (5 GHz band). ECDF of detected QMPSU nodes is also reported in dashed lines. All QMPSU nodes were configured to use channel bonding (i.e. combining two adjacent 20 MHz channels). Thus, they can potentially interfere with stations located in adjacent channels. Figure 2.23 confirm the low number of stations belonging to external networks interfering with QMPSU nodes: around 90% of nodes detect less than 1 station, on the average, from external networks in each of the 3 channels of interest. Looking at the stations in the same channel, around 50% detect more than 3.5, from which more than 2.5 are QMPSU.

2.1.8 Related Work

A lot of research has been done in recent years about wireless mesh networks, including design aspects (routing, scalability, security [1, 21, 48–56]), deployment (urban, rural, centrally-, individually-, or un-planned [57–60]), measurements and analysis (topologies, performance, usage patterns, evolution, mobility [24, 41, 42, 57–67]), as well as surveys of prior work and related aspects [26, 68–71].

Recent studies of Guifi net include various aspects of network and link characteristics [24, 41, 42], power laws [41], usage patterns, social participation [19, 26], and evolution over the last 10 years [24]. Like in our work, links are planned individually (decentralized) but the average link length in Guifi.net is (due to its deployment also in rural environments) typically more than 10 times longer than in the QMPSU cloud. Unlike in our system, nodes in Guifi.net are clustered into zones and in fact the network presented in this work is one of these clouds. The studies distinguish between the Guifi core network (or backbone network) which includes only non-leaf nodes and the base network (which including all nodes).

2.1.8.1 Roofnet

The Roofnet testbed presented by Bicket and Aguayo [57, 66] in 2005 and 2006 provides one of the first works on characterizing a real-life mesh network. Motivated by the need to understand the performance and topological consequences that result from a rather unconstrained participation of users and unplanned deployment of nodes, the network under study consist of 37 802.11b-based, single-radio nodes with omni-directional antennas that have been set up by individuals on rooftop and indoor locations in the urban environment of Cambridge/Massachusetts. A proactive, link-state, source-routing protocol called Srcr, integrated in each node's Linux/open-source operating system, is responsible for routing the traffic in the flat and fully meshed network topology. Further analysis provides in-depth measurements and simulation results on the link and end-to-end performance and about the topological characteristics of this network.

2.1.8.2 TFA

Other pioneering work is given by the technology-for-all (TFA) mesh project presented by Camp et al in [58, 67] where the authors analyze the performance of a single-radio, 802.11b based 18-nodes mesh network serving some 4000 users in a dense urban deployment in Houston TX. In contrast to Roofnet, the deployment in this work follows a measurement-driven approach with the objective to optimize the overall performance of the employed: a two-tier system architecture which distinguishes between the access and the backhaul part of the network.

2.1.8.3 MadMesh

In contrast to the open, flat, and decentralized networking approaches given above, MadMesh is a WMN planned deployment using proprietary technology from CISCO in 2007. MadMesh is a commercial-grade WMN providing Internet access to residential customers and small business, in Madison, Wisconsin. MadMesh is annualized by Brik et al. in [60]. The experimental study is based on 8 months of data collection using SNMP logs. The authors perform a wide analysis that cover topological aspects and robustness, user activity and radio channel characterization. Additionally, the authors study the feasibility and gain of introducing Network Coding.

2.1.8.4 GoogleWifi

Afansasyev [63] about the Google network in Mountain View, California present urban deployments and performance properties of large scale (several hundreds of nodes) mesh networks based on proprietary and centrally managed node systems. In contrast, the system we present is fully open source (GPL licensed) and can be deployed, managed, and maintained completely decentralized. The networks studied by both works are up to 10 times larger than the network studied in this work. While the topology in their networks is organized in a tree-like structure where each mesh node is bound to a parent node upstream to the few GW nodes where all out and incoming traffic aggregates, the topology of our network is managed in a flat (full mesh) structure where each node may become a GW node and/or a relay towards any other node. While the main focus of [63] is about usage patterns, both works present interesting and comparable performance characteristics about the links and end-to-end routes for their backbone networks.

2.1.8.5 Meraki

In [61], LaCurts present measurement and analysis of 110 real-world 802.11 mesh networks with an average size of 13 nodes (total of 1407 APs). The focus of this paper is on link-level measurements, investigating SNR versus bit-rate correlation, benefits of opportunistic routing, and the impact of hidden stations. In contrast to our network, only 2.4GHz 802.11bgn standard has been used (QMPUS also uses 5GHz 11an) for mostly indoor deployments and few measurements are given about end-to-end performance in the studied networks.

2.1.9 Comparison

In this section we compare the experimental results obtained for QMPSU, with those obtained for the networks listed in section 2.1.8. We summarize our comparison by means of table 2.1. Note that table 2.1 is built such that columns correspond to networks under comparison, and rows to measured parameters. The columns are ordered left to right in descending chronologically order of the research papers' publication date, used to derived the experimental results. Those parameters not provided in the research papers are left blank. The parameters in table 2.1 have been grouped into 5 categories:

- General characteristics: References used to derived the measurements; their year of publications; usage type of the network; environment (rural/urban); square area; type of deployment (planned/unplanned); and methodology used to collect data.
- System characteristics: Hardware used in the network; Operating System (OS) and license; type of MAC and antennas; routing protocol.
- Topology characteristics: Network structure (flat, tree, etc); number of nodes; number of edges; out-degree.
- Link characteristics: Length of the links; throughput.
- End-to-end performance characteristics: number of gateways; number of hops to the gateways; download throughput from the gateways.

Recall that Guifi.net is organized in zones (see [41, 42] for details). The results given in table 2.1 corresponds to Catalonia zone. Additionally, like in [42], for the topology characteristics of Guifi.net given in table 2.1 we distinguish between the *core* (or backbone) and *base* networks. The *core* is obtained by removing all nodes with degree 1 from the *base* network (which is the one including all nodes). This distinction is motivated by the way Guifi.net is deployed: A relatively small number of nodes, called *super-nodes*, located in strategic points, having a high number of wireless links connecting to single end customers or other super-nodes. For instance, the node having the maximum number of links (the node with degree equal to 476, as shown in table 2.1), is located in a hill (composed of several sectorial antennas) and provides access to Guifi.net to the users in the Village of Tona and its surroundings.

Table 2.1 shows that Catalonia zone (which has only around 46% of Guifi.net nodes) has $10,625$ nodes. This is, with difference, the largest network under consideration. However, table 2.1 shows that it is weakly connected, if compared with the other networks. For instance, the 0.75 quantile is 1 for Guifi.net base network, and even for the core is only 2, while it is between 4 and 6 for the other networks. This fact highlights one fundamental advantage of WMN: the robustness provided by the links that are automatically created, providing redundant paths between the nodes.

Regarding QMPSU, despite its decentralized and individually management and deployment, our measurements show promising performance characteristics. For example table 2.1 shows that 50% of the links allow for more than 10 Mbps throughput. This is in contrast with the 0.7 Mbps obtained in Roofnet. One reason of this improvement might be that link distances are rather short (50% of the links are less then 150 meters long and only 25% are more than 300 meters long). However, QMPSU is deployed in a dense urban area and many links does not have a line of sight free of obstacles. Taking into account that Roofnet has a similar number of nodes, and it is deployed in a similar area (in km^2), we conclude that the high throughput of QMPSU can be attributed to the improvements introduced by 802.11an over 802.11b, used in Roofnet. From the TP versus signal strength measurements we can observe similar characteristics as already identified on the TFA mesh ([58, 67]), that link TP increases almost linearly between it's minimum and maximum rates. However, with the 802.11an technology used in QMPSU (compared to 802.11b in TFA) an average TCP TP of more than 20Mbps can be achieved with the same signal power (-67dBm) that allowed only up to 5MBps UDP TP on the TFA network.

In terms of end-to-end performance, QMPSU deployment provides 50% of the nodes with more than 7 Mbps download via the nearest gateway. This is a small reduction over the 10 Mbps median link throughput, taking into account that 50% of the nodes have 2 or more hops to the gateway(compared with the commercial MadMesh network ([60] where 50% of the nodes have more than 4 hops to the nearest GW).

Finally, it is interesting to note that the routing protocol, one of the key components of a WMN, is different in all networks under study. This demonstrates the fact that there is not yet an optimal solution for this problem.

2.1.10 Conclusions

In this Section 2.1 we presented an experimental evaluation of QMPSU, a wireless community network deployed at *Universitat Politècnica de Catalunya* (UPC) and Sants, a quarter of Barcelona, Spain. QMPSU is rather dynamic due to many reasons, e.g.: its community nature in an urban area; it is a growing network; there is a high diversity of the quality of wireless links; the mesh nature of the network.

Characterizing such a dynamic network is challenging. To do so we have performed an extensive statistical study of the main parameters. These include topological properties, Internet access, usage of the network and characterization of the radio links.

Table 2.1: Comparison of related networks

	QMPSU	**Guifi.net**	**Meraki**	**MadMesh**	**Google Wifi**	**TFA**	**Roofnet**
General characteristics							
References	[2, 3]	[24, 41, 42]	[61]	[60]	[62, 63]	[58, 67]	[57, 66]
Published	2013	2012,2013	2010	2008	2008	2006,2008	2004,2005
Usage	community	community	real clients	commercial	non-comm.	non-comm.	testbed
Environment	urban	urban & rural	indoor	urban	urban	urban	urban
Area [km^2]	6	15,000		26	32	3	6
Deployment	unplanned	unplanned	planned	planned	planned	unplanned	unplanned
Data sources	sys. tools, probes	SNMP, publ. CNML DB	probes	sys.-portal, SNMP, sys. tools, probes	Tropos mgmt portal	sys. tools, probes	sys. tools, simulations, probes
System characteristics							
Hardware	open	open	Meraki	Cisco 1510	Tropos MetroMesh	VIA EPIA x86 1GHz,	small PC
OS / license	GPL, qMp [31], Linux openWRT	RouterOS (proprietary) or Linux (GPL)	Meraki	proprietary Cisco OS	proprietary Tropos OS	Linux	Linux
MAC / antennas	802.11an, / sect. & dir.	802.11abgn / sect. & dir.	802.11bgn	802.11a / 11dBi sect.	802.11b/g	802.11b / 15dBi omni.	802.11b / 8-12dBi omni.
Routing	BMX6 [1]	BGP, IGP, OSPF, OLSR, BMX6, static,..	Meraki	ease (SNR + ETX)	proprietary Tropos	AODV	Srcr (link-state source-routing, ETT)
Topology characteristics							
Structure	flat (full mesh)	clustered and grouped in zones	flat (full mesh)	trees	trees	flat (full mesh)	flat (full mesh)
Nodes	40.6 (avg)	base: 10,625 core: 735	12.8 (1407 APs / 110 networks)	250	500	18	37
Edges	62.5 (avg)	base: 10,949 core: 1,059					344
Degree .5/.75/1*	4/6/15	base: 1/1/476 core: 1/2/30		3/4/10	4/6/12		
Link characteristics							
Length [km] .5/.75/1*	0.15/0.3/3.3	0.59/1.36/34.6					
Throug. [Mbps] .5/.75/1*	11.3/22.9/59.5 (TCP)		throug. vs. SNR			throug. vs. SNR	0.4/0.7/4
End-to-end performance characteristics							
Number of gateways	avg. 3.2				3 + 75 (nodes with direct link to GW)		4
GW distance [hops] .5/.75/1*	2/4/11			4/5/8	1/2/5		2/3/5
GW throug. [Mbps] .5/.75/1*	6.8/11.6/43 (TCP)			0.6/1/1.2	limited to 1 Mbps		

* We use the notation .5/.75/1 to refer to quantiles. Note that quantile = 1 corresponds to the maximum measured value.

We have found simple distributions that fit some of these parameters. For instance, the network is not scale-free, the link length and traffic is fitted by a mixture of two exponentials, and the average throughput of the links is exponentially distributed. Regarding radio links, we have observed a non negligible asymmetry.

Our results show that the network is rather well connected and adaptive. Thus, demonstrating the advantages of a wireless mesh network. Furthermore, even if the network is deployed in an urban area with an average link length of around 500 m, an average link throughput of around 14 Mbps was obtained. This high performance can be attributed to the 802.11an devices used in the network.

We have also compared QMPSU with Guifi.net and other experimental WMN studies found in the literature. Most Guifi.net's infrastructure consists of wireless links manually set up, and use OSPF and BGP routing protocols. Our comparison shows that QMPSU results much more connected, and thus resilient than the rest of Guifi.net, thus, demonstrating the benefits of a WMN.

Our comparison with other WMN highlights the wide number of scenarios where WMN are being deployed. These include the pioneer research testbed of Roofnet, in 2004; commercial WMN used to provide Internet access, as Madmesh; companies that offer WMN solutions to small organizations, as Meraki; to QMPSU used in a wireless community network. We observe a significant performance improvement over time. However, all these networks are rather small (up to few hundreds of nodes), and all use different routing protocols. We conclude that the optimum routing protocol, and to what extend it can scale, are still open issues in WMN.

2.2 Experimental Evaluation of BMX6 Routing Metrics

In this section we experimentally evaluate the path-metric performance of the mesh routing protocol BMX6 [1, 72] which is the routing protocol used in the QMPSU production community network described in [2]. The results presented in this chapter are included in the paper [5].

We exploit the full access to the wireless routers of QMPSU to achieve two goals: The first is comparing the experimental measures of throughput on multi-hop paths that we perform on the network with the expected capacity estimation on the same paths derived using the well known conflict-graph model introduced in [73]. Our experiments show that even with an accurate knowledge of the network parameters the conflict-graph model introduces an overestimation of the available capacity. We discuss the possible causes for this error and propose a correction. The second goal is to test the capability of the BMX6 routing protocol used in the WCN to choose the path that can guarantee the highest throughput. We show that BMX6 is able to choose the best path in the large majority of the cases, which is a key feature for any routing protocol, enabled by the right combination of the protocol internals and the metric used for link and path quality estimation.

2.2.1 Related Work

The experimental evaluation of production-state wireless mesh network has been done only in a small number of papers in literature compared to the enormous amount of works that use simulations, a review of the experimental research papers can be found in [68]. Some of the works use a similar approach to this work for the extraction of real measurement data [57, 58], but most of the networks analyzed are single-channel networks using omnidirectional antennas. In our case the use of multi-channel and directional antennas makes the analysis more challenging, since we neither assume interference between every couple of neighbor links (like in [57]) nor its absence and thus have to rely on a complex model for capacity estimation [73]. Moreover, to our best knowledge, this is the first empirical evaluation of a real IEEE 802.11an-based community network. Other empirical works use controlled scenarios [74] to compare routing metrics (like Expected Transmission Count (ETX) [75], Expected Transmission Time (ETT) [76], Direct Air Time (DAT) [77] metrics). Finally, some other works use off-line evaluation of available data to estimate various network properties [42], including routing performance [78] but can not really be compared to on-field experimentation.

2.2.2 Theoretical Path Capacity

In order to evaluate the performance of the routing protocols we need to estimate the capacity of selected paths. Accurate capacity estimation in wireless is challenging, and the *Protocol Model* proposed in [79] is typically used in 802.11 networks. With this model any couple of nodes using the same channel and in interference range can not simultaneously transmit. The protocol Model was used to define the concept of *conflict graph* in [73] to estimate the capacity of wireless networks as an LP optimization problem. Afterwards, the conflict graph has been extensively used in the literature to estimate the capacity of wireless networks in resource optimization problems, e.g channel allocation [80–82]. In the following we will recall the concept of conflict graphs and will use it not to formulate an optimization problem but instead to estimate the capacity of a multi-hop path once the capacity of the single hops has been measured.

Let $G(V, E)$ be a graph in which the set of vertices V corresponds to the set of nodes in the network and the set of edges E corresponds to the set of links. Let $N = |V|$ the number of nodes of the network denoted by $n_i,\ 1 \leq i \leq N$. Take a generic path $P = \{n_1, \cdots n_d\}$ as the ordered set of nodes chosen by the routing protocol to deliver a packet from the source node n_1 to the destination node n_d. Let l_i be the link used to connect each node n_i to the next node n_{i+1} in P, c_i the capacity (in bit per second) of link l_i and $L = \{l_1, \cdots l_{d-1}\}$ the set of links used in P.

Let $G_c(E, C)$ be the conflict graph of G. In G_c vertices correspond to links in G, between two vertices there is an edge if the two links interfere and thus can not transmit simultaneously. Let $G_c(P)$ be the induced sub-graph of G_c, where the vertices are the links L in P, and the edges are the same as those that links L have in G_c.

Now, let $N_i(P),\ i = 1, \cdots d-1$ be the sets formed by each vertex of $G_c(P)$ and its neighbors. Consider two links $l_i, l_j \in N_i(P)$ that require a time $\frac{1}{c_i}$ and $\frac{1}{c_j}$ respectively to send one bit on the link. Note that each set $N_i(P)$ is formed by links that need to schedule their transmissions in different time intervals, so if a bit has to travel over link c_i and then c_j it will require a total time of $\tau = \frac{1}{c_i} + \frac{1}{c_j}$. The capacity of the path l_i, l_j is thus given by $\frac{1}{\tau}$. Generalizing, for each sub-path formed by links belonging to $N_i(P)$ the expected capacity is:

$$C_i(P) = \frac{1}{\sum_{l_j \in N_i(P)} \frac{1}{c_j}},\ i = 1, \cdots d-1. \tag{2.1}$$

The *theoretical capacity of the path*, $C_t(P)$, is given by the most restrictive sub-path, thus:

$$C_t(P) = \frac{1}{t_b(P)} \tag{2.2}$$

where

$$t_b(P) = \max_i \sum_{l_j \in N_i(P)} \frac{1}{c_j},\ i = 1, \cdots d-1 \tag{2.3}$$

Note that $N_{i=b}(P)$ is the set of links of the path P that minimize (2.1). Thus, we shall call $t_b(P)$ the *bottleneck airtime* of the path.

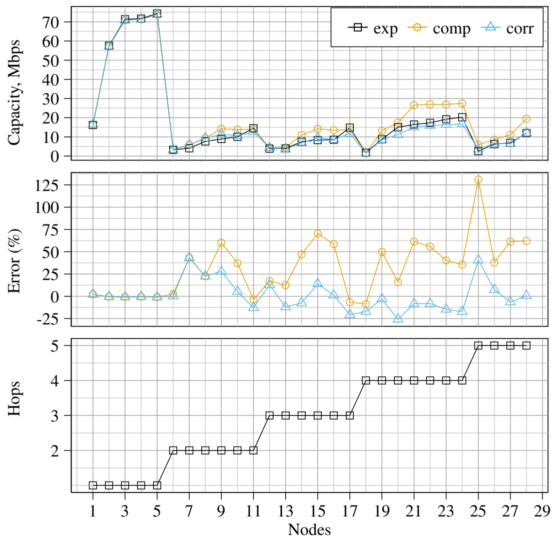


Figure 2.24: Experimental $C_e(P)$ (exp), theoretical $C_t(P)$ (comp), and fitted $C_f(P)$ (corr) capacities (top); relative error of theoretical $e_t(P)$ and fitted $e_f(P)$ capacities (middle); and number of hops (bottom).

2.2.2.1 Validation

In order to validate equation (2.2) we have proceeded as follows: We have experimentally estimated the capacity c_i of each link by measuring the throughput with netperf. The same has been done to estimate the capacity of the path to the gateway for each node. We shall refer to these measurements as the *experimental capacities*, and denote them as $C_e(P)$.

In order to compute the conflict graph $G_c(P)$ we proceeded as follows. First we defined the graph G and we assigned to each link a value for the capacity c_i that equals the measured one. Then we generated G_c, using as vertices the node links, and adding edges between neighbor links using the same channel. Thus, we assume that interference only occurs between WiFi neighbor interfaces using the same channel. For each node we computed the path used to reach the gateway by means of the routing tables and, on that path, we computed the theoretical capacity using (2.2). We shall refer as *theoretical computed capacity*, $C_t(P)$, to the capacity obtained by (2.2).

Figure 2.24 (top) shows the mean experimental ($C_e(P)$), and theoretical ($C_t(P)$) capacities to the gateway of each node. These are measured only for the most frequent route of each node.The means were obtained averaging more than 100 points in all cases. The resulting confidence intervals were rather small, less than 5% in most cases. In the same figure is shown a third curve ($C_f(P)$), which is a better estimation than $C_t(P)$ and will be explained in next section. Figure 2.24, middle, shows the relative error of $C_t(P)$ and $C_f(P)$ capacities with respect to the experimental ones, $C_e(P)$, computed as:

$$e_i(P) = \frac{C_i(P) - C_e(P)}{C_e(P)}, \; i = \{t, \, f\}. \tag{2.4}$$

Finally, Figure 2.24 bottom shows the number of hops of each route. Note that paths are sorted in increasing order of hops, and capacity.

Figure 2.24 shows that the theoretical capacity overestimates significantly the experimental one. Indeed, the absolute relative error has an average around 34%. This result concerns the usage of the conflict graph as an accurate tool to estimate the capacity of a wireless network. In the following section we discuss this mismatch and propose a better fit of equation (2.2) to the experimental path capacity.

2.2.2.2 Path Capacity correction

Given the results of 2.2.2.1, we can say that the definition we use for the conflict graphs leads to an overestimation of the available capacity. To build the correct conflict graph we need to know all the links that interfere with each other, and can not transmit simultaneously. In G_c we set an edge between two links only when the two links are in the same channel, and are separated by no more than one hop, we say that this approach describes only "*direct interference*". This assumption is reasonable considering that the majority of the radios use directive antennas, but is probably optimistic, since there are a number of factor that produce what we call "*collateral interference*". First we do not consider interference at a higher distance than one-hop, which instead can happen. The number of hops between two nodes depends on the way the radio are configured, and on the decision that the routing protocol takes. Two nodes can be close to each other, but configured with an incompatible MAC layer mode (for instance, both configured to be client of a third node) that prevents them to be direct neighbors. Second, neighbor-channel interference can happen when two radios are placed nearby [83] and even when directive antennas are used [84]. We can not capture this phenomenon with our abstraction so it is reasonable that this contributes to the overestimation of the available capacity.

Since it's impossible to perfectly model a network operating in real conditions with an analytic approach we chose to apply an empirical approach using the experimental data we have.

Thus, we propose to modify equation (2.2) to estimate the collateral interference in the QMPSU network, as:

$$C_e(P) \approx C_f(P) = \frac{1}{t_b(P) + f(P)} \tag{2.5}$$

We shall call *airtime bloat* the term $f(P)$, which represents the increment on the bottleneck airtime induced by the interference that we can not precisely model over path P.

$$f(P) = \theta \sum_{l_j \notin N_{i=b}(P)} \frac{1}{c_j},\ 0 \leq \theta \leq 1 \tag{2.6}$$

In order to estimate θ we used the available experimental data to compute the the mean square relative error of the mean capacities, i.e. by minimizing the cost function[8]:

$$J(\theta) = \sum_P \left(\frac{C_f(P) - C_e(P)}{C_e(P)} \right)^2 \tag{2.7}$$

From which we obtained that the most suitable value to approximate our data set is given by $\theta \approx 0.5$.

Figure 2.24, top, compares the experimental (*exp*) and computed capacities using equation (2.5) (*corr*). Figure 2.24, middle, reports the relative error. It can be observed that

[8]We have used the BFGS algorithm provided by the numerical tool R.

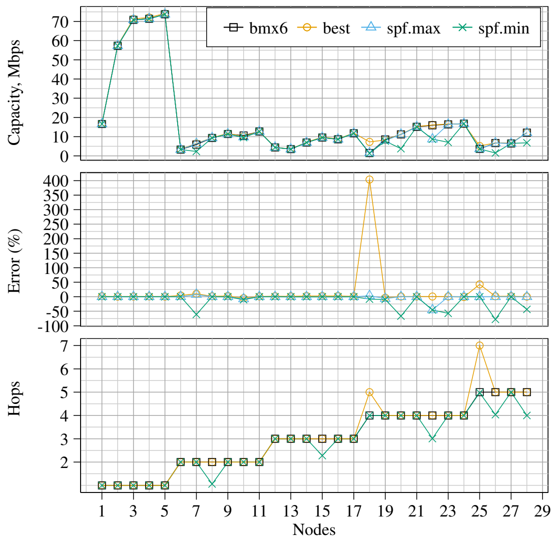


Figure 2.25: Fitted BMX6 (bmx6), best (best), and worst and best SPF paths (spf.min, spf.max) capacities (top); relative error with respect to corrected capacities (middle); and number of hops of fitted, best and SPF paths (bottom).

the capacity estimation is significantly improved. In fact, the absolute relative error has an average around 12%, which is almost 3 times smaller than the 34% error obtained with equation (2.2). Note that the value of θ is a characteristic of the QMPSU network, so it can not be simply re-used in other networks. Nevertheless, giving a reasonable good estimate for QMPSU, as discussed above, equation (2.5) will be used as reference to investigate the performance of BMX6 carried out in next section.

2.2.3 BMX6 Performance

In this section we compare the paths chosen by BMX6 with the best paths (having the highest capacity). For the sake of comparison we also use the paths obtained using the Shortest Path First (SPF) algorithm. Note that SPF correspond to hopcount metric. All capacities shown in this section are computed using the corrected equation (2.5). The best path to the gateway has been computed using the Algorithm 1. Basically, the algorithm performs a recursive search estimating the capacity of each path. In order to avoid a costly exhaustive search, it is first guessed a best path using a weighted SPF, with link airtimes as costs. Then, recursion is performed, stopping over paths that give worst bandwidth than the current best path estimate.

Figure 2.25 compares the capacity of the path chosen by BMX6 (*bmx6*); the best path (*best*); and paths yielding the maximum and minimum capacities using SPF (*spf.max*, and *spf.min*, respectively). Note that the points corresponding to *bmx6* are the same than those marked as *corr* in Figure 2.24. For the same number of hops, there might be different paths, having different capacities. As in the previous section, these capacities are computed averaging over the most frequent paths chosen by BMX6, and the best and SPF paths obtained in the same captures. Figure 2.25, middle, reports the relative error of best and SPF paths with respect to BMX6 (see equation (2.4)). Thus, positive error means better paths than BMX6, and negative error means worst. Finally, Figure 2.25, bottom, shows the number of hops to the gateway for the paths chosen by BMX6, best and SPF.

Algorithm 1: Find the best route (shortest route having the highest path capacity).

```
Data: g: network graph, s: source, d: destination.
Initialization
BestRoute ← WeightedShortestPath(g, from=s, to=d)
BestBw ← PathCapacity(g, BestRoute)
foreach n in Neighbors(g, s) do
    SearchBestRoute(g, s, d, n)
end
SearchBestRoute(g, r, d, c)
begin
    r ← push(r, c)                                        /* append c to route r */
    bw ← PathCapacity(g, r)
    if (bw > BestBw) or (bw = BestBw and length(r) < length(BestRoute)) then
        if c = d then
            BestRoute ← r
            BestBw ← bw
        else
            foreach n in Neighbors(g, c) do
                if n ∉ r then
                    SearchBestRoute(g, r, d, n)
                end
            end
        end
    end
end
```

Figure 2.25 shows that BMX6 Vector Metric behaves indeed very well: In most cases the best paths only give a slightly better capacity than BMX6. Only in 2 cases there exists a significantly better path (with relative increases of 400% and 40%, respectively), but having a larger number of hops. Regarding SPF, it was obtained that for the best choice (spf.max), only in 2 points SPF was slightly better, but less than 10%. While spf.min was always worse or equal than BMX6. Indeed, spf.min yielded 6 points (26% of the paths having more than 1 hop) with a relative reduction higher that 40% than BMX6.

2.2.4 Conclusions

In this Section 2.2, we used experimental evidence to analyze the performance of the BMX6 routing protocol. In particular we focused on the capacity of BMX6 Vector Metric to select the route that can achieve the highest throughput and we verified that the combination of metric and protocol internals used by BMX6 is very efficient in selecting a path that is very close to the optimal one. To achieve this goal we performed experiments on the QMPSU network that showed that the model proposed in [73] with simple assumptions on the interference among links produces an overestimation of the achievable throughput.

2.3 Reflection and Self-determination in a Community Mesh Network

Community Networks (CNs) are IP-based networks designed, built, operated and maintained by communities of individuals that join together and cooperate to satisfy their telecommunication needs. They consist of distributed and decentralized network devices

-linked via wired and wireless links- that interconnect computing systems, services providers, content repositories, etc. exchanging traffic between them. The size of CNs ranges from a few nodes to the tens of thousands. [21, 70]

The main difference between the CNs paradigm and the traditional commercial ISPs is that end users are not mere consumers, but active contributors and stakeholders of the infrastructure[21]. This empowerment comes along with rights and duties: having a voice for decision-making, the obligation by some sort of network compact or agreement [25, 85] and, to a limited extent, the freedom to audit and participate in the control and management of the network resources and infrastructure.

There are, however, several obstacles that significantly limit this freedom. Leaving aside issues in CNs like community organizations, technical and legal aspects of network deployment, etc., this document focuses particularly on the day-by-day monitoring and management of the CN. For such an important task, there is a lack for convenient tools to help the end user evaluating and understanding the state of the CN as a whole (or, at least, the part of the CN around him that plays the most important role in the perceived performance and quality of experience). Despite the existence of many networking tools [86, 87] to inspect particular characteristics of Metropolitan Area Networks (MANs), they are tailored for use cases that significantly differ from those of CNs. Typically, MANs are owned or controlled by a single or few entities. There, persons in charge of the network administration are well-skilled, able to perform complex evaluation tasks and have comprehensive control over all core components. This is not necessarily the case for an average CN user, whose boundaries for a comprehensive control are given by the nodes he owns, or with network skills similar to those of customers of a traditional ISP (this is, somebody able to deal with an intuitive web portal on the home router).

To address this shortcoming, CN members have created tools to visualize their network topology and evaluate link qualities and bandwidth capacities. However, these tools do not provide an integrated solution that combines feedback about the performance of individual links and end to end (e2e) routes and illustrates the chosen path and involved links when traffic is sent towards a given destination. This lack of integration significantly hinders the potential of Routing Protocols (RPs) like BMX6 [38, 72] to adapt to user-specific and application-dependent priorities.

The rest of this section is structured as follows. In Section 2.3.1 we discuss related work and existing solution that address aspects of topology monitoring and route management in CNs. In Section 2.3.2 we present the design of our integrated solution to facilitate the life interaction of CN users with their network. We describe experienced challenges, considered scenarios and experimentation results obtained from integrating our developed solution in a real-life CN in Sections 2.3.3 and 2.3.4. Eventually Section 2.3.5 concludes this work.

2.3.1 Related work

In recent years, the upraising of Community Mesh Networks [21, 70, 88] has perceived an increasing amount of attention and several studies have been published that analyse the characteristics [2, 28, 42, 78] of these networks from an academic perspective. However, apart from the promotion, increased recognition, and general understanding of their functioning and for the demand of such networks, these insights provide little concrete advantage for their standard users that are often faced with (academically wise) trivial but real-life problems such as capacity shortages and variances and a lack of means to detect, understand, and encounter related phenomena.

Various tools exist for CN providing a high-level overview of the network topology as well as detailed information about existing links and nodes. Some of them take the information from a rather static database (Guifi network Map [89], WiND [90], Nodeshot [91]), some retrieve data from the nodes dynamically via Simple Network Management Protocol (SNMP) (Freimap [92]) and others use a combination of both (Nodewatcher [93]). A downside of these tools is the requirements for setting up and maintaining the visualization services so that they can be used, which usually requires quite an advanced knowledge on systems administration and networking.

Another type of mapping tools are decentralized monitoring services, often integrated or shipped with node- or router-system firmwares. They are accessed via the node's web interface, like the case of FreiFunk [37], Gràcia Sense Fils (GSF) [94], qMp [31], Lugro-mesh [95], Commotion [96], etc. These tools mostly rely on the topology information locally available from the RP and optionally enhance the representation with street map data from external services like OpenStreetMap [97]. These tools can show nodes and links that are known to the running protocol but can not assist in any centralized management tasks like node registration, address assignment or representation of planned links. However, they are instantly available to the users and showing a live snapshot of the topology. Anyway, all the existing tools lack means to indicate the relation between a given CN topology (either on a map or as graph) and the network performance for a particular user. Furthermore, they do not provide details on how the RP could be stimulated to improve the network experience at all. There, tools like *traceroute*, *tracepath*, My Trace Route (MTR), etc. are only of partial help because they assume symmetric e2e routes. This is often not the case for routes in CMNs with enough redundancy, where different paths between two nodes may be eligible. In such context, the RP continuously evaluates the best path for transmitting packets. The decision taken is mostly conditioned by the following facts. First, the quality of a link between two nodes can be very asymmetric. For example, different transmission powers can be chosen independently at each device, one of the devices may operate in a noisy electromagnetic environment whereas the other might be isolated, etc. Second, this link quality can change at any time in an unpredictable fashion (the wind may cause antennas misalignment, remote interferences may appear and disappear, etc.). In conclusion, it can not be taken for granted that e2e routes between CMN nodes are symmetric nor invariable in time.

2.3.1.1 Routing metrics for Community Mesh Networks (CMNs)

The issue of routing in distributed system has raised new attention with the emergence of wireless CMNs, as they pose new challenges on this cooperative task because of their intrinsic heterogeneity and the impacts from wireless links characteristics. It has also become clear that, even when assuming a common objective, there is no "one size fits all" solution and there are multiple trade-offs amongst routing metrics [88]. To address these challenges and requirements, various routing metrics (Hop Count (HC), ETX [75], , ETT [76], DAT [77]) have been proposed to overcome the limitations of rather traditional routing approaches and to optimize routing for specific objectives.

Existing MANET protocol implementations have been extended to support new and experimental routing metrics. OLSR [98], originally considering HC as the base metric, which implementation [99] supports to define arbitrary metrics via a plug-in framework, now uses ETX as the default one. The concept of adaptable metrics got extended with OLSRv2 [100] for which the DAT metric is suggested. Babel [101] declares the exact computation of route metrics as out of the scope of its definition and only defines the requirement of being monotonic and isotonic. The babeld [102] implementation uses an additive ETX metric to calculate e2e path costs but makes it easy to define alternative metric functions.

Nevertheless, the RPs above mentioned assume that path metrics are calculated with the same metric by all nodes of a network. Therefore, to ensure a consistent calculation of the routing tables that are applied in a distributed fashion by all nodes, switching the routing metric in use demands a synchronized exchange of all RP implementations in all nodes. This requirement makes it practically impossible to test, or even let individual CN users temporary select a particular routing metric for their own traffic.

2.3.2 A Community Network Characterization and Interaction Tool

2.3.2.1 Objectives and Vision

The envisioned benefits of the NCT for CNs users shall be illustrated with an example: A non-expert CN user is trying to obtain the best possible down-link connection to a server out of the CN, reachable only via various multi-hop away Gateways (GWs) or proxies. By looking at the current network topology shown on the web interface of the home router (the CN node), the user can learn about the topology of the CN, the capacity of the links, etc. and actively measure network performance in real time. As a result, the user can take actions to improve the network usage experience, by configuring the home router to apply different routing policies that lead to better performance. This way, optimal trade-offs between path delay, bandwidth and packet loss can be selected for different applications (voice over IP, large downloads, etc.)

2.3.2.2 The NCT: Architecture, Components and Considerations

The NCT provides CN users a graphical tool to let them learn about the CMN and manage their CN home routers. This is achieved with two software components that integrate together. On the highest abstraction layer, the NCT user interface (NCui) web-based tool shows information about the network, evaluates performance parameters and provides interaction mechanisms to change routing metrics, etc. The interface integrates with the NCT daemon (NCd) on the lower abstraction layer. NCd manages the interaction with the local device components and the remote network nodes. The architecture of the NCT is depicted in Figure 2.26, showing the external components NCd is surrounded by.

The NCT daemon (NCd) The core of the NCd is the *lunced* daemon (which stands for ***Lu****a[103]* ***N****etwork* ***C****haracterization* ***D****aemon.* It interacts with the other system components and enables the communication between them. It has a modular design, where *plug-ins* provide specific interaction functionalities (e.g. with the routing algorithm). This interaction is divided in three blocks: *user I/O*, *[mesh] network I/O* and *system I/O*.

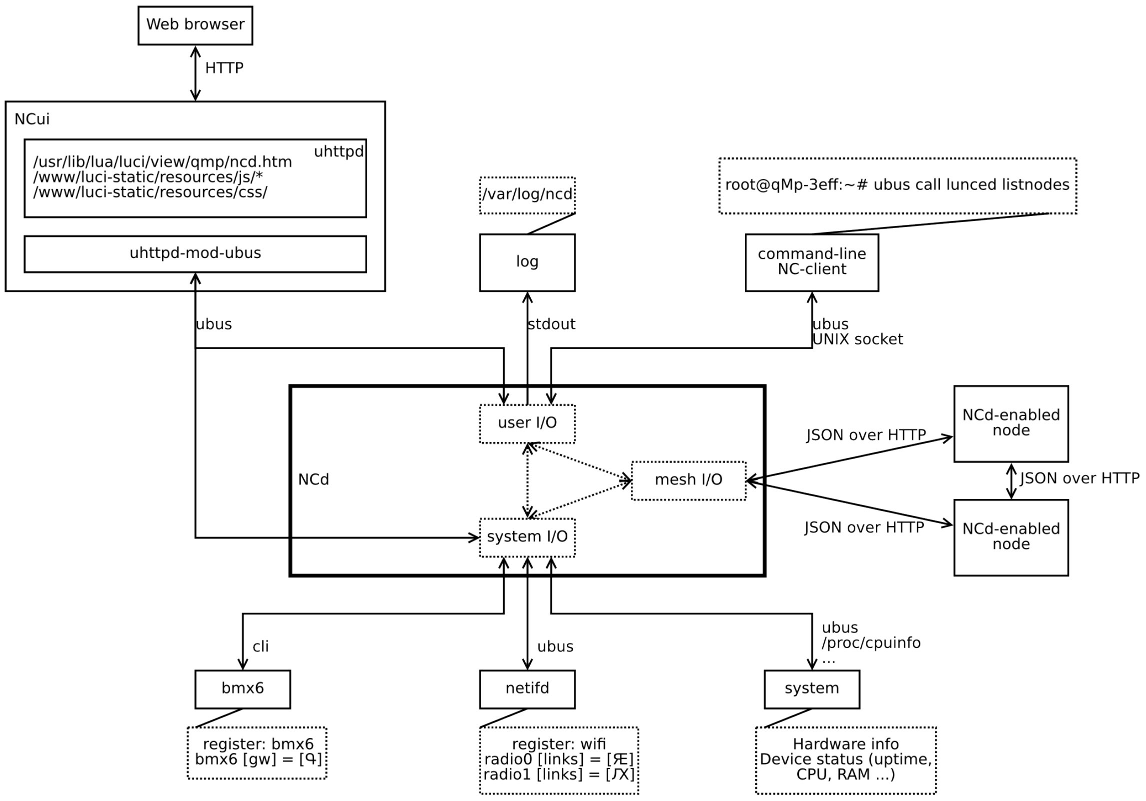


Figure 2.26: Architecture of the network characterization tool

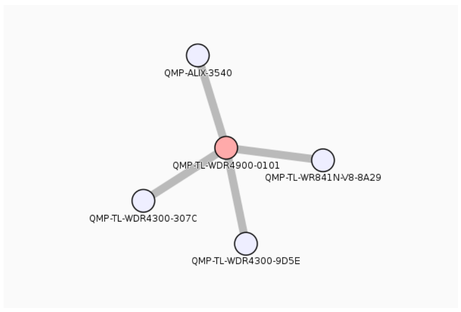
User I/O interaction in *lunced* comprehends a set of tools to allow the end user to interact with the NCd in three different ways so far: *web interface*, *system log* and *command line*. The *system log* interaction is possible via a specific plug-in that allows *lunced* to log certain system events (e.g. requests received from remote NCd-capable nodes). *Command-line* interaction is based on the interface provided by OpenWrt[32] Ubus[104]. The NCd is actually designed to perform the communication between the NCui and the NCd via Ubus.

In CMNs it is common to find two very different types of users. On the one hand there are those who, once their network node is installed, avoid as much as possible touching it. On the other hand, there are very enthusiastic users who are eager to test and update their node even with intermediate router-firmware releases. In terms of the software running, this leads to very heterogeneous networks. This has been taken into account, to ensure that different versions of the NCd can coexist with minimum backwards compatibility issues, or even with nodes that are not NCd-capable at all.

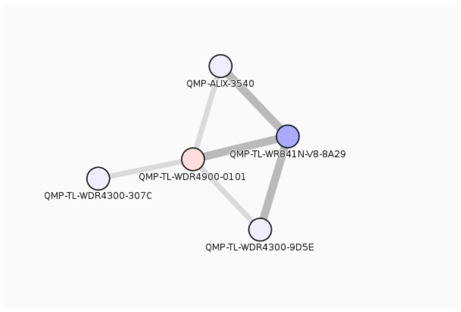
The modularity of the NCd allows room for all types of plug-ins and new developments. For instance, if a new RP is added, only a plug-in needs to be created.

The NCT user interface (NCui) The NCui provides CN end users a visual tool to monitor the network and perform network administration tasks accordingly to improve their usage experience. It is based on an graph showing the network nodes and the links connecting them. Its user-centric approach makes much sense in the context of a mesh network with BMX6 as the main routing protocol. Unlike other so-called *link-state* RPs, BMX6 is a *distance-vector* RP and it is not aware of the complete network topology (see 2.3.4.1).

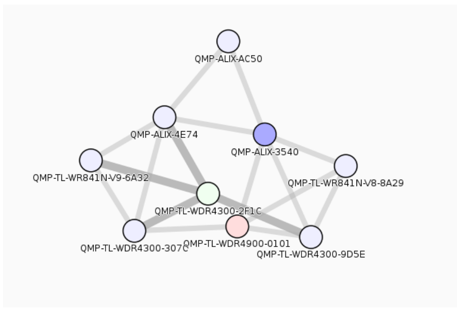
The process of discovering a whole CMN with the NCui can be seen in Figure 2.27. Fig. 2.27a shows the initial view, with the local node in the middle and its three neighbours around. In Fig. 2.27b, the graph is centred over the node on the right after its list of neighbours has been requested. No new nodes are added, but links previously unknown have appeared. The discovery process is repeated until all the nodes appear on the graph (Fig. 2.27c depicts the whole CMN). Finally, Fig.2.27d uses a palette (blue → green → yellow → orange → red) to colour the links in function of their measured quality. More features not shown here are provided by the NCui, like a tool to discover network paths from one node to another.



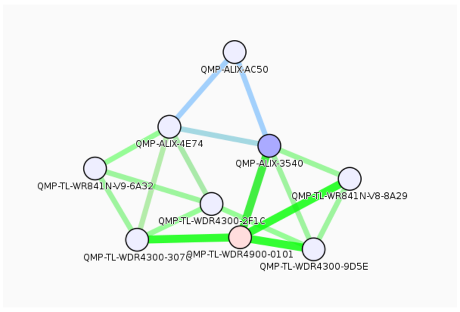
(a) Local node in the center, with all its neighbours and links.



(b) The second node's neighbours and new links are shown.



(c) All the mesh nodes are shown, and also the links connecting them.

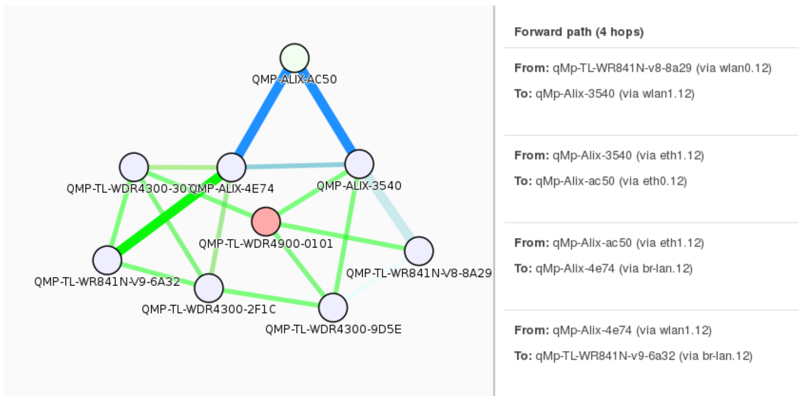


(d) The network links are coloured as a function of their quality.

Figure 2.27: Discovery of the nodes in a mesh network and the links' properties in four steps using the NCui.

In addition to the visual information displayed by the graph, the NCui also provides text-based data about the nodes, the routing algorithm, paths, etc. by means of a sidebar and floating HTML divs when needed. Figure 2.28 shows the BMX6 path discovery between two nodes in the mesh network.

Inter-NCd communication One of the key features of the NCd is that the *lunced* daemon can exchange information by sending and receiving JSON queries over HTTP. To do so, a special function is used to forward local queries to other nodes running the NCd. This occurs when the user manually requests information about a remote node or at time intervals, but only when the NCui is in use.



Forward path (4 hops)

From: qMp-TL-WR841N-v8-8a29 (via wlan0.12)
To: qMp-Alix-3540 (via wlan1.12)

From: qMp-Alix-3540 (via eth1.12)
To: qMp-Alix-ac50 (via eth0.12)

From: qMp-Alix-ac50 (via eth1.12)
To: qMp-Alix-4e74 (via br-lan.12)

From: qMp-Alix-4e74 (via wlan1.12)
To: qMp-TL-WR841N-v9-6a32 (via br-lan.12)

Figure 2.28: Screenshot of the NCui showing the mesh graph and highlighting the path from the far right node to the one on the far left.

One of the current shortfalls of the NCd implementation is that all the requests a node receives from other nodes in the mesh are executed, no matter where they come from. The next development effort will take this into account to, at least, provide a configuration option to tell *lunced* to only attend read-only requests and discard the ones intended at changing the node's settings. Ideally, an authentication mechanism should be implemented (for example, by exchanging RSA keys and sending write commands securely via SSH).

NCT source code and OpenWrt package for dissemination The source code of the NCT is freely available [105] under a GPL licence and can be run on the current (as of February 2015) OpenWrt stable version. Packages for OpenWrt are provided for the development and releases of qMp that could be included by node admins on demand.

2.3.3 Approaching and upgrading a real-life CN

The Network Characterization Tool (NCT) deployment is based on field experience in Guifi.net[36]. This very heterogeneous CN is mostly built up with nodes that work with the BGP and Open Shortest Path First (OSPF) RPs. Most of these nodes are connected by wireless links operated in infrastructure (Access Point (AP)/Client) mode. However, in the last years, some parts of Guifi.net have started to be deployed as CMN, operating in ad-hoc mode and using BMX6[38] as the main RP inside them. Most of these CMN are geographically located close to Barcelona (Catalunya), but operate as independent *mesh clouds* and are linked by *infrastructure* network.

CMNs in Guifi.net are mainly managed by the users who own the nodes and participate there, or by a few network administrators, on which users trust. This means that administrative access can not be immediately obtained (if at all) on certain nodes. Furthermore, very different versions of the qMp router-system firmware are in use in these CMNs, as some nodes are not regularly updated after installation. To install the NCT software and perform experiments with the nodes both administrative acces and up-to-date firmware versions are needed. This makes almost impossible to deploy it in the short term in larger CMNs like GuifiSants[35] (with some 40 active nodes).

2.3.3.1 History and current state of the Raval CMN

The Raval CMN is one of the more recently deployed ones in Barcelona. The first nodes were installed in 2013 by Routek[106] with the financial support of the city council as part of a bigger plan[107] to reduce the digital divide in this district of the city. This project involved local organizations related with the *community sector* and the *digital inclusion* (non-profits, etc.).

At the time of developing NCT, there are 17 nodes running qMp in the Raval CMN and using BMX6 as the main RP. All the nodes work on the same 5 GHz unlicensed frequency, and they are strategically deployed in an area of less than 1 km^2. On 4 of the locations there are two *backbone* mesh network nodes with sector antennae to cover, in total, about 240 degrees horizontally. The rest of the network locations have a single wireless device with more directional antennae. One of the backbone nodes has a dedicated infrastructure link that connects to the rest of Guifi.net and operates as an Internet GW. The Internet connection is provided by EXO[108], a non-profit closely related to Guifi.net. Additionally, a private user offers another Internet GW sponsored by him. The main part of the traffic is generated by the 7 residential users that connect to the network and the passers-by that use a free HotSpot.

For both technical and logistic reasons, this CMN is specially suitable for running the NCT experiments.

2.3.3.2 CNs organization, ownership and administration

The administration model of CN reflects the social background of the people behind them. In the case of Guifi.net, among the first promoters there use to be *hacktivists* that build and operate an embryonic network in some part of the city. During the initial growth phases they are the main administrators of the network, until more people joins the network and a critical mass is reached. From this point on, the network grows organically, and newcomers receive support not only from the first hacktivists, but from others already in and with a broader scope of interests and skills.

Very often, people find out about CNs in search for non-expensive or even free Internet access (the misconception of *free as in freedom* for *free as in "gratis"* is a common reason for disappointment and turn-down). However, those who stay engaged appreciate the values embedded in such organizations, like social economy, freedom of speech, alternative and responsible business models.

2.3.4 Network characteristics and NCT performance measurements

Some of the routing protocols adopted worldwide by ISPs, carriers, etc. (like BGP, Routing Information Protocol (RIP) or OSPF) are also widely used in the context of CNs. However, they are not always optimal for dealing with the particularities of a CN, specially in the case of the very dynamic CMNs. There, the links' physical properties rapidly change, needing for routing mechanisms that are able react to these changes in shorter times than those for other use cases.

The BMX6 RP is specially designed for CMNs, where every user contributes with his own network node. As this *home* network router is, in most cases, solely under the control of the user owning it, he is able to perform network administration tasks that deliver a better network usage quality of experience[9]. In BMX6 this is achieved by using different algorithms for calculating the route of the packets going towards the user's node.

2.3.4.1 BMX6 routing metrics

BMX6 is a distance-vector routing protocol that accounts for several metric calculation algorithms. The default one is Vector Bandwidth (VB), which calculates an estimation of the capacity of the path that packets will follow from one node to another. HC, Multiply Path (MP) and Expected Bandwidth (EB) can be found among the others available.

Metrics formulation Every node is able to inform the rest of nodes about the algorithm and the associated parameters to be used for calculating the path metric *towards* it. At the same time, every node has an estimation (either a default value or configured by the network administrator) of the maximum capacity of each one of the network interfaces. A *links matrix* c_{ij} represents the maximum capacity between node i and j.

A node running the BMX6 RP broadcasts UDP probe packets (named HELLO messages) to its neighbours. These messages are used by the neighbours to calculate a *quality matrix* q_{ij}, which is defined as the ratio of successful HELLO messages received from i by j. q_{ij} ranges from 0 (no link) to 1 (perfect link).

During an initial phase, the BMX6 daemon of every node creates a register of all the nodes in the mesh network and updates their identifiers. By means of an efficient flooding mechanism, BMX6 propagates information sets from one node to the rest of them. Then a path metric is calculated in order to set up proper routing decisions. This calculation is performed using different functions, which can be configured by defining the algorithm and its associated exponents.

Consider the set of adjacent nodes of node i indexed as $1 \le k \le M$. With the transmission and reception exponents α_j and β_j received from node j, the link quality matrices for every adjacent node k are built:

$$Q^j_{ik} = u_m q^{\alpha_j}_{ik} q^{\beta_j}_{ki} \tag{2.8}$$

Then, to take the capacity of the link between nodes into account, we also build the links' bandwidth matrices:

$$B^j_{ik} = c_{ik} q^{\alpha_j}_{ik} q^{\beta_j}_{ki} \tag{2.9}$$

Finally, μ^j_{ki} is defined as the metric to reach destination node j when transmitting from node k to node i. The metric function depends, in general, on Q^j_{ik}, B^j_{ik} and μ^j_{ki}. To sum

[9]In a CN this is perfectly acceptable as long as it is done under a fair use basis and not worsening other users' network performance

up, the *path metrics matrix* μ^*_{ij} contains the best metric from node i to node j expressed as in (2.10).

$$\mu^*_{ij} = \max_{1 \leq k \leq M} \phi(\mu^j_{ki}, Q^j_{ik}, B^j_{ik}) \tag{2.10}$$

where ϕ is the particular metric function used.

2.3.4.2 Metric functions

The BMX6 daemon implements a wide set of metric functions, which are introduced below.

The VB algorithm uses function 2.11 below to calculate the path metric for non-zero argument values.

$$\frac{1}{\phi^2} = \frac{1}{\mu^2} + \frac{1}{B^2} \tag{2.11}$$

The EB and Expected Quality (EQ) algorithms use very similar metric functions (2.12). Both of them are based on adding the inverse of the metric values.

$$\frac{1}{\phi} = \frac{1}{\mu} + \frac{1}{x} \tag{2.12}$$

where $x = Q$ for expected quality and $x = B$ for expected bandwidth. When the exponents take the particular values $\alpha = \beta = 1$, the so-called ETT metric (2.13) is obtained:

$$\frac{1}{\phi} = \frac{1}{\mu} + \frac{1}{c_{ik} q_{ik} q_{ki}} \tag{2.13}$$

Finally, another set of metric calculation functions (2.14) are based on the multiplication of different terms:

$$\frac{1}{\phi} = \frac{1}{x} \frac{u_m}{Q} \tag{2.14}$$

where $x = min(\mu, c)$ is used for the Multiply Bandwidth (MB) algorithm and $x = \mu$ for the Multiply Quality (MQ).

2.3.4.3 Experimentation environment in the Raval CMN (Barcelona)

In order to test the NCT capabilities for performing network experimentation outside of a controlled testbed environment, the Guifi.net CMN of the Raval district in Barcelona has been selected (see 2.3.3 for more details).

In this small∼middle-sized and very densely meshed network, almost all the nodes are directly connected with at least five other nodes. This means, a priori, that a large number of possible short e2e paths between two nodes can be used, rather than a few paths with many hops in between. In addition, all the nodes are built with similar hardware (in terms of CPU architecture, transmission power, antenna radiation patterns) and installation techniques (orientation, etc.) so they are expected to all offer very similar performance.

The first experiments that have been performed using the NCT in a real-life CMN analyse the paths diversity between two specific nodes over time and the round-trip time of ICMP ping probes with different routing metrics calculation algorithms in use. The preliminary information about the BMX6 RP behaviour in this CMN, the performance measurements, etc. could be very valuable for improving the network quality of experience of the end users.

Diversity in path discovery experiment The objective of this first experiment is to evaluate the diversity of paths that BMX6 chooses over a period of time to transmit packets from one node (the *origin* node) to another (the *destination* node). Since BMX6 is a distance-vector protocol, in order to discover the full path, next hops towards the destination need to be asked iteratively, starting at the origin node, until the last hop to the destination is found. Each node decides, at the time of forwarding the packet, which node to send it to. This decision is taken according to the calculated metrics matrix (see section 2.3.4.1) and the metrics calculation algorithm announced by the destination node. In this experiments set-up, the default algorithm (VB) and exponents (1/2, 1/1) are used.

To perform the path diversity experiments, the origin and destination nodes in the Raval CMN have been selected with the following criteria: they must not have direct connection between them (i.e. not be neighbours) and they must have good connectivity with at least three other mesh nodes. This way, it is potentially probable to obtain a greater paths diversity.

Four rounds of path discovery measurements have been performed, each finding the path from the origin node towards the destination node 100 times. In each round, different values for Δt (the pause between consecutive path discoveries) have been used: 0 s, 5 s, 10 s and 30 s.

The experiments data show that, after the 400 paths discovery measurements, a total of 11 different unique paths have been recorded. Fig. 2.29 shows the probability of their occurrence on the 4 rounds, along with the total experiments average. In the plots only the 9 more used paths have been represented, sorted by order of appearance in the results.

As can be seen in the figure, path 1 is clearly the preferred path, used in more thant than 60% of the occasions, followed by number 2 (15%) and number 3 (12%). These paths consist of, respectively, 2, 3 and 2 hops. The six next paths represented account for 2, 2, 3, 2, 3 and 4 hops, and the two remaining paths (not shown in Fig. 2.29) both consist of 3 hops. These numbers confirm the preliminary impression about the high density of the Raval CMN (the 2-hops paths indicate that origin and destination nodes share at least 5 common neighbours).

ICMP ping performance with different algorithm experiment The goal for this second experiment is to evaluate the Round-Trip Time (RTT) of ICMP ping probes when different BMX6 metric calculation algorithms are in use (see section 2.3.4.1 for reference). This is a first approach to one of the main objectives of the NCT: to provide

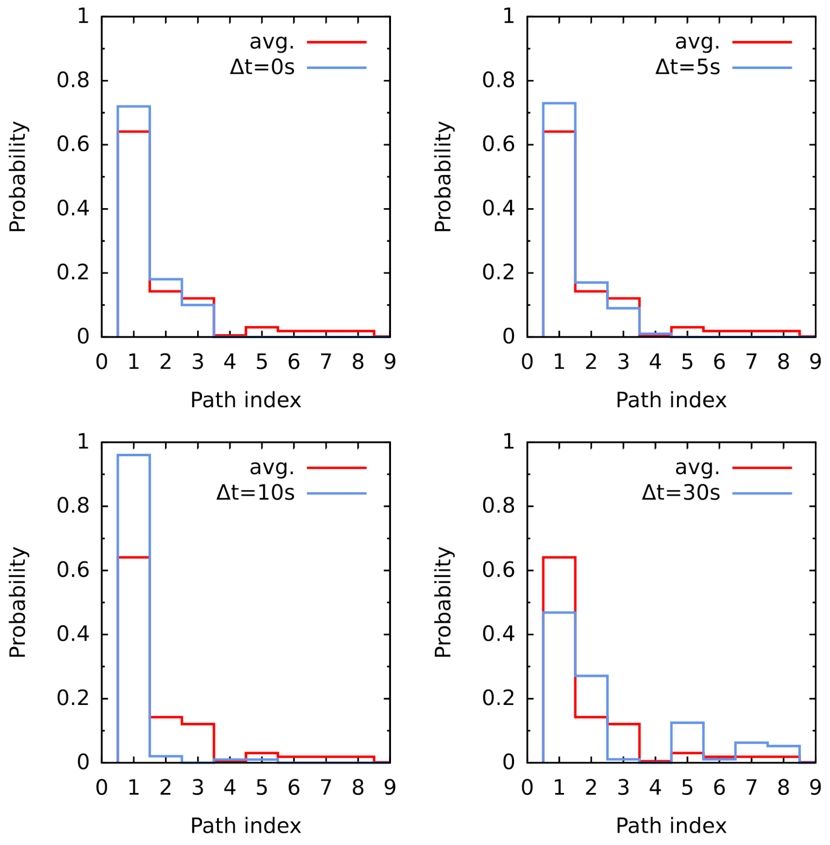


Figure 2.29: CDF of the paths' occurrences in the path discovery experiments, ordered by first appearance. The interval between consecutive path discoveries is indicated by Δt. The average corresponds to the four rounds of experiments.

end users information about e2e network performance parameters when the routing protocol configuration is modified.

In this experiment, the same two nodes as in section 2.3.4.3 have been selected, expecting that having a wide list of possible paths will be reflected in the ability of BMX6 to take advantage of them under different metrics calculation algorithms.

Eight rounds of ICMP ping tests have been performed, testing the EQ, MB, EB and VB metrics calculation algorithms with two different ICMP packet sizes (64 B and 1008 B). On each round, 5 series of 20 pings have been performed at intervals of 60 s.

The most convenient way to obtain valuable data from the results of the experiment is to plot their [cropped] CDF, as can be seen in Figure 2.30. Differences from one algorithm to another can be seen there. On the left sub-figure, for instance, VB clearly outperforms the other algorithms, while the EB algorithm incurs in significant delays (beyond 50 ms, though without packet loss). On the right sub-figure, all the algorithms perform worse than before (surely due to the higher packet size). Three of them (VB, MB and EQ) have similar performance characteristics, while EB outperforms them and even behaves better than with the smaller packet size.

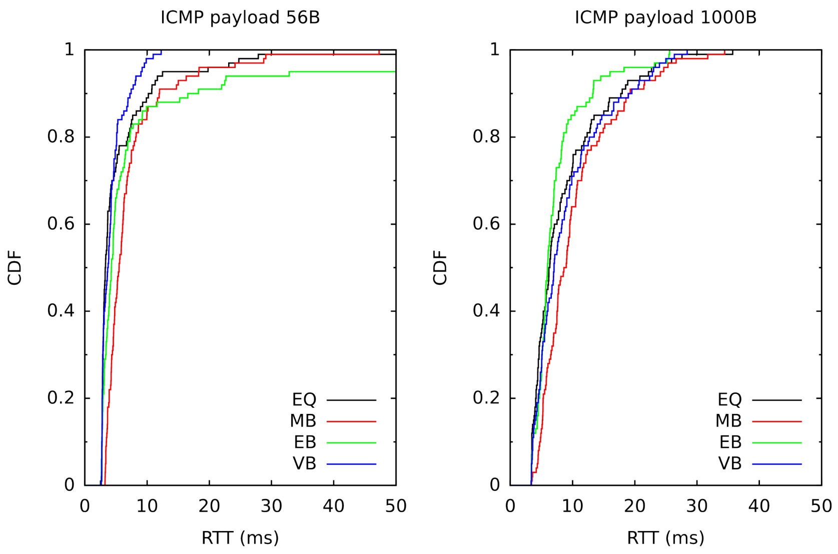


Figure 2.30: CDF of the paths' occurrences in the path discovery experiments, ordered by first appearance. The average corresponds to the four rounds of experiments.

2.3.4.4 Discussion

Before reaching any conclusion, it is worth noting that these two experiments presented have been the first ones to be performed with the NCT in a real-life CMN, with the difficulties described in section 2.3.3 in comparison to a laboratory testbed environment. Only a very small fraction of the many network performance aspects has been monitored and for a short period of time.

This said, however, these experiments become a proof of concept of the usefulness of the NCT for evaluating the network performance in a CMN. The preliminary results show that, in a very densely meshed network like the one described, even when many paths are potentially available only a few of them are intensively used while the others are marginally chosen. In terms of network performance, observable differences have been found between the tested metrics calculation algorithms and in function of the packet size.

For future experiments, in the Raval CMN or elsewhere, the effect of the different algorithms in the diversity of network paths shall be analyzed, in order to improve the understanding of the behaviour of the network as a whole. In terms of network performance, additional tests shall be performed (e.g. throughput) for different types of traffic (real-time multimedia streaming, web browsing, P2P transfers, etc.).

2.3.5 Summary

In this Section 2.3 we presented and discussed the NCT, a tool providing network interaction capabilities for CMNs users to control routing parameters and monitor and evaluate network performance from within the web interface of their home routers. The objective of the tool is to provide them with life information on the current topology, paths, and performance of their CN and allow them to take administrative actions on their devices to improve the experienced network quality for the different Internet applications involved.

We integrated the NCT into the wireless router firmware qMp and described the experienced challenges and benefits from its deployment in a productively used community mesh network in the Raval district in Barcelona/Catalunya.

2.3.5.1 Future work plan

In terms of development, the NCT is not yet ready to be used by average CMN users, as more testing has to be performed outside the laboratory testbed, in real CN environments. In terms of the adoption by the community, a few users have tested it at different development stages and have provided informal but valuable reports on usability and desired features. The tool, once it is mature enough, is expected to be included by default or optional with future qMp router-system version.

From the experimentation point of view, we plan to perform more systematic tests, considering additional routing-metric and mechanisms, broader diversities of possible e2e paths (as typically given by larger networks), and long-lasting experiments, to better understand and exploit the full potential of employed RPs and the whole CMN infrastructure.

Chapter 3

Evaluation of Mesh Routing Protocols

In this chapter we focus on one important self-management mechanism, routing, and we study the scalability, performance, and stability of three proactive mesh routing protocols: OLSR [99], BMX6 [38] and Babel [101], three common routing protocols in wireless community networks. We study different metrics on an emulation framework and on the W-ILab.T testbed at iMinds under different networks' sizes and characteristics, making the most of the two worlds. Emulation allows us to have more control over the topology and more systematically repeat the experiments, whereas a testbed provides a realistic wireless medium and more reliable measurements, especially in terms of interference and resource consumption. These routing protocols have been further characterized by studying their control overhead, convergence delay, CPU and memory consumption, and stability. Our results show the relative merits, costs, and limitations of the three protocols.

3.1 Related Work

Ad hoc network routing protocols have been extensively studied in the literature; however, most of the work done focuses on mobile ad hoc networks. The performance of routing protocols is expected to be different in wireless mesh networks, where the backbone mesh nodes are static and do not have energy constraints.

Among the three protocols considered, OLSR has received more attention: its protocol overhead, route convergence time, delivery ratio, end-to-end delay, and throughput have been compared in simulations with those of AODV, HWMP, DSR, TORA and DSDV ([49, 50, 109–112]). There are also some real testbed experiments that compare OLSR with BATMAN([48, 113]) or AODV([114–116]) in networks that range up to 49 nodes and some provide additional information such as CPU and memory consumption.

The results in these studies demonstrate that compared to on-demand routing protocols, OLSR has comparable results in terms of delivery ratio, while the end-to-end delay is lower, and the overhead is lower in dense networks with many flows and higher in the contrary situation. The OLSR protocol has been also compared with other proactive routing protocols, such as Babel and BATMAN in [52] and [113], showing that distance-vector protocols have lower overhead but do not necessarily achieve higher throughput.

The authors of [116] experimentally quantified and discussed the performance of OLSR, Babel, and AODV implementations based on a small (7-nodes) indoors testbed. Results are ranked using Kiviat diagrams to balance between the studied measures of overhead, energy consumption, packet reordering, delay, and loss. Although the obtained standard deviations for each measure indicate a high statistical validity and allow a clear and non-overlapping ranking between the different protocols, the relevance of equivalently compared but obviously correlated measures of different importance seems misleading.

Another shortcoming is the lack of plausible explanations for certain results, such as why the overhead of proactive routing protocols in an unchanging environment should depend on the distance (hops) between two communicating nodes or how application-traffic delay and loss measured between two nodes with only a single possible single-hop path between them could be affected by the routing protocol.

Regarding self-healing and convergence performance, while a tremendous amount of work has been employed in the last years on simulation-based research on the performance of mobile ad-hoc networking and mesh routing protocols regarding their performance [117–120] and tuning [121, 122], much less insight hs been published based on experimental analysis using emulation [1, 52, 123] or testbeds [116, 124, 125], leaving the experimental performance evaluation of changing topology aspects on mesh routing protocols surprisingly unattended.

Despite this gap of profound experimental evaluation, pioneering projects have started to work on concrete solutions targeted for low-budget IEEE802.11 enabled devices, such as the Serval project [126], promising a communication anytime and anywhere even in the absence of phone towers and other supporting infrastructure. Another project, however, which does not focus on mobile routers, is the Village Telco project [127]. It is interesting to note that both of these projects have changed the underlying routing protocols since their existence due to critical performance issues found in live deployments.

In addition, it is worth mentioning the Battlemesh event [34], which is an annual gathering of mesh-routing protocol developers with the objective to compare their protocols in different challenging environments and scenarios, including mobile scenarios. Unfortunately, due to the dynamics of these events, reproducible results obtained via systematically performed measurements have not yet been documented. However, measurement snapshots from past years, capturing the performance in semi-static and mobile scenarios suggest a huge gap between simulation-based and experimentation-based measurements.

This work extends our earlier studies in [1, 7, 128] which, compared with other work such as [52, 113, 116], also evaluates the effect of network dynamics (e.g. link failures) and topology changes; topological constraints that must be expected from any real-world wireless mesh deployment. Another relevant difference of our work is that results are based on the combination of various scenarios and topologies and are thus not tightly coupled to a particular deployment, which allows us to generalize results and derive expectations for deployments with different topological structures, sizes, or node densities. Also, to the best of our knowledge, none of the existing works has experimentally and comparatively studied the performance of existing proactive routing-protocol implementations when exposed to highly changing topologies as evaluated with our reactivity and mobility experiments.

3.2 Mesh Routing Protocols

Routing is a critical function in wireless mesh networks, since it decides the path any packet must follow to reach its destination. In a community network that grows organically, with several hops from the source to the gateway and where network management is not done by a single entity, but by many members of the community in a decentralized way, it is imperative that a routing protocol is able to continuously adapt to network changes. Routing protocols are usually classified as proactive or reactive, based on whether they learn routing paths proactively or just when needed (reactively). In [23], it can be seen that the vast majority of community networks use proactive routing protocols, since nodes do not have energy constraints. Additionally, proactive routing protocols are more efficient in terms of packet delay and outperform reactive protocols when the number of flows in the network increases [129].

One of the goals of our evaluation is to understand the consequences of choosing either a distance-vector or a link-state paradigm and how it affects scalability. Distance-vector routing protocols follow the Bellman-Ford algorithm, sharing only aggregated information about the path metrics, whereas link-state protocols share the whole view of the network, and the metric of every single link is known by every node. On the representation of link-state routing protocols, the choice is OLSR, which can be seen as a reference as it is the most studied [52, 130–134] and used [23] link-stated routing protocol. On the distance-vector side, we have chosen BMX6 and Babel. Babel has been chosen because it is a clear implementation of a distance-vector protocol and BMX6 because of its recent popularity in existing community network projects (e.g., Guifi.net/qMp [3] and Libre-mesh [33]). In addition, BMX6 uses the Secure Hash Algorithm (SHA1) hashes instead of IP addresses as node identifiers and implements a number of features to reduce the protocol overhead while keeping the protocol as reactive as possible. The other famous Layer-2 BATMAN [135] has not been included in our evaluation because of the difficulty of comparing it with the Layer-3 protocols. All four above protocols have been extensively discussed and experimentally challenged in last years Battlemesh [34] workshops.

In the following subsections, we explain how these three routing protocols work by describing the mechanisms used for neighbor discovery (How does a node know other mesh nodes in range?) and topology dissemination (How does a node learn about routes to nodes that are not directly reachable?).

3.2.1 Babel

Babel is a proactive, distance-vector routing protocol based on the Bellman-Ford protocol [101]. Its main concern is to limit routing pathologies as routing loops or black holes, which it achieves using a proper feasibility condition and attaching a sequence number to routing updates.

Babel's feasibility condition determines which of the received routing updates should be considered and which should not; a routing update for a route is feasible only if its metric is smaller than any of the routing updates for the previously advertised route.

The sequence number attached to a routing update is generated by the destination node it announces and determines to which other routing updates the metric can be compared. Only information with the same sequence number is comparable.

Neighbor discovery. Babel nodes discover its neighborhood by exchanging two types of messages.

- *Hello* messages are sent to a multicast address by every Babel interface with a sequence number that is increased locally every time a new *Hello* message is sent. By listening to *Hello* messages, a node not only discovers its neighboring nodes, but it also estimates the reception cost (*rxcost*) of that link. By default Babel sends a *Hello* message every four seconds.

- *I heard you (IHU)* messages are used to determine the bidirectionality of a link and share the *rxcost* with the neighboring node. The *IHU* messages are conceptually unicast; however, they are sent to a multicast address to avoid address resolution protocol (ARP) exchanges and to aggregate multiple messages in a single packet. They are also sent periodically, but usually not as often as a *Hello* messages; by default they are sent every 12 seconds.

Topology dissemination. In Babel, nodes discover far away nodes by sharing their routing table in route update messages.

- A *route update* message announces a route and its associated cost, and every Babel node sends a periodic update for every node it can reach to a multicast address. Additionally, when there is a significant change in the network topology, such as a route retraction or a significant change in the metric, an unscheduled route update is sent, so that periodic updates do not need to happen as often (by default every 16 seconds).

When a node receives an update, first of all it checks its feasibility, and if feasible, it computes the accumulated metric by combining the metric on the update message plus the cost of the link from where the update is received.

3.2.2 BMX6

The BMX6 protocol is also a proactive, destination-sequenced distance vector protocol whose main goal is to reduce the size of periodic messages to achieve low routing overhead while attaining high reactivity to network changes. The key concepts behind this are (i) using a stateful-compressed communication between neighbors and (ii) the context-specific propagation of local versus global and static versus dynamic information.

In a mesh network with flat addressing, reducing overhead using stateful communication translates largely to the use of compact (16 bit) local identifiers to refer to other nodes, since addresses (specially with IPv6) are very long. Therefore, every message sent by a node will use its own local identifiers instead of a global one, which has been previously shared.

On the categorization for information, static information refers to such addresses and other details about a node that are unlikely to change; those attributes are gathered together into the node's *description.* On the other hand, dynamic information refers mainly to link and path costs estimations. The global versus local separation determines which information is kept within the neighborhood and which is flooded through the network; local identifiers and link costs are kept locally, while path costs and node descriptions are shared globally.

Neighbor discovery. Neighbors are discovered in a similar fashion to Babel.

- *Hello* messages are sent to a multicast address periodically with a sequence number that is locally increased every time a new *Hello* message is generated. By default, *Hello* messages are sent every 0.5 seconds.
- *Report (RP)* messages are sent with the same interval as *Hello* messages and report the number of *Hello* messages received. By counting the number of *Hello* messages received from a node and knowing the number of *Hello* messages that a node has received, a node can compute both the transmission and reception costs of a link.

Topology dissemination. Routes to other nodes in the network in BMX6 are obtained as a result of the flooding of originator messages.

- *OriGinator Messages (OGM)* are sent periodically by every node (originator) to announce its presence and then re-sent if appropriate by any node that receives it. An *OGM* contains the sender's local identifier of the originator, a sequence number, and a metric that measures the cost of reaching it from the sender's perspective. When a node receives an *OGM*, it computes the cost of reaching the originator by combining the metric announced in the *OGM* with the cost of the sender's link; if this cost is smaller than the cost via any other neighbor, then the node will re-multicast the *OGM*, after updating it with its local identifier and the metric computed. By default, a node generates an *OGM* every five seconds.

Additionally, static information is shared on demand. When a node receives an *OGM* or a *Hello* message with an unknown local identifier, it will ask the sender for the node description's hash. This hash allows the node to determine whether the local identifier refers to any of the known nodes, and if it is not the case, then it will request the node's description and update its knowledge conccerning the network.

3.2.3 OLSR

In contrast, OLSR, as its name points out, is an optimized link-state routing protocol. The optimized part comes from the optimization on the flooding mechanism; only nodes selected as multi-point relays (MPR) retransmit the node's messages.

Topology dissemination. As any link-state routing protocol, OLSR provides every node in the network with a (partial) view of the whole topology by flooding the network with *Topology Control (TC)* messages.

- A *TC* message describes all the nodes that are reachable from the message creator, as well as the quality of the involved links in both directions. The TC messages are generated periodically by every node in the network and are then retransmitted unchanged throughout the network. By default, the implementation used in our experiments transmits a TC message every five seconds.

Neighbor discovery Neighborhood sensing is performed in OLSR by periodically sending *Hello* messages.

- A *Hello* message consists of a locally increased sequence number and the list of known links to the sender's neighbors as well as their quality and the quality from the neighbor's perspective. Link quality is computed as a function of the number of received *Hello* messages from that neighbor, while the quality from the neighbor's perspective is simply the quality reported in its *Hello* messages. *Hello* messages are sent to a multicast address periodically, by default every two seconds.

In this evaluation, we have used the popular implementation by olsrd.org[99], which implements three fundamental changes to the original RFC. First, the MPR optimization itself, originally designed for wired scenarios with loss-free links, is modified to require not only one but seven nodes to reach every two hop neighbor. Otherwise, when considering even the weakest detected link as a reliable resource for the dissemination of topology information, massive network instabilities must be expected. Second, to compensate for the overhead introduced by the increased MPR redundancy, the fish-eye extension has been introduced [130] where TC updates are exchanged less often between far away nodes than between nearby nodes. This is achieved by letting the originator of each TC message use different time to live (TTL) values with the consequence that only every second TC message propagates beyond the first hop. Third, the path metric used by the olsrd.org implementation is based on the expected transmit count (ETX) metric [75] which, compared to the originally proposed hop count metric, provides a better reflection of the real path cost for transmitting a packet via wireless links. Although controversy on the best parameterization of this protocol exists (such as the findings published by Johnson and Hancke in [132] on the performance of ETX and the hysteresis-based hop count metric), we decided to base our experiments on the defaults of this implementation because their current selection still represents a common ground that reflects the experience from its usage in several community networks over many years.

3.2.4 Summary

In essence, what differentiates these routing protocols is their topology dissemination mechanisms and how they solve and position themselves in the trade-off between convergence delay and overhead.

- In Babel, nodes only interact with their neighbors, sharing all the relevant information between them periodically. Routing updates are bigger because they contain the complete routing table, but are only shared locally. Overhead is reduced by retaining long intervals between periodic updates, while reactivity is increased by sending unscheduled updates when the network changes considerably.

- Additionally, BMX6 floods small OGM messages through the network, but principally the information shared is the same as on Babel, except messages are split and triggered differently. Overhead is reduced by compacting periodic messages as much as possible using stateful communication between neighbors, whereas convergence delay is minimized by having a very frequent exchange of messages.

- Further, OLSR is a link-state protocol; therefore, during topology dissemination, information concerning every link is shared, instead of path-aggregated information. Overhead is reduced using the fish-eye extension; updates are shared more frequently

Table 3.1: Periodic Messages

	Message	Size (B)	Interval	Messages/node
Babel	Hello	8	4 s	1
	IHU	16	12 s	$1 \times n$
	Update	$[\,12, 28\,]$	16 s	$1 \times route$
BMX6	Hello	$[\,4, 6\,]$	0.5 s	1
	RP	1	0.5 s	$1 \times n$
	OGM	4	5 s	1
OLSR	Hello	$[\,28, 32\,] + 20 \times n$	2 s	1
	TC	$28 + 20 \times n$	5 s	1

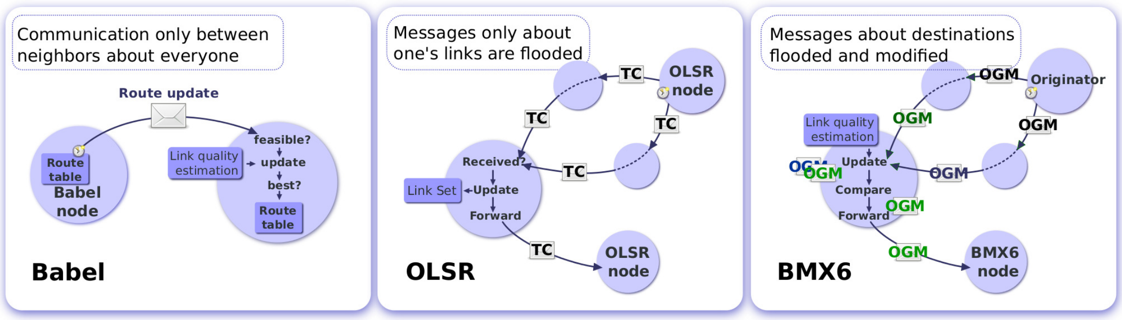


Figure 3.1: Topology dissemination mechanisms

with nearby nodes than with far away nodes, but to be adequately reactive, the time interval between updates is kept small.

Figure 3.1 summarizes the topology dissemination mechanism for each routing protocol. Table 3.1 lists the messages exchanged by each routing protocol, explaining its size and how many of them are exchanged depending on the number of neighbors (n) that a node has. The size given for each message type does not take into account the length of the headers.

3.3 Evaluation Metrics

There are several metrics to consider when evaluating the performance and overhead of a wireless mesh routing protocol.

The most common one is to measure its *network efficiency*, that is: how much routing traffic is necessary to be able to establish a connected network. Network efficiency is usually measured in terms of bytes/second or packets/second.

Given the relatively dynamic properties of wireless community networks we are interested in measuring how fast the routing protocol can adapt to these network changes or how stable the IP network is, given that the physical network is not stable. In terms of *stability*, we can measure the percentage of time the network is connected by pinging from some nodes to others or, similarly, measuring the longest time the network is not connected. From the perspective of reactivity, we can measure how long it takes to learn a new or better path, which is what we call *convergence time*. We are also interested in the cost in terms of the resource consumption of processing (CPU) and memory for each routing protocol. We must ensure the network nodes have the capabilities to run such protocols.

We measure the sensitivity of these metrics to the scale of the network, according to the variation of the size of the node neighborhood, variation of the total network size, variation of the length and number of hops of the network paths, and variation of the availability of links or the rate of changes in the network.

These measurements are performed and evaluated in different scenarios using container-based emulation and testbed-based experimentation. Both of them have different degrees of realism and flexibility. Emulation allows total control over environmental conditions, such as topology, availability, and changes but under limited realism. For instance, details taken from diverse real wireless community networks regarding the structure and events during a temporal period can be reproduced in a series of experiments and even be subjected to variations. In contrast, testbed experimentation provides a real environment, under stable environmental conditions, with a given set of nodes, radios, and locations, and control over a few aspects, such as transmit power and choice of nodes to use in an experiment among those available in the testbed.

3.4 Emulation Experiments

Using emulation, we can easily measure network overhead and convergence time. It is also possible to measure the cost in terms of memory, but the CPU cost is not reliable. Measuring the quality of the path is also complicated because the channels are not perfectly modeled. These metrics, therefore, are better studied in a testbed or real world deployment.

The emulation experiments presented in this chapter represent a summary of the results obtained during several experiments using the same emulation system. Table 3.2 shows the network characteristics used for each figure. The Barcelonès area corresponds to a large portion of the city and metropolitan area of Barcelona, and its topology was retrieved from Guifi.net Community Network Mark Up Language (CNML). *Generator* refers to topologies obtained using the generator from [41] with parameters from the Osona county, a representative semi-rural area where Guifi.net started. Each specific experiment with a given set of parameter values was repeated at least 20 times.

Our emulation system is based on mesh Linux containers (MLC) [39], which is a set of scripts based on Linux containers (LXC) and Linux networking tools, such as *ip* or *tc*. The MLC runs an LXC container for each node in the network and establishes the desired connections between them with a given link quality using *tc*, so that packets are randomly dropped and delayed with probabilities as configured. The system does not allow applying complex Wi-Fi models or reflecting the impact of interference between links of any kind.

3.4.1 Network Overhead

In our first set of experiments, we studied the effect of network size on the network overhead of each routing protocol. The topology emulated in this case corresponds with a

Table 3.2: Network characteristics of the emulation experiments

	Topology	Number of nodes	Number of links
Figs. 3.3a, 3.4a	Barcelonès	{10,20, 30, 40, 50, 60, 66}	{9,21,31,43,54,65,72}
Figs. 3.3b, 3.4b	Barcelonès	50	54
Figs. 3.3c, 3.4c	Barcelonès	69	{75,100,150,250,500,750,1000}
Fig. 3.5a	Barcelonès	66	72
Figs. 3.5b, 3.5c	Generator	50	75
Fig. 3.6a	Generator	49	74
Fig. 3.6b	Generator	{16,25,49,64,81,100}	{24,38,74,96,122,150}

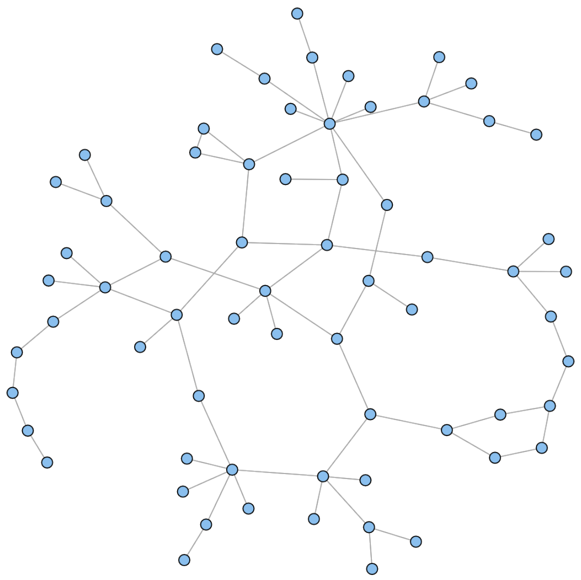


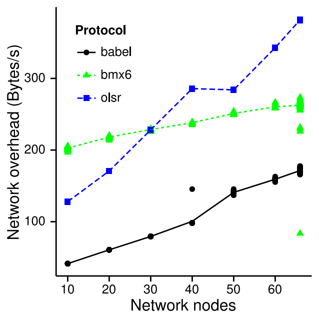
Figure 3.2: The topology of the Barcelonès network

representation of the Barcelonès area of Guifi.net (shown in Figure 3.2 and more details in Table 3.2), where each link quality was determined by averaging the measurements of one hour.

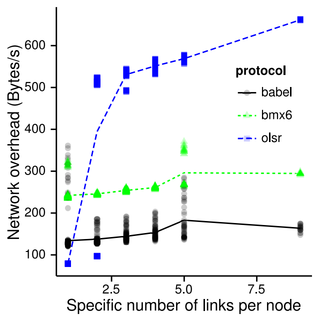
Figures 3.3a and 3.4a illustrate the network overhead in bytes and packets of each routing protocol on networks with different numbers of nodes. To obtain those networks, the original network was randomly sampled. As we can see, the number of bytes increases with the number of nodes, and OLSR seems to be the more heavily influenced protocol, Babel always has lower overhead but seems to increase at a faster rate than BMX6. Regarding the number of packets, all routing protocols seem quite stable except Babel, which has a step increase on 50 nodes.

Then for Figures 3.3b and 3.4b, we run the experiment using the Barcelonès network without any modifications, and it shows the overhead of each node depending on the number of neighbors each node has. As we can see, OLSR is more heavily affected by the number of neighbors, whereas Babel and BMX6's overhead in bytes only increases slightly with the number of neighbors. The number of packets looks stable in every case, except for a peak in Babel when there are five neighbors.

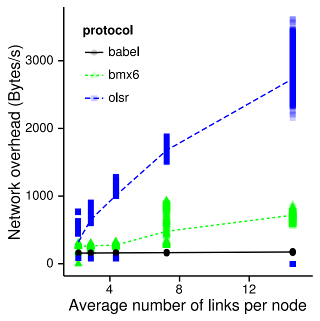
Finally, Figures 3.3c and 3.4c show the results when there is a fixed number of 69 nodes in the network, but the number of links is variable. As before, OLSR is the protocol that is



(a) Bytes vs. nodes

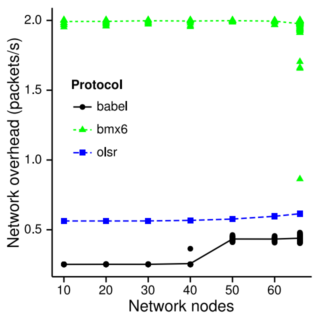


(b) Bytes vs. specific links

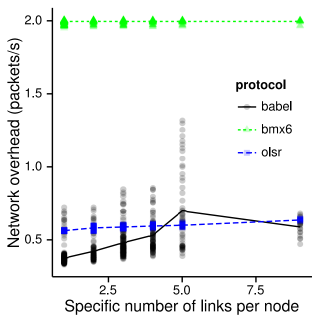


(c) Bytes vs. average links

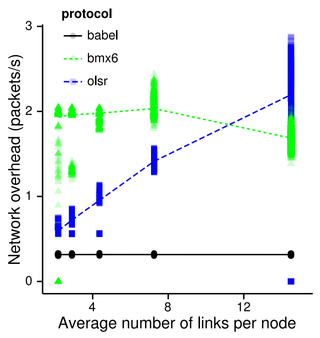
Figure 3.3: Network overhead in bytes



(a) Packets vs. nodes



(b) Packets vs. specific links



(c) Packets vs. average links

Figure 3.4: Network overhead in number of packets

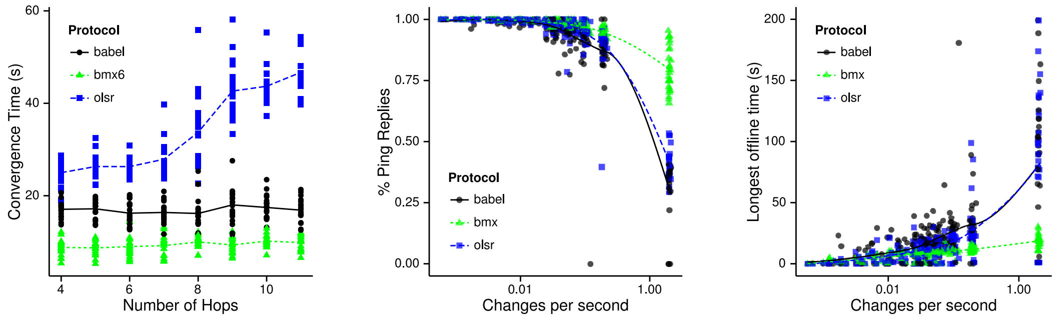
more heavily affected by the number of links, whereas Babel remains stable in every case. Further, BMX6 increases slightly as the number of links increases. Because every point represents the overhead for a node, nodes with higher numbers of links within the same network will have higher overhead, and this difference is greater for OLSR than for BMX6 and Babel. The results for the number of packets are similar.

3.4.2 Stability and Reactivity

Our second set of experiments attempts to characterize the stability and reactivity of the three routing protocols.

To measure the convergence time, we have looked for the longest path on the Barcelonès network. Then, we have measured how long it takes to discover new nodes that are attached to each of the nodes in the path from one of the endpoints. Figure 3.5a depicts our experiments' results. The OLSR obtains the worst performance, and we can clearly see a step function. This is due to the fish-eye extension [130], which sends link updates more frequently to nearby nodes, and not that frequently to far away nodes. Both BMX6 and Babel have flat responses, with BMX6 outperforming Babel.

In reference to measuring the stability, we have generated random network topologies using the community network generator presented in [41] with the parameters from the Osona zone in the Guifi.net [36] community network (degree=2.99, shape $\alpha = 0.2521602$, and rate



(a) Convergence vs. hops (b) Ping success vs. changes (c) Offline period vs. changes

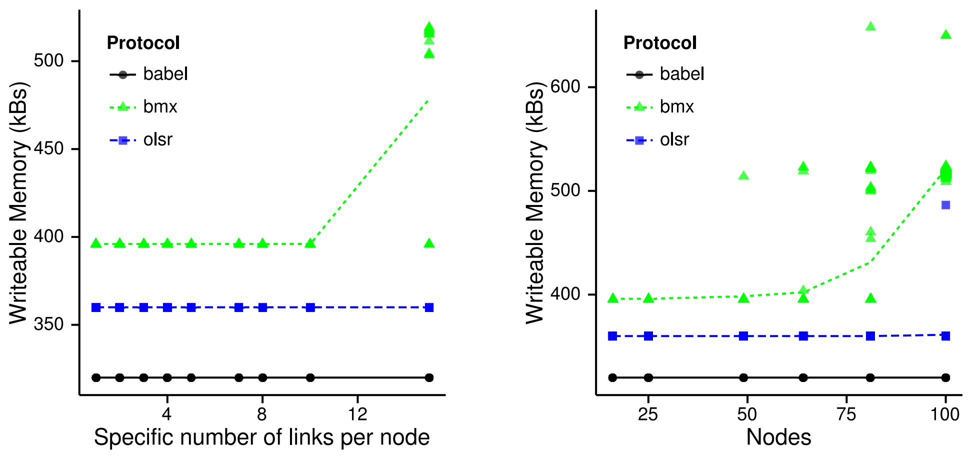
Figure 3.5: Network stability and reactivity

$\beta = 0.01147359$). Then links of the network were turned on and off with different times between changes but always ensuring that the network remained connected. Figure 3.5b illustrates the success rate of a ping between two randomly selected nodes of the network, depending on the number of changes per second. As expected, more changes imply that the routing protocol cannot keep up, and connectivity is lost sometimes. In this case, BMX6 outperforms both Babel and OLSR, which present similar results. Figure 3.5c presents the same results regarding which routing protocol has longer offline periods.

3.4.3 Memory Usage

The final metric studied through emulation is the cost in terms of memory. To measure the memory consumption, we have run the experiments on a computer with an Intel Core2 Duo E8400 processor running at 3.00 GHz with 4 GB of memory. We have retrieved statistics regarding memory usage using the *pmap* utility.

Our first experiment considers a network of 49 nodes and 74 links, and studies the memory consumed by each node based on the number of neighbors it has (Figure 3.6a). In this scenario, both OLSR and Babel show constant memory usage, independently of the number of neighbors; however, BMX6 requires more memory when the number of neighbors reaches 15. The results are similar in our next experiment (Figure 3.6b), which increases the size of the network (e.g., network with 16 nodes and 24 links, then 25 nodes and 38 links, etc.) and measures the average memory used by each node. Babel memory usage is stable, BMX6 increases with size and for OLSR, we observe an increase in one case for the biggest network. We believe that, for bigger networks, we would also see an increase in the memory use for OLSR and Babel, but this is not seen because memory is assigned in chunks, and, in contrast to BMX6, the OLSR and Babel protocols do not yet need the full memory provided by the current chunk. The assignment of memory in chunks also explains why memory usage of BMX6 seems non-linear. The measured usage remains unchanged until the last allocated memory chunk is exhausted and the probability for allocating the next chunk rises quickly and is non-linear in sections. The measured BMX6 memory usage in Figure 3.6a for 10 and 15 neighbors indicates such a section. In Figure 3.6b, the exponential growth in memory requirements for BMX6 is also caused by an increasing number of links that come with an increasing number of nodes.



(a) Memory vs. number of links (b) Memory vs. number of nodes

Figure 3.6: Memory consumption

3.5 Testbed Experiments

In contrast to emulation, a testbed-based experimentation allows us, at the cost of limited flexibility for considering CN-typical topologies, to study the impact of real embedded node hardware and wireless channel characteristics in an isolated environment. This way we aim to complement our picture of the protocol-respective overhead and efficiency (in terms of stability and reactivity) when processing capabilities of nodes are limited by hard CPU and memory boundaries and transmissions are subject to loss, delay, and jitter caused by interference among nodes.

3.5.1 Experiment Setup

Protocol performance measurements based on real hardware have been performed in the W-ILab.T wireless testbed [136]. The facility consists of approximately 60 stationary and 15 mobile experimentation devices deployed in a 60 x 20 meter indoor location at iMinds. The grid-like deployment structure of stationary devices is principally given by six rows with 10 devices (columns) each and an inter row space of 3.6 meters and inter column space of six meters. The devices consist of of-the-shelf computer hardware, each equipped with two 802.11abgn WLAN cards, which can be freely programmed by the experimenters. Further characteristics and configurations are summarized in Table 3.3.

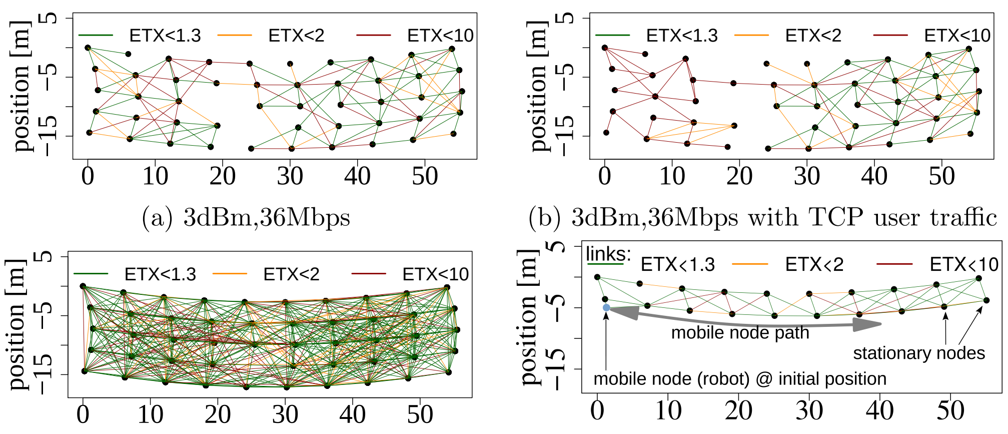
The devices, also called nodes, were configured to run a Linux operating system (OS) based on OpenWRT, a Linux distribution optimized for embedded wireless devices, and several convenient OS and measurement tools to control and collect measurement data. For the experiment, 50 nodes (the upper five node rows) have been used with one radio each configured in IEEE 802.11a ad-hoc mode. All protocols were configured to run on IPv6 and to announce only their primary interface addresses. Given the physically dense node deployment with an average neighbor distance of less than five meters, a number of preliminary measurements were performed to understand the wireless characteristics of the testbed and to avoid a fully connected mesh where the broadcast-based link detection

Table 3.3: General W-ILab.T testbed and protocol characteristics and configuration

Characteristic	Configuration
Environment	Laboratory 16x60 meter
Deployment	Regular 5x10 nodes grid (see Fig. 3.7a)
Operating system	Linux/OpenWRT BarrierBreaker rev41558
Protocol impl.	babeld v1.5.0, bmx6 rev8b0585e8, olsrd v0.6.6.2
Hardware	ZOTAC NM10-ITX
CPU model	Intel(R) Atom(TM) CPU D525 @ 1.80GHz
Memory	903460 kB
Wireless	Atheros - AR928X 802.11a/b/g/n
Wireless mode	80211a, ad-hoc, channel 36 (5.18GHz)

Table 3.4: Experimented parameters, defaults, and ranges

Parameter	Default	Range
Transmit power [dBm]	3	3, 4, 5, 6, 7, 8
Transmit rate [Mbit]	36	6, 9, 12, 18, 24, 36, 48, 54
Used nodes	50	10, 20, 30, 40, 50
Network Load	active	none versus active (single TCP stream between most distant nodes of a row)



(a) 3dBm,36Mbps

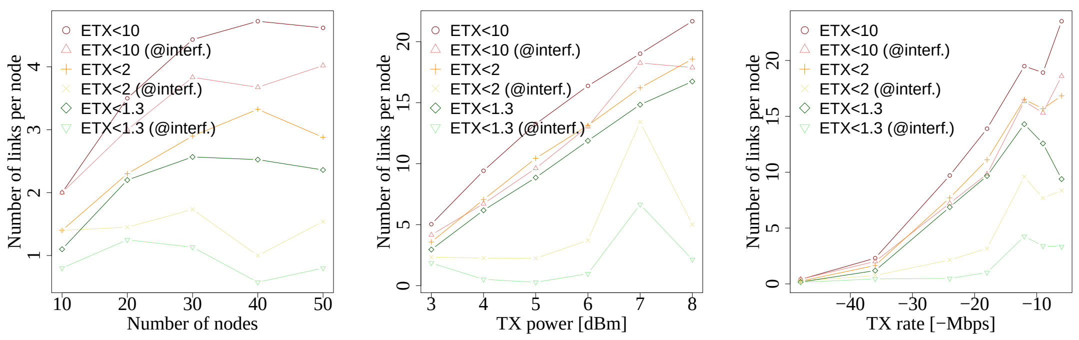
(b) 3dBm,36Mbps with TCP user traffic

(c) 8dBm,36Mbps

(d) mobile setup @4dBm,36Mbps

Figure 3.7: Topologies in different scenarios

mechanisms of the routing protocols detect links to nearly all other nodes (note that even the two most distant nodes are less than 63 meters away from each other) and achieve the establishment of true multi-hop topologies. The result of this exercise can be seen in Figure 3.7, showing topology snapshots (as detected by the OLSR protocol) resulting from different transmit power configurations with and without background traffic. The representation of node locations that are physically placed in a strict grid structure starts with the first node in the top left at (x, y)-position $(0, 0)$ and has been bended to allow the illustration of many otherwise hidden links. E.g. the weaker (yellow) direct link between nodes at positions $(0, 0)$ and $(0, -15)$ in Figure 3.7c would have been overdrawn by the four stronger (green) and shorter links between nodes at positions $(0, 0)$, $(0, -3.75)$, $(0, -7.5)$, $(0, -11.25)$, $(0, -15)$.

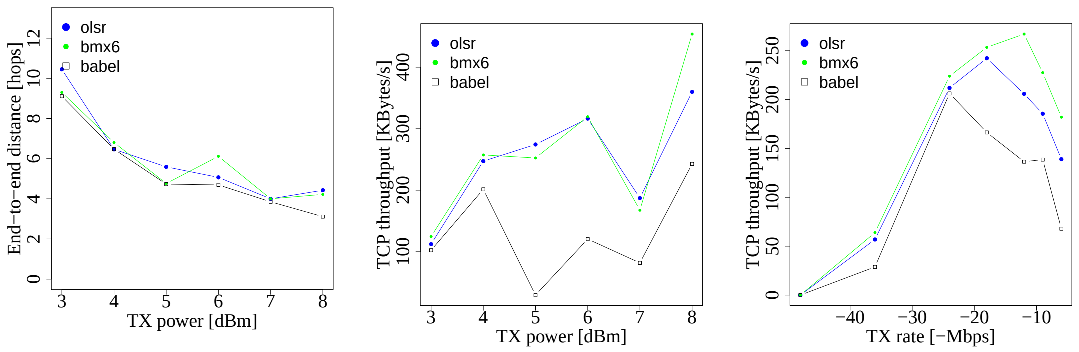


(a) Links depending on number of nodes (3dBm,36Mbit)

(b) Links depending on TX power (36Mbit)

(c) Links depending on TX rate (3dBm)

Figure 3.8: Network densities and link qualities for different testbed configurations



(a) Path length vs. TX power (36 Mbps)

(b) Throughput vs. TX power

(c) Throughput vs. TX rate

Figure 3.9: End-to-end path length and TCP throughput depending on network density varying power and rate

The average number of links per node is shown in Figure 3.8 for the number of nodes, transmit (TX) power, and TX rate grouped by their link quality and whether the links were captured with or without interference caused by TCP background traffic. It can be seen that the presence of TCP user traffic significantly decreases the perception of high-quality links (here those with an ETX rate of less than 1.3), while the total number of detected links is much less affected. The figure also indicate that, in our testbed scenario, the average number of links per node increase linearly with the selected transmit power. Figure 3.9a demonstrates the path length established by each routing protocol to route from the leftmost node in the second upper row (node Id 0) to the rightmost node in the same row (node Id 9). As expected, with a low transmit power (or high TX rate) and consequently small transmit range the selected end-to-end path relies on many intermediate hops, while less relaying nodes are required with an increased TX power (or a more robust but lower rate). Figures 3.9b and 3.9c give an impression on the end-to-end throughput achievable with each routing protocol when varying the node density by increasing power or rate.

Based on these findings, our following experiments will be configured to use the parameters shown in Table 3.4. Primarily, we use 3 dBm of transmit power and disabled transmit rates below 36 Mbps to enforce the establishment of topologies with more than seven hops.

3.5.2 Measurements

In order to measure the impact of neighbor size (density) and network size on cost and performance, all protocols have been sequentially exposed to a variety of testbed configurations. These testbed configurations were given by the range of parameterization values for each studied parameter and the default value for all other parameters. Each exposure (experiment) consists of the following standard procedure. First, all currently active protocols were disabled on all nodes. Then, the interfaces, IPv6 addresses, wireless settings (channel, mode, TX power, and enabled rates), and the currently probed protocol (only one at a time) were configured and activated only on those nodes relevant for this test. A stabilization period of 100 seconds was applied before continuing the actual measurement to avoid capturing of atypical bootstrapping effects. Each following measurement lasted 60 seconds and relied on common Linux tools (such as ping6, iperf, top, and tcpdump) for active probing of end-to-end path characteristics and monitoring and capturing CPU, memory, and traffic overhead.

After each experiment, the measurement data were offloaded and, once all experiments of a particular scenario were executed, post processed into graphs illustrating the dependency of one characteristic depending on a particular testbed parameter. Each measurement point in any of the graphs shown in Figures 3.8 to 3.15 represents the averaged relevant measurement data captured during a single experiment run.

The very limited exclusive testbed usage slots that allow interference-free experiments in the often overbooked W-ILab.T infrastructure do not allow the systematic repetition of scenarios required for a statistical validity analysis. Instead, the goal has been covering a wide range of selected parameters. However, atypical measurements have been selectively repeated to avoid the consideration of exceptional outliers. This way, the resulting and purposely unsmoothed plots also include strongly varying behavior that is a typical characteristic of any real wireless network. Still, a number of protocol-typical characteristics and tendencies can be identified and generally acknowledge the findings made via our previous emulation-based measurements.

The results in Figures 3.10a, 3.11a, 3.13a, and 3.14a illustrate the impact of network size on overhead, CPU, and memory consumption and are based on the experiments of a single scenario where nodes, grouped in rows with 10 nodes each, were successively added to the total number of participating nodes, starting with the second row (which also contained source and destination node used for end-to-end path probing and optional transmission of TCP user traffic), then adding the first, third, fourth, and fifth rows until the final size of 50 nodes was reached, eventually yielding a topology as illustrated in Figure 3.7a. During this scenario, the power and minimum rate was fixed to 3 dBm and 36 Mbps.

The impact of node density has been studied by either varying the transmit power of each node (with results shown in Figures 3.10b, 3.11b, 3.13b, and 3.14b) or by varying the minimal allowed transmit rate per node (see Figures 3.10c, 3.11c, 3.13c, and 3.14c). For these scenarios, all 50 nodes were used from the beginning, but transmit power or rate was successively changed in each experiment round.

The repetition of the above scenarios without background user traffic revealed that CPU and memory consumption do not significantly differ in both cases: thus, only the resulting impact on protocol overhead is shown in Figure 3.12.

In order to obtain an experimentation-based picture of the self-healing capabilities of each protocol, we probed the end-to-end path between two nodes. This was realized using a 20-node subset (given by the upper two rows) of the original 50-node topology and an additional "mobile" node installed on a robot that was programmed to move with different speeds just below the second row of nodes in the W-ILab.T deployment from near the leftmost node column to the eighth column and back. Figure 3.7d illustrates this setup. The robot turning points were located 42 meters (or seven inter-column spaces) apart from each other. Path health to this moving node was probed using the ping6 command from the fourth node of the second row, located in the middle between the turning points. Figure 3.15a shows, based on a TX power and rate setting of 3 dBm and 36 Mbps, that for each protocol the average ping success rate to this moving node depending on its velocity, each reflects a scenario with differently fast changing links. It can be seen how the success rate, around 90% at velocities of 5 cm per second for all protocols, decreases with increasing destination velocities down to around 60% for BMX6 and below 40% for OLSR and Babel. The repetition of this scenario with slightly increased power settings at 4 and 5 dBm (see Figure 3.15b and 3.15c) lead to a broader continuous coverage of the moving node, giving routing protocols more time to adapt to weakening links and narrow the performance gap between the three protocols.

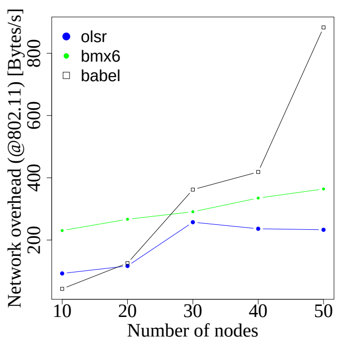
3.6 Discussion of Results and Conclusion

In this section, we take a closer look at the measurement results obtained via emulation and experimentation with the objective to gain a general understanding of how the characteristics of a network influence the performance of various routing protocols. We also discuss potential discrepancies of the two different methodologies. Table 3.5 provides a high-level summary of the identified protocol-specific behaviors by grouping the studied performance characteristics (in terms of overhead and self-healing capability) and dependencies (in terms of density, size and topology dynamics) into rows with a few comparative words for each protocol.

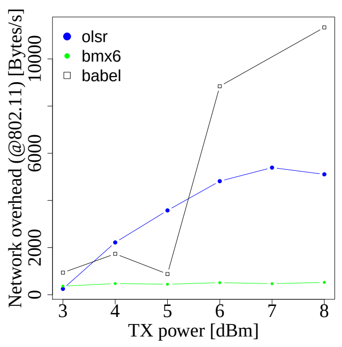
3.6.1 Protocol Data Overhead

Protocol data overhead has been measured in bytes and packets per second and depending on network size, density, and dynamics.

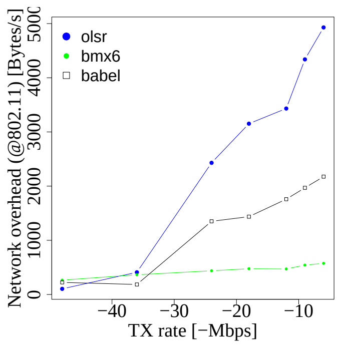
Regarding **byte overhead depending on network size**, both, emulation- and experimentation-based measurements (Figures 3.12a and 3.3a) show (apart from one exception in experimental Babel measurements, which we will discuss later) consistent results of an essentially linear increase with a protocol specific slope and base load. In addition, BMX6 shows the highest base load but lowest slope, while Babel shows the lowest base load and a slightly greater slope and OLSR shows the greatest slope which, given the emulation-based results, raises up to 400 bps, about 120% more than Babel and 50% more than BMX6 for a network of 70 nodes. The different absolute numbers between experimentation- and measurement-based results can be explained by the greater average number of links (neighbors) per node in the different scenarios, with three versus up to five (compare Figure 3.8a).



(a) Impact of Network size

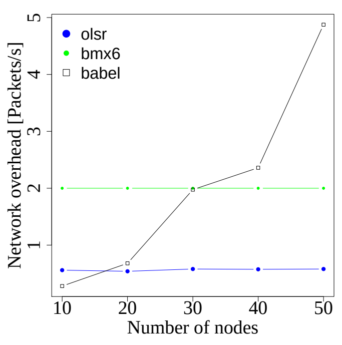


(b) Impact of TX power

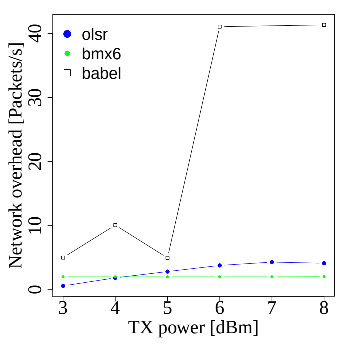


(c) Impact of TX rate

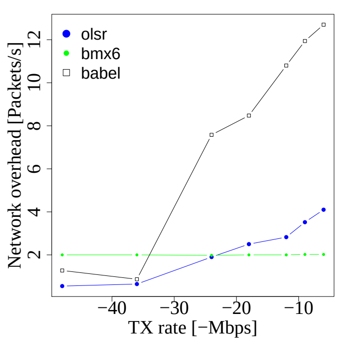
Figure 3.10: Data overhead (bytes/s) depending on network size and density (power, rate)



(a) Impact of Network size

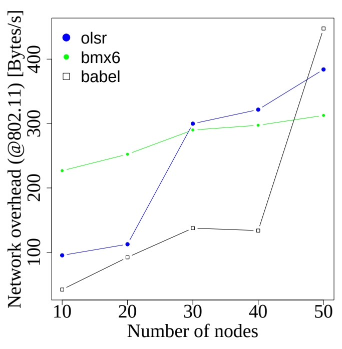


(b) Impact of TX power

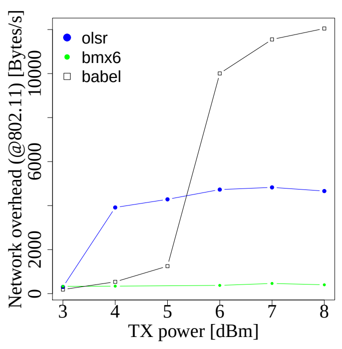


(c) Impact of TX rate

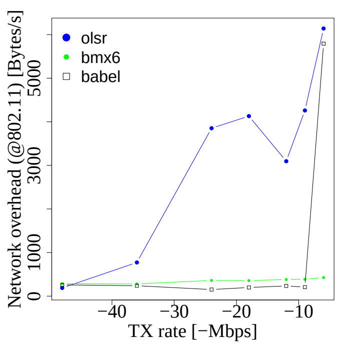
Figure 3.11: Data overhead (packets/s) depending on network size and density (power, rate)



(a) Impact of Network size

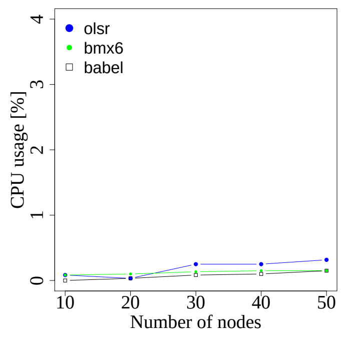


(b) Impact of TX power

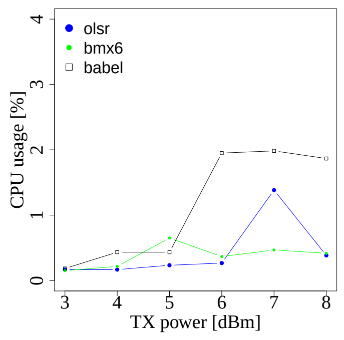


(c) Impact of TX rate

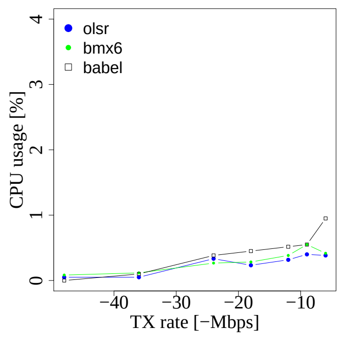
Figure 3.12: Data overhead (bytes/s) depending on network size and density (power, rate), no TCP user traffic



(a) Impact of Network size

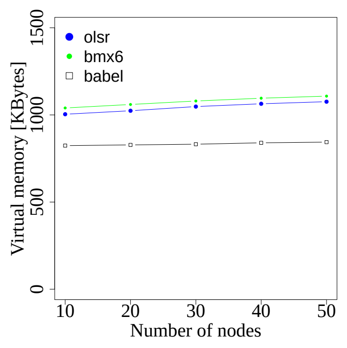


(b) Impact of TX power

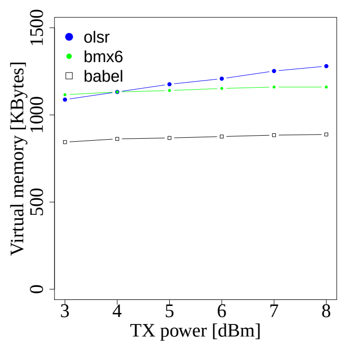


(c) Impact of TX rate

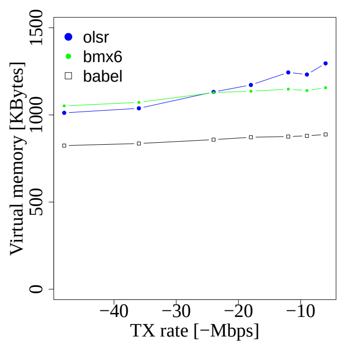
Figure 3.13: CPU consumption depending on network size and density (power, rate)



(a) Impact of Network size

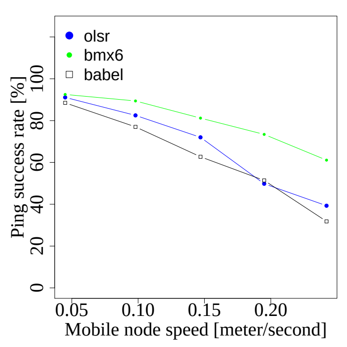


(b) Impact of TX power

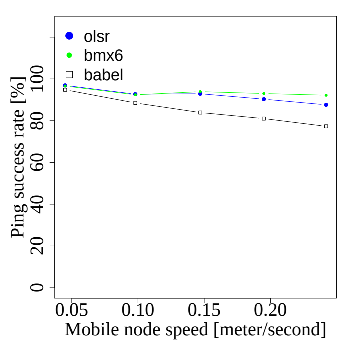


(c) Impact of TX rate

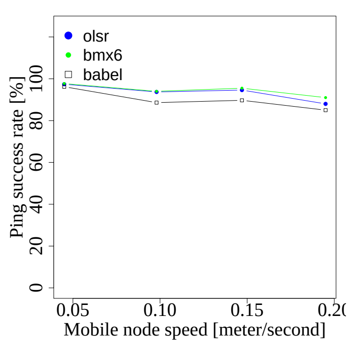
Figure 3.14: Memory consumption depending on network size and density (power, rate)



(a) Poor link redundancy (TX power 3dBm)



(b) Medium link redundancy (TX power 4dBm)



(c) High link redundancy (TX power 5dBm)

Figure 3.15: End-to-end delivery success depending on topology dynamics and link redundancy

A different picture arises when experimentally comparing the byte **overhead in the presence of TCP user traffic** as shown in Figure 3.10a. Then, transmissions naturally cause interference and thereby affect the perception of link qualities between neighbors (Figure 3.8a) as well as the propagation of routing information. In this scenario, we see that the overhead of Babel increases dramatically because Babel reacts to topology changes by sending unscheduled route updates. On the other hand, BMX6 and OLSR propagate routing updates periodically, independently of topology changes; therefore, this results in little effect on the overhead.

The slight increase in BMX6 can be explained by the requirement of acknowledgements when exchanging routing updates, which will cause overhead due to the retransmissions caused by traffic collisions. The opposite is the case for OLSR where the collision of link-information (TC messages) containing packets, if not successfully received via alternative links, are not further propagated and eventually result in an decreased overall overhead, an effect particularly likely in sparse networks with low link redundancy.

Another critical factor affecting protocol overhead is given by the **network density** where emulation- and experimentation-based measurements show quite different results. Looking at the former (Figure 3.3c), the observed shape of all protocols well matches with what one could expect from each protocol-dissemination algorithms. The highest, quite linear, slope for OLSR represents that of a non-optimized link-state protocol where information about all links in the network are propagated to all nodes for calculating a local view of the total topology. For this purpose, every node contributes to the propagation of this information by re-broadcasting new link state information once (via TC messages) and thus causing respectively increasing transmission overhead by each node. This non-optimized link-state behavior could be explained by the non-standard behavior of the OLSR implementation using a default MPR selector set of seven (instead of one, see also Section 3.2.3). However, compared to the experimentation-based results in Figure 3.10b, in the beginning the greatly increasing OLSR overhead quickly flattens for densities of around 10 or more links per node (corresponding to a TX power of 5dBm according Figure 3.8b), an effect which indicates that the OLSR-implementation specific MPR selector set value of seven still yields significant optimization in very dense networks.

The moderate overhead slope for BMX6 and the constant seeming slope for Babel from the emulation-based results correspond with the typical characteristics of distance-vector routing protocols, where link-quality information is only exchanged between neighboring nodes and thus affects each nodes' overhead only by a few additional link-probing related messages and as far as the size of its local neighborhood increases but not beyond. Here the aggregation of many messages into much less eventually broadcasted packets help to even flatten the observable overhead on layer 2.

Interestingly, a significantly different picture could be observed for the Babel overhead when considering the measurement-based experiments where the amount of transmitted data increases by a factor of 10 when the node density (at a power level of 6 dBm) exceeds 15 links per node. We attribute this behavior to the high susceptibility of the Babel protocol to topology dynamics, which was already observed for the impact of interfering TCP traffic in Figure 3.10a. In this case, such topology dynamics are caused by natural interference and consequent collisions due to the dense wireless deployments, a factor not existing in the emulation-based analysis. In fact, protocol performance instabilities, which are likely related to similar topology dynamics, were measured repeatedly in different scenarios. The exceptional experimental Babel measurements points mentioned earlier for byte overhead depending on network size are one example.

The measurement results for network **overhead in terms of packets** in Figures 3.4 and 3.11 illustrate on the one hand the dominance of protocol-specific link-probing and update intervals causing a constant and minimal packet transmission rate even in the simplest possible deployment. On the other hand, once protocol-stress factors (such as network size, density, or dynamics) cause a protocol to disseminate more data than could be aggregated into the packets sent at a minimal transmission rate, they show how packet overhead first increases in steps before scaling linearly with the byte overhead discussed earlier. In this sense, the high base rate of BMX6, at two packets/second compared to 0.5 packets/second for OLSR and even less for Babel, should only be considered relevant for deployments of rather stable and sparse networks. For more complex deployments the initial lowest packet rate of Babel can easily turn into a rate several times higher rate than that of OLSR and BMX6.

3.6.2 CPU and Memory Consumption

Experimentation-based measurements show (see Figures 3.13 and 3.14) that network size and density only have a very limited and generally non-critical impact on the CPU and memory consumption caused by the respective routing-protocol process. Given the embedded hardware and studied parameter space of up to 50 nodes and densities above 20 links per node, protocol-specific CPU usage always remained below 2% of the total CPU processing capacity, and virtual memory consumption (including all shared library objects mapped into the process) remained significantly below 1.5 MByte while showing only a very low increase over the studied ranges. Given these experimentation-based measurements, the memory consumption of Babel is the lowest and least increasing, while OLSR and BMX6 both show a very similar low linear increase depending on network size. Regarding density, BMX6 demonstrates a similar low slope as Babel, which matches what can be expected from any distance-vector protocol that only has to maintain the next hop towards any distant node. However, the link-state based OLSR protocol, which must keep track of all relevant links in the overall network topology, shows only a slightly greater increase of memory usage depending on density.

In contrast, the emulation-based measurements in Figure 3.6 depict the writable memory requirements (instead of virtual) of OLSR and Babel as completely unaffected by network size and density and below those of BMX6, which also shows a more spread requirement of memory for networks with more than 50 nodes or densities with more than 14 links per node. Nonetheless, the total memory requirements of all protocols remain non-critically low, given memory provisioning, even of resource-constrained embedded devices.

3.6.3 Self-healing Performance

A number of cases have been studied based on emulation or experimentation to characterize the capabilities of the different protocols to react to topology changes in different scenarios.

Emulation-based results studying the average time needed by each protocol to fix an end-to-end path depending on its length (Figure 3.5a) show that the two distance-vector based protocols outperform the link-state based OLSR, particularly in end-to-end scenarios with many intermediate hops.

While the poor performance of OLSR could be explained by the implementation of fish-eye optimisation, the better convergence time of BMX6 compared to Babel is surprising given the reactive nature of the Babel algorithm that should encounter spontaneously detected topology changes on demand instead of delaying the propagation of corresponding routing updates for the next update period. However, the better performance of BMX6 regarding topology dynamics is consistently confirmed in all further measurements (emulation- or experimentation-based, such as shown in Figures 3.5b, 3.5c, and 3.15) and must be attributed to the prevailing of other protocol characteristics. One reason is certainly given by the different link-probing and route update intervals (see Table 3.1) used by each protocol that allow BMX6 (using a Hello interval of only 0.5 seconds) to detect and react to local topology changes much faster than Babel and OLSR. On the other hand, to enhance the self-healing performance of a protocol, such intervals cannot be decreased without introducing additional protocol overhead, which is already significantly higher for the latter two protocols in large and dense networks and which, if further increased, would also lead to further protocol instabilities due to self-caused collisions and interference.

Table 3.5: Summary of observed and interpreted performance characteristics from Section 3.6

Characteristic	OLSR	BMX6	Babel
Increase of protocol overhead depending on size, density, and dynamics			
Size:	high linear	low linear	moderate linear
Density (low density):	high linear	low linear	lowest
Density (high wireless density):	logarithmic	low linear	in non-linear steps
Topology dynamics due to interference from TCP user traffic:	negative	unimpaired	highly susceptible with typical strong growth
Increase of memory usage depending on size and density			
Size:	low linear	low linear	lowest, unaffected
Density:	linear, acceptable	low linear	low, unaffected
Comment: Non-critical given the studied range of size, density, and dynamics			
Increase of CPU usage depending on size and density			
Size:	low, total max $<$ 0.5%	low linear, total max $<$ 0.2%	low linear, total max $<$ 0.2%
Density:	varying, total max $<$ 1.5%	varying, total max $<$ 1%	varying, total max $<$ 2%
Comment: Non-critical given the studied range of size, density, and dynamics			
End-to-end path-healing performance due to topology changes			
Off time versus path length:	high, increasing slope with jump between 7 and 9 hops, $avg \sim 35s$	low, unaffected, $avg \sim 8s$	medium, unaffected, $avg \sim 16s$
Outage versus changing rate:	BMX6 shows least outage in highly changing environments All protocols equally good at low changing rates		

3.6.4 Conclusion

This section presents an evaluation of mesh routing protocols for wireless community networks. We study the scalability, performance, and stability of BMX6, OLSR and Babel, three proactive routing protocols commonly used in these networks, through emulation and experimentation.

Our emulation and testbed-based experiments with various network conditions at different scales have provided several detailed results that compare the three protocols. In summary, we can say that Babel is the most lightweight protocol with the least memory, CPU, and control-traffic requirements as long as it is used in networks with stable links and low node densities.

However, if the protocol is used in large or dense wireless deployments with frequent link changes due to dynamic interference or nodes leaving or joining the network, then its reactive mechanisms to encounter topology changes by sending additional routing updates and route request messages turn into massive control-traffic and processing overhead. In such scenarios, OLSR and BMX6, with their strictly constant rate for sending topology and routing update messages, outperform Babel in terms of overhead, stability, and even self-healing capabilities.

The OLSR protocol significantly benefits from the MPR mechanism that (despite the highly redundant parametrization used within our experiments) achieves only a logarithmically increasing overhead depending on network density.

The BMX6 protocol benefits from its generally low control overhead due to the usage of compact local identifiers and the hiding of local state (e.g. link qualities) from globally propagated information. It differentiates from OLSR with higher memory requirements but lower control overhead and a better reaction on dynamic link changes.

Chapter 4

Enabling Individually Entrusted Routing

In this chapter we describe our efforts for the development of routing-protocol mechanisms than can enhance the autonomy and security of CN users, integrate and scale with the structural and technological implications given by existing deployments, and support the general commitment of communities for providing open, neutral, and decentralized network infrastructures.

Our results show the feasibility and performance of adversary-free routing, based on the user-individual decisions on the trustiness of components from the overall CN infrastructure while ensuring neutral recognition and support of even oppositional trustability perspectives of the participants.

4.1 Introduction

The operation of community mesh networks is based on the principle of cooperation among its members. These communities usually have participation rules as a membership license or peering agreement [19, 25, 26], that define their freedom, openness and neutrality. Nonetheless, current designs and implementations of mesh networks impose comprehensive technical definitions and restrictions to achieve functional data transit and end-to-end delivery among any pair of network nodes [22]. That includes the use of a specific routing protocol and routing metric so that nodes can consistently learn and inform about the state of the network and update their own routing tables. In practice, due to the lack of mature implementations, only a very few of the proposed routing protocols are used in real deployments or have been experimentally analyzed [8, 23, 52, 137]. Among them there are the AODV [138, 139], Babel [101, 102], BMX6 [38], the widely used OLSR [98–100], and batman-adv [140] protocol implementations.

One shortcoming of current solutions is given by the lack of routing-security support that comes without introducing centralized dependencies (e.g. certificate authorities) [141], which would contradict with the open and the decentralized objectives of such networks. Another problem lies in the protocol requirements for unified parametrization of metrics and policies to determine QoS, routing, trust and security decisions for all network nodes [4]. Such a strong level of unification prohibits the usage of individually defined policies and limits its openness. It also imposes a substantial effort, increasing with the number and diversity of community members, for finding consensus on related questions.

To dilute the limitations of a single and unified set of QoS parameters for routing, QoS Multi-Topology (MT) routing has been proposed [142, 143], allowing the concurrent support of multiple virtual topologies (on top of a single physical topology), each established based on a different definition of QoS parameters. This approach could also be adapted to concurrently support different security and trust sets. A network could for example maintain one topology (a) for nodes trusted by organization A, and a second topology (b) usable only by nodes certified via organization B.

The security design of the protocol proposed in this work ensures that each node is the only authority able to define and publish its set of individually-trusted nodes via which forwarding rules (routes) for delivering its traffic should be selected, propagated, and maintained. This way, our protocol pursues the multi-topology approach as it establishes dedicated virtual topologies for each participating node. It further supports the cooperative, open, and decentralized philosophy that enables community networking, as deployed network infrastructures remain open for other nodes to join and be used while being independent from any central entity.

In summary, the main contributions of this work are:

- Propose a novel secured and decentralized routing protocol called SEMTOR. In SEMTOR users can set up trust sets of nodes, which are the only ones allowed to route their traffic.

- Describe how SEMTOR is implemented as an extension of BMX6, a routing protocol currently used in production community wireless mesh networks.

- Summarize the assumptions, and respectively achieved safety and liveness properties provided by SEMTOR, and prove their correctness with formal reasoning.

- Experimental validation of the resistance of SEMTOR to attack scenarios and challenging network environments.

- Analysis of the hardware requirements of SEMTOR by investigating its performance in terms of traffic, CPU, and memory overhead. Our results show that SEMTOR can be deployed using off-the-shelf inexpensive WiFi routers.

The remainder of this chapter is organized as follows. After looking at related work in Section 4.2, we identify the addressed problems and design objectives in Section 4.3 and detail the system model with further assumptions and definitions in Section 4.4. Section 4.5 describes the design and mechanisms of our protocol to solve these objectives which is then validated from an formal and an experimental perspective in Section 4.6.3 and 4.7. This includes the presentation of our prototypical implementation and its functional and performance evaluation using embedded router hardware in a virtualized network environment. We discuss the contributions, open issues, and adjacent security solution in Section 4.8 and conclude in Section 4.9.

4.2 Background and Related Work

Existing work on secure routing for ad hoc and mesh networks has been reviewed in [56, 144]. Authenticated routing for ad hoc networks (ARAN) [145] as proposed by Sanzgiri et al. as well as Admittance-control enabling extensions for OLSRv2 proposed by Herberg et al. [146] use digital signatures to verify the authenticity and integrity of control messages. Both rely on the existence of a central certificate server trusted by all participating nodes. Babel HMAC cryptographic authentication [147] relies on one or several pre-deployed shared keys to validate messages via attached message authentication codes. However, the requirement for preserving a shared keys as a private secret within an open network community disqualifies related approaches for any open CN.

SEAD [148] and SAODV [149] encounter the dependency on a central trust authority with a self-securing control plane. Using Anchored Hash Chains (AHCs) to protect the mutable hop counter field of routing update messages they ensure that a malicious node cannot claim better distances to any remote node than it really has. However, both remain vulnerable to data-plane attacks such as packet dropping or routing-table poisoning. SAODV also proposes the use of digital signatures to protect non-mutable data in routing messages. To avoid the dependency of a certification authority as a central root of trust that guarantees the binding between node public keys and other node properties such as their IP address Zapata [150] proposes to bind the identity of nodes given by their public key to their allocated address by building it based on the hash of the public key.

Work in [151] and [152] address the problem of misbehaving nodes by punishing malicious nodes based on their forwarding behavior as observed and assessed by neighboring nodes. Adnane et al. [151] build on top of SOLSR and extend it with detection and reaction mechanisms. Mogre et al. [152] present another holistic approach combining self-securing routing, detection, reputation, and counter-measure mechanisms.

SEMTOR follows a different approach. In fact, guaranteeing in all aspects the correct operation of nodes is indeed hard and, as pointed out by Adnane et al. [151], cannot be guaranteed (e.g. data-plane attacks cannot be prevented) by securing the topological information exchanged between nodes. Therefore, instead of aiming to ensure or enforce correct operation, SEMTOR enables each node admin to freely define their individual subset (and resulting sub-topology) from the whole set of participating nodes that he considers sufficiently trustworthy to meet their security and data-delivery objectives and concerns. In addition, none of the yet presented work relying on asymmetric cryptography for verification of control messages has yet been analyzed in terms of performance and benchmarked based on real embedded hardware and exposed to traffic and network characteristics that are typical for existing community mesh network clouds.

An impressive amount of further related research about wireless mesh networks has been done in recent years. In the following, selected publications are ordered thematically according to the aspects considered of particular importance for the objectives of this work. The case of community mesh networks is discussed in terms or legal implications, motivation, design, and business models in [21, 26, 69–71, 141, 153]. In addition, scalability and performance aspects of routing protocols are hanlded in [1, 48–52, 61]. Trust and security related work is surveyed and discussed in [53, 56, 154–157], with solutions for particular routing functions in [148, 151, 158–164], and presentations of holistic security frameworks in [152, 165, 166]. The last four also present measurement results based on simulation. Approaches towards supporting different or user-defined routing policies are handled in [4, 54]. Traffic validation, or how to recognize a misbehaving path, node or link and which information is needed, is addressed by sketches [167–170], counters [171], fingerprinting [172] or sampling [168]. Distributed detection, the assessment of anomalous and faulty nodes based on sharing of distributed observations, considering the arbitrary behaviour of malicious nodes, is addressed by Π_2 and Π_{k+2} [173] in general, or by KDet [174] specifically for community networks. Frameworks and algorithms for self configuration, channel assignment, and resource allocation are presented in [175–177]. Testbeds, tools, and trends for experimentation and evaluation of mesh-network protocols are covered and reviewed in [11, 116, 178–183].

4.3 Problem Statement

The goal of SEMTOR is to provide secure mechanisms to ensure that **non-trusted nodes in an open network are effectively prevented from disrupting the routing between trusted nodes**. In multi-hop mesh networks, end-to-end routing is a distributed task that relies on the contribution of network resources by nodes and honest collaboration between them. However, in an open network, that allows new or unknown nodes to join without pre-conditions. Therefore malicious or misconfigured nodes with byzantine behavior can participate and interfere the collaboration with faulty operations or adverse contributions. As such, the routing system is one of the most complex and fragile, but unfortunately also one of the least protected component in a network infrastructure [155].

In such hostile environment, our mechanism shall ensure that routing-related tasks such as path detection, (efficient and consistent) route establishment, and end-to-end packet forwarding are robust against any faulty operations from non-trusted nodes.

With reference to the work of Mizrak et al [173], the problem of detecting and handling compromised nodes in a network can be split into the three sub problems of (i) Traffic validation: Behavior characterization based on local observation, (ii) Distributed detection: Assessment of anomalous and faulty nodes based on sharing of distributed observations, and (iii) Response: Enforcing the distributed exclusion of a given set of identified faulty nodes.

In that sense, our protocol only addresses the third sub problem of response. More precisely: Given a known subset of faulty or distrusted nodes (from the overall set of existing nodes) is known, then how to achieve that no subset of these nodes can negatively affect any routing-critical task. The other two sub problems of traffic validation and distributed detection are out of scope of this work, but there are complementary solutions suitable for this scenario with sketches [170] for the first problem, and KDet [174] for the second. Further, no assumptions are made on how a required subset of trusted nodes (or its relative complement from the set of overall nodes: the set of non-trusted nodes) is obtained. It is just given for granted as each community mesh can implement its own way.

4.3.1 Objectives

For the design of the protocol, the following four objectives have been identified and are briefly summarized as follows.

Regarding **ownership** of data traffic, forwarding routes, and node identities, the objective is that any community-network related routing-table entry and packet can be attributed to exactly one node of the network. Each node can be unambiguously identified with a secure identity and an identity-proving address. In addition, each node is the exclusive owner of exactly those routes pointing towards this address and those packets carrying this address as destination.

Regarding **security in terms of autonomy and robustness**, the general objective is that any contiguous group of nodes (node admins), trusting and willing to cooperate and support each other, can not be prevented by an external (e.g. an adversary) from doing so.

Regarding **openness and decentralization**, the objective is that each node admin can individually decide which of the other nodes he wants to trust and rely on, that multiple cooperative groups of nodes (admins) can coexist, group membership is not exclusive, and there is no need for a central registry or authority. Nonetheless, similar to social networking or so called networks of trust, public key-servers or other decentralized coordination platforms may be used in addition to facilitate the management of known and trustable nodes and corresponding node IDs.

Regarding **scalability**, an implementation of the proposed mechanisms should be feasible and scalable for the characteristics (e.g. number of nodes and links per node) of typical community-network clouds and given the resource limitations (e.g. CPU, memory, and bandwidth) of low-budged but state-of-the art embedded routers as used in today's community-network deployments.

4.4 System Model

4.4.1 Community Model

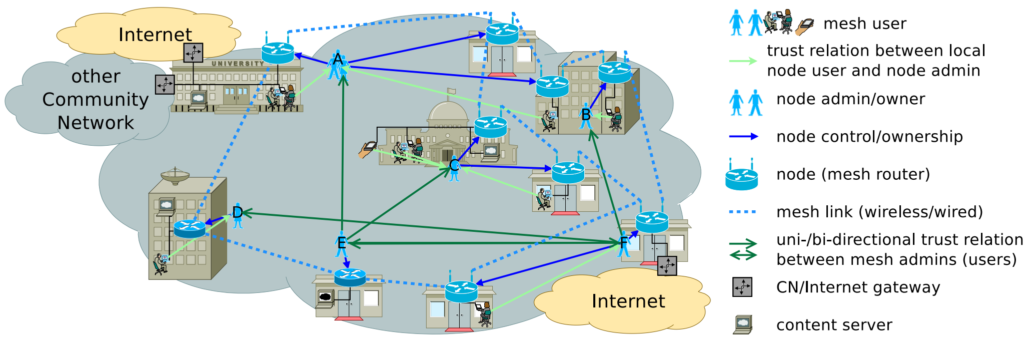


Figure 4.1: Model of a community mesh network

Figure 4.1 intends to illustrate the key scenarios, assumptions and characteristics that are consistent with real-world community mesh networks clouds. Mesh clouds incorporate up to thousands of users, hundreds of mesh nodes, and tens of links per node [3, 41, 42, 68, 184] and typcially extend from a smaall neighborhood to the coverage of a city or rural district. They interconnects via gateways with neighboring community clouds and other private or publich networks such as the Internet.

The technical enablers are given by **nodes** based on wireless or wired router hardware [23] with usually limited storage, bandwidth, and computation capabilities but that can be mounted at low cost on rooftops or even isolated rural locations powered by solar energy. The **links**, manually or automatically established between neighboring nodes, create a **restricted-route networks** without universal direct connectivity between all mesh nodes [27, 28].

As such, the overall infrastructure is owned, developed, and maintained by its **users**, united by the common idea to share their networking resources: Users are offering their own resources, in terms of routing hardware and processes and links, to others while being allowed to individually select from the overall pool of existing resources offered by others. Being able to select is desired because the usage and thereby implicated dependence, for example on the router of one particular admin, may, for various reasons, be disapproved by some of the users. For example because their nodes are notoriously error-prone. Therefore usage should not be imposed.

Users can be grouped into two stakeholder groups: users of the infrastructure; and **administrators** that take the additional responsibility to **control** and maintain individual **nodes** (mesh routers), and other locally connected infrastructure such as **content servers** or **border gateways** to neighboring networks.

Due to the different roles, a **local trust relation** exists between the directly connected users of a node and its representative, the node administrator who implements the local user decisions. It is expected that such local node communities are rather small (typically at the scale of a house or organization) so that consensus on commons such as a license or terms of usage (e.g. [19, 25, 26]) can be reached via direct communication. However, if consensus is not possible, an independent local node deployment, implementing a different usage policy, can be set up such as the case for the upper right building in Figure 4.1 which hosts two nodes operated by different administrators.

Another fundamental assumption for an open and decentralized CN that brings together individuals and organizations with different and even conflicting economic, political, and technical interests [21, 29, 30] is that the achievement of global consensus on the trustability of all other participating nodes and their administrators is actually impossible. Instead, it is expected that controversy may exist and trustability is only given among various subsets of the overall user groups. Such **directed trust relations between admins** are exemplary illustrated as green arrows for users of admin E and F. E users agreed on accepting the license and trusting in the administrative setup of A, C, and F, while admin F and its connected users trust in B, D, and E.

4.4.2 Network Model

The network model described in the following shall provide community-networks users (as modeled in previous Section 4.4.1) support for the specification of discretionary trust relationships between node admins.

A network *node* is defined as a (routing) process owning an individual public/private key pair to support cryptographic operations for authentication, integrity-verification, and non-repudiation of data created by the node. The private key is assumed to be permanent, globally unique, and accessible exclusively by the key-owning node. A node is supposed to correctly perform the tasks of path detection, route establishment, and forwarding of IP data-packets as specified by algorithms and implemented respectively by a specification-conforming *routing protocol*. A supposedly correct (benign) node may indeed not conform

Table 4.1: Summary of used symbols

Symbol	Description	Properties
$\mathbb{V}$	set of all nodes (correct or not)	
K_X^{pub}	public key of node X	
K_X^{priv}	private key of node X for K_X^{pub}	
$V_X = v(K_X^{pub})$	particular node X labeled as V_X	$V_X \in \mathbb{V}$
$A_X = a(V_X)$	IPv6 crypto-address (CGA) of V_X	
$tx(d)$	tx function broadcasting data d to all neighbors	
$rx(d)$	rx function receiving broadcasted data d from neighbor	
c^t, c^r	arbitrary transmitted or received message code, e.g. $c \in \{ping, pong, ...\}$	
I	interval between two points of time	
$S_{X,d} = s(K_X^{priv}, d)$	signature created by V_X with K_X^{priv} over data d	
$\overline{s}(K_X^{pub}, S, d)$	signature verification function	$\overline{s} : \{K, S, d\} \rightarrow \{false, true\}$
$D^c_{X,Q} = \{V_X, K_X^{pub}, Q^d_X, Q^a_{X,Q}, A_X, \mathbb{V}^t_X, \overline{M}_X, f_X\}$	unsigned description components of node V_X with sequence number $Q = Q^d_X$	
$D_{X,Q} = \{D^c_{X,Q}, s(K_X^{priv}, D^c_{X,Q})\}$	signed description of $D^c_{X,Q}$	
Q^d_X	description sequence number of D_{X,Q^d}	$Q^d \in \mathbb{N}^+$
$Q^a_{X,Q}$	AHC anchor of $D_{X,Q}$	
Q^b_X	AHC heartbeat matching $Q^a_{X,Q}$	
$Q^h_X = h(Q^a_{X,Q}, Q^b_X, V_X, Q^d_X)$	heartbeat sequence number of V_X	$Q^h \in \mathbb{N}^+$
$\mathbb{V}^c_X$	set of all correct nodes regarding V_X	
$\overline{\mathbb{V}^c_X}$	set of all adverse nodes regarding V_X	$\overline{\mathbb{V}^c_X} = \mathbb{V} \setminus \mathbb{V}^c_X$
$\mathbb{V}^t_X$	set of all nodes trusted by V_X	
$\overline{\mathbb{V}^t_X}$	set of all nodes not trusted by V_X	$\overline{\mathbb{V}^t_X} = \mathbb{V} \setminus \mathbb{V}^t_X$
$\mathbb{E}$	set of all links between all nodes	
$E_{B,A}$	particular directed link from V_A to V_B	$E_{B,A} \neq E_{A,B}$, $E_{B,A} \in \mathbb{E}$
$\mathbb{E}^t_X$	set of all links trusted by V_X	(see eq. 4.6)
$L_{B,A} = l(E_{B,A})$	directional link-metric quality for transmitting from A to B	$L \in \mathbb{R}^+$
$M_{T,S} = m(V_T, V_S, f_T, \hat{M}_T)$	path-metric quality from V_S to V_T depending on V_T-defined customizer function f and upper quality bound $\hat{M}_T$	$M \in \mathbb{R}^+$, $M \leq \hat{M}$
$G = g(\mathbb{V}, \mathbb{E})$	overall network topology graph	
$G^t_X = g(\mathbb{V}^t_X, \mathbb{E}^t_X)$	trusted topology graph of V_X	
$P_{T,S} = \{V_0, V_1, ..., V_n\}$	particular explicit path from V_S to V_T as ordered list of connected nodes $V \in \mathbb{V}$ from G with $V_0 \equiv V_S, V_n \equiv V_S$	
$\mathbb{P}_{T,S}$	set of all directed paths from V_S to V_T	
$\mathbb{P}^t_{T,S}$	set of paths trusted by V_T from V_S to V_T	$\mathbb{P}^t_{T,S} \subseteq \mathbb{P}_{T,S}$
$\mathbb{P}^c_{T,S}$	set of paths consisting only of correct nodes regarding V_T from V_S to V_T	$\mathbb{P}^c_{T,S} \subseteq \mathbb{P}_{T,S}$
$\mathbb{P}^{t,c}_{T,S}$	set of paths consisting only of trusted and correct nodes regarding V_T from V_S to V_T	$\mathbb{P}^{t,c}_{T,S} \subseteq \mathbb{P}^t_{T,S} \cap \mathbb{P}^c_{T,S}$
$R_{T,S} = r(V_T, V_S)$	established route from V_S to destination V_T along implicitly defined nodes	

with the specification of correct behavior. This may be due to accidental miss-configuration or *malicious* motivation of the person administering this node. The *public key* of a node X is denoted as K_X^{pub}. An identity function $V_X = v(K_X^{pub})$ is an injective function that assigns a (identity) label V_X to node X based on K_X^{pub}. A practical identity function $v(N)$ such as the secure SHA224 [185] hash function is assumed for this work. The set of all (correct and non-correct behaving) nodes is given as $\mathbb{V}$.

A *link* $E_{B,A}$ describes a directed connection that exists if a network interface of node V_B is in direct transmission range of a network interface of V_A and vice versa, providing a functioning bidirectional transmission opportunity between V_A and V_B. However, while the existence of $E_{B,A}$ implies the existence of $E_{A,B}$ it does not imply that these two links are symmetric or even identical: $E_{B,A} \neq E_{A,B}$.

Path detection denotes the task of identifying and maintaining, for a given network topology G and V_T, V_S-tuple, existing paths and path metrics so that identified (V_T, V_S)-paths could be differentiated and ranked. Depending on routing-protocol concepts, knowledge about path details could be distributed (such as in distance-vector protocols where only the next hop towards a given destination is known by each node) or complete (such as in link-state protocols where each node knows the complete topology).

Route establishment denotes the task of combining information from path detection with a path metric for setting up IP forwarding rules so that continuous and consistent forwarding paths establish between any tuple of source and destination nodes.

Forwarding denotes the task of forwarding and eventually delivering IP-data packets according to previously established routes from a source to a destination node.

The routing process relies on the following assumptions and service primitives provided by the data-link layer:

- The network connectivity over time is given by the topology graph G during interval I.
- Links of G either exist (are up) or not (are down) and their states do not change within one particular interval I_n.
- $tx(d)$: The transmit function broadcasts data d to all neighboring nodes to which a link from the broadcasting node exists.
- $rx(d)$: The receive function receives data d broadcasted by any node from which a link to the receiving node exists.
- The maximum transmission delay υ between any $tx(d)$ function call and corresponding $rx(d)$ events is much smaller than interval I: $\upsilon \ll I$

Routing processes exchange routing update messages containing information updates and announcements about nodes, links and network routes.

4.4.3 Adversary Model

The behavior of a network node V_Y regarding a particular other node V_X is either *correct* (benign) and complies with the protocol rules regarding V_X or it is *adverse* (malicious) and performs arbitrary behavior (byzantine) regarding V_X that, detectable or not, deviates from the protocol rules. The set of correct nodes regarding V_X is given as $\mathbb{V}^c_X$. The set of adverse nodes regarding V_X is given as $\overline{\mathbb{V}^c_X}$ with $\overline{\mathbb{V}^c_X} = \mathbb{V} \setminus \mathbb{V}^c_X$. No assumptions are made on the point of time (which may be any time in the future) and the extend of deviation (which may happen at the control or data plane), collaboration (allowing coordinated and distributed attack), or communication capabilities (using extraordinary and dedicated channels) of adversaries. Unless otherwise precluded, no assumptions are made on the methodologies and behavior applied by adversaries to achieve their objectives. The following assumptions apply to all nodes (including adversaries).

Assumption 1 *Nodes can not launch a worm-hole attack [186] without being detected and corresponding packets being discarded by the receiving node.*

This implies that it is impossible to replay entire packets without changing them so that they appear to a receiving nodes as being received directly from the node that originated the repeated packet in the first place. Such attack that can be addressed by lower protocol layers (Phy or MAC) via packet leashes [187] or NPAs [188], or by higher level distributed detection [189].

Assumption 2 *Nodes cannot tamper with cryptographic primitives.*

This means nodes can not attack fundamental cryptographic primitives and its key properties such as given by a secure hash function, the asymmetric encryption and decryption, signing and signature verification based on a public-private key pair, the Diffie-Hellmann secure key-exchange, and a the identity and ownership proving properties provided by Cryptographically Generated Addresss (CGAs). Nodes can not control other nodes beyond the specified protocol rules. In particular private key material is inaccessible for other nodes and other node services (e.g. ssh) are sufficiently secured.

Assumption 3 *Nodes cannot perform anonymous denial of service attacks.*

Nodes can not perform denial of service attack by exhausting processing capacities (in terms of memory or CPU) of neighboring nodes without being detected as the cause of the critical resource exhaustion and their packets being discarded. Network and traffic monitoring and detection can detect and mitigate these attacks.

Assumption 4 *Nodes can not physically disturb other nodes.*

This also means that wireless channels can not be disturbed via jamming and traffic incurred by concurrent transmissions sharing the same local channel environment have only a marginal impact on the remaining capacity of affected links.

4.5 Protocol Design

The basic idea to achieve our objectives may be best illustrated with a simple example: If person x declared that he fully trusts in persons a and b, then any person y, than knows about xs declaration, could hand out value for x to a or b while fully respecting x's assumptions of trust and knowing x as being in full charge of this action and any subsequent consequences. Further, these trust declarations should not be weakened by composition (intransitive). Retaking above example, intransitive means that even if a or b

assume (and declared) z as trusted, they must not hand out value for x to z unless also x explicitly declared z as trusted.

When applying this idea for routing in IP networks then a person is represented by a node which, being administrated by a person, uses a network protocol to communicate the declaration of its administrator.

SEMTOR extends the concepts of *receiver-driven routing* [4], which already allows nodes to express path-selection preferences via a descriptive profile. Both approaches employ principles of table-driven, proactive, destination-sequenced distant-vector (DSDV) routing [190] protocols where sequenced routing updates, originated by each node of a network and updated and re-broadcasted by each hop, are used to propagate path cost and versioning information in the network. In contrast to the traditional DSDV protocol, the routing update of SEMTOR contains a reference value, instead of a destination IP address or network, which unambiguously identifies a particular node of the network and a specific version of this node's self-defined description. This reference is given by the descHash, the SHA hash of the node's current description (also called **description reference**). Another unambiguous reference to a particular version of a node's current description is given by the the tuple of its public-key hash (*nodePKHash*) and description sequence number (*descSqn*) that are given in protocol packet headers and description messages. The relations between packet-header fields, descriptions, and other protocol messages and content as potentially contained in a protocol packet are illustrated in Figure 4.2 and is discussed in more detail in the following.

4.5.1 Global Node Identification and Description

A strong and permanent **node-public key** (nodePK) provides the root of trust for all operations performed on behalf of the node possessing the corresponding node-private key. The SHA hash of this public key (nodePKHash) is used as a **global ID**, an unambiguous and permanent reference for identifying the node.

The node **description** aggregates information about the node's current configuration and manifests at least the following information:

The **node-public key** (nodePK) and its **global ID** (nodePKHash).

A **description version** – sequence number (*descSqn*) that must be incremented with every **description update** or node reboot, allowing to always differentiate between the latest and out-of-date versions of any node's description. Sequence numbers of routing updates and descriptions do NOT wrap around. Instead, the exhaustion of the former requires the generation of a new description (update) after which routing-update SQNs are reset but also lead to a new descHash. The exhaustion of a *descSqn* indicates the end-of-life of a used nodePK.

A **description signature** (descSignature) matching the permanent node-public key given in this description and used for verifying the authenticity and integrity (i.e. this description was indeed created by the node with the corresponding global ID and was not changed by anyone else) of the description.

Instead of aggregating all description data directly into a single and self-contained description message, parts of the overall description are defined from the **root description** by

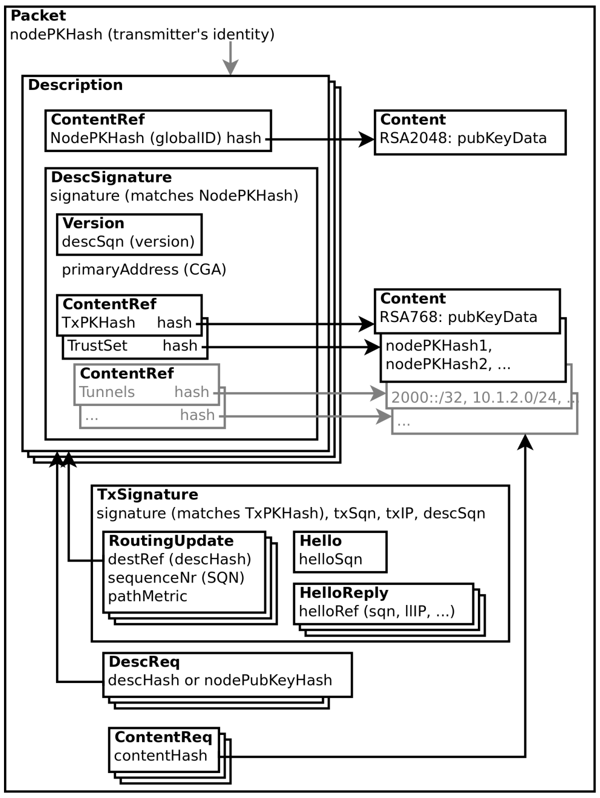


Figure 4.2: Protocol packet, description-, content-, and reference structure

the SHA hash of the actual data (similar to the description reference used within routing updates to identify a node and its description). Such references-based definitions are called **content references** and are used for example to define public-keys or the trust sets of a node.

The approach comes with the implication that any received packet header or message, containing description or content references for which the referenced raw data is not known, cannot be instantly processed and must be requested first. To resolve such situations, **description-** or **content-request** messages (containing only the unresolvable hash) are sent from the node not knowing the hash-matching raw data to the (neighboring) node from which the packet containing this hash was directly received. The node receiving a description or content request and knowing the requested data then responds with a corresponding **description-** or **data-advertisements** messages, providing the requested data.

4.5.2 Link Authentication

Neighbor and link authentication is used to (i) let neighboring nodes, with a direct communication channel between each other, detect and authenticate each other as the node they claim to be and (ii) to ensure authenticity, integrity, and non-repudiation of information exchanged between them (to avoid misunderstanding: link authentication can verify information X as being stated by neighbor A, but not that information X is necessarily true).

Link authentication is achieved by authenticating exchanged messages with signatures or with HMAC authentication codes. Since the authentication needs to be performed continuously and with a high frequency, the underneath cryptographic operation should be sufficiently lightweight to be accomplishable even by resource-constrained embedded routers.

4.5.2.1 RSA signature based

For signature based link authentication, a secondary and temporary RSA-based **transmission public key** (txPK) is added to the node description and used by neighboring nodes for (authenticity and integrity) verification of received link-discovery and routing-update messages signed and broadcasted by this node. The key can be updated periodically or sporadically (e.g. every hour) via description updates, allowing the originating node to use also RSA keys with weaker key strength and smaller crypto overhead (such as RSA896) than its permanent nodePK.

4.5.2.2 Diffie Hellmann, HMAC based

An alternative method for authenticating exchanged messages is based on keyed-hash message authentication codes (HMAC). HMAC essentially works as follows. The to-be-authenticated message is concatenated with the hash of a shared secret (known only to the two specific link neighbors interested in authenticating the messages exchanged among them). The hash of this concatenation is concatenated a second time with a differently padded version of the shared secret. The resulting hash is then sent as a message authentication code (MAC) together with the original message. The link neighbor verifies the received message and MAC by matching the latter against its self-calculated MAC using the received message and the shared secret as inputs.

A variation of Diffie-Hellman key exchange mechanism, tailored to the SEMTOR-protocol concepts, is used to let nodes establish the necessary shared secret over a public network. Therefore, each node (e.g. x) selects a private secret k_x (only known to itself) and publishes K_x, given as

$$K_x = g^{k_x} modp \tag{4.1}$$

via its description, using pre-defined protocol constants for the modulus p and the base g. This way, any pair of nodes, given that each others description is known, can calculate a mutual shared secret $S_{x,y}$ without additional handshake procedures as described by Garzia in [191]:

$$S_{x,y} = K_x^{k_y} modp = K_y^{k_x} modp \tag{4.2}$$

As with transmission public keys, K values are used only temporary and can be replaced via description updates at any time. Further, a shared-key exchange is neither explicitly announced nor is its completion explicitly acknowledged. A successful shared-key exchange is only implicitly confirmed by receiving messages with a valid HMAC based on a shared key calculated from the latest known descriptions of the two involved nodes.

4.5.3 Network Assets Authentication

4.5.3.1 Address, route, and traffic ownership

A cryptographically generated address (CGA) provides the basis for identifying and proving the owner of addresses, routes, and data packets.

The **primary IPv6 address** of a node provides such CGA. It is computed by combining a small (between 8–48 bits) well known ULA prefix, typically from IPv6 Unique Local Address (ULA) range [192], and the most-significant (at least 80 bits) of the node's permanent global ID (the nodePKHash). The concept of CGAs has been proposed by [193] which also describes mechanisms to calculate provable network ranges or further enhance the security of generated addresses and networks. Addresses or networks within this ULA prefix are owned by the node possessing the private key for the CGA mathing global ID.

Network announcements (published via node descriptions) are used for declaring the ownership of addresses and networks. All announcements overlapping with this ULA prefix must be provable as being owned by the node that originated the corresponding description. Otherwise the description must be considered as invalid and discarded upon reception.

Forwarding routes to a given CGA address of this ULA prefix, although installed in different routers, are owned by the node owning the destination address of the routing entry and must be maintained according to the rules specified via its description. Otherwise the route must be removed.

IP-data packets, containing a verified CGA address of this ULA prefix as destination address, are also owned by the node owning the address and must be forwarded according to the forwarding route configured to this address. Otherwise the packet must be dropped.

4.5.3.2 Routing and heartbeat updates

An **Anchored Hash Chain (AHC) value** substitutes the destination network and sequence number of routing updates from traditional destination-sequenced distance-vector protocols such as [190]. The approach has been inspired by the SEAD protocol [148] which uses an AHC to protect, in addition to the destination nodes sequence number, also the distance vector (metric) of routing updates.

In SEMTOR, the IP addresses and networks, towards which forwarding routes shall be maintained due to the reception of a routing update, are given by the network announcement published via the referenced description.

The creation, mapping, and validation of received AHC values has been designed as follows. Whenever a node creates a new description it picks two random 112-bit values s_0 and r_0, concatenates these with its node ID I and the upcoming *descSqn*, and calculates an initial

SHA224 hash H_0 of these ingredients. This hashing is repeated n_{max} times[1] where for each iteration the first 112 bits of the output hash are used as input for s:

$$H_{n+1} = s_{n+1} \oplus r_{n+1} = SHA224(s_n \oplus r_0 \oplus I \oplus DescSqn) \tag{4.3}$$

The initial hash H_0 (and particularly its ingredient s_0) is kept as a private secret and is called the AHC root with which all followup hashes could be easily calculated. The final hash $H_{n_{max}}$ is published together with r_0, I, *descSqn*, and n_{max} in the description of the node. It is called the anchor of the hash chain and allows to easily verify whether a given 112-bit value is part of this hash chain or not. Followup routing updates sent by an originating node start revealing the pre-calculated hash values starting with $n_{max} - 1$ towards n_1. The amount of iterations i needed to match the hash-chain anchor $H_{n_{max}}$ of a received hash-chain value s_n (thus, $H_{n_{max}} = H_{n+i}$) is used to calculate the routing update sequence number (routeSqn) as:

$$routeSqn = (descSqn * n_{max}) + (i - n_{max}) \tag{4.4}$$

By doing a brute-force search for a hash-chain anchor in previously received and verified descriptions that matches a particular iteration output of Formula 4.3, receiving nodes can retain the identity of the node that originated the update and verify its authenticity and freshness in terms of corresponding *descSqn* and *routeSqn*. The former because node Id and *descSqn* are mandatory inputs for calculating the chain anchor; all of them being part of the description that has been signed with the node Id matching public-private key pair. This way, the calculated *routeSqn* also provides a secure heartbeat which propagation can be supported by other nodes via flooding. But its testimony can not be anticipated by any other node because only the originating node itself can create hash-chain values that yield a newer (or greater) *routeSqn* than any previously yielded *routeSqn* for that node.

Although, the described concept does not facilitate integrity validation of the overall routing update message (because the contained distance vector of the update is left unprotected), the provable *routeSqn* can always be used to identify the newest heartbeat of a node and disqualify old updates, making so called "false destination sequence attacks" [194] impossible.

Optionally, to relieve updates-receiving nodes from the extensive search of finding the hash-chain anchors for a given chain value, routing update messages can be extended with an (16-bit) internal identifier field (IID). Each node uses an individual IID value for identifying the originator of each routing updates it propagates to its neighbors. Updates-receiving nodes maintain a list for each verified link neighbor for mapping their IID values to a globally unique originator ID. If a mapping for given IID-neighbor tuple is known than the latest description for the corresponding originator ID is used for verifying the given

[1] $n_{max} = 6000$ has been selected as default to limit the needed hash power of used routing hardware while allowing to send new routing updates every 6 seconds and avoid description updates (due to exhausted AHC values) for $6sec * 6000 = 10hours$. SHA224 has been selected as the most lightweight standard among current state-of-the art secure hash functions. The truncation to 112 bits has been chosen considering available data from http://bitcoin.sipa.be/ (April 2017) that indicates the global cumulative number of yet calculated bitcoin double-sha256 hashes around $10^{26} \sim 2^{89}$ which is assumed to roughly correspond to that of 2^{90} single-sha224 hashes. Thus, we assume that, even given a doubling of hash-power per year, a brute-force attack for finding the matching first 112 bits of a SHA224 hash remains unfeasible for the following $112 - 90 = 22$ years, especially if the attacked AHC root is invalidated every few hours with each description update.

hash-chain value. If an originator ID for a given IID-neighbor tuple is not known, or if the maintained originator ID for a given tuple could not be proven as the originator of the received hash-chain value, then the received update is ignored and an IID-request message containing the unknown IID is sent to the neighbor using this IID. The expected IID-reply messages are then supposed to contain the nodeId, the currently used IID and the latest descSqn and hash-chain value known by the requested neighbor. Upon reception of this reply, its values can be verified (as described above) and used for processing further updates from this neighbor. As such, IIDs may be re-used and changed at any time. They only provide an option for receiving nodes for faster chain value verification. However, nodes showing too frequent IID re-assignments may always be ignored by their neighbors to avoid the unjustified resource consumption.

Routing updates and IID-request and reply messages exchanged between neighboring nodes are covered by the link authentication mechanism described in Section 4.5.2, allowing to always verify the authenticity, integrity, and non-repudiation of exchanged information.

The signature of the sender covering the update allows the receiving node to verify if the sender is trusted by the originating node and whether contained metric can be accepted for further path maintenance. This way, a routing update can be seen as a signed promise by each propagating node, stating to the receiving nodes that (i) this particular update has been propagated exclusively via nodes trusted by the originator of the message, and (ii) that all yet intermediate nodes also stated their willingness and capability for routing packets (with destination addresses as expressed via the referenced and signed description) towards the originating node.

4.5.3.3 Sub-topology authentication

The **trust set** of a node is given as a list of other node's global IDs that constitute the **trusted nodes** of this node. This way, the trust set implicitly defines, together with the links possible between any pair of trusted nodes, the **trusted virtual topology** of this trusting node. Only the good-listed nodes identified via this trust set are authorized for propagating routing updates originated by the trusting node. However, instead of expecting untrusted nodes to respect this demand, all trusted nodes are expected to ignore routing updates originated by the trusting node that were not received directly and securely authenticated via any of the trusted nodes. As a consequence, only neighboring nodes that are trusted nodes of a described trusting node are considered as next hop towards the trusting node and eventually also end-to-end established routing entries towards this trusting node are only set along these trusted nodes.

In addition, particular node IDs of a trusting primary node's trust set could be flagged as super-trusted nodes. This means that all nodes that are explicitly identified via the trust set of such super-trusted nodes shall also be considered as trusted nodes of the primary node. More precisely, the current overall trust set of the primary node is given as the union of its own directly trusted nodes, defined via its current description, and the directly trusted nodes defined via the current description of the primary's super-trusted nodes.

This way, the responsibility and burden of node administrators for identifying trustiness of existing nodes and maintaining trustiness-relations and respective trust-set configurations can be **delegated** to one or several secondary sources of assessed trustiness. Such sources may be given by a single node (or, to increase redundancy, by a small subset of nodes) from a larger set of nodes administrated by the same individual(s). Then, delegation would relieve administrators from having to reconfigure all their nodes for each considered trustability change (i.e. only a single node needs to be reconfigured while all other nodes having this single node flagged as super-trusted would automatically adapt the trust set of the super-trusted single node).

The preference of CN-users to minimize maintenance efforts as much as possible represents another likely use case and motivation for delegating, in part or entirely, the assessment of trustiness to one or several other sources that are highly and individually trusted anyway and that may actually be able to treat the topic of trustability with a much greater responsibility (for example due to a higher technical knowledge or better social connection with many CN participants) than the administrator of the actual delegating (primary) node.

4.5.3.4 Optional description content

Without going into much further details, a number of optional node properties, that might be interesting in the context of computer networks and considered worth for being propagated via node descriptions, have been supported via our implementation and are briefly summarized in the following.

Routing metric announcements allow to define a non-default algorithm for identifying the best next hop for routing packets towards the originating node. This way end-to-end path establishment can be tailored to individual traffic demands such as optimizing for delay, bandwidth, or robustness. The approach has been discussed in more detail in preceding work [4] and [6]. Essentially, such announcement specifies the function and related parameters to be used by other nodes for processing and propagating the metric values of routing updates originated by the announcing nodes.

Hostname and email announcements are typically used to give a name or contact address to a node. This way propagated hostnames only provide an informal and insecure way to name nodes. Since the selection of these only depends on the choice of the node administrator at a given time, their values may change at any time and the same values may be used by different nodes. However, an email address announced via the signed description of a node may still be used to gain confidence that that the holder of the nodes public-private key pair used for signing the email-announcing description is indeed the owner of the announced email account.

IP4in6 and/or IP6in6 tunnels announcements can be used for announcing the availability of other hosts and networks via tunneling through the announcing node. Just as with hostname and email announcement, any tunnel network could be announced by any node without restriction. However, other nodes, trusting a given set of tunnel-announcing nodes, and willing to send data to destinations within the announced networks are free to do so without suffering from overlapping announcements from other, maybe non-trusted nodes. Therefore, they encapsulate related traffic by adding an outer IPv6 header using the tunnel-announcing nodes primary IPv6 as outer destination address. This allows an automatic but end-user controlled IPv4 and/or IPv6 overlay networking on top of a securely-trusted IPv6 network.

Local topology announcements can be used by any node to periodically reveal the identity of its local link neighbors. Publishing such information may be useful for generating global connectivity maps, a feature that proved to be very famous in community networks for real-time topology, coverage, health, and growth visualization[2].

Last, but not least, **arbitrary data announcements** can be used to publish arbitrary, even binary data. This feature comes with an optional plugin that, if enabled, synchronizes the content of given data files to a specific directory in all other nodes receiving and processing such announcements. The remotely created data file names will always start with the node id of the announcing node. Protocol parametrization allows to ignore descriptions exceeding a particular size.

4.5.4 Bootstrapping Example

Figure 4.3 illustrates related bootstrapping procedures of the protocol in four phases: (1) the local node discovery, (2) the link discovery, (3) the global node discovery and (4) the trusted path establishment.

The local node discovery (phase 1) shows how nodes $n2$ and $n3$ react on the reception of a new (for $n2$ and $n3$ yet unknown) *nodePKHash* (I) or *descSqn* (here given by a received packet header) and resolve the description, identity (pubKey) and further referenced description content by sending corresponding **description or content request** messages (containing only the unresolvable hash) to node $n1$. Then $n1$ replies with the requested information. When all requested contents have been resolved, $n2$ and $n3$ can assemble the full description and verify its authenticity with the signature and public Key as defined by the initially received nodePKHash that triggered this process. The descHhash and signature is calculated over the raw description data and contained data hashes as they occur (not the referenced original data).

A transmission signature message (txSignature in Fig. 4.2 and 4.3) verifies the authenticity and integrity of all following **signature-demanding messages**. These include the locally exchanged link-discovery and routing-update messages but do not include the eventually globally exchanged descriptions (already signed explicitly via description signatures) or content advertisements (already signed implicitly via SHA reference hashes used in signed descriptions).

To protect against link-local replay attacks, the txSignature also covers the transmission sequence number (txSqn) that again can only be reused after renewing also the currently used txPK and a transmission IP (txIP) that must match the link-local src IPv6 address of the IP packet containing the protocol data.

The link discovery (phase 2) briefly indicates how the provided functionality can be used to authenticate the exchange of link-probing messages and subsequent deduced link qualities. Here, the hello message contains a sequence number that is unique during the lifetime of a node's description and the corresponding hello-reply message unambiguously references a previously received hello message and link via which this message has been transmitted (by specifying the hello-sequence number, the current description hash of the hello-sending node, and the txIP from which the hello has been sent and via which own interface it has been received). By receiving correctly signed hello and hello-reply messages (referencing the receivers own hello messages via included txSqn and txIP), the exchange of link-probing

[2]As a living example one may look at existing on-line maps such as `http://tomir.ac.upc.edu/qmpmon`, `http://sants.guifi.net/mapa`, `https://berlin.freifunk.net/network/map/`

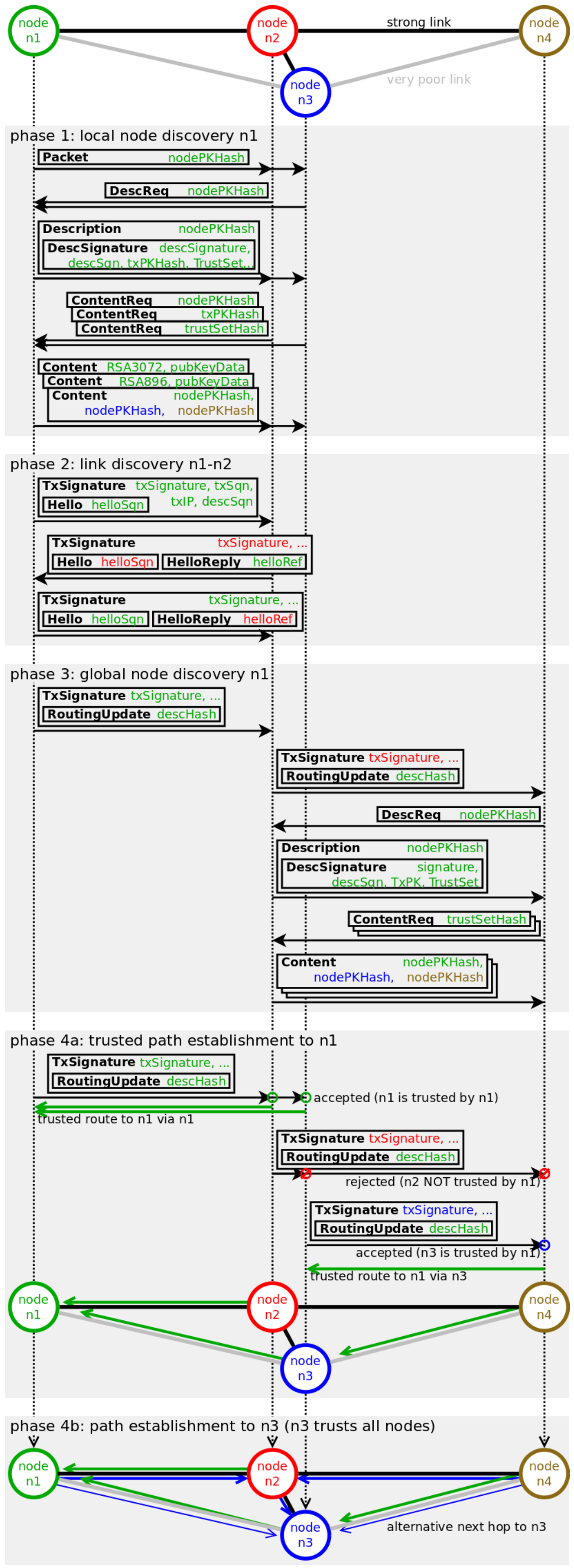

node n1
node n2
node n3
node n4
strong link
very poor link
phase 1: local node discovery n1
Packet nodePKHash
DescReq nodePKHash
Description nodePKHash
DescSignature descSignature, descSqn, txPKHash, TrustSet,..
ContentReq nodePKHash
ContentReq txPKHash
ContentReq trustSetHash
Content RSA3072, pubKeyData
Content RSA896, pubKeyData
Content nodePKHash, nodePKHash, nodePKHash
phase 2: link discovery n1-n2
TxSignature txSignature, txSqn, txIP, descSqn
Hello helloSqn
TxSignature txSignature, ...
Hello helloSqn
HelloReply helloRef
TxSignature txSignature, ...
Hello helloSqn
HelloReply helloRef
phase 3: global node discovery n1
TxSignature txSignature, ...
RoutingUpdate descHash
TxSignature txSignature, ...
RoutingUpdate descHash
DescReq nodePKHash
Description nodePKHash
DescSignature signature, descSqn, TxPK, TrustSet
ContentReq trustSetHash
Content nodePKHash, nodePKHash, nodePKHash
phase 4a: trusted path establishment to n1
TxSignature txSignature, ...
RoutingUpdate descHash
accepted (n1 is trusted by n1)
trusted route to n1 via n1
TxSignature txSignature, ...
RoutingUpdate descHash
rejected (n2 NOT trusted by n1)
TxSignature txSignature, ...
RoutingUpdate descHash
accepted (n3 is trusted by n1)
trusted route to n1 via n3
phase 4b: path establishment to n3 (n3 trusts all nodes)
alternative next hop to n3

Figure 4.3: Network bootstrapping (node, link, and path discovery)

messages becomes a continuous and bidirectional challenge-response handshake that only the holder of the corresponding private keys can maintain.

During the global node discovery (phase 3) nodes are discovered beyond the link neighborhood. The discovery is triggered by the detection of an unknown descHash contained in routing updates which causes the receiving nodes to resolve and verify the full node description (similar to local node discovery in phase 1).

Phase 4 illustrates the establishment of forwarding paths along trusted nodes. First $n2$ and $n3$ receive the routing update from $n1$, which description has already been fully resolved in phase 1. Because the update was received via an authenticated link (due to its txSignature) with known link-quality (see phase 2) and because $n1$ itself is listed as a trusted node in the description, $n2$ and $n3$ know that $n1$ trusts the transmitter of the received message with regards to correctly process, update, and re-broadcast its routing update messages. In the following, the end-to-end path cost to the originating node and via the trusted link peer can be calculated based on the pathMetric of the update and the current link quality. In case the calculated cost identifies this link as the best next hop towards the originating node, the routing update is re-broadcasted with a modified pathMetric value that reflects the cost of the additional link (see [5] for more details).

In the next illustrated step, node $n3$ and $n4$ receive the routing update originated by $n1$ and re-broadcasted by $n2$. This time, the update-conveying txSignature verifies $n2$ as the transmitter and last modifier of the routing update. However, the identity (nodePKHash) of $n2$ is not listed in the trustSet described by the originating node $n1$ and therefore the update is discarded (by $n3$ and $n4$) for path establishment towards $n1$. Node $n4$ does not have a trusted route to $n1$ unless it receives the routing update via $n3$ which is listed within the trustSet of $n1$ and can therefore be considered as next hop towards $n1$. The topology depicted at the bottom of phase 4a illustrates with arrows the resulting virtual topology for routing traffic towards $n1$. In this example, $n2$ is the only node which $n1$ does not trust. Consequently, its traffic is directed around $n2$. Given the assumption that $n3$ trusts all nodes, phase 4b illustrates with blue arrows also the concurrent established topology for routing traffic towards node $n3$. Traffic from $n4$ to $n3$ will be routed via $n2$ (due to potentially strong involved links $n4 - n2 - n1$). But traffic towards $n1$ remain routed via $n3$ (being the only trusted path option towards $n1$).

4.5.5 Summary of Properties and Advantages

In the following, fundamental properties and advantages of this approach are summarized:

- Descriptions can be rather short while being able to securely specify large amounts of data (e.g. a single 224-bit SHA2 hash compared to a 2048-bit RSA signature or a trust set listing the global IDs of hundreds of nodes).

- Referenced data that does not change over a description update or that is used by several nodes (e.g. defining equal trust sets) does not need to be re-propagated since their reference (hash) would not change and can be reused.

- A significant processing advantage is achieved by using a secondary and possibly weaker temporary key and verification mechanism (e.g. a weaker RSA key pair or a DH-HMAC based message authentication) for the continuous packet signing and validation operations (typically occurring several times per second), leaving a potential adversary with insufficient time for cracking this key before it gets replaced by a new one (typically every few hours). In contrast, a reasonable strong key can be used for signing the rather seldom propagated description updates (typically once per hour) and thereby allowing a long-term protection of node IDs and descriptions.

- The mechanisms allow (without requiring any pre-deployed infrastructure or registry) the network-wide, dynamic, and secure (in terms of authenticity and integrity) deployment of a public-key infrastructure that can be used for further integrity verification of link-discovery and routing-update messages between neighboring nodes and ensuring the propagation of routing updates along nodes trusted by the originator of these updates.

- With the ability to delegate the maintenance of trust sets to particular super-trusted nodes, it is possible to share and distribute the work related with a careful trustability assessment, while keeping the conceptual decentrality of the approach, and ensuring each actual node owner as the final root of trust that can withdraw or extend previously defined delegations at any time. This comes under the assumption that the administrators of super-trusted nodes are highly and individually trusted, and known to share the same routing preferences anyway but also they may actually be able to handle this topic with a much greater responsibility (for example due to a higher technical knowledge or better social connection with other CN participants) than the administrator of the actual delegating node itself.

4.6 Specification and Correctness

4.6.1 Definitions

- A *node description* $D_{X,Q}$ denotes a particular version $Q \in \mathbb{N}^+$ of a statements' summary created and signed by the node with the identity V_X. A valid description $D_{X,Q} = \{D^c_{X,Q}, S_{X,D^c_{X,Q}}\}$ contains unsigned description components $D^c_{X,Q}$ and a signature $S_{X,d} = s(K^{priv}_X, d)$ with data $d = D^c_{X,Q}$ that, created with the private key K^{priv}_X of V_X. $D^c_{X,Q}$, covers non-ambiguously definitions for V_X, K_X, $Q = Q^d_X$, A_X, $\mathbb{V}^t_X$, $\hat{M}_X$, and $f_X(M, L)$ as defined in the following. Nodes are supposed to learn the latest description (with highest description sequence number Q^d) of all other nodes. Therefore each node creates and broadcasts a description for itself which is then flooded by receiving nodes across the network. All nodes are also supposed to resend any earlier flooded description when demanded by a neighboring node so that new nodes can learn the descriptions of already existing nodes. Further background in Section 4.5.1.

- The *primary IPv6 address* $A_X = a(V_X)$ of a node is a Cryptographically Generated Address (CGA) based on a truncated version of the nodes identity V_X. The existence of a sufficiently secure CGA function $a(V)$ (as described in Section 4.5.3.1) that provides proof of identity and ownership is assumed for this work.

- A *link-metric function* $l(E_{B,A}) = L_{B,A}$ with $L \in \mathbb{R}^+$ is a heuristic function that quantifies the quality of a given link as a positive real value (including zero). For any $L_{B,A} \geq L_{Z,Y}$ it is defined that the link quality of link $E_{B,A}$ is better than (equal to) that of $E_{Z,Y}$. The existence of a sufficient link-metric function is assumed for this work.

A simple but sufficient link-metric function for letting V_A assess the quality of link $E_{B,A}$ could be given by the following two processes. For V_A the process would be as follows:

– V_A broadcasts $tx(d^T)$ with $d^T = \{s_A(d^t_A), d^t_A, D_{A,Q^d_A}\}$ and $d^t_A = \{V_A, c^t, Q^t_A\}$, $c^t = ping$ representing a message code, and $Q^t_A \in \mathbb{N}^+$ being a (ping) sequence number.

– Then V_A waits for an interval of I^l

– If, during I^l, V_A received $rx(d^R)$ and if
$d^R == \{s_B(d^r), d^r, D_{B,Q^d_B}\}$ and
D_{B,Q^d_B} is valid (contains V_B, K^{pub}_B,...) and
$V_B \neq V_A$ and
$\overline{s}(s_B(d^r), d^r, K^{pub}_B) == true$ and
$d^r == \{c^r, V^r, Q^r, V_A\}$ and
$c^r == pong$ and $V^r == V_B$ and $Q^r == Q^t_A$ (and $V_A == V_A$) then:
Set $l(E_{B,A}) = 1$
Else: Set $l(E_{B,A}) = 0$

– V_A sets $Q^l_A = (Q^l_A + 1)$ and repeat.

For any neighbouring node of V_A, such as V_B the process would be as follows (for any $V_X \neq V_B$ such as V_A):

– If V_B receives $rx(d^R)$ and if
$d^R = \{s_X(d^r), d^r, D_{X,Q^d_X}\}$ and D_{X,Q^d_X} is valid (contains V_X, K^{pub}_X,...) and
$\overline{s}(s_X(d^r), d^r, K^{pub}_X) == true$ and
$d^r == \{V^r, c^r, Q^r\}$ and
$c^r == ping$ and $V^r == V_X$ and $Q^r \in \mathbb{N}^+$ then:

– V_B immediately broadcasts $tx(d^T)$ with
$d^T = \{s_B(d^t), d^t, D_{B,Q^d_B}\}$ and
$d^t = \{V_X, c^t, Q^t\}$ with $c^t = pong$, and $Q^t = Q^r$.

If description D has already been propagated and known by neighbouring nodes as described above (and as considered by design in Section 4.5.1 and 4.5.2) they do not need to be transmitted with each d^T messages because they could be looked up in memory by receiving nodes via contained V^r components.

- The overall *topology* of a network is given by the graph $G = g(\mathbb{V}, \mathbb{E})$ representing all nodes $V \in \mathbb{V}$ as vertices and all links $E \in \mathbb{E}$ as edges if $l(E) > 0$.

- A *path-quality value* $M_{T,S}$ and *path-metric function* $m(V_T, V_s, f_T, \hat{M}_T)$ with $V_T \equiv V_0, V_S \equiv V_s, s \geq 1$ is defined as:

$M_{T,S} = M_{0,s} =$

$$m(V_T, V_s, f_T, \hat{M}_T) = \begin{cases} \text{if } s == 1 : f_T(\hat{M}_T, L_{T,s-1,s}), \text{ else} \\ \text{if } f_T(m(V_T, V_{s-1}), L_{T,s-1,s}) == 0 : 0 \\ \text{else } : f_T(m(V_T, V_{s-1}), L_{T,s-1,s} \end{cases}$$

$\hat{M}_T$ is a constant defined by V_T via $D_{T,Q}$ for the upper bound of any $M_{T,S}$. $f_T(M, L)$ is the customizer function specified via $D_{T,Q}$. It must ensure that

$$0 \leq f(M, L) < M < \hat{M} \text{ for any } M \in \mathbb{R}^+ \text{ , } L \in \mathbb{R}^+ \tag{4.5}$$

A function satisfying Equation (4.5) would also ensure that:

- Greater path quality values reflect better paths, thus for any path $P_{T,A} = \{V_T, ..., V_A\}$ with a better (equal) path quality than another path $P_{T,X} = \{V_T, ..., V_X\}$ that $M_{T,A} > (=)\ M_{T,X}$.
- A sub path $P_{T,A} = \{V_T, ...V_A\}$ of another path $P_{T,X} = \{V_T, ..., V_A, ..., V_X\}$ with equal destination V_T and path components up to including V_A must lead a greater path metric $M_{T,A}$ than that of $M_{T,X}$. The existence of a sufficient path metric is assumed for this work.

- A *route* $R_{T,S} = \{V_T, V_{s-1}, V_s\}$ explicitly identifies the next hop V_{s-1} of a particular path $P_{T,S}$ from $V_s \equiv V_S$ via V_{s-1} and implicitly defines also all following nodes $V_{s-2}, ...V_1, V_0$ to $V_0 \equiv V_T$.

- The *set of all routes* from V_S to V_T with different next hops is limited by the number of neighbors of V_S and is given as $\mathbb{R}_{T,S}$.

  A *routing function* $r(V_T, V_a) = R_{T,A} = \{V_T, V_{a-1}, V_a\}$ implicitly also identifies all consecutive nodes by iterative resolving V_{a-2} from $r(V_T, V_{a-1}) = \{V_T, V_{a-2}, V_{a-1}\}$.

- The *trust set* $\mathbb{V}^t_X \subseteq \mathbb{V}$ of a node V_X lists the node identities (including itself) of its trusted nodes. E.g.: $\mathbb{V}^t_X = \{V_A, V_B, V_C, V_X\}$.

- A *trusting node* is a node (e.g. V_S) that defines a non-empty trust set $\mathbb{V}^t_S$ to secure itself against attacks from non-trusted nodes. Therefore, the cardinality of $\mathbb{V}^t_S$ must be greater or equal than one: $|\mathbb{V}^t_S| \geq 1$. Thus, if a particular node V_X does not trust any other node then it would define only itself as trustable: $\mathbb{V}^t_X = \{V_X\}$.

- A *trusted node* is a node (e.g. V_X) which is explicitly listed in the trust set $\mathbb{V}^t_A$ of a given node V_A. This is formalized as $V_X \in \mathbb{V}^t_A$. A *non-trusted node* V_Y is a node which is not listed in the trust set of a given other node V_A, formalized as $V_Y \notin \mathbb{V}^t_A$. Thus, trust is a tuple of the trusted node (V_X) and the *trusting node* (V_A). A particular node V_Z can be a trusted node of node V_A and at the same time be a non-trusted node of node V_B: $V_Z \in \mathbb{V}^t_A \bigwedge V_Z \notin \mathbb{V}^t_B$. Trust relations are not implicitly symmetric, e.g.: $V_Z \in \mathbb{V}^t_A \bigwedge V_A \notin \mathbb{V}^t_Z$. Direct trust relations are not transitive, e.g.: $V_C \in \mathbb{V}^t_B \bigwedge V_B \in \mathbb{V}^t_A \bigwedge V_C \notin \mathbb{V}^t_A$.

- A link $E_{B,A}$ is a *trusted link* of node V_X if the two neighboring nodes V_A and V_B are trusted nodes of V_X. The *trusted link set* $\mathbb{E}^t_X$ of node V_X is defined as

$$\mathbb{E}^t_X = \{E_{B,A} | E_{B,A} \in \mathbb{E}, V_A \in \mathbb{T}_X, V_B \in \mathbb{T}_X\} \tag{4.6}$$

- The *trusted topology* G^t_X of node V_X is a sub graph of $G(\mathbb{T}, \mathbb{L})$ consisting of vertices $V \in \mathbb{T}^t_X$ and edges $E \in \mathbb{E}^t_X$ with $G^t_X = G(\mathbb{T}^t_X, \mathbb{E}^t_X)$.

- A *trusted path* is a connected end-to-end path between a given source-destination-node tuple that consists only of trusted nodes and links of the (trusting) destination node. It represents one particular combination of consecutive links from the trusted topology of the destination node. A *trusted route* denotes, according to a given path metric, the currently best existing trusted path.

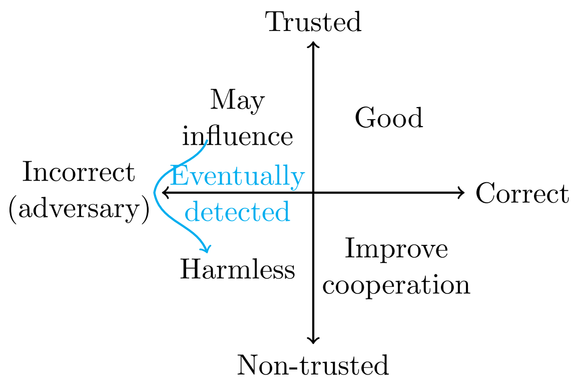


Figure 4.4: Cases in SEMTOR considering trust, correctness, and a detection mechanism

4.6.2 Desired Properties

For any given node V_T two orthogonal properties of nodes are considered: correct ($V^c \in \mathbb{V}_T^c$) versus adverse ($\overline{V^c} \notin \mathbb{V}_T^c$), and trusted nodes ($V^t \in \mathbb{V}_T^t$) versus non-trusted nodes ($\overline{V^t} \notin \mathbb{V}_T^t$). This leads to four possible combinations as illustrated by the four sectors in Figure 4.4: Non-trusted correct nodes, trusted correct nodes, non-trusted adversaries, and trusted adversaries.

For any given (T, S)-path tuple and network graph G one of the following topological constellations can be considered:

a) Incorrect path: S is isolated from T either because a continuous (T, S)-path in G does not exist at all or because not a single consecutive path exists that consists only of T-correct nodes: $\mathbb{P}_{T,S}^c = \varnothing$

b) Correct path: At least one (T, S)-path of T-correct nodes exists, that may (or may not) consist of T-trusted nodes, but some T-trusted nodes of G may be T-adverse (incorrect) nodes: $\mathbb{P}_{T,S}^c \neq \varnothing \wedge \mathbb{V}_T^t \setminus \mathbb{V}_T^c \neq \varnothing$

c) Correct trust and path: At least one (T, S)-path of only T-correct nodes exists and T-trusted adversaries do not exist but a (T, S)-path of only T-trusted nodes may not exist: $\mathbb{P}_{T,S}^c \neq \varnothing \wedge \mathbb{V}_T^t \setminus \mathbb{V}_T^c = \varnothing \wedge \mathbb{P}_{T,S}^t = \varnothing$

d) Correctly trusted nodes and path: At least one (T, S)-path of only T-trusted and correct nodes exists and T-trusted adversaries do not exist: $\mathbb{P}_{T,S}^c \neq \varnothing \wedge \mathbb{V}_T^t \setminus \mathbb{V}_T^c = \varnothing \wedge \mathbb{P}_{T,S}^c \neq \varnothing$

We are interested in the correct and continuous provisioning of *awareness* (node and node state discovery) and *reachability* (path and route discovery and packet delivery) services.

These services can be characterized by a set of properties [163, 195] that can be differentiated between safety and liveness properties [196, 197].

Safety properties state the guarantee that bad things do not happen:

- *Information authenticity and integrity* are mandatory properties for an awareness service that, in our case, shall propagate node descriptions and heartbeats. We define this property as *update-safety* which guarantees that if a $Q_{T,S}$-sequenced update (supposedly created by V_T) can be verified by a receiving node V_S as correct then it must have been sent earlier by the supposed originating node V_T as $Q_{T,T}$ and has propagated unchanged to V_S. Because every $Q_{T,T,\tau_n} \in \mathbb{N}^+$ sent by V_T at time τ_n must be greater or equal to any previously sent $Q_{T,T,\tau_{n-1}}, \tau_n > \tau_{n-1}$ this guarantee is expressed as: $Q_{T,T,\tau} \geq Q_{T,S,\tau}$. This property shall apply to description and heartbeat updates, sequenced with Q^d and Q^h respectively.

- *Path-adversary-freedom* is an important property for guaranteeing the reachability of V_T from V_S along a path. A path $P_{T,S}$ is adversary-free if it has no adversary nodes.
- *Adversary-resilience:* Routing is adversary-resilient if its behaviour under the presence of adversary nodes is equivalent to that of a network that has no adversary nodes. This means that adversary nodes, if present, do not have any noticeable effect. An update-safe awareness service and a delivery service that routes and forwards data packets only along adversary free paths can be considered adversary resilient.
- *Loop-freedom:* A path is loop-free if it has no repetition of nodes.
- *Consistency:* packets only traverse loop- and adversary-free paths between source and destination. We define this property as *route-safety* for any established route $R_{T,S}$ that consists only of non-repetitive nodes that behave correctly (non-adversary) regarding $V_T : R_{T,S} \subseteq \mathbb{V}^t_T$.

There are also liveness properties which state that good things [eventually] happen:

- *Update freshness:* An update is fresh with respect to an interval I and an incrementally increasing update-version Q (update-sequence number) if the last received update $Q_{T,S,\tau}$ from V_T by V_S at time τ is greater or equal than any update $Q_{T,T,\tau-I}$ sent by V_T since I. We define *update-liveness* as the property that guarantees update freshness so that: $Q_{T,S,\tau} \geq Q_{T,T,\tau-I}$.
- *Route freshness:* A route $R_{T,S,\tau}$ from V_S to V_T at time τ is fresh with respect to an interval $I = (\tau - I, \tau)$ and a given set of existing paths $\mathbb{P}_{T,S,I}$ if all implicitly via $R_{T,S,\tau}$ defined consecutive hops and respective links are up during I. We define *route-liveness* as the property that guarantees in addition to route freshness also consistency so that: $R_{T,S,\tau} \in \mathbb{P}^c_{T,S,I}$.
- *Route accuracy:* A route $R_{T,S}$ is accurate if it is consistent and fresh and, with respect to a path-metric function $m_T()$ defined via $D_{T,Q}$, if the path-metric value calculated and re-propagated by V_S conforms with the actual link-metrics $L_{B,A}$ between all successive path nodes $V_B, V_A \in R_{T,S}$.
- *Responsiveness:* Depending on the logic of the routing-update process, updates received by a node are validated (which, if necessary, may include the immediate request and resolution of depending information) and applied to its forwarding table and propagated to other routers, including those that potentially depend upon the outcome of the update. This affects how quickly the network reacts to changes and therefore it qualifies the timeliness of freshness.

  It is assumed that the transmission delay υ (from data-link layer assumptions in Section 4.4.2) of up links and the responsiveness ρ of correct nodes for processing

Constellation	correct path exists: $\mathbb{P}^c_{T,S} \neq \varnothing$	no trusted adversaries: $\mathbb{V}^t_T \setminus \mathbb{V}^c_T = \varnothing$	trusted path exists: $\mathbb{P}^t_{T,S} \neq \varnothing$	no adversaries: $\overline{\mathbb{P}^c_T} \neq \varnothing$	Update-Safety (authenticity): $Q_{T,T,\tau} \geq Q_{T,S,\tau}$	Update-Liveness (freshness): $Q_{T,S,\tau} \geq Q_{T,T,\tau-I}$	Route-Safety (consistency): $R_{T,S,Q^h} \subseteq \mathbb{V}^c_{T,Q^d}$	Route-Liveness (freshness): $R_{T,S,\tau} \in \mathbb{P}^c_{T,S,(\tau-I..\tau)}$
a)	*	*	*	*	✓	-	-	-
b)	true	*	*	*	✓	✓	-	-
c)	true	true	*	*	✓	✓	✓ $(R_{T,S} = \varnothing)$	-
d)	true	true	true	*	✓	✓	✓ $(R_{T,S} \neq \varnothing)$	✓

Table 4.2: Topology constellations and respective protocol properties.

received updates and responding to resolution requests from neighbors is much smaller than I: $\upsilon + \tau \ll I$

- *Delivery:* data packets get eventually delivered to its destination though a path, with statistical guarantees as channels are usually lossy, typically in a small percentage. This property depends on the liveness of updates (freshness) and routes (freshness and accuracy).

- *Efficiency:* The expected resource overhead cost for each node amortized over all usage. Resources can be processing (CPU, memory in nodes) and communication (traffic in links), and the overhead cost, the additional resource consumption due to the protocol translated into a performance penalty (cost).

Safety and liveness properties about updates and routes are the basis to ensure overall properties: the (safety) consistency and resilience to adversaries; and the (liveness) responsiveness, delivery and efficiency of routing.

Table 4.2 summarizes the desired safety and liveness properties and identifies the previously described topology constellations a) - d) for which these properties shall be satisfied. Here the $*$-symbol is used to indicate that no assumptions are made on the requirement given in the respective header row (thus, may be true or false) and the ✓-symbol is used to indicate that a corresponding property can be satisfied. None of the four constellations makes assumptions on the presence of non-trusted adversaries.

It should be noted that the assumptions and objectives outlined in Section 4.3.1 are actually reflected by the conditions and properties listed for constellation d) while constellation a) to c) list properties that shall be provided under even less restrictive conditions.

Regarding the security objective: Any group of nodes (e.g. subeset $\mathbb{V}_X \subseteq \mathbb{V}_Y$) that is trusting (so $\forall V_T \in \mathbb{V}_X : \mathbb{V}^t_T = \mathbb{V}_X$), willing to cooperate and support each other (correct, so $\mathbb{V}_X \subseteq \mathbb{V}^c_T$ with $\mathbb{V}^t_T = \mathbb{V}_X \Rightarrow \mathbb{V}^t_T \setminus \mathbb{V}^c_T = \varnothing$, so not trusted adversaries exist), and contiguous (connected, so with $\mathbb{V}^t_T = \mathbb{V}_X \Rightarrow \forall V_T, V_S \in \mathbb{V}_X : \mathbb{P}^t_{T,S} \neq \varnothing$) can not be inhibited by an external $V_Z \notin \mathbb{V}_X$ on the objective of routing packets to each other (which is satisfied by the given route-safety and -liveness properties).

4.6.3 Proving Correctness

In this section we analyze the safety and liveness properties defined in Section 4.6.2 for updates and routes as lemmas and theorems that we prove with formal reasoning.

Lemma 1 *Descriptions discovered by SEMTOR are authentic and can be differentiated between older and newer versions, independent of the presence of trusted or non-trusted adversaries.*

Proof: A description $D_{X,Q}$ received by a SEMTOR node is only accepted as valid if it contains the components $D^{c}_{X,Q} = \{V_X, K^{pub}_X, Q^d_X, A_X, \mathbb{V}^t_X, \overline{M}_X, f_X\}$ and a signature $S_{X,D^c_{X,Q}}$ for which V_X equals $v(K^{pub}_X)$ and $\overline{s}(K^{pub}_X, S, D^c_{X,Q})$ is *true* and Q^d_X is greater than any earlier received and accepted description from V_X. Description integrity is proven because, given Assumption 2, a $\overline{s}(K^{pub}_X, S, D^c_{X,Q})$-correct signature can not be created by any other node than V_X, so also not by any adverse node. Identity and authenticity is implicitly proven because of the matching signature and V_X matching $v(K^{pub}_X)$. Exclusive ownership of A_X is proven based on Cryptographically Generated Address (CGA) respective rules. Possibly earlier received descriptions from V_X can be reliably identified as outdated if contained $Q^d_{earlier} < Q^d_{current}$. □

Lemma 2 *Heartbeats discovered by SEMTOR are authentic, can be related to a particular description of the originating node, and differentiated between older and newer versions, independent of the presence of trusted or non-trusted adversaries.*

Proof: A heartbeat $Q^b_{X,Q}$ received by a SEMTOR node is only accepted as valid and identified as originated by V_X if, among the latest accepted descriptions known from any node, one can be found which contained AHC anchor $Q^a_{X,Q}$ yields a positive heartbeat-sequence number calculated as $Q^h_X = h(Q^a_{X,Q}, Q^b_X, V_X, Q^d_X)$ and one of the following conditions is true: Either Q^h_X is equal or greater than any previously accepted Q^h_X value from V_X that was based on the same $Q^a_{X,Q}$ and $Q = Q^d_X$. Or the last accepted heartbeat-sequence from V_X was based on an outdated description from V_X that has since then be replaced by the receiving node with an accepted description updated. The authenticity, belonging to a particular description, and differentiability (in older and newer versions) of heartbeats is proven because of the cryptographic properties of AHCs (see also Section 4.5.3.2 and Assumption 2) that also covers V_X and Q^d_X so that only the originating node V_X, that knows the secret AHC root, can create heartbeats that yield Q^h_X values that are acceptable as newer (greater) than the heartbeat with the greatest corresponding Q^h_X value ever sent before by V_X. □

Theorem 1 *SEMTOR guarantees update-safety: the authenticity, integrity, and differentiability of discovered node descriptions and heartbeats.*

Proof: From Lemmas 1 and 2. □

Theorem 2 *SEMTOR guarantees update-liveness: Descriptions and heartbeats discovered by SEMTOR in the presence of correct paths are fresh so that any description $D_{X,Q}$ or heartbeat $Q^b_{X,Q}$ originated at time t_1 by V_X with $Q = Q^d_X$ is discovered by any V_Y before time $t_2 = t_1 + I$ if a consecutive path $P^c_{X,Y}$ of V_X-correct nodes exists between V_X and V_Y that is up during the entire interval $I = (t_1, t_2)$.*

Proof: Let $I^h = (t_2 - t_1)$ be the interval duration with which each node V_X broadcasts a new heartbeat (update) $Q^b_{X,Q}$ yielding a heartbeat-sequence number $Q^h_{X,Q}$ that is one greater than the previously broadcasted heartbeat sequence of V_X. If an up-link between the $Q^b_{X,Q}$-broadcasting node and a correct neighboring node V_N of the broadcasting node exists during the broadcast of $Q^b_{X,Q}$ then V_N receives $Q^b_{X,Q}$ and one of the following three cases occur:

(i) V_N already knows and accepted $D_{X,Q}$ (see Lemma 1) that $Q^b_{X,Q}$ is based on and has also already accepted $Q^b_{X,Q}$ (see Lemma 2) due to an earlier reception from a neighboring node of V_N. Then V_N only responds to description requests received from its neighboring nodes that demand a description known by V_N.

(ii) V_N already knows and accepted $D_{X,Q}$ that $Q^b_{X,Q}$ is based on and accepts $Q^b_{X,Q}$ as a new heartbeat update. Then $Q^b_{X,Q}$ is re-broadcasted by V_N.

(iii) V_N does not know $D_{X,Q}$ that $Q^b_{X,Q}$ is based on. Then the acceptance-processing of $Q^b_{X,Q}$ is postponed by V_N for at most I^h, $Q^b_{X,Q}$ is cached for possible later processing for the duration of I^h, and a request is broadcasted to all neighboring nodes of V_N demanding for a description that allows V_N to accept $Q^b_{X,Q}$. If within I^h a description $D_{X,Q}$ is received by V_N that is acceptable and yet unknown by V_N then V_N accepts $D_{X,Q}$ and searches its cached heartbeats for one that can be accepted based on $D_{X,Q}$. If this is the case then V_N continues as described for case (ii).

As a consequence of this mechanism, and given that the transmission delay of up links and the responsiveness of correct nodes for processing received updates and responding to requests is much smaller than I, then $Q^b_{X,Q}$ and $D_{X,Q}$ will be received, accepted, and re-broadcasted by all correct neighboring V_N nodes of the originating node V_X to which a link exists during I. And they will be further propagated to all correct neighboring nodes of V_N which, if behaving correct, will further propagate these till they are accepted by all nodes V_Y to which a consecutive path $P^c_{X,Y}$ of V_X-correct nodes exists that is up during the entire interval I. $\square$

Theorem 3 *SEMTOR guarantees route-safety: Routes discovered by SEMTOR in the presence of non-trusted adversaries (implying that trusted adversaries do not exist) are loop free and adversary free.*

Proof: Any destination node V_T broadcasts a new and signed routing update message $U_{T,Q^b_T,T} = \{Q^b_T, M_{T,T}\}$ with each new heartbeat update Q^b_T and with $M_{T,T} = \hat{M}_T$.

Any route R_{T,S,Q^b_T} to destination V_T, propagated via routing update messages $U_{T,Q^b_T,N} = \{Q^b_T, M_{T,N}\}$, and broadcasted by node V_N (V_N may be equal with V_T) is only accepted by a correct receiving node V_S if all of the following conditions are satisfied.

(i) The current description D_{N,Q^d_N} of V_N is known and accepted (from proof of Lemma 1).

(ii) The transmission signature that covers $U_{T,Q^b_T,N}$ proves the authenticity and integrity of the update as having been created and broadcasted by V_N (note also Assumptions 1 and 2 and related mechanism details described in Sections 4.5.2 and 4.5.3.2).

(iii) A description D_{T,Q^d_T} is known that allows to accept Q^b_T from $U_{T,Q^b_T,N}$ as a valid heartbeat update from V_T (from proof of Lemma 2)

(iv) Q^b_T is either new or Q^b_T is equal to the newest heartbeat update from V_T and $M_{T,N}$ from $U_{T,Q^b_T,N}$ is the greatest path-metric value ever accepted (via any neighbor) for V_T. Q^b_T.

(v) $M_{T,N}$ from $U_{T,Q^b_T,N}$ is greater than zero.

(vi) V_N is listed as a trusted node in $\mathbb{V}^t_T$ from D_{T,Q^d_N}.

Only if a routing-update messages received by correct node V_S is accepted as valid then V_S configures a forwarding route to V_T via V_N and propagates the route further by creating and broadcasting a new and V_S-signed update as: $U_{T,Q^b_T,S} = \{Q^b_T, M_{T,S}\}$ with $M_{T,S} < M_{T,N}$.

Condition (vi) is only true if $V_N \in \mathbb{V}^t_T$, which, given that $\mathbb{V}^t_T \subseteq \mathbb{V}^c_T$, implies that $V_N \in \mathbb{V}^c_T$. Therefore routes are only accepted and configured if propagated along correct, therefore adversary free, paths.

Condition (iv) is only true if the contained heartbeat and/or path-metric value represents an update that is newer or better than any previously received updates by the processing node. Therefore neither the processing node, nor any other correctly processing node, could yet be part of a route established by any updates with the same heartbeat sequence. Therefore all accepted routes are loop free. $\square$

Theorem 4 *SEMTOR guarantees route-liveness: Routes discovered by SEMTOR, in the presence of non-trusted adversaries (implying that trusted adversaries do not exist) and at least one correct and trusted path, are fresh.*

Proof: The proof of route-liveness is provided by an equivalent reasoning as given for Theorem 2.

As a consequence of the routing-update propagation mechanism from the proof of Theorem 3, and given that the transmission delay of up links and the responsiveness of correct nodes for processing received updates and responding to requests is much smaller than I, then any routing update U_T originated by V_T will be received, accepted, and further propagated by all correct neighboring V_N nodes of V_T to which a link exists during I. And they will be further propagated to all correct neighboring nodes of V_N which, if behaving correct, will further propagate these till they are accepted by all nodes V_S to which a consecutive path $P^{c,t}_{T,S}$ of V_T-correct and trusted nodes exists that is up during the entire interval I. $\square$

4.7 Experimental Validation

4.7.1 Implementation and Validation Framework

In this section the key security and performance achievements of the SEMTOR protocol are validated experimentally. First the proposed protocol mechanisms have been implemented and then this implementation has been observed when exposed to key attack scenarios and challenging network environments.

For the implementation, the message structure and handling of the BMX6 [38] routing-protocol has been extended as it already provides functional support for node-descriptive profiles, SHA hashes as referencing identifier, and a "hash-based profile-propagation

Table 4.3: Protocol implementation details

Topic	Details and used default
Protocol Source code	BMX7 branch, git revision ee2f1d86
Crypto Library	MbedTLS version 2.4.0
Node-identity and description referenciation	SHA224 (RFC4634) hashes
Primary key foundation	RSA2048 (RFC8017)
Asymmetric txKey foundation	RSA896 (RFC8017)
Symmetric txKey foundation	112-bit truncated SHA224-MAC based on DHM2048 (RFC3526) negotiated shared key
Description-update interval	36000 s
Heartbeat authentication	112-bit truncated, SHA224-based, anchored hash chain
Routing updates interval	6 s
Link-probing interval	0.8 s
Aggregation (TX) interval	0.8 s

mechanism" to disseminate node descriptions. For cryptographic operations such as hashing, asymmetric-key generation, signing, and verification the MbedTLS [198] (former PolarSSL [199]) library has been used. Our SEMTOR implementation currently provides RSA512 to RSA4096 public-key authentication for node and description verification, SHA224-based hashes for node-, description-, and routing-update referenciations (the latter based on anchored-hash chains), and Diffie-Hellman (DH) key exchange for Message Authentication Codes (MAC) based neighbor verification. The complete source code is publicly available via the BMX7/SEMTOR git-repository branch at [38] and [200]. Protocol implementation details are summarized in Table 4.3.

The scenarios to which our SEMTOR-enabled BMX6 implementation got exposed was realized with an emulated network consisting of Linux nodes and using MLC [39], a suite of LXC[3]-based scripts, that allow to visualize hundreds of Debian systems inside a single host and to configure virtual topologies, essentially by linking virtual or real nodes' interfaces via a virtual bridge and using ebtables[4] and tc[5] to control packet loss and delay between nodes on layer two.

The host system was an Intel i5-3230M CPU @2.60GHz with 4 CPU cores and 8GB RAM, allowing to run more than 200 virtual systems and protocol instances in parallel.

Further emulation parameters and ranges used for the functionality validation in Section 4.7.2 and the performance measurements following in Section 4.7.3 are given in Table 4.4. The selected probing ranges comprise network sizes and densities that are typical for deployments in clouds and core-graph zones of real community networks [2, 23, 24].

4.7.2 Validation of Functionality

To validate our design and implementation (regarding the ownership, security, openness, and decentralization objectives as outlined in Section 4.3.1), a number of representative network attacks have been set up and analyzed respectively.

[3]Linux Containers `http://lxc.sourceforge.net/`

[4]Linux ebtables, `http://ebtables.sourceforge.net`

[5]Linux traffic control: tc, `http://tldp.org/HOWTO/Traffic-Control-HOWTO/`

Table 4.4: Emulation defaults and ranges

Parameter	Default [range]	Figures
Host hardware	Intel i5-3230M @2.60GHz, 4 cores, 8 GB RAM	
Host system	Linux, Debian	
Emulation system	MLC git rev. d63f726c	
Network virtualization	ebtables, qdisc, brctl	
Emulated systems	Linux, Debian	
Measurement tools	tcpdump, tshark, top, ping, iperf	
Network structure (topology)	Grid 10x10 [10x2..20]	4.13
Link loss	0% loss [0%,30%,50%]	4.9,4.10-4.12
Node interfaces	1	
Network size (# of nodes)	100 [10..200]	4.14a,4.15a
Density (# of links per node)	4 [4..20]	4.14b,4.15b
Primary key strength	RSA2048 [512..4096]	4.14d,4.15d
TxKey method	RSA896 vs. DHM2048	4.14, 4.15
TxKey strength	RSA512..1536, DHM1024..3072	4.14c, 4.15c
Others description-update interval	36000 s [100..2 s]	4.14e,4.15e
Own description-update interval	36000 s [10..2 s]	4.14f,4.15f

Therefore, Section 4.7.2.1 first details the implementation of different attack vectors and Section 4.7.2.2 then evaluates their impact on a node that naively (and falsely) trusts in the correct behavior of all nodes (including existing malicious and attacking nodes) which corresponds with constellation b) from Table 4.2.

Section 4.7.2.3 then validates our SEMTOR implementation and quantifies its performance to recover, given that attacking nodes have been correctly identified (as given by constellation d) from Table 4.2), from the most powerful combination of previously introduced attacks. Eventually, Section 4.7.2.4 uses obtained measurements to illustrate further the protocol contributions regarding openness, decentralization, and cooperativeness.

4.7.2.1 Attack Vectors

In order to experimentally validate the resistance of the SEMTOR protocol when exposed to attacks, an implementation is needed that supports the execution of a corresponding attack. In this section we describe and validate the implemented attacks by observing their impact on a node that is reachable via paths consisting of correct and adverse nodes and that considers all, thus also the attacking node(s), as trusted. That corresponds to the most adverse topology constellation b) from Section 4.6.2.

For the below detailed attacks, an attacking node behaves, with particular exceptions, just as a correct (non-attacking) node and only the deviations from the correct behavior are described (all not-explicitly mentioned mechanisms behave correct). Functions implementing these attack-specific deviation from the correct behavior are provided via a so called “evil” plugin, extending the default protocol implementation with malicious capabilities. The plugin allow to (i) define a set of attacked nodes based on a list of node IDs and (ii) define the combination of attacks performed on these nodes.

We have considered attacks to manipulate or selectively suppress routing information about the identity and description of the nodes, the routing updates, and route announcements.

An **Identity Hijacking Attack (IHA)** would be given by any successful attempt to propagate a manufactured or modified description of an attacked node, for example by replacing the trust set or transmission key of the attacked nodes original description. One variance of such description manufacturing has been implemented by transmitting descriptions of attacked node IDs with an incremented description SQN.

Test confirmed that this-way (or any way) manipulated descriptions are discarded immediately upon reception by any correctly behaving node because of the description signature mismatch with the node IDs public key (remember that a node ID is defined as the SHA hash of its public key). To successfully execute this attack, an adversary would need to manufacture also a matching signature which is possible only by knowing the node IDs private key. Therefore the IHA attack would fail as long as node ID private keys are not revealed.

A **Route (or primary IP) Hijacking Attack (RHA)** is implemented by letting an attacking node announce the primary (CGA) IP addresses of the attacked nodes via its own description and propagating this and this description referencing routing updates to the network via its neighboring nodes.

Test confirmed that this-way composed descriptions and this description referencing routing updates are discarded immediately upon reception by any correctly behaving node because the contained CGA announcements do not match with the descriptions' node ID.

A **Heartbeat (or routing-update) Manufacturing Attack (HMA)** is implemented by transmitting all internally scheduled routing update messages, that reference the description of an attacked node ID, with a hash-chain value replaced with random bits.

Test confirmed that this-way manipulated and received routing-update messages are not further processed nor propagated because the contained hash-chain value does not match with the hash-chain anchor of any known description.

A **Description Dropping Attack (DDA)** is achieved by discarding all internally scheduled description transmissions that contain a node ID matching any of the attacked node IDs.

A **Routing-Update Dropping Attack (UDA)** is achieved by discarding all internally scheduled routing message transmissions that reference the description of an attacked node ID.

A **Route Dropping Attack (RDA)** is achieved by configuring a route towards the primary (CGA) address of each attacked node, instead of via the next hop of its collectively established path, but into a virtual "/dev/null" interface that drops all IP packet routed into it.

A **Routing-Update Metric Manipulation Attack (MMA)** is achieved by transmitting all internally scheduled routing update messages, that reference the description of an attacked node ID, with a metric value modified to the maximum.

From above attacks, three categories can be identified: (i) Those that the SEMTOR protocol always secures against. The first three (IHA, HMA, RHA) of above attacks are of this kind. No further exploration on this kind of attack is needed, as prevented by design. (ii) Attacks that can be considered relatively harmless because the behavior of the attacker is similar to the case where the attacker does not exist: The DDA and UDA attacks are of this kind. However, it should be noted that the combination of the DDA attack with the RDA or MMA attack does make a difference that can be lucrative from the point of the attacker because not supporting the propagation of an attacked nodes' description updates may certainly increase the latency with which these are eventually received by all other nodes. Especially if such late received update then has the attacking node removed from its previous trust set. (iii) Attacks that the protocol only secures against if performed by non-trusted nodes. These are given by the latter RDA and MMA attacks which are analyzed (in combination with the DDA) in more detail in the following Sections. To neutralize these attacks it is needed to identify the incorrect nodes and mark them as non-trusted. This identification could be achieved either by improving the assessment of trustability on a social layer among CN members or it could be automated with a monitoring and distributed detection protocol. For the analysis of the re-coverage performance of the SEMTOR protocol in Section 4.7.2.3 we take the availability of eventually correctly assessed trustability as given and focus on the performance of the protocol to enforce such updated assessments.

4.7.2.2 Network Attack Scenarios

In order to further explore the power of these attacks we first assume a network scenario without non-trusted nodes (thus, all nodes are trusted, whether correct or not) and therefore also without counter-measures. A base network scenario has been designed as outlined in Figure 4.5 that allows to place one or more adversaries (intermediate B-nodes) on differently "good" strategical positions for attacking the routing process between an attacked destination node DA and a given source node S. As an example, by activating, in addition to the permanent links (illustrated as solid lines), the optional $SB4$ link between node S and $B4$, any communication between nodes S and DA would fully rely on the only possible end-to-end path $S - B4 - B3 - B2 - DA$, allowing any intermediate node $B2$, $B3$, or $B4$ to easily disrupt this communication, simply by dropping any packet sent via them to DA. However, if in addition, also the optional $SA2$ link is activated, then an alternative end-to-end path from S, via $A2$ to DA emerges which is also shorter than the previous path. In this case, the position of node $B3$ is less advantageous for attacking the communication between S and DA and additional attack measures would be needed to "convince node S to send packets to DA via the longer $S - B4 - B3 - B2 - DA$ path.

Further measurements are executed as follows. At time T_0, all nodes start bootstrapping a previously configured network topology and establishing end-to-end paths to all other nodes which takes less than 40 seconds to complete. At a randomly selected time T_x between 50 and 55 seconds, node S starts sending ICMPv6 packets with an interval of 10ms to both destination nodes in parallel. At the same time all intermediate B- nodes start performing DDA&RDA-combined attacks on destination node DA.

The graphs in Figures 4.6 and 4.7 illustrate the impact of applied attacks in terms of success and latency as the duration (averaged over 5 measurement probes) between the attack activation at T_x and its successful completion. The completion is measured from two perspectives. The *path collapse time* represents the duration between T_x and the last successfully delivered packet to the corresponding destination node. If none of the packets sent by node S since T_x ever got delivered, then this duration is assumed as zero, meaning that the attack succeeds immediately. If on the other hand, since T_x, packets continued to be delivered for more than 40 seconds, then this attack is considered as failed, the evaluated collapse time is counted as infinite, and no result is printed. The *path entrapment time* represents the duration between T_x and the first packet forwarded to any of the intermediate attacking nodes (which, due to their configuration, drop the packets instead of further forwarding them). If for more than 40 seconds since T_x no packets towards an attacked destination node could be received by any of the attacking intermediate nodes, then also this duration is counted as infinite and no result is printed.

The *DA path collapse time* and *DA path entrapment time* lines in Figure 4.6 show, depending on the length of the attacking B path, the impact of a data-packet dropping attack (performed by all intermediate B-path nodes) on packets sent (by S) to destination node DA. The length of the alternative A-path between S and DA was configured to 3, 6, or 9 hops respectively for Figure 4.6a, 4.6b, or 4.6c. From the results it can be seen that this attack is only possible from nodes that constitute the shortest path between a given source and an attacked destination node. If the attack is possible, then it succeeds immediately. However, whenever an existing non-attacked path is shorter (or in general better) than the attacking path, such as in 4.6a for a B-path length of 3 or more hops, in 4.6b for a B-path length of 6 or more hops, or in 4.6c for a B-path length of 9 hops, then at least one of 5 observed attacks failed.

As a double check, also the *DB path collapse time* and *DB path entrapment time* were measured to show the impact of all intermediate A nodes attacking the routing towards destination node DB (from the same source node S) with vice verse results. For the sake of brevity these measurements are not shown here.

In summary, for DDA&RDA attacks, it can be seen that the most vulnerable scenario for the attacked destination nodes is given if the attacking nodes are in the advantageous position of a shorter or better path.

A different impact can be seen if an attacking nodes combines the DDA&RDA with the MMA attack as illustrated in Figure 4.7. For these measurements only a singe B node attacked the packet delivery to DA. For Figure 4.7a, the network scenario is, apart from the singe attacking B node, equal to that of Figure 4.6c. It can be seen that the combined attack on DA also succeeds with a 9-hops B-path that has the same length as the non-attacking A-path. This is because the attacking node, that manipulates the distance-vector metric of DAs routing updates to the best possible value so that other nodes receiving these updates get the perception that the path to DA via the attacking node has a better path metric than it actually has. This effect proves even more drastic in the scenarios used for Figures 4.7b and 4.7c. In the former DA is successfully attacked by $B8$ for a B-path length between 5 and 9 hops. It should be noted that $B8$, in case of B-path length of 5 to 7 hops, is not on any reasonable path between S and DA. In the case of the 5-hops B-path it is actually 3 hops apart from the shortest path given by S-$B5$-$B4$-$B3$-$B2$-DA (that consisting only of correct behaving nodes) and still succeeds in redirecting routes towards itself. That is because $B5$, which resides on the shortest path, is closer to the MMA-attacking node $B8$ than to the destination node DA and affected by the promising updates manipulated by $B8$. The same effect causes the successful attacks shown in Figure 4.7c where the actual shortest path is given by the varied A-path length while the B-path, where the attacking $B8$ node resides, has been fixed to 9 hops. Still B succeeds to attack DA even from a path that is up to 6 hops longer than the alternative A-path.

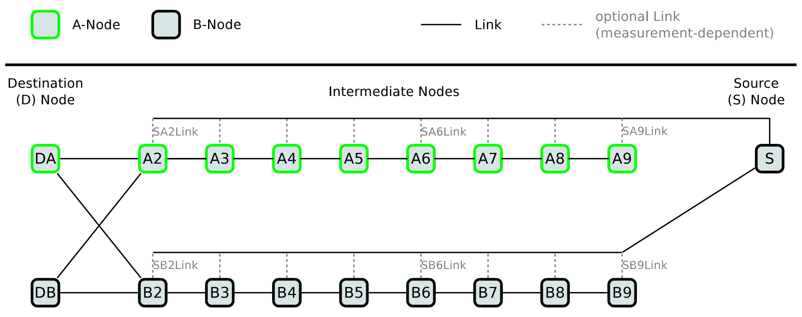
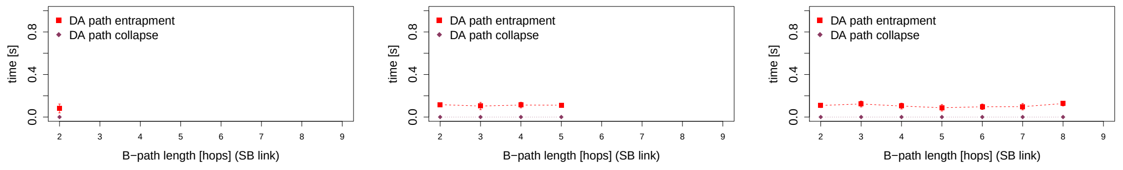


Figure 4.5: Base-network topology for validating attack vectors

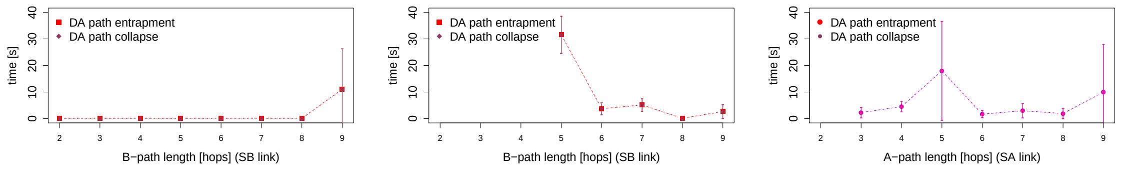


(a) Attack impact with 3 hops A path

(b) Attack impact with 6 hops A path

(c) Attack impact with 9 hops A path

Figure 4.6: Performance of route (and packet) dropping attack (RDA) performed by B nodes on DA node depending on A and B-path length



(a) Attack impact of B2 (9-hops A-path)

(b) Attack impact of B8 (9-hops A-path)

(c) Attack impact of B8 (9-hops B-path)

Figure 4.7: Performance of DDA&RDA&MMA-combined attack on DA node depending on A or B path length and adversary position

4.7.2.3 Functionality and Performance (of Eventual Consistency)

In this section we evaluate our SEMTOR implementation regarding its performance to recover, given that attacking nodes have been identified, from the most critical combination of previously introduced attacks by applying corrected trust sets that exclude all adverse nodes[6].

This attack combination is given if the adverse node(s) are placed in the advantageous position of a shorter path towards the attacked destination node, are as close as possible to the source node, and perform an DDA&RDA&MMA-combined attack. Such scenario is given for the transmission from S to DA when attacked from $B2$ in the topology illustrated in Figure 4.8. The topology extends that of Figure 4.5 with a third group and (optional) path of neutral (correct) C nodes which, in it its default configuration, corresponds (apart from the new C-nodes) with the scenario used for Figure 4.7a.

[6]Recovery performance could also be interpreted as the duration it takes to effectively exclude a set of identified adversaries from the routing towards a given destination node.

First, we deviate from this default configuration by varying the path length of correct and non-correct (attacking) A and B nodes. Later, the effect of an enabled neutral path length, as given by the intermediate C nodes, is discussed in more detail in Section 4.7.2.4. In order to capture the recovery and convergence performance of our implementation at the presence of lossy and fading links, which must be expected for any realistic WMN deployment, we also vary the broadcast packet loss of configured links between 0% (corresponding to perfect links) and up to 60%, using the Linux traffic control command mentioned earlier. However, the artificial loss is only applied to broadcast transmissions which the SEMTOR protocol uses for all its protocol data. All unicast transmission along all configured links are kept as loss free. This allows us to trace the routing decisions of the protocol by capturing the path of forwarded unicast packets such as the ICMPv6 ping request and reply packets used for probing end-to-end packet delivery between a given source-destination pair.

All B nodes perform DDA&RDA&MMA-combined attacks to disrupt packet delivery to DA. In parallel, all A nodes perform the same combination of attacks to disrupt packet delivery to DB. C nodes perform no attacks, are not attacked by any other node. C-nodes trust all other nodes but are not trusted by any A or B nodes. The S node is used as source node that sends packets to the destination nodes DA, DB, and DC in parallel with the same parametrization as given in Section 4.7.2.1 and 4.7.2.2.

We measure path entrapment in the same way as performed for Figures 4.6 and 4.7. However, once all attacked destination routes are entrapped we keep on running the emulation, wait a random period between 40-60 seconds to let protocol instances adapt to the given configuration, and then update the trust sets of DA and DB so that DA does not trust B nodes anymore (e.g. $V^t_{DA} = \{V_{DA}, V_{A2}, V_{A3}, V_{A4}, V_{A5}, V_{A6}, V_{A7}, V_{A8}, V_{A9}\}$) and DB does not trust A nodes anymore (e.g. $V^t_{DB} = \{V_{DB}, V_{B2}, V_{B3}, V_{B4}, V_{B5}, V_{B6}, V_{B7}, V_{B8}, V_{B9}\}$). Note that neither DA nor DB needs to trust S because for this scenario S is not needed to relay packets from S to DA or DB (S creates and sends packets but does not relay them). Also note that the trust set V^t_S of S is irrelevant for the delivery of packets sent by S. Such trust update actually reflects a constellation switch from Table 4.2 from b) to d). We then measure the elapsed time from the moment of the updated trust sets till the last packet from S to DA was routed via any B node and till the last packet from S to DB was routed via any A node. Therefore, tcpdump captures from the master switch, that covers all existing links, were searched for respective packets from S to DA, DB, and DC that contained a MAC address from the deprecated path. In the following Figures the average of these measures are represented as *path avoidance* latency. We also measure the elapsed time from the moment of the updated trust sets till the first packet reach their intended destination which we represented as *path recovery* latency.

For the measurements shown in Figure 4.9 the following attacks and links have been configured. From all optional SA links only the $SA9$ link is activated which enables a 9-hops A-path between source node S and the destination nodes DA, DB, and DC. From all optional SB links only the $SB2$ link is activated which enables an alternative 2-hops B-path between S and DA, DB, and DC. None of the optional SC links is activated so that no third alternative C-path exists.

Figure 4.9 indicates a clearly increasing entrapment latency for the S-DB route which is plausible given that routing updates from DB need to traverse several hops with increasing packet loss before an MMA-modified update by any A-path node could delude S to perceive the 9-hops A-path as shorter than the 2-hops B-path. The Figure also shows that the

$S - DA$ route is entrapped from the beginning, simply because the shortest path already traversed B nodes even before these started to attack the $S - DA$ routing.

The recovery and avoidance latencies of the $S - DA$ and $S - DB$ routes increase with increased protocol-data loss. The avoidance of a deprecated path is achieved faster than the convergence to the new trusted path. In the case of the $S - DA$ route the average recovery increases to up to 40 seconds which is acceptable given that routing updates along the only trusted A-path need to propagate along 9 links, each of it having a remaining success rate for propagating broadcasted protocol messages of only 40%.

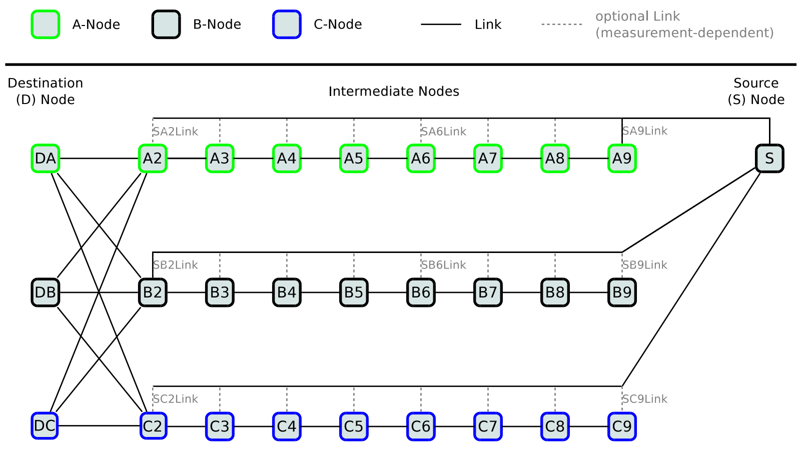


Figure 4.8: Network topology for measuring path-recovery performance

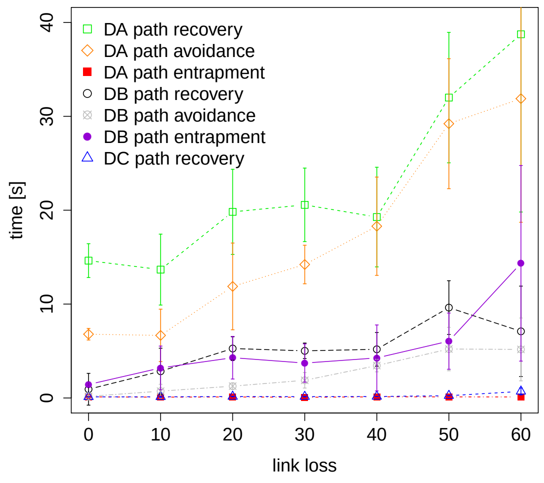


Figure 4.9: Path recovery latencies depending on average link loss

4.7.2.4 Openness, Decentrality, and Cooperativeness

In the following, the attainment of further protocol objectives and desired properties shall be illustrated with the measurements given in Figures 4.10, 4.11, and 4.12. The Figures represent the effect of path-length variations based on the topology of Figure 4.8 that extend the default scenario used for Figure 4.9 by enabling a third path of neutral C nodes.

Table 4.5: HW and OS characteristics of used target device

Characteristic	Details
Type / CPU	TP-Link TL-WR703N, Atheros AR7240@400 MHz
Wireless	AR9331, 802.11bgn 150 Mbps @100 mW
Flash / Memory	4 MB / 32 MB
Ports	100 MBit Ethernet, USB 2.0
Power supply	5 V, 100 mA, 0.5 W
Cost	approx 10 Euro (2013)
OS and distro	Linux OpenWrt/Lede (lede-17.01)
Further reading	http://wiki.openwrt.org/toh/tp-link/tl-wr703n

However, as a reference, the grey vertical bar in each Figure indicate measurements that can also be found in Figure 4.9.

Autonomy and decentrality, so to not relying on any central entity, was demonstrated indirectly via all performed experiments since none of the security enhancements provided by the gained capability for recovering functioning end-to-end routes required the existence of any central authority or entity to rely on and are all based exclusively on the individual and autonomous definition of trused nodes of each destination node.

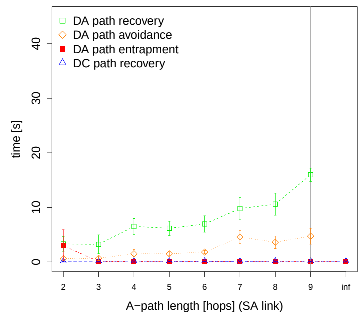
Support for openness, neutrality, and cooperativeness, in a sense that unknown nodes could join an existing network infrastructure and be neutrally supported without prior conditions, could be well observed in Figures 4.10a and 4.10b where the reachability of the DC node from S is always provided despite none of the C-nodes is ever trusted by any A or B nodes and an alternative S-DC path of intermediate C-nodes does not exist in any of the captured scenarios.

Our experiments also demonstrate how nodes benefit from the existence of paths given by unknown (so non-trusted) but correctly behaving nodes as can be seen from the measurements shown in Figures 4.10c, 4.11c, and 4.12c. Here, the existence of an alternative, although non-trusted C path between S and DA, clearly reduces the avoidance latency with which the non-trusted and DA attacking B-path nodes could be rendered as deprecated. This effect is especially shown in the presence of lossy links and short C-path length where the correctly behaving C nodes can contribute to a significantly faster propagation of DAs' description (and conained updated trust) set to S.

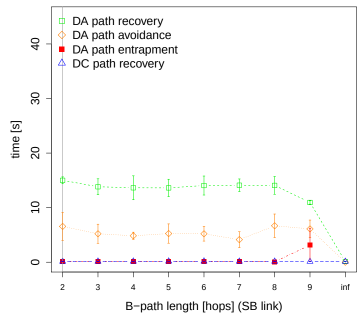
4.7.3 Validation of Resource Consumption and Scalability

To evaluate our design and implementation with respect to its performance implications and our scalability objectives outlined in Section 4.3.1, the effect of network and protocol parameters on total protocol overhead and performance has been measured by benchmarking our implementation on a cheap embedded device with hardware characteristics as summarized in Table 4.5 and that can be considered the bottom end of devices typically used within community networks [2, 23]. This device, running the Lede OS (an OpenWrt [32] based Linux system) and a cross-compiled version of our implementation, has been connected as a core node via Ethernet to the virtual network environment explained above. Figure 4.13 illustrates the setup.

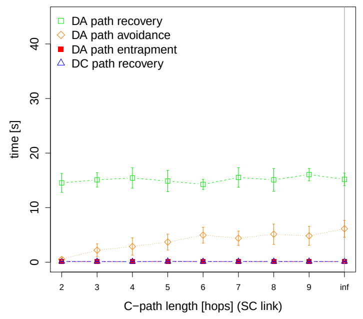
The overhead has been measured in terms of CPU usage (relative to the totally available of the node), absolute virtual memory usage (both using the Linux top command) and



(a) Performance vers. trusted path length

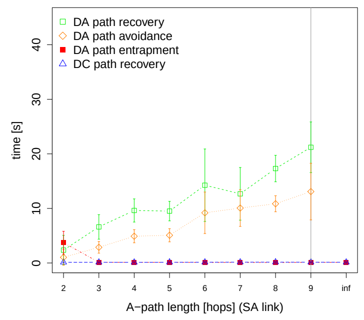


(b) Performance vers. attacked path length

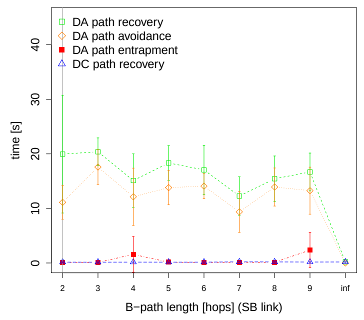


(c) Performance vers. neutral path length

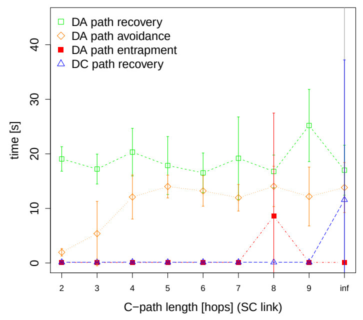
Figure 4.10: Path recovery performance depending on different physical path-lengths at 0% link loss



(a) Performance vers. trusted path length

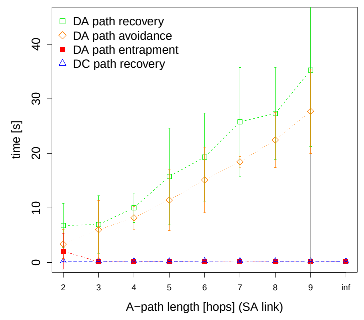


(b) Performance vers. attacked path length

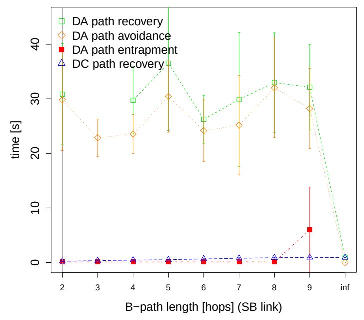


(c) Performance vers. neutral path length

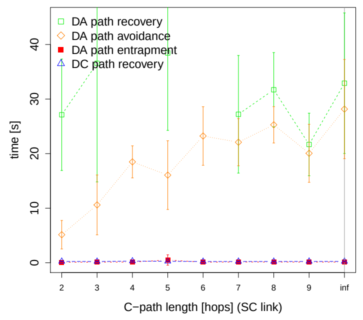
Figure 4.11: Path recovery performance depending on different physical path-lengths at 30% link loss



(a) Performance vers. trusted path length



(b) Performance vers. attacked path length



(c) Performance vers. neutral path length

Figure 4.12: Path recovery performance depending on different physical path-lengths at 50% link loss

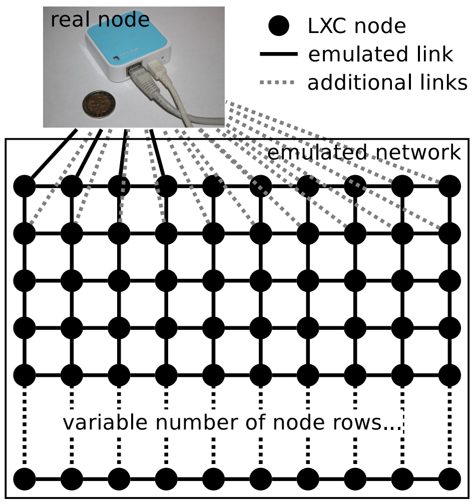


Figure 4.13: Experimentation setup consisting of benchmarked embedded router and MLC-emulated network environment

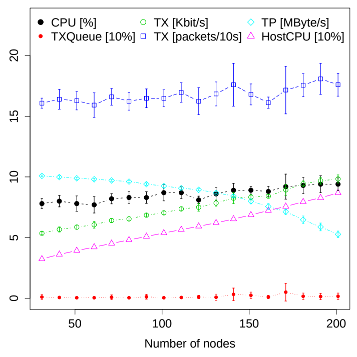
sent protocol-data (using tcpdump). Graphs in Figure 4.14,4.15 represent the results of these measurements averaged over 30 seconds. To save space and ease comparability, CPU, memory, and protocol-traffic overhead are represented in a single Figure for each analyzed input parameter. Sent (TX) Protocol-data is given per node and distinguish between sent data as Kbit/sec and sent packets per second. All measurements represent the resources consumed by the overall implementation and not just the extra overhead added over the BMX6 implementation that was used as a basis for integrating the SEMTOR mechanisms.

The impact of a node updating its description at higher (non-default) rates is illustrated in Figure 4.14e,4.15e . The measurements show that protocol-data overhead and CPU usage are indeed significantly affected by such bootstrapping events and that update intervals occurring at rates higher than twice per second have the potential to seriously harm the stability of the protocol; a finding that we'll get back to later. To avoid capturing massively parallel bootstrapping phases, an unlikely scenario for real nodes under distributed administration, further measurements have been delayed for a stabilization period of 120 seconds after each protocol-configuration change.

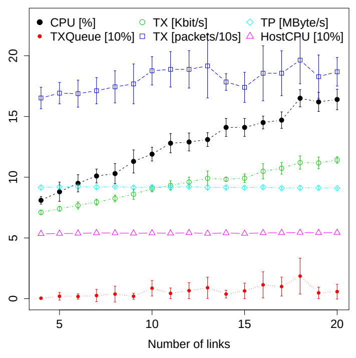
From Figures 4.14a,4.15a it can be seen that network size[7] roughly linearly affects CPU, memory, and traffic usage per node; but CPU raises slower than the others. This could be explained by the fact that the most CPU-intensive operation, being the continuous and asymmetric signing and verification of hello and routing update messages, is performed on a per-packet basis (and not on a message basis) and that almost no additional transmissions (packets) are needed to convey the increasing amount of protocol-data.

As can be seen from Figure 4.14b,4.15b , CPU and traffic overhead heavily depends on the number of links to which the test node is exposed while no additional memory requirements were observed for maintaining an increasing amount of links. This could be explained with the relatively small related memory requirements for maintaining these links compared to the allocations needed for maintaining node descriptions and thereby referenced data which must be tracked anyway, also for non-neighboring nodes.

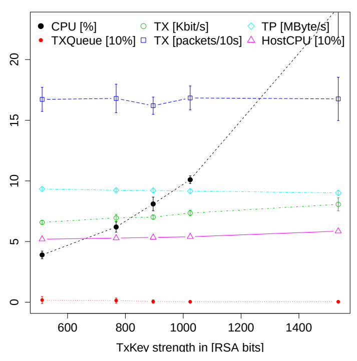
[7]Trust sets described per node were kept constant despite the varied number of participating nodes.



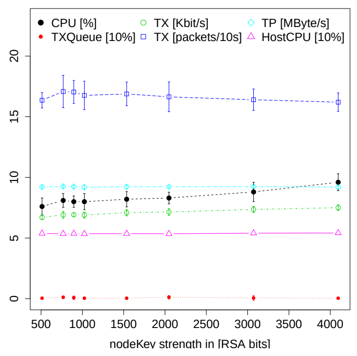
(a) Overhead vers. network size



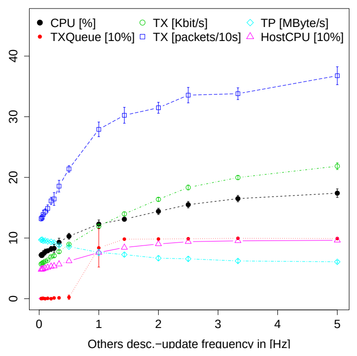
(b) Overhead vers. network density



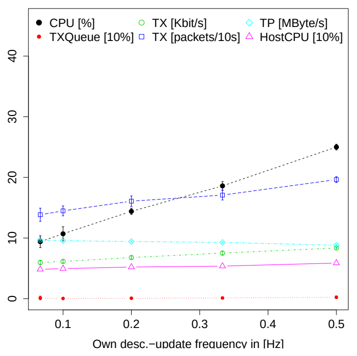
(c) Ovhd. vers. TX-signature



(d) Ovhd. vers. Node-signature



(e) Ovhd. vers. other's reconfigurations.



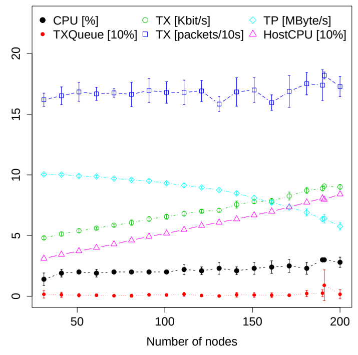
(f) Ovhd. vers. own reconfigurations.

Figure 4.14: Overhead using RSA-packet signatures depending on network and protocol characteristics

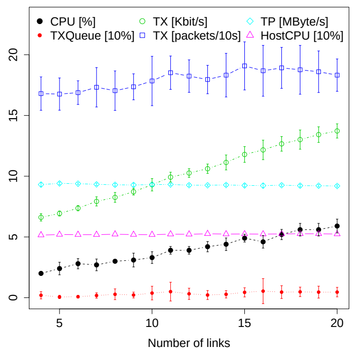
The impact of RSA transmit-signatures is illustrated in Figure 4.14c. The results confirm the expectation that stronger keys lead to linearly increasing protocol-data overhead and exponentially increasing CPU load. However, it is interesting to note that using RSA key strength such as RSA1024, which can be considered secure against factorization attacks by today and thus sufficient for the supposedly short lifetime of only few hours in our approach, for continuous signing and verification is well feasible and requires only about 8% of the target device CPU resources.

In parallel to the capturing of CPU and traffic overhead imposed by the variation of input parameters the following additional system characteristics were observed with the objective to continously validate the correct operation of the target device itself as well as the emulation environment provided by the host (desktop) machine:

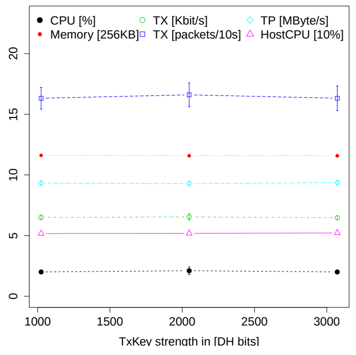
The routing performance in terms of throughput (TP) for forwarding user-data traffic, along SEMTOR-configured routes, in and out of the target device was measured by doing a TCP TP test (using iperf) between the $A1$ LXC node, the first node from the left in the top (A) row in Figure 4.13, to the $A4$ LXC node (the fourth in the same row). Since in all topological setups the target device was always linked to at least the nodes $A1$,$A2$,$A3$, and $A4$ (the first four top left nodes), the best (shortest) path between $A1$ and $A4$ always had to consist only of the target device as the only intermediate hop. To ensure that only correctly (along this best path) routed TCP traffic is considered, all TCP traffic between



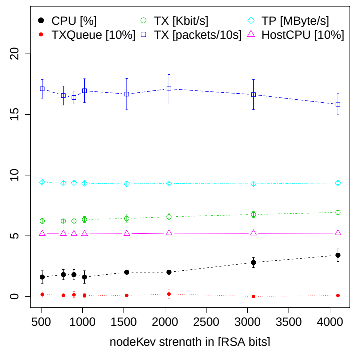
(a) Overhead vers. network size



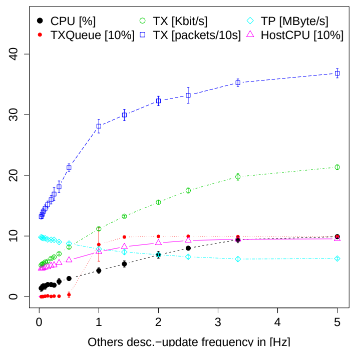
(b) Overhead vers. network density



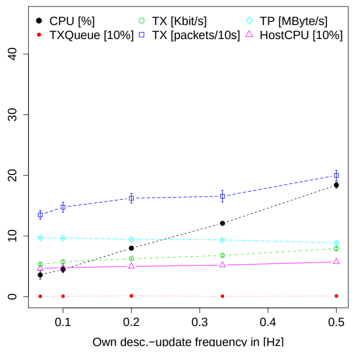
(c) Ovhd. vers. TX-signature



(d) Ovhd. vers. Node-signature



(e) Ovhd. vers. other's reconfigurations.



(f) Ovhd. vers. own reconfigurations.

Figure 4.15: Overhead using DHM-packet signatures depending on network and protocol characteristics

$A1$ and $A4$, that was not routed back and forth via the target device, got dropped using iptables.

The CPU load (in percent) relative to the total available processing capacity of the 4-core host machine (that emulated up to 200 nodes) was captured (again using top command) and represented as HostCPU. During all measurements all four host CPUs were fixed to their maximum frequency of 2.6GHz.

The fill rate (in 10 percent) of the TX-queue in the target device that is used by our implementation to throttle (via increased aggregation of scheduled messages) the transmission of protocol-data packets was captured as TXQueue.

The measurements confirm that (i) user traffic is continuously routed via the correct path and (ii) in most cases at the maximum speed the 100MBit ethernet HW is capable to process. However, a clear relation can be seen between the host CPU load and the TP which degrades when host CPU load exceeds 50% (e.g. Figures 4.14a, 4.14e, 4.15a, 4.15e) while it appears uncorrelated with CPU or protocol-traffic increase of the target device (e.g. Figures 4.14b, 4.14c, 4.14f, 4.15b, 4.15f). The sharp increase of the TX-queue fill rate in Figures 4.14e and 4.15e explains why the increase of the protocol-data overhead declines (in the same Figures). Once the fill rate reaches its maximum, scheduled messages are delayed for later aggregation, leading less totally transmitted packets. In summary, none of the observed deviations was considered critical enough to justify an invalidation of the obtained results.

4.8 Open Issues and Adjacent Solutions

In the following, routing-security aspects are discussed in the context of the objectives defined for this work and grouped into (i) those out of the scope or not considered an attack (although worth discussing), (ii) those addressed and explicitly handled, and (iii) those remaining a potential threat.

Confidentiality attacks via eavesdropping of disseminated protocols data fall into the first category. Such data includes topology, trust, identity, and if published, personal information. Particularly, the disclosure of trust sets has raised concerns[8] as its transparent dissemination allows others to deduce trust relations between the users of a community network and exploit this knowledge for attacking individuals or groups of them in higher layers (e.g. social layer). However, it should be noted that the provision of personal information via node descriptions or other parallel infrastructure services is left to the preferences of each node admin and could be omitted (thus revealing only trust relations between anonymous cryptographic identities) at the cost of complicating the task for identifying the nodes considered as trustworthy.

Malign attacks also fall into this category, being the case where a malicious node aims to influence the reputation of another node by incorrectly reporting on its negative behavior. SEMTOR does not facilitate such attacks simply because no functionality exist for detecting nor reporting behavior (good or bad) of other nodes.

This also means that **selfish, non-cooperative, and unfair behavior** is not considered by the protocol and (as yet) left to be solved independently. Here, approaches based on reputation such as [151, 152] or observed traffic validation [170] and distributed detection [174] mechanisms could be employed for detecting (groups of) faulty nodes and add them to the list of non-trusted nodes.

However, it should be noted that with the support for verifiable and dynamically-updatable node descriptions, self bootstrapping public-key infrastructure, and individually-definable trust topology, powerful tools are provided, that can be used for arguing on the trustability of nodes and enforcing individual decisions without requiring consensus among network participants.

Several **Data-plane attacks**, compromising confidentiality, authenticity, integrity, non-replication, and non-repudiation of user data are also considered as out of scope. Various existing and well established end-to-end security solution should be used instead such as TLS-based protocols like HTTPS, VPN solutions like OpenVPN or tinc, and anonymization and privacy protocols like TOR. In fact, the security aspects covered by our protocol, operating on layer 3, complement these higher-layer solutions as these build on the availability of a functioning (although best effort based) routing and data-forwarding service in the first place, partially for creating encrypted overlay networks on top of it. What these higher-layer security protocols typically require are identity- and ownership-

[8]E.g. during discussions and presentations of our approach at community events such as the International Summit for Community Wireless Networks 2013 and the Wireless Battle Mesh in Germany 2014 and Slovenia 2015.

proven addresses and the availability of a functioning end-to-end packet delivery along trusted paths established between nodes owning these addresses. Related guarantees should be provided by the **control plane** which can be attacked by disturbing the propagation of node and topology information or the forwarding of data packets. Attacks on this plane from non-trusted nodes are falling into the second category and preventing related attacks [56] (such as blackhole, partition, detour, routing-table poisoning, impersonation via replication, dropping, modification, or fabrication of protocol messages) is what the main contribution of this work is about. This is essentially achieved by excluding all non-trusted nodes from related attack-susceptible tasks such as route-discovery, establishment, and forwarding. The key enabler for this approach are the strictly non-overlapping receiver-driven responsibility for these "trusted tasks", the usage of a permanent strong asymmetric key pair (used for authenticating node identity, cryptographically generated addresses ensuring conflict-free IPv6 addresses and route ownership, trusted nodes, and secondary-key replacements) and a secondary lightweight key pair of which security relies on its short lifetime and frequent replacement (used for frequent tasks of verifying communication between neighboring nodes and to identify and discard data from non-trusted nodes).

Wormhole and Denial of Service (Dos) attacks are falling into the third category that remain unsolved with yet defined mechanisms. While several approaches, such as TIK [187] using packet leashes, or NPA [188] observing standard deviation of round trip times exist to defend against wormhole attacks, the threat of DoS attacks on the security of mesh routing protocols has received much less attention.

The susceptibility to DoS attacks is actually a pillar stemming from the open and decentralization objectives pursued with our approach that provides no means for excluding malicious or selfish nodes from exploiting or exhausting other nodes resources. For example, an adversary can fake and propagate large numbers of physically non-existing virtual node IDs with frequently updated node descriptions that can hardly be distinguished from real nodes but whose processing may easily exceed existing memory and CPU resources in real nodes. The consequences of such attack could already be seen from the measurements performed in Section 4.7.3 showing overhead depending on the number of nodes or description update frequency. To defend against such kind of attacks, without sacrificing the decentralization and open preambles defined for this work, nodes may prioritize the processing of certain known nodes and throttle the support of unknown, an approach we plan to research further in future work.

The design in Section 4.5 and experimental validation in Section 4.7 shows that: It can safely separate trusted nodes from the influence from non-trusted nodes. As illustrated in Figure 4.4, when incorrect nodes are detected by any means, if these nodes are not considered as trusted anymore and tagged as non-trusted, SEMTOR can render detected incorrect nodes as harmless to the network.

In summary, with respect to the design objectives in Section 4.3.1, SEMTOR can effectively ensure the ownership of data traffic, forwarding routes, and node identities, so that they can be attributed to exactly one node (ownership in objective 1); the ability to communicate by a contiguous group of trusting nodes, never prevented by an external (security as autonomy and robustness in objective 2); following an open and decentralized model where each node admin decides on the nodes to rely on (openness and decentralization in objective 3); in a scalable manner (scalability in objective 4). Regarding the liveness properties, *responsiveness* is satisfied by the proactiveness of the protocol and the progress as soon as correct messages with correct information are delivered. Data *delivery* is satisfied according to the conditions in the above paragraph: across trusted nodes that are expected to be trustworthy (correct). The *freshness* and *accuracy* of update and route information comes by design of the information ownership and security mechanisms, and *efficiency* of the protocol has been analyzed in detail in Section 4.7.3.

Regarding the safety properties, the *consistency* and *loop-freedom* are provided by the baseline routing algorithm, that ensures *route-safety* combined with the security and trust mechanisms for node and path information (*update-safety*). *Adversary-freedom* and *adversary-resilience* come from the separation between trusted and non-trusted nodes, particularly when trusted nodes are also trustworthy and incorrect nodes are or become non-trusted when detected.

4.9 Conclusions and Future Work

In this work we pointed to new trust and security requirements arising for open and decentralized networks such as community networks. We described SEMTOR, a novel routing protocol that can be used for satisfying these demands. SEMTOR allows the verifiable and undeniable definition and distributed application of user-individual trust topologies for routing traffic towards each node. One particular advantage of SEMTOR is that it does not require a global consensus on the trustiness of any node. This gives each node admin the freedom to individually define the subset (and resulting sub topology) from the whole set of participating nodes that he considers sufficient trustworthy to meet their security and data-delivery objectives and concerns.

Addressed security aspects have been discussed and put in context with related and orthogonal security solutions and how these can benefit by building on it. The proposed mechanisms have been implemented, tested and evaluated for their correctness and performance to exclude non-trusted nodes from the network. Therefore safety and liveness properties that are guaranteed by our protocol have been identified and proven with formal reasoning. The scalability of our implementation has been evaluated regarding network size, density, reconfiguration dynamics, and strength of used crypto parameters. Measurement results, obtained via benchmarking on low-end embedded hardware, show scalability limits and parameters with significant impact on performance and overhead. The results also show that the usage of strong asymmetric cryptography for building trusted routing-topologies is possible; even given the scalability requirements imposed by environment and characteristics of nowadays community-networks. In the future, we plan to research further the impact and challenges due to the presence of large numbers of anonymous nodes and denial of service attacks.

Chapter 5

Conclusions

Community networks, collectively created and maintained by citizens, allow for the establishment of network infrastructures that extend far beyond what could be reached with the capabilities of the participating individuals alone. Such initiatives can enable the affordable provisioning of intra- and inter-net services based on the cooperation among routing processes that are distributed across the nodes of a network. In this work we have addressed the challenging requirements for the routing in community networks, which must provide open, decentralized, and efficient mechanisms that scale with the characteristics of such networks while being able to respect the individual and potentially controversial performance, service, and security demands of their participants.

Although widely applicable, we have considered the scope of routing protocols for wireless mesh network (WMNs) deployments in community networks. By analyzing technologies and existing deployments of CNs based on field studies and comparison with related work from the literature we identify technological and topological properties and dynamics, achieved performance, and shortcomings of such networks. Our contributions, stemming from these studies, include detailed observations of network structure such as sizes and densities and its evolution over time. We analyze the implications, efficiency, and usage of these networks. We look at the distribution and usage of links, paths, and gateway nodes and compare the throughput experienced by the users with the theoretical achievable path capacities based on a conflict-graph model that considers the impact of interfering links. The results show benefits of WMNs based on licence-free MIMO technology and mesh-routing protocols, particular in terms of autonomy, resilience to outages, and performance, and that routes chosen by the used BMX6 routing protocol typically conform also with the best theoretically existing paths.

In a separate effort we evaluate the scalability, performance, and stability of the three favorite proactive mesh-routing protocol implementations of BMX6, OLSR, and Babel. Our results, based on emulation and testbed-based experimentation, show the relative merits, costs, and limitations of these protocols such as the low resource footprint of Babel given a stable and low-density network which, however, outperformed by OLSR and especially BMX6, once network size or density increases or varies.

Our first contribution in Chapter 2, with the characterization and performance analysis in the scope of wireless mesh networks used in community networks, and our second contribution in Chapter 3, with the evaluation of the most widely used routing protocols and implementations in this scope, constitute our answer to the research question RQA.

Building upon the identified characteristics of and the requirements for CN deployments we contribute with the development and evaluation of novel routing protocol mechanisms that can enhance the openness, decentralization, neutrality, autonomy, and security of CNs. This is essentially achieved by enabling the verifiable definition and distributed application of individually trusted topologies for routing traffic with respect to the data-delivery concerns of each traffic-owning user.

Security achievements have been analyzed regarding the correctness of specified guarantees and the observed performance with which our open-source implementation achieves to encounter attacks of wrongly-trusted adversaries by propagating and applying respectively-updated trust definitions. Scalability has been analyzed with respect to network characteristics and dynamics that are typical for existing WMN deployments, but also with respect to new parameters such as the strength of used crypto parameters or the frequency of reconfigured trustiness. These results also show that our approach, based on asymmetric cryptography, is feasible, even when executed on embedded-router hardware (10 Euro, Linux SoC) that, due to its very limited computation capabilities, can be considered the bottom end of what is commonly used in wireless CNs. This third contribution constitutes our answer to the research question RQB.

Together, with these contributions and answers, we can conclude that we provide a satisfactory answer to the overall research question of how to cooperate decentralized, efficiently and reliably on the collaborative provisioning of routing services in open and heterogeneous community networks.

5.1 Future Directions

Based on the insights gained through the evaluation of considerable scenarios and proposed solution for routing in open and decentralized CNs, remaining challenges (and possible approaches) have been identified whose further elaboration and the development of effective (counter) measures would be interesting for the future sustainability of CNs.

Our measurements identified the susceptibility to DoS attacks, where the overly exhaustion of certain processing resources caused by an adversary node may affect the liveness properties guaranteed by our protocol, which depend on the responsiveness of correctly behaving nodes. For example, an adversary may fake and propagate large number of physically non-existing virtual node IDs with frequently updated node descriptions that can hardly be distinguished from real nodes but whose processing may exceed the memory and CPU resources in real nodes. To defend against such kind of attacks, without sacrificing the decentralization and open preambles defined for this work, future research can explore how nodes can prioritize the processing of information coming from already known (as indeed existing) nodes and throttle the support of unknown nodes.

We classified nodes as trusted versus non-trusted, and correct versus non-correct, and achieved the establishment of adversary-free routes based on the general assumption that incorrectly behaving nodes are not trusted. However, in cases where a correct attribution of these properties is not easily achievable (for example based on the personal recognition among node administrators of a CN) without discarding an overly large fraction of correct nodes as non-trusted, distributed monitoring and detection mechanisms would be needed to identify and continuously verify the actual behavior of nodes. Future research can explore the integration of existing approaches to this problem with the response mechanisms provided by our work and further evaluate the performance of the resulting system.

References

1. Neumann, A., Lopez, E. & Navarro, L. *An evaluation of BMX6 for community wireless networks* in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on* (Oct. 2012), 651–658 (cit. on pp. viii, 8, 14, 24, 28, 29, 50, 73).
2. Cerdà-Alabern, L., Neumann, A. & Escrich, P. *Experimental Evaluation of a Wireless Community Mesh Network* in *Proceedings of the 16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (ACM, Barcelona, Spain, 2013), 23–30 (cit. on pp. viii, 8, 28, 29, 35, 99, 107).
3. Cerdà-Alabern, L., Neumann, A. & Escrich, P. *Experimental Evaluation of Wireless Mesh Networks: A Case Study and Comparison* in *NICST'2013, New and smart Information Communication Science and Technology to support Sustainable Development* (Clermont Ferrand, France, Sept. 2013) (cit. on pp. viii, 8, 28, 51, 75).
4. Neumann, A., Navarro, L., Baig, R. & Escrich, P. *Receiver-driven routing for community mesh networks* in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a* (June 2013), 1–7 (cit. on pp. viii, 9, 71, 74, 80, 86).
5. Cerdà-Alabern, L., Neumann, A. & Maccari, L. *Experimental Evaluation of BMX6 Routing Metrics in a 802.11an Wireless-Community Mesh Network* in *2015 3rd International Conference on Future Internet of Things and Cloud* (Aug. 2015), 770–775 (cit. on pp. viii, 8, 29, 89).
6. Centelles, R. P., Oncins, V. & Neumann, A. Enhancing reflection and self-determination in a real-life community mesh network. *Computer Networks* **93, Part 2.** Community Networks, 297–307 (2015) (cit. on pp. viii, 8, 86).
7. Neumann, A., Baig, R. & Oncins, V. *Mobility performance Evaluation of Mesh rOuting protocols (MEMO)* tech. rep. (Fed4FIRE, Dec. 2014) (cit. on pp. viii, 8, 50).
8. Neumann, A., López, E. & Navarro, L. Evaluation of mesh routing protocols for wireless community networks. *Computer Networks* **93, Part 2.** Community Networks, 308–323 (2015) (cit. on pp. viii, 8, 71).
9. Neumann, A., Lopez, E., Cerdà-Alabern, L. & Navarro, L. *Securely-entrusted multi-topology routing for community networks* in *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)* (Jan. 2016), 1–8 (cit. on pp. viii, 9).
10. Neumann, A., Navarro, L. & Cerdà-Alabern, L. Enabling Individually Entrusted Routing Security for Open and Decentralized Community Networks. *Under review for Ad Hoc Networks* (2017) (cit. on pp. viii, 9).

11. Neumann, A., Vilata, I., León, X., Escrich Garcia, P., Navarro, L. & López, E. *Community-Lab: Architecture of a Community Networking Testbed for the Future Internet* in *1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB'2012)* (Barcelona, Spain, Oct. 2012), 628–635 (cit. on pp. ix, 74).

12. Navarro, L., Escrich, P., Baig, R. & Neumann, A. *Community-Lab: Overview and Invitation to the Research Community* in *IEEE International Conference on Peer-to-Peer Computing* (IEEE Press, Tarragona, Sept. 2012) (cit. on p. ix).

13. Escrich, P., Baig, R., Vilata, I., Neumann, A., Aymerich, M., Lopez, E., Vega, D., Meseguer, R., Freitag, F. & Navarro, L. *Community home gateways for P2P clouds* in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (Sept. 2013), 1–2 (cit. on p. ix).

14. Escrich, P., Baig, R., Dimogerontakis, E., Carbo, E., Neumann, A., Fonseca, A., Freitag, F. & Navarro, L. *WiBed, a platform for commodity wireless testbeds* in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on* (Oct. 2014), 85–91 (cit. on pp. ix, 11).

15. Neumann, A. *"BMX6 Mesh Routing Protocol, Reclaim Your Mesh – Self-paced Networking in Multilateral Environments"* International Summit for Community Wireless Networks (IS4CWN). Berlin/Germany, 2013 (cit. on p. ix).

16. Neumann, A. *"Trusted Multi-Topology Routing with BMX6"* Wireless Battle of the Mesh version 7 (WBMv7). Leipzig/Germany, May 2014 (cit. on p. ix).

17. Neumann, A. *"Individually-Secured Mesh Routing with BMX7: Advances and Integration"* Wireless Battle of the Mesh version 9 (WBMv9). Porto/Portugal, May 2016 (cit. on p. ix).

18. Neumann, A. *"Secure and Decentralized Distance-Vector Routing with BMX7"* Seminar at Berlin Freifunk Community. Berlin/Germany, July 2016 (cit. on p. ix).

19. Baig, R., Roca, R., Freitag, F. & Navarro, L. Guifi.Net, a Crowdsourced Network Infrastructure Held in Common. *Comput. Netw.* **90,** 150–165 (Oct. 2015) (cit. on pp. 1, 2, 24, 71, 76).

20. Navarro, L., Baig, R., Freitag, F., Dimogerontakis, E., Treguer, F., Dulong de Rosnay, M., Maccari, L., Micholia, P. & Antoniadis, P. *Report on the Existing CNs and their Organization (v2)* 2016 (cit. on p. 2).

21. Szabó, C., Farkas, K. & Horváth, Z. Motivations, Design and Business Models of Wireless Community Networks. *Mob. Netw. Appl.* **13,** 147–159 (Apr. 2008) (cit. on pp. 2, 4, 24, 35, 73, 76).

22. Pathak, P. & Dutta, R. A Survey of Network Design Problems and Joint Design Approaches in Wireless Mesh Networks. *Communications Surveys Tutorials, IEEE* **13,** 396–428 (Mar. 2011) (cit. on pp. 2, 71).

23. Avonts, J., Braem, B. & Blondia, C. *A questionnaire based examination of community networks* in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on* (Oct. 2013), 8–15 (cit. on pp. 2, 51, 71, 75, 99, 107).

24. Vega, D., Baig, R., Cerdà-Alabern, L., Medina, E., Meseguer, R. & Navarro, L. A technological overview of the guifi.net community network. *Computer Networks* **93, Part 2.** Community Networks, 260–278 (2015) (cit. on pp. 2, 3, 24, 28, 99).

25. *Pico peering agreement* `http://picopeer.net` (cit. on pp. 2, 35, 71, 76).

26. Oliver, M., Zuidweg, J. & Batikas, M. *Wireless Commons against the digital divide* in *2010 IEEE International Symposium on Technology and Society* (June 2010), 457–465 (cit. on pp. 2, 14, 24, 71, 73, 76).

27. Evans, N. S. *Methods for Secure Decentralized Routing in Open Networks* MA thesis (Technische Universität München, Garching bei München, Aug. 2011), 234 (cit. on pp. 2, 76).

28. Maccari, L. *An analysis of the Ninux wireless community network* in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (Oct. 2013), 1–7 (cit. on pp. 2, 35, 76).

29. Vasilakis, G., Perantinos, G., Askoxylakis, I. G., Mechin, N., Spitadakis, V. & Traganitis, A. *Business opportunities and considerations on wireless mesh networks* in *2009 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops* (June 2009), 1–8 (cit. on pp. 4, 76).

30. Bina, M. *Wireless Community Networks: A Case of Modern Collective Action* PhD thesis (Athens University of Economics and Business, June 2007) (cit. on pp. 4, 76).

31. *Quick Mesh Project* `http://qmp.cat` (cit. on pp. 9, 10, 14, 28, 36).

32. *Lede/OpenWrt Linux distribution for embedded devices* `http://lede-project.org` and `http://openwrt.org` (cit. on pp. 9, 14, 38, 107).

33. *Libre-Mesh* `http://libre-mesh.org` (cit. on pp. 9, 51).

34. *Wireless Battlemesh* `http://battlemesh.org/` (cit. on pp. 9–11, 50, 51).

35. *GuifiSants* `http://sants.guifi.net/` (cit. on pp. 10, 40).

36. *Open, Free and Neutral Network Internet for everybody* `http://guifi.net/en` (cit. on pp. 10, 14, 40, 58).

37. *FreiFunk – a Community Network* `http://start.freifunk.net` (cit. on pp. 10, 36).

38. *BMX6 mesh networking protocol* `http://bmx6.net`. July 2014 (cit. on pp. 10, 35, 40, 49, 71, 98, 99).

39. Axel Neumann. *Investigating Routing-Protocol Characteristics with Mesh Linux Containers (MLC)* Workshop, UPC, Barcelona, Spain. `https://raw.github.com/axn/mlc`. Nov. 2011 (cit. on pp. 11, 56, 99).

40. *Community Networks testbed for Future Internet (CONFINE)* European Commission FP7, Future Internet Research and Experimentation Initiative (FIRE). `http://confine-project.eu` (cit. on p. 11).

41. Cerdà-Alabern, L. *On the Topology Characterization of Guifi.net* in *2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2012)* (Barcelona, Spain, Oct. 2012), 389–396 (cit. on pp. 14, 24, 26, 28, 56, 58, 75).

42. Vega, D., Cerdà-Alabern, L., Navarro, L. & Meseguer, R. *Topology patterns of a community network: Guifi.net* in *1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB'2012)* (Barcelona, Spain, Oct. 2012), 612–619 (cit. on pp. 14, 19, 24, 26, 28, 30, 35, 75).

43. Braem, B., Blondia, C., Barz, C., Rogge, H., Freitag, F., Navarro, L., Bonicioli, J., Papathanasiou, S., Escrich, P., Baig Viñas, R., *et al.* A case for research with and on community networks. *ACM SIGCOMM Computer Communication Review* **43,** 68–73 (2013) (cit. on p. 14).

44. *qMp Sants-UPC monitoring web page* `http://compnet.ac.upc.edu/~llorenc/qmpsantsupc/index.php` (cit. on p. 18).

45. Cerdà-Alabern, L. *Analysis of Guifi.net's Topology, Extension of Results* tech. rep. UPC-DAC-RR-2012-15. `http://www.ac.upc.edu/app/research-reports/html/2012/15/guifinet-topology-extension.pdf` (May 2012) (cit. on p. 19).

46. *Netperf benchmark* `http://www.netperf.org/netperf` (cit. on p. 22).

47. *Linux utility for wireless devices* `http://wireless.kernel.org/en/users/Documentation/iw` (cit. on p. 23).

48. Johnson, D., Ntlatlapa, N. & Aichele, C. *A simple pragmatic approach to mesh routing using BATMAN* in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries* (2008), 1–10 (cit. on pp. 24, 49, 73).

49. Zakrzewska, A., Koszalka, L. & Pozniak-Koszalka, I. *Performance Study of Routing Protocols for Wireless Mesh Networks* in *Proceedings of the 2008 19th International Conference on Systems Engineering* (IEEE Computer Society, Washington, DC, USA, 2008), 331–336 (cit. on pp. 24, 49, 73).

50. Ashraf, U., Juanole, G. & Abdellatif, S. *Evaluating Routing Protocols for the Wireless Mesh Backbone* in *Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications* (IEEE Computer Society, Washington, DC, USA, 2007), 40–48 (cit. on pp. 24, 49, 73).

51. Abolhasan, M., Hagelstein, B. & Wang, J. C.-P. *Real-world performance of current proactive multi-hop mesh protocols* in *Proceedings of the 15th Asia-Pacific conference on Communications* (IEEE Press, Shanghai, China, 2009), 42–45 (cit. on pp. 24, 73).

52. Murray, D., Dixon, M. & Koziniec, T. *An experimental comparison of routing protocols in multi hop ad hoc networks* in *Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian* (Oct. 2010), 159–164 (cit. on pp. 24, 49–51, 71, 73).

53. Hong, C. S. & Siddiqui, M. S. *Security Issues in Wireless Mesh Networks* in *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)* **00** (IEEE Computer Society, Los Alamitos, CA, USA, 2007), 717–722 (cit. on pp. 24, 73).

54. Shen, S., Huang, Y. & Ding, J. A cross-layer design for heterogeneous routing in wireless mesh networks. *International Journal of Pervasive Computing and Communications* **4,** 40–49 (2008) (cit. on pp. 24, 74).

55. Soni, D. M. K., Suri, P. P. & Tomar, P. *A Comparative study for Secure Routing in MANET* in. **4.** Published By Foundation of Computer Science (July 2010), 17–22 (cit. on p. 24).

56. Abusalah, L., Khokhar, A. & Guizani, M. A survey of secure mobile Ad Hoc routing protocols. *Communications Surveys Tutorials, IEEE* **10,** 78–93 (Apr. 2008) (cit. on pp. 24, 73, 113).

57. Bicket, J., Aguayo, D., Biswas, S. & Morris, R. *Architecture and evaluation of an unplanned 802.11 b mesh network* in *Proceedings of the 11th annual international conference on Mobile computing and networking, MobiCom '05* (Cologne, Germany, 2005), 31–42 (cit. on pp. 24, 25, 28, 29).

58. Camp, J., Robinson, J., Steger, C. & Knightly, E. *Measurement driven deployment of a two-tier urban mesh access network* in *Proceedings of the 4th international conference on Mobile systems, applications and services* (2006), 96–109 (cit. on pp. 24, 25, 27–29).

59. Chebrolu, K., Raman, B. & Sen, S. *Long-distance 802.11b links: Performance Measurements and Experience* in *in Proceedings of ACM Mobicom* (2006) (cit. on p. 24).

60. Brik, V., Rayanchu, S., Saha, S., Sen, S., Shrivastava, V. & Banerjee, S. *A measurement study of a commercial-grade urban wifi mesh* in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement* (2008), 111–124 (cit. on pp. 24, 25, 27, 28).

61. LaCurts, K. & Balakrishnan, H. *Measurement and Analysis of Real-world 802.11 Mesh Networks* in *Proceedings of the 10th Annual Conference on Internet Measurement (ACM)* (Melbourne, Australia, 2010), 123–136 (cit. on pp. 24, 26, 28, 73).

62. Afanasyev, M., Chen, T., Voelker, G. M. & Snoeren, A. C. *Analysis of a Mixed-Use Urban Wifi Network: When Metropolitan becomes Neapolitan* in *In ACM/USENIX IMC* (2008) (cit. on pp. 24, 28).

63. Afanasyev, M., Chen, T., Voelker, G. & Snoeren, A. Usage patterns in an urban WiFi network. *IEEE/ACM Transactions on Networking* **18,** 1359–1372 (2010) (cit. on pp. 24–26, 28).

64. Bychkovsky, V., Hull, B., Miu, A., Balakrishnan, H. & Madden, S. *A measurement study of vehicular internet access using in situ Wi-Fi networks* in *Proceedings of the 12th annual international conference on Mobile computing and networking* (2006), 50–61 (cit. on p. 24).

65. Robinson, J., Swaminathan, R. & Knightly, E. W. *Assessment of urban-scale wireless networks with a small number of measurements* in *The 14th ACM international conference on Mobile computing and networking* (ACM, San Francisco, California, USA, 2008), 187–198 (cit. on p. 24).

66. Aguayo, D., Bicket, J., Biswas, S., Judd, G. & Morris, R. Link-level measurements from an 802.11 b mesh network. *ACM SIGCOMM Computer Communication Review* **34,** 121–132 (2004) (cit. on pp. 24, 25, 28).

67. Camp, J., Mancuso, V., Gurewitz, O. & Knightly, E. W. *A measurement study of multiplicative overhead effects in wireless networks* in *in Proceedings of IEEE INFOCOM* (Uppsala, Sweden, Apr. 2008) (cit. on pp. 24, 25, 27, 28).

68. Vural, S., Wei, D. & Moessner, K. Survey of Experimental Evaluation Studies for Wireless Mesh Network Deployments in Urban Areas Towards Ubiquitous Internet. *Communications Surveys Tutorials, IEEE* **15,** 223–239 (2013) (cit. on pp. 24, 29, 75).

69. Camponovo, G. & Cerutti, D. *WLAN communities and Internet access sharing: a regulatory overview* in *Mobile Business, 2005. ICMB 2005. International Conference on* (2005), 281–287 (cit. on pp. 24, 73).

70. Frangoudis, P. A., Polyzos, G. C. & Kemerlis, V. P. Wireless community networks: an alternative approach for nomadic broadband network access. *IEEE Communications Magazine* **49,** 206–213 (2011) (cit. on pp. 24, 35, 73).

71. Cao, Y., Krebs, M., Toubekis, G. & Makram, S. *Mobile Community Information Systems on Wireless Mesh Networks-An Opportunity for Developing Countries and Rural Areas* in *Fifth International Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS'07)* (2007), 11–12 (cit. on pp. 24, 73).

72. Daneels, G. *Analysis of the BMX6 routing protocol* Master in Computer Science thesis (University of Antwerp. Faculty of Science. Mathematics-Computer Science Department, 2013) (cit. on pp. 29, 35).

73. Jain, K., Padhye, J., Padmanabhan, V. N. & Qiu, L. Impact of interference on multi-hop wireless network performance. *Wireless networks* **11,** 471–487 (2005) (cit. on pp. 29, 30, 34).

74. Houaidia, C., Van Den Bossche, A., Idoudi, H., Val, T. & Saidane, L. A. *Experimental performance analysis of routing metrics in Wireless Mesh Networks* in *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International* (2013), 1011–1016 (cit. on p. 29).

75. De Couto, D. S., Aguayo, D., Bicket, J. & Morris, R. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks* **11,** 419–434 (2005) (cit. on pp. 29, 36, 54).

76. Draves, R., Padhye, J. & Zill, B. *Routing in multi-radio, multi-hop wireless mesh networks* in *10th annual international conference on Mobile computing and networking* (2004), 114–128 (cit. on pp. 29, 36).

77. Barz, C., Fuchs, C., Kirchhoff, J., Niewiejska, J. & Rogge, H. OLSRv2 for Community Networks: Using Directional Airtime Metric with external radios. *Computer Networks* **93.** Community Networks, 324–341 (2015) (cit. on pp. 29, 36).

78. Maccari, L. & Lo Cigno, R. A week in the life of three large Wireless Community Networks. *Ad Hoc Networks. Modeling and Performance Evaluation of Wireless Ad-Hoc Networks* **24, Part B,** 175–190 (Jan. 2015) (cit. on pp. 30, 35).

79. Gupta, P. & Kumar, P. R. The capacity of wireless networks. *Information Theory, IEEE Transactions on* **46,** 388–404 (2000) (cit. on p. 30).

80. Kodialam, M. & Nandagopal, T. *Characterizing the capacity region in multi-radio multi-channel wireless mesh networks* in *11th annual international conference on Mobile computing and networking* (2005), 73–87 (cit. on p. 30).

81. Ding, Y. & Xiao, L. Channel allocation in multi-channel wireless mesh networks. *Computer Communications* **34,** 803–815 (2011) (cit. on p. 30).

82. Yang, D., Fang, X. & Xue, G. *Channel allocation in non-cooperative multi-radio multi-channel wireless networks* in *INFOCOM, 2012 Proceedings IEEE* (2012), 882–890 (cit. on p. 30).

83. Zubow, A. & Sombrutzki, R. *Adjacent channel interference in IEEE 802.11n* in *2012 IEEE Wireless Communications and Networking Conference (WCNC)* (Apr. 2012), 1163–1168 (cit. on p. 32).

84. Lakshmanan, S., Lee, J., Etkin, R., Lee, S.-J. & Sivakumar, R. *Realizing high performance multi-radio 802.11n wireless networks* in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on* (IEEE, 2011), 242–250 (cit. on p. 32).

85. *The Compact for a Free, Open & Neutral Network (FONN Compact)* `http://guifi.net/en/FONNC` (cit. on p. 35).

86. *Network Monitoring Tools* Stanford linear acceleration center (SLAC) and Internet End-to-end Performance Monitoring (IEPM). `http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html`. Oct. 2012 (cit. on p. 35).

87. *MOMENT: Monitoring and Measurement in the Next generation Technologies* Deliverable 2.1: Analysis of monitoring services, Euorepean commission community research, seventh framework programme. `http://telscom.ch/wp-content/uploads/Moment/Deliverables/FP7-MOMENT-WP2-D2.1.pdf`. Mar. 2008 (cit. on p. 35).

88. Parissidis, G., Karaliopoulos, M., Baumann, R. & Spyropoulos, T. *Routing metrics for wireless mesh networks* in *In Guide to Wireless Mesh Networks* (Springer: London, UK, 2009) (cit. on pp. 35, 36).

89. *Guifi.net network Maps* `http://guifi.net/en/guifi_zones` (cit. on p. 36).

90. *Wireless Nodes Database* `https://github.com/wind-project` (cit. on p. 36).

91. *Nodeshot* `https://github.com/ninuxorg/nodeshot` (cit. on p. 36).

92. *Freimap* `http://wiki.freifunk.net/Java_Freimap_Projekt` (cit. on p. 36).

93. *NodeWatcher* `http://dev.wlan-si.net/wiki/Nodewatcher` (cit. on p. 36).

94. *Comunitat GràciaSenseFils* `http://graciasensefils.net` (cit. on p. 36).

95. *Lugro Mesh* `http://www.lugro-mesh.org.ar/en/` (cit. on p. 36).

96. *Commotion Wireless* `http://commotionwireless.net` (cit. on p. 36).

97. *OpenStreetMaps* `http://openstreetmap.org` (cit. on p. 36).

98. T. Clausen, P. Jacquet. *Optimized Link State Routing Protocol (OLSR)* RFC 3626 (Experimental). Internet Engineering Task Force, 2003 (cit. on pp. 36, 71).

99. *olsrd - an adhoc wireless mesh routing daemon* `http://olsrd.org` (cit. on pp. 37, 49, 54, 71).

100. C. Dearlove, T. Clausen. *Optimized Link State Routing Protocol Version 2 (OLSRv2) and MANET Neighborhood Discovery Protocol (NHDP) Extension TLVs* RFC 7722 (Standards Track). Internet Engineering Task Force, 2014 (cit. on pp. 37, 71).

101. Juliusz Chroboczek. *The Babel Routing Protocol* RFC 6126 (Experimental). Internet Engineering Task Force, 2011 (cit. on pp. 37, 49, 51, 71).

102. *Babel – a loop-avoiding distance-vector routing protocol* `http://www.pps.univ-paris-diderot.fr/~jch/software/babel/` (cit. on pp. 37, 71).

103. *Lua – The Programming Language Lua* `http://www.lua.org` (cit. on p. 37).

104. *Ubus - OpenWrt micro bus architecture* `http://wiki.openwrt.org/doc/techref/ubus` (cit. on p. 38).

105. *NCD repository* `http://dev.qmp.cat/projects/nc/repository` (cit. on p. 40).

106. *Routek S.L. - Pla de Barris* `http://routek.net/pladebarris.html` (cit. on p. 41).

107. *Pla de Barris del Raval Sud* `http://www.ravalsudpladebarris.cat/` (cit. on p. 41).

108. *EXO.cat - Associació Expansió de la Xarxa Oberta* `http://exo.cat` (cit. on p. 41).

109. Feng, Q., Cai, Z., Yang, J. & Hu, X. *A Performance Comparison of the Ad Hoc Network Protocols* in *IEEE WCSE 2009* (), 293–297 (cit. on p. 49).

110. Mughal, F. Z. *Comparative analysis of proactive, reactive and hybrid ad hoc routing protocols in Client based Wireless Mesh Network* in *IEEE ICIET 2010* (), 1–6 (cit. on p. 49).

111. Kumar, S. & Sengupta, J. *AODV and OLSR routing protocols for Wireless Ad-hoc and Mesh Networks* in *IEEE ICCCT 2010* (), 402–407 (cit. on p. 49).

112. Zakaria, A., Mohamad, H., Ramli, N. & Ismail, M. *Performance Evaluation of Routing Protocols in Wireless Mesh Networks* in *IEEE ICACT 2013* (), 1111–1115 (cit. on p. 49).

113. Ikeda, M., Kulla, E., Hiyama, M., Barolli, L. & Takizawa, M. *Experimental Results of a MANET Testbed in Indoor Stairs Environment* in *IEEE AINA 2011* (), 779–786 (cit. on pp. 49, 50).

114. Borgia, E. *Experimental Evaluation of Ad Hoc Routing Protocols* in *IEEE PerCom 2005 Workshops* (2005), 232–236 (cit. on p. 49).

115. Rastogi, D., Ganu, S., Zhang, Y., Trappe, W. & Graff, C. *A Comparative Study of AODV and OLSR on the ORBIT Testbed* in *IEEE MILCOM 2007* (), 1–7 (cit. on p. 49).

116. Friginal, J., de Andrés, D., Ruiz, J.-C. & Gil, P. Towards benchmarking routing protocols in wireless mesh networks. *Ad Hoc Netw.* **9,** 1374–1388 (Nov. 2011) (cit. on pp. 49, 50, 74).

117. Bai, F., Sadagopan, N. & Helmy, A. *IMPORTANT: a framework to systematically analyze the Impact of Mobility on Performance of Routing Protocols for Adhoc Networks* in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)* **2** (Mar. 2003), 825–835 vol.2 (cit. on p. 50).

118. Shah, N., Qian, D. & Iqbal, K. *Performance evaluation of multiple routing protocols using multiple mobility models for mobile ad hoc networks* in *2008 IEEE International Multitopic Conference* (Dec. 2008), 243–248 (cit. on p. 50).

119. Maan, F. & Mazhar, N. *MANET routing protocols vs mobility models: A performance evaluation* in *IEEE ICUFN 2011* (), 179–184 (cit. on p. 50).

120. Javaid, N., Yousaf, M., Ahmad, A., Naveed, A. & Djouani, K. *Evaluating impact of mobility on wireless routing protocols* in *IEEE ISWTA 2011* (), 84–89 (cit. on p. 50).

121. Huang, Y., Bhatti, S. & Parker, D. *Tuning OLSR* in *IEEE PIMRC 2006* (), 1–5 (cit. on p. 50).

122. Azzuhri, S., Portmann, M. & Tan, W. L. *Evaluating the performance impact of protocol parameters on ad-hoc network routing protocols* in *IEEE ATNAC 2012* (), 1–6 (cit. on p. 50).

123. Fang, J., Goff, T. & Pei, G. *Comparison studies of OSPF-MDR, OLSR and Composite Routing* in *IEEE MILCOM 2010* (), 989–994 (cit. on p. 50).

124. Ikeda, M., Hiyama, M., Barolli, L., Xhafa, F. & Durresi, A. *Mobility Effects on the Performance of Mobile Ad hoc Networks* in *IEEE CISIS 2010* (), 230–237 (cit. on p. 50).

125. Kulla, E., Ikeda, M., Barolli, L., Miho, R. & Koliçi, V. *Effects of Source and Destination Movement on MANET Performance Considering OLSR and AODV Protocols* in *IEEE NBiS 2010* (), 510–515 (cit. on p. 50).

126. *Serval project* `https://www.servalproject.org` (cit. on p. 50).

127. *Village telco project* `https://villagetelco.org` (cit. on p. 50).

128. Baig, R. *Evaluation of Dynamic Routing Protocols on Realistic Wireless Topologies* MA thesis (Universitat Autònoma de Barcelona, 2012) (cit. on p. 50).

129. Clausen, T. *Comparative Study of Routing Protocols for mobile Ad-hoc NETworks* tech. rep. (INRIA, 2004) (cit. on p. 51).

130. Adjih, C., Baccelli, E., Clausen, T. H., Jacquet, P. & Rodolakis, G. Fish Eye OLSR Scaling Properties. Anglais. *Journal of Communications and Networks* **6,** 352–361 (Dec. 2004) (cit. on pp. 51, 54, 58).

131. Nguyen, D. & Minet, P. *Scalability of the OLSR Protocol with the Fish Eye Extension* in *Proceedings of the Sixth International Conference on Networking* (IEEE Computer Society, Washington, DC, USA, 2007), 88–94 (cit. on p. 51).

132. Johnson, D. & Hancke, G. Comparison of two routing metrics in {OLSR} on a grid based mesh network. *Ad Hoc Networks* **7,** 374–387 (2009) (cit. on pp. 51, 54).

133. Kuppusamy, P., Thirunavukkarasu, K. & Kalaavathi, B. *A study and comparison of OLSR, AODV and TORA routing protocols in ad hoc networks* in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on* **5** (Apr. 2011), 143–147 (cit. on p. 51).

134. Tamilarasan-Santhamurthy. A Quantitative Study and Comparison of AODV, OLSR and. TORA Routing Protocols in MANET. *International Journal of Computer Science Issues* **9,** 364–369 (Jan. 2012) (cit. on p. 51).

135. *Batman Advanced L2 routing protocol* `http://www.open-mesh.org` (cit. on p. 51).

136. Bouckaert, S., Vandenberghe, W., Jooris, B., Moerman, I. & Demeester, P. The w-iLab. t testbed (cit. on p. 60).

137. Seither, D., Konig, A. & Hollick, M. *Routing performance of Wireless Mesh Networks: A practical evaluation of BATMAN advanced* in *Local Computer Networks (LCN), 2011 IEEE 36th Conference on* (Oct. 2011), 897–904 (cit. on p. 71).

138. Chakeres, I. D. *AODV routing protocol implementation design* in *In ICDCSW '04: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04* (IEEE Computer Society, 2004), 698–703 (cit. on p. 71).

139. Perkins, C., Belding-Royer, E. & Das, S. *Ad hoc On-Demand Distance Vector (AODV) Routing* RFC 3561 (Experimental). Internet Engineering Task Force, July 2003 (cit. on p. 71).

140. *batman advanced - a layer 2 implementation of the B.A.T.M.A.N. routing protocol* `http://www.open-mesh.org/projects/batman-adv/wiki` (cit. on p. 71).

141. Akyildiz, I. F., Wang, X. & Wang, W. Wireless Mesh Networks: A Survey. *Comput. Netw. ISDN Syst.* **47,** 445–487 (Mar. 2005) (cit. on pp. 71, 73).

142. Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L. & Pillay-Esnault, P. *Multi-Topology (MT) Routing in OSPF* RFC 4915 (Proposed Standard). Internet Engineering Task Force, June 2007 (cit. on p. 72).

143. Hauge, M., Brose, M., Sander, J. & Andersson, J. *Multi-Topology routing for QoS support in the CoNSIS convoy MANET* in *Communications and Information Systems Conference (MCC), 2012 Military* (Oct. 2012), 1–8 (cit. on p. 72).

144. Gupte, S. & Singhal, M. Secure routing in mobile wireless ad hoc networks. *Ad Hoc Networks* **1,** 151–174 (2003) (cit. on p. 73).

145. Sanzgiri, K., LaFlamme, D., Dahill, B., Levine, B., Shields, C. & Belding-Royer, E. Authenticated routing for ad hoc networks. *Selected Areas in Communications, IEEE Journal on* **23,** 598–610 (Mar. 2005) (cit. on p. 73).

146. Herberg, U., Clausen, T. & Milan, J. *Digital Signatures for Admittance Control in the Optimized Link State Routing Protocol Version 2* in *Internet Technology and Applications, 2010 Int. Conf. on* (Aug. 2010), 1–4 (cit. on p. 73).

147. Denis Ovsienko. *Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication* RFC 7298 (Experimental). Internet Engineering Task Force, 2014 (cit. on p. 73).

148. Hu, Y.-C., Johnson, D. B. & Perrig, A. *SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks* in *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications* (IEEE Computer Society, Washington, DC, USA, 2002), 3– (cit. on pp. 73, 83).

149. Guerrero-Zapata, M. *Secure Ad hoc On-Demand Distance Vector (SAODV) Routing* draft-guerrero-manet-saodv-06.txt. Sept. 2006 (cit. on p. 73).

150. Zapata, M. G. Key Management and Delayed Verification for Ad Hoc Networks. *J. High Speed Netw.* **15,** 93–109 (Jan. 2006) (cit. on p. 73).

151. Adnane, A., Bidan, C. & de Sousa Júnior, R. T. Trust-based security for the {OLSR} routing protocol. *Computer Communications* **36,** 1159–1171 (2013) (cit. on pp. 73, 112).

152. Mogre, P. S., Graffi, K., Hollick, M. & Steinmetz, R. A Security Framework for Wireless Mesh Networks. *Wirel. Commun. Mob. Comput.* **11,** 371–391 (Mar. 2011) (cit. on pp. 73, 112).

153. Bina, M. & Giaglis, G. M. *Unwired Collective Action: Motivations of Wireless Community Participants* in *2006 International Conference on Mobile Business* (June 2006), 31–31 (cit. on p. 73).

154. Barbir, A., Murphy, S. & Yang, Y. *Generic Threats to Routing Protocols* RFC 4593 (Informational). Internet Engineering Task Force, Oct. 2006 (cit. on p. 73).

155. Hollick, M., Nita-Rotaru, C., Papadimitratos, P., Perrig, A. & Schmid, S. Toward a Taxonomy and Attacker Model for Secure Routing Protocols. *SIGCOMM Comput. Commun. Rev.* **47,** 43–48 (Jan. 2017) (cit. on pp. 73, 74).

156. Papadimitratos, P. & Haas, Z. J. Securing the Internet Routing Infrastructure. *Comm. Mag.* **40,** 60–68 (Oct. 2002) (cit. on p. 73).

157. Papadimitratos, P. & Haas, Z. J. in *The Handbook of Ad Hoc Wireless Networks* (eds Ilyas, M. & Dorf, R. C.) 551–567 (CRC Press, Inc., Boca Raton, FL, USA, 2003) (cit. on p. 73).

158. Hong, F., Hong, L. & Fu, C. *Secure OLSR* in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications - Volume 1* (IEEE Computer Society, Washington, DC, USA, 2005), 713–718 (cit. on p. 73).

159. Ogier, R. & Spagnolo, P. *Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding* RFC 5614 (Experimental). Updated by RFC 7038. Internet Engineering Task Force, Aug. 2009 (cit. on p. 73).

160. Smith, B. R., Murthy, S. & Garcia-Luna-Aceves, J. J. *Securing Distance-Vector Routing Protocols* in *Proceedings of the 1997 Symposium on Network and Distributed System Security* (IEEE Computer Society, Washington, DC, USA, 1997), 85– (cit. on p. 73).

161. Buchegger, S. & Le Boudec, J.-Y. *Performance Analysis of the CONFIDANT Protocol* in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing* (ACM, Lausanne, Switzerland, 2002), 226–236 (cit. on p. 73).

162. Papadimitratos, P. & Haas, Z. J. *Secure link state routing for mobile ad hoc networks* in *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings.* (Jan. 2003), 379–383 (cit. on p. 73).

163. Papadimitratos, P., Haas, Z. & Samar, P. The Secure Routing Protocol (SPR) for Ad Hoc Networks. *drail-papadimitratos-secure-routing-protocol-0O. txt,* 12–1 (2002) (cit. on pp. 73, 93).

164. Papadimitratos, P., Haas, Z. J. & Hubaux, J. P. *How to Specify and How to Prove Correctness of Secure Routing Protocols for MANET* in *2006 3rd International Conference on Broadband Communications, Networks and Systems* (Oct. 2006), 1–10 (cit. on p. 73).

165. Wachs, M. *A Secure and Resilient Communication Infrastructure for Decentralized Networking Applications* PhD (Technische Universität München, München, Feb. 2015), 250 (cit. on p. 73).

166. Hadjichristofi, G. C., DaSilva, L. A., Midkiff, S. F., Lee, U. & Sousa, W. D. Routing, Security, Resource Management, and Monitoring in Ad Hoc Networks: Implementation and Integration. *Comput. Netw.* **55,** 282–299 (Jan. 2011) (cit. on p. 73).

167. Rusu, F. & Dobra, A. *Statistical Analysis of Sketch Estimators* in *ACM SIGMOD International Conference on Management of Data* (2007), 187–198 (cit. on p. 74).

168. Goldberg, S., Xiao, D., Tromer, E., Barak, B. & Rexford, J. Path-quality monitoring in the presence of adversaries. *ACM SIGMETRICS Performance Evaluation Review* (June 2008) (cit. on p. 74).

169. Zhang, X., Lan, C. & Perrig, A. *Secure and scalable fault localization under dynamic traffic patterns* in *IEEE Symposium on Security and Privacy* (2012), 317–331 (cit. on p. 74).

170. Lopez, E. & Navarro, L. Tight bounds for Sketches in Traffic Validation. *14th IEEE International Conference on Networking, Sensing and Control* (2017) (cit. on pp. 74, 112).

171. Bradley, K., Cheung, S., Puketza, N., Mukherjee, B. & Olsson, R. Detecting disruptive routers: a distributed network monitoring approach. *IEEE Network* (1998) (cit. on p. 74).

172. Mizrak, A. T., Savage, S. & Marzullo, K. Detecting Malicious Packet Losses. *IEEE Transactions on Parallel and Distributed Systems* (Feb. 2009) (cit. on p. 74).

173. Mizrak, A. T., Cheng, Y.-C. C., Marzullo, K. & Savage, S. Detecting and isolating malicious routers. *IEEE Transactions on Dependable and Secure Computing* **3,** 230–244 (July 2006) (cit. on p. 74).

174. López, E. & Navarro, L. *KDet: Coordinated Detection of Forwarding Faults in Wireless Community Networks* in *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 01* (IEEE Computer Society, Washington, DC, USA, 2015), 734–741 (cit. on pp. 74, 112).

175. Robitzsch, S., Murphy, S. & Szczechowiak, P. *Architecture of a Self Configuration Framework for 802.11-based multi-radio Mesh Networks* in *2013 IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)* (June 2013), 1–8 (cit. on p. 74).

176. Ko, B.-j. & Misra, V. *Distributed channel assignment in multi-radio 802.11 mesh networks* in *In WCNC* (2007), 3978–3983 (cit. on p. 74).

177. Guardalben, L., Villalba, L. J. G., Buiati, F., Sobral, J. B. M. & Camponogara, E. Self-Configuration and Self-Optimization Process in Heterogeneous Wireless Networks. *Sensors* **11,** 425–454 (2010) (cit. on p. 74).

178. Navarro, L., Vinas, R. B., Barz, C., Bonicioli, J., Braem, B., Freitag, F. & Vilata-i-Balaguer, I. Advances in wireless community networks with the community-lab testbed. *IEEE Communications Magazine* **54,** 20–27 (July 2016) (cit. on p. 74).

179. Blywis, B., Guenes, M., Juraschek, F. & Schiller, J. Trends, Advances, and Challenges in Testbed-based Wireless Mesh Network Research. *Mobile Networks and Applications* (Feb. 2010) (cit. on p. 74).

180. Wu, D., Gupta, D. & Mohapatra, P. QuRiNet: A Wide-Area Wireless Mesh Testbed for Research and Experimental Evaluations. *Ad Hoc Networks* (Feb. 2011) (cit. on p. 74).

181. Raychaudhuri, D., Seskar, I., Ott, M., Ganu, S., Ramachandran, K., Kremo, H., Siracusa, R., Liu, H. & Singh, M. *Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols* in *WCNC 2005, IEEE Wireless Communications and Networking Conference* (eds Chua, K. C., Roy, S., Leung, V., Jamalipour, A., Lim, T. J. & Varma, V.) **3** (IEEE, New Orleans, LA, USA, Mar. 2005), 1664–1669 (cit. on p. 74).

182. Tan, K., Wu, D., Jack Chan, A. & Mohapatra, P. *Comparing simulation tools and experimental testbeds for wireless mesh networks* in *2010 IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (IEEE, Montreal, QC, Canada, June 2010), 1–9 (cit. on p. 74).

183. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C. & Joglekar, A. *An Integrated Experimental Environment for Distributed Systems and Networks* in *OSDI02* (Boston, MA, Dec. 2002), 255–270 (cit. on p. 74).

184. Castignani, G., Loiseau, L. & Montavont, N. *An evaluation of IEEE 802.11 community networks deployments* in *The International Conference on Information Networking 2011 (ICOIN2011)* (Jan. 2011), 498–503 (cit. on p. 75).

185. National Institute of Standards and Technology. *FIPS PUB 180-4: Secure Hash Standard (SHS)* (Federal Information Processing Standards Publication, Aug. 2015) (cit. on p. 77).

186. Hu, Y.-C., Perrig, A. & Johnson, D. B. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications* **24,** 370–380 (Feb. 2006) (cit. on p. 79).

187. Hu, Y.-C., Perrig, A. & Johnson, D. *Packet leashes: a defense against wormhole attacks in wireless networks* in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies* **3** (Mar. 2003), 1976–1986 (cit. on pp. 79, 113).

188. Zhou, J., Cao, J., Zhang, J., Zhang, C. & Yu, Y. *Analysis and Countermeasure for Wormhole Attacks in Wireless Mesh Networks on a Real Testbed* in *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on* (Mar. 2012), 59–66 (cit. on pp. 79, 113).

189. Lopez, E. & Navarro, L. Coordinated Detection of Forwarding Faults in Wireless Community Networks. *Journal of Network and Computer Applications,* 1–25 (2016) (cit. on p. 79).

190. Perkins, C. E. & Bhagwat, P. *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers* in *Proceedings of the Conference on Communications Architectures, Protocols and Applications* (ACM, London, United Kingdom, 1994), 234–244 (cit. on pp. 80, 83).

191. Garzia, F. *Handbook of Communications Security* (WIT Press, 2013) (cit. on p. 82).

192. Hinden, R. & Haberman, B. *Unique Local IPv6 Unicast Addresses* RFC 4193 (Proposed Standard). Internet Engineering Task Force, Oct. 2005 (cit. on p. 83).

193. Aura, T. *Cryptographically Generated Addresses (CGA)* RFC 3972 (Proposed Standard). Updated by RFCs 4581, 4982. Internet Engineering Task Force, Mar. 2005 (cit. on p. 83).

194. Wang, W., Lu, Y. & Bhargava, B. *On security study of two distance vector routing protocols for mobile ad hoc networks* in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).* (Mar. 2003), 179–186 (cit. on p. 84).

195. John, J. P., Katz-Bassett, E., Krishnamurthy, A., Anderson, T. & Venkataramani, A. *Consensus routing: The Internet as a distributed system* in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (2008), 351–364 (cit. on p. 93).

196. Lamport, L. Proving the correctness of multiprocess programs. *IEEE transactions on software engineering* **1,** 125–143 (1977) (cit. on p. 93).

197. Datta, A., Franklin, J., Garg, D., Jia, L. & Kaynar, D. On Adversary Models and Compositional Security. *IEEE Security Privacy* **9,** 26–32 (May 2011) (cit. on p. 93).

198. *MbedTLS – SSL Library* `https://tls.mbed.org/ssl-library` (cit. on p. 99).

199. *POLARSSL – Straightforward, Secure Communication* `http://polarssl.org` (cit. on p. 99).

200. *BMX/Semtor related public git repositories* `https://github.com/axn/bmx6` (cit. on p. 99).