Next Generation Overlay Networks: Security, Trust, and Deployment Challenges

Jordi Paillissé Vilanova

Advisors:

Prof. Albert Cabellos Aparicio

UPC

Dr. Fabio Maino

Cisco

A thesis presented for the degree of
Doctor of Philosophy in Computer Architecture

Department of Computer Architecture
Universitat Politècnica de Catalunya - BarcelonaTech
Barcelona, Spain

May 2021

Acknowledgments

A lot of people have helped with this thesis. I hope to mention all of them, but I apologize in advance to those I unintentionally forget. First of all, my deep gratitude to my advisor Prof. Albert Cabellos. Albert has taught me a lot, not only about research. He has shown me how to focus on the essential, how to deal with publication rejects, and he has helped me out of the maze when I got stuck in an endless loop. He has also been available to discuss any new ideas, even those that did not make sense. Also, thanks to my co-advisor, Dr. Fabio Maino, both for his invaluable comments and vision, and for inviting me to his team at Cisco. Thanks for giving me a unique opportunity to understand how the networking industry works, right from the inside. Thanks also to the CTO team at Cisco Intent-Based Networking group for welcoming me, and especially to Lori Jakab and Elis Lulja for their help during my internship. A huge thanks to Marc Portolés and Alberto Rodríguez for their help with the enterprise networks part of this thesis, for their discussions about networking, and for making me feel at home during my internship.

Thanks to all the Bachelor's and Master's Thesis students who worked with me: Alejandro Barcia, Guillem Bonet, Emanuele Cesari, Miquel Ferriol, Eric Garcia, Lluis García, Ismael Julià, Krisztian Koteles, Hamid Latif, Jan Manrique, Carles Juan Martínez, Jose Ortiz, Carlos Piris, Ernest Puigdemont, Luis Eduardo Sosa, and Jordi Subirà. Your hard work and commitment has been a key part of this thesis.

Thanks also to all the people on the 008 office of the Computer Architecture Department: Sergi Abadal, Paul Almasan, Arnau Badia, Ismael Castell, Jérémy Dumont, Albert López, Oriol Marí, Albert Mestres, Kurdman Rasol, Giorgio Stampa, Maria Stefan, Jose Suárez-Varela, Hamidreza Taghvaee, and Faezeh Zarrinkhat, who make the work much more fun and are always willing to help. A special thanks to Albert López, for helping me whenever I needed, from administrative stuff to creative LISP questions. Thanks to Paul Almasan for his motivation during his Bachelor's Thesis, and our further exploration of packet processing algorithms later. And thanks to Jose Suárez-Varela for walking the PhD path in parallel with me, and for putting up with my political ramblings during our coffee breaks. Thanks also to Prof. Josep Solé for his advice during my PhD.

This thesis has been partially funded by Cisco, and by the Spanish MINECO ALLIANCE project, reference TEC2017-90034-C2-1-R. Part of this thesis was performed during an internship at Cisco in San Jose, California. Thanks also to Donald Knuth, Leslie Lamport, and the TEX community for creating the wonderful typesetting system that is LATEX.

Finally, an enormous thanks to my family and friends, who have been with me all the time, no matter what, sometimes without understanding why the Internet is so complex, sometimes when I was down, but always supporting me. Thank you!

Abstract

Overlay networks are a technique to build a new network on top of an existing one. They are a key tool to add functionality to existing networks, and are used in different layers of the Internet stack for a wide variety of purposes, like confidentiality, Quality of Service, virtual networking, etc. Specifically, network overlays in the IP networking layer are widely used in some of these use cases. However, these kind of overlay networks do not have as many functionalities as overlays in other layers. For example, thanks to the Zero Trust Networking paradigm it is possible to build secure overlay networks at L7 using HTTPS.

Taking this into account, this thesis strives to add new features and improve on others of IP overlay networks, in order to support emerging challenges. This thesis focuses on three axes: security, trust, and deployment in enterprise scenarios. First, regarding security, we explore how to simplify the setup of secure tunnels over the Internet, without relying on external Public Key Infrastructure or proprietary solutions. To this purpose, we leverage WireGuard, a state of the art VPN protocol, and add a control plane on top of it to distribute encryption keys. In addition, we present the implementation of a prototype and a performance evaluation.

Second, with respect to trust, we investigate how emerging blockchain technology can be used in distributed mapping systems. Mapping systems are a database used in some overlay network deployments to assist in the creation of tunnels, by storing overlay to underlay pairs of addresses. Mapping systems are not commonly used in scenarios with multiple administrative domains, due to configuration complexity and centralized control. We explore how some of the properties of blockchains, such as distributed control, or auditability, can help in building these type of mapping systems. We take into account both the policy aspects, that is, the advantages of a distributed trust scheme, and the technical ones, like simplified management. In addition, we present two deployment scenarios: one to increase the security of BGP-based inter-domain routing, and a set of cooperating companies that want to establish communications among themselves.

Finally, we focus on the deployment of enterprise networks leveraging overlay networks. First, we discuss the challenges present in current enterprise networks, such as segmentation, mobility, or simplified operations. Then, we present a design based on overlay networks and SDN principles to address them, along with an evaluation of two real-life deployments. We conclude with a design tailored for future enterprise networks, based also on overlay networks and a layered approach. This solution aims to provide mobility, multi-homing, confidentiality, user and application identity, and access control policies for enterprise endpoints connected from any network, either in the campus or outside.

Keywords: overlay networks, IP networks, mapping system, blockchain, enterprise networks, inter-domain routing security, secure tunnels, software defined networks, network mobility.

Resum

Les xarxes superposades o overlays són una tècnica per a construir una xarxa sobre una d'existent. Són una eina fonamental per a afegir funcionalitat a xarxes ja existents, i es fan servir en diferents capes de la pila de protocols d'Internet per a diferents objectius, com ara confidencialitat, Qualitat de Servei, virtualització de xarxes, etc. Específicament, les xarxes superposades a la capa de xarxa IP es fan servir àmpliament per alguns d'aquest casos d'ús. No obstant, aquest tipus de xarxes superposades no tenen tanta funcionalitat com overlays en altres capes. Per exemple, gràcies al paradigma Zero Trust és possible construir xarxes superposades segures a la capa 7 amb HTTPS.

Tenint això en compte, aquesta tesi vol afegir noves funcionalitats i millorar-ne d'altres en xarxes IP superposades, amb l'objectiu d'afrontar nous reptes. Aquesta tesi es centra en tres eixos: seguretat, confiança i desplegament en escenaris de xarxes empresarials. En primer lloc, pel que fa a seguretat, explorem com simplificar la configuració de túnels segurs a través d'Internet, sense dependre d'una infraestructura de clau pública externa o de solucions propietàries. Per a aquest objectiu, utilitzem WireGuard, un protocol VPN d'última generació i li afegim un pla de control per a distribuir les claus d'encriptació. A més, presentem la implementació d'un prototip i una avaluació del seu rendiment.

En segon lloc, respecte la confiança, investiguem com les tecnologies emergents basades en cadenes de blocs (*blockchain*) es poden fer servir en sistemes de mapatge distribuïts. Els sistemes de mapatge són una base de dades que es fa servir en alguns desplegaments de xarxes superposades per a ajudar en la creació dels túnels; normalment guarden parelles d'adreces que tradueixen l'adreça de la xarxa superposada a la de la xarxa de sota. Els sistemes de mapatge no es solen utilitzar en escenaris amb múltiples dominis administratius, degut a la complexitat de la configuració i a la centralització del seu control. En aquesta part explorem com algunes de les propietats de les cadenes de blocs, com el control distribuït o l'auditabilitat poden ajudar a construir aquests tipus de sistemes de mapatge. Tenim en compte tant els aspectes polítics, és a dir, els avantatges d'un esquema de confiança distribuït, com tècnics, per exemple, una gestió més simple. A més, presentem dos escenaris de desplegament: un per a incrementar la seguretat de l'enrutament basat en BGP entre diferents dominis, i un altre d'un conjunt d'empreses que cooperen per a establir comunicacions entre elles.

Finalment, ens centrem en el desplegament de xarxes per a empreses fent servir xarxes superposades. En primer lloc, detallem els reptes que hi ha actualment a les xarxes per a empreses, per exemple, segmentació, mobilitat o simplificació de les operacions. A continuació, presentem un disseny basat en xarxes superposades i principis SDN que aborda els reptes que hem mencionat, a més d'una avaluació de dos desplegaments reals. Acabem amb una solució dissenyada per a les xarxes empresarials del futur, també basat en xarxes superposades i una aproximació per capes. Aquesta solució està dirigida a oferir mobilitat, connexió simultània (*multi-homing*), confidencialitat, identitat de l'usuari i l'aplicació, i polítiques de control d'accés per a dispositius connectats des de qualsevol xarxa, sigui des de l'oficina o des de fora.

Paraules clau: xarxes superposades, xarxes IP, sistema de mapatge, blockchain, xarxes empresarials, seguretat en l'enrutament entre dominis, túnels segurs, xarxes definides per software, mobilitat de xarxa.

Resumen

Las redes superpuestas u overlays son una técnica para construir una nueva red encima de una ya existente. Son una herramienta clave para añadir funcionalidad a redes existentes, y se usan en diferentes capas de la pila de protocolos de Internet para una gran variedad de propósitos, como confidencialidad, Calidad de Servicio, redes virtuales, etc. Específicamente, las redes superpuestas en la capa de red IP son ampliamente usadas para algunos de estos casos de uso. No obstante, este tipo de redes no disponen de tantas funcionalidades como redes superpuestas en otras capas. Por ejemplo, gracias al paradigma Zero Trust es posible construir redes superpuestas seguras en la capa 7 usando HTTPS.

Teniendo esto en cuenta, esta tesis tiene como objetivo añadir nuevas funcionalidades y mejorar otras de las redes IP superpuestas, con el propósito de afrontar los nuevos retos que van apareciendo. Esta tesis se centra en tres ejes: seguridad, confianza y despliegue en escenarios empresariales. En primer lugar, y respecto a la seguridad, exploramos cómo simplificar la configuración de túneles seguros a través de Internet, sin usar una infraestructura de clave pública externa o soluciones propietarias. Para este objetivo, utilizamos WireGuard, un protocolo VPN de última generación, y le añadimos un plano de control para distribuir las claves de encriptado. Además, presentamos la implementación de un prototipo y una evaluación de rendimiento.

En segundo lugar, en relación a la confianza, investigamos como las tecnologías emergentes basadas en cadena de bloques (*blockchain*) se pueden usar en sistemas de mapeado distribuidos. Los sistemas de mapeado son una base de datos que se utiliza en algunas redes superpuestas para ayudar en la creación de los túneles. Normalmente, estos sistemas guardan parejas de direcciones que traducen direcciones de la red superpuesta a la de la red subyacente. Los sistemas de mapeado no son muy utilizados en escenarios con múltiples dominios administrativos, debido a la complejidad de la configuración y a la centralización del control. En esta parte exploramos como algunas de las propiedades de las cadenas de bloques, como el control distribuido, o la auditabilidad, pueden ser de ayuda en la construcción de este tipo de sistemas de mapeado. Tomamos en consideración tanto los aspectos políticos, esto es, las ventajas de un esquema de confianza distribuido, como los técnicos, por ejemplo, una gestión más simple. También presentamos dos escenarios de despliegue: uno para aumentar la seguridad del enrutamiento basado en BGP entre diferentes dominios, y otro de un conjunto de empresas que cooperan para establecer conexiones entre ellas.

Finalmente, nos centramos en el despliegue de redes empresariales usando redes superpuestas. En primer lugar, detallamos los retos que existen hoy en día en las redes empresariales, por ejemplo, segmentación, movilidad, o simplificación de las operaciones. A continuación, presentamos un diseño basado en redes superpuestas y principios SDN que aborda los retos que hemos mencionado, junto con la evaluación de dos despliegues reales. Concluimos con una solución diseñada para las redes empresariales del futuro, basada también en redes superpuestas y una aproximación por capas. Esta solución tiene como propósito ofrecer movilidad, conexión simultánea (*multi-homing*), confidencialidad, identificación de usuario y aplicación, y políticas de control de acceso para dispositivos conectados desde cualquier red, sea en la oficina o fuera.

Palabras clave: redes superpuestas, redes IP, sistema de mapeado, blockchain, redes empresariales, seguridad del enrutamiento entre dominios, túneles seguros, redes definidas por software, movilidad de red.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

The last 10 years have seen a paradigm shift in computer networking, both in the industry and in research due to the emergence of Software Defined Networking (SDN [1]). The key principles of SDN, like control and data plane separation, centralized control or programmability [2] have been successfully applied to all kinds of production networks: datacenter, ISPs, enterprises, etc. On the research front, the query "Software Defined Networking" returns around 2 million results in Google Scholar.

More recently, the principle of programmability has been significantly expanded by the introduction of programmable data planes, i.e. programs in domain-specific high-level languages that are compiled and executed in novel hardware platforms capable of modifying their forwarding behavior according to these programs [3]. This trend is creating new research areas like network-specific programming languages [4], or their formal verification [5].

One of the main elements in SDN-enabled networks is the southbound protocol, that connects the data plane and the control plane, and transfers forwarding tables, route updates, statistics, etc. Interestingly, the vast majority of the southbound protocols leverage a proactive approach, that is, sending the forwarding information in advance from the SDN controller to the switch. Indeed, OpenFlow [6], as the *de facto* standard for southbound protocols, follows this model, although it includes the packet-in message to ask the controller new rules for packets that don't match the ones present in the switch. Its natural successor, P4Runtime [7], takes a similar approach, because it generates an API that can be accessed by the SDN controller to program the forwarding tables.

On the other hand, reactive protocols, i.e. requesting forwarding entries on demand, driven by traffic patterns, are not common in SDN scenarios. Nevertheless, they present interesting advantages like a reduction in data plane state, or better adaptability to traffic patterns. In this thesis we aim to explore such benefits and improve several of their associated elements, more specifically overlay networks. Overlay networks are strongly related to SDN; they have been used in different SDN designs, for example, to create virtual networks with VXLAN [8], as a southbound SDN protocol [9], to simplify SDN deployment [10], or to offer QoS [11].

Hence, the combination of SDN and overlay networks is powerful and can support a wide range of use cases. Taking this into account, in this thesis we aim to improve on several elements of overlay networks in SDN scenarios, but with the perspective of reactive southbound protocols. Prior related work has explored how to use the Locator/ID Separation Protocol as a southbound SDN protocol [9]. In this thesis we advance this line of research, exploring this type of overlay-based SDN networks in three directions: data plane security, trust in distributed mapping systems, and deployment in enterprise networks.

1.2 Background: Overlay Networks

Overlay networks are a technique to layer computer networks on top of other networks. Since they don't require upgrading all the network infrastructure, i.e., they can be incrementally deployed,

they have been used to easily add new services or functionalities to exiting networks. Indeed, this concept has been applied to a wide variety of use cases, such as resilient networks [12], network mobility [13, 14], virtual networking [8], scalable extension of L2 domains [15, 16], peer-to-peer systems [17, 18], multicast [19], QoS [20], improved routing [21], etc.

Since this thesis revolves around overlay networks and possible improvements to several of their elements, here we present a reference architecture of an overlay network that will be used in this thesis. Overlay-based networks consist of several independent elements that work collaboratively (figure 1.1), namely:

Underlay: the bottom network layer, composed of routers that connect overlay endpoints. It is commonly a plain IP network.

Overlay: the upper network layer. Its routers add and remove the headers used in the underlay network (commonly know as en- or decapsulation routers, respectively).

Overlay endpoints: the collection of hosts that interact with the overlay, i.e. they are not aware of the existence of the underlay.

Mapping System: a database server that maintains mappings of overlay endpoints to underlay routers. We must note that the mapping system is optional (for example VXLAN-based networks [22]). It can also be a key-value store, like in the Locator/ID Separation Protocol [23]. Finally, it can be part of a fully-fledged SDN operating system [24], offering more complex services.

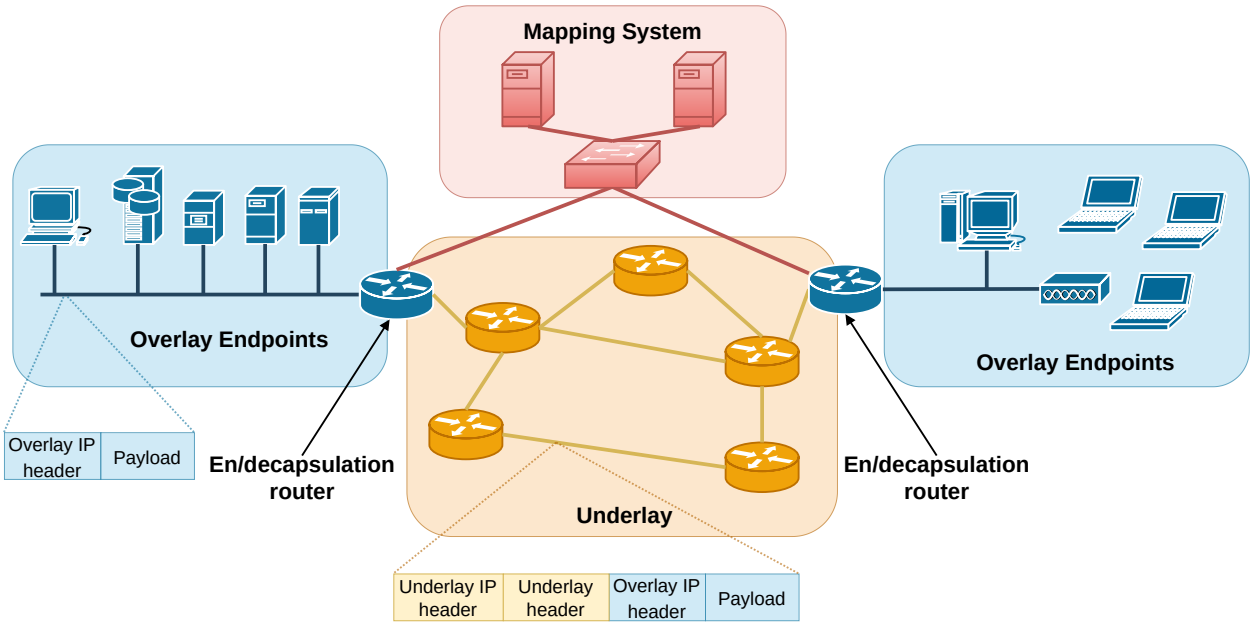


Figure 1.1: Reference architecture for overlay networks.

1.3 Motivation

In the light of the evolution in networking research (software-defined networks, programmable data planes, etc.) and trends on the global Internet (encryption of all communications, increase in security threats, etc), some of the aforementioned elements require additional features or redesigns to support emerging use cases and trends. Specifically, from bottom to top:

Underlay data plane security: it is common to encrypt all communications crossing the public Internet, but the configuration of such encrypted streams is either via: (i) TLS keys derived from a Public Key Infrastructure (PKI), or (ii) manual or automated configuration of IPsec tunnels. In both cases the configuration is complex: TLS needs its own Certification Authority, or trusting the global Internet PKI, while IPsec offers a wide range of configuration knobs that can be overwhelming for inexperienced users.

Mapping system: typical mapping systems consist of a single (possibly redundant) server limited to a single administrative domain. Mapping servers spanning different administrative domains are not commonly deployed because they require complex configuration, such as LISP-TREE [25], or lack secure authentication of the mappings, e.g. LISP-MSX [26].

Overlay networks in enterprises: in recent years, there has been few research regarding enterprise networks (with the notable exception of Zero Trust Networks [27]). Nevertheless, the requirements of enterprise networks have kept evolving and posing new challenges which are hard to meet by classical enterprise network architectures. We address this shortcoming by investigating how overlay networks can be leveraged to address these challenges.

Future overlay networks for enterprise use cases: finally, taking into account the current trends in networking, we ask which are the requirements of *future* enterprise networks and if overlays play a role in it.

1.4 Objectives

The following list details the objectives of this thesis. Each objective aims to enhance or add new features to a particular element of overlay networks (figure 1.2):

Underlay data plane security: design a simple control plane for the WireGuard VPN protocol [28], in order to make it easier to establish multipoint encrypted overlay tunnels without the burden of IPsec configuration.

Multi-domain mapping system: we intersect the aforementioned gap regarding multi-domain mapping systems with emerging blockchain technologies. The goal is designing a mapping system that can work in such environment, but preserving the control of the mappings to each participant. Our approach it two-fold: (i) requirements to securely transfer mappings

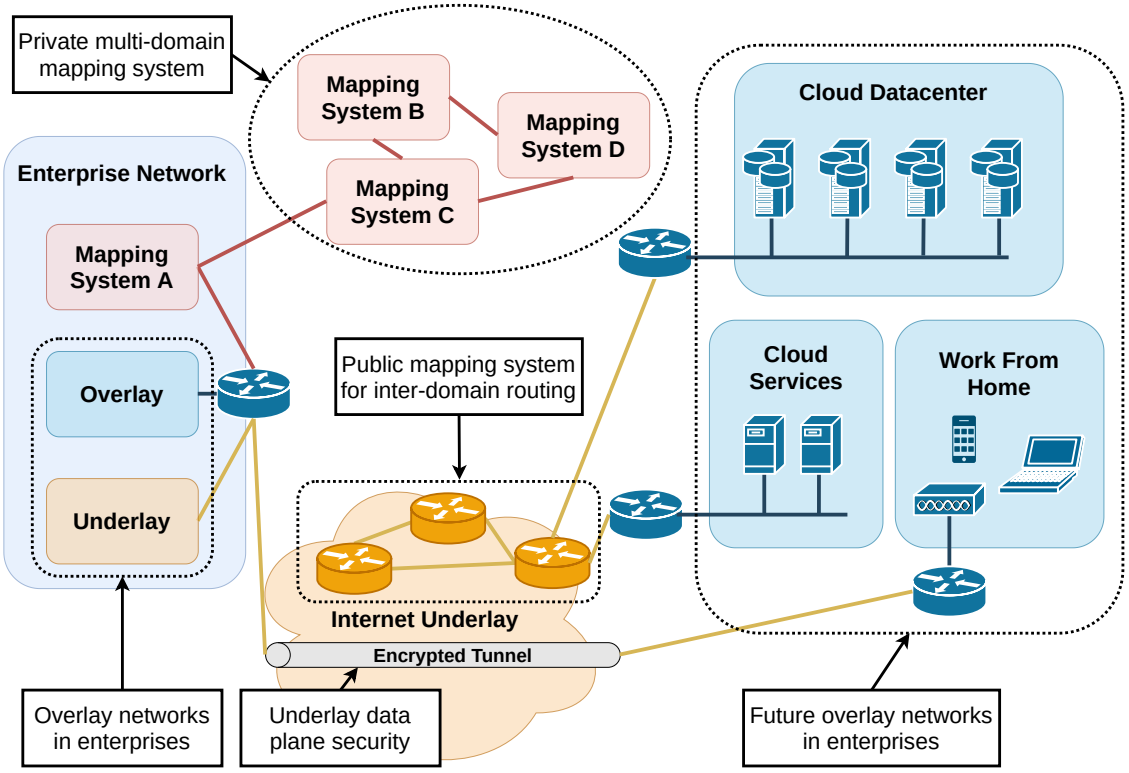


Figure 1.2: Thesis objectives with respect to the reference architecture.

among participants, and (ii) definition of trust architectures for such separated administrative domains.

Mapping system applications to inter-domain routing: the problem of mapping IP prefixes to AS numbers is equivalent to the mapping of overlay to underlay identifiers. Taking into account this similarity, we investigate how a blockchain-based mapping system could be used in the public Internet to disseminate IP prefix to AS mappings to enhance BGP routing security.

Overlay networks in enterprises: study the requirements of current enterprise networks, and evaluate if an architecture based on network overlays can address the challenges of these networks. Specifically, we investigate how overlay networks in an enterprise scenario: (i) increase scalability of the wireless data plane, (ii) reduce data plane state, and (iii) simplify how segmentation is implemented.

Future overlay networks for enterprise use cases: finally, we elaborate on the requirements of future enterprise networks and design a layered architecture based on network overlays and the current Internet stack to address such requirements.

1.5 Methodology

As described in section 1.7, this thesis is separated in two major parts. We used a different methodology for each part due to their different characteristics:

Part I: Blockchain Applications to Address Assignment

This part contains all the topics related to a blockchain-based mapping system. Since blockchain technologies are evolving continuously, and its application to production scenarios is not immediate, we took an iterative approach (figure 1.3). In parallel, we (i) investigated the state of the art of blockchain technology, and (ii) presented our ideas in the IETF and networking conferences to gather feedback. Then we used the feedback to improve our idea, and repeated the process several times until we believed it was sufficiently refined for publication.

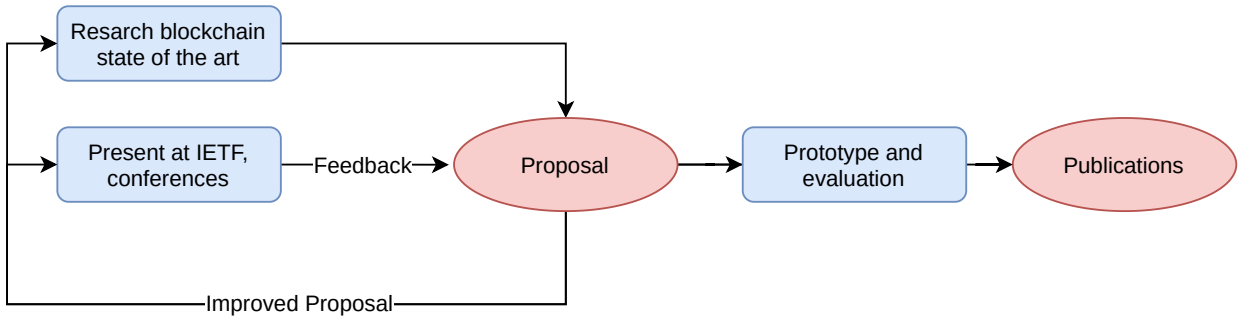


Figure 1.3: Methodology for the blockchain applications part.

Part II: Network Architectures for Enterprise Networks

This part encompasses the rest of the topics discussed previously: underlay data plane security, current, and future overlay networks in enterprises. As opposed to the other part, this part is directly applicable, so we leveraged a contact with a network vendor to understand the problem and possible solutions (figure 1.4). Hence, here the methodology involved discussions with the vendor about problems, possible solutions and relevant metrics to evaluate. Once these were clear, we designed and evaluated prototypes, and published the results.

1.6 Contributions

As we mentioned previously, this thesis is divided in two major parts, with the following contributions:

Part I: Blockchain Applications to Address Assignment

- Analysis of the applicability of blockchain technologies to Internet address assignment, both for the public Internet and in private networks.

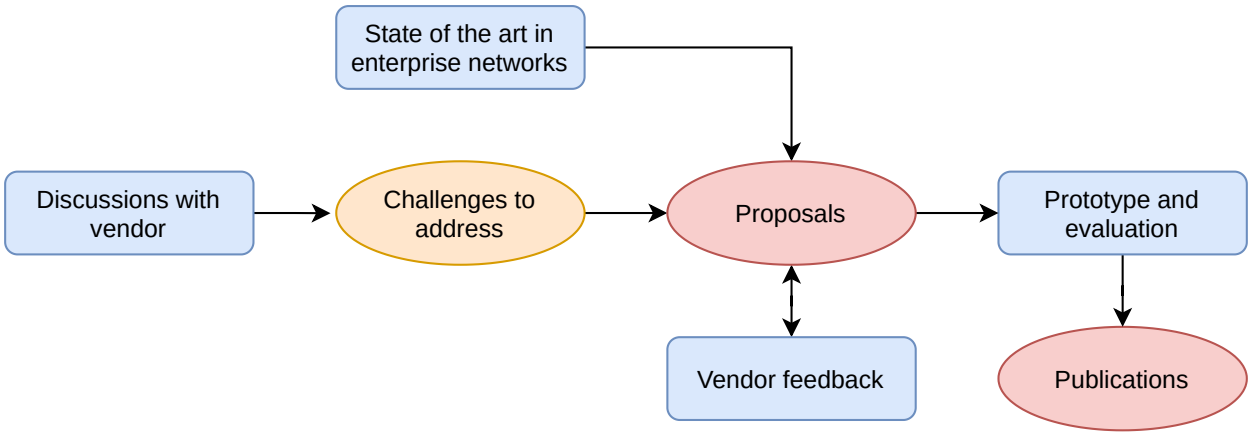


Figure 1.4: Methodology for the enterprise networks part.

- Discussion of the appropriate blockchain consensus algorithm for such use cases.
- Design and evaluation of a blockchain prototype for address assignment, again for both use cases.

Part II: Network Architectures for Enterprise Networks

- Design and evaluation of a simple control plane to safely exchange the encryption keys of the WireGuard VPN protocol, in order to build encrypted underlays.
- Analysis of the challenges and requirements of current enterprise networks.
- Design rationale and evaluation of a commercial solution for current enterprise networks, based on SDN principles and network overlays.
- Analysis of the challenges of future enterprise networks, and design of a layered architecture to support them.

1.7 Thesis Outline

This thesis is divided in two major parts. The first part, *Blockchain for Internet Applications*, contains all the work related to mapping systems for different administrative domains, leveraging blockchain technology. The second part, *Architectures for Enterprise Networks*, deals with securing the underlay segment of overlay networks, and applications of overlay networks in enterprise networks, both present and future. Finally, the rest of this chapter presents the state of the art related to the three major topics of this thesis: security of overlay networks, trust in distributed mapping systems, and deployment in enterprise networks.

1.7.1 Blockchain for Internet Applications

The first part is divided in three chapters. Chapter 2 provides a background on blockchain technologies and its core component, consensus algorithms. Then, chapter 3 discusses the application of a mapping system based on blockchain to inter-domain routing security. This chapter includes: (i) a detailed comparison with state of the art PKI-based systems for inter-domain routing security, (ii) a thorough discussion on which consensus algorithm is the most appropriate for this use case, and (iii) several considerations regarding the deployment of a blockchain in this scenario. It also presents the design and evaluation of a prototype that leverages a simplified Proof of Stake consensus algorithm. Finally, chapter 4 re-evaluates the previous proposition but in a scenario of several collaborating enterprises that want to establish communications among themselves, i.e. a private network. Again, this chapter discusses the benefits of leveraging blockchain, and the selection of the appropriate consensus algorithm. Finally, it includes the design and evaluation of a prototype of such blockchain.

1.7.2 Architectures for Enterprise Networks

The second part consists of three chapters. First, chapter 5 presents a simple control plane to secure underlay networks, that leverages the WireGuard VPN protocol in the data plane. It also includes the design and evaluation of a prototype. Then, chapter 6 discusses the challenges of current enterprise networks, and proposes an architecture based on SDN principles, overlay networks, and group-based policies to address them. Additionally, it evaluates such proposal regarding control plane latency or reduction in data plane state, and details several lessons learned. Finally, chapter 7 builds on the current state of the art to propose a layered architecture based on overlays to support the requirements of future enterprise networks.

1.8 State of the Art

1.8.1 Security

As we mentioned before, creating encrypted connections over the Internet - or other networks - is not straightforward. In a nutshell, there are three possible approaches: IPsec tunnels, TLS VPNs, or proprietary SDWAN (Software-Defined Wide Area Network) systems. Table 1.1 summarizes their pros and cons. We have mainly focused on operational cost and supported features, such as the configuration complexity, the kinds of traffic that they support, or their cost.

First, IPsec VPNs are convenient because they can encapsulate any kind of L3 traffic, and they use regular IP routing. On the other hand, they are complex to configure, especially the key distribution mechanism, or the cryptographic properties due to the wide range of cipher suites. Second, in TLS-based VPNs we can leverage the web PKI to distribute keys easily, but on the flip side we are limited to TCP flows, or we will need additional protocols such as DTLS (Datagram TLS). Note here that it is also possible to deploy a private PKI, but this comes at the expense of managing a Certification Authority and additional infrastructure. Finally, SDWANs, or dynamic

VPNs are commercial products that automate all the steps to create secure tunnels, and accept all kinds of L3 traffic, but are costly due to their commercial aspect.

Approach	Advantages	Disadvantages
IPsec	Encrypt any L3 traffic	Complex configuration
TLS VPNs	Simplified key distribution thanks to the web PKI	Only for TCP flows, or use other protocols for UDP
SDWAN or dynamic VPNs	Simple configuration, any L3 traffic	Proprietary and costly

Table 1.1: Possible approaches for encrypted connections over the Internet.

In addition, there are other features that are becoming more relevant for certain users, like mobility, that are not natively supported by these approaches. Therefore, it would be interesting to design another solution that:

1. Is easy to configure
2. Supports all kinds of L3 traffic
3. Is inexpensive
4. Can handle additional features like mobility or multi-homing

In this context, WireGuard [28] is a new VPN protocol that aligns well with some of these requirements: it is easy to configure (similar to SSH), works at L3, is open-source, and supports mobility. However, it still presents some limitations, such as multi-homing or automating the configuration of tunnels among several endpoints.

1.8.2 Trust

Currently, trust in the Internet is hierarchical and centralized, i.e. Certification Authorities (CA) issue and revoke certificates, creating a hierarchy. Participants in this system have to trust the CAs, which have total control over their downstream certificates (figure 1.5, left). On the other hand, the emergence of Bitcoin and blockchains has offered a way to manage such trust in a completely decentralized way: each participant has complete control of their private key. In other words, only the owner of such private key can trigger any action associated with it (figure 1.5, right).

This decentralized approach has spurred a large amount of research on blockchains and decentralized ledgers, as well as several successful production blockchains (Bitcoin, Ethereum, etc). In the networking area, the usage of blockchain as a tool for networking use cases is at an early stage. Although there is a remarkable amount of research papers, the most mature application are decentralized naming systems, with equivalent functionality to that of the DNS [29, 30]. In other words, the application of blockchain technology to networking scenarios is still a relatively new research topic, with interesting challenges and trade-offs [31].

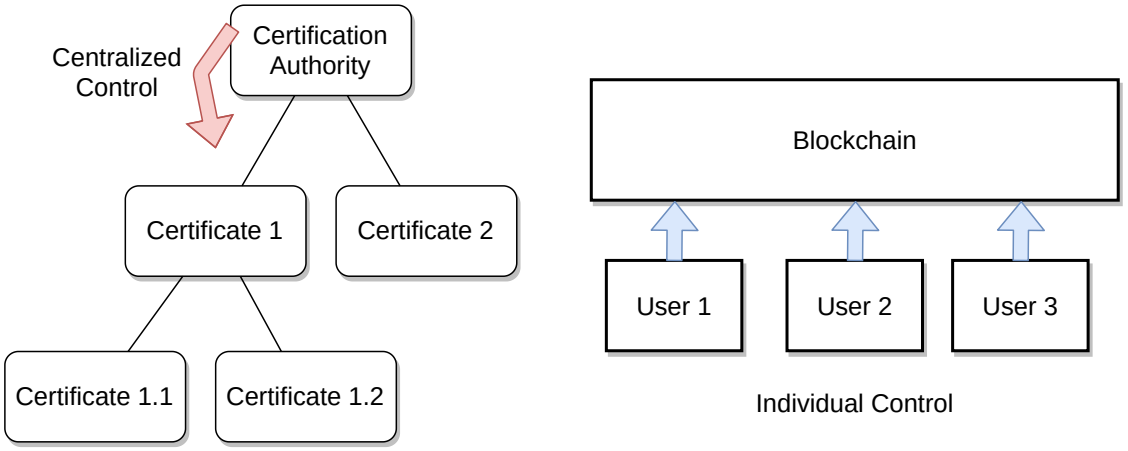


Figure 1.5: Centralized trust (left) and distributed trust (right).

More specifically, with respect to overlay networks, this decentralization can be leveraged in their control plane, i.e. the mapping system. Thus, we can let each endpoint individually manage their mapping, as if it was a cryptographic coin. This decentralized approach creates new trust architectures that have not been discussed yet. On the policy side, these new trust architectures can mitigate centralization concerns about some systems running on the Internet, like DNSSEC or the secure assignment of Internet number resources. On the technical side, blockchains offer several properties that contrast with classical PKIs, such as auditability or simplified management, that can be used to improve current mapping systems.

1.8.3 Deployment in Enterprise Networks

The last 10 years have seen a significant addition of functionalities to networks at L7, basically leveraging HTTP [32]. Two key examples are Zero Trust Networks and Service Meshes. Zero Trust Networking [27] is a novel approach to security in enterprise networks based on HTTP: all communications between clients and servers run on HTTP. Then, an HTTP proxy that sits between the clients and the servers, the *access proxy*, continuously verifies all connections to the servers with data from a credentials database. Data is always encrypted with the use of HTTPS, and the credentials database allows a wide range of access control policies, from a list of allowed applications to blocking access depending on the version of the device OS.

Service Meshes [33, 34] also rely on HTTPS to transport data, but in data centers that are running microservices. In this case, their main element is the *sidecar proxy*, that runs alongside the microservices. This sidecar proxy intercepts all communications between the microservices, monitors their status, and enforces policies. Moreover, their architecture presents some similarities with SDN networks. For example, Istio is a centralized controller for service meshes running the Envoy proxy [35], that can monitor traffic, enforce policies, route flows, etc. Both of these improvements are, in fact, overlays based on HTTP.

On the contrary, L3 network overlays have seen fewer improvements. Nevertheless, tackling

some problems at L3 instead of L7 offers three main advantages:

- Ability to protect the infrastructure (routers, etc) from attacks, eg. Distributed Denial of Service.
- Finer grain control of the infrastructure, which brings performance improvements. This is especially relevant for QoS policies.
- Support legacy equipment that does not use HTTP or work at L7.

We must also remark that L3 overlays present some drawbacks when compared with L7 overlays, especially regarding deployment: while HTTP is commonly implemented in software and we need to update fewer devices, L3 functionalities usually require hardware implementations in order to run at line rate, and more devices have to be updated.

Taking this into account, we can see that the state of the art of L3 overlay networks does not offer all the functionalities present in L7 technologies. At the same time, the latter cannot provide the benefits of L3 overlays. Therefore, it would be interesting to bring some of these L7 overlay features to L3 overlays. Specifically, in the context of enterprise networks there has been few research on improving current architectures, with the notable exception of Zero Trust Networks. We identified two main gaps in the state of the art:

- Applications of L3 overlays in enterprise networks in order to tackle their current challenges.
- How to add some of the L7 functionalities to L3 enterprise networks in order to:
  - Include the aforementioned benefits of L3 overlays in new deployments.
  - Tackle future requirements of such networks.
  - Make deployment as simple as possible.

Interestingly, recent publications tend towards similar ideas. For example, Full Stack SDN [36] proposes unifying network functions from L2 to L7 in order to increase performance and efficiency. Another relevant idea in this proposal is creating different types of sockets depending on the application using them, a concept that we reuse in chapter 7. More recently, Compositional Network Architecture [37] presents a new approach to describe network architecture, that we have found useful when designing a solution for future enterprise networks.

Part I

Blockchain Applications to Address Assignment

Chapter 2

Background

2.1 Blockchain

In a nutshell, a blockchain is a distributed, secure, tamper-resistant, and append-only database. Each blockchain participant stores a replica of the database. Participants exchange messages via a peer-to-peer (P2P) network in order to: (i) modify the database, and (ii) distribute copies of the database to new users. Usually, the blockchain database tracks the owner of some kind of asset or token (coins for example). It is often regarded as a virtual ledger that records pairs of (owner, asset) with specific rules that control state transitions, i.e. a replicated state machine. The typical workflow is as follows:

1. Participants broadcast *transactions*, messages signed by their private key that alter the state of the database.

2. All participants store transactions in a temporary storage. Invalid signatures or malformed transactions are rejected.

3. At some fixed intervals in time, one of the participants (usually known as *miner*) puts all the current transactions together and signs them, in a data structure known as *block*. Then, it broadcasts the block to the P2P network.

4. The rest of the participants receive the block, validate the signature, and run the consensus algorithm.

5. If the algorithm determines the block is correct, it is added right after the previous block, thus updating the database state. The new block contains the hash of the previous block, thus linking one block after another, and hence the term *blockchain*.

Two mechanisms provide security to the blockchain: the consensus algorithm and a chain of signatures. On one hand, the consensus algorithm ensures that all participants agree on the same blockchain state, thus providing data integrity and consistency. On the other hand, the chain of signatures protects the individual ownership of each asset in the chain: assets are bound to its owner by a public-private keypair. The public key identifies the owner, and the private key is used to alter the state of the token or coin. Since the private key is solely owned by the user, only the entity that controls this private key can alter the state of the asset. In order to transfer an asset, the sending party creates a *transaction*: it signs the public key of the new owner with its private key and broadcasts this signature to the network, hence the chain of signatures.

In a more broad sense, a blockchain can be described as a four-layer system (figure 2.1):

Peer-to-peer network: Transmits and receives new blocks and transactions.

Consensus algorithm: Enforces a particular and unique order of blocks in the chain, making sure that all participants reach the same conclusion on which blocks get added at the end of the blockchain, and their order.

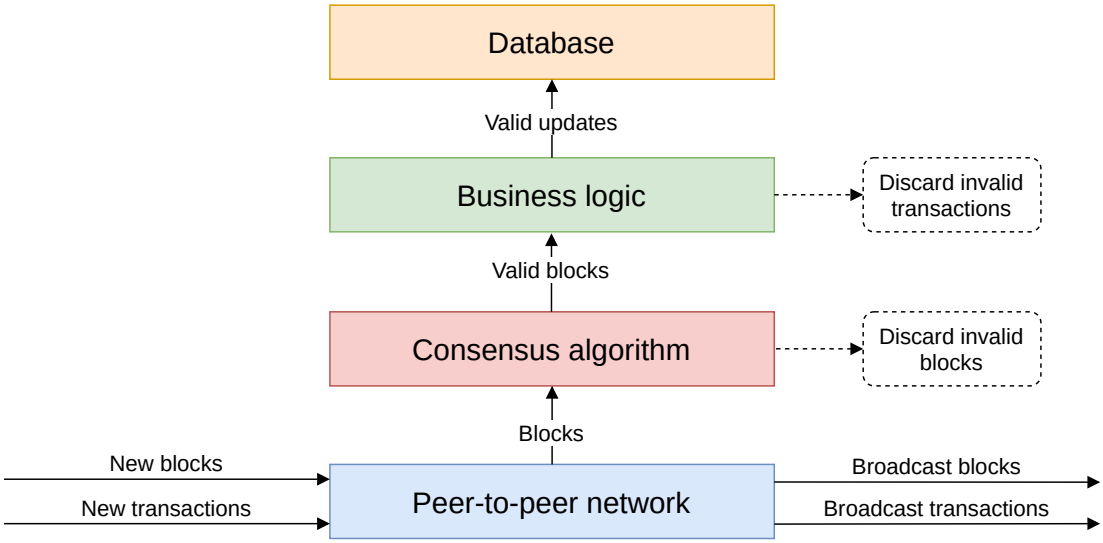


Figure 2.1: Simplified blockchain operation in four layers.

Business logic: Allows specific state transitions, thereby enforcing a set of rules. The blockchain rejects any transition that does not conform to this logic. For example, in Bitcoin it is not possible to spend the same coin twice.

Database: Stores all the blocks in the chain and the current state, i.e. a list of public keys that can sign new transactions, and their respective asset.

The original application of blockchain was Bitcoin, in which assets are coins and transactions move money between different parties. However, thanks to blockchain's properties, a wide range of applications are possible. Some examples are notarization (tracking of land titles, academic degrees, contracts, etc) or supply chain management. Regarding Internet applications, it is possible to create naming systems (DNS-like), PKIs, access control for IoT devices, or applications to BGP security.

Finally, another important concept are *forks*. Depending on the consensus algorithm, it is possible that at some moments in time there are multiple chains. In such situation, the consensus algorithm determines which fork will become part of the chain, and discards the rest.

2.1.1 Advantages and Disadvantages

The most relevant advantages of a blockchain are:

Decentralized: No central entity controls the blockchain, it is shared among all participants.

No CAs, CRLs or certificates needed: Generally speaking, blockchains do not need digital certificates, Certification Authorities (CA) or Certificate Revocation Lists (CRL). However, some consensus algorithms use them as a basis for their operation.

Simplified rekeying: A key rollover can be performed easily by issuing a new transaction. This transaction has to allocate the assets to a new keypair controlled by the same holder. This

process can be performed without involving any third party.

Censorship-resistant: Since the control of a transaction is completely under the holder of the private key, the revocation of an asset without the legitimate holder's permission involves obtaining its private key. Even if the private key of the previous owner was compromised, ownership of the current transaction is still preserved, as opposed to the compromise of a CA's private key (or a misbehaving CA).

Limited prior trust: It is not required to trust other nodes. However, it is worth noting that some consensus algorithms rely on limited levels of trust.

Simplified management: in situations that do not require a CA, we can avoid their management overhead.

Auditable: The transactions can be tracked back in the blockchain to determine if they originate from the legitimate holder.

No single point of failure: again, due to the fact that each user controls its private key, the compromise of a user's key does not compromise the entire system. This starkly contrasts with the compromise of a CA, which can potentially invalidate all downstream certificates.

Simplified state update: PKIs need specific subsystems to update its state (e.g. issue/revoke certificates). On the other hand, in a blockchain all these operations are embedded in it thanks to its transactional nature.

Tamper-resistant: Since data can be only added but never modified, we can detect attempts to alter previous records.

Non-repudiation: All nodes share a common, immutable view on the status of the blockchain, which provides a non-repudiation mechanism.

Privacy: Entities participating in the blockchain can achieve privacy using anonymous keys, i.e. randomly-generated keys not related to their identity. However, (i) a new keypair should be generated for each new transaction in order to prevent tracking, (section 10 of [38]), and (ii) it is possible to link keys to users depending on the nature of the assets in the chain.

Among its disadvantages we must remark:

No cryptographic guarantees: Some consensus algorithms do not rely on strong cryptographic guarantees. As opposed to PKI-based systems that rely on strong and well-established cryptographic mechanisms, consensus algorithms ultimately rely on the good behavior of a significant portion of their users.

Costly bootstrapping: When a node is activated it has to download and verify the entire blockchain.

Large storage: The size of the blockchain keeps growing forever, because data blocks are always added. This may result in scalability issues.

Slow updates: New transactions have to be verified, added to a block and received by all nodes. This results in a delay since the transaction is created until it is finally available to all the nodes. This delay will depend on the consensus algorithm, the block creation rate, and the network latency.

Complex revocation: once a transaction is added to the blockchain, only the recipient can revert it. This situation is not always desirable, for example, in case the private key is lost or stolen. On the other hand, it is possible to devise mechanism to cope with this limitation (section 2.2).

Finally, table 2.1 presents a comparison of PKIs and blockchains.

Concept	PKI	Blockchain
Revocation	Complex management	Not supported natively
Trust	Required	Limited or not required
Auditability	Complex	Built-in
Rekeying	Cumbersome	Easy
Technology maturity	Very mature	Early stages
Security	Fully tested	In progress

Table 2.1: Comparison between classical PKIs and blockchains

2.1.2 Relationship to Security on the Internet

From an Internet security perspective blockchains represent a major breakthrough. Until now, the only way to provide information security and privacy across the Internet was by means of PKI systems (Public Key Infrastructure), that is, hierarchies of certificates with a Certification Authority (CA) on top.

With blockchain, we can move away from these centralized architectures to more decentralized ones, in what could be called *flexible trust*. We can balance power between users and authorities along a line, with PKIs on one end (centralized: the CA has full control), and Bitcoin-like on the other (decentralized: only users can alter its own state). Between these two ends, there is a wide range of options that trade-off trust (centralization) for decentralization (figure 2.2).

This is because with blockchain we can enforce any set of rules its participants agree upon. For example, we could sit in a middle point between a PKI and Bitcoin: users always own their assets, but a central authority can revoke them only in special situations (that need to be proven in the blockchain). This depends heavily on the use case.

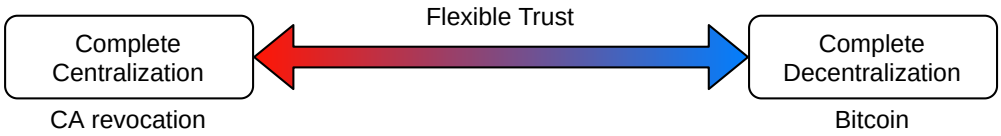


Figure 2.2: Balance between complete centralization or decentralization.

2.2 Consensus Algorithms for Blockchains

The consensus algorithm is the most critical subsystem in any blockchain, because it ultimately determines a wide range of performance and security parameters, such as time between blocks, maximum number of adversaries, computational complexity/cost, etc. This relevance has caused a huge amount of research in this area, in order to improve previous algorithms and, in turn, blockchain implementations.

We could define a consensus algorithm as a set of rules that ensure agreement among all the participants in the blockchain, so that they reach exactly the same conclusion regarding its state at any point in time. Formally speaking, consensus algorithms are a new solution to the Byzantine Generals Problem (section 2.2.1), usually know as Byzantine Fault Tolerance.

In this section we provide a high-level overview of all the available consensus algorithms, outlining their main benefits and disadvantages. This list is not exhaustive, it rather aims to group the algorithms by families based on the fundamental idea behind each one, with some examples. Interested readers can refer to more detailed surveys on the topic [39, 40].

2.2.1 Theoretical Foundation: Byzantine Fault Tolerance

From a research perspective consensus algorithms are a novel way to resolve the Byzantine Generals' Problem [41]. We can find a similar formalization dating back to 1975 [42]. In this problem, a group of generals have to agree on the best moment to attack an enemy city at the same time. If each general attacks at a different moment, their chances of winning are drastically reduced. The generals are located in different areas surrounding the city and can communicate each other. However, their messages can be intercepted and delayed, modified or dropped. Which protocol should they use to agree on the best moment to attack the city?

In their seminal 1999 paper [43], Castro and Liskov introduced a practical solution based on digital signatures that could work in an asynchronous environment like the Internet but with lower latency that previous approaches. Indeed, this algorithm is the foundation of several other consensus algorithms [44]. For example, Raft [45], which is also used in some blockchains [46].

2.2.2 Proof of Work

Concept

Proof of Work (PoW) consensus algorithms consume an external resource when creating a new block for the chain. In other words, miners have to spend some resource (work) to produce a block

eligible to be added. Usually the kind of work is computationally intensive or hard to replicate, but easy to verify. Each block is assigned a score according to the work required to create it. Forks in the chain accumulate the work of each block in it. Finally, the definitive chain is the one with higher accumulated work. Note that PoW decouples the control of the chain from its users, because in some cases miners don't need to be an active user of the blockchain to be able to mine blocks.

Example

Bitcoin [38] is the canonical example of PoW, in which the resource are CPU cycles: miners have to generate a block with a hash that starts with a fixed amount of zeroes. They do this with brute force, by sequentially increasing a nonce included in the block until they find the required hash. On the other hand, the verification is trivial because members only need to calculate the hash of the block and determine if it starts with the required amount of zeroes (which is advertised in the P2P network and periodically recalculated).

At the same time, these hard-to-find hashes prevent the modification of the block or the chain, because altering a block changes its hash, and in turn the hash of its succeeding blocks (each block contains the hash of its predecessors). This allows detecting attempts to alter the blockchain.

Variants

It has been proposed using other resources instead of compute power, for example disk space [47], or algorithms to find prime numbers [48].

2.2.3 Proof of Stake

Concept

Proof of Stake (PoS) couples the blockchain users and miners. In such scheme, the new block signer is selected randomly from the pool of current users (public keys). The selection is weighted according to each user's number of coins/tokens, so users with more coins are more likely to add more blocks. The underling idea is that the more coins a user has, the lesser the incentive to tamper the blockchain.

Example

NEM is an active blockchain that uses Proof of Importance [49], a variation of PoS that takes into consideration, apart from the stake, the dynamics of the transactions (topology or frequency).

Variants

PoS algorithms are an active area of research in which we can find many proposals exploring different angles. Due to several criticisms regarding its correctness and security [50, 51], a lot of proposals focus on designing provably secure algorithms, such as Algorand [52] or Ouroboros [53], among many others. Some improve on previous protocols like Snow White [54], HoneyBadgerBFT

[55], or DFINITY [56]. Finally, other algorithms leverage additional tools: punishment mechanisms to nodes that don't adhere to the protocol, like Ethereum Casper [57], delegating block production to a subset of stakeholders, such as Delegated PoS in EOS [58], or the simplified termination mechanism in Tendermint [59].

2.2.4 Network of Trust

Concept

These algorithms leverage a user's social network by means of carefully selecting which users are trusted (similarly to PGP keyrings). Users only accept messages from peers they trust. This creates groups of users that trust each other, and also intersections between these groups. Thanks to these intersections, and after several voting rounds, nodes reach consensus regarding which transactions will be accepted. Figure 2.3 presents a simplified example, in which users from Group A can trust users in Group C because they trust Group B.

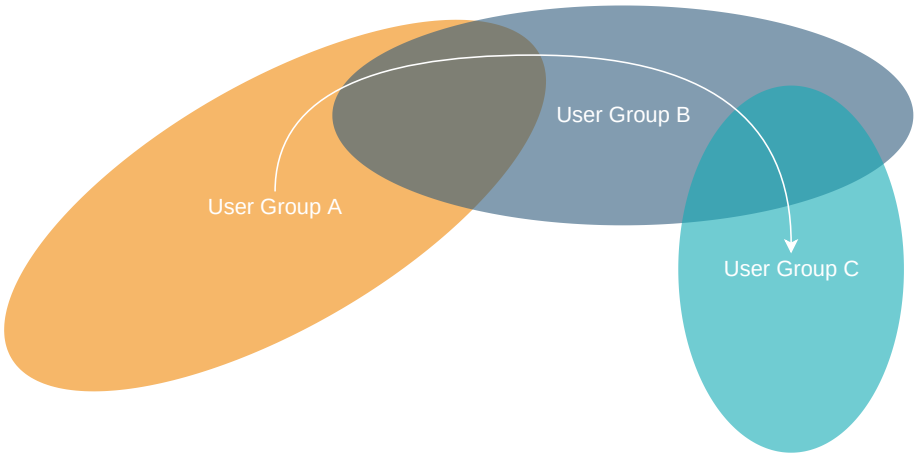


Figure 2.3: Simplified principle of operation of network of trust algorithms.

Example

The Stellar blockchain uses an algorithm with the same name [60], that creates groups of nodes called *quorum* that later reach agreement after a few voting rounds.

Variants

Ripple [61] is another example of such type of algorithms that operates similarly to Stellar.

2.2.5 Graph-based

Concept

These algorithms cannot be considered exactly for blockchains, because they do not store data in the form of blocks (chunks of transactions). However, the end result is the same, both in terms of blockchain functionality (maintaining a shared ledger), and consensus algorithm (agreeing on a shared state). The fundamental difference between a classical blockchain and a graph chain is the absence of blocks: these ledgers construct a Directed Acyclic Graph (DAG) in which nodes are transactions and edges are references to previous transactions. In this case, the consensus algorithm depends heavily on the specific blockchain. It usually boils down to an algorithm that selects which transactions have to be referenced, and that ensures that only legitimate transactions are referenced.

Example

An example of such kind of algorithms is Tangle, used in the IOTA blockchain [62]. It leverages a Markov Chain Monte Carlo algorithm to select which transactions will be approved. Since the security of this method depends on the computational power of a potential attacker, the IOTA Foundation has been running a centralized coordinator to ensure correctness, but plans to eliminate it soon [63].

Chapter 3

Public Internet

3.1 Introduction

Inter-domain routing security is a pressing issue in today's Internet. In a nutshell, inter-domain routing security encompasses the correct announcement and propagation of IP prefixes across the Autonomous Systems (AS) that conform the Internet. Currently, the protocol that communicates these announcements is BGP (Border Gateway Protocol [64]). BGP allows ISPs and other Internet companies to announce routes, i.e. how to reach a specific destination. BGP security is typically based on manual and careful configuration via out-of-band mechanisms where network operators communicate each other which prefixes to announce. Hence, an accidental misconfiguration or a malicious attacker controlling a BGP router can disrupt normal Internet routing [65]. This can lead to denial of Internet services, traffic redirection, data leaks, etc.

One of the most relevant attacks to the inter-domain routing infrastructure is known as prefix hijacking. Since BGP messages are not authenticated, it is easy to perform a BGP hijack by forging BGP announcements and propagating them to neighboring ASes. There is a long history of prefix hijacks on the Internet. As an example of this, in 2008 the Pakistani government ordered national ISPs to censor YouTube. Due to a configuration error, they attracted large portions of non-Pakistani YouTube traffic which resulted in the service being down during 2 hours worldwide [66].

Given the severity of these attacks, the IETF (Internet Engineering Task Force) has designed a solution to Inter-domain routing security by means of the RPKI (Resource Public Key Infrastructure [67]), a PKI repository to record the legitimate owners of IP prefixes, AS numbers and ROAs (Route Origin Authorization, a certificate to allow an AS to announce an IP prefix). Despite these efforts, RPKI deployment is slower than expected: only ∼14% of the total /24 IPv4 address blocks owned by the five Internet Registries are protected by the RPKI (figure 3.1).

There are several reasons that contribute to this limited deployment, related to technical issues but also to policy aspects. By policy aspects we refer to RPKI's inherent centralization, that leaves ultimate control of Internet routing to the RPKI Certification Authorities (CAs), in this case the five Regional Internet Registries (RIRs) [69]. This centralized security model does not align well with the current situation of the Internet, which is at a crossroads with different competing visions on how it should be [70], and risks splintering into isolated networks known as *splinternet* [71]. In other words, we face a dilemma between centralization and decentralization.

On the technical side, the RPKI presents various obstacles to widespread deployment: management complexity (PKIs are cumbersome to manage, e.g. when performing a key refresh), implementation challenges [72], and concerns on transparency [73]. In addition, deploying these extensions is not trivial and requires trained staff [74] and financial investment.

In the light of this situation, we propose securing Inter-domain routing by storing IP address allocation data in a blockchain. Thanks to its decentralized nature, we can distribute trust among all of its participants (i.e. all the owners of IP addresses). This way, each entity maintains its independence but at the same time they have a common framework to agree on routing security, thereby reducing the incentives to isolate parts of it. In addition, we can create flexible trust models

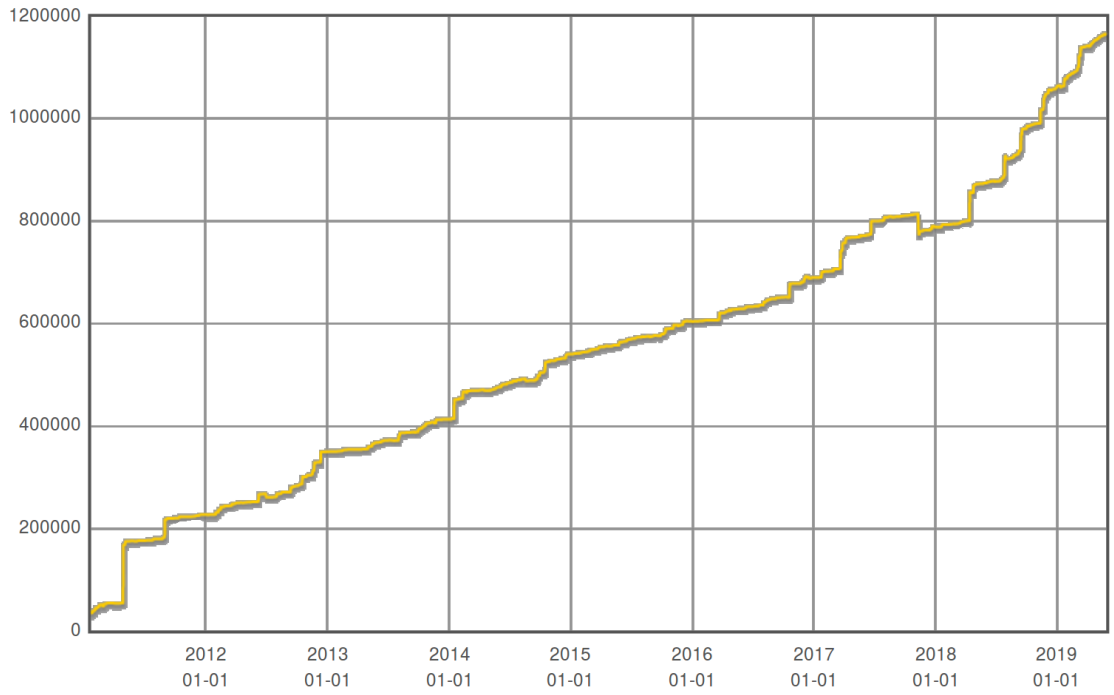


Figure 3.1: Amount of RIPE's IPv4 address space covered by ROAs, in /24 units. Of the five Registries, RIPE is the one with highest RPKI adoption [68].

that capture the complexities of geopolitics and replace single points of control (CAs) with global agreement (blockchain consensus algorithms).

Moreover, with a blockchain we can address the aforementioned technical drawbacks: (i) simplify management, especially regarding common PKI operations such as key rollover, (ii) offer auditability: blockchain's append-only ledger can detect possible configuration errors even before a modification[75], and (iii) create a consistent vision of the state that does not depend on additional systems (i.e. PKI Certificate Revocation Lists, CRLs), because all the required operations can be embedded in the blockchain.

This chapter describes IPchain, a blockchain to store IP address allocation and delegation data. The underlying argument is that IP prefixes are very similar to cryptographic coins, e.g. they are unique or can be divided into smaller amounts. Just like in Bitcoin users send money, participants in IPchain can transfer IP addresses. For example, an ISP can store its IP prefixes in the chain, along with the originating AS number. Then, other ISPs can use this data to verify the origin of BGP messages associated with these prefixes.

We have developed a prototype to assess the feasibility of our proposal, focusing on scalability and performance. Such prototype follows the blockchain transaction paradigm to allocate and delegate IP prefixes. We leverage a Proof of Stake consensus algorithm, i.e. select randomly the signer of the next block among all participants, weighted by their number of IP addresses. Finally, we performed an experimental evaluation: we converted the IP prefixes of the Internet Registries into blockchain transactions and stored approximately 70% of them in a chain of 2.5 GB.

3.2 IP Address Allocation and RPKI Architecture

The allocation of IP addresses follows a hierarchical scheme. It is usually formed of three tiers (figure 3.2): the Internet Assigned Numbers Authority (IANA), the Regional Internet Registries (RIRs) and ISPs. Initially, IANA holds all the address space, since it is charge of Internet numbers. Then, it transfers large blocks of addresses to the RIRs (1). Afterwards, the Registries delegate smaller blocks to their customers, usually ISPs (2). Finally, ISPs can delegate blocks to their customers (3). The procedure is equivalent for AS numbers.

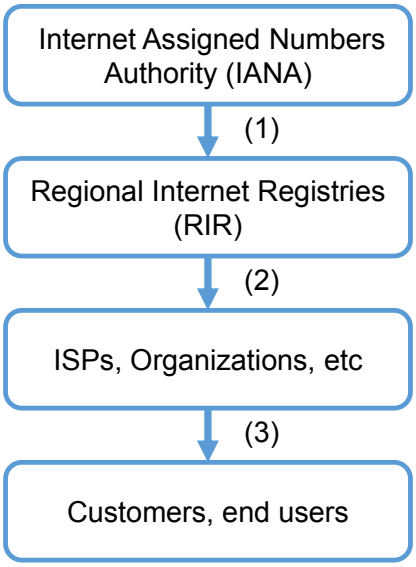


Figure 3.2: IP address allocation hierarchy.

The Resource Public Key Infrastructure (RPKI) is an IETF-defined standard to authenticate the legitimate owners of IP prefixes and AS numbers. The RPKI replicates the aforementioned IP prefix delegation structure with digital certificates, which authenticate the allocation of IP prefixes and AS numbers (figure 3.3). In summary, there are two types of certificates: Resource Certificates (RC) and Route Origin Authorizations (ROA). RCs bind IP prefixes or AS numbers to a public key (1), and ROAs specify which AS number can advertise a particular IP prefix (3). It is also possible to sub-allocate resources to other entities (2) or issue more than one ROA (4).

This way, network operators download the certificates and use them to verify BGP announcements. If the (IP prefix, AS number) pair in the BGP message does not match the corresponding certificate in the RPKI, the announcement is considered invalid.

In this architecture, trust is centralized in the five RIRs, who act as CAs and own the Trust Anchor used to validate all downstream certificates.

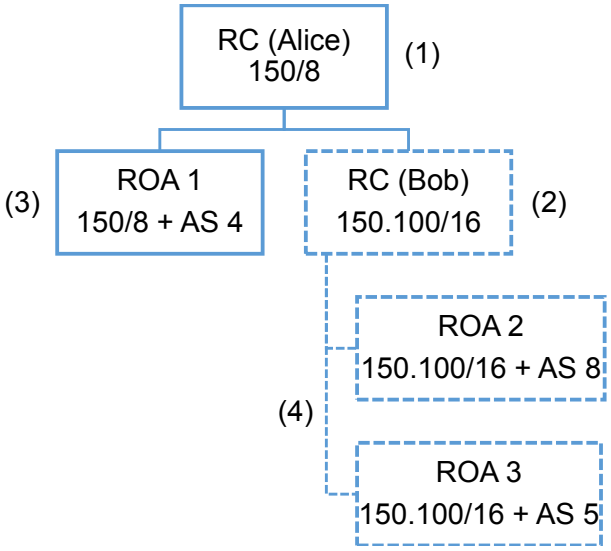


Figure 3.3: Sample RPKI certificate hierarchy.

3.3 Why blockchain?

This section we discusses the advantages provided by a blockchain when compared to the state of the art solution, the RPKI. We focus both on technical and policy benefits. By policy we refer to the possibility of modifying current trust architectures in order to better align the interests of all parties.

3.3.1 Technical Advantages

Consistent vision of the state: Both in Bitcoin and the RPKI, we need a mechanism to update the state and notify all participants, so that all of them agree on the same data. While the RPKI makes use of specific protocols to update state (e.g. RPKI Publication Protocol [76] or Certificate Revocation Lists), in a blockchain this can be directly encoded in its transactions.

Auditability: Since blockchain transactions cannot be eliminated, we can easily detect updates, for example, when a new ROA is issued for the same IP address. Moreover, this persistence prevents unwanted deletions that could affect other users [75]. Even though we can build auditability systems for PKIs like a Public Notary [77], a blockchain has this feature out of the box.

Simplified management: Common RPKI operations, such as key refresh or certificate revocation are complex and require multiple steps or dedicated subsystems. On the other hand, a blockchain makes those operations much more simple, usually it is only necessary to add a new transaction. In a PKI, when we perform a key rollover we have to re-sign all downstream certificates. For example, RFC 6489 is specifically devoted to key rollover in the RPKI [78]. In addition, if we want to revoke certificates we need manifests and CRLs. On the contrary, in a blockchain we can perform these two operations with a new transaction.

Privacy: Blockchain transactions are not linked to the user's identity, just to a public key. It is worth noting that the RPKI also offers privacy, because its certificates do not contain identity information.

3.3.2 Policy Advantages

The main benefit of using blockchain in our scenario is the decentralization of trust. As mentioned earlier, RPKI users (typically ISPs) have to trust the five RPKI CAs (one for each RIR), which act as central points of trust, and can arbitrarily revoke any downstream certificate [69]. This situation can be uncomfortable for ISPs, because IP prefixes are a key asset for most of them (their connectivity is based on proper attribution of IP addresses).

On the contrary, with a blockchain we can tackle these concerns, because their inherent decentralization leaves control of resources to the owner of the public-private key pair. Thus, in a blockchain we can effectively shift the power from CAs to the users, so that after the prefix allocation, users do not depend anymore on the actions of the CA.

Moreover, due to the fact that we can enforce complex policies inside a blockchain, it is possible to define flexible trust schemes between RIRs and users (c.f. section 3.5.5), i.e. a middle ground between complete centralization and decentralization.

3.4 Which consensus algorithm?

Consensus algorithms are probably the most important building block of a blockchain. This section details the motivations when deciding which one to use for the use case of securing IP address allocation and delegation.

3.4.1 Proof of Work

In Proof of Work (PoW), nodes in the blockchain have to solve a complex mathematical problem to add a block, thus requiring some computational effort. The definitive chain is the one with most computing power spent to create.

Despite its widespread usage, PoW is not suitable for our use case. The main reason is that the security of a PoW chain is directly linked to computing power. In other words, if we can accumulate enough computing power, we can rewrite the blockchain with false data (e.g., incorrect delegations of IP addresses). This is very expensive in blockchains accounting for millions of participants (like in Bitcoin or Ethereum), due to the large amount of computing power powering them. However, in our situation this kind of attack is feasible: consider that the current number of Autonomous Systems in the Internet is only around 65k [79].

In addition, PoW blockchains (typically) decouple its users from the management of the chain, i.e., we can transfer money in Bitcoin without having to create blocks. This user-miner separation can impact negatively on the chain, since the capability to add new blocks and the security of

the chain itself depend on the computing power of the participants (miners), which is not always aligned with their interest in the well-being of the blockchain (users).

3.4.2 Proof of Stake for IP prefix allocation and delegation

In a Proof of Stake (PoS [80]) blockchain, participants with more assets/coins are more likely to add blocks. Like in PoW, the algorithm randomly selects one participant to add a block, but it takes into account *how many* coins each user has. The underlying idea is that users with more coins (stake) have an incentive to contribute in the chain because they are its primary users.

Taking into consideration these particularities, we advocate that PoS is the most suitable option for our use case, due to three key reasons.

First, in PoS only blockchain users can make modifications, i.e. we don't depend on external actors and their computing power. This is of paramount importance in our scenario, because users that own a large quantity of IP address will have higher chance to add blocks. Usually, such participants also profit from an Internet that operates properly, so they have a clear motivation to keep its normal operation. In other words, blockchain users do not have any incentive to forge information because they would suffer the consequences: an insecure Internet [81].

Second, with a PoS algorithm we can reduce the risk of takeover, i.e. buying a large amount of assets in order to accumulate enough stake to rewrite the blockchain. In our context, this means buying a large amount of IP prefixes from other participants. However, it is not clear that an attacker may be able to perform such attack, because the other users lack a clear reason to sell their IP addresses: as we mentioned previously, they are an important economical asset for most ISPs.

And third, with a PoS algorithm we benefit from a low computational cost and we don't need special hardware. These two facts lower the entry barrier for new participants in the blockchain.

3.4.3 PoS Resistance to Monopolies

Nevertheless, PoS presents a fundamental weakness: monopolies. If a participant controls half or more of the assets, it will eventually take control of the blockchain. In order to determine if this could happen in a chain for IP addresses, we have calculated how many addresses own the five RIRs and selected countries or political unions. Figure 3.4 presents the percentage of IPv4 addresses of each, derived from IP to AS mappings [82], and CAIDA AS to organization mappings [83]. As the figure shows, no country or RIR owns more than 40% of addresses, rendering PoS resistant to monopolies in this scenario. Note that we are assuming that a collusion of two or more of the presented entities is highly unlikely. A similar analysis focusing on individual companies reveals that the one with most prefixes holds 3.5% of all the advertised IPv4 addresses.

Furthermore, in some PoS algorithms we can configure the minimum number of participants needed to create a monopoly, e.g. a participant needs to accumulate at least 75% of the total stake in order to successfully perform an attack. This way, we can adapt to the changing political situation of the Internet, increasing this threshold if required (section 3.6.1).

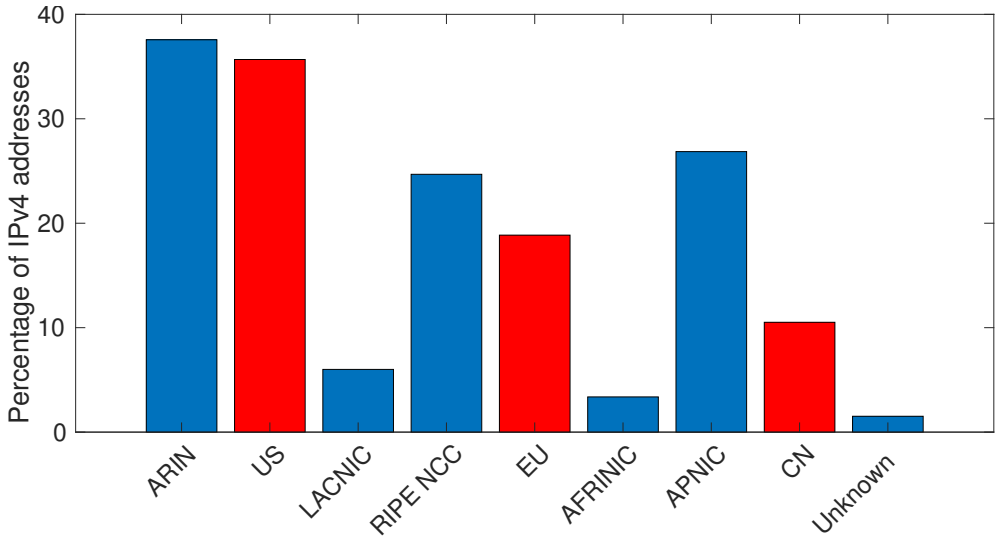


Figure 3.4: Percentage of IPv4 addresses of each RIR. Selected countries/unions are also included in the corresponding RIR.

3.4.4 Network of Trust Consensus Algorithms

These kind of algorithms, such as Stellar [60], are based on a user's trust relationships with other users (section 2.2.4). Although these algorithms present an interesting alternative, we consider them less suitable for IPchain than Proof of Stake, due to two key reasons. First, they typically require some kind of certificate-based identification system for the nodes, thus losing one of IPchain's fundamental advantages. And second, due to the Internet's decentralized nature, the trust relationships among Internet participants (mainly ISPs) are not fully well known [84, 85], making it difficult to ensure consensus among all the players. Nevertheless, these algorithms remain as a possible alternative and a relevant research question.

3.5 Architecture of IPchain

In this section we describe the architecture of IPchain regarding workflow, intended deployment, PoS consensus algorithm, and solutions to recover lost keys.

3.5.1 IP prefixes as coins

IP prefixes share some fundamental characteristics with the coins or assets we find in any blockchain:

- They are unambiguously allocated to the participants.
- Can be transferred (delegated) between them.
- Can be divided up to a certain limit.
- Cannot be assigned to two participants at the same time.

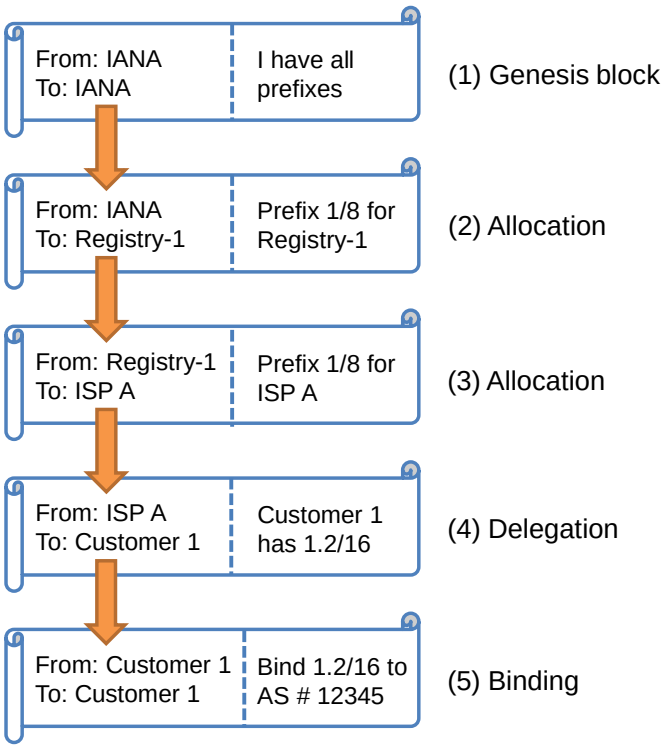


Figure 3.5: Transaction workflow example

Taking into account these similarities, we can devise a blockchain to record and transfer IP prefixes, equivalently to a financial blockchain. By chaining different transactions we can replicate the allocation hierarchy of the RPKI in a blockchain and create a consistent registry of owners of IP prefixes. Figure 3.5 shows the intended deployment of such blockchain: first, IANA, as the top-level regulator of Internet numbers writes transactions assigning all the address space to itself (1). Ideally, this first transaction is encoded in the genesis block. Second, IANA transfers large blocks of addresses to the RIRs (2). Third, the Registries allocate prefixes to ISPs (3), which in turn delegate them to their customers (4). Finally, customers bind metadata to their prefixes, such as their AS number (5).

The genesis block contains all existing prefixes, so any participant can download the blockchain, validate all the transactions and determine the legitimate owner of a particular prefix.

3.5.2 Overview

Figure 3.6 presents a sample of IPchain's workflow. First, router r1 writes in the chain its legitimate prefix and associated AS number (1). Now, consider that the announcement propagates through the network and is modified by the rough router (center). When router r3 receives the announcement of `150/8 to AS2`, it can check in the blockchain (2) if 150/8 should be originated by AS2. In this case 150/8 should be originated by AS1, so the announcement is deemed invalid.

It is worth noting that all the blockchain processes occur offline (equivalently to the RPKI) in a

standard server, so they don't need to be co-located with the router. Usually, the server generates a list of valid (IP prefix, AS number) and sends it to the BGP routers.

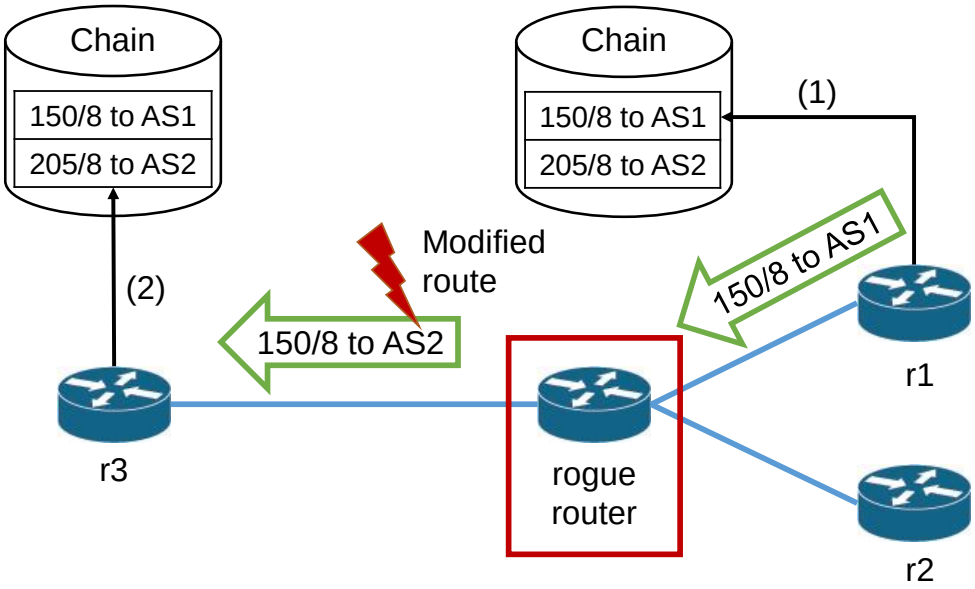


Figure 3.6: Sample usage scenario of IPchain.

3.5.3 PoS Consensus Algorithm

As mentioned in section 3.4.2, PoS fits the requirements of our use case. In this scenario, the selection of the next block signer works in the following way:

1. Count the number of addresses of each participant
2. Generate a random number
3. Select one of the participants with the random number weighted by their number of addresses

We must remark that the random number is generated in a distributed fashion, i.e. participants exchange several messages to create the same random number. This way, they will also select the same signer. The key challenge in a PoS algorithm is creating a publicly verifiable random number in a distributed environment. We can find examples in the literature on how to this, such as the Round-Robin Random Number Generator [86] or the Shamir Secret Sharing scheme [87]. Recently, new algorithms have been proposed specifically designed for blockchains (section 2.2.3), such as Ouroboros [53] or Snow White [54].

Usually, PoW and PoS algorithms include some kind of reward for new blocks, like Bitcoin, or a punishment for misbehavior, e.g. Ethereum Casper [57]. In our scenario, we consider both unnecessary: first, the reward is the security of the routing infrastructure; apart from the fact that a PoS algorithm does not require a significant financial investment. Second, in our case a punishment would mean removing the IP addresses of a participant, however, this is very unlikely in real life and can have harsh consequences.

3.5.4 Supported Operations

We define the following operations for IPchain, that are equivalent to those in the RPKI:

Allocate: Assign a block of IP prefixes to an entity, allowing it to further allocate or delegate it to other entities.

Delegate: Like **Allocate**, but without the permission to further allocate prefixes to other entities. A delegated prefix cannot be further allocated.

Metadata: Add additional data to a prefix, e.g. AS number authorized to announce the prefix.

3.5.5 Flexible Trust: Revocation

Blockchain transactions are irreversible, i.e. once we have allocated an IP prefix to an entity, we cannot undo or modify this transaction. Generally speaking, this is desirable from the point of view of the ISPs, but there are some situations when it is necessary to reclaim a block of addresses, e.g. stolen or lost keys, human error, misuse, etc. In addition, IP addresses are a finite good and must be preserved: the loss of an IP prefix impacts the whole community, as opposed to a cryptographic coin: only its owner is affected.

Taking into account that in a blockchain we can define an arbitrary set of rules, we can design some schemes to recover a block of addresses, but at the same time preserve the decentralization in any blockchain. For instance, we can let a widely recognized third party (e.g. IANA) resolve disputes between conflicting parties by issuing a special transaction that reallocates the resource. Other mechanisms are possible: time-limited allocations, multi-signature transactions, etc.

Nevertheless, the revocation approach should be agreed among the relevant players (IANA, RIRs, ISPs, institutions, etc). It is worth noting that behind these mechanisms there is a fundamental trade-off between complete centralization (traditional PKI, trust the upstream provider) and total decentralization like in a blockchain (section 2.2).

3.5.6 Other considerations

Rekeying

Rekeying is a common operation in the RPKI and involves re-signing all downstream certificates with the new key. On the contrary, this operation is greatly simplified in blockchain: we only have to add a new transaction re-allocating the IP prefix to a new keypair controlled by ourselves. In addition, since transactions are independent from each other, we can perform rekeying operations individually without affecting other users.

Privacy

Since IP addresses are linked to their owners' public key, it is not possible to identify the holder only with the data in the blockchain. In that sense, blockchain offers a similar degree of privacy to the RPKI.

IPv6 support

Since IP version 4 and 6 prefixes are currently being used, IPchain needs to support both. However, this is not trivial because there are more IPv6 addresses (128 bits) than IPv4 (32 bits). In a PoS blockchain, randomly selecting from both pools of addresses would create an imbalance of power between and v6 and v4 owners (the first would create much more blocks than the latter).

Taking this into account, we suggest never mixing v4 and v6 addresses in IPchain, but create alternatively blocks of v4 or v6 transactions. For example, even blocks contain v4 transactions and are signed by the owner of a v4 prefix, and the same for odd blocks and v6 transactions. This solution does not require two different blockchains and isolates v4 and v6 stake.

Finally, it should be noted that, due to the huge size of v6 address space, large parts remain unallocated and still owned by IANA (less than 0.5% of v6 address space has been allocated to the RIRs). This space should be ignored (not counted) to avoid IANA signing nearly all v6 blocks and thus, preventing an IANA monopoly.

3.6 Implementation

We have built an open-source prototype and made it publicly available [88]. We did not to fork an existing blockchain implementation since they do not fit our needs, particularly regarding the PoS consensus algorithm.

The IPchain prototype is written in Python (figure 3.7), and supports typical blockchain operations: receive new blocks and transactions, validate new blocks and add them to the chain, maintain a pool of unconfirmed transactions, create new blocks, etc. It also implements the operations defined in section 3.5.4 for both IPv4 and IPv6 addresses. With the aim to ease user interaction, the prototype reads new transactions from a file, signs and sends them to the network, and includes a keystore to encrypt the user's private keys.

In addition, the prototype interfaces with OpenOverlayRouter (OOR [89]). OOR is an open-source implementation of the Locator/ID Separation Protocol (LISP, RFC 6830 [90]) that creates programmable overlay network tunnels. OOR leverages IPchain to retrieve metadata related to the IP prefixes that is used in some LISP signaling messages. In other words, the prototype is designed to store LISP mapping system information. As such, apart from supporting the allocation and delegation of IP prefixes, we extended the transaction capability to also allow users to add LISP metadata with additional types of transactions. Specifically, the locator of a prefix or the IP address of the Map Server containing the mappings. In what follows, we describe relevant modules of the prototype.

Data Structures: IPchain builds on Ethereum's account system, which maps pairs of blockchain addresses with the associated IP addresses. Transactions are encoded as modifications to accounts. We chose this model instead of Bitcoin's UTXO because it requires less storage and data access is easier. We modified the PyEthereum Trie, DB, Utils and Transactions classes [91] to fit our needs, and capped the block size at 2 MB.

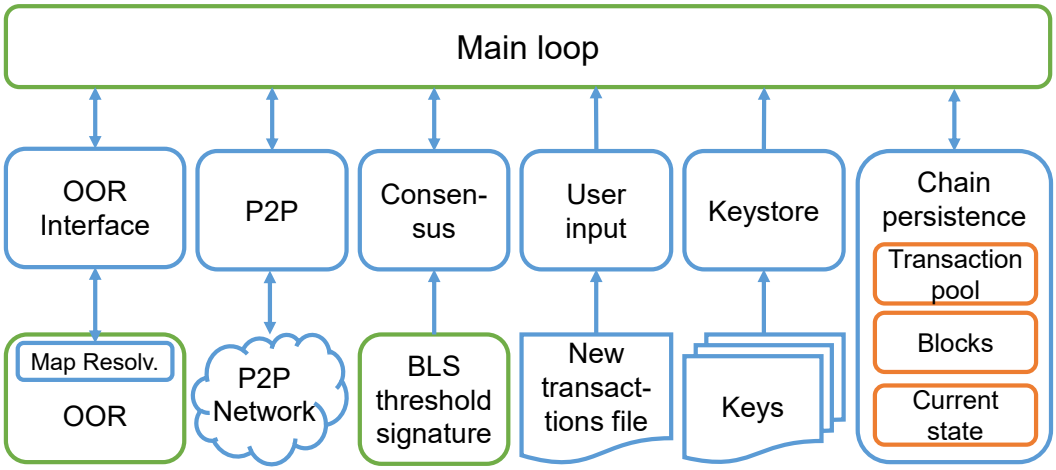


Figure 3.7: IPchain prototype architecture.

Peer-to-Peer Network: The P2P module implements all communication functions in a broadcast-all fashion, leveraging Pyhton's Twisted library for network communication [92]. Since it does not connect all the nodes between themselves, a Distributed Hash Table [93] keeps track of the last block number, so that if a node misses some blocks it can request them.

OOR interface: A simple request-response interface connects OOR and IPchain via local sockets. OOR sends queries requesting the metadata associated with a specific IP address. IPchain maintains an internal tree that maps IP prefixes to blockchain addresses, so it can locate the associated account in the state tree, retrieve the metadata and send the response back to OOR.

3.6.1 PoS Consensus Algorithms

Algorithm Selection

We can find several PoS algorithms already in production systems, such as NEM's Proof of Importance [49] or Ethereum Casper [57]. NEM's Proof of Importance is a modified version of PoS that takes into account not only the stake but also the transaction graph topology and frequency. However, in IPchain we cannot rely on the frequency of transactions like NEM because our scenario is not as dynamic as financial systems, both from a topology and frequency standpoint.

Another promising PoS algorithm is Ethereum's Casper, still in development. However, Casper uses a punishment mechanism that, as we described earlier in section 3.5.3, is not suitable for our use case. Other proposed algorithms, like Algorand [52] or Ouroboros [53], are a good fit, but we were unable to find an open-source implementation.

Algorithm Implementation

We finally leveraged part of the DFINITY blockchain PoS algorithm for our chain [56]. This chain operates a complex three-step algorithm that combines a Decentralized Random Number Generator (DRNG), a block notarization process and a finalization process. The two latter are used to resolve chain forks and achieve higher scalability. For simplicity, our prototype only uses the DRNG and does not tolerate chain forks.

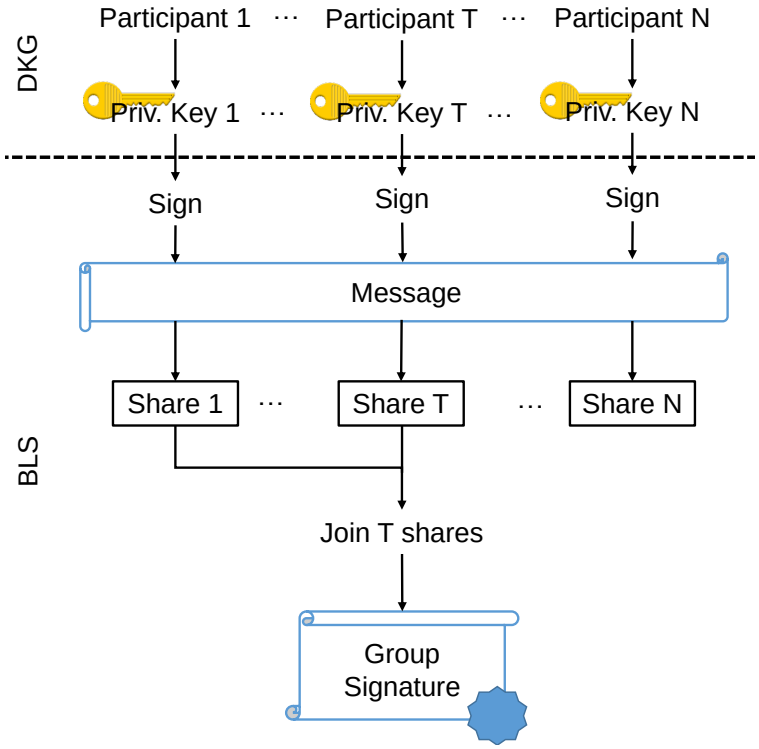


Figure 3.8: DKG (top) and BLS (bottom) operation.

The DRNG builds on the well-know topic of threshold signatures (figure 3.8), in which a group of participants jointly generate a group key and a set of private keys, known as Distributed Key Generation (DKG). Later, they can use these private keys to sign a message; however, the signature is only valid if a portion of the participants (defined by a tunable parameter, the Threshold) have signed the message. The scheme we use there is the Boneh-Lynn-Shacham (BLS). The message is commonly know as *share* because it is a part of the whole signature.

In the blockchain context, we calculate the random number as the hash of the signed message (n.b. all participants sign the same message). Due to the fact that we cannot recover the signed message unless Threshold participants have signed it, we can be sure no single participant can manipulate this number. We repeat this process for each new block to get a fresh random number to select the next block signer. In addition, the Threshold signature scheme offers two key advantages: (i) since it is not necessary that *all* the participants sign the message, we can tolerate several malicious or disconnected nodes, and (ii) we can adjust the Threshold to tune the upper limit

of malicious or disconnected participants depending on the current situation, as we mentioned in section 3.4.3.

Finally, our prototype also: (i) repeats the private key generation process after a predetermined number of blocks to refresh the public and private keys, and to take into account new participants that have been added in previous blocks, and (ii) in order to scale more easily, not all blockchain addresses take part in the DRNG, we select some of them with the previously generated random number.

3.7 Experimental evaluation

We carried out several experiments in order to determine the applicability of IPchain in real life. We measured several blockchain metrics to characterize the performance and scalability of IPchain, focusing on six key metrics: (i) throughput, (ii) block time, (iii) bootstrap time, (iv) chain size, (v) block time depending on BLS threshold, and (vi) DKG refresh time. Finally, we present an analytical estimation of the required storage in the long-term.

3.7.1 Blockchain Metrics

Throughput

For this test, we encoded in the genesis block the entire v4 and v6 address space, splitting them in large blocks of addresses, similarly to IANA's registries of v4 and v6 address space [94]. In order to simulate the typical allocation scheme of IP prefixes (section 3.2), we generated three groups of transactions. The first one referenced prefixes in the genesis block and split them uniformly into smaller prefixes, and the second also created smaller prefixes but referencing the transactions from the first group. Finally, we extracted the prefixes for the third level directly from the publicly available lists of the Registries' address allocations [95], with the final goal of storing real prefixes in the chain. In total, we generated around 380k transactions (82k in levels 1 and 2, 298k in level 3 - two thirds of the 430k in the RIR files).

We set up a single node (VM with two associated virtual CPUs, Intel Xeon Platinum @ 3 GHz, 4 GB RAM) with these transactions, all the associated keys and a configured DKG with 100 keys and 66% BLS Threshold, 40s block time and 2 MB block size. We can see the result of this experiment in figure 3.9, which plots the number of transactions per block, separated into v4 and v6 transactions. We can see two distinct phases in the experiment: before and after block ∼1580. In this moment we increased the transaction injection speed from 1tx/s to 10tx/s. Indeed, we can see that the number of transactions per block increases from 25 to 150, approximately.

During the whole test, and especially after block 2000, we observe a high variability in the number of transactions per block, roughly spanning from 100 to 250 transactions. This is due to the non-uniform distribution of v4 and v6 transactions in the input file: the proportion of v4 and v6 transactions is not constant for the entire file. Since the node processed transactions at a fixed speed, regardless if they were v4 or v6, in a given period of time a we may not inject a constant

number of v4 or v6 transactions. We can also notice this in the reduction of v4 transactions when the number of v6 increases, and viceversa.

The maximum number of transactions per block revolves around 400, so we can estimate IPchain's throughput to be approximately 10 transactions per second, in the same order of magnitude of Bitcoin. If we consider that average of number of BGP updates is between 10-15 per second [79], and that our system targets a subset of those, we can conclude that IPchain presents sufficient throughput for this application.

Blocktime

In order to verify the correct operation of the prototype, figure 3.9 also presents the time between each consecutive block in the right y axis. We can see that in nearly all cases it remains in the configured interval of 40 seconds, since we automatically trigger the block creation just after 40s from the timestamp of the last block. In some cases, due to data processing delays it reaches 41s.

Bootstrap test

With the aim of quantifying IPchain's cost in terms of time and compute resources, we performed a bootstrap test, i.e. adding a new node to the network and measure: (i) time to validate all the chain, and (ii) total chain storage. We must note that the time to download the blocks is negligible compared to the validation time. We used a VM with two associated virtual CPUs (Intel Xeon Platinum @ 3 GHz) and 4 GB RAM. It took 3.5 hours to verify the chain, that contained 350k prefixes and required 2.5 GB of storage. This last metric lets us conclude that the chain can scale well in terms of storage for this scenario.

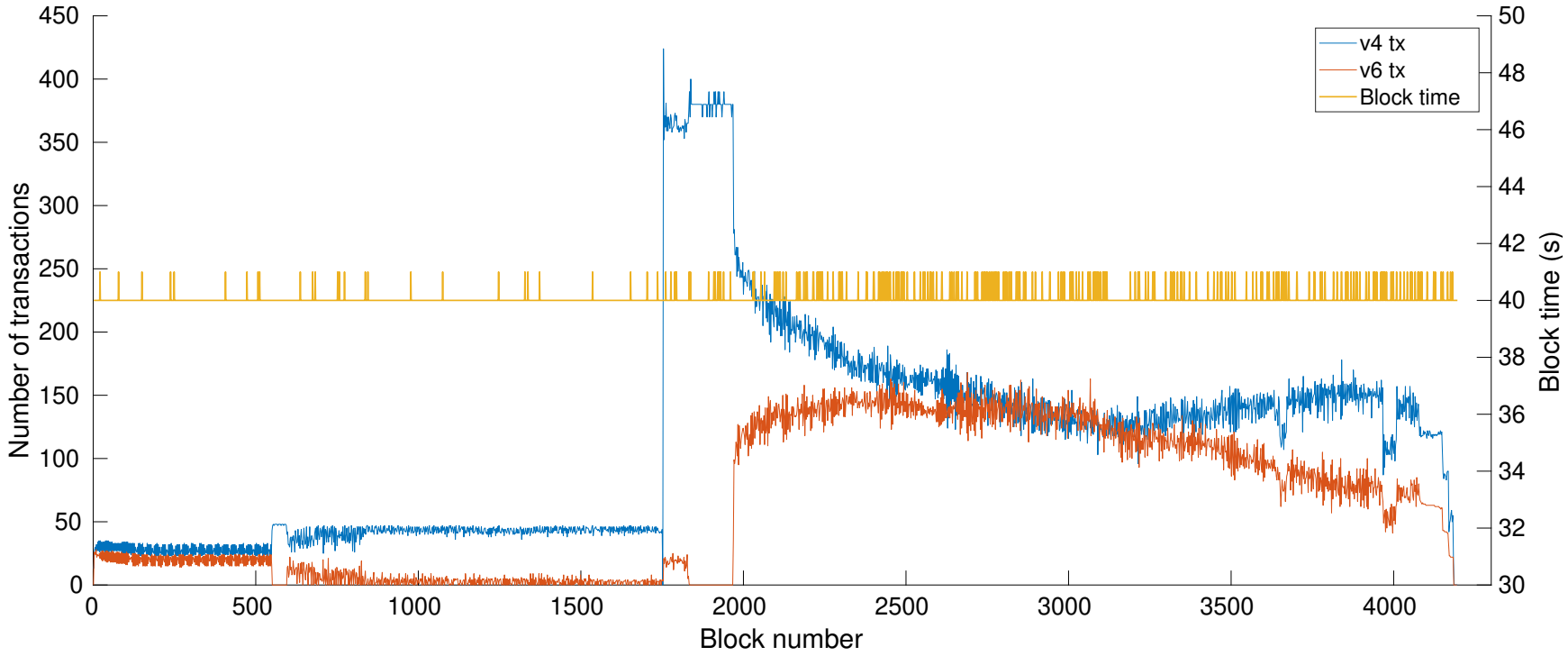


Figure 3.9: Number of transactions in each block and block time. We can see the consequence of the non-uniform distribution of v4 and v6 transactions especially after block 2000.

3.7.2 Cryptography Metrics

Block time depending on BLS threshold

In this test we aim to calculate the minimum block time depending on the BLS threshold setup, which in turn determines the chain throughput. We define the minimum block time as the time it takes to generate the BLS shares, send them to the network, recover the group signature, calculate the random number and create an empty block. Figure 3.10 presents the average of this time for 100 blocks for different threshold values. We set up 10 nodes in a cloud provider, each in a different region of the world, and VMs with two virtual CPUs, Intel Xeon @ 2.5 GHz, 4GB RAM. The nodes exchanged BLS shares, with 10 blockchain addresses for each (100 participants in the BLS in total). As we can see, the minimum block time revolves around 6.15 seconds regardless of the Threshold setup. Two main reasons explain this phenomenon: (i) the operation that joins the BLS shares is not computationally intensive, and (ii) the share size is too small (each share weights approximately 60 bytes) to cause an increase in the communication delay, i.e. the delay does not increase significantly if we send 80 shares instead of 20 across the network. These results suggest that the BLS Threshold does note have a noticeable impact on performance.

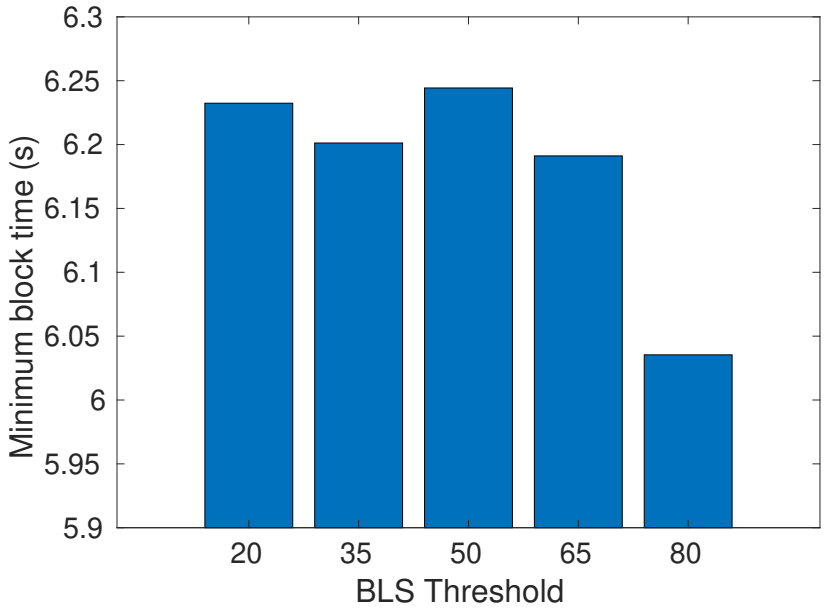


Figure 3.10: Minimum block time depending on BLS Threshold setup.

DKG keys renewal time

Finally, we measured how long it took to perform a key refresh depending on the number of nodes. We used the same cloud nodes than in the previous experiment. Figure 3.11 presents a linear interpolation of data from three experiments. We can see that the delay grows linearly with the number of nodes, basically because in this process: (i) each node has to send a message to the rest, and (ii) the crytographic function to generate the group key is more costly than the one that joins

the BLS shares.

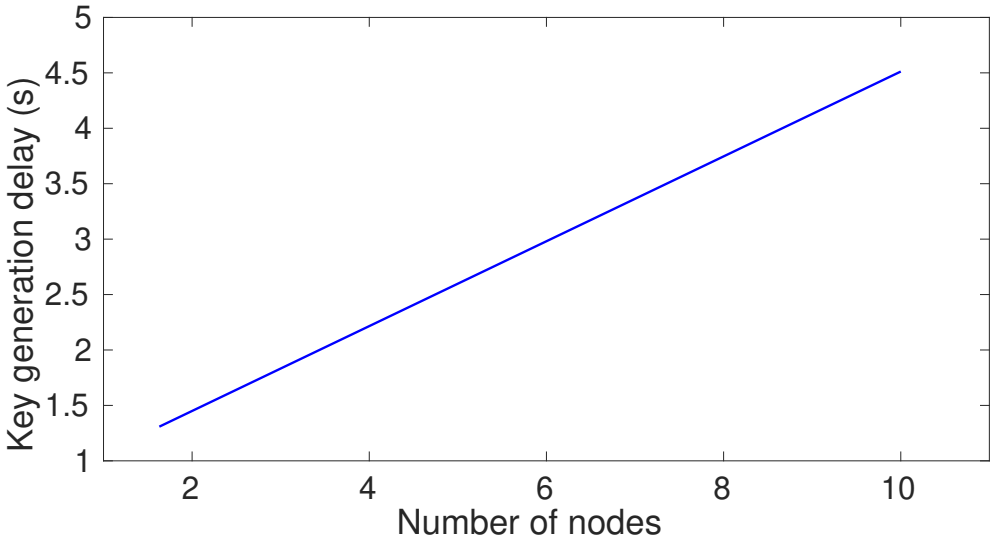


Figure 3.11: Delay to create private keys.

3.7.3 Long-term Storage Estimation

We performed a storage estimation in order to evaluate IPchain's long-term feasibility. We estimated separately the number of IP prefix allocations and AS-to-prefix bindings, and made the following assumptions. For the prefix allocation, transactions of 500 bytes, an initial load of the current 600k prefixes in the BGP RIB table, plus re-allocating all BGP prefixes each year, and a growth in new prefixes exponentially adjusted to the BGP RIB table growth [96]. Regarding the bindings of AS numbers to prefixes, we assumed one AS-to-prefix binding for each block in all the /24 IPv4 address space, each transaction weights around 400 bytes, and an update rate similar to the BGP churn [97], increasing linearly each year.

Figure 3.12 presents the estimation for both AS numbers and IP prefixes, their sum, and a comparison with Bitcoin, assuming it starts in 2019, keeps the 0.5 MB block size, the rate of 1 block each 10 min, and all blocks are full. We can see that prefixes account for a reduced part of the transactions when compared to AS bindings; this is due to the continuous increase in the churn and the number of bindings. Nevertheless, in 20 years' time the chain accounts for approximately 600 GB, a figure that is easily attainable by current storage systems. Finally, we can see that in this 20-year interval, we stay below the requirements of Bitcoin, which supports our thesis that a blockchain for IP addresses has moderate storage requirements in the long term.

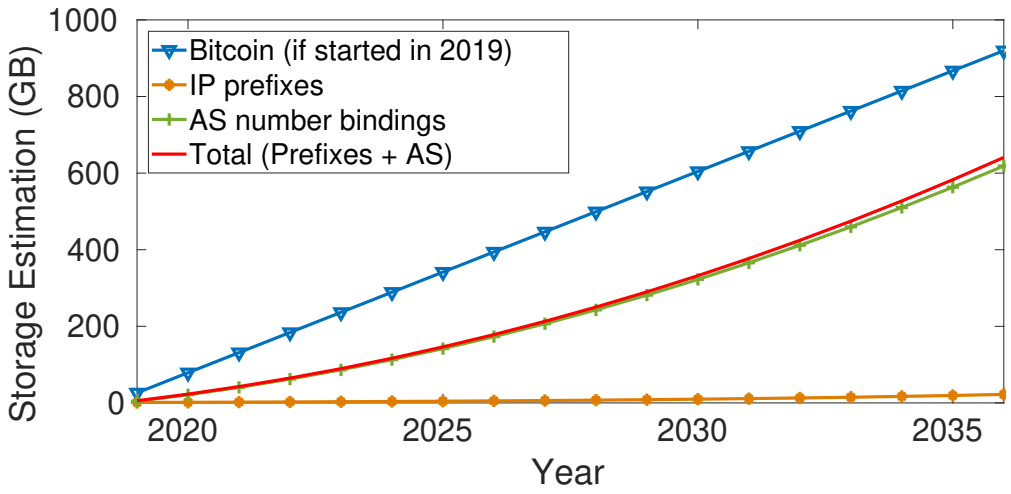


Figure 3.12: 20-year storage estimation for IPchain

3.8 Related Work

3.8.1 Blockchain Applications to Networking

We can find several proposals in the literature oriented towards providing network services or applications on top of blockchains [31], such as the Internet of Things [98], BGP messages [99, 100], Information Centric Networking [101], mesh networks [102], or distributed access control [103]. However, the largest body of work focuses on providing naming applications with similar functions to the DNS: Namecoin [29], Blockstack [30], Ethereum Name System [104], etc.

3.8.2 IP Address Allocation in the Public Internet

To the best of our knowledge, IPchain is the first blockchain specifically tailored for the allocation and delegation of IP addresses. The first reference that mentions storing Internet routing data in a blockchain is [99], but without providing any potential design guidelines, consensus algorithm, or implementation. The most closely related work has been published later [105], and advocates for an automatic IP prefix distribution system on top of an Hyperledger private blockchain. However, in our proposal we maintain the manual allocation of IANA and RIRs and we use a PoS consensus algorithm instead of Hyperledger's certificate-based system. Other similar works focus on embedding BGP path announcements in the blockchain [106], or deploying an IP allocation system as a smart contract in Ethereum [107], but neither of them focus on the benefits of PoS algorithms.

3.9 Summary of Outcomes

In this chapter we have introduced IPchain, a Proof of Stake blockchain to store the allocation and delegation of IP addresses. We have discussed its benefits over existing systems, both in the policy side and the technical side. On the policy side, we argue that the decentralization of

blockchains can mitigate the centralization concerns of current systems, offering an alternative to distribute trust among all its participants. With respect to the technical issues, we have considered how a blockchain can ease management or offer auditability when compared to current solutions. Moreover, we have emphasized the advantages of a Proof of Stake consensus algorithm for our specific use case. Finally, we have evaluated the performance of our prototype storing Internet Registry data, and demonstrated that it can achieve the requirements for real-world deployments regarding throughput and long-term storage.

Chapter 4

Private Networks

4.1 Introduction

Group-Based Policy (GBP), or policy-based networking [108] is a declarative approach to defining network behavior. Network administrators specify network endpoints, groups of endpoints and their policies using a high level language, which is later translated to network configurations. GBP is widely employed in the industry, for example in OpenStack's Neutron network API [109]. It can be used to define rules between servers and clients, create service chains, etc.

Until now, GBP has been conceived as a language for a single administrative domain. In this chapter, we analyze if we can extend it to several administrative domains, preserving at the same time their independence. For example, we want to make it possible for an administrator in company B to allow a VPN connection from a user in company A by simply typing:

```
createPolicy from=userA to=VPNserverB action=allow
```

Typically, there are two solutions for this scenario: (i) manual, by means of issuing a digital certificate and giving it to the users (so they use it later to authenticate the connection), or (ii) leveraging structures based on PKI systems, namely cross-domain certification or bridge CA certificates [110]. These structures allow the co-existence of several CAs and ensure mutual trust.

However, these approaches present some limitations that have hindered their deployment. First of all, scalability: in scenarios with thousands or tens of thousands of users, the manual approach is unfeasible, and cross-certificating N domains means -in the worst case- issuing $\sim N^2/2$ certificates [110, 111]. Second, granularity: it is not possible to define different policies for different users without issuing more certificates, further affecting scalability. Finally, management: PKIs are cumbersome to manage, especially day-to-day operations like adding and removing users, revocation (requires a CRL subsystem) or key rollover.

With these limitations in mind, in this chapter we propose using a blockchain to overcome them. In such blockchain, each organization defines its users and resources, and specifies which users -from other organizations- can access its resources. Upon an access request, routers query the blockchain to verify authorization (figure 4.1).

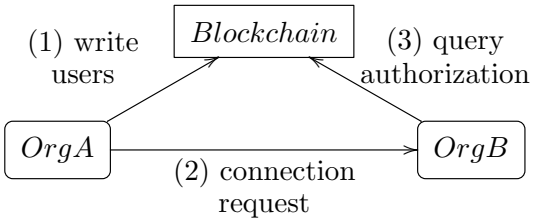


Figure 4.1: Global architecture.

Thanks to blockchain's particular properties, we can design an access control system that improves on several of PKI's limitations:

Increased scalability: When we establish a new relationship in a PKI, we have to cross-certificate the new entity with the rest. In a blockchain, however, we can directly reference previous

transactions/users. This reduces the number of required certificates (transactions in this case).

Improved granularity and flexibility: Since we can associate each resource or user with a private key, we can alter its state without affecting the rest. This includes both its validity and other data, for example, we can assign different policies to different users.

Simpler management: The transactional nature of blockchain makes management simpler: the aforementioned common operations (key rollover, revocation) can be encoded as new transactions, instead of requiring a dedicated subsystem, like CRLs and manifests.

In this chapter we present an architecture to support this use case, which actually is a multi-domain mapping system. Indeed, the blockchain is a mapping system shared by all the companies in order to connect different overlay networks across an underlay, for example the Internet. We describe a practical end-to-end implementation and evaluate its performance to demonstrate its feasibility. Our results show that we can store thousands of access polices with modest storage, and achieve linear update times on a permissioned blockchain.

4.2 Why Blockchain?

Our use case presents two particular characteristics: (i) its participants have limited trust in each other, and (ii) they want to retain full control over the access policies. This is because in a multi-enterprise scenario: (i) companies are not willing to leave access control to a third party, and (ii) each company must be able to revoke any access policy at any moment in time, respectively.

These requirements match the characteristics of any blockchain. For the first demand, its consensus algorithm ensures that no single entity controls the blockchain and avoids having to fully trust its participants. The second requirement is covered by the fact that blockchain assets are controlled by their associated private key owner, not by a centralized entity.

On the other hand, an approach like this is much more complex in a classical PKI because it cannot fulfill the previous two requirements. The first one because the CA is the single point of trust in the system, which forces all participants to trust it. Furthermore, it cannot meet the second as a consequence of the centralized trust: the CA can unilaterally alter state by means of certificate revocation.

As mentioned before, other PKI schemes could provide an equivalent functionality, such as bridge CA certificates, but in this case there is still a single point of trust in the bridge CA certificate, thus it is not significantly different from a conventional CA. Cross-domain certification may prove useful, however, it presents scalability limitations because each new CA has to cross-certify with all the existing ones.

In addition, a blockchain can also alleviate these scalability concerns: we can reduce the number of required certificates (transactions in this case) since a blockchain allows directly referencing existing transactions, instead of re-certifying with the PKI CA. In turn, this simplifies the verifi-

cation of the chain of authorizations, i.e. it is not necessary to go up the CA, then down to the cross-certificated. Authorization emerges directly from the originating organization.

Finally, thanks to emerging private blockchain platforms we can provide a certain degree of privacy for their users (as opposed to public blockchains like Ethereum) and improve some of their performance metrics (section 4.3.2).

4.3 Architecture

We can describe our architecture as a three-layer system (figure 4.2):

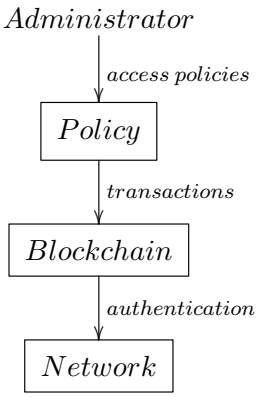


Figure 4.2: Layered architecture.

Policy: An intent-driven interface allows administrators to specify users, resources and access policies. These polices are rendered into blockchain transactions.

Blockchain: A blockchain stores all the information and ensures its integrity and accuracy.

Network: Routers access the blockchain via an API to determine if a particular user can access a specific resource. If the user is authorized, they retrieve authentication information to establish a security association and allow the connection.

The following sections provide details on each element.

4.3.1 Policy interface

Administrators use a simple CLI, based on GBP, to perform management operations, such as creating/deleting users, groups of users, policies and resources, as well as querying the blockchain for specific policies, users, etc. We have chosen GBP because it is widely adopted in the industry [109] and its semantics align pretty well with our use case.

Specifically, we can accommodate our use case to the OpenStack syntax simply re-using some of its commands. For example, consider that organization B wants to grant access to its internal

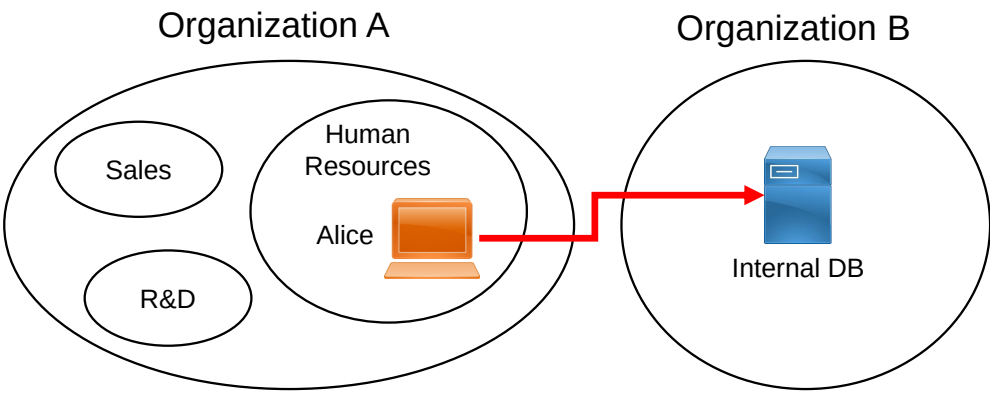


Figure 4.3: Example scenario

database to Alice from the Human Resources department of an external organization A (figure 4.3). First, company A creates a member for Alice:

```
gbp member-create alice
```

Then, company B creates a group for company A's user and adds Alice into it:

```
gbp group-create dbaccess --add:orga.alice
```

It also creates the internal database as a member:

```
gbp member-create internalDB
```

Finally, organization B creates the policy associated to its database and company A's user, allowing access from the group *dbaccess* its member *internalDB*:

```
gbp policy-rule-create external-human-res
    --src:dbaccess
    --dst:internalDB
    --actions allow
```

The GBP syntax can be extended with more options, for example, adding a one week timeout to Alice's membership in the *internalDB* group:

```
gbp group-create dbaccess --add:orga.alice --timeout 1w
```

This custom logic can be easily implemented thanks to the ability of some blockchains to run smart contracts.

4.3.2 Blockchain

In this section we discuss two major design decisions we took for our blockchain.

Participants: We believe that a private blockchain (only authorized members can access it) fits better in this scenario than a public, mainly because its participants are not willing to make their access policies public. Communicating access policies only to a group of companies is sufficient for correct operation.

Consensus algorithm: We argue that a BFT protocol suits our use case, due to the following reasons:

1. Security: classical BFT protocols such as XFT [112] or BFT-SMART [113] offer proven security guarantees, as opposed to PoW or PoS algorithms, some of which lack a formal security analysis or a mature implementation.
2. We can re-use the access control PKI of the private chain in the BFT protocol, since they require some kind of node authentication.
3. Higher throughput: BFT algorithms typically reach consensus faster than PoW or PoS, thus increasing the amount of transactions per second.
4. Immediate finality: in a BFT protocol, when a transaction has been added in the chain, it will never be removed. On the contrary, a Bitcoin fork prevents immediate finality.
5. We can avoid well-known PoW/PoS drawbacks, e.g. high energy consumption or limited throughput.

Finally, it should be noted that BFT-based chains suffer from scalability concerns, i.e., they cannot scale to as many users as well-know PoW or PoS chains like Bitcoin (in the order of millions). However, this is not our case: a chain with hundreds or even tenths of companies would be perfectly functional.

4.3.3 Network

In order to perform the access control, we have chosen the Locator/ID Separation Protocol (LISP, RFC 6830 [90]). LISP is a request-response protocol that allows the communication of control and data planes. In our scenario, routers can easily retrieve the blockchain access control policies from the control plane with minimal modification of the base protocol. For this particular use case, LISP is conceptually equivalent to OpenFlow [6]. Hence, we can use other protocols for this task, such as the aforementioned OpenFlow or P4 Runtime [7].

In a nutshell, we store the access policies in the LISP control plane and update them through the blockchain. LISP-enabled routers query the control plane to determine if a particular user can access the requested resource. Users authenticate to the router by means of including their signature in the LISP control plane messages (Map Request and Map Reply).

4.3.4 Typical Workflow

Figure 4.4 presents an example of the typical workflow in this architecture, in which two companies set up a secure connection from User A of company A to Resource B located in company B.

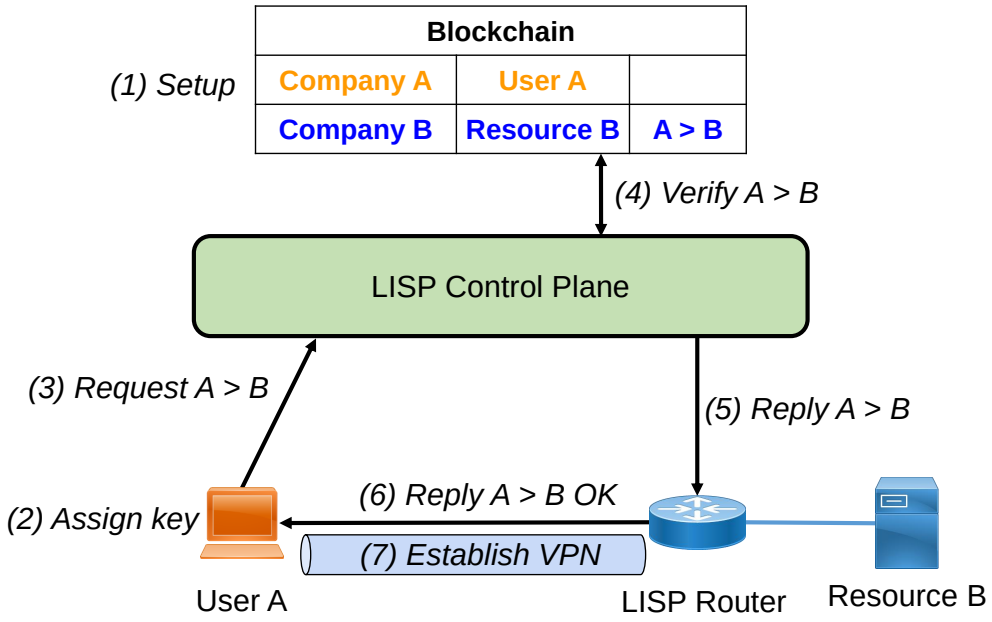


Figure 4.4: Typical architecture workflow.

1. At setup time, administrators from both companies store the required information in the chain. Company A adds User A and its public key. Company B details its resource (Resource B) and grants access to company A's user (A > B). They use a similar CLI to the example in section 4.3.1.

2. Company A assigns User A its credentials (public-private keypair), with the public key being the one in the blockchain.

3. When User A wants to connect to Resource B, it sends a LISP control message to the LISP control plane. The message is signed by User A.

4. The LISP Control Plane verifies the signature and checks the access policy against the blockchain.

5. If they are correct, it sends a reply message to Company B's LISP Router.

6. The LISP router sends a reply message with the cryptographic material for data plane encryption, in order to establish a security association with User A. This message is encrypted with User A's public key.

7. The LISP router and User A start a secure connection, e.g. with LISP-CRYPTO (RFC 8061 [114]) or other VPN protocols (L3VPN or equivalent).

4.4 Implementation

We have built an end-to-end prototype encompassing the three components of the aforementioned architecture: GBP interface, Blockchain, and Network. It is available as open-source code [115].

4.4.1 GBP Command Line

We designed a CLI inspired on GBP commands that allows the creation, deletion, and retrieval of users, groups of users, resources and access policies, highly similar to the examples in section 4.3.1. For example `create resource` creates a new resource for the organization. Additional options in the commands specify the IP address of the resource, the public key of a user, etc.

4.4.2 Blockchain

Our implementation heavily leverages the Hyperledger (HL) project, an open-source permissioned blockchain implementation. Specifically, we have chosen its Fabric [116] framework because of its maturity, flexibility and business-orientation. Moreover, thanks to Fabrics's channels, we can establish private communications among sub-sets of companies if privacy is a strong concern.

In this section we detail the different configuration parameters and rules for our prototype in the HL Fabric framework.

Assets

We defined the following elements in the chain:

Users: Source endpoints, identified by public key K+ and including other information: originating organization, LISP Endpoint Identifier (EID), name and department. The LISP EID is their overlay IP address inside the the corresponding enterprise network.

Departments: A group of users within an organization, identified by department name and their belonging organization. They identify groups of users in order to create simultaneously policies for several users.

Resources: Destination endpoints, identified by a LISP EID and the belonging organization. Again, the LISP EID is the overlay network IP address in the enterprise network.

Policies: Access control lists that grant access either from a source endpoint (user) or from a group of users (department) to a destination endpoint (resource). They are identified by a composite key (source-destination). Typically, the source endpoint is a user or department of another organization and the destination is a resource of the issuing organization. Policies can contain other information, such as the time frame in which the connection is allowed or an expiry time.

Membership Service Provider

Each organization is identified by a Membership Service Provider (MSP). MSP is the Hyperledger component that generates and manages the digital certificates to identify the participants in the blockchain.

Chaincode

We imposed as a global constraint that only the organization that creates an asset can alter its state (e.g. delete, associate with another asset, etc). We enforce this by binding all assets to their respective MSP, and rejecting any modification from a non-owner MSP.

On the other hand, any organization within the same HL channel can query information about any asset of any organization in the chain.

Endorsement policy

In our implementation, all members have to endorse any transaction. However, other schemes are possible thanks to Fabric's flexibility. Depending on the level of trust among the participating organizations and the particular use case, we can adjust the minimum number of endorsements. Some examples are: half + 1 of the members, $2f + 1$ valid signatures out of n endorsers (assuming f faulty endorsers and $n > 3f$), or AND/OR syntax (`member A OR members (B,C,D)`), etc.

Ordering Service

We leveraged the SOLO ordering service (i.e. a centralized orderer), so we could ease development. However, in a production setup Apache Kafka could be a good fit, because it can tolerate several faulty or disconnected nodes.

In scenarios with low trust among participants, Byzantine Fault Tolerant (BFT) ordering services can be easily plugged thanks to Fabric's modular design. However, we believe that a Crash Fault Tolerant (CFT) algorithm is enough for this use case since a double-spend does not make sense here[1]. In addition, HL's endorse-order-validate transaction lifecycle offers a variety of mechanisms to prevent or detect misbehavior.

4.4.3 OpenOverlayRouter Software Router

In order to effectively perform access control, we took advantage of an open-source LISP implementation, Open Overlay Router (OOR [89]). We made a slight modification to its Tunnel Router mode: when it receives a Map Request packet, it queries Hyperledger with the source and destination endpoints, via an ad-hoc API. Hyperledger checks if the pair of (source, destination) is allowed to establish a connection and notifies OOR. If they can connect, OOR then responds to the source with a Map Reply message, otherwise takes no action. This way, unauthorized users do not receive

[1]Note that in a CFT environment some attacks, such as censoring a transaction, may become feasible.

a Map Reply and do not know where to connect. Of course, a production setup requires additional security mechanisms, as discussed in section 4.3.3.

4.5 Experimental Evaluation

4.5.1 Scenario

We set up an experimental scenario on a PC running Ubuntu 16.06 and a quad-core Intel i5 CPU 650 @ 3.20GHz. Table 4.1 summarizes HL parameters during the experiment. Thanks to the Docker containerization of HL, we could emulate 4 organizations, each with 2 peers, all in the same PC. We artificially generated around 1 million policies and users to evaluate the read latency, and added at most 15 endorsers to estimate the write latency.

Number of organizations	4
Typical key + value size	32 bytes
Number of channels	1
State database	LevelDB key-value store
Endorsement Policy	AND(Org1,Org2,Org3,Org4)
Ordering Service	SOLO
Block timeout	100 ms

Table 4.1: Experimental Setup

4.5.2 Results

We carried out several experiments on our implementation to characterize its performance and have an understanding of its scalability.

Read latency

Figure 4.5 presents the average query time for different number of elements in the chain. In this case the state DB was CouchDB (HL allows using both LevelDB and Couch DB as state DB, with similar performance). We can see that it revolves around 40 ms regardless of the number of elements, because Couch DB is a key-value store (these type of databases present constant query latency). It should be noted that the query performs exact matches of pairs of source and destination IP addresses, and that future work should also support longest-prefix matching.

Write latency

Figure 4.6 plots the time required to add a new user depending on the number of endorsers in the network. As we can see, the latency grows linearly with the number of endorsers, because each new

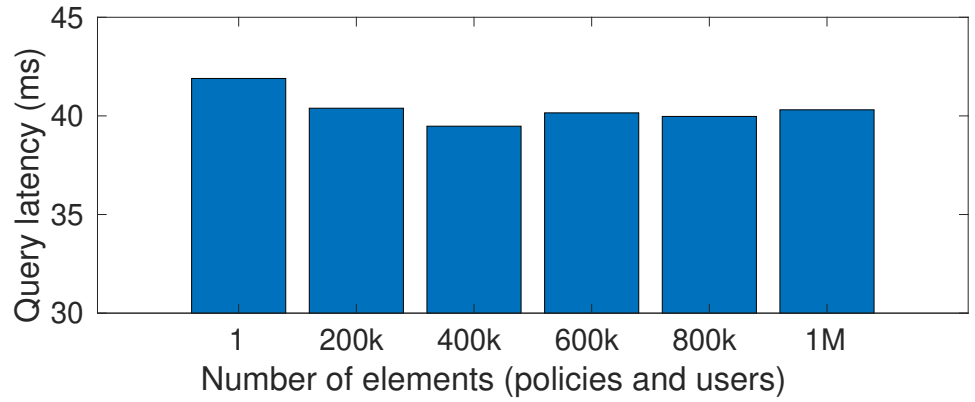


Figure 4.5: Hyperledger CouchDB query latency.

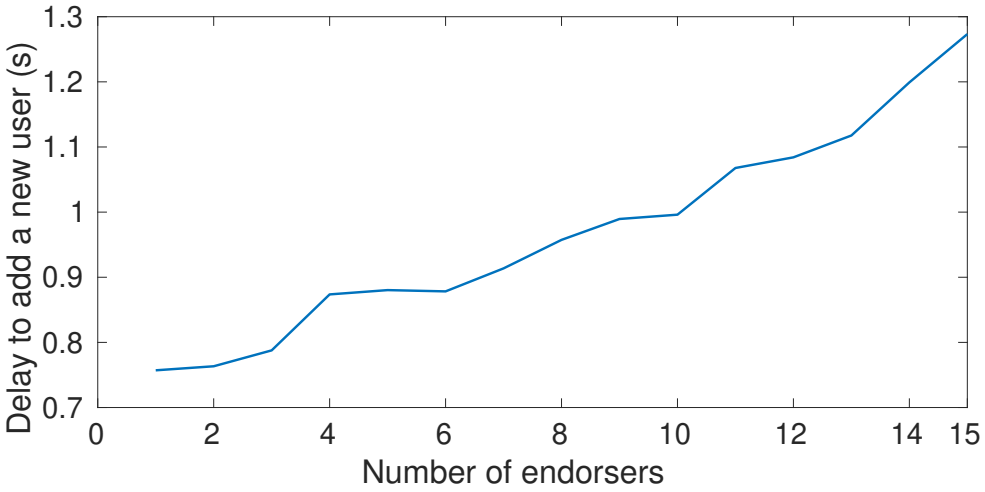


Figure 4.6: Latency to add a new user for different number of endorsers.

endorser is an additional signature that the issuer has to verify (the current HL implementation makes this verification sequentially). This result is in line with in a recent benchmark of the HL platform [117].

Chain size

We were also interested in the chain size, i.e. required storage. Figs. 4.7 and 4.8 show the total chain size depending on the number of transactions and endorsers, respectively. As expected, in both cases the size grows linearly with the number of transactions or endorsers (in the latter case because more endorsers mean more signatures per transaction). We can see that these situations require very modest storage. Thus, we can safely assume that scenarios with a considerable amount of participants (e.g. 1k endorsers would demand ∼25 GB) or a long transaction history (1M transactions take up ∼10 GB) can be easily supported.

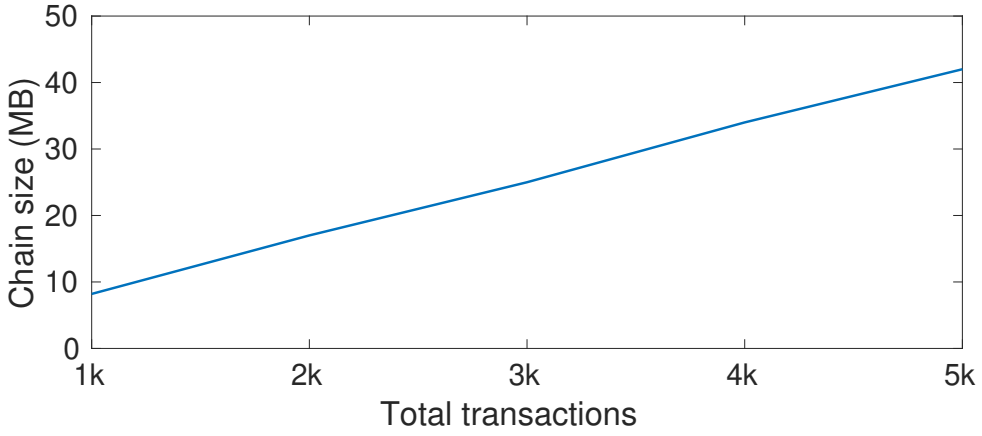


Figure 4.7: Chain size vs. number of transactions, in a setup with four endorsers.

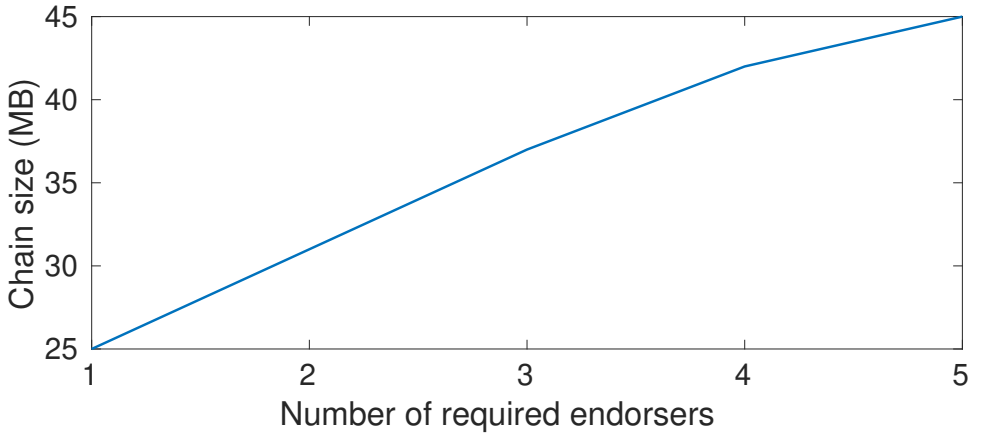


Figure 4.8: Chain size depending on the number of required endorsers (5k transactions in the chain).

Network latency

Figure 4.9 presents the query time CDF of a LISP control plane node (Map Server) storing 1k, 10k or 100k pairs of source, destination pairs. In other words, given a source node, how long does it take to find the authorized destination node(s). Since this test was performed in a local network, we consider the communication delay negligible.

We can see that the majority of the queries are completed in less than 0.35 ms, roughly two orders of magnitude below the HL database delay. This is mainly due to two reasons: (i) the Map Server is implemented in C (whereas the queries in figure 4.5 go through HL's Node.js API, CouchDB and back), and (ii) data is stored in a Patricia Trie, a tree optimized for prefix queries. In addition, the delay is independent of the number of elements thanks again to the Patricia Trie: the delay depends on the length of the elements (source endpoints, IP addresses in our implementation), not the number.

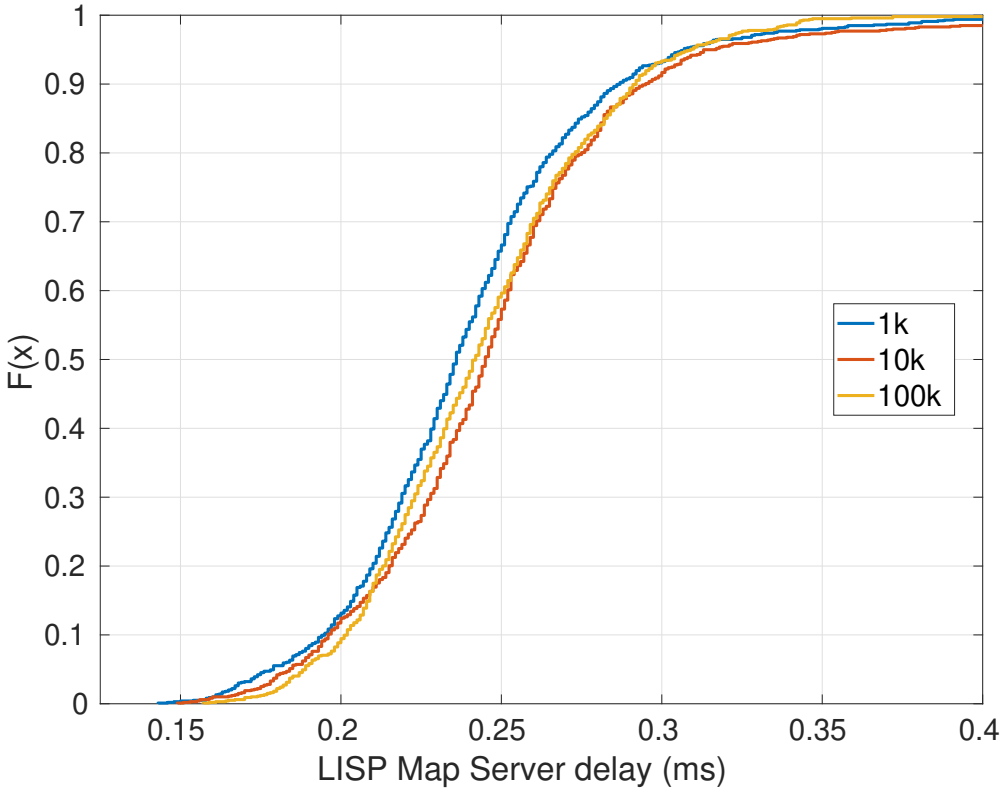


Figure 4.9: CDF for LISP Map Server query delay for 1k, 10k or 100k pairs of source, destination nodes.

4.5.3 Discussion

The previous results demonstrate that the proposed system can easily scale to meet the demands of a federation of several organizations. Table 4.2 outlines several metrics and their requirements in terms of scalability.

On one hand, both the read and network latencies can support high query rates. The read latency presents a constant response time regardless of the number of elements in the chain (in case of exact matches), and the network server storing the access policies, linear with the length of identifiers.

On the other hand, the write latency can suffer if we have a large amount organizations in the same blockchain. However, it is not as critical as the read latency because we can tolerate a delay up to several seconds when adding a user.

Finally, the chain size obviously depends linearly on the number of transactions, but also on the number of endorsers. This last relationship puts an additional strain on scalability, because it affects both chain size and write latency. There is a trade-off here between a small number of endorsers (small write latency and chain size, but more centralized trust in a narrow set of participants), and a large number of them (higher write latency and size but more distributed trust). Thus, special consideration should be put in the number of endorsers and the endorsement policy to achieve an equilibrium between a tolerable write latency and the expected number of endorsements.

Read latency (exact match)	constant
Write latency	linear w.r.t. number of organizations
Chain size	linear w.r.t. number of transactions and endorsers
Network latency	linear w.r.t. identifier size + propagation delay

Table 4.2: Scalability Analysis

4.6 Related Work

The most closely related work is [103], which implements Attribute-Based Access Control (ABAC) policies over the Bitcoin blockchain. It presents three main differences with our work: (i) it focuses on access control for individual users, unlike our organization-based approach, (ii) it allows transferring access control rights between users, and (iii) does not consider using a private chain or a different consensus algorithm. Hadi [118] proposes a data distribution system in which the blockchain is the data persistence layer, but is also user-centric and more oriented towards data storage and messaging services rather than networking. Similarly, we can find research on blockchain as an access control technology for distributed cloud storage [119, 120].

In addition, there is also a growing body of work on blockchain-based access control both for IoT: [121] leverages a blockchain to store access permissions for IoT devices with a strong emphasis on key management and distribution. [122] also provides authentication, authorization and auditing for IoT but separates them in four independent blockchains, and is generic enough to support a

wide range of access control models typical of IoT, while our proposal is restricted to a specific language, GBP.

Finally, [123] presents an interesting perspective on revocation management in distributed access control with blockchains that aligns well with our ideas.

4.7 Summary of Outcomes

In this chapter we have presented a multi-domain mapping system based on an permissioned blockchain. We have implemented and evaluated a prototype to support the use case of access control in cross-domain communications, specifically different enterprise networks that do not trust each other. In order to reduce the burden on network administrators, the front-end builds on GBP, a well-known intent-driven language. Such blockchain distributes network polices, and helps overcome drawbacks of classical solutions while at the same time maintains the independence of each organization. The experimental evaluation shows that this design can easily scale to -at least- tenths of organizations with modest storage requirements.

Part II

Architectures for Enterprise Networks

Chapter 5

Data Plane Security

5.1 Introduction

In this chapter we present a design to secure the underlay part of overlay networks based on the WireGuard VPN protocol as a data plane, and LISP as a control plane. WireGuard (WG [28]) is a recent VPN protocol that has managed to disrupt the mature field of VPNs [124–127]. One of the main reasons behind WireGuard's success is that it trades off flexibility for simplicity. As opposed to traditional VPNs that support a large set of cipher suites and that negotiate its capabilities before establishing the secure connection, WireGuard only supports one cipher suite, specifically -at the time of this writing- ChaCha20 and Poly1305 (RFC 8439 [128]). This greatly simplifies the architecture as well as the code, providing also important performance advantages.

In short, in WireGuard users only have to exchange the public keys of the endpoints and the IP address of one of them to start exchanging data. In fact, the setup is very similar to that of SSH. The underlying implementation takes care of the cryptography aspects of a typical VPN: key derivation, re-keying timers, anti-replay protection, etc. In addition, WireGuard:

- Supports mobility
- Sends keepalives to maintain NAT holes open
- Has a reduced codebase to ease auditing
- Is implemented in the Linux kernel, which yields high performance

Thanks to these outstanding features, WireGuard is gaining popularity and it is becoming the *de facto* VPN standard for certain Internet communities. However, WireGuard lacks some critical features to enter Internet professional scenarios, such as ISPs or Enterprise networks. On the other side, end users that would like to add confidentiality to their connections usually have to rely on deploying IPsec in the underlay, with its associated complexity.

One of the main limitations is that WireGuard lacks a control plane. Hence, users have to manually setup the public keys and IP addresses in all of their endpoints, i.e. when we add a new device to the network we have configure its public key in *all* the existing devices. This process is time-consuming and error-prone.

Taking this into account, in this chapter we aim to design, prototype and evaluate a control plane for WireGuard. Specifically, we focus on three key points:

1. Automate the distribution of WireGuard keys *without* relying on PKI-based schemes or DNS-based schemes (e.g. IETF DANE [129, 130]). These systems require complex configuration or additional infrastructure like a DNS server. On the other hand, although there exist commercial systems that can offer equivalent functionality, such as Dynamic Multipoint VPN [131], or modern SD-WAN systems [132], their details are not public.

2. Reduce as much as possible the number of public keys stored in the endpoints, in order to reduce communication overhead and avoid sharing unnecessary cryptographic information. Also, we want to be as fast as possible when creating new tunnels.

3. Understand and pinpoint other potential limitations of the WireGuard protocol to be considered for Enterprise or ISP scenarios.

The proposed control plane works as follows: we store the public keys of all peers in a central server, that sends them to the WireGuard peers upon request (figure 5.1). This way, users only have to setup a secure connection between the peer and the central server on bootstrap (0); new peers store their public key and endpoint IP address in the server (1). Afterwards, peers retrieve any information from the central server (2) and can establish WireGuard tunnels with this information (3). Note that the devices in figure 5.1 can be either standalone devices (for example, when securing the network of an end user), or routers connecting different branches of an enterprise network (which have an entire network behind them).

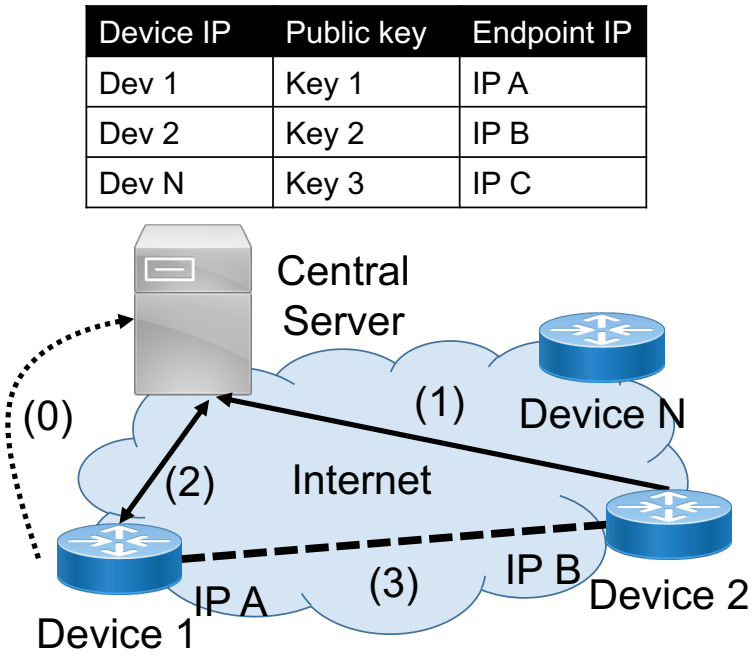


Figure 5.1: Global design. Dashed lines represent data plane traffic flow, solid lines control plane traffic.

We must remark that the idea of storing public keys in a centralized server is not new, but rather a well-know topic (section 5.7.1). On the contrary, the contributions of this chapter lie on the practical implementation side, specifically:

1. The design caveats of a centralized key distribution control plane
2. The selection of a protocol to reduce the query latency when requesting a public key
3. Implementation details
4. Performance evaluation

To the best of our knowledge, this is the first control plane for WireGuard-based VPNs.

5.2 Background: WireGuard

From a network architecture perspective, WireGuard adds two additional headers to a standard IP datagram (figure 5.2). First, an outer IP header, that contains the peer IP address. This header supports mobility; its IP address can change arbitrarily as the peer roams across different networks (e.g WiFi, LTE). Afterwards, there is a UDP header plus a custom header that contains the message type, cryptography information, and the encrypted payload. Inside the payload we find the inner IP header, the one used by applications. This header is used on the receiving end to perform access control, i.e. we can specify a list of allowed IP prefixes.

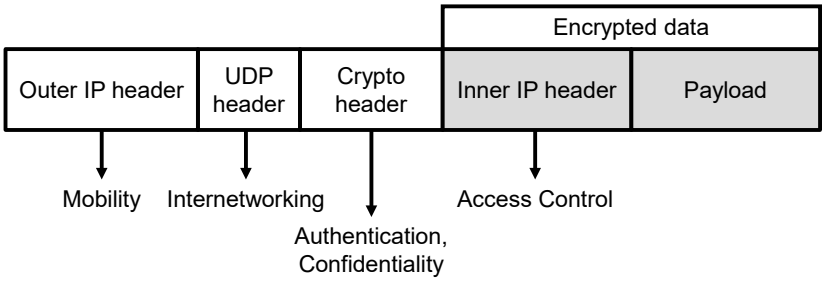


Figure 5.2: WireGuard header structure

The key advantage of this architecture is that the three headers are independent: we can modify any of them without impacting the others. For example, in case that a peer changes its IP address due to a mobility event, the crypto header and the encrypted data will be successfully decrypted in the destination. In summary, each header performs a different task:

- **Outer IP header:** mobility, can change freely
- **Crypto header:** authentication, confidentiality
- **Inner IP header:** access control

The key element of WireGuard's operation is the *cryptokey routing table*, that binds source IP addresses (usually IP addresses in the private range) to peer public keys (table 5.1). In other words, the source IP is used to determine the encryption key and the receiving peer. Additionally, it stores the Internet IP address of the peer, that is used in the outer IP header. Figure 5.3 presents the packet flow of an outgoing WireGuard packet. First, users configure the cryptokey routing table with the peers and adjust the Linux routing table to forward this packets to the WireGuard interface (2). This way, new packets destined to the peers (1) are forwarded to the WireGuard interface (2, 3). The interface does a reverse IP lookup (i.e. source address) on the cryptokey routing table to find the peer's key and Internet endpoint. In figure 5.3, the selected key is Peer B's. Using this data, it encrypts, encapsulates and sends the packet to the physical interface (4).

Allowed Source IP	Public key	Internet Endpoint
10.10.0.0/16, 10.11.0.0/16	Peer A key	80.80.80.80
172.16.1.0/24, 172.16.2.2/32	Peer B key	100.128.128.128
192.168.4.0/24	Peer C key	40.0.0.0

Table 5.1: Sample WireGuard Cryptokey Routing Table

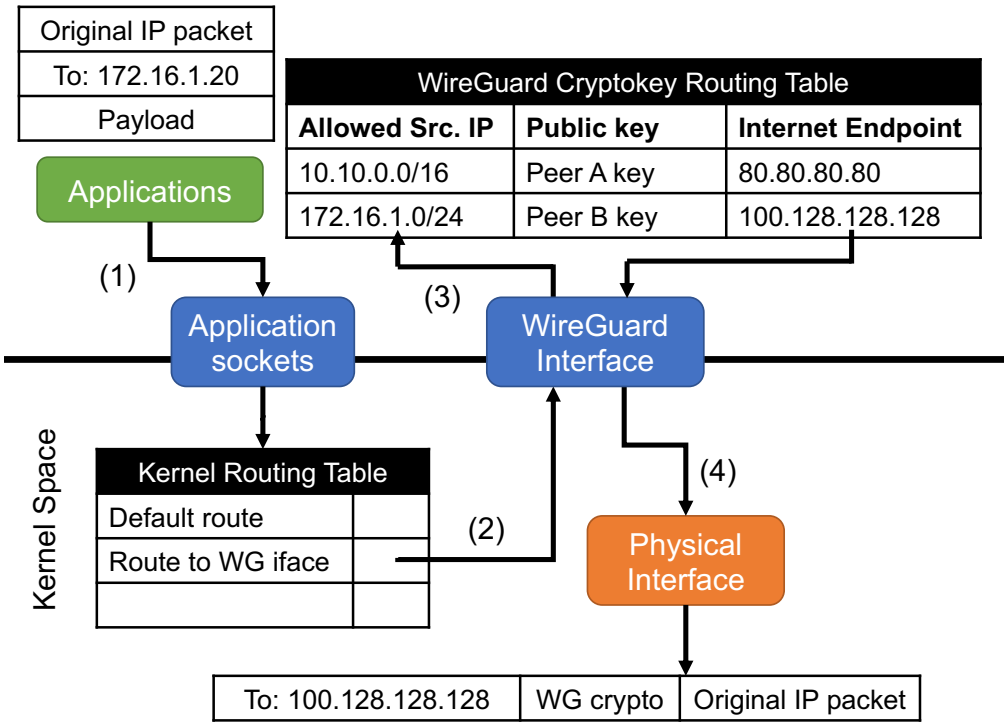


Figure 5.3: WireGuard packet transmission

Upon reception, the previous process is reversed. First packets are decrypted. WireGuard uses a local identifier (equivalently to IPsec) to match incoming packets to the their security association. If decryption is successful, the WireGuard interface performs access control. Specifically, it verifies that the source IP address of the packets is in the list of IP prefixes of the associated public key. This determines if packets are accepted or dropped. Finally, they are forwarded back to the kernel routing table, and in turn to the corresponding application socket.

We must remark that other common VPN protocols, like IPsec, TLS or OpenVPN do not support mobility out of the box like WireGuard (section 5.7.2).

5.3 Architecture

5.3.1 Threat Model

For this specific use case, either for end users or enterprise networks, we assume we can trust both the endpoints and the centralized server, i.e. we consider legitimate any control plane message they send. On the other hand, the underlying network is not trustworthy.

In order to protect the signaling between clients and server, we establish a WireGuard tunnel between them (section 5.3.3). The keys for these tunnels are part of the configuration stage and are exchanged out of band. This way client and server have a way to authenticate each other. In addition, the centralized server stores only the endpoint location (Internet IP address), and its corresponding public key. These assumptions raise a set of security concerns in both the centralized server and the endpoints that we discuss below.

Central server

Single point of failure: using a centralized server creates a single point of failure, especially vulnerable to DDoS attacks. We can mitigate this risk with: (i) replicating the central server, and (ii) since the centralized server uses WireGuard tunnels, they include an anti-DDoS mechanism, detailed in section 5.3 of [28]. This mechanism is based on a cookie from the server that the client has to re-send when establishing a connection.

Leak of peer locations and/or public keys: in this situation, although an attacker can initiate connections with the rest of the peers, it won't be able to finalize the handshake because the peers won't receive the attacker's public key from the central server (section 5.3.3). Moreover, in case of compromise it is possible to remove the affected key from the central server so the rest of the peers cannot establish communication.

Leak of the private key of the central server: this is the worst scenario, since an attacker can impersonate the central server and mount a monkey-in-the-middle-attack. The secrecy of the private key depends on the OS security precautions.

Endpoints

Leak of peer locations and/or public keys: similarly to the central server, this leak does not allow an attacker to connect to any peer, because it will not be able to successfully authenticate in the central server. It *may* be able to take advantage of peers that that previously stored its public key in their cache, though.

Leak of the private key of the peer: on the contrary, here the attacker will be able to impersonate the peer when connecting to the central server, retrieve connection data and successfully establish connections to the rest of endpoints. Likewise, it is possible remove this key from the central server to prevent future connections.

5.3.2 Discussion

The key architecture decision we took was the state distribution model: send all the keys to all peers (*push* model) versus each peer retrieving only the keys it needed (*pull* model).

We chose the pull approach for two reasons. First, security, because we avoid sharing all the public keys with all the peers. Second, minimizing the amount of state in the data plane. Since we only download the keys we need, the WireGuard cryptokey routing table is less crowded. In turn, this reduces control plane signaling overhead and increases scalability. We must remark that we are assuming that the traffic pattern among the peers is *not* full mesh, i.e., peers do not need *all* the existing keys, but just a bunch of them. On the other hand, the main drawback of a pull architecture is an increased connection establishment time. In order to minimize this delay, we chose an UDP-based protocol to retrieve the keys. Section 5.5.1 quantifies this delay.

5.3.3 Architecture Description

In a nutshell, our architecture lays two elements on top of a WireGuard deployment (figure 5.1). First, a centralized database that contains, for each WireGuard peer, its associated public key and endpoint IP address. Second, a secure control channel between the centralized database and the WireGuard peers, that we use to retrieve the aforementioned WireGuard configuration (solid lines). The central database indexes peers by their inner IP address, i.e. the one used by applications and encrypted by WireGuard.

The centralized database distributes the configuration data on demand: when a peer does not have the public key for a particular destination IP (i.e. we don't have an entry in the WG crypto table), we request the public key and endpoint IP via the control channel, and configure WG appropriately (step (2) in figure 5.1 in order to communicate with Device 2). The central server answers this request but also pushes Device 1 public key and Endpoint IP to Device 2. This is due to the fact that, in order to successfully establish a new WireGuard tunnel, both peers need to have each others' public key. In other words, we use a pull-and-push approach: the node that initiates a connection pulls the key, while we push the key to the receiving node. Finally, once both peers have each other's data, they can establish the WireGuard tunnel (3, dashed line).

Taking this into account, we defined the following messages for the control channel to interact with the database: (i) Store a peer public key + endpoint IP, (ii) Request a peer public key, and (iii) Send a public key to a peer.

5.4 Implementation

In order to implement the protocol for the control channel, we leveraged the control plane part of the Locator/ID Separation Protocol (LISP, RFC 6833 [133]). LISP is a mature and standardized protocol, designed to dynamically create network overlays. This makes it an excellent candidate for this application, because LISP's control messages present nearly a 1:1 match to our requirements (table 5.2). Moreover, the LISP architecture includes the Mapping System, a centralized server that

stores pairs of overlay to underlay IP addresses. However, it is possible to adapt other SDN southbound protocols such as P4runtime [7] or OpenFlow [6] for this use case, or even use HTTP-based interfaces, like REST (RFC 7231 [134]), WebSocket (RFC 6455 [135]) or gRPC [136]. However, we chose to use LISP due to performance concerns: since LISP runs on UDP, it offers lower latency when compared to TCP and HTTP-based protocols. We consider the control plane latency relevant for our use case since it is in the critical path to establish new connections (section 5.5.2 compares such delay with gRPC). On the other hand, we must remark that the aforementioned interfaces offer more extensibility than LISP.

Design message	**LISP message**
Store key	Map Register
Request key	Map Request
Send key	Map Notify

Table 5.2: Equivalence of section 5.3.3 messages and LISP messages

Our prototype is based on an open-source LISP implementation, Open Overlay Router (OOR, [137]). OOR is implemented in C and works in Linux user space, which makes it easy to implement new features [89]. In a nutshell, we modified OOR to:

1. Detect flows that do not have an active WireGuard tunnel configuration.
2. Request its associated public key with the aforementioned control plane messages.
3. Configure accordingly the WireGuard interface.

5.4.1 Endpoints

Figure 5.4 presents a diagram of our implementation. First, we configure a WireGuard interface to tunnel all control plane packets to the central server (i.e. server endpoint IP and pubic key). We use a statically defined IP prefix to identify and route such requests through this interface. We also create a TUN interface and add a default route in the kernel routing table pointing to it. The purpose of the TUN interface is allowing OOR to connect, capture packets and examine their destination IP address. If the IP address is not in its local database, OOR will send a LISP Map Request to the central server to retrieve the key. We must remark that the OOR database is a copy of the WireGuard Cryptokey Routing table. Once it receives this information, OOR configures the new peer in a second WireGuard interface that handles only data plane traffic. Note here that the prototype uses two WireGuard interfaces, one for data plane traffic, and another for control plane traffic.

Then, OOR adds a more specific route for this new prefix in the kernel routing table. This route points to the WireGuard Data interface. New traffic for this prefix will bypass the TUN interface and go directly to the WireGuard Data interface. This is especially relevant since it avoids copying packets from kernel to user space and viceversa, significantly improving performance.

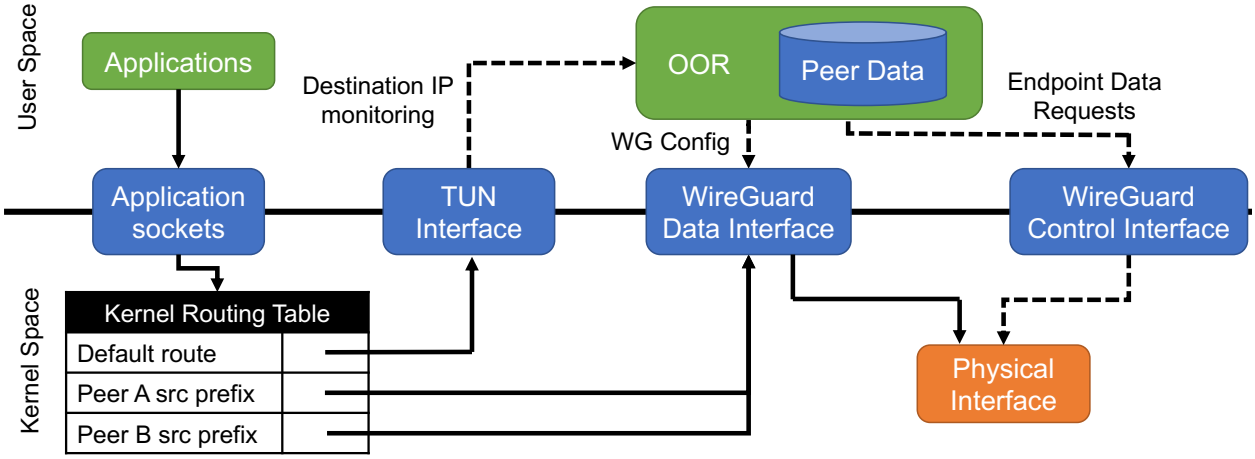


Figure 5.4: Simplified prototype operation diagram. Solid lines represent data plane traffic flow, dashed lines control plane traffic.

In case a peer does not want to tunnel all its traffic through the dynamically-created WireGuard tunnels (split tunneling), it can bypass them by adding more specific routes in the kernel routing table pointing to an alternative destination.

5.4.2 Central Server

We implemented the aforementioned centralized server as a LISP Mapping System, more specifically a single Map Server. In other words, all peers send their requests to the same Map Server. This Map Server is running a modified version of the OOR Map Server, in which incoming Map Requests trigger a Map Notify to the destination WireGuard peer. This way, the destination peer receives the sender key and the tunnel can be established. In order to carry the peer public key in the LISP messages, we leveraged the LISP Canonical Address Format, LCAF (RFC 8060 [138]), an extension of such protocol that allows adding extra data in the Map Request / Map Reply messages (LCAF 11 - security key).

Finally, in case a peer wants to update its public key, we can easily update this information with a LISP Map Register message.

5.5 Evaluation

We evaluated different metrics of our implementation in order to quantify the overhead of adding security (i.e. the WireGuard cryptography), as well as the delay of the pull architecture. Since OOR also implements the LISP data plane, i.e. it can create unencrypted network overlays, we took OOR's performance (not modified) as a baseline. We carried out all tests in a lab setup with 1 Gb Ethernet interfaces, and we considered the network delay negligible. The servers were running Linux Ubuntu 16.04 on an Intel Xeon E5-2650 v4 @ 2.20 GHz.

5.5.1 End-to-end Delay

Figure 5.5 presents the CDF of the delay to establish a tunnel, i.e the time since we send a probe packet until we receive the first response. The setup consisted of two instances of our implementation in the same LAN, directly connected with each other and also to the central server. We setup a data rate of 10000 pings/s to have an adequate time resolution. The measured time includes all the control plane signaling and the establishment of the WireGuard tunnels (for OOR+WG) or the LISP tunnels (for OOR). We repeated this experiment 100 times. We can see that WireGuard adds an extra delay of approximately 1 ms due to the cryptography. We consider this delay acceptable, taking into account that we are comparing an unencrypted connection (OOR), and an encrypted one (OOR+WG). We must remark that OOR packets go through user space, while in OOR+WG they stay in kernel space. The former adds an extra delay because data packets are copied from kernel to user space and vice-versa (figure 5.6 shows a diagram of how OOR processes data packets).

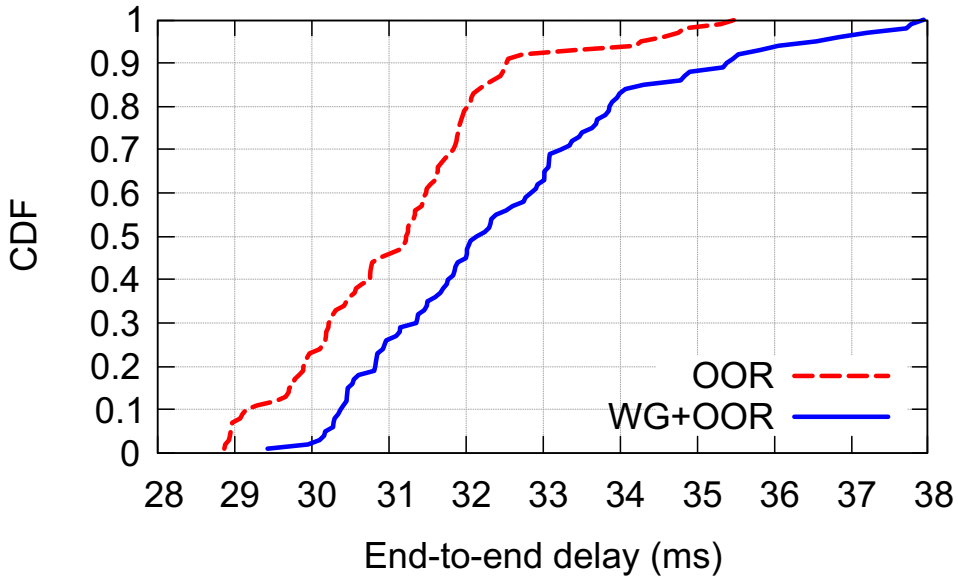


Figure 5.5: Tunnel Establishment end-to-end delay CDF

5.5.2 Control Channel Performance

We evaluated the performance of our control plane implementation and compared it to: (i) OOR with LISP data plane, and (ii) a prototype that uses gRPC instead of LISP to retrieve the public key. This prototype is based on a C version of gRPC [139] , and uses a WireGuard secure tunnel to communicate with the central server, like the OOR+WG prototype. Figure 5.7 presents the CDF of the end-to-end delay to add data for a new endpoint, or connection establishment time. We measured the time it takes to retrieve the public key for a new endpoint from the central server. We repeated the experiment 150 times for the three scenarios. We can appreciate that OOR with

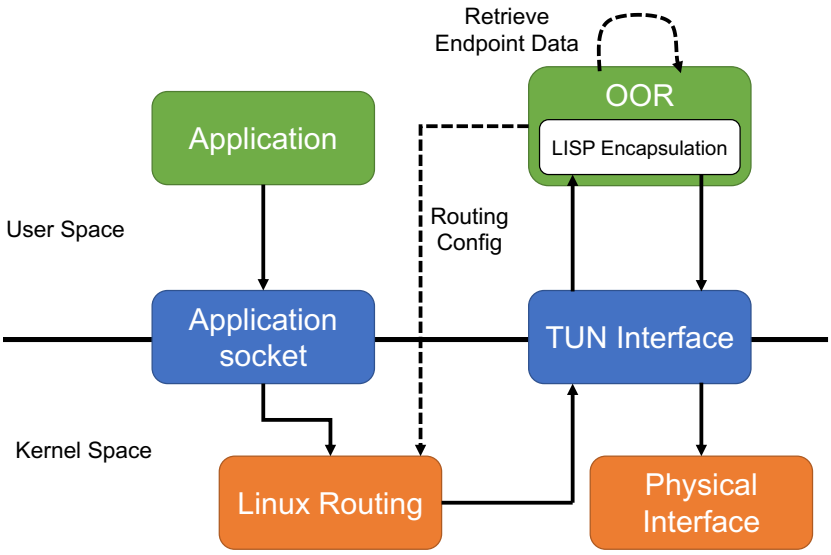


Figure 5.6: Out of the box OOR implementation diagram. Solid lines represent data plane traffic flow, dashed lines control plane traffic. As opposed to the OOR + WG implementation in figure 5.4, data plane traffic is copied between kernel and userspace, adding overhead.

WireGuard takes approximately 1.5 ms more than OOR to complete this operation. This is due to the fact that we are retrieving more information (WireGuard needs the public key, apart from he endpoint IP address) and the overhead of configuring the WireGuard interface (as opposed to OOR, in which we can start sending packets directly). Nevertheless, since the WireGuard public keys are 32 bytes, the performance penalty of downloading this extra information is negligible compared to typical Internet latency [140]. On the other hand, the gRPC prototype presents a delay double to OOR+WG, due to the overhead of its HTTP/2 transport.

5.5.3 Central Server Scalability

We also evaluated the scalability of the centralized server. Figure 5.8 presents the delay to answer a request for endpoint data for both OOR and OOR with WireGuard, depending on the number of elements in the central server. Specifically, we measured the processing delay of the central server, i.e. the time since it receives a data request until it generates the corresponding response. We repeated this experiment 50 times for each number of elements in the server. We can appreciate that the delay does not depend on the number of elements because the server implementation uses a Patricia Trie [141]. In other words, the maximum number of peers a single server can handle will be limited by its CPU, not the database size. Additionally, this experiment quantifies the overhead of adding a public key in the central server database. We can see that it increases the delay on average 40 μs. Again, this modest increase is due to the aforementioned extra 32 bytes.

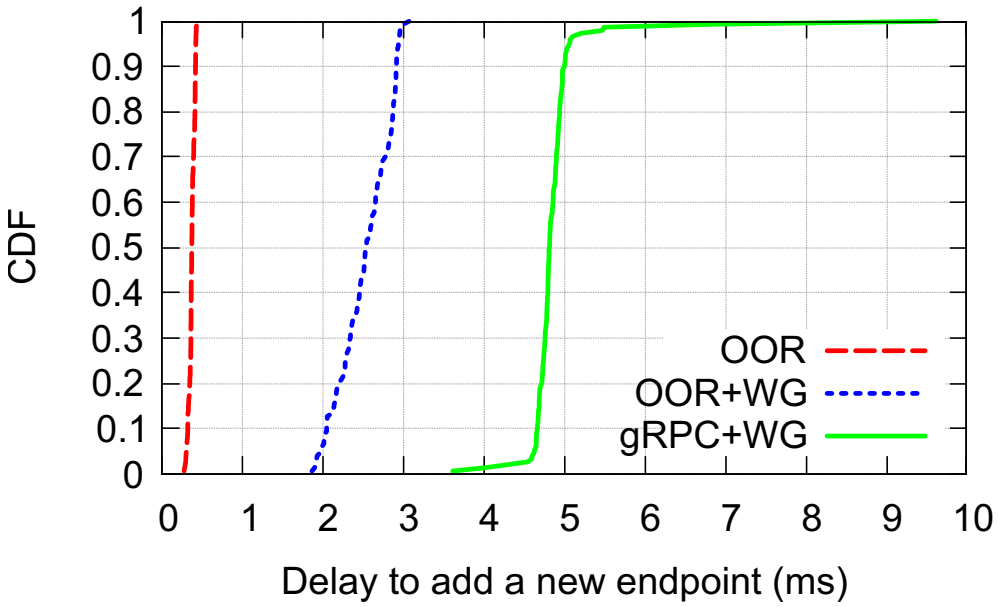


Figure 5.7: Delay to request and configure a new endpoint

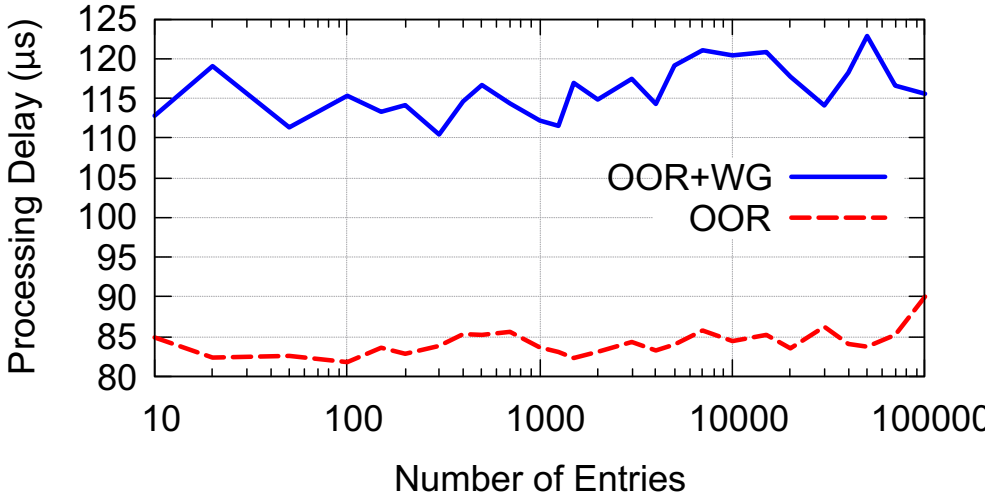


Figure 5.8: Central server response delay

5.5.4 Handover Delay

Regarding mobility events, figure 5.9 presents the handover delay for OOR and OOR+WireGuard. We modified the setup of section 5.5.1 by connecting one of the two peers via WiFi through two possible access points (figure 5.10), and we triggered handovers by manually changing the access point of this peer. We ran a ping of 10 requests/ms between the two machines and measured the amount time without ICMP replies. We repeated this experiment 20 times. This test shows that WireGuard presents a handover delay nearly one order of magnitude less than OOR. This is due to the fact that the WireGuard handover: (i) operates in kernel space, and (ii) it does not require

control plane signaling, as opposed to OOR, that issues a new Map Request / Map Register in this situation.

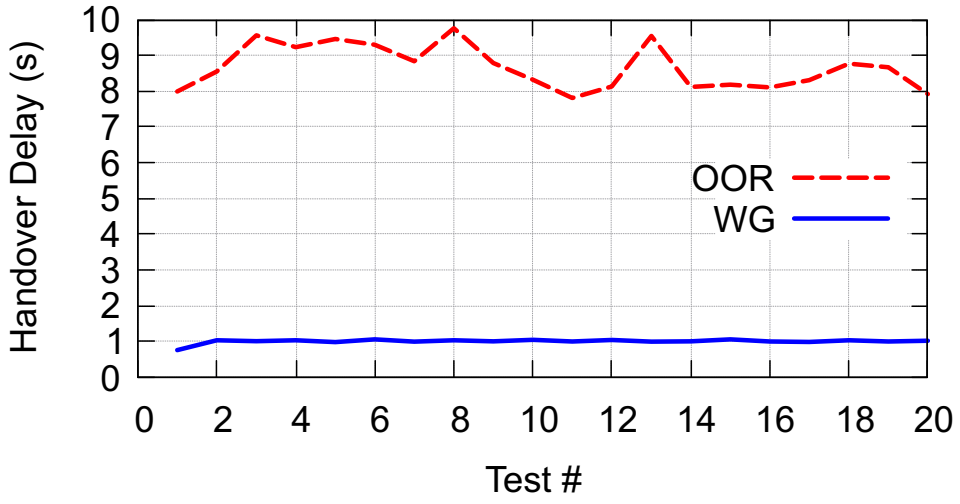


Figure 5.9: Handover delay

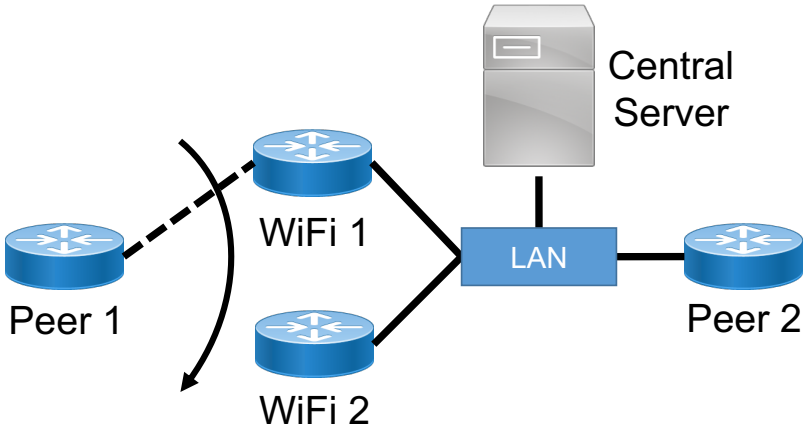


Figure 5.10: Handover evaluation setup

From an architecture point of view, we observe that adding a security association simplifies mobility, because we can learn new endpoints through the data plane with virtually no signaling. In addition, if the endpoint IP address is spoofed, the payload is protected by the WG crypto header. On the other hand, this technique does not cover corner cases such as double jump. This is in line with the design philosophy of WireGuard, that sacrifices flexibility for simplicity.

5.5.5 Data Plane Performance

Finally, we measured the data plane throughput. Figure 5.11 presents the input and output rate for our OOR+WireGuard implementation, IPsec, and out-of-the box OOR. In the same setup as in section 5.5.1, we configured a tunnel between the two servers, and we used the `nuttcp` utility to

determine real throughput between them for several input rates, and for each protocol. We sent UDP packets, and adjusted the lengths of the send/receive buffers taking into account the size of the different headers of each protocol, in order to fill in the packets but avoid L3 fragmentation. We configured IPsec in tunnel mode with the same encryption algorithm as WireGuard (ChaCha20 and Poly1305). We can see that both OOR+WG and IPsec offer similar performance, and start to degrade gracefully around 900 Mbps. On the other hand, the OOR version that uses the LISP data plane degrades abruptly at 800 Mbps due to the overhead of copying form kernel to user space.

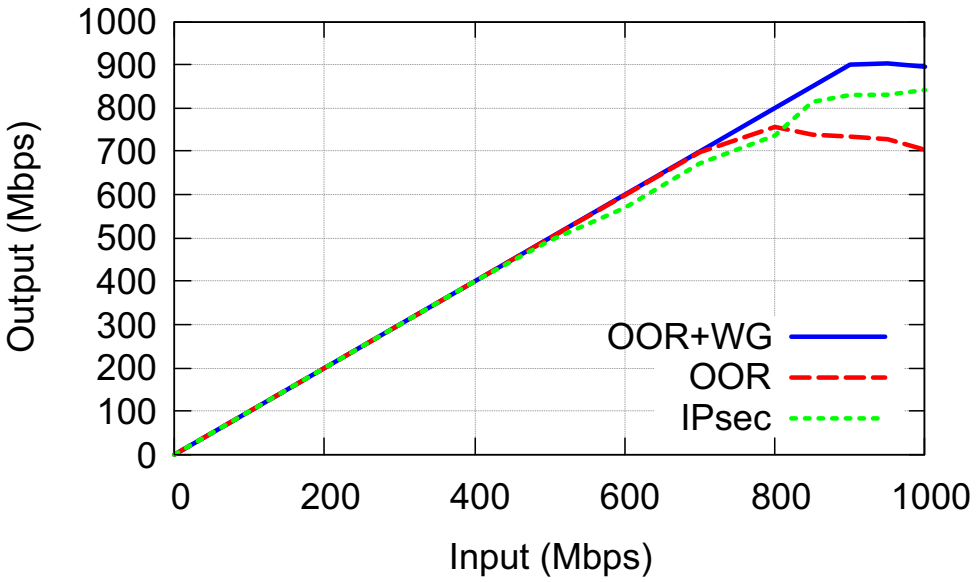


Figure 5.11: Throughput comparison

5.6 Lessons Learned

During the implementation and evaluation of our prototype, we found a few shortcomings in WireGuard. Some are especially relevant for enterprise (Virtual Networking) and ISP (multi-homing) scenarios. In this section we comment on them and outline possible approaches. Some of them can be addressed with our proposed control plane, others require modifying the WireGuard data plane.

5.6.1 Virtual Networking

It is common to support several instances of the same addressing space, especially in enterprise environments. The most common solution is adding an extra field in the headers that acts as identifier, e.g. VXLAN Network Identifier [22]. Since WireGuard packets do not include a Virtual Network Identifier, it is not possible to make this distinction. However, it can be accomplished by adding this field to the WireGuard header, or adding a VXLAN header before sending packets to the WireGuard interface, for instance. We believe this is a relevant feature that could be considered among the WireGuard community.

5.6.2 Multi-homing

Our implementation cannot send encrypted traffic from multiple Internet endpoints at the same time (i.e. multi-homing). The main reason is that WireGuard only supports a single endpoint IP at any moment in time. If the endpoint IP changes it is interpreted as a mobility event, not traffic coming from different sources. A possible solution consists in leveraging the control plane channel to advertise the set of possible Internet endpoints to both peers. However, this means that the WireGuard data plane cannot update the Internet endpoint of a peer when it receives a packet with a different IP, thus making mobility more complex. This feature is especially interesting in the context of increased mobile device usage, e.g. WiFi and LTE interfaces in a smartphone, or ISPs that offer VPN solutions.

5.6.3 Mobility Double Jump

In case two peers change their Internet endpoint at the same time, it is not possible to re-establish connection only with data plane mechanisms. However, thanks to our control plane we can account for this situation. When a peer detects it has not received packets after some time, it sends a new key request to the control plane. Since the other peer will have updated its new endpoint in the control plane, the former peer will receive the new endpoint IP and communication will resume.

5.7 Related Work

5.7.1 Key Distribution

The problem of storing public key data in a centralized server, and distributing it to establish secure tunnels is a well-know topic and has been previously explored, such as in Kerberos [142], or the IETF DNS-Based Authentication of Named Entities (DANE [129, 130]), that makes it possible to add different types of public keys in DNS records like IPsec keys (RFC 4322 [143]). Furthermore, there is extensive research in dynamic VPN configuration and management [144]. More recently, SDN-inspired solutions propose to configure IPsec endpoints in a centralized fashion [145], as well as the widespread usage of Software-Defined WANs that automatically setup IPsec tunnels [132]. The most closely related work is a theoretical analysis to configure WireGuard with OpenFlow extensions [146]. However, it does not provide an implementation or performance data, rather focusing on the design of control plane messages.

5.7.2 Secure Data Planes

Table 5.3 compares several features of WireGuard and two of the most popular L3 VPN protocols: IPsec [147] and OpenVPN [148], along with the overhead of their headers. We can see that WireGuard supports both NAT traversal and mobility, with the same overhead of OpenVPN with DTLS. OpenVPN does not support mobility but can deal with NAT with additional configuration [149]. Finally, IPsec does not offer NAT traversal nor mobility but incurs in a smaller overhead.

However, we must note that IPsec can handle both NAT traversal and mobility with additional extensions: IPsec over UDP, and the MOBIKE extension (RFC 4555 [150]), respectively.

Protocol	**NAT-trav.**	**Mobility**	**Overhead (Bytes)**
IPsec tunnel ESP	×	×	28
OpenVPN DTLS	√	×	38
WireGuard	√	√	38

Table 5.3: Comparison of common data plane encryption protocols

Regarding other data plane encapsulations, such as VXLAN (RFC 7348 [22]), VXLAN-GPE [151], GENEVE [152] or ILA [153], it is interesting to remark that they do not usually consider security or mobility (with the notable exception of ILA mobility), as opposed to the protocols in table 5.3.

5.8 Summary of Outcomes

In this chapter we have presented a control plane for nodes using the WireGuard VPN protocol. Such control plane stores the WireGuard public keys in a centralized server and distributes them on-demand. The proposed architecture offers two main advantages: (i) automates the configuration of WireGuard tunnels, and (ii) reduces state in the data plane by creating security associations only if a particular tunnel is required. This control plane can be used to automate deployment, especially in enterprise or ISP scenarios with a large amount of peers. We also took advantage of WireGuard to secure the control plane channel between the nodes and the centralized server. In addition, we have discussed the main challenges, such as pushing the sender's public key to the destination, and possible additions like virtual networking or multi-homing, in order to support the aforementioned use cases. Finally, we have presented the design of our implementation, an evaluation of the overhead of such control plane, and performance measurements to validate that we retain the throughput and mobility benefits of WireGuard.

Chapter 6

A Design for Current Enterprise Networks

6.1 Introduction

In this chapter we present a design for current Enterprise Networks (EN). Following an analysis of the challenges that EN face today, we designed a solution based on overlay networks that leverages state of the art architectures and protocols. Current Enterprise Networks (EN) present a high degree of complexity derived from their organic evolution. Traditional ENs are built in a three-tier structure [154, 155]: access, distribution and core, with each layer leveraging a distinct set of protocols (figure 6.1). The access-distribution segment uses L2 technologies such as Virtual Local Area Networks (VLANs), Spanning Tree Protocol (STP), and VLAN Access Control Lists (ACLs). On the other hand, the distribution-core segment is usually L3 with a combination of Open Shortest Path First (OSPF), Multiprotocol Label Switching with Label Distribution Protocol (MPLS-LDP), and IP ACLs. This design varies from deployment to deployment, but the end result are complex networks that can be running up to tens of different protocols across hundreds of switches. This makes it hard to adapt to new requirements, such as isolating IoT devices or stretching L2 domains across distributed locations. Even if some of these challenges might have been addressed by technologies designed for service providers, such as Virtual Routing and Forwarding (VRF), given the high port density required in typical campus networks, adopting the same technologies in enterprise solutions is not cost effective.

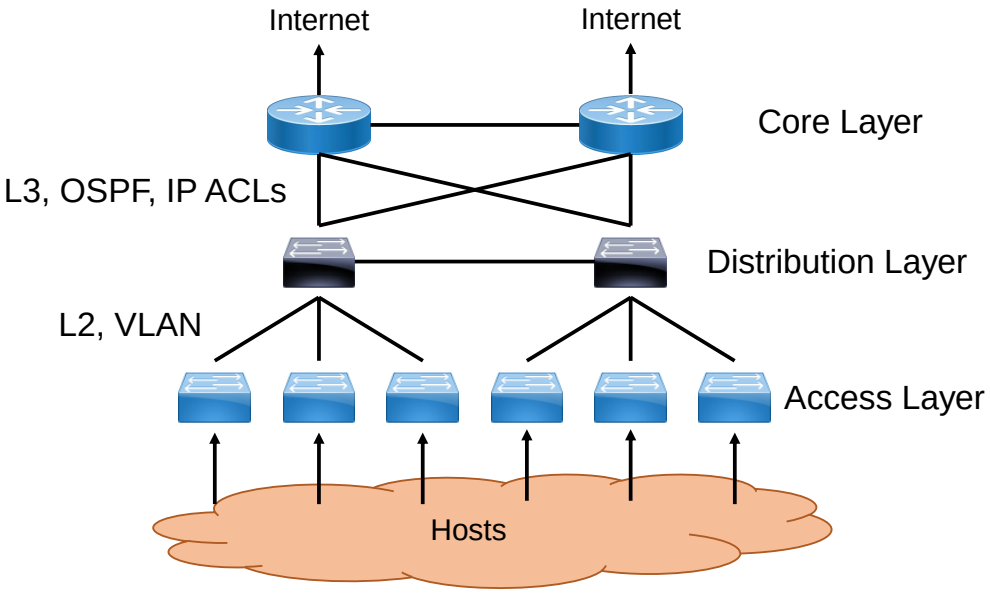


Figure 6.1: Classical design for enterprise networks.

Specifically, current ENs lack three key elements. First, scalable endpoint mobility across all the enterprise facilities, to address the ever-increasing amount of roaming devices. Usually this is handled via sending all wireless traffic through centralized Wireless LAN (WLAN) controllers, which limits scalability, and reduces bandwidth. Second, simple to operate segmentation. The most common forms of segmentation in ENs are VLANs or VRFs, which do not scale well and can be difficult to configure at scale. Another example are IP-based ACLs, that over time can easily

become long and difficult to map to the original intent. Third, simplified operations. Network administrators might configure each router individually, and, as we mentioned previously, have to deal with a myriad of different protocols. Although there exist more modern solutions [156, 157] that satisfy some of these requirements, we believe that the state of the art does not deal with scale and dynamism in a cost-effective manner, i.e. without requiring large capital expenditures (CAPEX).

CAPEX plays, indeed, a very important role in the context of Enterprise Networks. First, because of brownfield deployments: typically, network administrators do not want to upgrade *all* of their switches for new features. Since these are usually legacy devices with limited features, new network designs require a way to add new functionality without a forklift upgrade. Second, because deploying devices with reduced FIB size or CPU power decreases CAPEX but in turn means less powerful devices that require resource optimization.

This chapter presents the rationale, implementation, evaluation and experience matured in deploying SDA (Software Defined Access). The objective of SDA is to address the aforementioned requirements of modern EN. SDA leverages a vast spectrum of research ideas, architectures and protocols produced by the community in the last decade [8, 158, 10, 159, 109, 160], and integrates them in a practical and deployable solution. First, we leveraged network overlays as discussed in the Fabric architecture [10], in the form of the Locator/ID Separation Protocol (LISP [133]). This offers three benefits:

1. We can upgrade existing deployments (brownfields) with minimal touch.
2. Their layer of indirection makes it easy to support L3 mobility.
3. They make segmentation with VRFs more scalable.

Second, we applied the Software-Defined Networking (SDN) principle of centralized control to track endpoint location, and map endpoints to segmentation policies across the whole network. Finally, we chose a reactive protocol (LISP) to distribute network state to the data plane. In other words, we populate the switches forwarding tables only if required by the active traffic pattern. This reduces the overall switch requirements in terms of FIB size and CPU power which results in reduced CAPEX.

This chapter has four main blocks. First, we elaborate on the limitations of current solutions, and explain our approach. Then, we describe how we combine state of the art techniques to realize a practical solution, along with its implementation. Third, we detail our experience through two real-life deployments. First, a medium-sized enterprise campus network serving around 450 endpoints that includes fixed hosts, mobile hosts, application servers, IoT devices, etc. We show that in this scenario, our reactive protocol optimizes data plane state with a 70% reduction of FIB entries in the data plane compared to solutions that store all the state in all routers (e.g. BGP). Second, a large warehouse where hundreds of robots are moving at speed to fulfill shipping orders, such as those ran by large on-line retailers. Here we evaluate the handover delay of 16,000 robots triggering 800 mobility events per second. Our solution achieves 5 times lower handover delay compared to

existing approaches. Finally, we present our lessons learned from deploying SDA, such as reducing the initial connection delay due to the reactive protocol, coping with connectivity issues in the underlay or dealing with policy updates at scale.

6.2 Requirements and Design Decisions

This section delves deeper into the requirements of current ENs, explains limitations in the state of the art and discusses design decisions, summarized in table 6.1.

Resource optimization

Routing protocols commonly found in enterprises, such as OSPF, IS-IS or BGP make it difficult to reduce the FIB space without losing granularity. It is usual to leverage BGP prefix aggregation or OSPF Areas to reduce the overall number of routes in the network, at the price of less granularity. We approached this challenge by leveraging a reactive protocol (LISP in our implementation), rather than a proactive. Instead of pushing all routing entries beforehand to the routers, we only retrieve the necessary forwarding entries from a centralized server on-demand, and only for the routers that need them [161, 160]. We track endpoints by their IP address, so that routers download routes for the remote endpoints they need to reach, based on incoming traffic from local endpoints.

In addition, this reactive approach is helpful with mobility because we can reduce convergence time. This arises from the fact that a reactive approach reduces the churn generated by the location updates: we only notify the parties affected by a specific mobility event. We must remark that a reactive protocol presents several challenges, such as a potential initial delay for the establishment of flows (section 6.3.2), or detecting connectivity outages in the underlay (section 6.5 discusses our learnings in this space).

Mobility

The current trend of wireless first makes it critical to support a large amount of wireless endpoints. Traditionally, ENs handle mobility at L2 in a centralized way for both data plane and control plane. A gateway device (WLAN controller) acts as a sink for all traffic from all access points, performs access control, and re-injects it to the L3 network. This approach presents a serious scalability limitation because the gateway device becomes a bottleneck[1]. In addition, it creates triangular routing because all L3 traffic is forced to go to the gateway and then back to the actual destination.

SDA tackles mobility at layer 3 using network overlays [162, 163], specifically with the mobility features of LISP. We keep the wireless control plane centralized for authentication purposes, but we let packets coming from the access point to be directly routed to destination. This distributed data plane greatly increases scalability.

[1] Although this particular concern can be alleviated with hierarchical controllers, it comes at the price of increased complexity and number of devices.

Requirement	**Current Approach**	**Limitations**	**Our approach**	**Benefits**
Resource efficiency	BGP and OSPF prefix aggregation	Granularity	Traffic-driven route learning	Reduced CAPEX, device-level granularity
Mobility	L2 centralized control and data plane	Scalability, triangular routing	L3 centralized control, distributed data plane	Increased scalability, optimized routing
Segmentation	VLANs and VRFs	Scalability, network-wide policies	Limited L2 stretching, 'centralized' VRFs	Increased scale with less resources
Simplified administration	VLAN and IP-based ACLs	Error-prone, no mobility	Group-based policies	Smaller ACLs, end-to-end enforcement

Table 6.1: Summary of current state of the art, challenges, and design decisions

Segmentation

Traditionally, the most common form of segmentation in enterprise networks are Virtual Local Area Networks (VLANs [164]) and Virtual Routing and Forwarding tables (VRFs [165]). Despite their simplicity, the scope of VLANs must be kept limited to prevent flooding of broadcast traffic or L2 forwarding loops, hence, they do not scale well. Regarding VRFs, they scale better than VLANs, but since each device has to be individually configured it is hard to implement global polices across the whole network. A direct consequence is that administration becomes too cumbersome as the number of VRFs increases. In addition, both of these approaches present similar limitations that make it hard to deal with mobility at scale.

SDA addresses these issues at different levels: for L2 segmentation [8, 166], we carefully stretch L2 domains (c.f. section 6.3.5). For L3, we still use VRFs, but map local VRFs to global virtual networks in order to handle L3 segmentation at scale. This way, network administrators only have to specify the virtual network for each endpoint [159]. Finally, we add a layer of indirection to ease administration, detailed in the next paragraph.

Simplified administration

A direct consequence of using network primitives such as IP addresses or VLANs for segmentation and access control is operational complexity [167]. In other words, network administrators have to translate business intent into IP addresses and ACLs, and backwards. In the long run, this approach does not scale, is error prone, and increases complexity.

To overcome this problem, we make use of the well-established group-based paradigm [109, 168] to define ACLs between groups, instead of IP prefixes. First, the network operator defines a connectivity matrix among all groups. Then it adds endpoints to each group. On the network level, routers track each endpoint by its IP address and add a 16-bit tag representing its group, so they can enforce the connectivity rules in the matrix. The benefit is that network administration is radically simplified and common operations, such as IP address planning or ACL configurations can be automated.

6.3 Design and Implementation

In this section, we describe the design and implementation details of SDA. First we provide an overview, then we describe the control plane and data plane. Finally, we detail how we support endpoint mobility and L2 services.

6.3.1 Overview

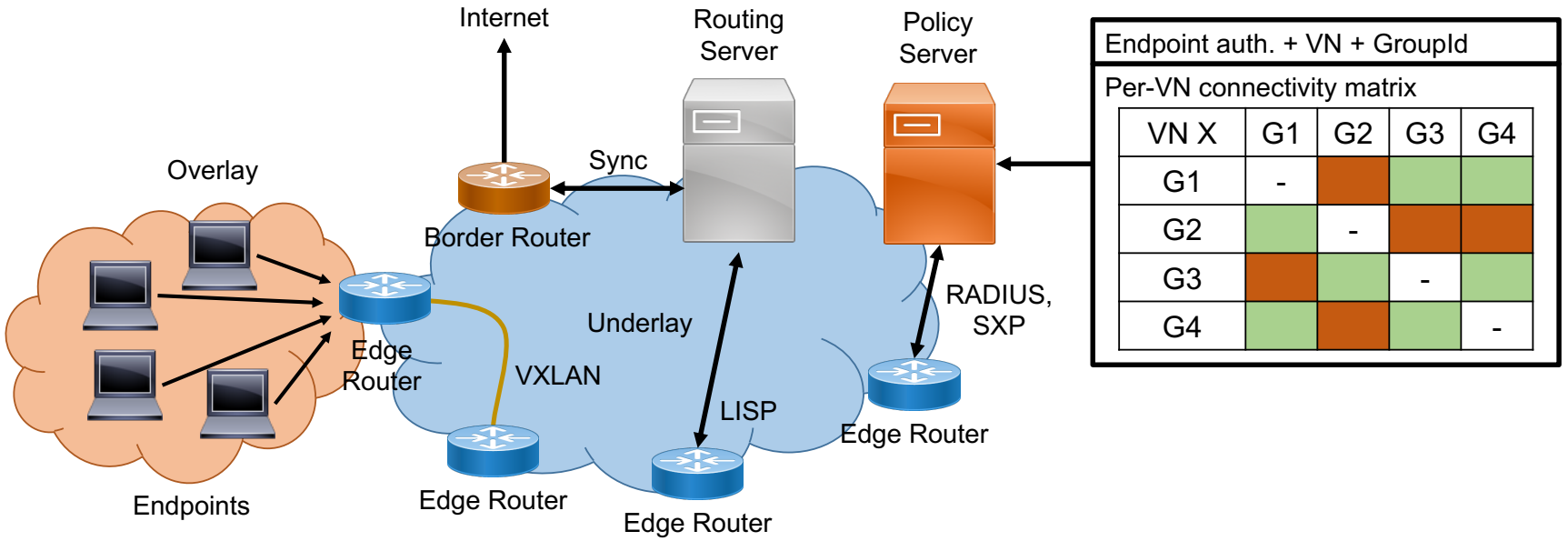


Figure 6.2: Global design

On a conceptual level our design presents the usual data/control plane layer separation typical of SDN [6]. Figure 6.2 presents an overview of the design. We expose a simple declarative interface for network operators to define:

1. The group and Virtual Network (VN) of each endpoint
2. The endpoint authentication data
3. The connectivity matrix across groups of endpoints

We store this information in the control plane in two different servers, a routing server and a policy server. The policy server authenticates endpoints, assigns them a group and configures the data plane routers with the required group rules from the connectivity matrix. The routing server keeps track of all endpoints by their IP address and provides routes upon demand by the data plane.

On the data plane, the overlay routers -hereafter referred to as *edge* routers- enforce the connectivity matrix and route packets to the corresponding edge router. A special *border* router provides access to external networks. Finally, endpoints can roam freely across edge routers.

6.3.2 Control Plane

We based the control plane design on a database-focused approach similar to [158], as opposed to more traditional designs [169–171]. The control plane consists of two logically centralized servers: policy server and routing server. The reason for this separation is that usually the host onboarding process needs both the endpoint credentials and group permissions, while the normal packet flow only needs the location mappings. Note that some deployments may have more than one routing server or policy server for redundancy and load balancing. Table 6.2 presents a summary of all our control plane data.

Name	**Key**	**Value**	**Updated by**
Endpoint Data	Endpoint authentication info	GroupID, VN	Network Operator
Group Rules	Source GroupId + Dest. GroupId	Action (allow, deny)	Network Operator
Endpoint Location	VN + Overlay IP address	Underlay IP address (i.e. edge router)	Edge Routers

Table 6.2: Types of Control Plane Data

Control Plane Policy Sever

We offer two degrees of segmentation: Virtual Networks (VN), and the group connectivity matrix. These group rules are independent for each VN. On one hand, VNs offer strong isolation at a

'macro' level. An example is a hospital network, where we want to isolate the network for doctors, guests and medical devices. We never expect them to be able to communicate with each other. This is especially relevant to isolate legacy devices susceptible to attacks, e.g. an MRI machine running an outdated OS. In addition, it is a way to mitigate lateral spread attacks.

On the other hand, the group rules offer a 'micro' segmentation for finer grain control inside a VN. For example, this level of segmentation can separate different types of devices within a VN in Bring Your Own Device scenarios.

The policy server stores the connectivity rules from the connectivity matrix, and, for each endpoint: its authentication data, and associated GroupId and VN. GroupId and VN are 16-bit and 24-bit identifiers, respectively. The authentication data is variable since we support different RADIUS-based authentication protocols [172], both with EAP or without. We use a specific protocol, Scalable-Group Tag eXchange Protocol (SXP [173]) to distribute the GroupIds and connectivity rules to edge routers. From the network perspective, VNs are mapped to isolated routing-switching domains, while GroupIds are inputs to group-based ACLs.

Control Plane Routing Server

The routing server stores the endpoint location, i.e. pairs of overlay-to-underlay IP addresses plus its associated VN. The overlay IP is the IP used by endpoints, while the underlay IP is the IP of the edge router serving this endpoint. The other edge routers encapsulate traffic for such endpoint towards the underlay IP. After a successful device onboarding (section 6.3.3), or upon detecting a mobility event, edge routers update the underlay location of an overlay IP address. Edge routers also retrieve this mapping when they receive a connection request to a particular device. The routing server is implemented leveraging the control plane aspects of the Locator/ID Separation Protocol (LISP, [133]).

In a nutshell, the LISP control plane offers two messages: Map Request, to retrieve the underlay address of an overlay endpoint, and Map Register, to update the location of an endpoint, i.e. the overlay to underlay mapping. This way, we can store in the data plane the overlay to underlay mappings that are required by the edge router to serve incoming traffic. In addition, the LISP control plane supports mobility, is well suited for SDN architectures [9], and accommodates different overlay address families apart from IP, e.g MAC addresses. This is especially helpful to support L2 services (section 6.3.5).

Border Router

A drawback of using a reactive protocol such as LISP is the initial packet loss until the edge router downloads the route for a new destination. We have overcome this issue by installing a default route in all edge routers that points to the border router, and by synchronizing the routing state in the border with the information in the routing server (sync arrow in figure 6.2). This way, edge routers forward packets to the border until they finish the resolution process, during this time the border router forwards such packets to the destination (dashed line in figure 6.3). The border

router is usually more powerful than edge routers in order to handle this extra load.

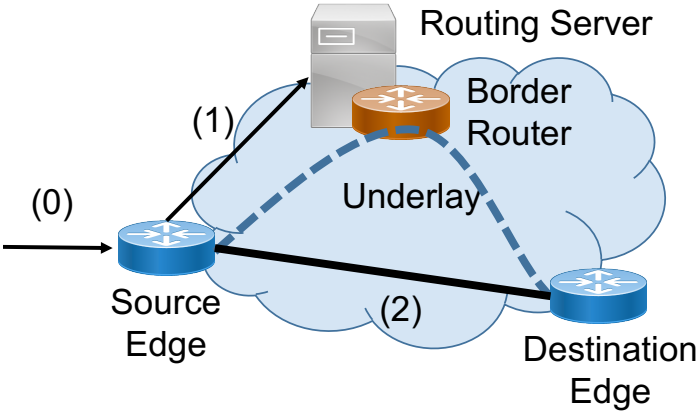


Figure 6.3: Initial packet loss reduction with default entry in edge routers.

6.3.3 Data Plane

The data plane presents two distinct routers (figure 6.2): edge and border.

Edge Routers

Edge routers perform four key functions:

1. Encapsulate and decapsulate traffic from and to endpoints, respectively.
2. Provide inter-VN isolation ('macro' segmentation). We implement such segmentation with VRFs: LISP populates the VRF tables, and each entry has an associated GroupId.
3. Detect roaming endpoints and update their location in the routing server.
4. Enforce group permissions from the connectivity matrix ('micro' segmentation).

Border Routers

Border routers perform the same functions as edge routers, with two exceptions. First, their FIB table is synchronized with the routing server. In other words, they don't use a reactive protocol, rather they are subscribed to all route updates from the routing server [174]. And second, they have routes to other networks, e.g. Internet, datacenter. Because of this, border routers have more powerful CPU and larger FIB tables.

Encapsulation and Underlay

The data plane encapsulation leverages VXLAN with Group Identifier extension [175]. We chose this encapsulation over the native LISP data plane because of the need to encapsulate both L2 and

L3 payloads (LISP supports L3 only). Additionally, in this VXLAN variant we can add the source GroupId in the group field (figure 6.4). Finally, the underlay is a network with plain IP connectivity that routes encapsulated packets between edge routers. The underlay routing is provided by either OSPF or IS-IS, we leverage MACsec [176] for packet integrity protection and confidentiality, and ECMP for redundancy [177].

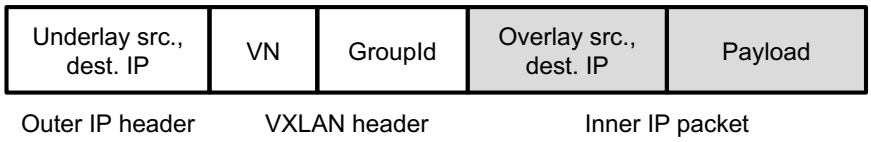


Figure 6.4: Packet Format

Host Onboarding

When an endpoint connects to the overlay for the first time, the edge router detects it and starts the authentication process with the policy server (step 1 in figure 6.5). After a successful authentication, the edge router downloads the endpoint's VN, GroupId, and associated connectivity rules from the policy server (step 2). Specifically, it downloads the rules where the endpoint's group is the destination (c.f. section 6.5.3), stores locally the GroupId value, and associates it to the switch port where the endpoint is connected. Then it can assign an overlay IP address to the endpoint (step 3), obtained from a DHCP server. Finally, the edge router stores the location of the endpoint in the control plane, i.e. update the (VN + overlay IP, underlay IP) pair in the database (step 4).

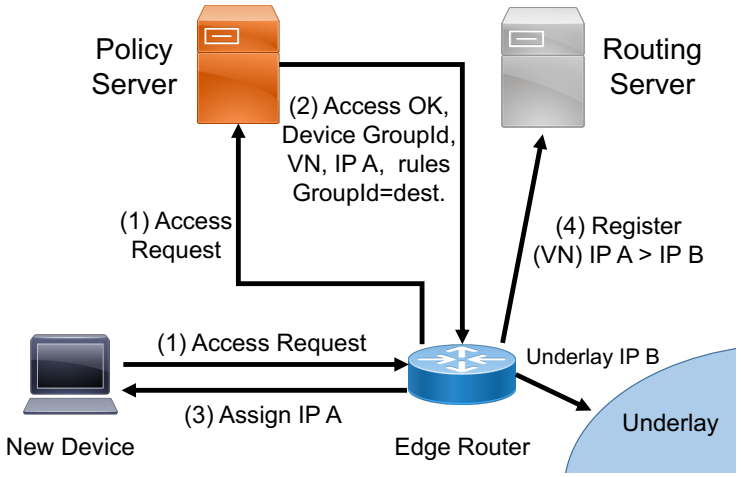


Figure 6.5: Host Onboarding Process

Common Packet Flow

On ingress, packets from endpoints are assigned their corresponding GroupId and VN (figure 6.6, ingress edge router). The router knows these values from the onboarding process. Then, it does

a VN + overlay destination IP lookup in the VRF table for that VN. If there is no match, it will query the control plane database. This query returns the underlay IP address of the destination overlay IP. Finally, the packet is encapsulated towards the corresponding edge router, carrying both VN and GroupId.

On egress, the destination edge router decapsulates the packet and injects it into a two-stage pipeline (figure 6.6, egress edge router). First, it performs a VN + overlay destination IP lookup in the local VRF table corresponding to the VN in the packet. This query returns the output port and the associated destination GroupId. Each entry in the VRF has its associated GroupId, that is stored during the onboarding process. After authentication, the edge router creates an `(Overlay IP, GroupId tag)` association in the VRF table.

The second stage is an exact match lookup in an group-based ACL of (source GroupId, destination GroupId). This ACL enforces the aforementioned group rules. Finally, the router forwards the packet to the destination overlay IP address. We perform the policy enforcement on egress due to increased scalability (section 6.5.3).

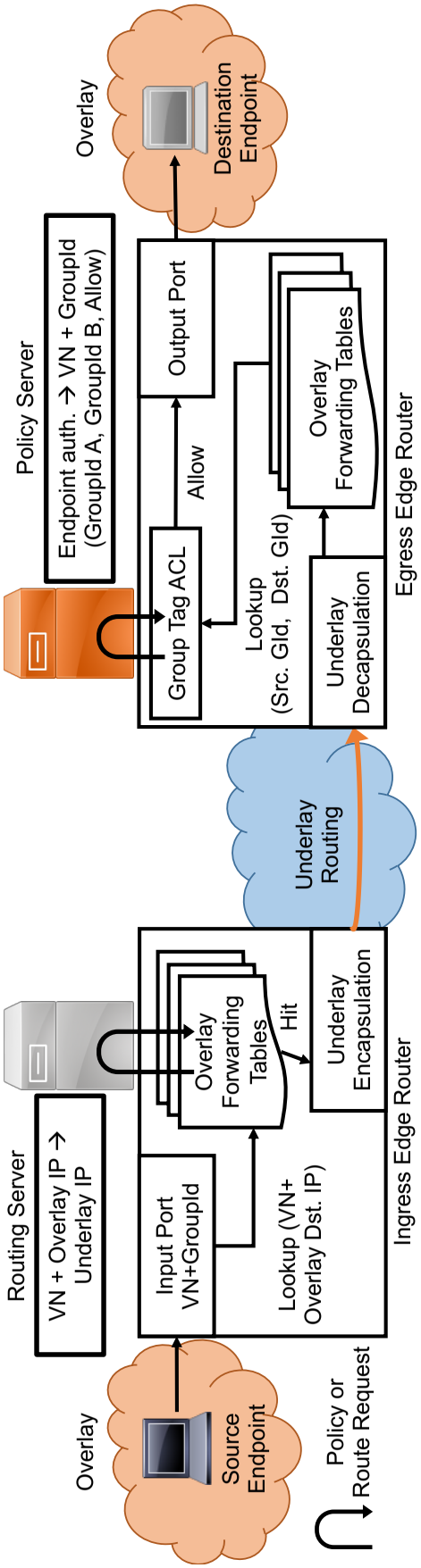


Figure 6.6: Ingress and egress pipelines

6.3.4 Mobility Support

When an endpoint roams and attaches to a new edge router, the latter triggers the authentication process again, and registers the new location (figure 6.7, step 1). After this registration, the control plane sends a message to the previous edge router instructing it to: (i) pull the new location data (2,3), and (ii) forward all traffic for that particular endpoint to the new edge router. Hence, handover signaling is linear with the number of roaming endpoints, as opposed to proactive protocols, in which it also depends on the number of routers (section 6.4.3).

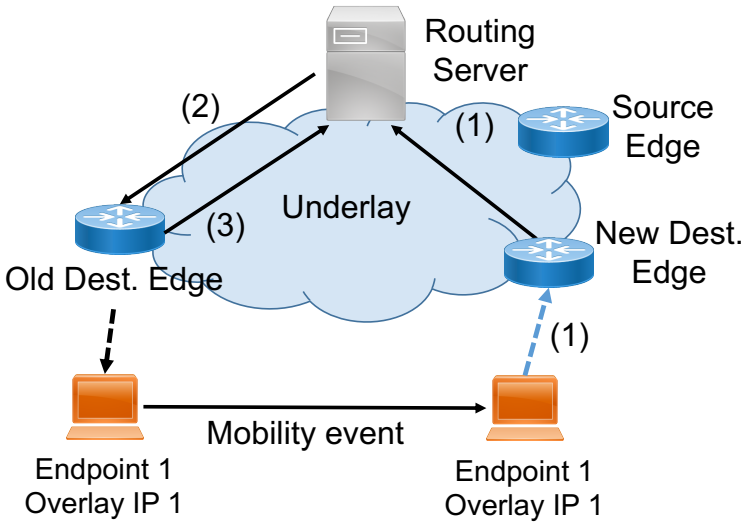


Figure 6.7: Endpoint mobility

Additionally, we apply the event-driven approach to update edge routers storing stale entries. To this purpose, we use a specific data-triggered control plane message (figure 6.8): when the old edge router receives traffic for the roaming endpoint (1) it sends a control message to the source (2), instructing it to retrieve the new location (4). At the same time, the old destination router forwards this traffic to the new destination router (3).

Regarding signaling scalability, this method depends on traffic patterns: if the roaming endpoint is very popular, we will have to update a significant portion of edge routers. On the contrary, endpoints that receive traffic from few sources, require less signaling. The advantage of this technique is that it is triggered by traffic, in other words, the control plane doesn't need to update *all* edge routers that have the stale location, but only those that *require* it. Finally, these control plane messages will be staggered over time as traffic from different senders will arrive at different times, thus spreading the signaling load in time.

6.3.5 Support for L2 services

In enterprise scenarios, it is common that some devices or users require L2 connectivity. Common use cases are some forms of load-balancing, certain IoT devices, and basic services such as DHCP

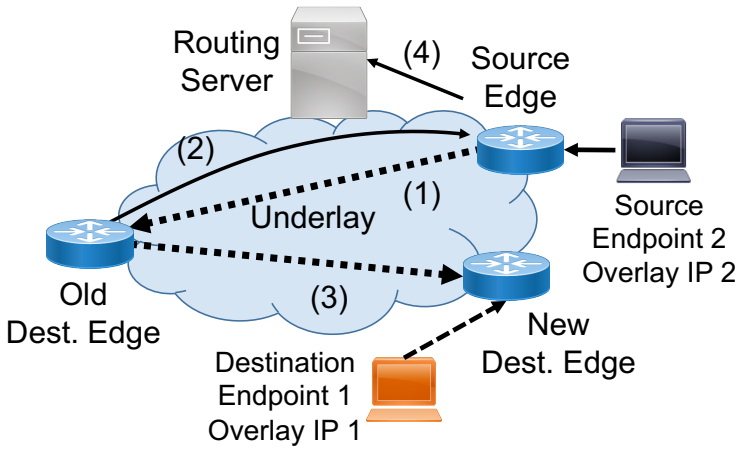


Figure 6.8: Updating stale entries via data plane messages.

or service discovery (a significant amount of applications rely on broadcast domains, e.g. Apple Bonjour).

In order to support such services, but avoid sending broadcast traffic to the entire network, our implementation leverages four elements:

1. VLANs, but limited to the edge router ports.
2. Indexing endpoints by MAC address in the routing server, in addition to IP address.
3. Storing overlay IP to MAC pairs in the routing server.
4. Deployment of L2 gateways in edge routers.

The combination of these four elements helps us to provide scalable L2 connectivity: first, VLANs limit broadcast domains; second, MAC address indexing locates endpoints of the same VLAN that are in different edge routers. Finally, L2 gateways absorb broadcast traffic and convert it to unicast: for instance, they capture ARP requests and perform a lookup in the routing server to find the MAC associated to the IP in the ARP request. Then they use this MAC to replace the broadcast MAC in the ARP request, creating a unicast L2 message. This message is injected in the L2 pipeline, which will use the MAC-to-underlay IP to encapsulate the request to the intended L2 MAC.

6.4 Evaluation

SDA is implemented in a commercially available line of routers, leveraging the protocols mentioned previously: LISP [133], RADIUS [172], and Scalable-Group Tag eXchange Protocol [173] for the control plane, and VXLAN [22] for the data plane. We have deployed our implementation in two different real-life scenarios: two campus networks with 150 and 450 endpoints, and a large

warehouse (partially emulated) with massive mobility serving 16,000 emulated endpoints. Table 6.3 presents a summary of their characteristics.

Deployment	**# Border Routers**	**# Edge Routers**	**Number of endpoints**
Building A	1	7	150
Building B	2	6	450
Warehouse	2	200	16,000 (emulated)

Table 6.3: Deployments used for evaluation

We evaluated three key elements of SDA: first, the response of the routing server under stressful conditions, because it is critical in the process of establishing new data flows. Second, quantifying the state optimization in the data plane due to the edge-border design. And third, assess the difference between a proactive and a reactive protocol in face of massive mobility events.

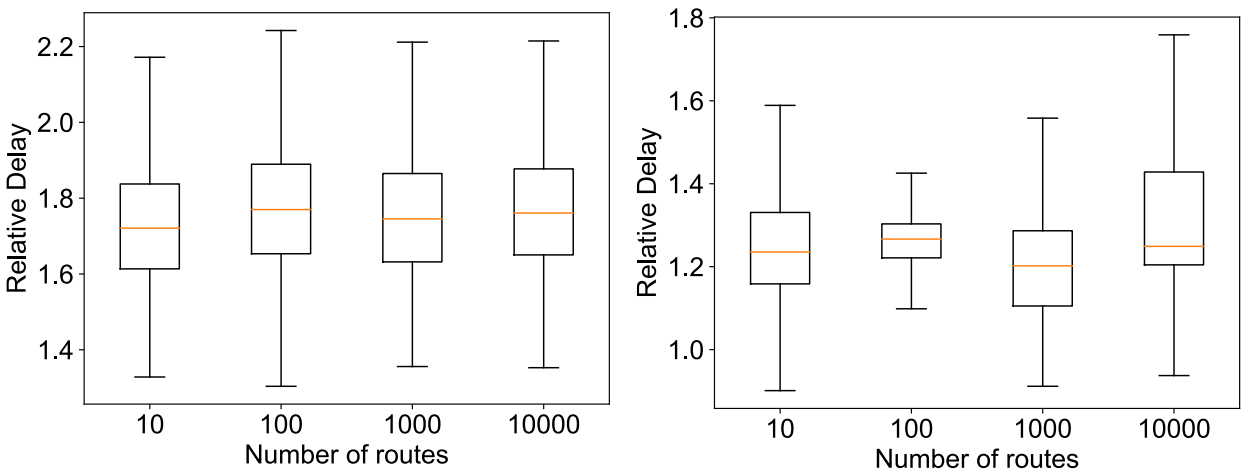
6.4.1 Routing Server Scalability

The routing server is a critical part of the design, because it allows establishing communication between any pair of endpoints. Because of this, we evaluated its performance depending on the number of routes and queries per second.

To this end, we setup a routing server implemented in a commercial virtual router with 8 GB RAM, 8 vCPU, on top of a virtualization platform with an 8-core 2.1 GHz processor. We measured the delay to answer route requests and route updates with a script running in a local machine that sent 800 queries per second. We repeated the experiment for different number of routes configured in the routing server. Each query requested or updated a different route, in order to avoid optimizations due to intermediate caches. We consider the network delay negligible. Figures 6.9a and 6.9b present boxplots of the time required to answer an IPv4 route request and a route update, respectively, for four different number of configured routes in the routing server. The values are relative to the minimum delay of a routing server with only one route.

We can see that the delay is not dependent on the number of routes. Since this architecture is designed to store network state hierarchically, it makes it easy to implement the routing server with a Patricia Trie. The delay of this data structure depends on the number of bits of the keys, not the number of elements [178]. Based on the data collected for this particular test, an equivalent deployment using a similar setup should scale to at least 3k endpoints without noticeable performance degradation. Each endpoint requires registering 3 routes (IPv4, IPv6 and MAC addresses), then 10k / 3 = ∼3k.

We chose to send 800 queries/s to the server since it is the peak requirement in the massive mobility scenario (section 6.4.3). We consider this a highly loaded server, however, in case we



(a) Boxplot (95%) of the delay of 10k route requests for four different number of configured routes.

(b) Boxplot (95%) of the delay of 10k route updates for four different number of configured routes.

Figure 6.9: Routing server performance evaluation depending on the number of stored routes. The values are relative to the minimum delay of a routing server with only one route.

needed to increase such figure, the architecture scales horizontally and can deploy more routing servers. Then, we load balance across edge routers by grouping them and pointing each group to a different routing server for the route requests, and perform route updates on all servers.

Finally, using the same routing server as before, we repeated the previous experiment but for different number of queries/s. Figure 6.10 presents boxplots of the delay to answer IPv4 route requests for four different number of queries per second to the routing server. The values are relative to the minimum delay of all samples. Assuming a mobility pattern similar to the warehouse scenario (section 6.4.3) with 800 moves/s and that each move triggers 2 queries to the routing server, we conclude that the routing server could support this use case (800*2 = 1600 queries/s).

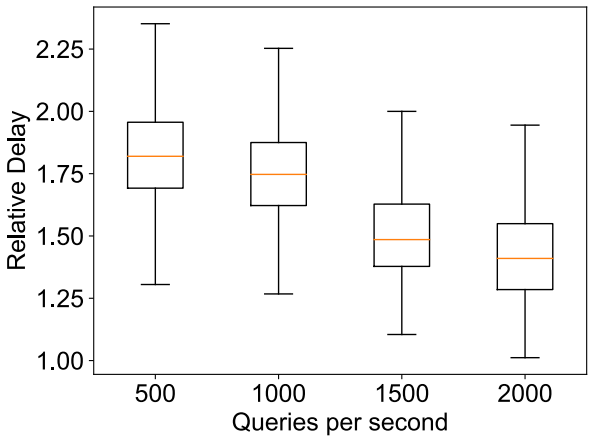


Figure 6.10: Boxplot (95%) of the delay of route requests for four different number of queries per second. Values are relative to the minimum of all.

6.4.2 State Reduction

In order to quantify the state reduction due to the reactive protocol, we counted the number of overlay-to-underlay IPv4 mappings in the FIB of the edge and border routers. We setup a VM that collected this data hourly from the router CLI. The routers were in two separate buildings (A and B), with three floors each, and providing network connectivity to between 200 and 500 users. Table 6.4 provides additional details about each deployment, and figure 6.11 shows the network topology. The control plane of border routers has a 4-core 2.4GHz, x86 CPU with 16 GB RAM, and edge routers a 4-core 1.8 GHz, x86 CPU with 8 GB RAM. Both of them use a custom ASIC for the data plane. The border-to-edge links are 10 Gbps and the edge-to-AP 1 Gbps.

	Bldg. A	**Bldg. B**
Border Routers	1	2
Edge Routers	7	6
Floors	3	3
AP per floor	40	40
Total AP	120	120
AP per edge	~20	20

Table 6.4: Details of campus deployments

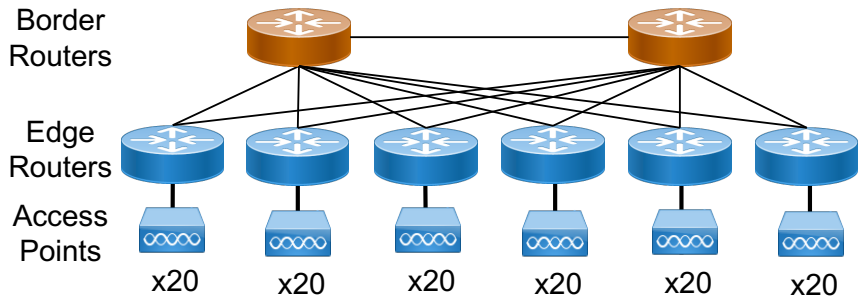


Figure 6.11: Campus Network Topology (underlay routers not shown for clarity)

Figure 6.12 shows the average number of FIB entries for the border and edge routers, for both buildings, and for three different weeks. We can see that, on average, edge routers store less FIB entries than border routers: in building A edge routers carry only about 30% of the FIB entries on the border routers, while in building B this figure is as low as 6%. Table 6.5 presents also averages of FIB entries for border and edge routers for a period of 5 weeks. We can also see reductions in FIB entries, even if we calculate separate averages for day and nighttime. We can conclude that the reactive approach helps in optimizing data plane state, while the border router absorbs the convergence delay of the route resolution.

It is also worth mentioning that the usage pattern of the FIB table on border routers shows a common daily and weekly pattern: during daytime in working week days routers host more routes

than during nighttime and over weekends. This is due to the fact that the border router is always up to date with the information in the routing server regarding the endpoints in the deployment. Thus, the number of entries in the border router follows closely the presence of authenticated users in the network, i.e. users in the office. In contrast, edge routers cache routes learned on demand and may retain them during longer periods, even when users have left the office. We can clearly see this in building A (figure 6.12, top row), where edge routers seem to keep their routes for most of the time between workdays, but eventually clear they caches during weekend (note that the edge router cache entries have a timeout of 24h). On the other hand, this effect is less noticeable in building B, where edge routers follow the daytime/nighttime routine more closely. The reason behind this could be nighttime traffic patterns: when some endpoints leave the office at night, the remaining ones may initiate communication with those that left, that will trigger a route resolution with a negative result, and thereby deleting that FIB entry.

Finally, another relevant aspect of building B is a substantial amount of end-hosts that are permanently connected to the network and do not follow the day/night routine. Examples of such end-hosts are desktops and IoT devices (VoIP phones, cameras) that do not move with the users.

Period	**Router**	**Building**	
		A	**B**
Day	Border	85	362
	Edge	47	42
Night	Border	19	227
	Edge	38	27
All	Border	50	291
	Edge	42	34
Decrease (All)		16%	88%

Table 6.5: Average number of FIB entries for a 5 week period, for day work hours (9 am to 19 pm), and nighttime.

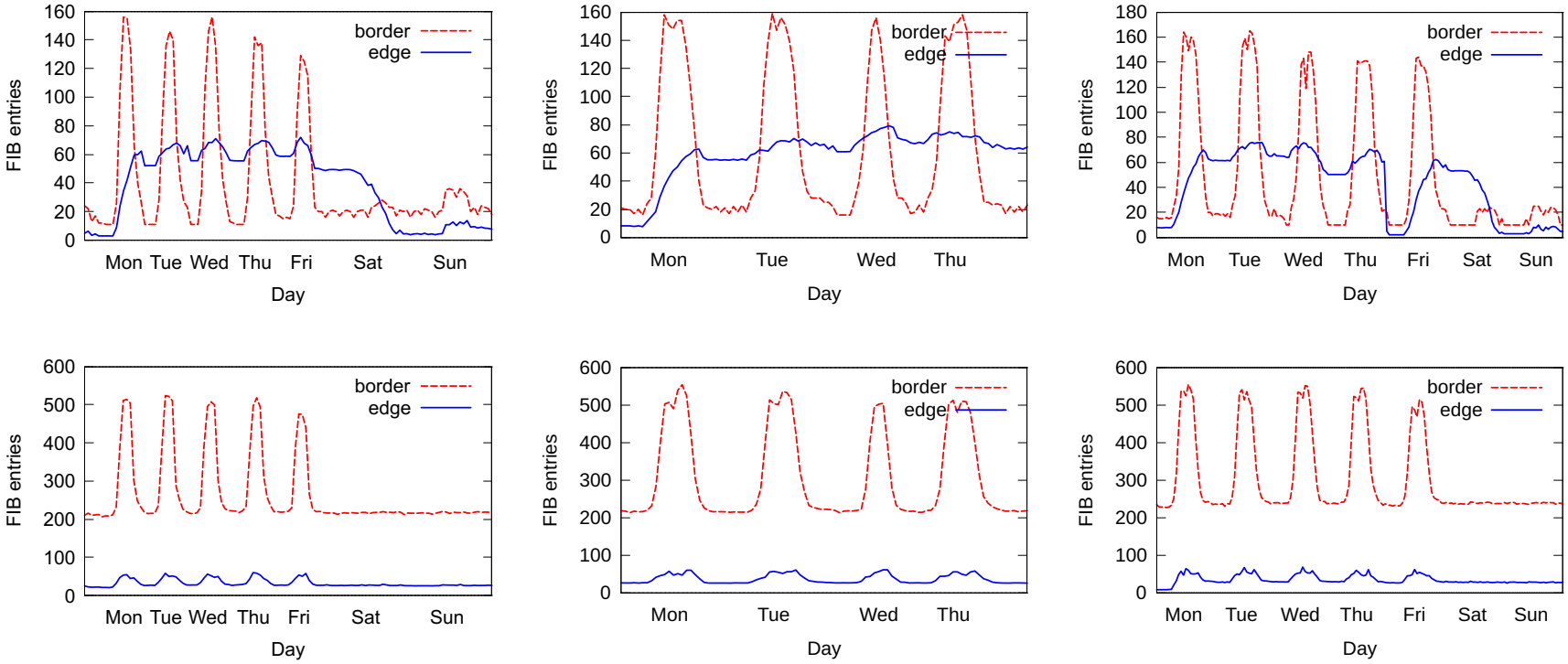


Figure 6.12: Number of FIB entries in border vs. edge router for three different weeks. Top row corresponds to building A, bottom row building to B.

6.4.3 Massive Mobility Events

In this experiment we focus on the handover delay of large mobility events for a reactive and proactive protocol. We recreated in the lab the specifications of a real life deployment, a large warehouse with hundreds of robots continuously roaming across WiFi access points. Figure 6.13 presents the topology:

- A commercial border router, 4-core 2 GHz CPU, 8 GB RAM for the control plane, custom ASIC in the data plane, and with an embedded routing server.
- Two commercial edge routers, 4-core 1.8 GHz CPU, 8 GB RAM for the control plane, and custom ASIC in the data plane.
- 198 edge routers emulated with a commercial traffic generator.

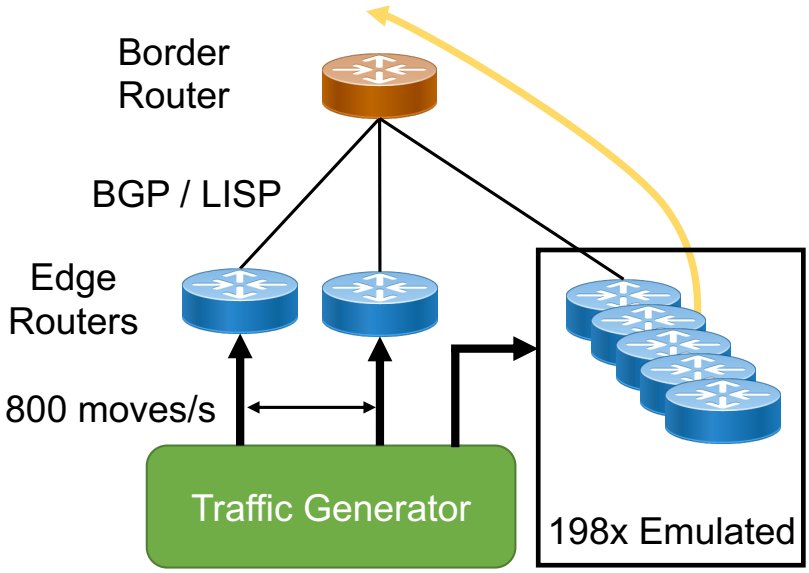


Figure 6.13: Warehouse Network Topology

We configured the traffic generator to: (i) create unidirectional UDP traffic from the 200 edge routers towards the border router (yellow arrow in figure 6.13), (ii) send 1500 bytes packets from 16,000 emulated hosts, and (iii) generate 800 mobility events per second by changing the attachment port of the end hosts between the two physical edge routers. We selected this amount of mobility events because in the real-life scenario, around 5% of the endpoints change their attachment point every second.

We measured the convergence time as the handover delay, i.e. the time since the emulated host is detached until traffic is restored after it attaches to the new edge router. In order to compare both proactive and on-demand approaches, we measured the handover delay in the same topology, but with two different control plane configurations: BGP as the proactive, and LISP as the reactive. In the BGP case we used a centralized route-reflector in the edge router to distribute route updates.

Figure 6.14 plots the CDF of the handover delay for BGP and LISP. All values are normalized to the minimum convergence time observed during the measurement process. We can see that the proactive solution takes around 10 times more to converge than the reactive one. The reason is that the proactive approach replicates network updates to all 200 edge routers, while the on-demand approach follows traffic patterns and only updates edge routers that have active traffic to roaming hosts.

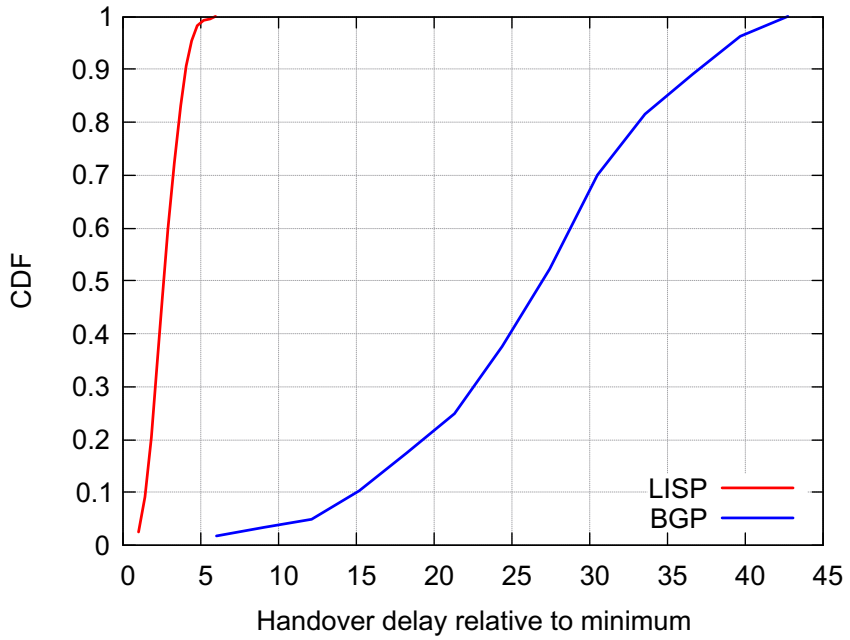


Figure 6.14: Handover delay for event-driven (LISP) and proactive (BGP) protocols

Another important observation is that the variance of the handover delay is consistently higher in the proactive approach than the reactive. This is due to the fact that the reactive architecture selectively updates only routers that are actively sending traffic to the end-host that moved, while the proactive approach updates edge routers randomly, i.e. not by their need for such update. These results show that SDA's reactive approach can be beneficial in stressed environments such as automated warehouses or large gatherings with highly mobile end-hosts.

6.5 Lessons Learned

In this section we summarize several challenges and our learnings from implementing and deploying SDA in enterprise networks.

6.5.1 Underlay Connectivity Issues

In order for the overlay to work, there needs to be underlay connectivity. However, it is possible that an edge router fails or that an underlay IP changes, interrupting normal traffic flow. It can be challenging for the overlay to know the state of the underlay without explicit probing. To cope

with these situations, edge routers monitor the address announcements of the underlay routing protocol (IS-IS or OSPF) to know about their reachability to underlay IP addresses of the other edge routers. This way, when they detect a connectivity outage, they update their local forwarding table deleting such route and falling back to the default route to the border, until a new edge router registers the overlay address in the routing server.

6.5.2 Edge Routers Rebooting

One issue that can happen is a forwarding loop between the edge and border router. First, assume the network is forwarding traffic, and an edge router reboots. It will start with an empty FIB for the overlay entries. When it receives traffic for one of its former endpoints, it will use the default route and forward it back to the border router, since it does not know yet its endpoints. Then, the border router will forward this traffic to the rebooted router according to its current information, creating a forwarding loop.

Although this loop is transient and disappears once the edge router detects its endpoints, we rely on two mechanisms in such situation. First, since the edge router will not announce its underlay IP address through the underlay routing protocol while rebooting, the tracking of the connectivity state of the underlay IP address that the other edge routers perform will remove the routes to the rebooting edge router. Second, the rebooting router will not recognize the incoming traffic, so it will send the data plane message we mentioned in section 6.3.4 to the originating edge router. This will trigger a refresh in the overlay FIB entries of the sending edge router.

6.5.3 Selecting the Policy Enforcement Point

In this section we discuss a bandwidth vs. network state trade-off related to the deployment of group policies. On one hand, we can save bandwidth by enforcing policies on ingress, because we don't forward traffic that will be dropped on egress. On the other hand, we can reduce data plane state by enforcing them on egress, because we only need policies for the local destination groups of the endpoints that are attached to a particular edge router. Note that on ingress we would need policies for *all* possible destination groups, thereby increasing data plane state. Taking this into account, we chose to enforce policies on egress to reduce overall state in the data plane.

In order to quantify the wasted bandwidth due to enforcing on egress, we analyzed the packet drops due to group-based ACLs in a real-life deployment leveraging egress-based policy enforcement. This deployment is a medium-large enterprise network with a campus and a few branches. For our analysis, we looked at three different devices in this deployment: a branch router, an edge device in the campus, and a VPN gateway. Combined, these three devices were serving around 11,000 endpoints during the period we monitored them. Figure 6.15 presents the permille of dropped traffic for the period of 5 days. We can see that in the worst case the drop rate is extremely low: 2 out of each 10k packets. The VPN router has a significantly larger amount of drops than the other routers due to the fact that it receives all the traffic from remote users, which present a different usage pattern from the users in the office.

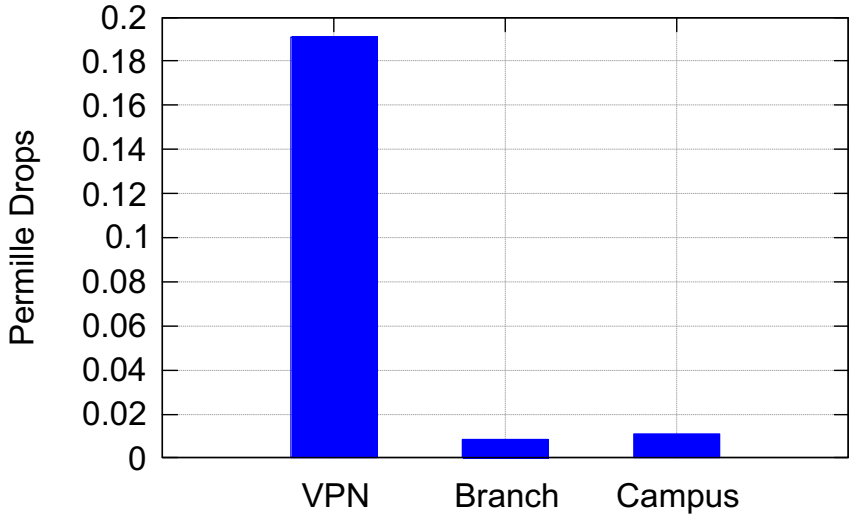


Figure 6.15: Permille hits on drop rules over all hits.

Since this deployment performs the policy enforcement on egress, we expected a significant percentage of drops. Surprisingly, we discovered that after a new policy is applied, there is a transient period with an increase in drops, but when endpoints (which are usually humans) realize they cannot access this particular destination, they stop requesting it. Hence, the operational experience in the most common enterprise use cases shows that *enforcing policies on egress does not impact significantly on the amount of wasted bandwidth.*

An additional benefit of enforcing on egress is a simplification of signaling. When a group rule is updated, it is necessary to notify all the affected edge routers. However, this is not easy in our design, because requests to the control plane are only triggered by data plane events. For example, consider that the policy check is implemented on ingress and that the edge router has already learned the GroupId associated to a particular destination (figure 6.16, top part). Now suppose that the group associated to this destination endpoint is updated, the ingress router has no way to know it. Hence, we need a way to signal this change.

On the contrary, if the policy is enforced on egress, the (Overlay IP, GroupId) pair in the VRF is automatically updated, because the modification of endpoint data automatically triggers the authentication process again. In other words, on egress the (Overlay IP, GroupId) pair is always up to date because it is linked to the endpoints connected to that edge router. This way, we can avoid implementing an extra signaling mechanism, and the associated complexity.

6.5.4 Updating Policies

In our deployment experience, we found an interesting trade-off when updating policies: it can be more scalable (i.e. less signaling) moving users to different groups rather than directly updating the group-based ACLs. Indeed, this trade-off depends on the distribution of endpoints within groups of each particular deployment, i.e. few groups with large amounts of endpoints vs. high number of groups with few endpoints each. Thus, it is not always the case that changing the endpoint's group is more scalable, but here we present two examples from our experience:

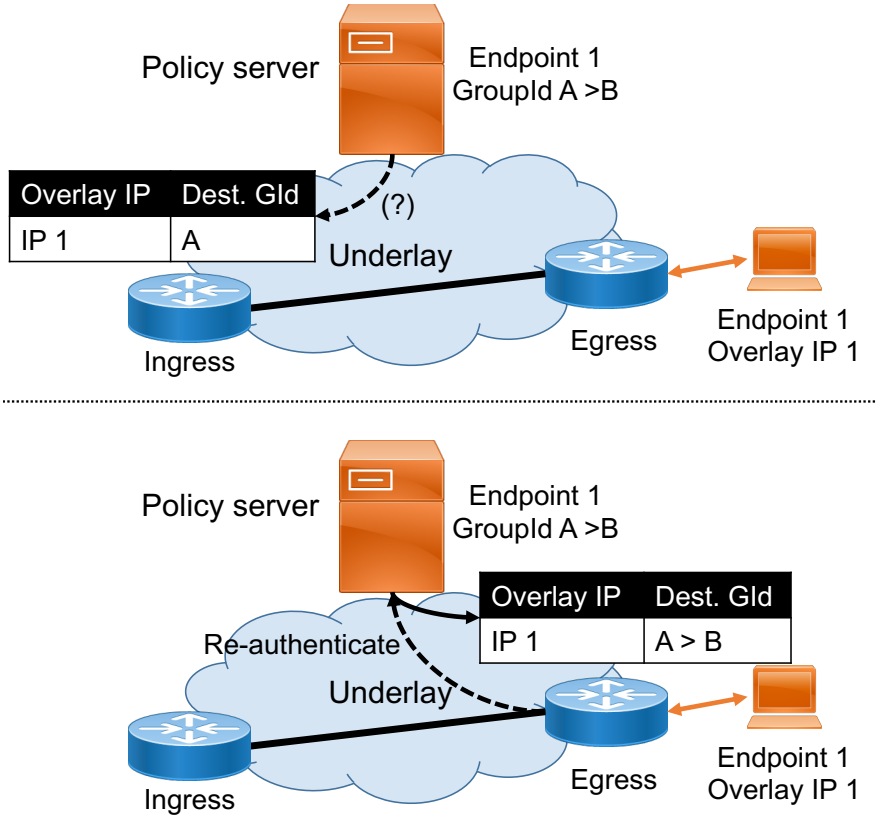


Figure 6.16: Policy Enforcement on Ingress (top) and Egress (bottom)

Acquisitions

In case of an acquisition of an enterprise, the new employees are progressively moved through different groups until they get the same group as regular employees. The reverse also holds, when part of a company is sold, their users are moved to a group that is associated with more restrictive policies.

Service Insertion

It is common that traffic has to go through middleboxes, e.g a firewall or a WAN optimizer. In some deployments, the SDA operators decide to update the group from the packets so that devices in the service chain decide whether to apply a policy or not. In other words, instead of applying different policies across the path for the same group, they change the group along the way so that different policies are applied across this same path.

6.6 Related Work

6.6.1 SDN Pioneering Work

Ethane [156], one of the first SDN designs, presents numerous similarities with our proposal: it also targets enterprise networks, presents a similar architecture with a centralized controller, and supports incremental deployment. However, Ethane specifically focuses on three key elements, mostly around the control plane: (i) providing network access control, (ii) a rich high-level policy language, and (iii) controller design and fault-tolerance. Conversely, SDA pays more attention to data plane related aspects such as network isolation and resource efficiency.

Campus networks also motivated the inception of OpenFlow [6], that champions the decoupling of data plane and control plane, gives a strong focus to the interface between the router and the controller and defines and approach to implementing rich policies on capable switches. SDA in its turn gives more emphasis to the ability to support scalability in heterogeneous environments with devices with different capabilities.

Finally, SANE [179] offers a simple, high-level policy interface like the one used by SDA, but tackles the problem in the border between L2 and L3 and does not trust the data plane routers.

6.6.2 BeyondCorp and Zero Trust Networks

In terms of securing the enterprise network, a closely related work is Beyond Corp [27], also known as the Zero Trust model. Like SDA, Beyond Corp focuses on access control between network endpoints, with an especial emphasis on user-to-server connections. On the other hand, it should be noted that networking improvements (e.g. seamless mobility, etc) are not in the scope of BeyondCorp and therefore this section only discusses how SDA relates to BeyondCorp in terms of enterprise security.

BeyondCorp offers a solid approach to build secure enterprise networks by means of keeping healthy endpoints and redirecting traffic through access proxies. By doing so, BeyondCorp presents a security model that is agnostic to the underlying networking infrastructure. While this is a reasonable approach in certain scenarios, in our operational experience we have found certain enterprise requirements that are hard to meet with a BeyondCorp-only approach. First, while BeyondCorp protects the access to the enterprise network, its focus is on protecting the access to enterprise applications at layer 7. However, it is not always possible to redirect traffic through the proxies (e.g. L2 traffic). Second, despite enterprise efforts around building healthy fleets of devices, the reality we observe is that insecure devices are still present in typical enterprise networks, even more due to the explosion of different IoT devices. These endpoints make the BeyondCorp approach harder to implement and, in many cases, secure on-boarding and admission still need to be performed by the network infrastructure. Third, with only a BeyondCorp security model the overall network performance could be degraded by malicious actors attempting to get access (even if unsuccessful) or explicitly looking to disrupt the network operation. SDA operates lower in the stack and not only protects the connection of devices to the network but also the network

infrastructure itself. We believe that SDA complements BeyondCorp, and the combination of both could contribute to strengthen the overall security of an enterprise network.

6.6.3 Other Related Work

Software Resolved Networks [157] also leverages a reactive protocol (DNS extensions) and a centralized database, but requires involving endpoints when resolving routes, and allows a wider range of policies than our proposal.

Other proposals for enterprise networks center their work on specific elements of an enterprise network, such as ACL configurations [167], systematic design of VLANs and ACL placement [180] or incremental SDN deployment [181].

6.7 Summary of Outcomes

In this chapter we have presented SDA, a solution designed for modern enterprise networks. The main goal of the architecture is supporting emerging requirements, with a strong focus on mobility, segmentation, and incremental deployment. At the same time, it provides scalability and optimizes data plane resources for heterogeneous environments with devices of diverse capabilities.

SDA leverages common practices in networking such as centralized control or network overlays, and makes use of a reactive approach to distribute network information and support mobility. Our experimental results show that, when compared with traditional approaches, SDA exceeds a 70% reduction in the overall forwarding state used by the network. Also, in very large deployments that need to deal with massive mobility events, network convergence is an order of magnitude faster than the status quo. For all these reasons we believe that SDA successfully addresses the unique requirements of modern enterprise networks that include scalable mobility, end-to-end segmentation, simplified administration, and overall resource optimization.

Chapter 7

A Design for Future Enterprise Networks

7.1 Introduction

In this chapter we present a design for future enterprise networks. First we explain the requirements we believe are necessary for such networks, and then we present a design that strives to fulfill them. We base our requirements, analysis, and design on the following use case. Consider an employee of a company that:

- Has several devices, e.g. smartphone, laptop. Some may be owned by the company, others by the employee.
- Needs to access the Internet, the company's application servers, and several cloud services.
- Wants to connect from virtually anywhere: company office, home, cafeteria, etc., and without noticeable difference for the user.
- The applications have different QoS requirements, ranging from email to real-time videoconferencing.

And that the company also requires:

- A way to define and apply policies.
- Support different kinds of applications, either web-based or not.
- Confidentiality for all communications
- Employee identification

In order to support this use case, we extracted a list of concrete requirements from the previous list, and designed a data and control plane architecture that can satisfy them. We leveraged part of our knowledge of VPNs and overlay networks from the previous chapters to build a layered stack. In such stack, each layer is represented by a protocol header, and we assign a single function to the headers. We must remark that this layered approach is not actually novel, rather it is already present in the wild. In a nutshell, our contributions are:

1. Formalize a trend that is already present in the global Internet.
2. Reduce the semantic overload of current headers: since our design assigns a unique function to each header, we reduce complexity and simplify the implementations of some functions. The most common example is using IP addresses to identify endpoints or users, while IP addresses should be used only to route packets. This type of coupling makes management and policy enforcement more complex, i.e. an endpoint cannot change its IP address because it would loose access to services that use it for identification.
3. Focus on enterprise use cases.

7.2 Requirements

Here we specify the technical requirements extracted from the previous high-level requisites, and their rationale.

Ability to traverse the public Internet: in order to connect from anywhere, we need a header that can be routed in the public Internet. This translates to using TCP or UDP on top of IP.

Seamless mobility and multi-homing: basically, this requisite comes from the need to support switching between WiFi and LTE interfaces in smartphones without breaking established connections, e.g. in conference calls. Or to be able to use both interfaces at the same time (multi-homing) to increase throughput, e.g. uploading a large file. Finally, it simplifies connecting from different devices.

Confidentiality: if we're crossing the public Internet, and we're in an enterprise scenario, it is essential to guarantee the confidentiality of any communication.

Policy: two key aspects (but not the only ones) of policy in an enterprise scenario are QoS and access control.

- Regarding QoS, it is valuable to have mechanisms to improve connection metrics. For example, reducing video call latency, or decreasing cost in large data transfers that are not time sensitive, e.g. backups.
- With respect to access control, we need a way to identify endpoints, group them, and control the resources they can access, as we saw in chapter 6. This is also related to the next requirement, identity.

Identity: in enterprise scenarios, it is important to provide a way to authenticate users easily instead of credentials based on user and password. In addition, in this case identity also refers to application identity. If the network is able to easily map packets to applications or services, then:

- QoS and policy enforcement are more simple: no need to perform Deep Packet Inspection (DPI), just use the label in the packet header.
- If the application labeling is implemented correctly, it can increase privacy versus other solutions. For example, a label that maps to `'latency < 50 ms'` offers more privacy than `'VIP conferencing call'`.

Support for applications using the socket API: as we'll see later, in some cases the socket API that we expose to applications is not the standard one that uses IP and port. Since we augment the network layer with additional features, we need to expose them to the application, e.g. user identity. However, since the existing code base uses the well-established socket API, we have to support it.

7.3 Design

Table 7.1 presents the layered stack that is the basis of our approach, with a brief description and the function of each layer. Then, table 7.2 translates these layers to a candidate protocol that can implement them. The sign $*$ denotes that the protocol needs modifications.

Layer	**Description**	**Function**
Payload	Data to transfer	
Transport	Transport protocol	
Application Data	Information to identify the application	Policy enforcement
Identity	Information to identify the endpoint or user	
Confidentiality	Data to encrypt the layers above	Confidentiality
Routing	Standard IP headers	Routing, mobility

Table 7.1: Layered approach for our design

Layer	**Candidate Protocol**
Payload	n/a
Transport	Any transport protocol determined by the application
Application Data	Custom header
Identity	
Confidentiality	WireGuard$*$
Routing	Plain IP + UDP

Table 7.2: Proposed protocol stack.

The custom header carries information about policies and identity, with the fields detailed in table 7.3. These fields are used to enforce policies and offer applications a different socket API with extended characteristics. Specifically:

- `userID` identifies the specific user in the endpoint, and is used to create security associations for the confidentiality header and to identify users in applications that require login.
- `groupID` is an identifier of a collection of users, similar to the one in section 6.3.2. It is used to apply access control policies.
- `appID` identifies the application in the data packets, so it can be used to provide different

QoS levels for each application. We are assuming that these fields are inserted either by a trusted software on the endpoint or the first hop router (section 7.4).

- Together, `userID` and `appID` can be offered to the application to authenticate users if needed.

Field	Description
`socket_API=1`	A bit that indicates if the application is using the standard socket API (section 7.3.1) or the application and user fields.
`userID`	User identifier within the enterprise network.
`groupID`	Group identifier to which the `userID` belongs.
`appID`	Identifier for the application that the user is accessing.

Table 7.3: Fields in the custom header.

In addition, we can see that this layered stack can support the aforementioned requirements. First, the custom header is used to apply different kinds of policies and identify users. Second, the WireGuard header provides confidentiality. Third, the plain IP + UDP header can cross the Internet easily. Finally, since the IP address in the IP header is not linked to the endpoint, it can change freely, thereby allowing mobility and multi-homing. Note that for multi-homing we need cooperation form the control plane, as described in section 7.4.

7.3.1 Support for Socket API

Table 7.4 adds an additional IP layer before the transport layer, so that applications that use the socket API have access to an IP + port pair to open sockets. The custom header uses the `socket_API` bit to signal the presence of this IP header.

Layer	Candidate Protocol
Payload	n/a
Transport	Any transport protocol determined by the application
Standard socket API	Plain IP
Application	Custom header `socket_API=1`
Identity	
Confidentiality	WireGuard∗
Routing	Plain IP + UDP

Table 7.4: Proposed protocol stack with support for standard socket API (IP + port).

7.3.2 Extended Socket API

In case we're not using the standard socket API, we offer applications a different kind of socket with extended functionalities. For example, they can use the `userID` field to authenticate application users. Or they don't need confidentiality at application level, that is, TLS, because all connections are encrypted at L3. In addition, in order to keep transport protocols unmodified, we can use the mapping system to tell applications which port to use in case they are not aware of it. Another option is using new transport independent socket APIs that offer similar additional functionalities [182].

7.4 Deployment

In our target deployment, we make the following assumptions:

- We have a collection of WAN routers across the Internet that can process the custom header. In conjunction they form a Software-Defined WAN (SDWAN).
- We have one or more servers acting as a control plane; they host a key-value database with different information required to establish connections.
- We can modify the host protocol stack, either in endpoints or servers, so they can use the custom header and the extended socket API. This stack is trusted.
- In case we cannot modify the host stack, or we don't trust it, the endpoints or servers work as usual, with the IP + port socket API. Then the WAN routers add and remove the necessary headers. In addition,
  - Endpoints are able to create a secure tunnel to the nearest WAN router, either with IPsec or WireGuard.
  - Servers are connected to a WAN router in their datacenter, or have an IPsec or WireGuard tunnel towards the nearest.

Figure 7.1 presents an example of a deployment of such architecture. Consider an endpoint that wants to connect to the on-premise servers in the enterprise, and that it supports our stack. First, it will contact the mapping system to obtain the IP address of the closest WAN router, and create the secure WireGuard tunnel. Then, it will ask for the necessary data to fill the custom header, especially `userID` and `groupID`, and register its underlay IP in the mapping system. The mapping system supports different types of queries and read / write operations depending on the the requesting user.

After gathering all these data, the endpoint can start sending application data. Packets that arrive at the WAN router are decapsulated and the router processes the custom header. Specifically, the router checks if the user has the permission to access that application, and determines the best destination router for the packet depending on the QoS associated to the application. The WAN

router can map the fields in the custom headers to specific polices thanks to the information in the mapping system. Then, the packet is routed across the necessary WAN routers. When it arrives at destination, the on-premise server strips all the headers and handles the payload to the application.

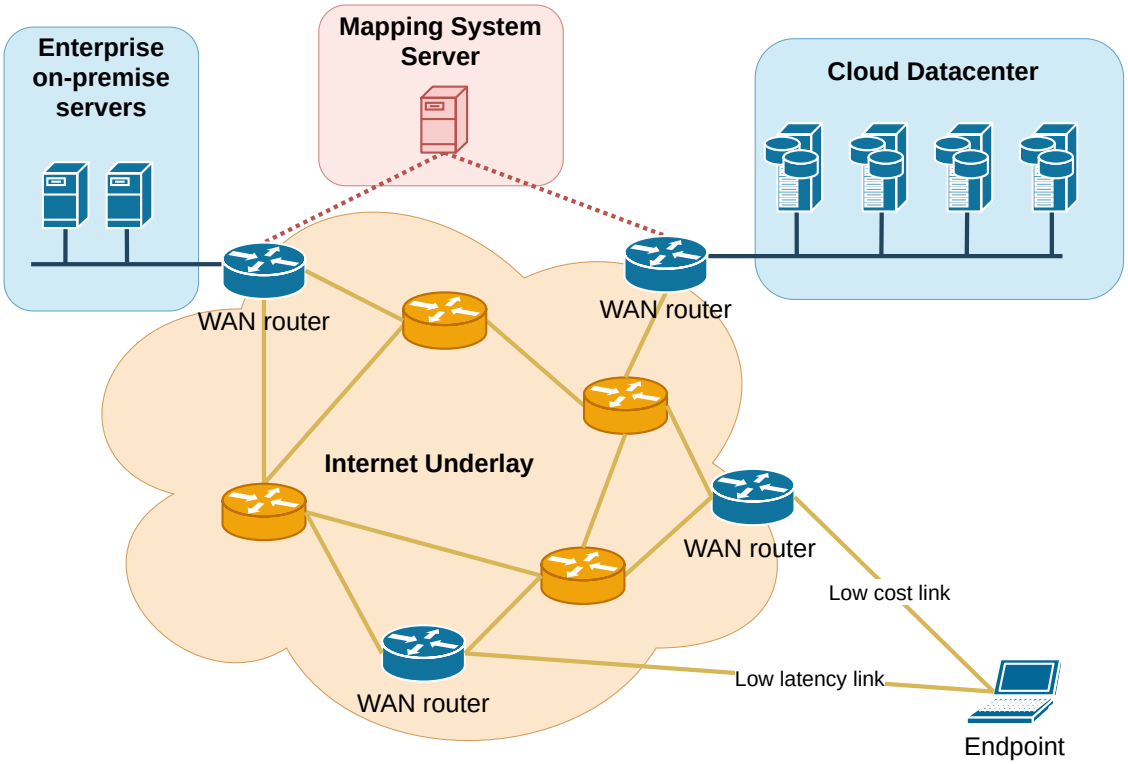


Figure 7.1: Deployment example of the layered architecture. Some connections have been omitted for clarity.

Regarding multi-homing, we leverage the control plane connection between the endpoint and the mapping system to help the endpoint establish such connections. For example, the endpoint can register two different underlay IP addresses in the mapping system, so the receiving WAN router knows that the endpoint is not performing a handover but rather sending data from two different interfaces. Another example is when the mapping system gives the IP addresses of two WAN routers to the endpoint, so it can connect to a each router from a different interface (split tunneling).

Finally, in case the endpoint cannot use the custom header, it establishes an IPsec tunnel to the closest WAN router. Then, this router adds all the required headers on top of the regular IP packet. For example, it retrieves the `userID` from the mapping system using the credentials from the IPsec security association. Equivalently for the destination server.

7.5 Summary of Outcomes

In this chapter we have presented the basis of a design to support the emerging requirements of enterprise networks. Such networks require confidential connections from any point on the

Internet, and they need a way to implement application and user-specific polices. We have started with a solution based on layers, where each layer performs a specific function, and translated each layer to a protocol header. This way, we can offer mobility, multi-homing, data confidentiality, access control, QoS, and user identification. However, this design needs future refinements, like the packet processing in WAN routers, or the specification of an SDN southbound protocol that supports different types of queries, and read / write permissions.

Chapter 8

Conclusions

8.1 Contributions

In this thesis we have presented several improvements to different elements of overlay networks, based on the latest trends in networking research, especially SDN and programmable dataplanes. We have paid special attention to distributed mapping systems and southbound SDN protocols. The following paragraphs summarize the contributions of each chapter of this thesis.

The fist part of the thesis has explored the applicability of blockchain technology to address assignment, a topic that has received little attention when compared with financial applications or Internet naming systems. Specifically, we have studied the relationship between IP addresses and cryptographic tokens, that allows recording them in a blockchain. In order to understand better the potential benefits of such technology, we have explored the similarities and differences of blockchain technology and classical Public Key Infrastructure solutions. Moreover, thanks to a detailed study of the state of the art of consensus algorithms for blockchains, we have presented two practical use cases for blockchain-based address assignment. First, improving the security of inter-domain routing, and second, establishing connections between collaborating but distrusting enterprises. In both cases, we have paid special attention to the selection of the consensus algorithm, scalability metrics like chain size or throughput, and other deployment considerations, such as dealing with IPv4 and IPv6 addresses or simplifying the administrator's job. Thanks to this work, we have successfully shown that blockchain technology can be used to assign IP addresses, and that it constitutes a feasible alternative to classic PKI designs.

The second part of the thesis has presented improvements in three aspects related to overlay and enterprise networks: secure data planes, current enterprise networks, and future refinements for enterprise networks. First, we have presented and evaluated a design to automate the deployment of VPNs based on LISP and WireGuard. This design strives for simplicity, focusing on reducing data plane state and connection establishment time. We have shown that we can automate the deployment of a secure overlay network on top of the Internet, while avoiding complex systems like DNS or certification authorities, and without impacting performance.

Second, we have reviewed the challenges of current enterprise networks, and discussed why current practices are not sufficient. This analysis has helped in designing a solution based on network overlays and state of the art research such as SDN centralized control, network databases, or group-based policies. The usage of a reactive protocol for the routing part in this solution is especially relevant, because this approach is not widespread in the industry although it can yield interesting benefits. Specifically, we have proven how this different approach can reduce data plane state and handover time, and, in turn, decrease router cost. Finally, we must remark that this design has contributed in advancing the already mature field of research in enterprise networks, that has seen few contributions in the last years. And third, by combining the previous contributions, we have outlined an approach to extend enterprise networks outside of the campus. This design aims to extend common enterprise functionalities like access control or confidentiality to endpoints connected outside of the campus perimeter, and at the same time, support QoS, mobility and multi-homing. We have chosen to clearly separate each function in a different layer, and use a

centralized mapping system to distribute network state. Finally, we have presented a new socket design that aims to provide more features to applications, along with an option to be compatible with the current socket API.

8.2 Future Work

8.2.1 Blockchain Applications to Address Assignment

Regarding the application to inter-domain routing, the most relevant future work revolves around adding all or part of the BGP AS-path in IPchain, so that BGP routers have additional data to verify BGP messages. Specifically, it is worth investigating how to ensure that only authorized neighboring ASes can append a new ASN, and a scalability analysis for this new data, because the churn, throughput, and storage requirements are significantly higher. Indeed, this means storing all of the active Internet BGP routes in the blockchain. On the performance side, it is worth investigating data structures for efficient management of IP prefixes in the chain (split / merge operations), and a detailed analysis on the PoS protocol scalability, e.g. how does the DKG group size affect security and block time. On the same line, it is also interesting to review the applicability of new consensus algorithms to this use case, like novel PoS approaches that keep appearing in the literature, or the ones leveraging a network of trust. Finally, it is worth investigating how to implement the flexible trust architectures we mentioned before, and the possible trade-offs that may appear.

With respect to deployments for private networks, we believe that the two key points that should be addressed are: first, more scalability testing, in order to determine an upper bound on the maximum number of participants without degrading performance. Second, refining or standardizing the interface between the router and the blockchain. In addition, it is valuable understanding how to use a private chain as a secure and distributed mapping system for roaming endpoints with limited computational power, especially regarding read / write performance during a handover event.

8.2.2 Architectures for Enterprise Networks

Possible improvements for the WireGuard control plane are the ones detailed in section 5.6, like multi-homing, and improving mobility management to support double jumps. In addition, it would be interesting to automate the IP address assignment of overlay IP addresses.

Regarding current enterprise networks, we believe it is worth studying the traffic pattern of such networks, since there are few publications on the topic in the last 5-10 years, and it can help in future designs. Other relevant topics are quantifying the number of packets that use the default route and the associated cost, translating FIB reduction to cost, or a deeper study on the trade-offs when distributing group policies to data plane routers.

Finally, the design for future enterprise networks offers a wide range of open challenges, such as building a prototype to detail router workflow, how to ensure QoS, or the design of the southbound

protocol between the mapping system, and the routers and endpoints.

8.3 List of Publications

These are the publications associated to this thesis. For the *Blockchain Applications to Address Assignment* part:

- Jordi Paillissé, Miquel Ferriol, Eric Garcia, Hamid Latif, Carlos Piris, Albert López, Brenden Kuerbis, Alberto Rodríguez-Natal, Vina Ermagan, Fabio Maino, and Albert Cabellos. IPchain: Securing IP Prefix Allocation and Delegation with Blockchain, *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 2018*, pp. 1236-1243, DOI: `https://doi.org/10.1109/Cybermatics_2018.2018.00218`.

- Jordi Paillissé, Jordi Subira, Albert López, Alberto Rodríguez-Natal, Vina Ermagan, Fabio Maino, and Albert Cabellos. Distributed Access Control with Blockchain, *ICC 2019 - 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 2019*, pp. 1-6, DOI: `https://doi.org/10.1109/ICC.2019.8761995`.

- Jordi Paillissé, Jan Manrique, Guilem Bonet, Alberto Rodríguez-Natal, Fabio Maino, and Albert Cabellos, Decentralized Trust in the Inter-Domain Routing Infrastructure, *IEEE Access*, vol. 7, pp. 166896-166905, 2019, DOI: `https://doi.org/10.1109/ACCESS.2019.2954096`.

And regarding the *Architectures for Enterprise Networks* part:

- Jordi Paillissé, Marc Portolés, Albert López, Alberto Rodríguez-Natal, David Iacobacci, Johnson Leong, Victor Moreno, Albert Cabellos, Fabio Maino, and Sanjay Hooda. 2020. SD-Access: Practical Experiences in Designing and Deploying Software Defined Enterprise Networks. *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2020.* Association for Computing Machinery, New York, NY, USA, 496–508. DOI: `https://doi.org/10.1145/3386367.3431288`

- Jordi Paillissé, Alejandro Barcia, Albert López, Alberto Rodríguez-Natal, Fabio Maino, and Albert Cabellos. A Control Plane for WireGuard. Accepted for publication, April 2021. *ICCCN 2021 - 30th IEEE International Conference on Computer Communications and Networks, Athens, Greece, 2021 (Online).*

Bibliography

[1] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015. doi: 10.1109/JPROC.2014.2371999.

[2] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, and W. Kellerer. Interfaces, attributes, and use cases: A compass for sdn. *IEEE Communications Magazine*, 52(6):210–217, 2014. doi: 10.1109/MCOM.2014.6829966.

[3] R. Bifulco and G. Rétvári. A survey on the programmable data plane: Abstractions, architectures, and open problems. In *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–7, 2018. doi: 10.1109/HPSR.2018.8850761.

[4] Carolyn Jane Anderson, Nate Foster, Arjun Guha, Jean-Baptiste Jeannin, Dexter Kozen, Cole Schlesinger, and David Walker. Netkat: Semantic foundations for networks. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '14, page 113–126, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450325448. doi: 10.1145/2535838.2535862. URL `https://doi.org/10.1145/2535838.2535862`.

[5] Y. Li, X. Yin, Z. Wang, J. Yao, X. Shi, J. Wu, H. Zhang, and Q. Wang. A survey on network verification and testing with formal methods: Approaches and challenges. *IEEE Communications Surveys Tutorials*, 21(1):940–969, 2019. doi: 10.1109/COMST.2018.2868050.

[6] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008. ISSN 0146-4833. doi: 10.1145/1355734.1355746. URL `https://doi.org/10.1145/1355734.1355746`.

[7] P4 runtime, March 2021. URL `https://p4lang.github.io/p4runtime/spec/v1.3.0/P4Runtime-Spec.html`.

[8] Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Natasha Gude, Paul Ingram, Ethan Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, Rajiv Ramanathan, Scott Shenker, Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt,

Alexander Yip, and Ronghua Zhang. Network virtualization in multi-tenant datacenters. page 203–216, 2014. URL https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-koponen.pdf.

[9] Alberto Rodriguez-Natal, Marc Portoles-Comeras, Vina Ermagan, Darrel Lewis, Dino Farinacci, Fabio Maino, and Albert Cabellos-Aparicio. Lisp: a southbound sdn protocol? *IEEE Communications Magazine*, 53(7):201–207, 2015. doi: 10.1109/MCOM.2015.7158286.

[10] Martin Casado, Teemu Koponen, Scott Shenker, and Amin Tootoonchian. Fabric: A retrospective on evolving sdn. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, pages 85–90, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1477-0. doi: 10.1145/2342441.2342459. URL http://doi.acm.org/10.1145/2342441.2342459.

[11] Pablo Belzarena, Gabriel Gómez Sena, Isabel Amigo, and Sandrine Vaton. Sdn-based overlay networks for qos-aware routing. In *Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks*, LANCOMM '16, page 19–21, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450344265. doi: 10.1145/2940116.2940121. URL https://doi.org/10.1145/2940116.2940121.

[12] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, SOSP '01, page 131–145, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133898. doi: 10.1145/502034.502048. URL https://doi.org/10.1145/502034.502048.

[13] C. E. Perkins. Mobile ip. *IEEE Communications Magazine*, 35(5):84–99, May 1997. ISSN 1558-1896. doi: 10.1109/35.592101.

[14] Ved P. Kafle and Masugi Inoue. Locator id separation for mobility management in the new generation network. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 1(2/3):3–15, 9 2010. doi: 10.22667/JOWUA.2010.09.31.003. URL http://isyou.info/jowua/papers/jowua-v1n23-1.pdf.

[15] L. E. Li and T. Woo. Vsite: A scalable and secure architecture for seamless l2 enterprise extension in the cloud. In *2010 6th IEEE Workshop on Secure Network Protocols*, pages 31–36, Oct 2010. doi: 10.1109/NPSEC.2010.5634451.

[16] Changhoon Kim, Matthew Caesar, and Jennifer Rexford. Floodless in seattle: A scalable ethernet architecture for large enterprises. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, page 3–14, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581750. doi: 10.1145/1402958.1402961. URL https://doi.org/10.1145/1402958.1402961.

[17] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys Tutorials*, 7(2):72–93, Second 2005. ISSN 1553-877X. doi: 10.1109/COMST.2005.1610546.

[18] Arturo Crespo and Hector Garcia-Molina. Semantic overlay networks for p2p systems. In Gianluca Moro, Sonia Bergamaschi, and Karl Aberer, editors, *Agents and Peer-to-Peer Computing*, pages 1–13, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31657-2. doi: 10.1007/11574781_1. URL `https://doi.org/10.1007/11574781_1`.

[19] S. Y. Shi and J. S. Turner. Routing in overlay multicast networks. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1200–1208 vol.3, June 2002. doi: 10.1109/INFCOM.2002.1019370.

[20] Zhi Li and P. Mohapatra. Qron: Qos-aware routing in overlay networks. *IEEE Journal on Selected Areas in Communications*, 22(1):29–40, Jan 2004. ISSN 1558-0008. doi: 10.1109/JSAC.2003.818782.

[21] Ben Y. Zhao, Yitao Duan, Ling Huang, Anthony D. Joseph, and John D. Kubiatowicz. Brocade: Landmark routing on overlay networks. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 34–44, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45748-0. doi: 10.1007/3-540-45748-8_3. URL `https://doi.org/10.1007/3-540-45748-8_3`.

[22] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Larry Kreeger, T. Sridhar, Mike Bursell, and Chris Wright. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC 7348, August 2014. URL `https://rfc-editor.org/rfc/rfc7348.txt`.

[23] Dino Farinacci, Vince Fuller, David Meyer, and Darrel Lewis. The Locator/ID Separation Protocol (LISP). RFC 6830, January 2013. URL `https://rfc-editor.org/rfc/rfc6830.txt`.

[24] Bruce Davie, Teemu Koponen, Justin Pettit, Ben Pfaff, Martin Casado, Natasha Gude, Amar Padmanabhan, Tim Petty, Kenneth Duda, and Anupam Chanda. A database approach to sdn control plane design. *SIGCOMM Comput. Commun. Rev.*, 47(1):15–26, January 2017. ISSN 0146-4833. doi: 10.1145/3041027.3041030. URL `https://doi.org/10.1145/3041027.3041030`.

[25] L. Jakab, A. Cabellos-Aparicio, F. Coras, D. Saucez, and O. Bonaventure. Lisp-tree: A dns hierarchy to support the lisp mapping system. *IEEE Journal on Selected Areas in Communications*, 28(8):1332–1343, October 2010. ISSN 1558-0008. doi: 10.1109/JSAC.2010.101011.

[26] M. Boucadair, C. Jacquenet, D. Phung, and S. Secci. Lisp-msx: Decentralized interconnection of independent lisp mapping systems. *IEEE Communications Magazine*, 57(1):35–41, January 2019. ISSN 1558-1896. doi: 10.1109/MCOM.2018.1701323.

[27] Rory Ward and Betsy Beyer. Beyondcorp: A new approach to enterprise security. *;login:*, Vol. 39, No. 6:6–11, 2014. URL https://www.usenix.org/system/files/login/articles/login_dec14_02_ward.pdf.

[28] Jason A Donenfeld. Wireguard: Next generation kernel network tunnel. In *NDSS*, 2017. doi: 10.14722/ndss.2017.23160. URL http://dx.doi.org/10.14722/ndss.2017.23160.

[29] Namecoin. Namecoin, March 2020. URL https://namecoin.org/.

[30] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*, pages 181–194, Denver, CO, June 2016. USENIX Association. ISBN 978-1-931971-30-0. URL https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali.

[31] N. Bozic, G. Pujolle, and S. Secci. A tutorial on blockchain and applications to secure network control-planes. In *2016 3rd Smart Cloud Networks Systems (SCNS)*, pages 1–8, Dec 2016. doi: 10.1109/SCNS.2016.7870552.

[32] Lucian Popa, Ali Ghodsi, and Ion Stoica. Http as the narrow waist of the future internet. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304092. doi: 10.1145/1868447.1868453. URL https://doi.org/10.1145/1868447.1868453.

[33] Amine El Malki and Uwe Zdun. Guiding architectural decision making on service mesh based microservice architectures. In Tomas Bures, Laurence Duchien, and Paola Inverardi, editors, *Software Architecture*, pages 3–19, Cham, 2019. Springer International Publishing. ISBN 978-3-030-29983-5. doi: 10.1007/978-3-030-29983-5_1. URL https://doi.org/10.1007/978-3-030-29983-5_1.

[34] Wubin Li and Yves Lemieux et al. Service mesh: Challenges, state of the art, and future research opportunities. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 122–1225, 2019. doi: 10.1109/SOSE.2019.00026.

[35] Lee Calcote and Zack Butcher. *Istio: Up and running: Using a service mesh to connect, secure, control, and observe*. O'Reilly Media, October 2019. ISBN 9781492043782.

[36] Gianni Antichi and Gábor Rétvári. Full-stack sdn: The next big challenge? In *Proceedings of the Symposium on SDN Research*, SOSR '20, page 48–54, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371018. doi: 10.1145/3373360.3380834. URL https://doi.org/10.1145/3373360.3380834.

[37] Pamela Zave and Jennifer Rexford. The compositional architecture of the internet. *Commun. ACM*, 62(3):78–87, February 2019. ISSN 0001-0782. doi: 10.1145/3226588. URL https://doi.org/10.1145/3226588.

[38] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, 2008. URL `https://nakamotoinstitute.org/bitcoin/`.

[39] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys Tutorials*, 22(2):1432–1465, 2020. doi: 10.1109/COMST.2020.2969706.

[40] NguyenGiang-Truong and KimKyungbaek. A survey about consensus algorithms used in blockchain. *Journal of Information Processing Systems*, 14(1):101–128, 02 2018. doi: 10.3745/JIPS.01.0024.

[41] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982. ISSN 0164-0925. doi: 10.1145/357172.357176. URL `https://doi.org/10.1145/357172.357176`.

[42] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. Some constraints and tradeoffs in the design of network communications. *SIGOPS Oper. Syst. Rev.*, 9(5):67–74, November 1975. ISSN 0163-5980. doi: 10.1145/1067629.806523. URL `https://doi.org/10.1145/1067629.806523`.

[43] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186, USA, 1999. USENIX Association. ISBN 1880446391. URL `https://www.usenix.org/legacy/publications/library/proceedings/osdi99/full_papers/castro/castro.ps`.

[44] Miguel Correia, Giuliana Santos Veronese, Nuno Ferreira Neves, and Paulo Verissimo. Byzantine consensus in asynchronous message-passing systems: a survey. *International Journal of Critical Computer-Based Systems*, 2(2):141–161, 2011. doi: 10.1504/IJCCBS.2011.041257. URL `https://www.inderscienceonline.com/doi/abs/10.1504/IJCCBS.2011.041257`.

[45] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA, June 2014. USENIX Association. ISBN 978-1-931971-10-2. URL `https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro`.

[46] Oracle. Oracle blockchain platform adds the raft consensus algorithm, February 2020. URL `https://blogs.oracle.com/blockchain/oracle-blockchain-platform-adds-the-raft-consensus-algorithm`.

[47] Sunoo Park, Krzysztof Pietrzak, Joël Alwen, Georg Fuchsbauer, and Peter Gazi. Spacemint: A cryptocurrency based on proofs of space. *IACR Cryptology ePrint Archive*, 2015. URL `https://eprint.iacr.org/2015/528`.

[48] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. *Citeseer*, July 2013. URL `http://primecoin.io/bin/primecoin-paper.pdf`.

[49] NEM Platform. Nem technical reference, February 2018. URL https://nemplatform.com/wp-content/uploads/2020/05/NEM_techRef.pdf.

[50] Nicolas Houy. It will cost you nothing to 'kill' a proof-of-stake crypto-currency. *Available at SSRN 2393940*, 2014. doi: 10.2139/ssrn.2393940. URL https://dx.doi.org/10.2139/ssrn.2393940.

[51] P. Gaži, A. Kiayias, and A. Russell. Stake-bleeding attacks on proof-of-stake blockchains. In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 85–92, 2018. doi: 10.1109/CVCBT.2018.00015.

[52] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68, 2017. doi: 10.1145/3132747.3132757.

[53] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 357–388, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63688-7. doi: 10.1007/978-3-319-63688-7_12. URL https://doi.org/10.1007/978-3-319-63688-7_12.

[54] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In Ian Goldberg and Tyler Moore, editors, *Financial Cryptography and Data Security*, pages 23–41, Cham, 2019. Springer International Publishing. ISBN 978-3-030-32101-7. doi: 10.1007/978-3-030-32101-7_2. URL https://doi.org/10.1007/978-3-030-32101-7_2.

[55] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 31–42, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978399. URL https://doi.org/10.1145/2976749.2978399.

[56] Timo Hanke, Mahnush Movahedi, and Dominic Williams. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548*, 2018. URL https://arxiv.org/abs/1805.04548.

[57] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017. URL http://arxiv.org/abs/1710.09437.

[58] block.one. Eos.io technical white paper v2, February 2020. URL https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.

[59] Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on bft consensus. *arXiv preprint arXiv:1807.04938*, 2018. URL https://arxiv.org/abs/1807.04938.

[60] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 2015. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.696.93&amp=&rep=rep1&amp=&type=pdf`.

[61] Brad Chase and Ethan MacBrough. Analysis of the xrp ledger consensus protocol. *arXiv preprint arXiv:1802.07242*, 2018. URL `https://arxiv.org/abs/1802.07242`.

[62] Serguei Popov. The tangle. *White paper*, 2018. URL `https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf`.

[63] IOTA Foundation. Coordicide - the next step in iota's evolution, February 2020. URL `https://coordicide.iota.org/`.

[64] Yakov Rekhter, Susan Hares, and Tony Li. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006. URL `https://rfc-editor.org/rfc/rfc4271.txt`.

[65] Sharon Goldberg. Why is it taking so long to secure internet routing? *Commun. ACM*, 57 (10):56–63, September 2014. ISSN 0001-0782. doi: 10.1145/2659899. URL `http://doi.acm.org/10.1145/2659899`.

[66] Ryan Singel. Pakistan's accidental youtube re-routing exposes trust flaw in net, February 2008. URL `https://www.wired.com/2008/02/pakistans-accid/`.

[67] Matt Lepinski and Stephen Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480, February 2012. URL `https://rfc-editor.org/rfc/rfc6480.txt`.

[68] RIPE NCC. Rpki certification statistics, March 2019. URL `http://certification-stats.ripe.net/`.

[69] Danny Cooper, Ethan Heilman, Kyle Brogle, Leonid Reyzin, and Sharon Goldberg. On the risk of misbehaving rpki authorities. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, pages 16:1–16:7, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2596-7. doi: 10.1145/2535771.2535787.

[70] Kieron O'hara and Wendy Hall. Four internets: the geopolitics of digital governance. *CIGI Papers No. 206*, 2018. URL `https://www.cigionline.org/publications/four-internets-geopolitics-digital-governance/`.

[71] Scott Malcomson. *Splinternet: How geopolitics and commerce are fragmenting the World Wide Web*. OR Books, 2016. ISBN 978-1-682190-30-2. URL `https://www.orbooks.com/catalog/splinternet-by-scott-malcomson/`.

[72] Yossi Gilad, Omar Sagga, and Sharon Goldberg. Maxlength considered harmful to the rpki. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments*

and Technologies, CoNEXT '17, pages 101–107, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5422-6. doi: 10.1145/3143361.3143363.

[73] Ethan Heilman, Danny Cooper, Leonid Reyzin, and Sharon Goldberg. From the consent of the routed: Improving the transparency of the rpki. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 51–62, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2836-4. doi: 10.1145/2619239.2626293.

[74] Wes George. Adventures in RPKI (non) deployment. Technical report, NANOG 62, 2014. URL `https://archive.nanog.org/sites/default/files/wednesday_george_adventuresinrpki_62.9.pdf`.

[75] Brenden Kuerbis and Milton Mueller. Internet routing registries, data governance, and security. *Journal of Cyber Policy*, 2(1):64–81, 2017. doi: 10.1080/23738871.2017.1295092. URL `https://doi.org/10.1080/23738871.2017.1295092`.

[76] Samuel Weiler, Anuja Sonalker, and Rob Austein. A Publication Protocol for the Resource Public Key Infrastructure (RPKI). RFC 8181, July 2017. URL `https://rfc-editor.org/rfc/rfc8181.txt`.

[77] Internet Engineering Task Force Public Notary Transparency Working Group, January 2018. URL `https://datatracker.ietf.org/wg/trans/about/`.

[78] Stephen Kent, Geoff Huston, and George G. Michaelson. Certification Authority (CA) Key Rollover in the Resource Public Key Infrastructure (RPKI). RFC 6489, February 2012. URL `https://rfc-editor.org/rfc/rfc6489.txt`.

[79] Geoff Huston. Bgp table, asn and cidr reports, October 2019. URL `http://bgp.potaroo.net/`.

[80] Ethereum Foundation. Proof of stake faq, October 2019. URL `https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ`.

[81] Jordi Paillisse, Miquel Ferriol, Eric Garcia, Hamid Latif, Carlos Piris, Albert Lopez, Brenden Kuerbis, Alberto Rodriguez-Natal, Vina Ermagan, Fabio Maino, and Albert Cabellos-Aparicio. Ipchain: Securing ip prefix allocation and delegation with blockchain. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1236–1243. IEEE, 2018. doi: 10.1109/Cybermatics_2018.2018.00218.

[82] Frank Denis. Free ip address to asn database, March 2021. URL `https://iptoasn.com/`.

[83] Center for Applied Internet Data Analysis. Inferred as to organization mapping dataset, March 2021. URL `https://www.caida.org/data/as-organizations/`.

[84] Lixin Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, Dec 2001. ISSN 1558-2566. doi: 10.1109/90.974527.

[85] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 618–627 vol.2, June 2002. doi: 10.1109/INFCOM.2002.1019307.

[86] Baruch Awerbuch and Christian Scheideler. Robust random number generation for peer-to-peer systems. In Mariam Momenzadeh Alexander A. Shvartsman, editor, *Principles of Distributed Systems*, pages 275–289, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-49991-6. doi: 10.1007/11945529_20. URL `https://doi.org/10.1007/11945529_20`.

[87] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, November 1979. ISSN 0001-0782. doi: 10.1145/359168.359176.

[88] Open Overlay Router Project. Ipchain: a blockchain-based mapping system, July 2019. URL `https://github.com/OpenOverlayRouter/blockchain-mapping-system`.

[89] Alberto Rodriguez-Natal, Jordi Paillisse, Florin Coras, Albert Lopez-Bresco, Lorand Jakab, Marc Portoles-Comeras, Preethi Natarajan, Vina Ermagan, David Meyer, Dino Farinacci, et al. Programmable overlays via openoverlayrouter. *IEEE Communications Magazine*, 55 (6):32–38, 2017. doi: 10.1109/MCOM.2017.1601056.

[90] Dino Farinacci, Vince Fuller, David Meyer, and Darrel Lewis. The Locator/ID Separation Protocol (LISP). RFC 6830, January 2013. URL `https://rfc-editor.org/rfc/rfc6830.txt`.

[91] Ethereum Foundation. Python implementation of the ethereum protocol, May 2019. URL `https://github.com/ethereum/pyethereum`.

[92] Twisted Matrix Labs. Twisted event-driven networking engine for python, Feb 2021. URL `https://twistedmatrix.com/trac/`.

[93] Marc Clifton. Python distributed hash table, March 2017. URL `https://github.com/cliftonm/kademlia-1`.

[94] Internet Assigned Numbers Authority. Iana number resources, March 2021. URL `https://www.iana.org/numbers`.

[95] Regional Internet Registries. Regional internet registries statistics information, March 2021. URL `https://www.nro.net/about/rirs/statistics/`.

[96] Geoff Huston. Bgp in 2016, January 2017. URL `http://www.potaroo.net/ispcol/2017-01/bgp2016.html`.

[97] Ahmed Elmokashfi, Amund Kvalbein, and Constantine Dovrolis. Bgp churn evolution: A perspective from the core. *IEEE/ACM Transactions on Networking*, 20(2):571–584, 2012. doi: 10.1109/TNET.2011.2168610.

[98] K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016. doi: 10.1109/ACCESS.2016.2566339.

[99] Adiseshu Hari and T V Lakshman. The Internet Blockchain : A Distributed , Tamper-Resistant Transaction Framework for the Internet. *Fifteenth ACM Workshop on Hot Topics in Networks*, pages 204–210, 2016. doi: 10.1145/3005745.3005771.

[100] Alfonso de la Rocha Gómez-Arevalillo and Panos Papadimitratos. Blockchain-based Public Key Infrastructure for Inter-Domain Secure Routing. In Jan Camenisch and Doğan Kesdoğan, editors, *International Workshop on Open Problems in Network Security (iNetSec)*, volume IFIP eCollection-1 of *Open Problems in Network Security*, pages 20–38, Rome, Italy, May 2017. URL `https://hal.inria.fr/hal-01684192`.

[101] N. Fotiou and G. C. Polyzos. Decentralized name-based security for content distribution using blockchains. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 415–420, April 2016. doi: 10.1109/INFCOMW.2016.7562112.

[102] Mennan Selimi, Aniruddh Rao Kabbinale, Anwaar Ali, Leandro Navarro, and Arjuna Sathiaseelan. Towards blockchain-enabled wireless mesh networks. *arXiv preprint arXiv:1804.00561*, 2018. URL `https://arxiv.org/abs/1804.00561`.

[103] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control. In Lydia Y. Chen and Hans P. Reiser, editors, *Distributed Applications and Interoperable Systems*, pages 206–220, Cham, 2017. Springer International Publishing. ISBN 978-3-319-59665-5. doi: 10.1007/978-3-319-59665-5_15. URL `https://doi.org/10.1007/978-3-319-59665-5_15`.

[104] True Names LTD. Ethereum name system (ens), March 2020. URL `https://ens.domains/`.

[105] S. Angieri, A. Garcia-Martinez, B. Liu, Z. Yan, C. Wang, and M. Bagnulo. A distributed autonomous organization for internet address management. *IEEE Transactions on Engineering Management*, pages 1–17, 2019. doi: 10.1109/TEM.2019.2924737.

[106] Ilias Sfirakis and Vasileios Kotronis. Validating ip prefixes and as-paths with blockchains. *arXiv preprint 1906.03172*, 2019. URL `https://arxiv.org/abs/1906.03172`.

[107] Qianqian Xing, Baosheng Wang, and Xiaofeng Wang. Bgpcoin: Blockchain-based internet number resource authority and bgp security solution. *Symmetry*, 10(9), 2018. ISSN 2073-8994. doi: 10.3390/sym10090408. URL `https://www.mdpi.com/2073-8994/10/9/408`.

[108] D. C. Verma. Simplifying network administration using policy-based management. *IEEE Network*, 16(2):20–26, March 2002. ISSN 0890-8044. doi: 10.1109/65.993219.

[109] OpenStack Foundation. Group-based policy for openstack whitepaper, November 2017. URL `https://wiki.openstack.org/w/images/a/aa/Group-BasedPolicyWhitePaper_v3.pdf`.

[110] Audun Jøsang. Pki trust models. *Theory and Practice of Cryptography Solutions for Secure Information Systems*, page 279, 2013. doi: 10.4018/978-1-4666-4030-6.ch012. URL `https://www.igi-global.com/chapter/content/76520`.

[111] A. Slagell, R. Bonilla, and W. Yurcik. A survey of pki components and scalability issues. In *2006 IEEE International Performance Computing and Communications Conference*, pages 10 pp.–484, April 2006. doi: 10.1109/.2006.1629442.

[112] Shengyun Liu, Paolo Viotti, Christian Cachin, Vivien Quéma, and Marko Vukolic. Xft: Practical fault tolerance beyond crashes. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 485–500, USA, 2016. USENIX Association. ISBN 9781931971331. URL `https://www.usenix.org/system/files/conference/osdi16/osdi16-liu.pdf`.

[113] A. Bessani, J. Sousa, and E. E. P. Alchieri. State machine replication for the masses with bft-smart. In *2014 44th Annual IEEE/IFIP Intl. Conf. on Dependable Systems and Networks*, pages 355–362, June 2014. doi: 10.1109/DSN.2014.43.

[114] Dino Farinacci and Brian Weis. Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality. RFC 8061, February 2017. URL `https://rfc-editor.org/rfc/rfc8061.txt`.

[115] Jordi Subira. Blockchain for distributed group based policies, June 2018. URL `https://github.com/JordiSubira/DGBP`.

[116] Linux Foundation Projects. Hyperledger fabric, June 2018. URL `https://www.hyperledger.org/projects/fabric`.

[117] Parth Thakkar, Senthil Nathan, and Balaji Vishwanathan. Performance benchmarking and optimizing hyperledger fabric blockchain platform. *arXiv preprint arXiv:1805.11390*, 2018. URL `https://arxiv.org/abs/1805.11390`.

[118] Sayed Hadi Hashemi, Faraz Faghri, and Roy H Campbell. Decentralized user-centric access control using pubsub over blockchain. *arXiv preprint arXiv:1710.00110*, 2017. URL `https://arxiv.org/abs/1710.00110`.

[119] S. Wang, Y. Zhang, and Y. Zhang. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access*, pages 1–1, 2018. doi: 10.1109/ACCESS.2018.2851611.

[120] I. Sukhodolskiy and S. Zapechnikov. A blockchain-based access control system for cloud storage. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 1575–1578, Jan 2018. doi: 10.1109/EIConRus.2018.8317400.

[121] O. Alphand, M. Amoretti, T. Claeys, S. Dall'Asta, A. Duda, G. Ferrari, F. Rousseau, B. Tourancheau, L. Veltri, and F. Zanichelli. Iotchain: A blockchain security architecture for the internet of things. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, April 2018. doi: 10.1109/WCNC.2018.8377385.

[122] O. J. A. Pinno, A. R. A. Gregio, and L. C. E. De Bona. Controlchain: Blockchain as a central enabler for access control authorizations in the iot. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Dec 2017. doi: 10.1109/GLOCOM.2017.8254521.

[123] Thanh Bui and Tuomas Aura. Application of public ledgers to revocation in distributed access control. *arXiv preprint arXiv:1608.06592*, 2016. URL `https://arxiv.org/abs/1608.06592`.

[124] It's FOSS. What is wireguard? why linux users going crazy over it?, February 2020. URL `https://itsfoss.com/wireguard/`.

[125] Linux's wireguard vpn is here and ready to protect you, March 2020. URL `https://www.zdnet.com/article/linuxs-wireguard-vpn-is-here-and-ready-to-protect-you/`.

[126] Benjamin Lipp, Bruno Blanchet, and Karthikeyan Bhargavan. A mechanised cryptographic proof of the wireguard virtual private network protocol. In *2019 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 231–246, 2019. doi: 10.1109/EuroSP.2019.00026.

[127] Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R Zimmermann. Post-quantum wireguard. (2020/379), 2020. URL `https://eprint.iacr.org/2020/379.pdf`.

[128] Yoav Nir and Adam Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC 8439, June 2018. URL `https://rfc-editor.org/rfc/rfc8439.txt`.

[129] Paul E. Hoffman and Jakob Schlyter. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698, August 2012. URL `https://rfc-editor.org/rfc/rfc6698.txt`.

[130] Paul Wouters. DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP. RFC 7929, August 2016. URL `https://rfc-editor.org/rfc/rfc7929.txt`.

[131] Dynamic multipoint vpn configuration guide, February 2020. URL `https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/sec_conn_dmvpn/configuration/xe-16/sec-conn-dmvpn-xe-16-book/sec-conn-dmvpn-dmvpn.html`.

[132] Sergey Gordeychik, Denis Kolegov, and Antony Nikolaev. Sd-wan internet census. *arXiv preprint 1808.09027*, 2018. URL `http://arxiv.org/abs/1808.09027`.

[133] Dino Farinacci, Fabio Maino, Vince Fuller, and Albert Cabellos-Aparicio. Locator/ID Separation Protocol (LISP) Control-Plane. Internet-Draft draft-ietf-lisp-rfc6833bis-27, Internet

Engineering Task Force, January 2020. URL https://datatracker.ietf.org/doc/html/draft-ietf-lisp-rfc6833bis-27. Work in Progress.

[134] Roy T. Fielding and Julian Reschke. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231, June 2014. URL https://rfc-editor.org/rfc/rfc7231.txt.

[135] Alexey Melnikov and Ian Fette. The WebSocket Protocol. RFC 6455, December 2011. URL https://rfc-editor.org/rfc/rfc6455.txt.

[136] grpc - a high-performance, open source universal rpc framework, October 2020. URL https://grpc.io/.

[137] Open overlay router project, April 2019. URL https://openoverlayrouter.org/.

[138] Dino Farinacci, David Meyer, and Job Snijders. LISP Canonical Address Format (LCAF). RFC 8060, February 2017. URL https://rfc-editor.org/rfc/rfc8060.txt.

[139] Grpc-c - c implementation of grpc, December 2020. URL https://github.com/lixiangyun/grpc-c.

[140] Ramakrishnan Durairajan, Sathiya Kumaran Mani, Joel Sommers, and Paul Barford. Time's forgotten: Using ntp to understand internet latency. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450340472. doi: 10.1145/2834050.2834108. URL https://doi.org/10.1145/2834050.2834108.

[141] Donald R. Morrison. Patricia—practical algorithm to retrieve information coded in alphanumeric. *J. ACM*, 15(4):514–534, October 1968. ISSN 0004-5411. doi: 10.1145/321479.321481. URL https://doi.org/10.1145/321479.321481.

[142] B Clifford Neuman and Theodore Ts'o. Kerberos: an authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, 1994. doi: 10.1109/35.312841.

[143] Dr. D. Hugh Redelmeier and Michael Richardson. Opportunistic Encryption using the Internet Key Exchange (IKE). RFC 4322, December 2005. URL https://rfc-editor.org/rfc/rfc4322.txt.

[144] Michael Rossberg and Guenter Schaefer. A survey on automatic configuration of virtual private networks. *Computer Networks*, 55(8):1684 – 1699, 2011. ISSN 1389-1286. doi: 10.1016/j.comnet.2011.01.003. URL http://www.sciencedirect.com/science/article/pii/S1389128611000053.

[145] David Carrel and Brian Weis. IPsec Key Exchange using a Controller. Internet-Draft draft-carrel-ipsecme-controller-ike-01, Internet Engineering Task Force, March 2019. URL https://datatracker.ietf.org/doc/html/draft-carrel-ipsecme-controller-ike-01. Work in Progress.

[146] Anna Selvåg Braadland. Key management for data plane encryption in sdn using wireguard. Master's thesis, NTNU, 2017. URL http://hdl.handle.net/11250/2457832.

[147] Karen Seo and Stephen Kent. Security Architecture for the Internet Protocol. RFC 4301, December 2005. URL https://rfc-editor.org/rfc/rfc4301.txt.

[148] Openvpn project, May 2020. URL https://openvpn.net/.

[149] Openvpn reference manual, February 2021. URL https://openvpn.net/community-resources/reference-manual-for-openvpn-2-4/.

[150] Pasi Eronen. IKEv2 Mobility and Multihoming Protocol (MOBIKE). RFC 4555, June 2006. URL https://rfc-editor.org/rfc/rfc4555.txt.

[151] Fabio Maino, Larry Kreeger, and Uri Elzur. Generic Protocol Extension for VXLAN. Internet-Draft draft-ietf-nvo3-vxlan-gpe-09, Internet Engineering Task Force, December 2019. URL https://datatracker.ietf.org/doc/html/draft-ietf-nvo3-vxlan-gpe-09. Work in Progress.

[152] Jesse Gross, Ilango Ganga, and T. Sridhar. Geneve: Generic Network Virtualization Encapsulation. Internet-Draft draft-ietf-nvo3-geneve-16, Internet Engineering Task Force, March 2020. URL https://datatracker.ietf.org/doc/html/draft-ietf-nvo3-geneve-16. Work in Progress.

[153] Ila kernel documents, May 2020. URL https://www.kernel.org/doc/Documentation/networking/ila.txt.

[154] Cisco. Campus lan and wireless lan solution design guide, May 2020. URL https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Campus/cisco-campus-lan-wlan-design-guide.html.

[155] Juniper Networks. Understanding the design of the midsize enterprise campus solution, November 2020. URL https://www.juniper.net/documentation/en_US/release-independent/nce/topics/concept/cs1-design-understanding.html.

[156] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: Taking control of the enterprise. volume 37, page 1–12, New York, NY, USA, August 2007. Association for Computing Machinery. doi: 10.1145/1282427.1282382. URL https://doi.org/10.1145/1282427.1282382.

[157] David Lebrun, Mathieu Jadin, François Clad, Clarence Filsfils, and Olivier Bonaventure. Software resolved networks: Rethinking enterprise networks with ipv6 segment routing. In *Proceedings of the Symposium on SDN Research*, SOSR '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356640. doi: 10.1145/3185467.3185471. URL https://doi.org/10.1145/3185467.3185471.

[158] Bruce Davie, Teemu Koponen, Justin Pettit, Ben Pfaff, Martin Casado, Natasha Gude, Amar Padmanabhan, Tim Petty, Kenneth Duda, and Anupam Chanda. A database approach to sdn control plane design. *SIGCOMM Comput. Commun. Rev.*, 47(1):15–26, January 2017. ISSN 0146-4833. doi: 10.1145/3041027.3041030. URL `https://doi.org/10.1145/3041027.3041030`.

[159] Ayoub Bahnasse, Mohamed Talea, Abdelmajid Badri, Fatima Ezzahraa Louhab, and Sara Laafar. Smart hybrid sdn approach for mpls vpn management on digital environment. *Telecommunication Systems*, 73(2):155–169, 2020. doi: 10.1007/s11235-019-00603-6. URL `https://doi.org/10.1007/s11235-019-00603-6`.

[160] A. Boukerche, R.W.N. Pazzi, and R.B. Araujo. Hpeq a hierarchical periodic, event-driven and query-based wireless sensor network protocol. In *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*, pages 560–567, 2005. doi: 10.1109/LCN.2005.75.

[161] C. Mbarushimana and A. Shahrabi. Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks. In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, volume 2, pages 679–684, 2007. doi: 10.1109/AINAW.2007.123.

[162] Charles E. Perkins. IP Mobility Support for IPv4. RFC 3344, August 2002. URL `https://rfc-editor.org/rfc/rfc3344.txt`.

[163] H. Badis and K. A. Agha. An efficient mobility management in wireless overlay networks. In *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003.*, volume 3, pages 2500–2504 vol.3, 2003. doi: 10.1109/PIMRC.2003.1259173.

[164] Ieee standard for local and metropolitan area network–bridges and bridged networks. *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pages 1–1993, July 2018. ISSN null. doi: 10.1109/IEEESTD.2018.8403927.

[165] Yakov Rekhter and Eric C. Rosen. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, February 2006. URL `https://rfc-editor.org/rfc/rfc4364.txt`.

[166] Yukihiro Nakagawa, Kazuki Hyoudou, and Takeshi Shimizu. A management method of ip multicast in overlay networks using openflow. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, HotSDN '12, page 91–96, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314770. doi: 10.1145/2342441.2342460. URL `https://doi.org/10.1145/2342441.2342460`.

[167] Bingchuan Tian, Xinyi Zhang, Ennan Zhai, Hongqiang Harry Liu, Qiaobo Ye, Chunsheng Wang, Xin Wu, Zhiming Ji, Yihong Sang, Ming Zhang, and et al. Safely and automatically updating in-network acl configurations with intent language. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM 2019, pages 214–226, New York,

NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359566. doi: 10.1145/3341302.3342088. URL https://doi.org/10.1145/3341302.3342088.

[168] Gary N Stone, Bert Lundy, and Geoffrey G Xie. Network policy languages: a survey and a new approach. *IEEE network*, 15(1):10–21, 2001. doi: 10.1109/65.898818.

[169] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. Nox: Towards an operating system for networks. *SIGCOMM Comput. Commun. Rev.*, 38(3):105–110, July 2008. ISSN 0146-4833. doi: 10.1145/1384609.1384625. URL https://doi.org/10.1145/1384609.1384625.

[170] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, and Scott Shenker. Onix: A distributed control platform for large-scale production networks. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, page 351–364, USA, 2010. USENIX Association. URL https://static.usenix.org/events/osdi10/tech/full_papers/Koponen.pdf.

[171] Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, and et al. Onos: Towards an open, distributed sdn os. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 1–6, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329897. doi: 10.1145/2620728.2620744. URL https://doi.org/10.1145/2620728.2620744.

[172] Allan Rubens, Carl Rigney, Steve Willens, and William A. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000. URL https://rfc-editor.org/rfc/rfc2865.txt.

[173] Michael Smith, Rakesh Reddy Kandula, and Syam Appala. Scalable-Group Tag eXchange Protocol (SXP). Internet-Draft draft-smith-kandula-sxp-10, Internet Engineering Task Force, May 2020. URL https://datatracker.ietf.org/doc/html/draft-smith-kandula-sxp-10. Work in Progress.

[174] Alberto Rodriguez-Natal, Vina Ermagan, Johnson Leong, Fabio Maino, Albert Cabellos-Aparicio, Sharon Barkai, Dino Farinacci, Mohamed Boucadair, Christian Jacquenet, and Stefano Secci. Publish/Subscribe Functionality for LISP. Internet-Draft draft-ietf-lisp-pubsub-05, Internet Engineering Task Force, March 2020. URL https://datatracker.ietf.org/doc/html/draft-ietf-lisp-pubsub-05. Work in Progress.

[175] Michael Smith and Larry Kreeger. VXLAN Group Policy Option. Internet-Draft draft-smith-vxlan-group-policy-05, Internet Engineering Task Force, October 2018. URL https://datatracker.ietf.org/doc/html/draft-smith-vxlan-group-policy-05. Work in Progress.

[176] Ieee standard for local and metropolitan area networks-media access control (mac) security. *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pages 1–239, Dec 2018. ISSN null. doi: 10.1109/IEEESTD.2018.8585421.

[177] Christian Hopps and Dave Thaler. Multipath Issues in Unicast and Multicast Next-Hop Selection. RFC 2991, November 2000. URL `https://rfc-editor.org/rfc/rfc2991.txt`.

[178] Donald R. Morrison. Patricia-practical algorithm to retrieve information coded in alphanumeric. *J. ACM*, 15(4):514–534, October 1968. ISSN 0004-5411. doi: 10.1145/321479.321481. URL `https://doi.org/10.1145/321479.321481`.

[179] Martin Casado, Tal Garfinkel, Aditya Akella, Michael J. Freedman, Dan Boneh, and Nick McKeown. SANE: A protection architecture for enterprise networks. In *15th USENIX Security Symposium (USENIX Security 06)*, Vancouver, B.C. Canada, July 2006. USENIX Association. URL `https://www.usenix.org/conference/15th-usenix-security-symposium/sane-protection-architecture-enterprise-networks`.

[180] Yu-Wei Eric Sung, Xin Sun, Sanjay G. Rao, Geoffrey G. Xie, and David A. Maltz. Towards systematic design of enterprise networks. *IEEE/ACM Transactions on Networking*, 19(3): 695–708, 2011. doi: 10.1109/TNET.2010.2089640.

[181] Dan Levin, Marco Canini, Stefan Schmid, Fabian Schaffert, and Anja Feldmann. Panopticon: Reaping the benefits of incremental SDN deployment in enterprise networks. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 333–345, Philadelphia, PA, June 2014. USENIX Association. ISBN 978-1-931971-10-2. URL `https://www.usenix.org/conference/atc14/technical-sessions/presentation/levin`.

[182] N. Khademi, D. Ros, M. Welzl, Z. Bozakov, A. Brunstrom, G. Fairhurst, K. Grinnemo, D. Hayes, P. Hurtig, T. Jones, S. Mangiante, M. Tuxen, and F. Weinrank. Neat: A platform- and protocol-independent internet transport api. *IEEE Communications Magazine*, 55(6): 46–54, 2017. doi: 10.1109/MCOM.2017.1601052.