Improving Throughput, Latency and Privacy with Hybrid Networks and Multipath Routing

THÈSE N^{O} 9056 (2018)

PRÉSENTÉE LE 23 NOVEMBRE 2018

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 3
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Sébastien Christophe HENRI

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Acknowledgements

First and foremost, I would like to thank my advisor, Patrick Thiran, for his help, support and guidance during the five years of my PhD. I could not have dreamt of a better working environment. Patrick enabled me to have everything that could be useful for my research, and he offered me the flexibility that I needed to balance my professional and personal life; for this in particular, I am extremely grateful. In addition to being a great advisor, Patrick is one of the most fair and honest people I know, and it has been a privilege to have the opportunity to work with him.

I would also like to thank Katerina Argyraki, Olivier Bonaventure, Jean-Yves Le Boudec and Thrasyvoulos Spyropoulos for accepting to be members of my committee, for taking the time to read this dissertation and for their precious feedback.

I would like to thank the support team of the lab: Angela, Danielle, Patricia, and Simone, for helping me with so many things, from finding an apartment to organizing trips, drinks and outings for the team (I hope to be able to attend some again soon!). Marc-André and Yves, for their help in setting up all the infrastructure and the equipment that I needed for my research; a lot of what I have done would not have been possible without them. Holly, for her tireless efforts to correct my texts and to make my English better; I learned a lot, joyfully, and I take her lessons with me!

I was very fortunate to work and collaborate with people who were both bright and extremely generous, and I would like to express my gratitude to Albert Banchs, Ginés García Avilés, Julien Herzen, Pablo Serrano, Seva Shneer and Christina Vlachou. I am extremely grateful in particular to Christina and Julien for helping me when I arrived in the lab, and for continuing to help me after their PhD work was completed. It has been a true pleasure to work so closely with them and to be able to work in such a positive environment. They are competent and trustful, they helped me a lot with my work, but above all, they are great people and have made work easy and fun. I want to thank particularly Albert as well for his bright and insightful ideas — I am lucky to have had the opportunity to work with him. I would also like to thank all the members of the LCA3.5/Indy lab (Victor, Mohamed, Vincent, Lucas, William, Brunella, Farnood, Arnout, Greg, Daniyar, Mladen, Mahsa, Aswin, Ehsan, Lyudmila, Young-Jun, Farid, Runwei, Elisa and Matthias) for the feedback they gave me on my projects, and for all the lunches, coffee breaks, dinners and ski days we shared.

I am very thankful to my family for their support, before and during this PhD adventure. They played a very important part in making me who I am today. During my five years in Lausanne, I also met great friends who made my life better, even during the difficult moments. For everything they did, in my personal life but also professionally, I want to deeply thank Kévin, Mathias, Sam

Oun, Aude, Nathan and Lucas. Finally, nothing would have been possible without the support, each day renewed, of Julie: I know what I owe you.

Abstract

Over the last few years, residential and enterprise networking have faced several challenges due to the increasing demand of users for high-throughput connectivity. As a result, efforts are being made to improve coverage, throughput, and robustness. Several solutions have been recently proposed. The first solution is to use mesh networking; it is gaining momentum, as it effectively improves performance, but at the cost of an increased complexity compared to the infrastructure mode, as several paths can now be employed with potentially several hops. The second solution is to exploit the different technologies that are available, wired (e.g., power-line communication (PLC) or Ethernet) and wireless (e.g., WiFi or cellular). Networks with various technologies are referred to as hybrid networks. When the technologies do not interfere with each other, it is possible to aggregate their capacity, thus enabling immediate throughput improvements; by increasing the number of possible paths that a packet can take, hybrid networks also increase complexity. The third solution is to use multipath routing, which can improve performance significantly. But again, this comes at the cost of an increased complexity.

In this dissertation, we study the effect of these solutions in terms of throughput and coverage, latency, and privacy. We focus, in particular, on hybrid networks with *shared-medium* and *orthogonal* technologies, where two links that use the same technology are subject to interference (shared-medium), but not two links that use two distinct technologies (orthogonal).

First, we study the effect of these solutions on throughput and coverage. We show that, in hybrid mesh networks, the optimal number of paths achieving maximal throughput with multipath routing is tightly linked with the number of technologies. This result makes it possible to develop an efficient and practical multipath routing protocol that yields the maximal throughput. Next, we introduce two novel algorithms for optimizing throughput: A distributed multipath congestion controller that, when each flow uses one multipath fixed in advance, provably achieves optimal throughput and an algorithm based on the multi-armed-bandit framework that finds the best multipath and converges to the best achievable throughput. We implement these algorithms in a real testbed with PLC and two orthogonal WiFi channels. Their experimental evaluation shows that using technologies with distinct physical layers, such as PLC and WiFi, improves spatial diversity compared to using multi-channel WiFi and brings further improvements of throughput and coverage.

Then, we investigate latency in hybrid networks. We study analytically how the variance of a time-varying service rate affects queueing delays. We also study latency when multipath routing is used, i.e., when traffic is split between two technologies. We show that finding the optimal splitting scheme is difficult, as it depends on the rate at which packets arrive, and that the

best *static* scheme, where the splitting probability remains the same for all arrival rates, can be significantly sub-optimal in time-varying networks.
Finally, we study how hybrid networks and multipath can improve privacy. We show that they can significantly improve the resistance against traffic analysis attacks, such as website fingerprinting, by enabling the user to split the traffic between two networks.

Keywords: Hybrid networks, multipath routing, power-line communications, wireless technologies, mesh networks, performance analysis, testbed experiments, privacy, website fingerprinting attacks.

Résumé

Au cours de ces dernières années, les réseaux résidentiels et d'entreprise ont eu à faire face à plusieurs défis découlant de la demande toujours plus forte des utilisateurs pour une connectivité à très haut débit. En conséquence, d'importants efforts sont entrepris pour améliorer la couverture, le débit et la fiabilité de ces réseaux. Plusieurs solutions ont été récemment proposées. La première est d'utiliser des réseaux maillés, avec plusieurs points d'accès au réseau au lieu d'un seul. Cette solution gagne du terrain parce qu'elle peut efficacement améliorer les performances des réseaux. Mais cela se fait au prix d'une complexité accrue par rapport au mode infrastructure, car plusieurs chemins sont dorénavant disponibles avec, possiblement, plusieurs sauts. La deuxième solution est d'exploiter les différentes technologies disponibles, qu'elles soient filaires (par exemple, les courants porteurs en ligne (CPL) ou Ethernet) ou sans-fil (par exemple, WiFi ou le réseau de téléphonie mobile). Ces réseaux avec plusieurs technologies sont appelés réseaux hybrides. Quand les technologies n'interfèrent pas les unes avec les autres, leurs capacités peuvent être agrégées, ce qui permet d'augmenter directement le débit disponible. Mais là encore, l'augmentation du nombre de chemins que le trafic peut emprunter et le choix à faire entre les technologies rendent les choses plus complexes. Une troisième solution est, enfin, d'utiliser plusieurs chemins en même temps. Elle peut entraîner des gains de performance importants, mais, à nouveau, au prix d'une complexité accrue.

Dans cette thèse, nous étudions l'effet de ces solutions en termes de débit et de couverture du réseau, de latence et de vie privée. Nous nous concentrons en particulier sur les réseaux hybrides constitués de technologies *orthogonales* et à *ressources partagées*, pour lesquelles deux liens utilisant la même technologie interfèrent l'un avec l'autre (technologies à ressources partagées), mais pas deux liens utilisant deux technologies différentes (technologies orthogonales).

Nous étudions tout d'abord leur effet sur le débit et la couverture du réseau. Nous montrons d'abord que le nombre optimal de chemins nécessaires pour atteindre le débit maximal, lorsque le routage multichemin est utilisé dans un réseau maillé hybride, est intrinsèquement lié au nombres de technologies orthogonales. Grâce à ce résultat, nous développons un protocole de routage multichemin efficace et pratique. Nous présentons ensuite deux algorithmes novateurs pour optimiser le débit. Nous décrivons tout d'abord un algorithme distribué et multichemin de contrôle de la congestion et prouvons qu'il atteint un débit optimal en présence de plusieurs utilisateurs, lorsque chaque flux de données utilise un seul multichemin choisi en avance. Ensuite, nous présentons un second algorithme, fondé sur le paradigme du bandit manchot, qui trouve le meilleur multichemin et converge vers le débit maximal. Ces algorithmes sont mis en œuvre dans un réseau réel avec CPL et deux canaux WiFi orthogonaux. Leur évaluation expérimentale montre qu'utiliser deux technologies avec deux couches physiques différentes, comme les CPL et le WiFi, améliore la diversité spatiale et, en conséquence, le débit et la couverture, par rapport à l'utilisation de deux canaux WiFi orthogonaux.

Nous nous intéressons ensuite à la latence dans les réseaux hybrides. Nous étudions un modèle de file d'attente avec des taux de service variant au cours du temps, et en particulier l'effet de la variance de ce taux de service sur les temps d'attente. Nous étudions ensuite la latence lorsque le routage multichemin est utilisé dans un réseau hybride et le trafic divisé entre deux technologies. Nous démontrons que trouver le mécanisme de division optimal est difficile car celui-ci dépend du taux d'arrivée des paquets. Nous démontrons également que le meilleur mécanisme *statique* (c'est-à-dire ne changeant pas en fonction du taux d'arrivée des paquets) peut être significativement sous-optimal dans un réseau qui varie au cours du temps.

Enfin, nous examinons comment les réseaux hybrides et le routage multichemin peuvent améliorer la vie privée des utilisateurs. Nous montrons que ces solutions, en permettant de diviser le trafic entre deux réseaux différents, peuvent améliorer fortement la protection contre des attaques d'identification des sites web fondée sur leur signature.

Mots-clés : Réseaux hybrides, routage multichemin, courants porteurs en ligne, technologies sans-fil, réseaux maillés, analyses théoriques de performance, analyses expérimentales, vie privée.

Contents

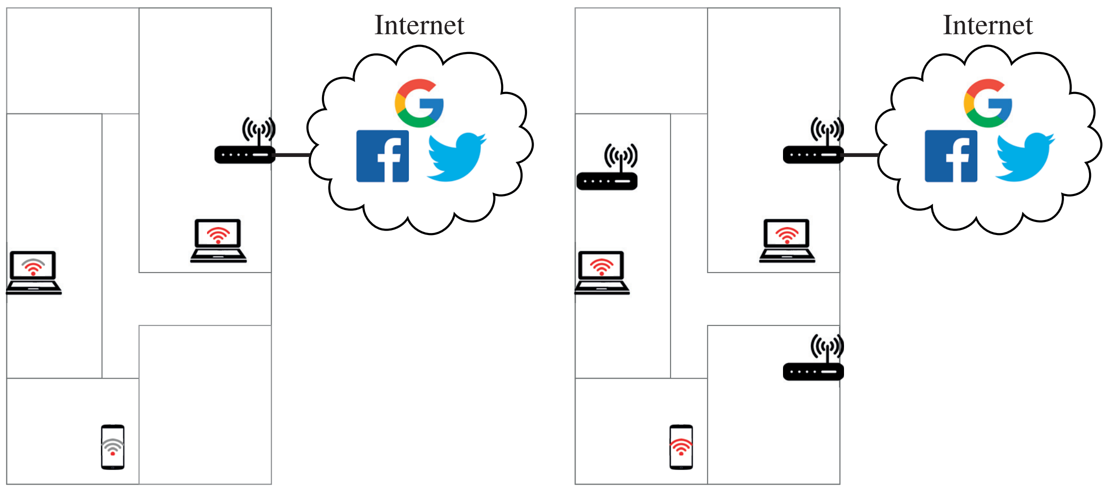
1 Introduction

1.1 Mesh Networks, Hybrid Networks and Multipath Routing: Solutions to Connectivity Problems and Increased Performance Demand

Recent years have seen the development, in residential and entreprise networks, of a multitude of new applications that require an always-increasing demand for high performance. High-definition streaming requires high data-rates; gaming and voice-over-IP require low latency and high reliability; and smart-home and smart-city solutions require ubiquitous network coverage for an increasing number of devices, to name only a few of these applications. Also, the recent revelations about the large-scale surveillance of people's web activities have highlighted the privacy risks caused by these new usages (social networks, etc.).

Because they offer mobility along with attractive data-rates, wireless technologies, in particular WiFi that is specified by the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards, are used in the vast majority of the local networks (local networks refer to the networks that lie within a limited area, such as residential, enterprise and university networks). But WiFi suffers from the fast attenuation of its signal, in particular due to walls. This leaves large zones of the residential or enterprise buildings with a weak signal, which results in low throughput or no connectivity. A recent study [wif15] indicates that nearly 40% of the American households with a WiFi router suffer from such problems. This study also indicates that these problems are exacerbated when the number of connected devices is large. This stems from the very nature of wireless technologies that share the medium hence are prone to interference when a large number of devices lies within a limited area. With the explosion of the number of connected devices that comes with the Internet of things (IoT), such issues are likely to become even more problematic in a close future.

Over the last few years, significant efforts have been made to improve the throughput offered by wireless technologies. New WiFi standards (such as 802.11n [wif09] and 802.11ac [wif13]) have been proposed and another (802.11ax [wif18]) is currently under discussion. These standards



(a) Infrastructure mode of WiFi (b) Mesh WiFi network

Figure 1.1 – Typical residential WiFi network in the infrastructure mode (left) and with a mesh network (right).

offer high maximal data-rates (theoretical physical rates of up to 1.2 Gb/s for 802.11ac and 802.11ax) and a better efficiency with several users, but they continue to suffer from the inherent signal attenuation and the high level of interference when the number of devices is large. For this reason, other directions for improving the performance of local networks have been pursued.

1.1.1 Mesh Networks

Weak signals typically occur when the user is too far from the wireless access-point. Today, most 802.11 WiFi networks — especially in homes — use the infrastructure mode, in which there is a single access-point to which all users' devices connect. When there is a single access-point, it is often impossible, particularly in large houses, to find a spot where the entire house is covered by a strong wireless signal. Even worse, the location of the access point is usually determined by the location of the wall socket for the Internet access (fiber optics receptacle, telephone plug, etc.); this wall socket is rarely chosen to maximize coverage. This can cause some user's devices to have a weak signal or no signal at all, which means low throughput or no connectivity: This is for example the case of the smartphone in the residential infrastructure WiFi network depicted in Figure 1.1a.

For this reason, mesh networks have recently gained a renewed attention as a way to improve throughput and coverage. In a wireless mesh network, several routers are connected together through the wireless medium in an ad-hoc manner. A user's device can associate with any of the different routers, typically the one with which it has the highest signal; if the user moves, her device might associate with a different router. For example, in Figure 1.1b that depicts the same network as Figure 1.1a but in mesh mode, the smartphone communicates with the closest router and gets a strong signal, i.e., it gets a high throughput. To reach another of the routers (typically, the Internet gateway), the traffic is routed through the network of routers, which might require several hops (for the smartphone of Figure 1.1b, this requires two hops).

Mesh networks are becoming increasingly popular [Che17] and a large number of commercial solutions has been launched onto the market [Del18]. They are used for improving high-rate local networks and are also gaining momentum with the development of smart homes and smart cities, in which a large number of devices need to be connected to a same network. Many smart-home and smart-city solutions use mesh networks [Ros18]. Mesh networks can effectively improve performance, but this comes at the cost of an increased complexity compared to the infrastructure mode, because several paths can now be employed with potentially several hops.

1.1.2 Hybrid Networks

To improve the performance of local networks, it is possible to combine WiFi with other technologies. Networks where several technologies cohabit are called *hybrid* networks. When the technologies do not interfere with each other, it is possible to aggregate their capacity, thus enabling immediate performance improvements. Different technologies can be used to form hybrid networks.

Possible Technologies

Ethernet Ethernet is extensively used in enterprise networks along with WiFi: Ethernet wires are used to connect the fixed devices (e.g., desktop computers, TVs) and to provide a backbone for the wireless routers; WiFi is used to connect mobile devices to the enterprise network. This solution is very effective, as Ethernet wires offer very high rates (the latest Ethernet standard offer theoretical physical rates of up to 400 Gb/s) and are not subject to interference. Its main downside is that Ethernet wires are complex and expensive to install, which makes this solution often impossible to employ.

Technologies other than Ethernet are easier to employ because they do not require any additional wiring. They can be wireless technologies (such as Bluetooth and cellular) or wired technologies that exploit the existing infrastructure and do not require new wiring (such as coaxial and power-line).

WiFi WiFi offers several *orthogonal* channels, i.e., channels with frequencies that do not overlap hence do not interfere with each other. Consequently, it is possible to improve the performance of local networks by using two orthogonal WiFi channels, which is considered as a particular case of hybrid networks.

Bluetooth Bluetooth uses the same frequency band as WiFi (around 2.4 GHz). Consequently, it interferes with WiFi, suffers from the same attenuation problems, and offers data rates well below those of WiFi. Bluetooth typically targets low-rate, low-power applications, whereas WiFi targets high-rate applications.

Cellular Cellular technologies use radio frequencies different from those used by WiFi, hence they do not interfere with WiFi. Cellular technologies are supported by every smartphone, and all laptops can easily connect to a cellular network by the addition of a lightweight and low-cost component (e.g., 3G/4G USB dongles) or with a smartphone sharing its connection through Bluetooth or USB. Cellular is considered as a good candidate for handover when the user moves out of WiFi coverage [KTZ04, PDD$^+$12]. Recently, small cells, in particular femtocells, have gained popularity as a solution to improve the coverage of WiFi in local networks; they gave rise to hybrid cellular/WiFi networks [BSC$^+$13].

Coaxial Coaxial communication exploits the coaxial cables that are already installed in a large number of houses (for example, 90% of the households in the USA have coaxial cables installed [coa13]). It is standardized by the Multimedia over Coax Alliance (MoCA) [MoC] and offers theoretical physical rates of up to 1.4 Gb/s. Coaxial wires are shielded hence not subject to external interference, in particular from neighboring households or from other technologies. The main limitation to coaxial communication is that it does not offer much flexibility, because its performance, especially in terms of coverage, depends on the coaxial system that is already established, in particular on the number of coaxial outlets.

Power-line Power-line communication (PLC) exploits the electrical wires that are in all buildings, which means that PLC does not require any new wiring. It is standardized by the HomePlug alliance [Homa] and offers theoretical physical rates of up to 1.5 Gb/s with the latest standard. It offers a coverage typically wider than that of WiFi and is very easy to install, because a PLC device simply needs to be plugged on an electrical outlet (*plug and play*). PLC is becoming very popular, especially in home networks, and 220 million devices have been sold worldwide [Homb]. PLC presents some downsides. Similarly as WiFi, PLC is shared-medium, i.e., it is subject to interference, and having a large number of connected devices affects the performance (PLC, as standardized by HomePlug, employs a CSMA/CA scheme relatively similar to that of WiFi [VHT13]). Also, the PLC performance varies depending on the wiring quality and on the structure of the electrical network. Nevertheless, PLC has been shown to be a good candidate for improving the performance of WiFi [Vla16].

Technologies Considered in this Dissertation To be able to aggregate the capacities that they offer, the technologies need to be *orthogonal*, i.e., two links that use two distinct technologies are not subject to interference (for example, Bluetooth and WiFi are not orthogonal when they use the same frequency band). Here, we focus on technologies that, in addition to being orthogonal, are *shared-medium*, i.e., where two links that use the same technology are subject to interference. We consider in particular WiFi (single-channel and multi-channel), PLC and cellular. The results presented in this dissertation can easily be extended to non-shared-medium technologies by considering that each link is a single technology (e.g., two Ethernet links, which do not interfere

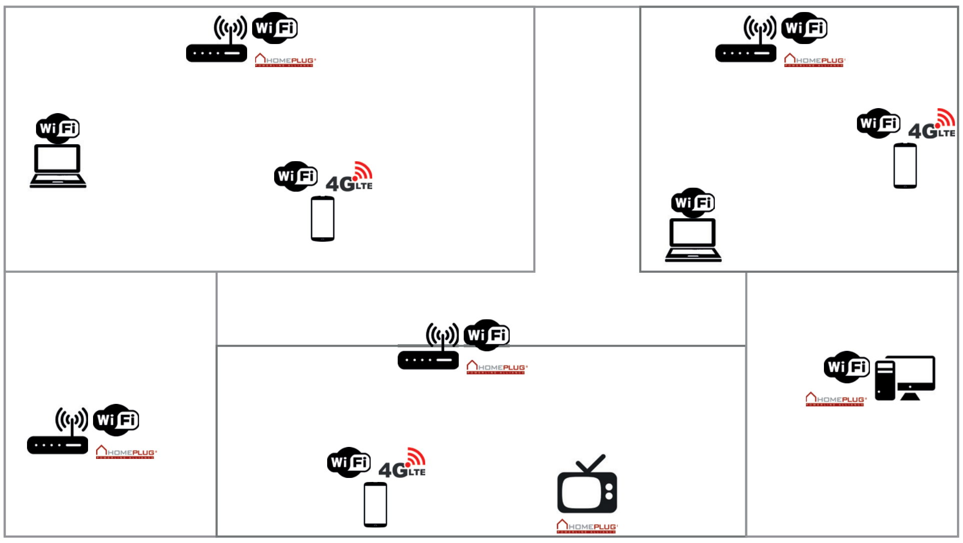


Figure 1.2 – Example of a residential hybrid mesh network with hybrid WiFi/PLC routers, a desktop computer with WiFi and PLC, a TV connected with PLC, laptops with WiFi, and smartphones with WiFi and cellular.

with each other, are considered as two different technologies). An example of a residential hybrid mesh network is presented in Figure 1.2.

Standardization of Hybrid Networks

The increasing popularity of hybrid networks is illustrated by the standardization efforts made by the IEEE. The IEEE 1905 standard [Hyb13] specifies abstraction layers to hide the diversity of technologies in a hybrid network (wireless, coaxial, power-line, Ethernet), which enables a seamless interoperability between the technologies. According to the IEEE 1905 standard, hybrid networks operate at layer 2.5, between the MAC and IP layers. The IEEE 1905 standard specifies link metrics for hybrid networks, but it does not specify routing or load-balancing algorithms that are vendor specific. Many hybrid devices compatible with the IEEE 1905 standard have been launched onto the market, such as hybrid PLC/WiFi or coaxial/WiFi range extenders that simply act as a bridge between the two technologies and hybrid routers that are able to choose their next hop; recently, hybrid PLC/WiFi routers that work in a mesh network have been announced [hyb18].

1.1.3 Multipath Routing

To improve throughput, reliability and latency, it is possible to send traffic onto several paths, as illustrated by Figure 1.3: To reach the laptop (C), the hybrid router (A) can use the WiFi-WiFi path (Path 2) simultaneously with the PLC-WiFi path (Path 1). Multipath routing has recently gained popularity with the emergence of multipath TCP (MPTCP) [FRH$^+$11, FRHB13]. Multipath

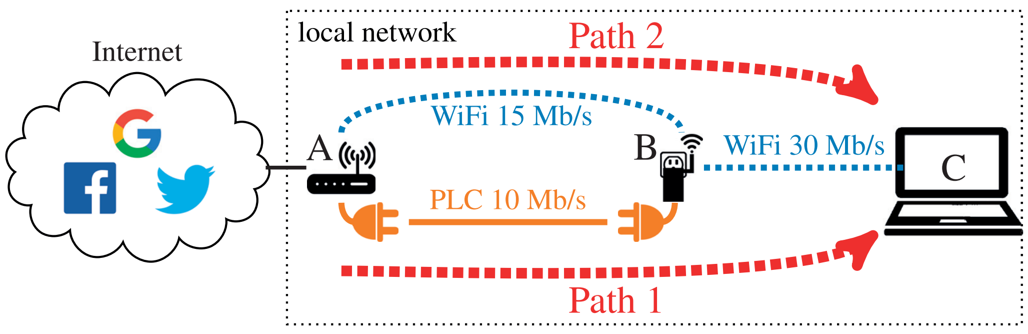


Figure 1.3 – Scenario with a PLC/WiFi router (A), a PLC/ WiFi range extender (B), and a WiFi user (C).

solutions that support UDP traffic are also being developed [DCB17]. In single-hop hybrid networks with WiFi and cellular, MPTCP has been shown to improve throughput [RPB+12, CLG+13], reliability [LAPJ15], and latency [FEB+16]. MPTCP has been adopted by major vendors: In particular, all Apple operating systems (for smartphones and computers) now support MPTCP and an MPTCP implementation exists for all Linux-based devices [PB].

1.1.4 Improved Performance and New Challenges

Mesh networks, hybrid networks, and multipath routing can effectively improve the performance (throughput, coverage, latency, users' privacy) of local networks. Mesh networks can extend the coverage and improve throughput (as we show in Part I of this dissertation). By aggregating the capacities of different technologies, hybrid networks can improve coverage and throughput (Part I), latency (Part II), and privacy (Part III). By simultaneously exploiting multiple paths, multipath routing can improve throughput (Part I), latency (Part II), and privacy (Part III).

They also increase, however, the complexity of the algorithms that need to be deployed in local networks: (*i*) In *mesh networks*, paths are multi-hop and several possible paths between two nodes exist; (*ii*) *hybrid networks* increase the number of links, because two nodes can now communicate with several technologies, hence they increase the number of possible paths between two nodes; and (*iii*) *multipath routing* makes it possible to use several paths, giving a number of possibilities that is polynomial in the number of existing paths. For this reason, specific algorithms are required. Because we consider shared-medium technologies, these algorithms must also take into account interference, which is another challenge.

In this dissertation, we study these challenges and tackle them by introducing several algorithms to improve throughput (Part I), latency (Part II) and privacy (Part III) with hybrid networks and multipath routing. We also present fundamental theoretical and experimental results on throughput (Part I) and latency (Part II). We consider both local mesh networks (Part I) and local single-hop networks (Part II and III).

1.2 Dissertation Outline and Contributions

We study three main performance metrics: throughput, latency, and privacy. Consequently, this dissertation is divided into three parts, each corresponds to one performance metric.

Part I: Throughput We study throughput in hybrid mesh networks. This part is divided into three chapters.

- **Chapter 2**: We study multipath routing in hybrid mesh networks. After presenting our network model, we study the optimal number of paths needed to reach the maximal throughput in a hybrid mesh network. For certain classes of networks, we show analytically that there is a tight relation between the optimal number of paths and the number of orthogonal technologies. Through simulations and experiments, we verify this result and extend it to general networks. We then describe a multipath-routing protocol that uses heuristics to compute the set of paths (called a *multipath*) that offers the highest throughput.

  Main contribution: To the best of our knowledge, we are the first to show analytically a relation between the optimal number of paths and the number of technologies in a hybrid mesh network, which shows that multipath and hybrid networks are intrinsically related. We propose a novel multipath routing-protocol that focuses on optimizing throughput.

- **Chapter 3**: We describe a novel multipath congestion-control algorithm that finds, in a hybrid mesh network, the optimal amount of traffic to send on each path of a pre-computed multipath and avoids congestion. This congestion controller is interference-aware and distributed. We show that it maximizes a global utility function in the presence of multiple users. We describe the implementation of EMPoWER, the system that combines this congestion-controller and the multipath-routing protocol presented in Chapter 2. We extensively evaluate EMPoWER with simulations and testbed experiments in a hybrid multi-channel WiFi network and in a hybrid PLC/WiFi network.

  Main contribution: We show experimentally that, despite similar aggregate capacities, hybrid PLC/WiFi networks improve throughput compared to hybrid multi-channel WiFi networks because, for shared-medium technologies, using two distinct technologies brings an increased spatial diversity.

- **Chapter 4**: We describe HyMAB, an algorithm that finds the best multipath in a dynamic hybrid mesh network and converges to the maximal throughput. It is based on the multi-armed-bandit framework. HyMAB continuously adapts when the network conditions change (e.g., due to mobility or capacity variations) and it is able to smoothly perform a handover to another multipath, if needed. HyMAB is implemented and extensively evaluated in a hybrid PLC/WiFi testbed.

  Main contribution: To the best of our knowledge, HyMAB is the first implementation of a multi-armed-bandit strategy in the context of routing and congestion control.

Part II: Latency We study latency in time-varying hybrid networks. This part contains one chapter.

- **Chapter 5**: We study, in hybrid networks, the effect of variations of the link capacities (e.g., due to variations of the signal or users contending for a network medium) on latency. We apply a queueing model where the service rate varies between two states, a high-rate state and a low-rate state. We first consider the case where the user employs single-path, i.e., she chooses between two technologies that have the same average service rate but different variances of the service rate. We show analytically and verify experimentally that, as can be expected when the average service rates are equal, the variance is an important quantity, but that, surprisingly, the technology with the largest variance can sometimes yield the smallest delays. We then consider the case where the user employs multipath, i.e., she splits the traffic between two technologies with different service-rate characteristics (potentially different means and variances). We show analytically and verify experimentally that when the traffic arrives at a low rate, it is better to use a single technology, and that multipath improves latency only when the arrival rate is larger than a certain threshold. We show that finding the optimal splitting scheme is not an easy task and that the impact of the splitting scheme is high in time-varying networks.

  Main contribution: We show that when the average service rates of two technologies are equal, the technology with the largest variance can sometimes yield the smallest delays. We show that multipath improves latency only for sufficiently high rates.

Part III: Privacy We study how the users' privacy can be improved with hybrid networks and multipath. This part contains one chapter.

- **Chapter 6**: We study the so-called *website fingerprinting* attacks, where an adversary guesses which website a user visits, even when the user employs encryption and anonymous communication tools. When the user is connected to two networks (e.g., when she uses hybrid networks and is connected to the Internet with two technologies) and splits her traffic between the two networks (i.e., she uses multipath routing), we show that with the adequate multipath splitting scheme, she significantly improves her privacy without any performance overhead.

  Main contribution: To the best of our knowledge, we are the first to propose a defense against website fingerprinting attacks that exploits hybrid networks and multipath routing.

Throughput Part I

2 Multipath Routing in Hybrid Mesh Networks

2.1 Introduction

In this chapter,[1] we study multipath routing in hybrid mesh networks. When several technologies cohabit, multipath routing can provide significant throughput gains. Take again the network of Figure 1.3 on page 6, with a hybrid PLC/WiFi router (Node A), a PLC/WiFi range extender (Node B), and a laptop with WiFi (Node C). Node C downloads a file from the Internet (i.e., through A): Because PLC and WiFi do not interfere with each other, 10 Mb/s can be sent over the hybrid PLC-WiFi Path 1. This is what would happen with a typical PLC/WiFi extender, but it would consume only $1/3$ of WiFi resources. To maximize throughput, *some* traffic can be sent over the two-hop WiFi Path 2. The amount of traffic needs to be carefully calibrated, as it is well known that saturating multi-hop paths is inefficient and can lead to congestion collapse with packet losses and instabilities [GSK04, Sri04]. This amount depends on the characteristics of the two WiFi links A – B and B – C (these links cannot transmit simultaneously and need to share the capacity because WiFi is a shared-medium technology). In this simple case (see Section 2.4 for general cases), a back-of-the-envelope computation gives us the rate x to send over Path 2 as the solution of $x/15 + x/30 = 2/3$, i.e., $x \simeq 6.6$ Mb/s. Hence using both Path 1 and Path 2 provides a 66% improvement, compared to using Path 1 alone. In the general case, when there are a large number of possible paths between the source and the destination, finding the best paths to use is a challenging task that we study in this chapter.

Outline of the Chapter This chapter is structured as follows. After presenting our network model in Section 2.2, we show in Section 2.3 that multipath and hybrid networks are intrinsically related, and that using multipath helps in terms of throughput only when several technologies are available. We then present a novel multipath routing protocol in Section 2.4. We discuss related work in Section 2.5 and close the chapter with a summary in Section 2.6.

[1] This chapter is based on the papers by Henri et al. [HVHT16] and by Henri and Thiran [HT18].

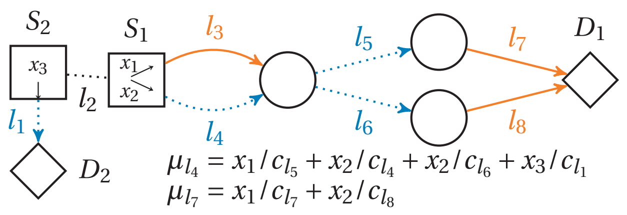

S_2
S_1
x_3
x_1
x_2
l_1
l_2
l_3
l_4
l_5
l_6
l_7
l_8
D_1
D_2
$\mu_{l_4} = x_1/c_{l_5} + x_2/c_{l_4} + x_2/c_{l_6} + x_3/c_{l_1}$
$\mu_{l_7} = x_1/c_{l_7} + x_2/c_{l_8}$

Figure 2.1 – Illustration of a multigraph, with two flows. Dotted blue lines represent WiFi, plain orange lines PLC. Sources send traffic on the different paths P_i at respective rates x_i ($\Lambda_{P_1} = \{l_3, l_5, l_7\}$, $\Lambda_{P_2} = \{l_4, l_6, l_8\}$, and $\Lambda_{P_3} = \{l_1\}$).

2.2 Network Model and Model Validation

We first present and validate our network model that will be used in the three chapters of Part I.

2.2.1 Network Model

We consider a multi-hop mesh network with K *orthogonal* and *shared-medium* technologies (e.g., PLC, WiFi, LTE): Two links that use the same technology are subject to interference (shared-medium technologies), but two links that use two distinct technologies are not subject to interference (orthogonal technologies). Two orthogonal WiFi channels are considered as two different technologies. The network is modelled by a *multigraph* $G(\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ the set of nodes and $\mathcal{E}$ the set of links. The total number of links in the network is denoted by L (i.e., $L = |\mathcal{E}|$). $\mathcal{E}$ is partitioned into K sets $\mathcal{E}_k$, $k \in \{1, \dots, K\}$, the sets of links available with each technology. A link is present whenever its two endpoints can communicate with each other with a non-zero rate on the corresponding technology. Figure 2.1 shows an example of a multigraph with $K = 2$ technologies, for example PLC and WiFi (here, $\mathcal{E}_1 = \mathcal{E}_{\text{PLC}} = \{l_3, l_7, l_8\}$ and $\mathcal{E}_2 = \mathcal{E}_{\text{WiFi}} = \{l_1, l_2, l_4, l_5, l_6\}$). For a link $l \in \mathcal{E}$, c_l is the capacity of l, i.e., the maximum rate achievable on l. For a link $l \in \mathcal{E}_k$, $\mathcal{I}_l \subset \mathcal{E}_k$ is the *interference domain* of l, defined as the set that contains l as well as all links that cannot transmit simultaneously with l because otherwise it would create a collision at one of the links. For example, in Figure 2.1, no link interferes with the PLC link l_3, i.e., $\mathcal{I}_{l_3} = \{l_3\}$; all other WiFi links interfere with the WiFi link l_4, i.e., $\mathcal{I}_{l_4} = \{l_1, l_2, l_4, l_5, l_6\}$; and the PLC link l_8 interferes with the PLC link l_7, i.e., $\mathcal{I}_{l_7} = \mathcal{I}_{l_8} = \{l_7, l_8\}$. For any links $l, l' \in \mathcal{E}$, $l \in \mathcal{I}_{l'} \Leftrightarrow l' \in \mathcal{I}_l$.

If a node transmits data to another node, we call the source-destination pair a *flow*. A *path* is a self-avoiding path of the multigraph G that connects two nodes. The source of a flow can use M paths $P_1, \dots, P_M$ simultaneously; the set $\mathcal{P} = (P_1, \dots, P_M)$ is called a *multipath*. When $M = 1$, the multipath is a single path. The set of links belonging to any path P_i is denoted by Λ_{P_i}, with $\Lambda_{P_i} \subset \mathcal{E}$; for a multipath $\mathcal{P} = (P_1, \dots, P_M)$, we write $\Lambda_{\mathcal{P}} = \bigcup_{i=1}^{M} \Lambda_{P_i}$, and $L_{\mathcal{P}} = |\Lambda_{\mathcal{P}}|$ for the total number of links in the multipath. For example, in Figure 2.1, a possible multipath with $M = 2$ paths from S_1 to D_1 consists of P_1 with $\Lambda_{P_1} = \{l_3, l_5, l_7\}$ and P_2 with $\Lambda_{P_2} = \{l_4, l_6, l_8\}$.

We define the *busy time* μ_l of a link l as the *fraction of time* during which no transmission can be initiated on l, because either (i) a transmission is already occurring on a link in its interference domain $\mathscr{I}_l$, or (ii) the channel is idle, but the node cannot transmit because, according to the distribution coordination function (DCF) and CSMA/CA protocols run by WiFi [wif09] and PLC [Plc10], it needs to wait for the expiration of an inter-frame space, or because it is in backoff stage. When a node sends traffic at rate x_l on a single link l with no other link transmitting, we assume that when the link is not saturated ($x_l \leq c_l$), then it will obtain a busy time proportional to x_l:

$$\mu_l = \frac{x_l}{c_l} = x_l \cdot d_l. \tag{2.1}$$

The validity of this assumption is discussed in Section 2.2.2; in the general case, it is in not valid in practice, but for the problems we consider, we show that it is valid. When the link is saturated, then $\mu_l = 1$.

The source of a flow sends data at rate x_i on each path P_i for $i \in \{1, \dots, M\}$, and we denote by $\boldsymbol{x}_{\mathscr{P}}$ the rate vector $[x_i]_{i \in \{1,\dots,M\}}$. If $x_i = 0$, path P_i is not used. For each link $l \in \Lambda_{\mathscr{P}}$, the total busy time (accounting for interference) follows, if links are not saturated, from Equation (2.1), and is given by

$$\mu_{l,\boldsymbol{x}_{\mathscr{P}}} = \sum_{l' \in \mathscr{I}_l} \mu_{l'} = \sum_{i=1}^{M} x_i \sum_{l' \in \mathscr{I}_l \cap \Lambda_{P_i}} \frac{1}{c_{l'}} = \sum_{i=1}^{M} x_i \alpha_{P_i,l}, \tag{2.2}$$

where we define

$$\alpha_{P_i,l} \doteq \sum_{l' \in \mathscr{I}_l \cap \Lambda_{P_i}} \frac{1}{c_{l'}}.$$

Figure 2.1 illustrates the busy time with interfering links. We say that a rate vector $\boldsymbol{x}_{\mathscr{P}}$ is *admissible* if for all $l \in \Lambda_{\mathscr{P}}$, $\mu_{l,\boldsymbol{x}_{\mathscr{P}}} \leq 1$ (the busy-times on the links of the multipath never exceed 100%).

Writing $\boldsymbol{\alpha}_{\mathscr{P},l} \in \mathbb{R}^M$ for the vector $[\alpha_{P_i,l}]_{i \in \{1,\dots,M\}}$, Equation (2.2) can be recast as $\mu_{l,\boldsymbol{x}_{\mathscr{P}}} = \boldsymbol{\alpha}^T_{\mathscr{P},l} \cdot \boldsymbol{x}_{\mathscr{P}}$ with T denoting transposition. $\boldsymbol{\alpha}_{\mathscr{P},l}$ is called the *multipath-impact vector* of $\mathscr{P}$ on l; it depends only on the network topology (i.e., the link capacities and interference domains) and on the paths, and *not* on the rate vector $\boldsymbol{x}_{\mathscr{P}}$. We denote by $\boldsymbol{\mu}_{\boldsymbol{x}_{\mathscr{P}}} \in \mathbb{R}^{L_{\mathscr{P}}}$ the vector $\boldsymbol{\mu}_{\boldsymbol{x}_{\mathscr{P}}} = [\mu_{l,\boldsymbol{x}_{\mathscr{P}}}]_{l \in \Lambda_{\mathscr{P}}}$ and by $\boldsymbol{A}_{\mathscr{P}} \in \mathbb{R}^{L_{\mathscr{P}} \times M}$ the matrix

$$\boldsymbol{A}_{\mathscr{P}} = [\boldsymbol{\alpha}^T_{\mathscr{P},l}]_{l \in \Lambda_{\mathscr{P}}}. \tag{2.3}$$

Table 2.1 – Main notation of the network model.

$\mathcal{V}$	set of nodes
$\mathcal{E}$	set of all links of the network
L	total number of links of the network ($L = \lvert\mathcal{E}\rvert$)
$\mathcal{E}_k$	set of links of technology k
c_l	capacity of link l
I_l	interference domain of link l
μ_l	busy time on link l
$\mathscr{P} = (P_1, \ldots, P_M)$	multipath with M paths $P_1, \ldots, P_M$
Λ_P	links that constitute path P
$\Lambda_{\mathscr{P}}$	links that constitute a multipath $\mathscr{P}$: $\Lambda_{\mathscr{P}} = \bigcup_{i=1}^{M} \Lambda_{P_i}$
$\alpha_{P,l}$	$\sum_{l' \in \mathscr{I}_l \cap P} 1/c_{l'}$
$\boldsymbol{\alpha}_{\mathscr{P},l} \in \mathbb{R}^M$	multipath-impact vector $[\alpha_{P_i,l}]_{i\in\{1,\ldots,M\}}$
$\boldsymbol{A}_{\mathscr{P}} \in \mathbb{R}^{L_{\mathscr{P}} \times M}$	matrix $[\boldsymbol{\alpha}_{\mathscr{P},l}^T]_{l \in \Lambda_{\mathscr{P}}}$
$\boldsymbol{x}_{\mathscr{P}} \in \mathbb{R}^M$	vector of rates sent on multipath $\mathscr{P}$
$\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}$	optimal rate on the multipath $\mathscr{P}$
Π	set of all multipaths for the flow
x^{opt}	optimal rate for the flow, $x^{\text{opt}} = \max_{\mathscr{P}\in\Pi} \left\lVert \boldsymbol{x}_{\mathscr{P}}^{\text{opt}} \right\rVert_1$
$\boldsymbol{X}_{\mathscr{P}}$	matrix $[\boldsymbol{x}_{\mathscr{P}}^{(i)T}]_{i\in\{1,\ldots,M\}}$

With the example of Figure 2.1 and $\mathscr{P} = (P_1, P_2)$, we have

$$\boldsymbol{A}_{\mathscr{P}} = \begin{bmatrix} \boldsymbol{\alpha}_{\mathscr{P},l_3}^T \\ \boldsymbol{\alpha}_{\mathscr{P},l_4}^T \\ \boldsymbol{\alpha}_{\mathscr{P},l_5}^T \\ \boldsymbol{\alpha}_{\mathscr{P},l_6}^T \\ \boldsymbol{\alpha}_{\mathscr{P},l_7}^T \\ \boldsymbol{\alpha}_{\mathscr{P},l_8}^T \end{bmatrix} = \begin{bmatrix} 1/c_{l_3} & 0 \\ 1/c_{l_5} & 1/c_{l_4} + 1/c_{l_6} \\ 1/c_{l_5} & 1/c_{l_4} + 1/c_{l_6} \\ 1/c_{l_5} & 1/c_{l_4} + 1/c_{l_6} \\ 1/c_{l_7} & 1/c_{l_8} \\ 1/c_{l_7} & 1/c_{l_8} \end{bmatrix}.$$

The optimal rate vector on multipath $\mathscr{P}$, denoted by $\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}$, is the admissible rate vector that maximizes the 1-norm. Because $\boldsymbol{A}_{\mathscr{P}} \cdot \boldsymbol{x}_{\mathscr{P}} = \boldsymbol{\mu}_{\boldsymbol{x}_{\mathscr{P}}}$, $\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}$ is a solution of the following system:

$$\begin{aligned} &\max_{\boldsymbol{x}} \ \mathbf{1}^T \cdot \boldsymbol{x} \\ &\text{subject to } \boldsymbol{A}_{\mathscr{P}} \cdot \boldsymbol{x} \preceq \mathbf{1} \text{ and } \boldsymbol{x} \succeq \mathbf{0}, \end{aligned} \tag{2.4}$$

where $\succeq$ and $\preceq$ denote component-wise inequalities.

For a given flow, the optimal rate or optimal throughput (the two terms are used interchangeably in this dissertation) is

$$x^{\text{opt}} \doteq \max_{\mathscr{P}\in\Pi} \left\lVert \boldsymbol{x}_{\mathscr{P}}^{\text{opt}} \right\rVert_1, \tag{2.5}$$

where Π denotes the set of all possible multipaths for the flow. We define the optimal number of paths M^{opt} as the minimal number of paths in a multipath $\mathscr{P}^{\text{opt}}$ reaching the optimal rate x^{opt}. In particular, all the M^{opt} paths of $\mathscr{P}^{\text{opt}}$ are used.

Table 2.1 summarizes the main notations of the model.

Using Equation (2.1), we prove the following lemma.

Lemma 1. *If n links $l_1, \ldots, l_n$ are all contending for the same medium in a single collision domain ($\forall i, j, \mathscr{I}_{l_i} = \mathscr{I}_{l_j}$), then the rate R_{max}, defined as the maximum rate simultaneously achievable by each link (i.e., each individual link transmits at rate R_{max}), is given by*

$$R_{max} = \left(\sum_{i=1}^{n} \frac{1}{c_{l_i}} \right)^{-1} = \left(\sum_{i=1}^{n} d_{l_i} \right)^{-1}. \tag{2.6}$$

Proof. Recall that μ_{l_i} is the airtime of link l_i (i.e., the proportion of time during which link l_i is active). This problem is thus equivalent to solving $\sum_{i=1}^{n} \mu_{l_i} = 1$ (because all links contend for the same medium) and $\mu_{l_i} c_{l_i} = \mu_{l_1} c_{l_1}$ for all $i \geq 2$ (all links send at same throughput). The solution to this linear problem is

$$\mu_{l_i} = \frac{\prod_{j \neq i} c_{l_j}}{\sum_{j=1}^{n} \prod_{j' \neq j} c_{l_{j'}}}.$$

Setting $R_{\text{max}} = \mu_{l_i} c_{l_i}$ yields the result. □

2.2.2 Model Validation

The key assumption made in the previous subsection is the linear relationship (2.1). To evaluate its validity, we carry out the following experiment on our testbed that consists in 22 nodes spread over an entire floor of an office building of 65×40 m (see Figure A.1 on page 135). The testbed is described in details in Section A in the Appendix. A node sends traffic on a link l, in a first stage with no other link contending, at various rates x_l bytes per second (i.e., it sends one packet of S bytes every S/x_l second), and it measures the busy times.[2] The experiment is repeated 50 times for each rate x_l, and the results are shown in Figure 2.2 for both WiFi (left) and PLC (right). The link capacity $\bar{c}_l$, indicated by the black vertical line, is the maximum rate received by the destination. The dotted orange line indicates the linear relationship (2.1), clearly not valid in this experiment.

The invalidity of the linear relationship (2.1) comes from the introduction of frame aggregation in recent standards (e.g., IEEE 802.11n/ac for WiFi, IEEE 1901 for PLC) to increase throughput. At low rates, frame aggregation is barely used, because the interval between two consecutive packets

[2]For WiFi, ath9k drivers expose directly the measured busy-times that can be accessed using netlink sockets [Hor04]. For PLC, the node sniffs all packets using `faifa` [CF08], and uses the *duration* field of every PLC packet to compute the busy time.

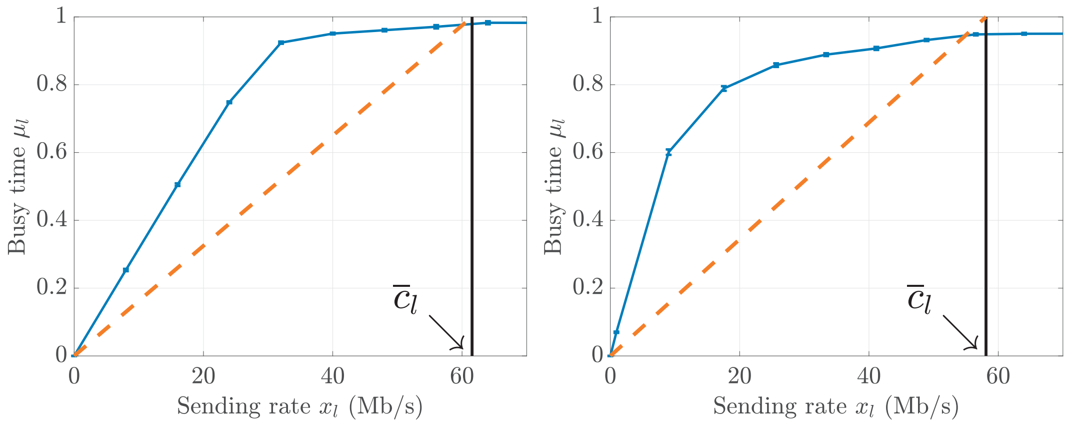


Figure 2.2 – Busy time μ_l versus rate x_l sent on link l, without batches. WiFi (left) and PLC (right). The experiment is repeated 50 times for each rate x_l; each point represents the average busy time μ_l, with the bars representing standard deviations.

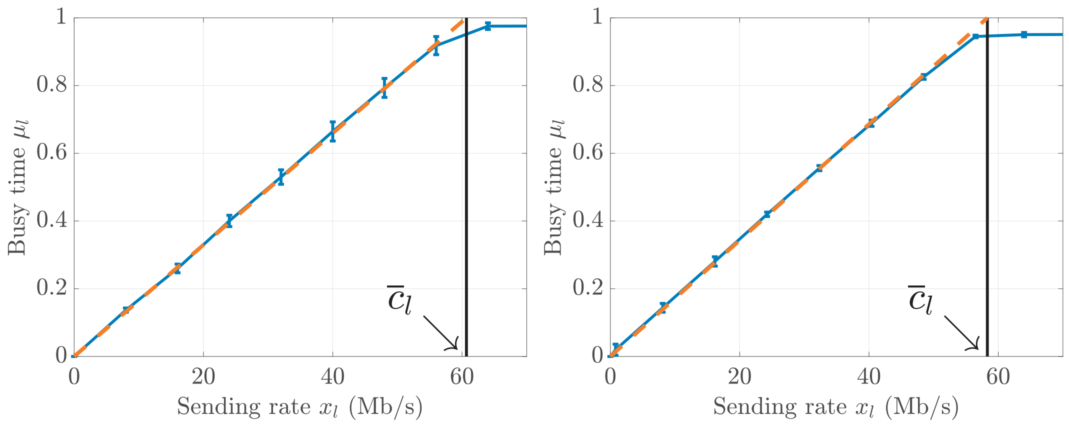


Figure 2.3 – Busy time μ_l versus rate sent on link l, with batches of B packets. WiFi (left, $B = 100$) and PLC (right, $B = 200$).

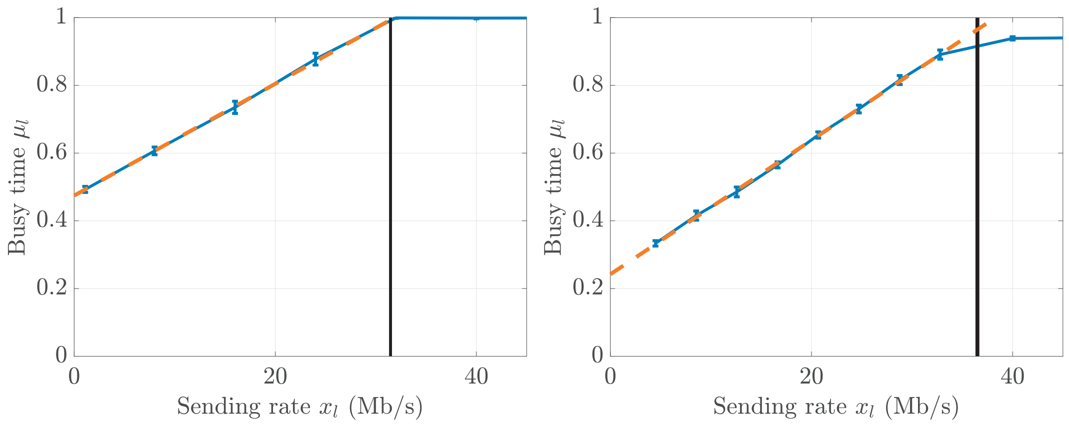


Figure 2.4 – Busy time μ_l versus rate sent on link l, with 3 contending nodes. WiFi (left) and PLC (right).

is too large for the network interface to wait for packets to aggregate. In contrast, it is fully used when the source sends saturated traffic. Having two different operating regimes invalidates the linear relationship (2.1). To make it valid, we can force frame aggregation even at low rates by sending batches of packets. We repeat the experiment with the node now sending traffic by batches of B packets: For a rate x_l, the node sends B packets of S bytes every $B \cdot S / x_l$ second. In Figure 2.3, we see that the linear relationship (2.1) is now valid for both technologies. We repeat this experiment on several links with similar results. A difference between PLC and WiFi is that PLC requires longer batches; we use $B = 100$ for WiFi and $B = 200$ for PLC. Note that compared to Figure 2.2, we see an increase of the standard deviation, in particular for WiFi.

The relation busy time vs. rate is also linear when more than one node transmits, which justifies Equation (2.2). We repeat the same experiment and send traffic at various rates on link l; two other nodes of $\mathscr{I}_l$ now contend and transmit traffic at constant rate. The transmitting nodes send traffic by batches of $B = 100$ packets for WiFi, $B = 200$ for PLC. In Figure 2.4, we show the measurements of the busy times μ_l. The black vertical line now indicates the maximum achievable rate under the contention of the two other nodes. Busy times μ_l are also linear in rates x_l when other nodes transmit.

In addition, sending packets by batches is required for Equation (2.2) to be valid, but only when the medium is significantly below saturation. When the total busy-time (accounting for interference) approaches 100%, then all nodes use frame aggregation to maximize their throughput, and the busy-time per link μ_l is given by Equation (2.1). We show this by verifying the validity of Lemma 1 with testbed experiments. We select randomly n links in the center of the testbed, i.e., where interferences are likely to happen (in Figure A.1 on page 135, the lines between respectively Nodes 5 and 8, Nodes 1 and 6, Nodes 17 and 19, and Nodes 18 and 22 are thick walls through which WiFi signal is largely attenuated). For WiFi, we use Nodes 6 to 18; for PLC, we use Nodes 6 to 13, because Nodes 1 to 13 and Nodes 14 to 22 form two different PLC subnetworks and because a link from one subnetwork does not interfere with a link from the other subnetwork. We check that the n links indeed interfere with each other by checking that when a link is active, all the other sources overhear the transmitted packets. We first measure the individual link capacities by sending UDP traffic (*without* batches) with `iperf` on the n links in a row; we compute the theoretical $R_{\max}$ by using Equation (2.6). We then measure the experimental $R_{\max}$ by sending UDP traffic simultaneously on the n links, with increasing rates, keeping the maximum rate that all links support. We show the results for 10 randomly selected experiments, with $n = 2$ in Figures 2.5 and 2.6 (respectively for WiFi and PLC), and with $n = 3$ in Figures 2.7 and 2.8 (respectively for WiFi and PLC). It is clear that Lemma 1 gives very precise results, which validates Equation (2.1) when the total busy-time approaches 100%.

The results of this section indicate that it is indeed possible to find $\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}$ by solving System (2.4). Section 4.6 presents additional experimental results that show that this method gives precise results.

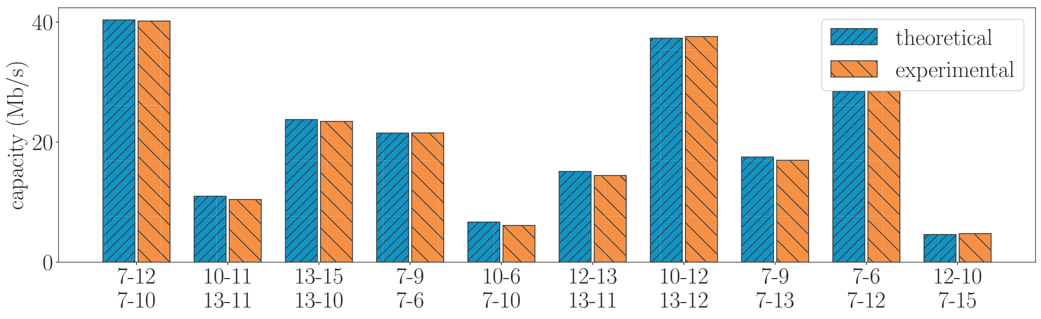


Figure 2.5 – Theoretical and experimental R_{max}, WiFi ($n = 2$).

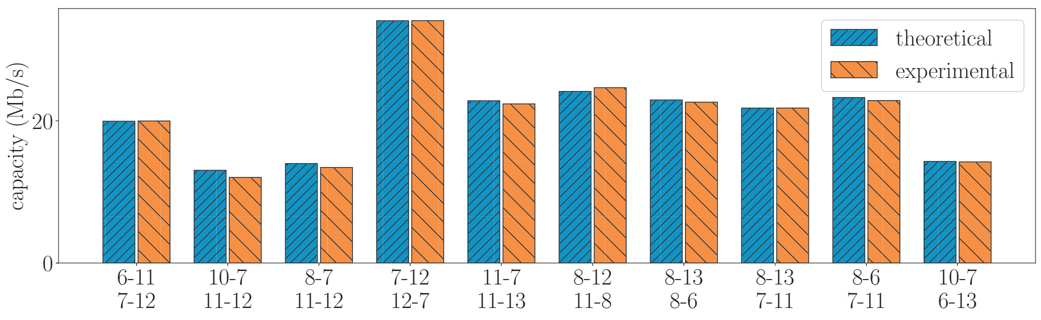


Figure 2.6 – Theoretical and experimental R_{max}, PLC ($n = 2$).

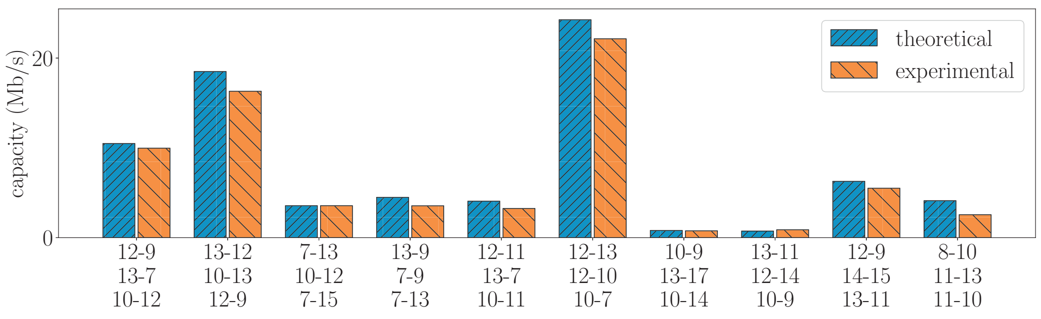


Figure 2.7 – Theoretical and experimental R_{max}, WiFi ($n = 3$).

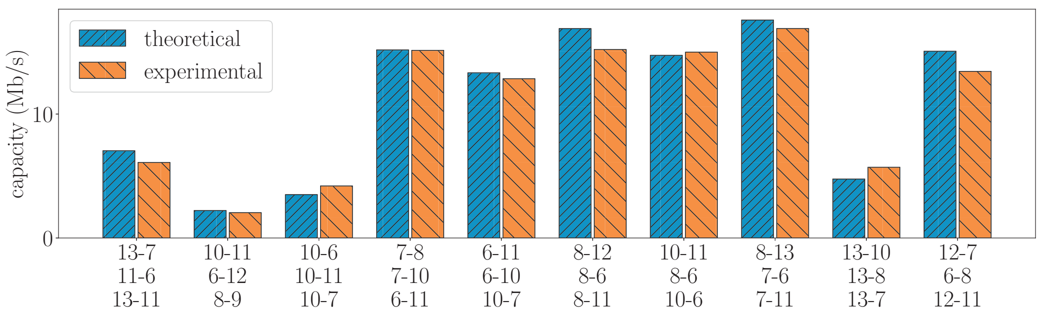


Figure 2.8 – Theoretical and experimental R_{max}, PLC ($n = 3$).

2.3 Optimal Number of Paths with Multipath Routing in Hybrid Mesh Networks

In this section, we study the optimal number of paths M^{opt}, as introduced in Section 2.2, when using multipath routing in hybrid networks with shared-medium and orthogonal technologies. We first explain why this is a difficult problem in practice. We then present our analytical findings valid for certain classes of networks. Finally, these findings are verified with simulations and experiments, and we also present numerical and experimental results for general networks.

2.3.1 Practical Difficulties for Computing the Optimal Number of Paths

The optimal number of paths M^{opt} is the minimal number of paths in a multipath $\mathscr{P}^{\text{opt}}$ that reaches the optimal rate x^{opt}. Because a path has no loop, there is a finite number of paths between the source and the destination, and in theory, it is possible to find x^{opt} by solving System (2.4) and computing $\boldsymbol{x}^{\text{opt}}_{\mathscr{P}^{\text{all}}}$ where $\mathscr{P}^{\text{all}}$ is the multipath containing all the possible paths for a given flow. However, the number of paths in $\mathscr{P}^{\text{all}}$ grows exponentially with the number of nodes and technologies, which makes this method impractical. But in fact, there is no better solution for finding x^{opt} and $\mathscr{P}^{\text{opt}}$: It has been shown that in a network with interference, finding an optimal multipath is NP-hard [JPPQ05].

Even if we can find the optimal rate x^{opt}, there might be several optimal multipaths with different number of paths, and finding the minimal number M^{opt} is still challenging: It has been shown that computing the minimal-rank solution of a linear problem is also NP-hard [RFP10]. This means that finding M^{opt} by computing the minimal-rank solution of System (2.4) is not practical: The number of possible multipaths grows exponentially with the number of paths, that grows itself exponentially with the number of nodes and technologies.

Because of these practical difficulties, we study here M^{opt} without searching for an optimal multipath of minimum rank. Finding M^{opt} without knowing the corresponding optimal multipath $\mathscr{P}^{\text{opt}}$ remains of practical interest, because it makes it possible to limit the size of the multipaths returned by a multipath-routing protocol without harming the performance of the protocol.

2.3.2 Analysis

The key result is that in a hybrid mesh network with shared-medium technologies, M^{opt} is tightly linked with the number K of orthogonal technologies. This result is straightforward in a single-hop network (i.e., in the infrastructure mode): If a node is connected to another node with K distinct technologies, it is obvious that the optimal throughput will be reached when the K technologies are used, i.e., that $M^{\text{opt}} = K$. In a multi-hop mesh network, this is not true in general, and we show that the network needs to verify some specific conditions to prove analytically a relation between M^{opt} and K.

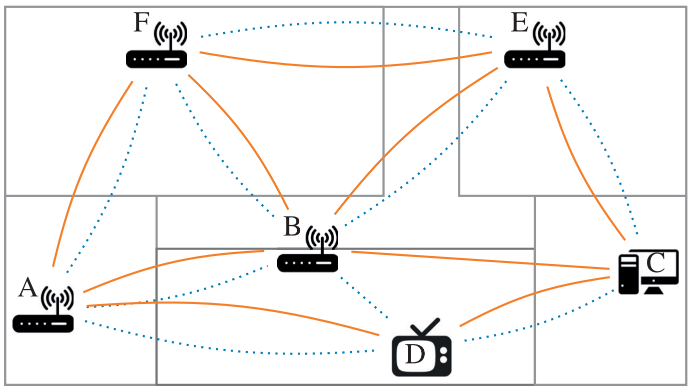


Figure 2.9 – Example of a typical network for a five-room home with 6 nodes and $K = 2$ technologies: WiFi (dotted blue lines) and PLC (plain orange lines). This network is both multi-complete and multi-connected.

We define the following terms.

Definition 1. *The network is* multi-complete *if for every technology k, every link $l \in \mathscr{E}_k$ of the network interferes with every other link $l' \in \mathscr{E}_k$ (i.e., the interference graph for each technology is complete).*

Note that this does *not* mean that the graph $(\mathscr{V}, \mathscr{E}_k)$ is complete (i.e., that every node is directly connected with every other node): For example, in Figure 2.9 that represents a typical network for a five-room home, all WiFi links (dotted lines) interfere with each other and all PLC links (plain lines) interfere with each other, i.e., the network is multi-complete; but there is no direct WiFi and PLC link for example between Node A and Node C and between Node D and Node E.

Definition 2. *The network is* multi-connected *if all the K sub-networks $(\mathscr{V}, \mathscr{E}_1), \dots, (\mathscr{V}, \mathscr{E}_K)$ are connected: For each technology k, each node in $\mathscr{V}$ can communicate with each other node in $\mathscr{V}$, possibly with multiple hops, by using only links of $\mathscr{E}_k$.*

The network represented in Figure 2.9 is multi-connected: every node can reach every other node by using only WiFi links and only PLC links, possibly with multi-hop paths.

We start by showing that for networks that are not multi-complete or not multi-connected, it is possible to find examples where $M^{\text{opt}} \neq K$.

Let us first consider the network example in Figure 2.10. There are $K = 3$ technologies, represented by plain, dashed, and dotted lines with different colors. Clearly, the network is not multi-connected, because Node 1 cannot reach Node 3 by using any single technology only. There are only $M = 2$ possible paths between Node 1 and Node 3; consequently, $M^{\text{opt}} \leq 2$, i.e., $M^{\text{opt}} < K$. In fact, depending on the link capacities, either $M^{\text{opt}} = 1$ if the green dashed link between Node 2 and Node 3 is the bottleneck link; otherwise, $M^{\text{opt}} = 2$.

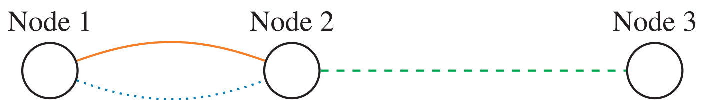


Figure 2.10 – Example of a network that is not multi-connected, with $K = 3$ technologies (plain, dashed and dotted lines).

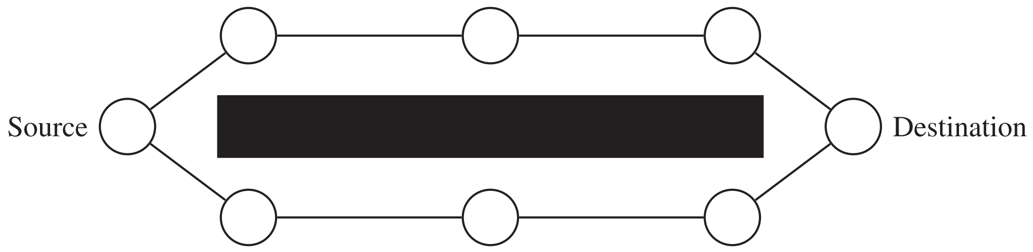


Figure 2.11 – Example of a network that is not multi-complete, with $K = 1$ technology.

Let us then consider the network example in Figure 2.11 with $K = 1$ technology. Links on one side of the wall (the black rectangle) do not interfere with the links on the other side of the wall, i.e., the network is not multi-complete. There are two possible paths, and if the bottleneck of these paths is not the first-hop link or the last-hop link, then using $M = 2$ path yields a better throughput than using only one path, and $M^{\text{opt}} = 2$, i.e., $M^{\text{opt}} > K$.

We then present our analytical results. We show that the multi-complete and multi-connected conditions are sufficient to prove analytically a relation between M^{opt} and K. The proofs of the theorems of this section are presented in the Appendix B.1.

Theorem 1. *In a multi-complete network, $M^{opt} \leq K$.*

Because small networks (e.g., typical home networks) are likely to be multi-complete, Theorem 1 shows in particular that in small networks with a single technology ($K = 1$), multipath routing is likely to be useless in terms of throughput.

The next analytical results are valid under the following assumption. In the numerical and experimental results presented in Section 2.3.3, this assumption is not made, and we show that the results remain true in the vast majority of the cases.

Assumption 1. *A property that depends on the link capacities is true if and only if there is $\epsilon_0 > 0$ such that for all $0 < \epsilon < \epsilon_0$ and all links $l_0 \in \mathscr{E}$, the property remains true if the capacity of l_0 is modified as $c_{l_0}(1 \pm \epsilon)$ whereas the capacities of the other links remain unchanged.*

This means that to be true, a property must be robust against small variations of the link capacities. For example, Assumption 1 yields that for two links $l, k \in \mathscr{E}$, $c_l = c_k$ if and only if $l = k$: If we assume $c_l = c_k$ and $l \neq k$, then adding any small $\epsilon > 0$ to one of the two link capacities invalidates the equality, because $c_l \pm \epsilon \neq c_k$. More generally, Assumption 1 yields that

$$\sum_{l \in S_1} \frac{1}{c_l} = \sum_{l \in S_2} \frac{1}{c_l} \text{ if and only if } S_1 = S_2. \tag{2.7}$$

Lemma 2. *With $M \leq K$, an admissible rate vector $\mathbf{x}_{\mathscr{P}}$ sent on a multipath $\mathscr{P}$ with M paths saturates at most M technologies, i.e., at most M technologies have a link whose busy time is exactly 1.*

A counter-example of Lemma 2 when Assumption 1 is not verified is the case of a single two-hop path ($M = 1$) with two links l_1 and l_2 that use two different technologies ($K = 2$), but have the exact same capacity ($c_{l_1} = c_{l_2}$, which contradicts Assumption 1). In this scenario, sending traffic only on this single-path at a rate c_{l_1} saturates both l_1 and l_2, i.e., it saturates the $K = 2$ technologies. In practice, it is unlikely that two links with different technologies have the same capacity, and we show in Section 2.3.3 that the result holds in the vast majority of the cases.

Theorem 2. *In a multi-connected network, $M^{opt} \geq K$.*

Corollary 1. *In a multi-complete and multi-connected network, $M^{opt} = K$.*

For example, in the typical home network presented in Figure 2.9 that is both multi-complete and multi-connected, Corollary 1 shows that the optimal number of paths M^{opt} is equal to the number of technologies K.

2.3.3 Numerical and Experimental Results

In this section, we verify with simulations and testbed experiments the findings of our analysis for multi-complete and multi-connected networks. We also present numerical and experimental results for more general networks that are not necessarily multi-complete and multi-connected.

Benchmarking Methods for the Optimal Number of Paths

Brute-force Method As explained in Section 2.3.1, finding an optimal multipath in a network with interference is NP-hard [JPPQ05]. To the best of our knowledge, all practical multipath-routing protocols use heuristics and do not guarantee the optimality of their results. We can find an optimal multipath by using brute-force, i.e., by computing the set $\mathscr{P}^{\text{all}}$ of all possible paths and by solving System (2.4) with $\mathscr{P} = \mathscr{P}^{\text{all}}$. However, the number of possible paths is exponential in the number of nodes and technologies, which makes this method highly computation-intensive. In fact, in the simulations of Section 2.3.3, we have to limit the number of paths in order to be able to solve System (2.4). We do so, in such a way that this is very unlikely to change the final result, by limiting the number of hops in the paths. We set the maximum number of hops to three times the minimum number of hops between the source and the destination. For example, if there exists at least one two-hop path between the source and the destination but no single-hop path, we limit the paths between the source and the destination to those with six or fewer hops. If the source and the destination are two-hop away, it is very unlikely that paths of seven or more hops are required to reach the optimal rate x^{opt}.

Backpressure Method It is also possible to find a multipath arbitrarily close to an optimal one with the method described by Neely et al. [NML08]. This method employs a backpressure scheme: The source initially floods the network by sending traffic in all directions; packets that arrive at the destination are removed from the network, whereas other packets stay in the queues of the nodes. Gradually, traffic is sent only to nodes that have small queues, which indicates that they are in the "right" direction. This scheme is shown to converge arbitrarily close to the maximal achievable rate x^{opt} as given by Equation (2.5), namely to $(1-1/V)\cdot x^{\text{opt}}$ for some constant V. In the simulations and experiments, we choose $V = 1000$, such that the difference between the optimal rate and the rate that is found by this method is at most $0.001 \cdot x^{\text{opt}}$. By considering only the links that are used once the scheme has converged (i.e., the links on which traffic is sent at a rate above the threshold $0.001 \cdot x^{\text{opt}}$), we compute a multipath that yields a rate arbitrarily close to optimum, and we assume that it uses the same number of paths as an optimal multipath.

These two methods give us one multipath reaching the optimal rate (or arbitrarily close to it), but they are not guaranteed to return the optimal multipath with a minimal number of paths, consequently we can only compute an upper bound of M^{opt}. As explained in Section 2.3.1, computing the minimal-rank solution of a linear problem is NP-hard; here, with up to several millions of possible paths, the number of possible multipaths is far too large to enable us to find the exact value of M^{opt}. However, we believe that in practice, there is a single optimal multipath in most of the cases, and that the upper bound is therefore tight in most of the cases.

We have only a benchmarking goal when we experimentally evaluate M^{opt}, and the efficiency of the benchmarking schemes is not the subject of this work. In fact, both these benchmarking schemes are impractical. The brute-force scheme requires solving a system whose size is exponential in the number of nodes and technologies; most of our simulations take several hours to find the result. The backpressure scheme would also be difficult to use in a real-world application for several reasons: (i) It requires knowing the interference domain of each link in advance, which is typically challenging or impractical [PD11]. (ii) It requires a centralized coordinator that decides at each time slot which links are to be used. (iii) It initially floods the entire network.

Simulation Results

We present results obtained with a Matlab simulator.[3] Each node has $K = 3$ technologies, with random ranges between 20 m and 40 m, and random maximum rates between 20 Mb/s and

[3]The code of the simulator and the code for the experiments presented in Section 2.3.3 are available at https://c4science.ch/diffusion/6360.

180 Mb/s. Each link capacity is distributed according to a linear function that decreases with the distance, to which is added a zero-mean normally-distributed noise, with parameters chosen such that the capacities are close to the ones observed on our WiFi-PLC testbed (see Section A.3 in the Appendix). One technology uses the parameters found for PLC, the other two use the parameters found for WiFi (i.e., we simulate networks that have PLC and two orthogonal WiFi channels). The technologies are shared-medium and orthogonal. Two links l and l' of a same technology interfere if one node of l and one node of l' are within range of each other. We compute an upper bound on M^{opt} with the two different benchmarking methods described in Section 2.3.3, and we keep the minimum of the two results (in the following, we slightly abuse notation and write M^{opt} for this upper bound).

We first simulate a multi-connected and multi-complete network in order to compare our analysis with the simulation results. The network, denoted by Network 1, is a 40×40 m square with 10 nodes randomly placed. We simulate 1000 different random instances of Network 1: For each instance, the placement of the nodes is made uniformly at random, the choice of the link capacities is made randomly according to the distribution described in the previous paragraph, and the choice of the source and destination is made uniformly at random. If the network instance is not a multi-connected and multi-complete network, we remove the experiment (this occurs in 3% of the 1000 experiments). In 99.4% of the cases, the optimal number of paths is 3, which shows that in a multi-complete and multi-connected network, $M^{\text{opt}} = K$, as proven by Corollary 1. In a very few instances (0.2%), we find $M^{\text{opt}} = 4 > K$; theses cases appear when the optimal multipath with minimal number of paths is not found. In a very few instances (0.4%), we find $M^{\text{opt}} = 2 < K$; these cases appear when the rate for a third path is below the threshold described in Section 2.3.3, equal to $0.001 \cdot x^{\text{opt}}$.

We then study through simulations whether the analytical results presented in Section 2.3.2 can be extended to more general networks. We simulate three larger networks that are not necessarily multi-connected and multi-complete: Network 2, a 100×100 m square with 15 nodes; Network 3, a 200×150 m rectangle with 20 nodes; and Network 4, a 200×150 m rectangle with 30 nodes. In these larger networks, we simulate the fact that a PLC link exists only when two nodes are connected to the same *central coordinator* [Plc10] (in particular, when two nodes are on the same electrical panel) by dividing the square in two equal parts, and by considering that, for one of the three technologies (the technology that simulates PLC), a link exists between two nodes only if the two nodes are in the same part. In particular, this means that the networks are never multi-connected. We simulate 1000 random instances of Network 2 and Network 4, and 1500 random instances of Network 3; there are more instances for Network 3 because the instances where there is no connectivity between the source and destination nodes are more frequent. In total, there are 5% of instances with no connectivity between the source and destination nodes for Network 2, 39% for Network 3 and 17% for Network 4, and these instances are not included in the results. Figure 2.12 shows the cumulative distributive function of M^{opt} for respectively Network 2 (left), Network 3 (center) and Network 4 (right). Even if no theoretical result has been proven for this network, we see that the optimal number of hops M^{opt} remains tightly linked with the number of technologies K: In a large majority of the instances (respectively 85%, 92% and

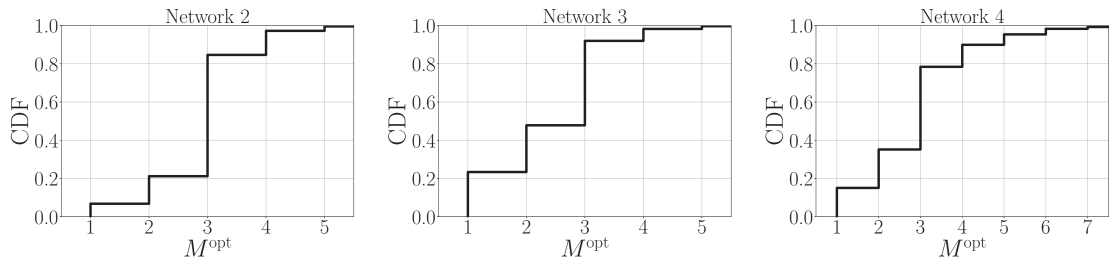


Figure 2.12 – Cumulative distribution function of the optimal number of paths M^{opt} for Network 2 (left), a general 100×100 m network with 15 nodes, Network 3 (center), a general 200×150 m network with 20 nodes, and Network 4 (right), a general 200×150 m network with 30 nodes. Simulations.

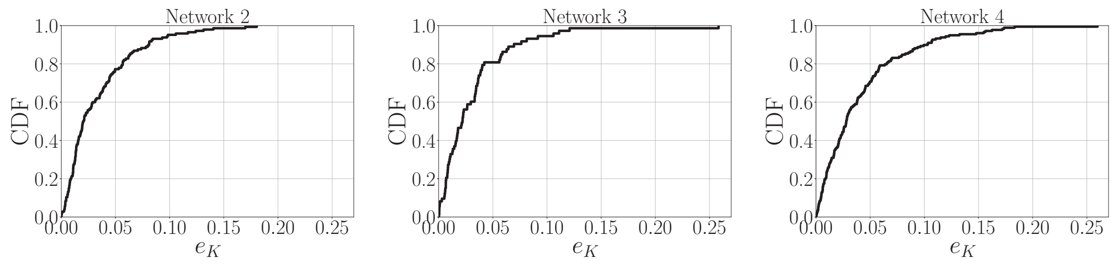


Figure 2.13 – Upper-bound e_K on the error made by using at most K paths when $M^{\text{opt}} > K$ for Network 2 (left), a general 100×100 m network with 15 nodes, Network 3 (center), a general 200×150 m network with 20 nodes, and Network 4 (right), a general 200×150 m network with 30 nodes. Simulations.

79%), we have $M^{\text{opt}} \leq K$.

We finally study, for the instances where $M^{\text{opt}} > K$, the rate loss caused by using at most K paths. We do so by comparing the optimal rate x^{opt} obtained on an optimal multipath $\mathscr{P}^{\text{opt}}$, with the rate x_K obtained by computing the optimal rate on the K best paths in $\mathscr{P}^{\text{opt}}$. Note that x_K is only a lower bound on the optimal rate x_K^{opt} achieved with multipaths of K paths, as there is no guarantee that the optimal multipath with K paths contains only paths that belong to the optimal multipath $\mathscr{P}^{\text{opt}}$. As explained at in Section 2.3.1, computing the actual optimal rate with multipaths of $M = K$ paths is NP-hard, and it cannot be computed in practice with up to several millions of possible paths, hence more than 10^{18} possible multipaths of $M = 3$ paths when $K = 3$. Using the K best paths of $\mathscr{P}^{\text{opt}}$ is simple and enables us to find an upper bound $e_K \doteq \frac{x^{\text{opt}} - x_K}{x^{\text{opt}}}$ on the minimum relative error $e_K^{\text{opt}} \doteq \frac{x^{\text{opt}} - x_K^{\text{opt}}}{x^{\text{opt}}}$.

Figure 2.13 shows e_K for Network 2 (left), Network 3 (center) and Network 4 (right), in the (respectively) 17%, 9% and 21% of the instances where $M^{\text{opt}} > K = 3$. We see that the error made by using only K paths is very small: In (respectively) 95%, 95% and 90% of the instances where $M^{\text{opt}} > K$, the relative error made by using only K paths is smaller than 0.1. Over all instances for each network, the relative error made by using only K paths is smaller than 0.1 in (respectively)

99.2%, 99.6% and 97.8% of the instances; and the relative error made by using only K paths is smaller than 0.05 in (respectively) 96.5%, 98.5% and 93.6% of the instances. Over all instances of Networks 1 to 4, the relative error made by using only K paths is smaller than 0.1 in 99.2% of the instances and smaller than 0.05 in 97.3% of the instances.

Experimental Results

We now present results obtained on our testbed described in Section A in the Appendix. All the nodes have two WiFi interfaces, and a HomePlug AV PLC interface connected to the electrical network of the building. The first WiFi channel is connected to a channel in the 2.4 GHz band, the second to a channel in the 5 GHz, consequently, they do not interfere. We run our experiments at night to avoid external interference from the WiFi network of our university that operates in the 2.4 GHz band. To compute the interference domain $\mathscr{I}_l$ of each link $l \in \mathscr{E}$, we run saturated traffic simultaneously on l and l' for each link $l' \neq l$, and we say that $l' \in \mathscr{I}_l$ if we observe a throughput degradation compared with the throughput when traffic is sent only on l. Note that this method is quadratic in the number of links, and is therefore not practical.

We first carry experiments with Nodes 6 to 13 only. This network is multi-complete (for each technology, all links interfere with each other). Again, this does not mean that the network itself is complete (e.g., Node 6 cannot communicate directly with Node 12 with any technology). We start with a scenario where the PLC network is not multi-connected. This is achieved by setting logically two PLC networks with two different network management keys [Plc10], one for Nodes 6 to 9, one for Nodes 10 to 13. Links in the two different PLC networks still interfere with each other, i.e., the network is multi-complete. We choose randomly 28 different flows (i.e., source-destination pairs) and run the optimal backpressure algorithm described above. The measurements of the link capacities and interference domains take several hours, and the algorithm converges in about 20 minutes on average. Because the network is multi-complete, we expect that $M^{\text{opt}} \leq 3$ (Theorem 1); and because the two WiFi networks are connected, we expect that $M^{\text{opt}} \geq 2$ (Theorem 2). Figure 2.14 (left) shows that this is indeed the case.

Next, we connect Nodes 6 to 13 to the same logical PLC network, i.e., the network with $K = 3$ technologies is multi-connected and multi-complete. We choose randomly 32 different flows and run the optimal backpressure algorithm. The proportion for each value of M^{opt} is shown in Figure 2.14 (center). In more than 90% of the cases, $M^{\text{opt}} = 3$, as expected. In one case, we find $M^{\text{opt}} = 4$; this is because link capacities vary slightly, and the algorithm alternates between different paths that yield very close rates. In one case, $M^{\text{opt}} = 2$ because the capacities of two links of two different technologies are too close for a third path to exist (i.e., Assumption 1 is not verified).

Finally, we perform an experiment with the whole testbed (Nodes 1 to 22) that is neither multi-complete nor multi-connected: Nodes 1 to 13 and Nodes 14 to 22 are on two different electrical panels, i.e., on two different PLC networks, and the two PLC networks do not interfere with each

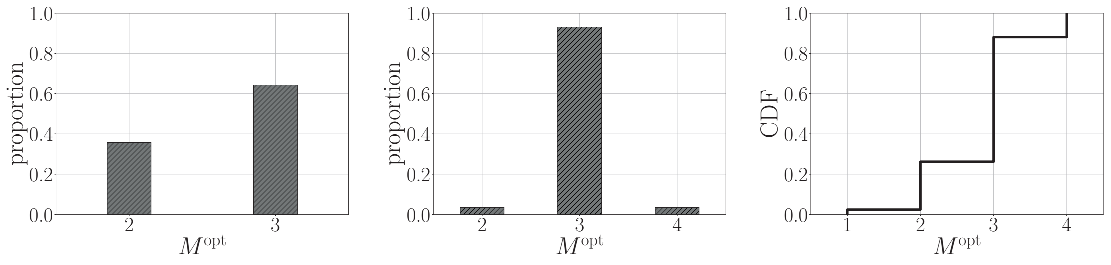


Figure 2.14 – Proportion of each value of M^{opt} in a multi-complete but not multi-connected network (left) and in a multi-complete and multi-connected network (center). Cumulative distribution function of M^{opt} in a general 65×40 m network (right). Testbed experiments.

other; also, WiFi links from one side (e.g., between Node 1 and Node 2) do not interfere with WiFi links from the other side (e.g., between Node 18 and Node 19). We choose randomly 42 flows. The cumulative distribution function of M^{opt} is shown in Figure 2.14 (right). Similarly to the simulations of Section 2.3.3, we see that in most of the cases (about 90%), $M^{\text{opt}} \leq K$. In the remaining cases where $M^{\text{opt}} > K$, the relative error e_K made by using only K paths, as defined in Section 2.3.3, is always below 0.1 (the maximum relative error is 0.08), and it is below 0.05 in 95.2% of the cases (all cases but two).

2.4 A Multipath Routing Protocol for Improved Throughput

We now present a novel multipath-routing algorithm for hybrid networks. Its purpose is to obtain an efficient multipath, i.e., a combination of paths that can be simultaneously employed by a given flow. As explained in Section 2.3.1, computing the optimal multipath is NP-hard, and our protocol relies on heuristics. As opposed to existing procedures that only focus on finding maximally disjoint paths [e.g., LG01, THT08, GRS11, DQZ+15], it aims at finding the combination of paths that yields the highest total throughput in the presence of interference: In our protocol, a same link can, and it should if useful, be used by several paths. The multipath protocol, described in Section 2.4.2, relies on an efficient single-path procedure, described in Section 2.4.1. This single-path procedure is based on an algorithm for multi-channel wireless networks, proposed by Yang et al. [YWK05].

2.4.1 Single-Path Procedure

The single-path procedure applies an efficient shortest-path algorithm (e.g., Dijkstra's algorithm) on the multigraph $G(\mathcal{V},\mathcal{E})$. A weight $W(l)$ is assigned to each link l. In addition, because contiguous links that use the same technology necessarily interfere for both WiFi and PLC, we want to favor paths whose contiguous links use different technologies, which mitigates intra-path interference. Similarly to Yang et al. [YWK05], we add a *channel-switching cost* (CSC). The CSC is an *extra* weight assigned to each node $u \in \mathcal{V}$, equal to $w_s(u)$ when a path switches interface at node u, and to $w_{ns}(u)$ when a path does not switch interface. The weight of a path is the sum of the weights of its links and of the CSCs of its intermediate nodes. For example, in Figure 1.3 on page 6, Path 1 employs two links with two different technologies (denoted by $l^{\text{PLC}}_{A\to B}$ and $l^{\text{WiFi}}_{B\to C}$) and hence the total weight of Path 1 is given by

$$W(\text{Path 1}) = W\left(l^{\text{PLC}}_{A\to B}\right) + w_s(B) + W\left(l^{\text{WiFi}}_{B\to C}\right).$$

Similarly, because Path 2 uses two WiFi links $l^{\text{WiFi}}_{A\to B}$ and $l^{\text{WiFi}}_{B\to C}$, its total weight is

$$W(\text{Path 2}) = W\left(l^{\text{WiFi}}_{A\to B}\right) + w_{ns}(B) + W\left(l^{\text{WiFi}}_{B\to C}\right).$$

Requiring $w_s(u) < w_{ns}(u)$ at each node u favors paths with alternating technologies. To guarantee that Dijkstra's algorithm returns the shortest path, the link-metric must be isotone [Sob01]; it is possible to make the CSC compatible isotone by performing the shortest-path computations on the virtual graph of the network *interfaces* [YWK05].

In our multipath-routing protocol, at any link l, we set $W(l) = d_l$ in order to favor the paths of higher capacities; up to a constant factor, d_l is equivalent to the ETT metric [DPZ04], used by many schemes [e.g., DPZ04, KV06, ECM$^+$08, SZ10]. If the weights w_{ns} and w_s can be chosen differently for each path, it is easy to show that, for two-hop paths, the optimal switching cost for adjacent links is $w_{ns} = 0$ and $w_s = -\min(d_{l_i}, d_{l_e})$, where l_i is the ingress link and l_e the egress link: Take the two-hop topology of Figure 2.15. There are four possible paths from A to C:

- WiFi-WiFi, of capacity $C_{WW} = (\frac{1}{c_{l_1^W}} + \frac{1}{c_{l_2^W}})^{-1}$ (see Lemma 1 in Section 2.2.1) and of weight $W_{WW} = \frac{1}{c_{l_1^W}} + \frac{1}{c_{l_2^W}} + w_{ns}$.
- WiFi-PLC, of capacity $C_{WP} = \min(c_{l_1^W}, c_{l_2^P})$ and of weight $W_{WP} = \frac{1}{c_{l_1^W}} + \frac{1}{c_{l_2^P}} + w_s$.
- PLC-PLC, of capacity $C_{PP} = (\frac{1}{c_{l_2^1}} + \frac{1}{c_{l_2^P}})^{-1}$ and of weight $W_{PP} = \frac{1}{c_{l_2^1}} + \frac{1}{c_{l_2^P}} + w_{ns}$.
- WiFi-PLC, of capacity $C_{PW} = \min(c_{l_2^1}, c_{l_2^W})$ and of weight $W_{PW} = \frac{1}{c_{l_2^1}} + \frac{1}{c_{l_2^W}} + w_s$.

By rewriting

$$C_{WP} = \left((\frac{1}{c_{l_1^W}} + \frac{1}{c_{l_2^P}} - \min(\frac{1}{c_{l_1^W}}, \frac{1}{c_{l_2^P}}) \right)^{-1},$$

we see directly (remember that $d_l = 1/c_l$) that by choosing $w_s = -\min(d_{l_i}, d_{l_e})$ and $w_{ns} = 0$, we have, for any path P, $W(P) = C(P)^{-1}$, which guarantees that the shortest path is the path with maximum throughput.

However, to be able to guarantee that the metric is isotone and that efficient shortest-path algorithms (such as Dijkstra's algorithm or Bellman-Ford's algorithm) converge to the shortest

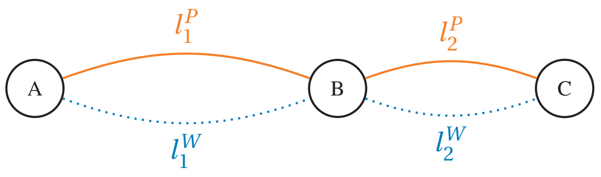


Figure 2.15 – Network example with WiFi (dotted lines) and PLC (plain lines).

path, the weights w_{ns} and w_s need to be chosen globally for a node, and they cannot be chosen depending on the path. For this reason, due to the above result and because a link is more likely to be used in a path if it has high capacity (i.e., small d_l), we believe that the best choice under this constraint is to have, at any node u, $w_{ns}(u) = 0$ and $w_s(u) = -\min_{l \in L(u)} d_l$, where $L(u)$ is the set of egress links for node u. It is more efficient to have non-negative weights, because it enables us to employ Dijkstra's algorithm rather than Bellman-Ford's algorithm; for this reason, we use instead $w_{ns}(u) = \min_{l \in L(u)} d_l$ and $w_s(u) = 0$.

The link-metric W is different from the metric, called IRU, of Yang et al. [YWK05]: Our routing protocol focuses on intra-flow interference (*via* the CSC), whereas IRU also accounts for inter-flow interference. In this dissertation, we propose a multipath congestion-controller (presented in Section 3.3) for handling inter-flow interference.

This single-path procedure is not necessarily optimal and the shortest path is not always the path with highest throughput (as illustrated further in Section 2.4.2). We account for this non-optimality in the multipath procedure presented next.

2.4.2 Finding Efficient Combinations of Paths

We now introduce our novel multipath-routing protocol and describe our procedure for finding an efficient multipath. To quantify the effect of employing several paths simultaneously, we define a procedure $\tilde{G} = \texttt{update}(P, G)$ for a multigraph G and a path P. $\tilde{G}$ is a view of the multigraph G where the capacities of the links have been updated to reflect the consumption of resources when traffic is sent over P. Let $R(P)$ be the maximum rate achievable (end-to-end) on path P. Once the procedure $\texttt{update}(P, G)$ has been applied, the capacities of all the links in the network $\tilde{G}$ are the available capacities if P is fully loaded, i.e., if traffic is sent on P at rate $R(P)$. $R(P)$ is the maximal rate supported simultaneously by all the links of the path, and can thus be computed as follows. From Lemma 1, the maximal traffic rate $R(l, P)$ on path P supported by a link $l \in P$ is given by $\left(\sum_{l' \in \mathscr{I}_l \cap P} d_{l'}\right)^{-1}$. A traffic rate R is supported by P if and only if $R \le R(l, P)$ for all $l \in P$, hence $R(P) = \min_{l \in P} R(l, P)$, or, equivalently,

$$R(P) = \left(\max_{l \in P} \sum_{l' \in \mathscr{I}_l \cap P} d_{l'} \right)^{-1}. \tag{2.8}$$

We use Equation (2.1) to find the airtime of a link $l \in P$ when data is sent on P at rate $R(P)$: $\mu_l = R(P) \cdot d_l$. For each link l of the network, the remaining proportion of idle time when data is sent on path P at rate $R(P)$ is

$$r(l,P) = \left(1 - \sum_{l' \in \mathscr{I}_l \cap P} R(P) \cdot d_{l'}\right).$$

By definition of $R(P)$, for $l \in P$, we have $r(l,P) \geq 0$, and there is at least one link l_0 of P for which $r(l_0,P) = 0$ (namely, the bottleneck link $l_0 = \arg\min_{l \in P} R(l,P)$). The procedure `update` is then defined as follows.

Procedure $\widetilde{G} = \texttt{update}(P, G)$

For each link $l \in \bigcup_{l' \in P} \mathscr{I}_{l'}$, update the capacity by:

$$C_{\widetilde{G}}(l) \leftarrow \max\{0, C_G(l) \cdot r(l,P)\},$$

where $C_G(l)$ denotes the capacity of link l in multigraph G.

The procedure `update` assumes that when a path P is used, traffic is sent on it at the maximum rate $R(P)$. This choice is optimal for simple topologies like the typical example of Figure 1.3. For general networks, this assumption makes the computation much faster, which enables the procedure to be used in practice, while yielding performance very close to optimal-but-impractical schemes (see the evaluation in Section 3.4).

To compute efficient combinations of paths, we recursively apply the procedure `update`, along with the single-path procedure of Section 2.4.1. Observe that the best path is not necessarily part of the best multipath: Consider for example the network of Figure 2.16. P_2 can accommodate a traffic rate of 11 Mb/s, whereas P_1 and P_3 can both accommodate only 10 Mb/s: P_2 is the best isolated path. However, the best multipath with two paths is $\mathscr{P} = (P_1, P_3)$: With Lemma 1, we compute that $\mathscr{P}$ can accommodate $5 + 10 = 15$ Mb/s. To avoid being constrained to always using the best isolated path, we compute in G the n shortest paths of our single-path procedure described above. We denote this step by n`-shortest`(G). Considering the n shortest paths also enables us to account for the potential non-optimality of the single-path procedure.

To obtain the final combination of paths, we build an *exploration tree* $\mathscr{T}$ in which the root G_0 is the initial multigraph. Each edge represents a path, and each vertex a multigraph with updated link capacities. The exploration tree is built recursively; its construction is illustrated in Figure 2.17, for the example network of Figure 2.16. To a vertex G of $\mathscr{T}$, we add $j \leq n$ edges that are the j non-empty paths $(P_i)_{i \leq j}$ returned by n`-shortest`(G), and j children vertices that are the multigraphs `update`(P_i, G), for which link capacities have been updated to reflect the consumption of resources by P_i. After `update`(P, G), we know that at least one link $l_0 \in P$ has a zero capacity, thus the procedure for building $\mathscr{T}$ eventually terminates; but the depth of

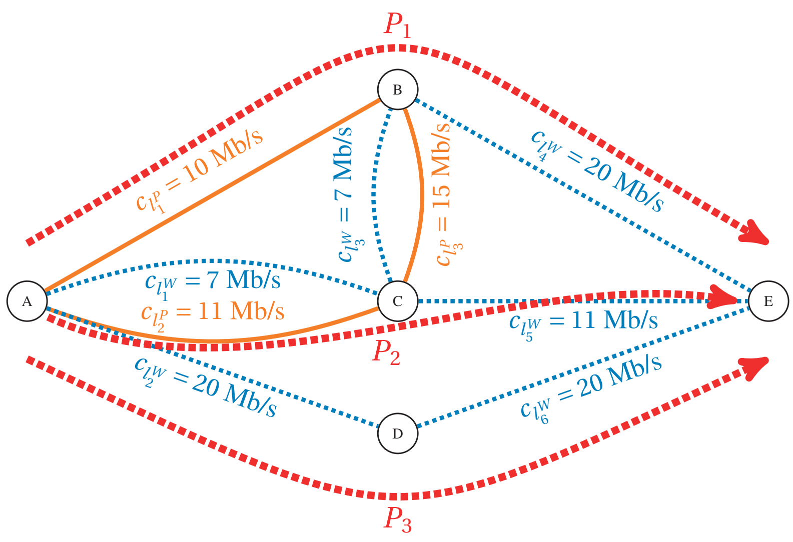


Figure 2.16 – Example of a network with five nodes, two technologies (WiFi with blue dotted lines, PLC with orange plain lines), three PLC links l_1^P, l_2^P, l_3^P and six WiFi links $l_1^W, \dots, l_6^W$. In this example, we consider that all links of a same medium interfere with each other, i.e., $\mathscr{I}\left(l_i^P\right) = \{l_1^P, l_2^P, l_3^P\}$ for all $1 \le i \le 3$ and $\mathscr{I}\left(l_i^W\right) = \{l_1^W, \dots, l_6^W\}$ for all $1 \le i \le 6$.

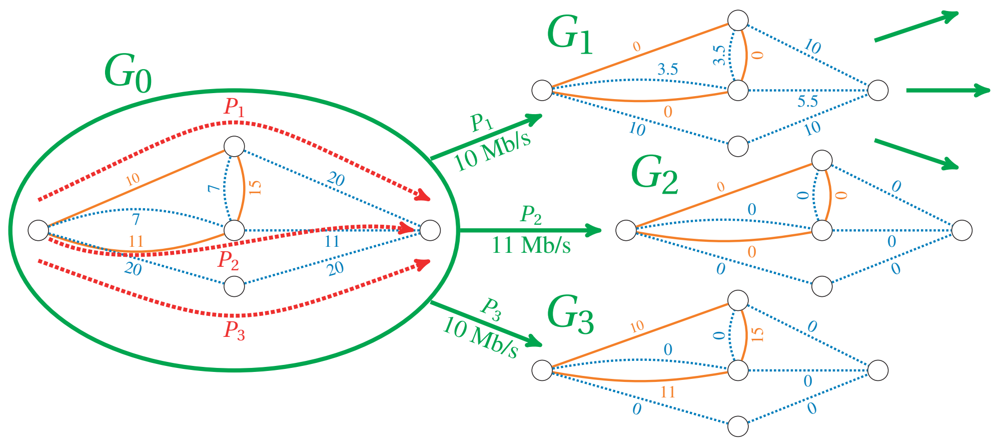


Figure 2.17 – Illustration of the tree construction, using the network multigraph of Figure 2.16, with $n = 3$. Link capacities are in Mb/s. G_1, G_2 and G_3 are multigraphs with capacities updated to reflect the consumption of resources by resp. P_1, P_2 and P_3. There is connectivity between the source and the destination only in G_1, therefore, the construction process continues only from G_1, after which all links have 0 capacity.

$\mathcal{T}$ can, in the worst case, be the total number of links existing in the network, in which case the number of vertices would scale exponentially in the network size. To avoid this problem, the number of paths can be limited by stopping the `update` process along a branch of $\mathcal{T}$, once this branch has reached a certain depth: In Section 2.3, we have shown that limiting the number of paths M to the number of technologies K barely harm the throughput experienced with multipath routing. Here, the number of paths is the depth of the tree, and we stop the procedure once the branch has its depth equal to K. For example, when there are $K = 2$ technologies, we limit the depth of the tree $\mathcal{T}$ to two, which makes this procedure efficient. In the example of Figure 2.17, the three shortest paths in the original network G_0 are P_1, P_2 and P_3. Therefore, G_0 has three children G_1, G_2, and G_3, whose link capacities reflect the potential consumption of P_1, P_2 and P_3, respectively. In G_2 and G_3, there is no path between the source and the destination, and they do not have children in $\mathcal{T}$. In G_1, there are three paths between the source and the destination and G_1 has three children in $\mathcal{T}$.

To each edge P out of a vertex G, we associate a weight equal to the capacity $R(P)$ of P in G. Let $\mathcal{L}$ be the set of multigraph leafs of the tree $\mathcal{T}$. For each multigraph leaf $G \in \mathcal{L}$, the set of edges (i.e., paths of the network) from G_0 to G forms a multipath, denoted by $\mathscr{P}_G$. Then, $C(\mathscr{P}_G)$ defined by

$$C(\mathscr{P}_G) = \sum_{P \in \mathscr{P}_G} R(P) \tag{2.9}$$

is the estimated capacity of the multipath $\mathscr{P}_G$, i.e., the estimated achievable rate when using simultaneously all the paths in the multipath $\mathscr{P}_G$. The final multipath returned by the whole routing procedure is $\mathscr{P}^{\max}$ of maximum associated capacity, i.e., $\mathscr{P}^{\max} = \mathscr{P}_{G^{\max}}$ where

$$G^{\max} = \arg\max_{G \in \mathcal{L}} C(\mathscr{P}_G).$$

In the example of Figure 2.17, the best multipath is $\mathscr{P}^{\max} = (P_1, P_3)$ with $C\left(\mathscr{P}^{\max}\right) = 15$ Mb/s. With this method, the number of paths returned by the routing algorithm depends on the number of edges (i.e., paths) in the branch of the leaf $G^{\max}$. This is a desirable feature, as it means that the number of paths in the multipath depends on the network topology, and that additional paths are considered only if they provide additional gain.

If we limit our routing protocol to returning only one path, we do *not* necessarily find the path returned by the single-path procedure described in Section 2.4.1. For example, take the network depicted in Figure 2.18. We assume that all links of a technology interfere with all other links of this technology (the network is multi-complete). P_1 is the path such that $\Lambda(P_1) = \left\{l_1^P, l_2^W, l_3^W\right\}$ and P_2 is the path such that $\Lambda(P_2) = \left\{l_1^W, l_2^P\right\}$. The weight of P_1 for the single-path procedure of Section 2.4.1 is

$$W(P_1) = \frac{1}{30} + \frac{1}{20} = \frac{5}{60}.$$

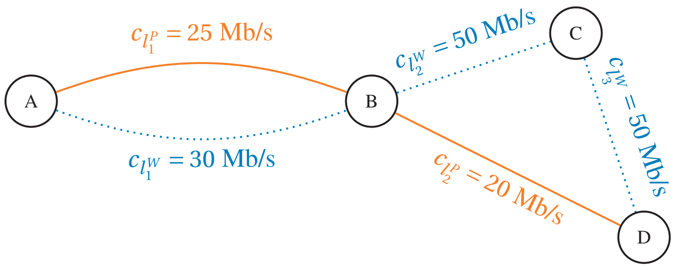


Figure 2.18 – Network example with WiFi (dotted lines) and PLC (plain lines) that illustrates the sub-optimality of the single-path procedure described in Section 2.4.1. P_1 is the path such that $\Lambda(P_1) = \{l_1^P, l_2^W, l_3^W\}$ and P_2 is the path such that $\Lambda(P_2) = \{l_1^W, l_2^P\}$.

The weight of P_2 is

$$W(P_2) = \frac{1}{25} + \frac{1}{50} + w_{ns}(c) + \frac{1}{50} = \frac{1}{25} + \frac{3}{50} = \frac{1}{10}.$$

Consequently, as $W(P_1) < W(P_2)$, the single-path procedure will return P_1.

In contrast, if we apply the multipath procedure described in this section, we compute the maximum achievable rate on both paths

$$R(P_1) = \left(\max(\frac{1}{30}, \frac{1}{20})\right)^{-1} = 20 \text{ and } R(P_2) = \left(\max(\frac{1}{25}, \frac{1}{50} + \frac{1}{50})\right)^{-1} = 25.$$

These values are the actual path capacities. Our multipath procedure will correctly return P_2, which has a capacity of 25 Mb/s, higher than the capacity of P_1 (20 Mb/s). In fact, the single-path procedure tends to favor short paths that might sometimes be less efficient.

As explained in Section 2.1, the performance gains obtained with multipath routing are important only if the rate at which traffic is sent on each path is carefully selected. For this reason, in the next chapter, we present a novel multipath congestion controller. In the experimental evaluation of Section 3.5, we show that our multipath-routing protocol performs better than the single-path procedure and than other multipath-routing protocols. This comes at the cost of an increased complexity, because we need to build the tree $\mathcal{T}$ and to estimate the capacity for each leaf of $\mathcal{T}$. However, our protocol remains practical: On the routers of our testbed (the routers are described in Section A in the Appendix), with $n = 5$, the paths are computed in about 20 ms on average, with a maximum value of about 40 ms. The routing protocol is not responsible for dealing with short-term variabilities that are handled by the congestion controller, described in the next chapter; the paths need to be recomputed only when there is a link failure or a large capacity variation, which occurs infrequently (order of minutes or hours). Each source can also compute in advance (before a flow is initiated) the multipath towards each destination, which is linear in the number of nodes and is reasonable in a local network where there are typically only a few tens of nodes. Hence this computation time is not an issue in practice.

2.5 Related Work

Multipath routing has been widely studied in several contexts: mobile ad-hoc networks (MANETs) [TTAE09], wireless sensor networks (WSNs) [AKK04], mesh networks [TM06] and traffic engineering [LC02]. In MANETs and WSNs, multipath-routing protocols have been shown to have several advantages, such as reduced delays and better reliability and throughput, and protocols have been proposed both with a single wireless channel [GGSE01, LLZ06] and with multi-channel wireless networks [DQZ$^+$15, GRS11, THT08]. These protocols use heuristics to build the paths and they are consequently not guaranteed to be optimal. In addition, they mostly look for maximally disjoint paths, and do not try to optimize throughput as we do in Section 2.4. Similarly to our protocol, they use heuristics and do not guarantee optimality. Multipath routing has recently received renewed attention, in particular with the development of multipath TCP (MPTCP) [FRH$^+$11, FRHB13]. For practical implementations of multipath (e.g., with MPTCP), the set of paths is chosen in advance and congestion control is then carried on these chosen paths.

Optimal multipath routing and scheduling have also been studied in several works, mostly at a theoretical level [LS06, NML08, ZWT$^+$10]. These papers do not study the optimal number of paths; rather, they find the optimal rate provided by doing joint routing and scheduling. To the best of our knowledge, our work presented in Section 2.3 is the first to address the question of finding the optimal number of paths when using multipath routing in hybrid mesh networks with shared-medium technologies.

2.6 Summary

After introducing the problem and describing our network model, we have presented in Section 2.3 analytical results that, for certain classes of mesh networks that include typical home networks, give bounds on the optimal number of paths when using multipath routing in hybrid networks with shared-medium technologies. They show that hybrid networks and multipath routing are intrinsically related and that the optimal number of paths M^{opt} is tightly linked with the number K of non-interfering technologies. We have verified these analytical results with simulations and experiments on a three-technology testbed. We have also presented numerical and experimental results for more general networks. These results show that for general networks, the optimal number of paths M^{opt} remains close to the number of technologies K, and that the rate loss incurred by using at most K paths is very small. This finding has a practical consequence of importance: It means that in home or enterprise networks with K distinct shared-medium technologies (e.g., PLC, WiFi with 2.4 GHz and 5 GHz, LTE), limiting a multipath-routing protocol to multipaths of at most K paths does not harm significantly the performance of the protocol. Building on these results, we have introduced in Section 2.4 a novel multipath-routing algorithm that efficiently computes combinations of paths for simultaneous use. As opposed to existing work that looks for maximally disjoint paths, our multipath-routing protocol explicitly aims at maximizing throughput.

3 A Multipath and Interference-Aware Congestion Controller

3.1 Introduction

In this chapter,[1] we assume that for any flow (source-destination pair), one multipath has been computed by the source. For example, a multipath is computed by using the practical multipath-routing protocol described in the previous chapter (Section 2.4). As already mentioned in Section 2.1, saturating multi-hop paths is inefficient and can lead to congestion collapse with packet losses and instabilities [GSK04, Sri04]. As we will show in this chapter, it is particularly true with multipath routing. Consequently, in order to reach the optimal throughput, the amount of traffic to be sent on each path of the multipath needs to be carefully selected. In this chapter, we present a distributed multipath congestion-controller that decides the amount of traffic to inject over each path. Its goal is to maximize aggregate network utility (i.e., a chosen performance/fairness tradeoff), while avoiding congestion (thus keeping the queues stable). Because we consider shared-medium technologies, our congestion controller must take into account the interference generated by neighboring links. We extensively evaluate this multipath congestion-controller as well as the multipath-routing protocol presented in Section 2.4 by implementing them on a hybrid PLC/WiFi testbed.

We also evaluate with testbed experiments the throughput gains offered by hybrid networks. We confirm that, as expected, using two technologies instead of one significantly improves the throughput by aggregating the capacities of the technologies. More interestingly, we also show that using different technologies such as PLC and WiFi, as opposed to using two WiFi channels (multi-channel WiFi), has the potential for additional performance improvements. With multi-channel WiFi, the medium quality is similar in all channels, even if they are orthogonal, as fading or other channel characteristics have a similar impact in all channels. Thus, link capacities or link failures in different channels are correlated. Combining PLC and WiFi brings spatial and temporal diversity, which enables further performance improvements in terms of throughput, coverage, and reliability.

[1] This chapter is based on the paper by Henri et al. [HVHT16].

Outline of the Chapter This chapter is structured as follows. In Section 3.2, we extend the model described in Section 2.2. In Section 3.3, we present our multipath congestion-controller. In Section 3.4, we extensively evaluate through simulations this congestion controller, as well as the multipath-routing protocol presented in Section 2.4. In Section 3.5, we describe the implementation of our algorithms and present extensive testbed experiments that show that, compared to using two orthogonal WiFi channels, using hybrid PLC/WiFi yields significant throughput improvements. We discuss related work in Section 3.6 and close the chapter with a summary in Section 3.7.

3.2 Model and Notations

We extend the model described in Section 2.2 with the following notations. We define $\mathscr{F}$ as the set of flows, where each flow is a source-destination pair (and can potentially employ several paths), and $\mathscr{P}^{\mathscr{F}}$ as the set of all paths (across all flows). The set of paths that go through a link l is denoted by $\Pi(l) \subset \mathscr{P}^{\mathscr{F}}$. We denote by $\boldsymbol{B}$ the binary $L \times L$ interference matrix indexed by the links, such that $[\boldsymbol{B}]_{l_i,l_j} = [\boldsymbol{B}]_{l_j,l_i} = 1$ if links l_i and l_j interfere with each other (i.e., $l_i \in \mathscr{I}_{l_j}$ and $l_j \in \mathscr{I}_{l_i}$), otherwise 0. Let P^F be the total number of paths in the network (across all flows). We define the binary $L \times P^F$ routing matrix $\boldsymbol{R}$ index by the links and the paths, such that $[\boldsymbol{R}]_{l,P} = 1$ if path P goes through link l, and 0 otherwise. Finally, with each path $P \in \mathscr{P}^{\mathscr{F}}$, we associate x_P, the traffic rate that the source of P sends over path P. We define the vector $\boldsymbol{x} = [x_P]_{P \in \mathscr{P}^{\mathscr{F}}}$. The congestion controller decides the value of x_P for each P.

To avoid congestion, the rate injected on each path should be less than what can be accepted by each intermediate link (each of which might be subject to interference). We quantify this constraint in terms of airtime: From Equation (2.1), the airtime μ_l consumed by an unsaturated link l is given by

$$\mu_l = d_l \sum_{P \in \Pi(l)} x_P.$$

A sufficient constraint to keep the queues finite is to require that the traffic intensities satisfy

$$\sum_{l' \in \mathscr{I}_l} d_{l'} \sum_{P \in \Pi(l')} x_P \quad \leq 1 \quad \forall l \in \mathscr{E}. \tag{3.1}$$

This requires that the *airtime demand* does not exceed 100% in each interference domain. For example, consider again the network of Figure 2.16 on page 31 and assume that P_2 and P_3 carry some traffic (sent by node A at rates x_2 and x_3, respectively). Because $\mathscr{I}_{l_2^W} = \mathscr{I}_{l_5^W} = \mathscr{I}_{l_6^W}$ in this example, the constraints defined by Equation (3.1) for the three WiFi links l_2^W, l_5^W and l_6^W are the same and are given by $x_3/20 + x_3/20 + x_2/11 \leq 1$. For instance, if $x_2 = 5.5$ MB/s, links l_2^W and l_6^W need to share 50% of the airtime, and Constraint (3.1) specifies that x_3 cannot be more than $20/4 = 5$ Mb/s.

In practice, when the airtime approaches 1, the delays increase rapidly; for this reason, we might want to place a more conservative constraint

$$\sum_{l' \in \mathscr{I}_l} d_{l'} \sum_{P \in \Pi(l)} x_P \quad \leq 1 - \delta \quad \forall l \in \mathscr{E}, \tag{3.2}$$

with a *constraint margin* δ in $[0,1]$.

Constraints (3.1) and (3.2) are conservative: The airtime could be reused by some links of an interference domain if there is no pairwise interference between them. However, Constraints (3.1) and (3.2) have the crucial advantage of being formulated locally (with respect to each link), making it possible to use them for distributed load-balancing optimization; this is the key basis for this formulation of conservative constraints. In Section 3.4, we study the effect of this conservative policy and compare our scheme with optimal solutions based on perfect centralized scheduling. Finally, note that Constraint (3.1) can be written in matrix form as

$$\big(\boldsymbol{B} \cdot \operatorname{diag}(\boldsymbol{d}) \cdot \boldsymbol{R}\big)\, \boldsymbol{x} \leq \boldsymbol{1}, \tag{3.3}$$

where $\boldsymbol{d} = [d_l]_{l \in \mathscr{E}}$ denotes the vector of link transmission delays.

3.3 A Distributed and Interference-Aware Congestion Controller

We first address the single-path case, where there is exactly one path between any source-destination pair, before moving to the extension to multiple paths.

3.3.1 Congestion Control on a Single Path

We consider in this section that a single path is used by each flow: There is a one-to-one mapping between paths and flows ($|\mathscr{F}| = |\mathscr{P}^{\mathscr{F}}|$). To each path $P \in \mathscr{P}^{\mathscr{F}}$ we attach an increasing and strictly concave utility function $U_P : \mathbb{R}^+ \to \mathbb{R}^+$; it describes the benefit that the source of P gets by sending traffic at rate x_P. We formulate an optimization problem similar to what was proposed for wired networks [KMT98, LL99]. However, because they focus on wired networks, these works do not account for interference. To address this issue, we introduce the more general interference constraint given by Equation (3.1), and we replace the constraints on the link *capacities* with constraints on the *airtimes* of the interference domains. The problem then reads

$$\max_{\boldsymbol{x}} \sum_{P \in \mathscr{P}^{\mathscr{F}}} U_P(x_P) \tag{3.4}$$

$$\text{subject to} \sum_{l' \in \mathscr{I}_l} d_{l'} \sum_{P \in \Pi(l')} x_P \quad \leq 1 \qquad \forall l \in \mathscr{E}, \tag{3.5}$$

$$x_P \geq 0 \qquad \forall P \in \mathscr{P}^{\mathscr{F}}. \tag{3.6}$$

From Equation (3.3), the constraint given by Equation (3.5) is linear. The problem is thus the maximization of a strictly concave function over a convex set.

The Lagrange dual is

$$D(\boldsymbol{\gamma}) = \max_{\boldsymbol{x}} \sum_{P \in \mathscr{P}^{\mathscr{F}}} U_P(x_P) - \sum_{l \in \mathscr{E}} \gamma_l \left(\sum_{l' \in \mathscr{I}_l} d_{l'} \sum_{P \in \Pi(l')} x_P - 1 \right),$$

where γ_l is the Lagrange multiplier associated with the interference domain of link l.

Taking the partial derivatives with respect to x_P, the first-order condition is

$$U'_P(x_P) = \sum_{l \in \mathscr{E}} \gamma_l \sum_{l' \in \mathscr{I}_l} d_{l'} \boldsymbol{R}_{l',P} = \sum_{l \in P} d_l \sum_{l' \in \mathscr{I}_l} \gamma_{l'},$$

where for the second equality we have used the equivalence $l' \in \mathscr{I}_l \Leftrightarrow l \in \mathscr{I}_{l'}$ for any two links l, l'. Let $q_P \doteq \sum_{l \in P} d_l \sum_{l' \in \mathscr{I}_l} \gamma_{l'}$ be the sum of dual variables over all interference domains along path P. The first order condition becomes $x_P = U'^{-1}_P(q_P)$.

The dual problem is given by

$$\min_{\boldsymbol{\gamma}} D(\boldsymbol{\gamma}). \tag{3.7}$$

Due to the convex nature of the problem, there is no duality gap and a gradient descent procedure is guaranteed to find a global optimum in the fluid limit. We define $y_l \doteq \sum_{l' \in \mathscr{I}_l} d_{l'} \sum_{P \in \Pi(l')} x_P$. This quantity represents the total *airtime demand* in the interference domain of link l. After some computations, we find

$$\frac{\partial D}{\partial \gamma_l} = 1 - y_l.$$

We assume that time is slotted, and we denote by $f(t)$ the value of a quantity f at time slot t. Applying gradient descent, we obtain the following corresponding discrete time controller for the paths rates and interference domain dual variables:

$$y_l(t) = \sum_{l' \in \mathscr{I}_l} d_{l'} \sum_{P \in \Pi(l')} x_P(t), \tag{3.8}$$

$$\gamma_l(t+1) = \left[\gamma_l(t) + \alpha_t(y_l(t) - 1)\right]^+, \tag{3.9}$$

$$q_P(t) = \sum_{l \in P} d_l \sum_{l' \in \mathscr{I}_l} \gamma_{l'}(t), \tag{3.10}$$

$$x_P(t+1) = U'^{-1}_P\left(q_P(t)\right), \tag{3.11}$$

with $\alpha_t > 0$ a sequence of step sizes, and $[y]^+ = y$ if $y > 0$ and 0 otherwise. It can be shown [LS04] that if the step sequence α_t is such that

$$\alpha_t \to 0 \text{ as } t \to \infty, \text{ and } \sum_t \alpha_t = \infty,$$

then the the sequence $q_P(t)$ converges to q_P^* as $t \to \infty$, where q_P^* is an optimizer of Equation (3.7). Conversely, because the mapping between q_P and x_P is continuous, the rate allocation vector $[x_P(t)]_{P \in \mathscr{P}^{\mathscr{F}}}$ converges to the optimal solution of Problem (3.4)–(3.6). In our implementation, we use a fixed step size $\alpha_t = \alpha$ in order to continuously adapt to changes in the network. Again, it can be shown [LS04] that if α is small enough, the rate allocation converges to small neighborhood of the optimizer of Problem (3.4)–(3.6).

Although the optimization objective is computed globally over the sum of all users' utilities, a practical and lightweight controller can be built in a distributed fashion. Each node in the network monitors the traffic that it forwards and measures the airtime demand

$$\mu_l = d_l \sum_{P \in \Pi(l)} x_P$$

on each of its egress links l. For each technology $k \in \{1, \ldots, K\}$, the node computes (i) the aggregate airtime demand by summing the airtime demands μ_l over each egress link l employing technology k, and (ii) the sum of the dual variables γ_l for each egress link l employing technology k. The node periodically broadcasts over the technology k a packet that contains these two values. All the nodes in the interference domains of the outgoing links that overhear these broadcasts compute y_l for each of their own outgoing link l of technology k. They do so by adding (i) their own airtime demands for their egress links employing technology k, and (ii) the airtime demands received by all their neighbors, as described by Equation (3.8). Knowing y_l, the nodes then update γ_l using Equation (3.9). Finally, when forwarding a packet on link l, the nodes add the current value of $d_l \sum_{l' \in \mathscr{I}_l} \gamma_{l'}$ to a dedicated field in the packet's 2.5-layer header (see Section 3.5 for more details). At the destination of path P, the value of this field is thus equal to q_P as given by Equation (3.10), and the destination can send back q_P to the source, via an acknowledgement. Upon reception of q_P, the source updates the rate x_P for path P using Equation (3.11). This mechanism requires only local measurements and collaboration with a small communication overhead among the nodes. These properties enable us to implement this algorithm on a real testbed, as described in Section 3.5.

3.3.2 Multipath Congestion Control

In this section, we present the extension of our interference-aware congestion controller to multiple paths. We now differentiate between *flows* and *paths*: Each flow can potentially employ several paths. To express the availability of path $P \in \mathscr{P}^{\mathscr{F}}$ to flow $f \in \mathscr{F}$, we write $P \in f$. The flow of a path P is denoted by $f(P)$. The utility obtained by each flow f is now given by $U_f(x_f)$ with $x_f = \sum_{P \in f} x_P$. U_f is strictly concave in x_f, but the main challenge comes from the fact that the objective function is *not* strictly concave in $\boldsymbol{x} = [x_P]_{P \in \mathscr{P}^{\mathscr{F}}}$ anymore. To overcome this problem, we adopt the approach of *proximal optimization*, introduced by Wang et al. [WPL03] when no interference is present, and we maximize another objective function, which has the same optimizer as the original:

$$\max_{\boldsymbol{x},\bar{\boldsymbol{x}}} \sum_{f \in \mathscr{F}} \left(U_f \left(\sum_{P \in f} x_P \right) - \frac{1}{2} \sum_{P \in f} (x_P - \bar{x}_P)^2 \right) \tag{3.12}$$

$$\text{subject to} \sum_{l' \in \mathscr{I}_l} d_{l'} \sum_{P \in \Pi(l')} x_P \quad \leq 1, \qquad \forall l \in \mathscr{E}, \tag{3.13}$$

$$x_P \geq 0, \qquad \forall P \in \mathscr{P}^{\mathscr{F}},$$

where $\bar{x}_P$ is an auxiliary variable for path P and $\bar{\boldsymbol{x}} = [\bar{x}_P]_{P \in \mathscr{P}^{\mathscr{F}}}$. Subtracting a quadratic term strictly convex in $\boldsymbol{x}$ in Equation (3.12) makes the objective function strictly concave in $\boldsymbol{x}$ [WPL03]. The optimization is now performed over the two variables $\boldsymbol{x}$ and $\bar{\boldsymbol{x}}$. The new objective function given by Equation (3.12) has the same optimizer $\boldsymbol{x} = \bar{\boldsymbol{x}}$ as the original one given by Equation (3.4). We obtain the following discrete-time multipath congestion controller:

$$x_P(t+1) = \left[(1-\alpha) x_P(t) + \alpha \left(\bar{x}_P(t) + U'_{f(P)} \left(\sum_{P' \in f(P)} x_{P'}(t) \right) - q_P(t) \right) \right]^+$$

$$\bar{x}_P(t+1) = (1-\alpha)\bar{x}_P(t) + \alpha x_P(t),$$

with $y_l(t)$, $\gamma_l(t+1)$, and $q_P(t)$ given by respectively Equations (3.8), (3.9), and (3.10).

Accounting for interference does not change the linearity of Equation (3.13), and the convergence of this controller can be established using similar techniques as Lin and Shroff [LS06]. This controller employs the same update rules as the single-path controller for γ_l, y_l, and q_P. In addition, the update rule for x_P depends only on q_P and $\bar{x}_P$. Thus, it can be implemented in a distributed way, exactly as the single-path controller.

This controller is able to account for external interference (i.e., interference that comes from nodes that do not use the controller): Nodes can measure traffic from external nodes and add the corresponding airtimes in Equation (3.8). However, because the controller has no control over external nodes, it converges to the optimal allocation under this external load, which means that the clients who do not use our controller are not affected, except during a short transition phase, by the clients who do. Although this allocation does not affect external nodes, it is not necessarily fair: For example, if one external node saturates WiFi, the controller converges to an allocation that never uses WiFi. Extending our controller to support external interference in a fair way is out of the scope of this dissertation.

The main downside of this controller is that it takes some time to converge (see Sections 3.4 and 3.5). For this reason, it is designed mostly for long-run best-effort flows (e.g., large-file download or video streaming). Yet, in Section 3.5, we show that it helps also for short flows.

3.4 Numerical Evaluation

In the next sections of this chapter, we evaluate jointly the multipath-routing protocol described in Section 2.4 and the controller described in Section 3.3. The joint algorithm is called EMPoWER: With EMPoWER, the congestion controller is used on the multipath computed with the multipath-routing protocol.

Before validating the performance gains of EMPoWER via testbed experiments in Section 3.5, we evaluate its performance via simulations in a controlled environment, over several thousand instances of randomly generated networks. We compare it with other algorithms, including the optimal algorithm based on backpressure, proposed by Neely et al. [NML08] and described in Section 2.3.3. As explained in Section 2.3.3, implementing this optimal algorithm would be very challenging, because it includes an optimal NP-hard [GNT06] scheduling algorithm, that (i) requires solving a centralized optimization problem at each time step, and (ii) requires modifying the underlying MAC layer.

3.4.1 Simulation Settings

We write a packet-level simulator in Matlab.[2] We simulate three different network topologies.

The first topology is a typical residential network, that we evaluate in two modes, infrastructure and mesh. In the infrastructure mode, we drop uniformly at random on a 20×10 m rectangle a single hybrid PLC/WiFi router and two client devices, one PLC/WiFi device (e.g., a desktop computer, a connected television) and one device with only WiFi (e.g., a mobile phone, a laptop). In the mesh mode, there are three hybrid PLC/WiFi routers, one PLC/WiFi device and one device with only WiFi. For both modes, the source is chosen among the two client devices (either a hybrid device or a WiFi-only device) and the destination is chosen among the hybrid routers (in the infrastructure mode, there is only one).

The two other topologies are typical enterprise networks (e.g., company, hospital, university). They are evaluated in the mesh mode. For the first enterprise technology, we drop ten nodes on a 50×30 m rectangle; for the second enterprise technology, we drop ten nodes on a 100×60 m rectangle. Half of the nodes are PLC/WiFi APs and are randomly located on a 10×10 m grid (values close to what we observe on the managed WiFi network of our building). The remaining nodes have only single-channel WiFi and are placed uniformly at random. In the two enterprise network scenarios, the source of a flow is chosen among all the nodes, and the destination is chosen among the PLC/WiFi APs, in both cases uniformly at random (we consider that there is no flow between two WiFi-only nodes). Because EMPoWER focuses on long-run applications, such as large-file downloads or video streaming, typically not well supported by moving nodes, we assume all nodes to be static.

[2]The source code of our simulator and of our Click implementation described in Section 3.5.1 is available at https://c4science.ch/diffusion/1252/empower.git.

A link between two nodes A and B exists if the distance between A and B is smaller than a given connection radius R. Based on measurements from our indoor testbed and reported in Section A.3 in the Appendix, R is set to 35 m for WiFi, and to 50 m for PLC. Additionally, a PLC link exists only when two nodes are connected to the same *central coordinator* [Plc10], in particular, when two nodes are on the same electrical panel. In our building, we have two electrical panels that distribute power over two equal parts. We assume that buildings of 100×60 m typically employ two panels and therefore, for the 100×60 enterprise topology, we divide the building area in two equal parts; a PLC link exists only if both nodes are in the same part of the building. The capacities of WiFi and PLC links are then sampled from a distribution close to the capacity distributions measured on our real testbed, reported in Section A.3 in the Appendix. As previously noted [VHT15], the capacities for WiFi and PLC with the technologies used, respectively 802.11n and HPAV 200, are similar. This is illustrated further in Table A.1 in the Appendix that presents the statistics of the capacities of all the links of our testbed and where we see that the average capacity is 31 Mb/s for WiFi and 30 Mb/s for PLC. When employing two non-interfering WiFi channels, we consider for the simulations that the two channels have the same bandwidth, consequently the same link capacities.

In this section and Section 3.5, we use the proportional fairness utility function, given by $U_f(x_f) = \log(1 + x_f)$, for each flow f. This metric, extensively used in the literature, quantifies how well an algorithm tunes the "throughput vs. fairness" tradeoff. The underlying MAC scheduling is simulated through a simplified version of CSMA/CA, with perfect sensing and no back-off. Our statistics are computed over 1000 simulation runs with different random seeds, i.e., with different topologies each time.

To quantify separately the gains provided by (i) the use of two different technologies, by (ii) our multipath-routing protocol, and by (iii) the congestion-control algorithm (CC), we evaluate in this section and Section 3.5 several combinations of algorithms and scenarios:

- **EMPoWER** Multipath routing, CC, PLC/WiFi,
- **SP** Single-path routing, CC, PLC/WiFi,
- **MP-WiFi** Multipath routing, CC, single-channel WiFi,
- **SP-WiFi** Single-path routing, CC, single-channel WiFi,
- **MP-mWiFi** Multipath routing, CC, two-channel WiFi,
- **MP-w/o-CC** Multipath routing, no CC, PLC/WiFi,
- **SP-w/o-CC** Single-path routing, no CC, PLC/WiFi,
- **MP-2bp** Naive multipath routing returning two best paths (2-`shortest`), CC, PLC/WiFi.
- **MP-W+P** Multipath routing returning the best WiFi-only path and the best PLC-path, CC, PLC/WiFi.

WiFi is available in each scenario to provide access to mobile clients. SP and SP-WiFi use the single-path routing protocol presented in Section 2.4.1.[3] When using only WiFi, the CSC is set to 0. EMPoWER, MP-WiFi, MP-mWiFi, and MP-w/o-CC use the multipath-routing protocol presented in Section 2.4. We use $n = 5$ for n-`shortest` (as described in Section 2.4.2), which enables path diversity while limiting the number of possible combinations to be explored. MP-2bp is a naive multipath protocol that returns the two best paths of the single-path procedure described in Section 2.4.1. MP-W+P is a multipath protocol that returns two paths, the best WiFi-only path and the best PLC-only path (when it exists); in our hybrid networks with two different technologies, this corresponds to two maximally disjoint paths.

We then compare EMPoWER with the optimal backpressure algorithm [NML08] that we briefly described in Section 2.3.3. EMPoWER differs from the optimum for two reasons: (i) It employs preselected paths, whereas the optimal scheme finds the optimal paths using backpressure, and (ii) it uses the conservative constraint given by Equation (3.1). For this reason, we show results of this backpressure scheme in two different scenarios

- with an optimal centralized scheduler that yields the best theoretical throughput. But this is impractical and would not be stably supported by real-world mechanisms such as CSMA/CA. This scheme is denoted by *optimal*.
- with a centralized scheduler that gives the optimal result under the condition given by Equation (3.1) used by our congestion controller. This scheme is denoted by *conservative opt*.

Comparing *conservative opt* and EMPoWER enables us to evaluate the performance of the multipath-routing protocol, as they both use the constraint given by Equation (3.1).

3.4.2 Simulation Results

We first consider scenarios with one flow, and then move to a scenario with several contending flows.

Mesh vs. Infrastructure Mode

We first compare, for the residential topology, the optimal rate obtained with the mesh and infrastructure modes. The cumulative distribution of the rates for each mode is presented in Figure 3.1. We show clearly that using the mesh mode significantly improves throughput, especially when the rate in the infrastructure mode is low: With this topology, the lowest rate is 5 Mb/s with the infrastructure mode and 40 Mb/s with the mesh mode, i.e., it is 8x higher with the mesh mode. This shows that, as expected, the mesh mode improves throughput and coverage. In the following, we only evaluate the topologies in the mesh mode.

[3]We also implemented other single-path procedures employing different metrics, such as IRU [YWK05], ETT [DPZ04], and CATT [GS08]; all gave worse results in our experiments.

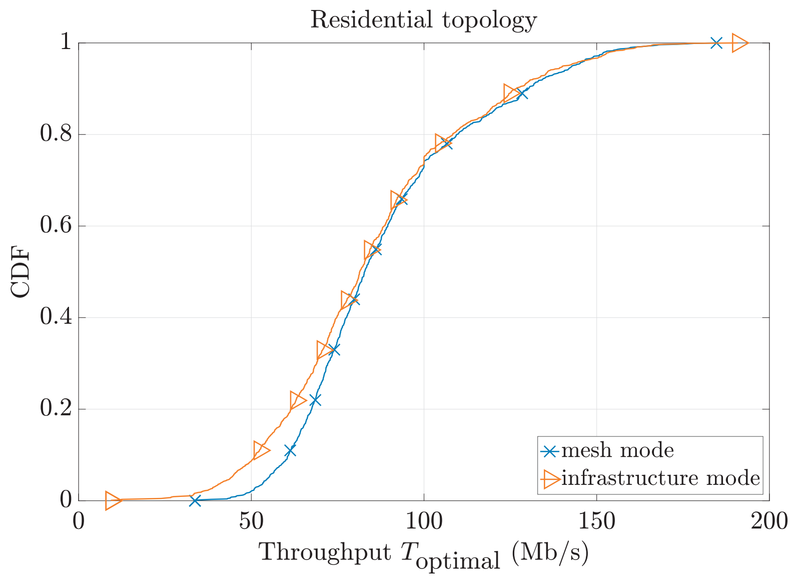


Figure 3.1 – Distribution of T_{optimal} in mesh and infrastructure modes.

Hybrid vs. Single-Technology Networks

We compare the performance of EMPoWER with that of the other schemes. Here, the congestion controller is always used; the only differences are the technologies employed (WiFi with or without PLC), and the number of paths returned by the routing protocol (one for SP, two for MP). The results are presented in Figure 3.2, where we show the empirical cumulative distribution of the flow throughput T_X achieved by each scheme X for residential (left) and enterprise (center and right) topologies. For the sake of clarity, we do not show the results of MP-WiFi, as they coincide in all cases with those of SP-WiFi: This confirms that only if there are two or more non-interfering technologies does multipath improves the throughput, as indicated by the results of Section 2.3. Hybrid networks and multipath routing both contribute significantly to performance gains. In the residential topology, the average gain compared to WiFi alone is 88%, and it is 28% compared to hybrid single-path. In the enterprise topologies, the average gains compared to WiFi alone are 59% (50×30) and 68% (100×60), and they are 39% (50×30) and 31% (100×60) compared to hybrid single-path.

We now compare PLC/WiFi with multi-channel WiFi (with two non-interfering channels). Because the two WiFi channels have the same capacities, $T_{\text{MP-mWiFi}} = 2T_{\text{SP-WiFi}}$. Figure 3.2 shows that on average, EMPoWER and MP-mWiFi are very close. However, it also shows that multi-channel WiFi does better primarily when the throughput is already good (the EMPoWER curve is above the MP-mWiFi curve for small throughput, and below for large throughput). This is because WiFi typically has a greater capacity than PLC at short range [VHT15]. In contrast, PLC/WiFi performs better on flows with outage or poor connectivity. In fact, PLC/WiFi, compared to multi-channel WiFi, improves network coverage. In Figure 3.3, we present the cumulative distribution of the ratio $T_{\text{MP-mWiFi}} / T_{\text{EMPoWER}}$ for the *worst flows*: We call worst flows

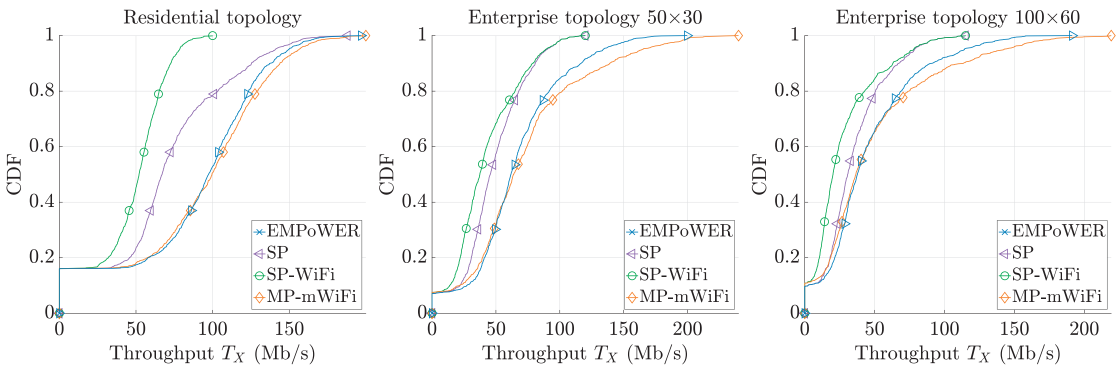


Figure 3.2 – Distribution of T_X for different schemes X. Residential (left) and enterprise (center and right) topologies.

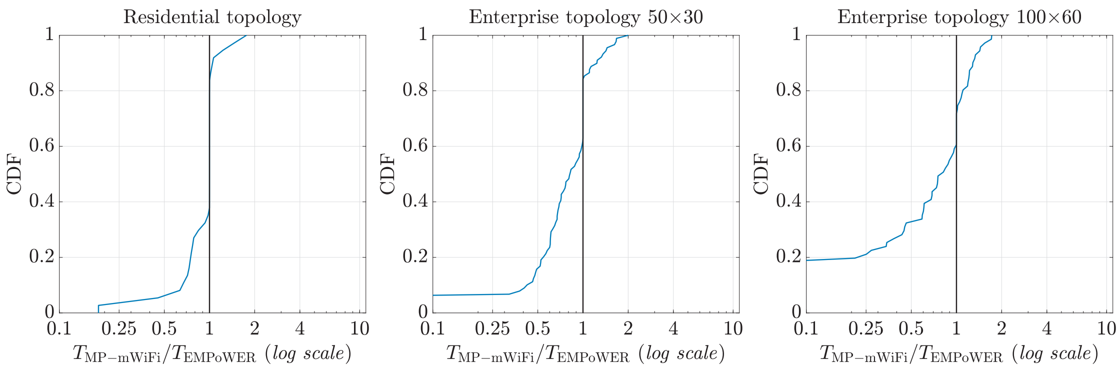


Figure 3.3 – Distribution of $T_{\text{MP-mWiFi}}/T_{\text{EMPoWER}}$, worst flows. Residential (left) and enterprise (center and right) topologies.

the bottom-20% of the flows with respect to the minimal throughput $\min(T_{\text{MP-mWiFi}}, T_{\text{EMPoWER}})$. We remove the cases where neither EMPoWER nor MP-mWiFi have connectivity. For the worst flows, EMPoWER performs better than MP-mWiFi, with about 60% of the flows having higher throughput, up to 4x; our experiments on a real testbed (see Section 3.5) even show improvements up to 10x. In some cases (15% to 25%), MP-mWiFi does better, but the maximum throughput improvement over EMPoWER is only 1.7x. In the enterprise topology, PLC/WiFi even brings connectivity in some cases where multi-channel WiFi does not ($T_{\text{MP-mWiFi}} = 0$ whereas $T_{\text{EMPoWER}} > 0$). This is the case of about 6% of the worst flows in the 50×30 topology, and of 19% in the 100×60 topology.

EMPoWER vs. Optimal Schemes

We evaluate the performance of EMPoWER by comparing it with a backpressure scheme, shown to be optimal [NML08] in two scenarios *optimal* and *conservative opt*, as described in Section 3.4.1. For baselines, we use SP, MP-w/o-CC, MP-2bp, and MP-W+P. In Figure 3.4, we show the empirical cumulative distribution of the ratio $T_X/T_{optimal}$ for different schemes X. EMPoWER achieves results very close to *conservative opt*: In the residential topology, the

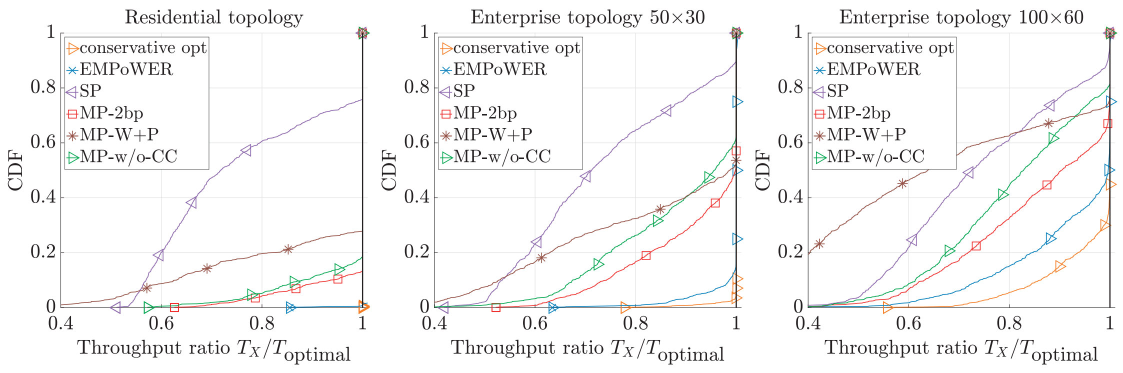


Figure 3.4 – Distribution of $T_X/T_{optimal}$ with one flow. Residential (left) and enterprise (center and right) topologies.

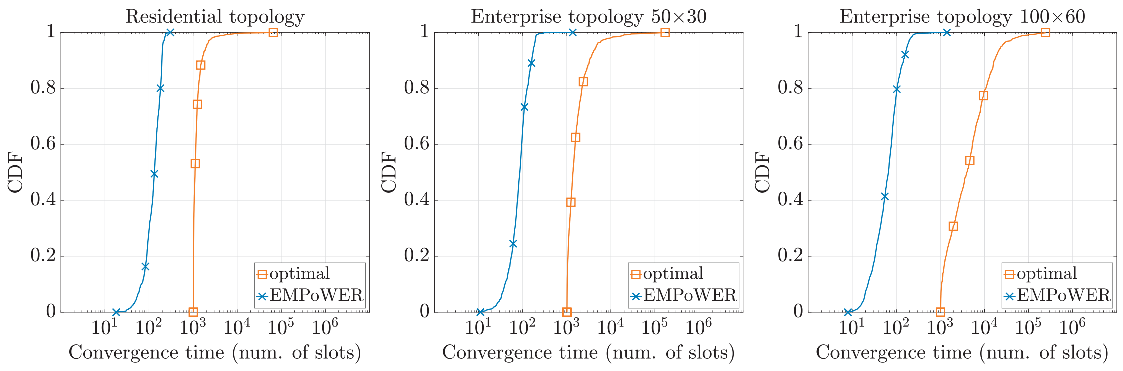


Figure 3.5 – Distribution of convergence time with one flow. Residential (left) and enterprise (center and right) topologies.

performance loss of EMPoWER is less than 10% in 99% of the cases. In the enterprise topology, it is less than 10% in 98% (50×30) and 85% (100×60) of the cases. This shows that our multipath-routing protocol succeeds in finding good multipaths. The performance differences with SP, MP-2bp, and MP-W+P unveil that multipath routing is beneficial, and that finding the good multipaths is not trivial; in addition, the performance of multipath routing strongly depends on the congestion controller. The penalty of using the condition given by Equation (3.1) is not severe in the vast majority of the cases. In total, in the residential topology, EMPoWER achieves optimal throughput in 96% of the cases and, in virtually all cases, the performance loss with respect to *optimal* is less than 15%. Similar results are observed in the enterprise 50×30 topology: EMPoWER achieves optimal throughput in 88% of the cases and, in 99% of the cases, the performance loss with respect to *optimal* is less than 15%. In the enterprise 100×60 topology, the penalty of using the constraint given by Equation (3.1) is a bit higher, but EMPoWER significantly outperforms SP, MP-2bp, and MP-W+P and still achieves optimality in 60% of the cases. In 83% of the cases, the performance loss with respect to *optimal* is less than 15%. Furthermore, Figure 3.5, where we present the number of time slots required to reach the steady-state ("steady" meaning that the throughput is within 1% of the final throughput), shows that EMPoWER provides drastically faster convergence than *optimal* (results for *conservative*

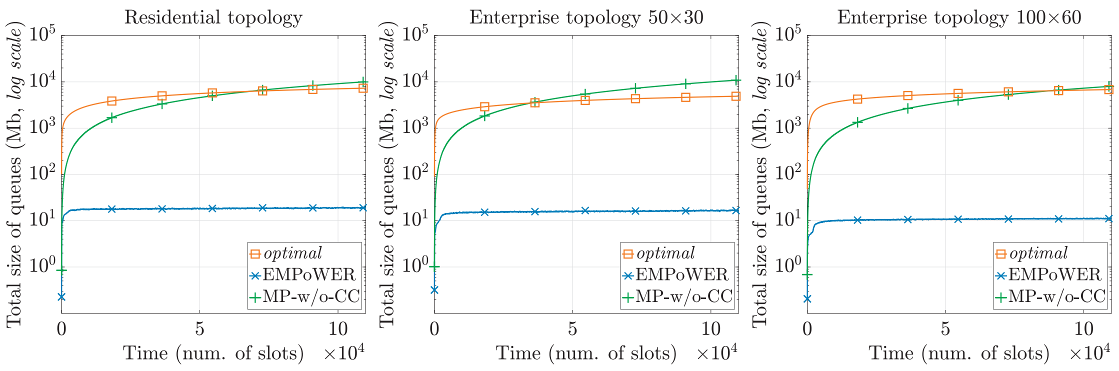


Figure 3.6 – Queue occupancy over time with one flow. Residential (left) and enterprise (center and right) topologies. y-axis is in log scale.

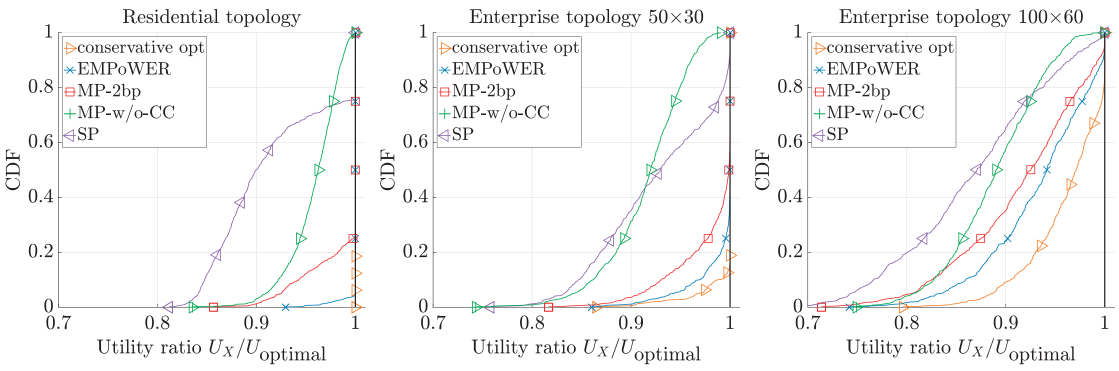


Figure 3.7 – Distribution of $U_X / U_{optimal}$ with several flows. Residential (left, with two flows) and enterprise (center and right, with three flows) topologies.

opt are similar, and they are not shown for clarity's sake), both for residential and enterprise topologies. In fact, *optimal* and *conservative opt* suffer from symptoms of backpressure-based routing; although they are throughput-optimal at steady-state, good paths start being employed only after the queues on the bad paths start to fill up. This phenomenon is confirmed in Figure 3.6, where we plot the sum of all queue sizes in the network as a function of time. Both *optimal* and EMPoWER are stable, but our congestion controller reduces queue occupancy (and thus delay) by more than two orders of magnitude compared to backpressure. Not surprisingly, congestion control is required to keep the queues stable, and the queues grow linearly when it is not employed (see the curve for MP-w/o-CC with the y-axis in logarithmic scale).

Utility Maximization and Proportional Fairness

We now evaluate how EMPoWER distributes network resources among several competing flows. In Figure 3.7, we show the distribution of the total network utility when there are $F = 2$ (for the residential topology) or $F = 3$ (for the enterprise topology) different saturated flows between randomly chosen source-destination pairs, as a proportion of the total utility obtained with *optimal*, denoted by U_{optimal}. The total utility for a scheme X where each flow $1 \le f \le F$ gets a

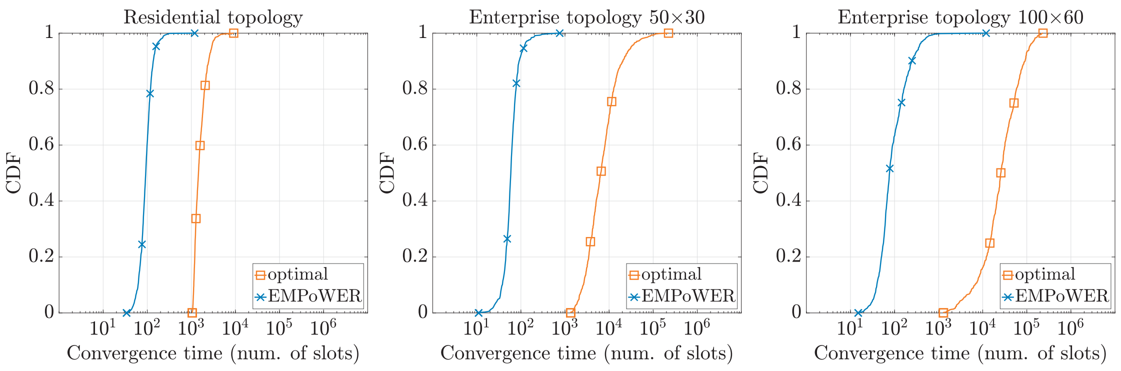


Figure 3.8 – Distribution of convergence time with several flows. Residential (left, with two flows) and enterprise (center and right, with three flows) topologies. x-axis is in log scale.

throughput x_f is given by $U_X = \sum_f \log(1 + x_f)$. Clearly, the gains of employing multiple paths are conditioned on using congestion control. Note that even if our multipath-routing protocol maximizes throughput for a single flow, it also improves performance with respect to to MP-2bp when several flows are competing. Results for the convergence time are presented in Figure 3.8.

Overall, compared to optimal centralized schemes, EMPoWER brings significant gains in convergence and delays, and offers performance close to optimum.

3.5 Practical Implementation and Experimental Evaluation

In this section, we evaluate EMPoWER on a real testbed, and demonstrate its practical usability. We first describe some implementation details, then we show an example of how EMPoWER works. We then evaluate EMPoWER over a large number of random runs, and finally discuss its interactions with TCP.

3.5.1 Practical Implementation

We implement EMPoWER on the testbed described in Section A in the Appendix, by using the *Click Modular Router* [KMC$^+$00] in user space. The main components of EMPoWER are shown in Figure 3.9. We use two non-interfering 40 MHz WiFi bands, one from 5.785 GHz to 5.825 GHz (not used by any other WiFi device) that we call Channel 1, and one from 2.412 GHz to 2.452 GHz that we call Channel 2. To evaluate fairly our algorithm, we avoid external interference: Channel 2 is used by the WiFi network of the university, therefore, we run all multi-channel WiFi experiments at night and we verify that there is no external traffic. For PLC/WiFi experiments, PLC and Channel 1 are used.[4] Link capacities with WiFi and PLC are comparable (see Section A.3 in the Appendix), which means that PLC/WiFi and multi-channel WiFi have comparable aggregate capacities. The maximum link capacity is about 100 Mb/s in

[4]Experiments with PLC and Channel 2 yielded similar results.

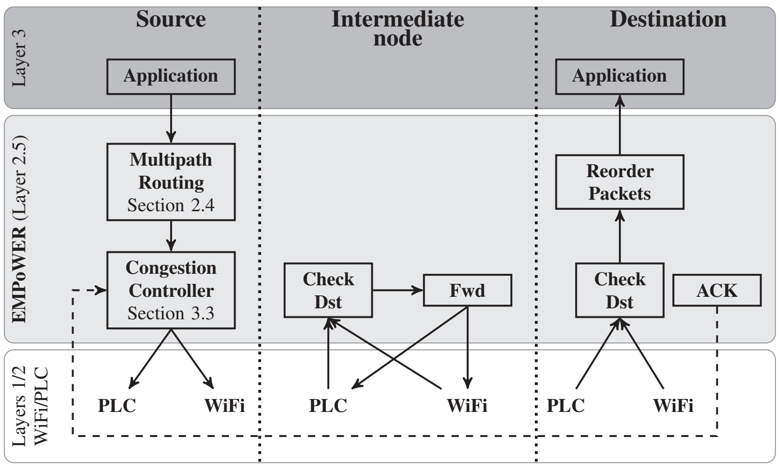


Figure 3.9 – The main components of EMPoWER at layer 2.5. The source determines the routes (Multipath Routing) and the rates at which data is sent on each route (Congestion Control). Congestion control is separated from route selection for the sake of practical implementation. Using the header of our EMPoWER protocol, intermediate nodes simply check whether they are the destination (Check Dst) and, if needed, forward packets to the next hop (Fwd). Finally, the destination reorders the packets based on a sequence number included in their EMPoWER header. Acknowledgements (ACK) are sent by the destination every 100 ms.

both cases.

When launched, the program creates a virtual tun/tap interface that, with a local IP address, can be transparently used by the applications. When a packet is received from the application, our routing protocol replaces the ARP discovery protocol and selects one or two paths. If several paths exist, each packet is sent over path P with a probability proportional to the rate x_P. We employ source routing: Path P is set by the source in a layer 2.5 header that is used by the intermediate nodes to transmit the packets to the next hop. We use short hashes of the MAC addresses of the network interfaces as identifiers at layer 2.5. In our current implementation, the header has a fixed size of 20 bytes, among which 12 are reserved for the path (2 bytes are used to identify each ingress interface along the path, and the total length is limited to 6 hops). 4 bytes are used to store the variable q_P along path P, as described in Section 3.3.1. These values are sent back to the source via dedicated acknowledgments, which are sent (at most) 10 times per second, using the best single-path. These acknowledgments use prioritized queues to minimize delays.

The header contains also a 4-byte sequence number, which is used by the destination for reordering packets that arrive from different paths. We do not use timeouts for missing packets. To identify a lost packet (because of channel errors or congestion), the destination stores the last sequence number received from each path: A packet with a sequence number S is lost when it has received packets with sequence number greater than S on all paths from a certain source.

The capacities of the links are estimated for UDP traffic, using modulation information inherent in the frame header: the modulation and coding scheme (MCS) index for 802.11n, and the bit loading estimate (BLE) for PLC. These two metrics are extremely accurate when traffic is sent at a high rate [VHT15]. When no flow is active, link capacities can be estimated precisely (although not perfectly) by sending probes at a low rate (about 1 kB/s) [VHT15], which yields very low overhead. It reacts to capacity changes in a few seconds. This estimation is sufficient for our routing protocol, as it does not require high precision. The congestion controller requires higher precision, because an overestimated link capacity yields congestion. When a flow is active, the traffic sent on this flow is used for estimating the link capacities. Because traffic is sent at high rate, this estimation is extremely precise, and it is able to detect capacity changes and link failures very rapidly (to the order of hundred of milliseconds). To this end, it is possible to use the acknowledgements described above, sent every 100 ms.

For the value of the congestion controller step size α, we use a simple heuristic based on the observation that, for short paths and single-paths, the congestion controller can support a higher α to converge. α is initially set to 0.02. We multiply α by 2 when there is a single-path or when the longest path is two-hop; and by 4 when the longest path is one-hop. Finally, in order to react to a too large α, we divide α by 2 whenever we find 6 or more non-decreasing oscillations. This heuristic works well in our experiments.

3.5.2 An Example

We first give an example of how EMPoWER works in practice. In the following, a flow from Node A to Node B will be denoted by Flow A-B. We propose an experiment with two flows, Flow 1-13 and Flow 4-7. The scenario is described and the capacities of the links involved are depicted in Figure 3.10. Our routing algorithm selects, for Flow 1-13, the two paths shown on the figure: P_1, a two-hop WiFi-PLC path, and P_2, a single-hop PLC path. Flow 4-7 has a single-hop WiFi path, P_3. During the first 1950 seconds, we send UDP saturated traffic with `iperf` on Flow 1-13. In Figure 3.11, we show the traffic rate injected over each of the two paths for Flow 1-13, along with their sum, that is, the total rate sent by Node 1, and with the throughput received at Node 13. We also show the average throughput achievable over the best possible single path (horizontal line), which is P_1. Our congestion controller uses 100% of the capacity of P_1. As this consumes only about 50% of the capacity of P_2, it also injects a rate roughly equal to 50% of the available capacity on this path. Using two paths simultaneously provides an important gain in throughput (about 45% in this example) compared to a single path.

After 1950 seconds, we send UDP saturated traffic on Flow 4-7. Our congestion controller adapts to the situation by offloading all the traffic of Flow 1-13 onto P_2 and by avoiding altogether WiFi for Flow 1-13 (leaving 100% of the WiFi capacity available to Flow 4-7). After 3950 seconds, we stop the traffic on Flow 4-7. The situation reverts back to what it was during the first 1950 seconds, with throughput gains due to multipath. Overall, the injected traffic rates match the admissible capacities of the paths: Observe the difference between the injected rate and the received throughput; when the injected rate is too large, the variance of the throughput increases

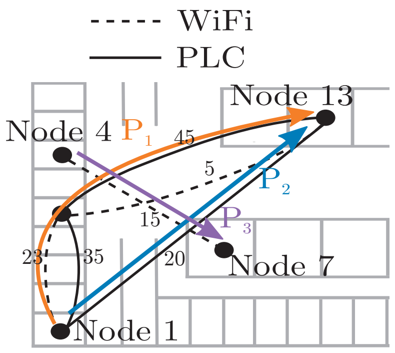


Figure 3.10 – Scenario with two flows on a subset of the testbed described in Section A in the Appendix; the numbers on the figure indicate the measured link capacities in Mb/s at the time of the experiment.

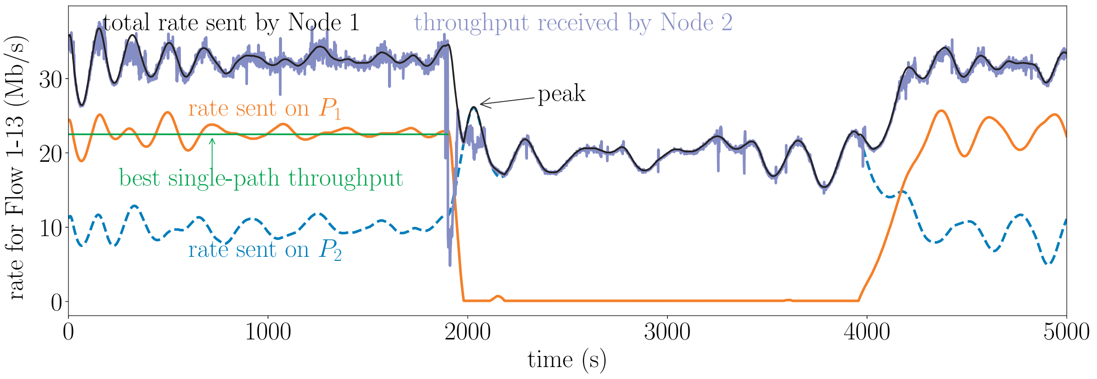


Figure 3.11 – Time evolution of the injected rates and measured throughput on Flow 1-13 for the scenario of Figure 3.10. Flow 4-7 starts at time 1 950 s and stops at time 3 950 s.

(e.g., see the indicated “peak” on Figure 3.11). Although quite simple, this example shows that EMPoWER finds efficient paths and dynamically load-balances traffic in a way that explicitly accounts for interference.

3.5.3 Extensive Performance Evaluation

We now show the performance gains that EMPoWER yields for an isolated flow, by broadly evaluating it on 50 randomly selected pairs of stations. As opposed to the simulations of Section 3.4, PHY layer conditions are not ideal: We use a small constraint margin $\delta = 0.05$ in Equation (3.2). For each experiment, we send UDP saturated traffic with `iperf` during 1000 seconds.

We compare EMPoWER to four different configurations: SP-WiFi, SP, MP-mWiFi, and MP-2bp. For baselines, we show the results on the two single-paths (PLC/WiFi and WiFi-only) obtained

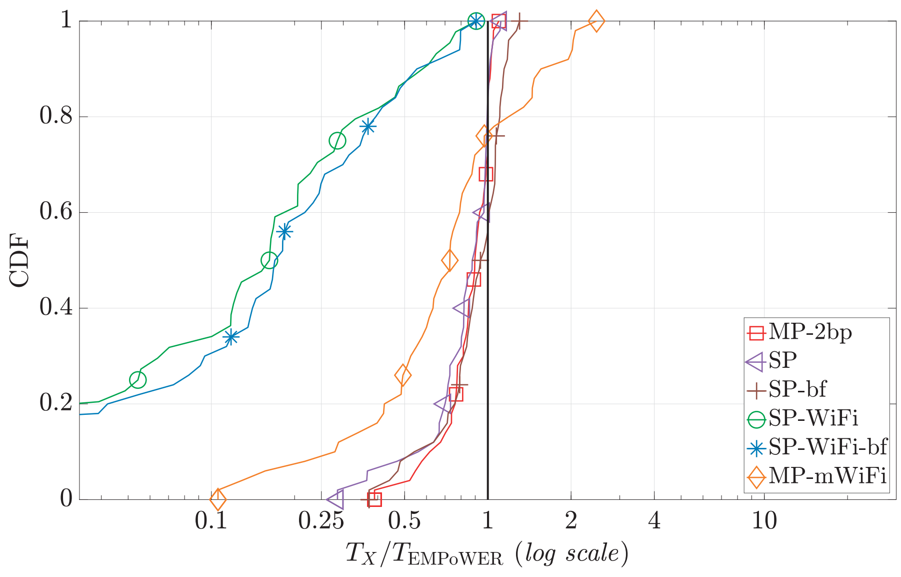


Figure 3.12 – Distribution of T_X / T_{EMPoWER}. x-axis is in log scale.

with a brute-force approach, i.e., by sending rates from 0 to the maximum possible rate with 1 Mb/s increments, and keeping the maximum rate received (denoted by SP-bf and SP-WiFi-bf). We write T_X for the final throughput of the flow for each scheme X (averaged over 10 seconds). In Figure 3.12, we show the empirical cumulative distribution of the throughput ratio T_X / T_{EMPoWER} over the 50 runs.

We make the following observations:

- Hybrid PLC/WiFi logically yields very high throughput gains compared to single-channel WiFi: SP does much better than SP-WiFi-bf in all cases. This confirms our findings of Section 3.4, where we see that PLC/WiFi extends coverage by reducing the number of flows with poor connectivity.

- More interestingly, hybrid PLC/WiFi also yields gains much higher than multi-channel WiFi, despite comparable aggregate capacities. In fact, EMPoWER gives better results than MP-mWiFi in 75% of the cases and yields throughput improvements up to 10x. In a few cases (about 25%), MP-mWiFi does better, but the maximum throughput improvement over EMPoWER is only 2.5x. The fundamental reason of the improvement enabled with hybrid PLC/WiFi networks is the spatial diversity increase that is offered when using two different physical mediums. For example, in our testbed (shown in Figure A.1 on page 135), there are wide walls (e.g., between Node 1 and Node 6 and between Node 5 and Node 8) that strongly attenuate the WiFi signal. This attenuation is strong for all WiFi channels, even when they are in two different bands (2.4 GHz and 5 GHz). In contrast, the walls have no impact on the PLC signal that uses the electrical wires. Inversely, there is no PLC connectivity between Node 12 and Node 14 they are connected to two different electrical panels, but the WiFi signal is strong between these two nodes. This absence of correlation between the signals of the different technologies increases

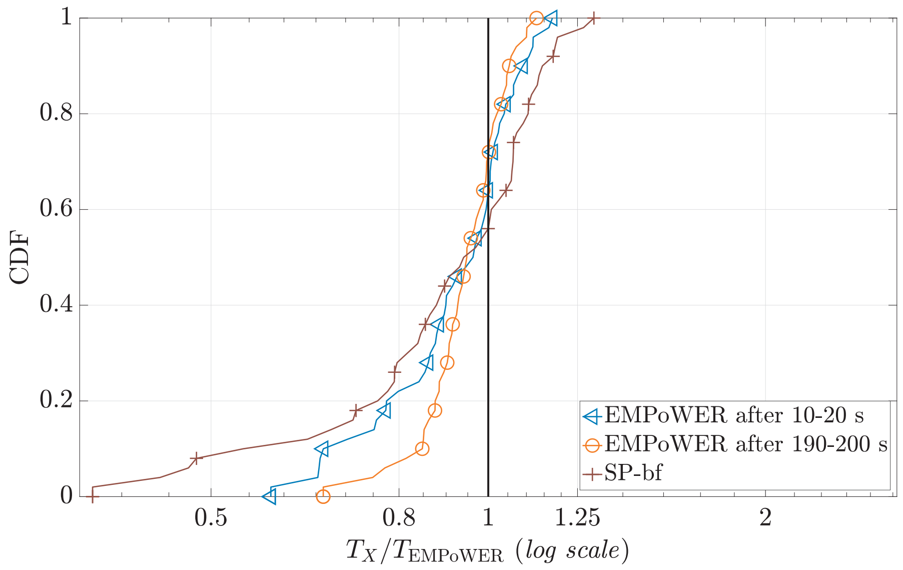


Figure 3.13 – Distribution of T_X / T_{EMPoWER}. x-axis is in log scale.

the spatial diversity and the throughput.

- Our multipath-routing algorithm is beneficial: EMPoWER does almost always better than MP-2bp; and in 60% of the cases, it does better than SP-bf, obtained with a brute force approach. In 70% of the cases where EMPoWER does better than SP-bf, it uses two paths; in the remaining 30% cases, our multipath-routing protocol returns only one path, which is better than the one returned by the single-path procedure. In some cases, SP-bf does better, but the difference is less than 30%, whereas EMPoWER can yield an improvement up to 2.7x over SP-bf. EMPoWER does almost always better than SP. SP employs the same path as SP-bf, which means that the difference between EMPoWER and SP-bf does not come from using multiple paths, but only from the constraint margin δ or from slight imprecisions in the link capacity estimations.

We now study the convergence time of EMPoWER. In Figure 3.13, we plot the average throughput achieved between 10 and 20 s and between 190 and 200 s, as a proportion of the final throughput T_{EMPoWER}. For a baseline, we plot SP-bf. EMPoWER rapidly reaches a rate close to the final one: In 80% of the cases, it is within 80% of the final rate after 10 s. This is also what

	EMPoWER	MP-w/o-CC
Tiny, F. 6-13 (100 kB)	0.128 s ± 0.03	0.159 s ± 0.09
Short, F. 6-13 (5 MB)	9.9 s ± 2.1	13.3 s ± 1.9
Long, F. 6-13 (2 GB)	333.2 s ± 27.7	534.5 s ± 12.6
Conc, F. 6-13 (2 GB)	416.8 s ± 30.3	581.0 s ± 61.4
Conc, F. 12-8 (25 MB)	64.9 s ± 6.5	155.2 s ± 24.3

Table 3.1 – Download times for the four experiments.

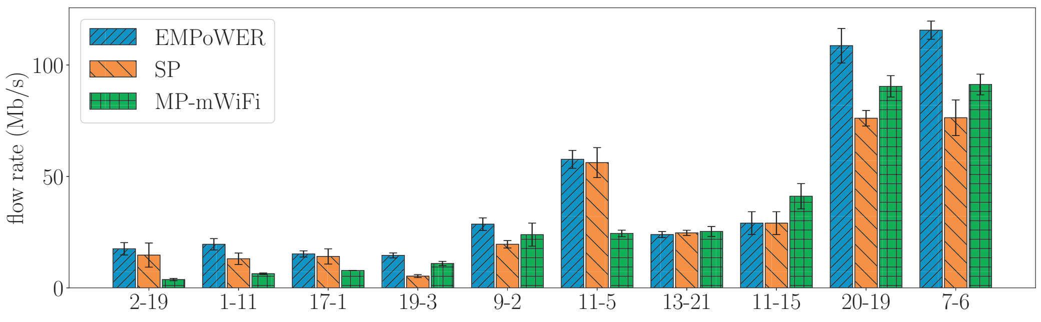


Figure 3.14 – Average rate and standard deviation of throughput measurements during the last 100 seconds for EMPoWER, MP-mWiFi, and SP.

we observe in our example of Section 3.5.2: The oscillations take some time to dampen, but their amplitude is small, and the rate is rapidly close to the final value. After only 10 s, EMPoWER already outperforms the baseline SP-bf. Still, this convergence time is quite long for bursty flows that last only a few seconds, and these flows do not get optimal rates: EMPoWER is designed mostly for long-run best-effort flows. Nevertheless, EMPoWER also improves performance for bursty flows, compared with MP-w/o-CC. To see this, we conduct four experiments: *Tiny*, *Short* and *Long*, where Flow 6-13 is a download of a file of respectively 100 kB, 5 MB and 2 GB, without concurrent traffic; and *Conc*, where Flow 6-13 is a download of a 2 GB file, and Flow 12-8 is a concurrent download of five 5 MB files, with Poisson-distributed starting times (mean 60 s) for the files. Both flows use two two-hop paths: Flow 6-13, PLC-WiFi and PLC-PLC, both through Node 7; and Flow 12-8, WiFi-PLC through Node 7 and WiFi-WiFi through Node 10. Tiny and Short are repeated 40 times, Long and Conc are repeated 10 times; mean value and standard deviation of the download times are shown in Table 3.1. Clearly, EMPoWER is also beneficial for short flows (Tiny and Short); but without concurrent traffic, improvement is more moderate than for long flows (24-34% vs. 60% improvement).

Finally, in Figure 3.14, we show the average throughput after convergence for 10 randomly selected flows, along with a bar showing the standard deviation of the throughput measurements during the last 100 seconds (one measurement per second). This enables us to evaluate potential throughput variations due to packet reordering at the destination when using multiple paths. In general, multipath does not cause variations larger than single-path (see for example the results for Flows 20-19 and 7-6). The figure also further confirms our findings of Section 3.4: Compared to multi-channel WiFi, EMPoWER extends coverage by boosting performance especially for flows with poor connectivity (e.g., Flows 4-19 and 1-11). It also boosts performance for other flows in general, but not always (e.g., not for Flow 11-15), and more moderately so.

3.5.4 TCP Friendliness

EMPoWER interacts with TCP on two levels: First, TCP reacts to congestion (in an interference-agnostic way) and attempts to ensure fairness. This is already achieved by our congestion

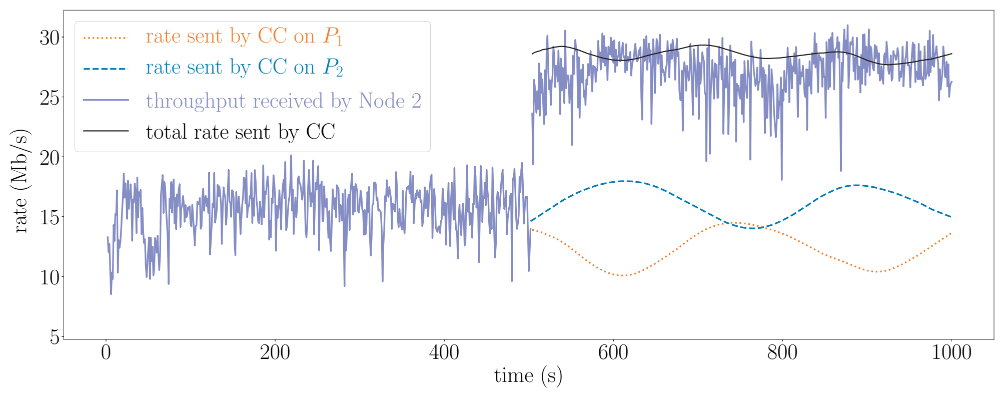


Figure 3.15 – Experiment with TCP for Flow 9-13: SP-w/o-CC from 0 s to 500 s; EMPoWER from 500 s to 1000 s.

controller that drops packets if the rate sent by the above layers goes above the total rate for the flow. TCP will naturally adapt to the rate of our congestion control, because it will perceive the dropped packets as congestion. Because the link capacities are estimated for UDP, and to avoid being in a congested state with possible packet drops and high delays, it is possible to set a non-zero value for the constraint margin δ in Equation (3.2); we discuss the impact of δ in this section. Second, TCP expects packets to be (i) in order and (ii) within some time-frame. Multipath however does not satisfy these conditions, because different delays can occur on the different paths. Our reordering algorithm addresses issue (i), but not issue (ii), and TCP timeouts might still occur, which would degrade TCP throughput. This takes place typically because one path has delays much smaller than the other one, and packets sent on the fast path timeout while waiting for packets sent on the slow path. To improve performance, we add some delay on the fast path at the destination, so that both paths have approximately the same delays, by using the following algorithm. When it sends it, the source timestamps each packet p with its current time $t_s(p)$. For each path P_i, the destination keeps an estimate of the skewed one-way delays $d(P_i)$, with a weighted moving average algorithm: for a packet p coming from path P_i, $d(P_i)$ is updated by $d(P_i) = (1-\beta)d(P_i) + \beta(t_d(p) - t_s(p))$, where $t_d(p)$ is the arrival time of p and β some constant. Note that $d(P_i)$ is *not* the real one-way delay, because it includes the skew between the clocks at the source and at the destination. However, this skew is the same for P_1 and P_2, it is consequently possible to compute the real average delay difference $\Delta = d(P_2) - d(P_1)$. Now, the destination can delay packets coming from the path with smallest delays (without loss of generality, we assume it is P_1): if a packet p comes from P_1, then the destination looks at the skewed delay $t_d(p) - t_s(p)$. If it is greater than $d(P_1) + \Delta$, then the packet goes through directly; else it is delayed during a time $d(P_1) + \Delta - (t_d(p) - t_s(p))$. The packets are then reordered. This procedure does not ensure that TCP timeouts do not occur; however, it reduces the number of such events, thus improving the throughput.

In Figure 3.15, we show how EMPoWER works with TCP for Flow 9-13. P_1 is a two-hop WiFi-WiFi path through Node 12. P_2 is a three-hop PLC-PLC-WiFi path through Nodes 6

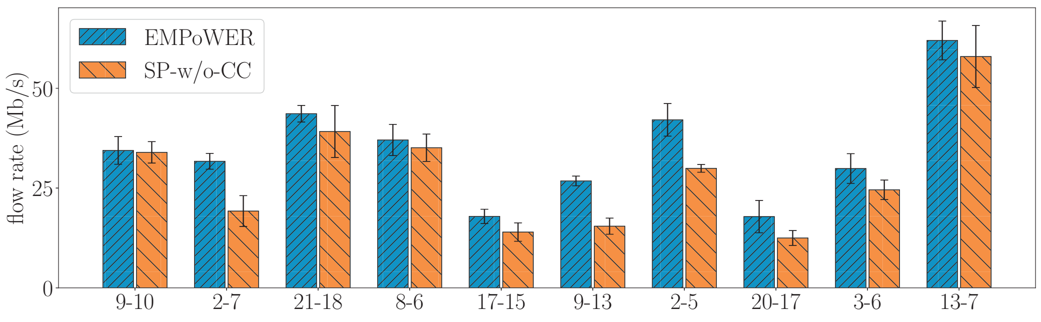


Figure 3.16 – Average TCP rate, with standard deviation.

and 12. Traffic is sent during the first 500 s on P_2 with TCP alone (SP-w/o-CC); during the next 500 s, our congestion controller is active and both P_1 and P_2 are used (EMPoWER). TCP acknowledgements are always sent on the best reversed path. This example is a "critical" case, as it uses paths of different lengths causing different delays, and both mediums have contending links. Despite this, the received throughput matches the traffic sent by our congestion controller, which means that TCP interacts well with EMPoWER. These results are obtained with $\delta = 0.3$. Results depend on the value of δ; when δ gets smaller, the performance of EMPoWER rapidly degrades, because more packets are lost due to contention. We run EMPoWER on 10 randomly selected flows that use two paths in the multipath case, and compared with single-path TCP, the value $\delta = 0.3$ is found to improve performance in all the cases, with no variance increase in general (see Figure 3.16).

In a practical scenario with multiple traffic types, only the nodes in the contention domain of a TCP flow should use this value of δ. If a node receives TCP messages, it informs its neighbors by piggybacking this information in the broadcast messages described in Section 3.3.1: TCP and UDP flows can be handled efficiently at the same time.

3.6 Related Work

Routing and Scheduling

From a theoretical point of view, several works propose joint routing and scheduling, building on the backpressure idea first proposed by Tassiulas and Ephremides [TE92]. In particular, some works propose utility maximizing schemes for wireless multi-hop networks [e.g., LS04, CLCD06, ES06]. However, backpressure scheduling is NP-hard [GNT06] and difficult to implement in practice. Furthermore, it would require changing the scheduling algorithm of the MAC layer, which we avoid in order to maintain compatibility with existing devices. These constraints are the basis for works such as *Horizon* [RGGK08], an algorithm for balancing the load over several wireless multi-hop paths. *Horizon* is based on backpressure, but simplifies the problem to implement the algorithm on existing 802.11 networks, in particular by separating congestion control from routing, an approach that we also choose. *Horizon* focuses on single-channel wireless networks and uses queue sizes to identify congestion, which might prove difficult to extend in practice to different technologies: Because of frame-aggregation amendments in both WiFi and PLC, the number of Ethernet packets in a PHY layer frame, as well as the MAC overheads, can vary significantly between technologies and for different links. Using link capacities, we explicitly account for the different interference patterns occurring in hybrid networks. Neely et al. [NML08] introduce a combination of flow control, routing and scheduling for hybrid networks; it also relies on a backpressure component. These schemes require a central controller and have not been implemented in real networks. Furthermore, converging to efficient (i.e., nearly utility-optimal) steady states requires large queues and long convergence time.

There is also a large body of work that studies congestion control on pre-defined paths in multi-hop networks. The congestion controller presented in this chapter borrows concepts from works that study wired networks [KMT98, LL99, WPL03]. In order to be implemented with shared-medium technologies, it extends these concepts to consider interference. Other congestion control algorithms consider interference in multi-channel wireless networks [e.g., MRW06, GSK08], but they do not consider multiple paths involving different nodes; and they consider joint congestion control and channel assignment, whereas channel assignment is irrelevant in our hybrid PLC/WiFi networks. Moreover, these schemes are not implemented on a real testbed. The works that consider joint multipath-routing and congestion-control [LR07, TT07, ZWT$^+$10, ABL05] are challenging to implement in practice (in particular, they require centralized decisions and time sychronization) and studied only through simulations. In contrast, EMPoWER is fully distributed, which makes it amenable to implementation on a real testbed.

Layer 4 vs. Layer 2.5 Approaches

The most popular multipath-routing approach is MPTCP [FRHB13]. However, this solution targets end-to-end paths, not home networks, because it operates at layer 4. It requires end-hosts to be multihomed (i.e., have several network interfaces directly exposing different IP sub-stacks). This may be a limitation in practice: MPTCP, or any other layer-4 approach, limits the possibility of using several paths in a home network without multihoming. In contrast, solutions working at layer 2.5 are transparent to other protocols and do not require any modification of the underlying MAC layers. Moreover, layer 2.5 solutions can react faster to channel or topology changes, compared with layer 4, again in a transparent-to-higher-layers fashion. IEEE 1905.1 standardizes hybrid networks at layer 2.5. For these reasons, EMPoWER operates at layer 2.5. It is confined to home networks and transparent to other Internet hosts.

3.7 Summary

In this chapter, we have presented a novel multipath congestion-controller that converges to utility-optimal allocations in a distributed fashion. To fully exploit the gains enabled by the multiple technologies, it relies on an interference model that describes how links that employ the same technology share the available capacity. Along with the multipath routing-protocol presented in the previous chapter, it forms a complete system for optimizing throughput in hybrid networks, called EMPoWER. We have evaluated EMPoWER by simulations and on a testbed implementation, over WiFi and PLC stations. To the best of our knowledge, this is the first implementation of congestion-control and multipath-routing algorithms in hybrid PLC/WiFi networks. EMPoWER is practical and distributed, and it offers performance close to that of optimal-but-impractical algorithms. We have also substantiated the gains of introducing PLC in local networks. In particular, we have found that due to medium diversity, hybrid PLC/WiFi improves throughput (up to 10x) and coverage compared to multi-channel WiFi.

4 Choosing the Best Multipath with Multi-armed Bandits

4.1 Introduction

In the previous chapter, we have presented an algorithm that optimally controls the congestion on one multipath fixed in advance. This algorithm is able to react to dynamic conditions by adapting the rate it sends on each path of this multipath, but it cannot react by switching the multipath it uses. In this chapter,[1] we tackle this issue and present an algorithm that finds the best multipath in a dynamic hybrid mesh network.

Finding the best multipath and controlling the congestion along the different paths, by sending traffic at the appropriate rate, is far from trivial because of the following challenges:

Challenge 1: Multiple possible multipaths with *unknown optimal rates*;

Challenge 2: *Dynamic network conditions*, with unknown environments in terms of topology, capacity and interference;

Challenge 3: *Shared-medium technologies* and different interference graphs when employing multiple media.

To confront all the aforementioned challenges, we employ a multi-armed-bandit (MAB) strategy. In the MAB problem, a *player* can choose one of N actions (in the original problem, the player chooses which one of N slot machines to play). At each round, each action, also called an *arm*, has a reward associated with it (the amount of money the player receives each time a slot machine is played). The player's goal is to maximize this reward that is a priori *unknown*. The strategies employed address the tradeoff between *exploration* (play each machine to improve the estimates of the reward distributions) and *exploitation* (maximize the long-term reward given the current estimates). The MAB problem has been widely studied, and strategies ensuring the convergence to the optimal arm have been proposed, whether the rewards are stationary [ACBF02, BCB12] or not [ACBFS95, BGZ14]. When applied to multipaths, MAB strategies accommodate Challenge 1. However, to the best of our knowledge, no existing strategy has actually been implemented in

[1]This chapter is based on the paper by Henri et al. [HVT18].

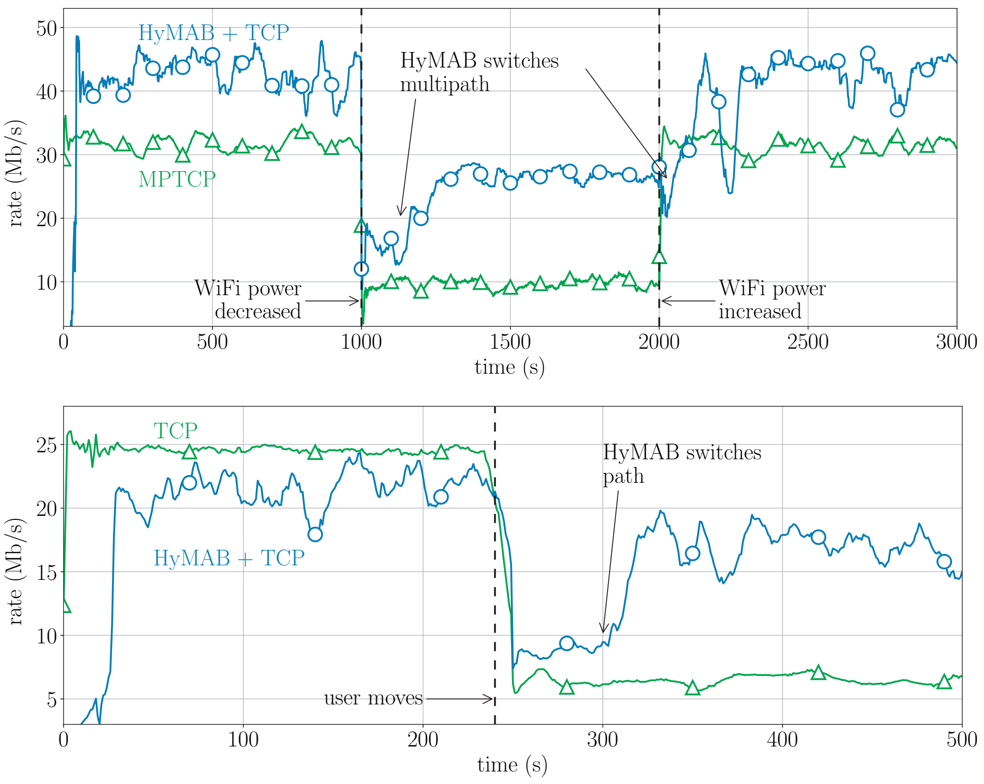


Figure 4.1 – HyMAB in action under dynamic conditions due to capacity drops (top, with a multipath of two paths) or mobility (bottom, with a single path). Results from two different flows in our hybrid WiFi/PLC testbed.

the context of routing. Achieving maximal throughput in a mesh network requires solving two problems: (i) finding the optimal rate on a multipath, i.e., the rate that yields maximal throughput at the destination, and (ii) finding the best multipath, i.e., the multipath for which the rate is maximal among all multipaths. In hybrid shared-medium networks, computing the best multipath is extremely challengin, as discussed in Section 2.3.1. Most protocols used in current networks are based on heuristics and are not guaranteed to be optimal.

In this chapter, we address the above-mentioned problem (i) and present a measurement-based method for computing the optimal rate on a multipath. This method accommodates shared-medium technologies and diverse interference graphs, thus addressing Challenge 3 above. It gives precise results when the rate is averaged over several measurements. On the one hand, to address problem (ii) above, we need to *explore* several multipaths and estimate their optimal rate with sufficient precision. This requires sending traffic several times on each of these multipaths, including the sub-optimal ones, which means that traffic is sent at a sub-optimal rate. On the other hand, *exploiting* only the estimated best multipath in order to send traffic at optimal rate carries the risk of imprecise rate estimations due to insufficient explorations, which can lead to mistakes in identifying the actual best multipath. MAB is the ideal framework for finding the best tradeoff between these two conflicting goals, and we use it to develop HyMAB, a new algorithm to find the optimal rate in a mesh hybrid network. As a preliminary example, Figure 4.1 presents testbed experiments illustrating two typical use cases for HyMAB in dynamic conditions: capacity degradation (top, with a multipath of two paths) and user moving from a room to another (bottom, with a single path). In the top figure, the reported experiments show that when employed with the Transmission Control Protocol (TCP), HyMAB performs better than multipath TCP (MPTCP) [FRHB13], the most popular multipath solution today. In the bottom figure, HyMAB outperforms TCP protocols that have been configured on the best multipath at the beginning of the experiment. Indeed, HyMAB can adapt to dynamic conditions by switching to the multipath that is the best in the new conditions.

Outline of the Chapter This chapter is structured as follows. In Section 4.2, we present a measurement-based method for computing the optimal rate on a multipath. We discuss the challenges that need to be addressed in designing a MAB algorithm for dynamic networks. In Section 4.4, we introduce HyMAB and prove that it is optimal under static conditions. In Section 4.5, we show how HyMAB can also naturally adapt to dynamic conditions, thus confronting Challenge 2, which poses the tradeoff between optimality and adaptability. To validate the practicability of our solution, we implement it on a real testbed. We present an extensive performance evaluation over a network of 22 PLC/WiFi nodes in Section 4.6, showing the practical usability and performance gains of HyMAB. We discuss related work in Section 4.7 and close the chapter with a summary in Section 4.8.

4.2 Computing the Optimal Rate

We present how the optimal rate on a given multipath can be computed, based on the model described in Section 2.2.1. The results of this section are valid when a single flow is present; the extension to multi-flow is described in Section 4.4.3. HyMAB maximizes the rate of the flow. For this reason, the flow is assumed to be saturated, i.e., the source always has packets to send (we discuss the case of non-saturated flows in Section 4.4.4).

Remember that we denote by $\boldsymbol{\alpha}_{\mathscr{P},l}$ the multipath-impact vector of $\mathscr{P}$ on l, by $\boldsymbol{A}_{\mathscr{P}} \in \mathbb{R}^{L_{\mathscr{P}} \times M}$ the matrix $[\boldsymbol{\alpha}_{\mathscr{P},l}^{T}]_{l\in\Lambda_{\mathscr{P}}}$ and by $\boldsymbol{\mu}_{\boldsymbol{x}_{\mathscr{P}}} \in \mathbb{R}^{L_{\mathscr{P}}}$ the vector of the busy-times $[\mu_{l,\boldsymbol{x}_{\mathscr{P}}}]_{l\in\Lambda_{\mathscr{P}}}$. The multipath-impact vector depends only on the network topology and on the paths, and *not* on the rate vector $\boldsymbol{x}_{\mathscr{P}}$. If the multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l}$ are known for all links $l \in \Lambda_{\mathscr{P}}$, it is easy to find an optimal rate $\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}$: As we have seen in Section 2.2.1, it is a solution of System (2.4).

But directly computing the multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l}$ would require knowing the link capacity c_l and the interference domain $\mathscr{I}_l$ of all links $l \in \Lambda_{\mathscr{P}}$, which is challenging or impractical [PD11], especially in hybrid networks with diverse interference graphs and dynamic conditions. Instead, to account for interference per collision domain, the nodes can measure the busy time $\mu_{l,\boldsymbol{x}_{\mathscr{P}}}$ when the source sends traffic on $\mathscr{P}$ at rate $\boldsymbol{x}_{\mathscr{P}}$: For WiFi, this is achieved by using information exposed by WiFi drivers; for PLC, by using specific fields in the IEEE 1901 frame headers (more details are provided in Section A.3 of the Appendix). In Section 4.6.1, we describe how the source of $\mathscr{P}$ gathers the busy-time measurements $\mu_{l,\boldsymbol{x}_{\mathscr{P}}}$ for all $l \in \Lambda_{\mathscr{P}}$. Once the source knows the busy-time measurements, it is easy to get $\boldsymbol{\alpha}_{\mathscr{P},l}$. As an example, consider first the case $M = 1$: There is one path P on which the source sends traffic at rate x_P. If x_P is such that the links $l \in P$ are not saturated, $\alpha_{P,l}$ can be computed from Equation (2.2) by $\alpha_{P,l} = \frac{\mu_{l,x_P}}{x_P}$. This result can easily be extended when there are $M \geq 2$ paths in the multipath $\mathscr{P}$. We choose M linearly independent rate vectors $\boldsymbol{x}_{\mathscr{P}}^{(i)}$ for $i \in \{1,\ldots,M\}$, and for each rate vector $\boldsymbol{x}_{\mathscr{P}}^{(i)}$ at which traffic is sent on multipath $\mathscr{P}$ and each link $l \in \Lambda_{\mathscr{P}}$, the corresponding busy time $\mu_l^{(i)}$ is measured. If $\boldsymbol{X}_{\mathscr{P}}$ denotes the $M \times M$ matrix of the rate vectors $\boldsymbol{X}_{\mathscr{P}} = [\boldsymbol{x}_{\mathscr{P}}^{(i)T}]_{i\in\{1,\ldots,M\}}$ and $\boldsymbol{\mu}_l$ denotes the vector of the busy-time measurements $\boldsymbol{\mu}_l = [\mu_l^{(i)}]_{i\in\{1,\ldots,M\}}$, we have $\boldsymbol{X}_{\mathscr{P}} \cdot \boldsymbol{\alpha}_{\mathscr{P},l} = \boldsymbol{\mu}_l$. By construction, the $\boldsymbol{x}_{\mathscr{P}}^{(i)}$'s are linearly independent, hence $\boldsymbol{X}_{\mathscr{P}}$ is a $M \times M$ full-rank matrix, and we get $\boldsymbol{\alpha}_{\mathscr{P},l} = \boldsymbol{X}_{\mathscr{P}}^{-1} \cdot \boldsymbol{\mu}_l$ for all links $l \in \Lambda_{\mathscr{P}}$, i.e., we get the matrix $\boldsymbol{A}_{\mathscr{P}}$. We then solve the linear system given by Equation (2.4) using standard techniques, which gives the optimal rate $\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}$ achievable on $\mathscr{P}$ without having to measure the link capacities or the interference domains.

In Section 2.2.2, we have validated the precision of the busy-time measurements and we have shown experimental results that confirm that the method described here can be used to compute the optimal rate of a path. But we have also shown that it requires sending packets by batches of B packets ($B = 100$ for WiFi, $B = 200$ for PLC). Sending packets by batches has two consequences. First, it might increase the jitter for the application. However, this is true only when traffic is sent at a rate much lower than the link capacity: When sending at half of the capacity on an average link, batches do not significantly increase jitter, as evaluated by using `iperf`; even when sending at 10% of the capacity, the jitter increases from 4 ms to only 6 ms. Moreover, sending by batches is needed only during probing phases, and not during exploitation phases (see next section). Second, as we have shown in Section 2.2.2, sending packets by batches increases the variability of the busy-time measurements. Therefore, to compute the rate with sufficient precision, the measurements need to be repeated several times and averaged out. This means that one measurement is not enough, and this challenge justifies the use of a MAB strategy.

4.3 Towards a Practical MAB Strategy for Dynamic Networks

As explained in Section 2.3.1, finding the best multipath (denoted by $\mathscr{P}^*$ in the following) is a difficult problem. For this reason, all existing routing protocols rely on heuristics, and they do not guarantee the optimality of their result. It would be possible, using the approach of Section 4.2, to compute the optimal rate of several multipaths, and to keep the best multipath as the one benefiting from the maximum rate. However, as we have seen in Section 2.2.2, the busy times (hence, the computation of the optimal rate) require several measurements to be precise, and one measurement is not sufficient. Consequently, there is a conflict between exploring several multipaths to estimate the optimal rate of each of them with sufficient precision, and exploiting the multipath found to be the best so far: The former yields a sub-optimal throughput, whereas the latter involves the risk of choosing a sub-optimal multipath if the rate estimations are imprecise due to insufficient explorations.

MAB strategies have the potential to address this exploration-exploitation tradeoff. In fact, routing was identified early on as a potential application for MAB [ACBFS95]. Here, we employ multipath routing, i.e., several paths can be employed simultaneously. Combinatorial MABs [GKJ12, CWY13] typically study the problem of routing. Gai et al. [GKJ12] find the shortest path in a graph with varying link capacities. However, with interfering links, finding the path with highest throughput is NP-hard [JPPQ05], and it cannot be achieved by finding the shortest path in a graph; it is even more complex when considering multipath routing. Chen et al. [CWY13] introduce a model where several arms (the paths) are played simultaneously and grouped in so-called *super-arms* (the multipaths). However, their solution assumes that the rewards of each arm are independent of the super-arm that is played. This is not the case here, because links of different paths might interfere with each other.

Instead, we consider in this work that heuristic-based routing protocols are not perfect (they do *not* necessarily return the best multipath), but that they are “not too bad”, in the sense that the best multipath, although unknown, is among the N multipaths that the protocol finds to be the N best, with N fixed in advance. In our experimental results of Section 4.6, we show that N can be set to a small value, e.g., $N = 5$. Our goal is thus to find the best multipath in a given set $\mathscr{S}$ of N multipaths, which can be solved with more classic MAB approaches: The source of the flow is the *player*, and the multipaths are the *arms*; the *reward* that the player receives when playing an arm is the optimal rate at which the source of the flow can send traffic on the corresponding multipath. However, to the best of our knowledge, existing MAB strategies have never been actually implemented on a real testbed, for the following reasons.

In existing MAB strategies [BCB12, ACBFS95, BGZ14], the player gets a reward each time an arm is played. In these strategies, at each trial, the only choice that the player makes is the arm to play, i.e., either to *explore* an arm in order to learn its associated reward, or to *exploit* the arm that the player estimates to be the best. In contrast, in our problem, the reward of an arm (i.e., the optimal rate that can be sent on the multipath) is obtained by carrying out a *probing* phase that consists in sending traffic by batches at M different rate vectors $\boldsymbol{x}^{(i)}$ for $i \in \{1,\dots,M\}$ such that no link is saturated, and in measuring busy times over the links, as described in Section 4.2. In practice, a probing phase is costly, in the sense that it prevents from sending at the optimal rate: To ensure that no link is saturated and because the rate vectors $\boldsymbol{x}^{(i)}$ must form a linearly independent family, the $\boldsymbol{x}^{(i)}$ are all smaller than the optimal rate. Therefore, the source must not only choose the arm to play, i.e., the multipath to use, but it must also decide to either probe the arm (send traffic at a sub-optimal rate, which enables to measure the busy times and to compute the optimal rate), or *exploit* the estimated best arm (send traffic at the optimal rate).

The ϵ-greedy strategy [SB98]introduces a clear distinction between exploring an arm or exploiting the best arm: The source chooses to explore with a fixed probability ϵ, and chooses the arm it explores uniformly at random among all arms. In our case, we can similarly choose to explore with probability ϵ, and probe one arm randomly chosen. However, the value of ϵ is difficult to determine in practice: A large ϵ makes the algorithm converge far from the optimum, whereas a small ϵ makes it too long to converge, because of the noisy measurements. In the ϵ_n-greedy strategy [ACBF02], this issue is solved by introducing an exploration probability $\epsilon(t)$ that is a decreasing function of the time t, equal to

$$\epsilon(t) = \min\left(1, \frac{cN}{d^2 t}\right), \tag{4.1}$$

where d is a lower bound on the difference between the expected reward of the arms, and c a positive real number. If exploration is chosen, the arm to explore is chosen uniformly at random. But d needs to be known a priori, which is not the case in practice. Furthermore, even if we could know d, the performance of the algorithm rapidly deteriorates if c is not appropriately tuned [ACBF02]. Finally, the ϵ_n-greedy strategy is not efficient if two arms yield very similar rewards (small d), because most of the initial time (small t) is spent exploring all arms uniformly at random and because exploration is a costly process. In the context of multipath routing, this is likely to happen, because the multipaths might share a common bottleneck link, and hence have optimal rates that are close to each other (see also the experimental results in Section 4.6). The limitations of ϵ-greedy and ϵ_n-greedy are illustrated through simulations in Section 4.4.2.

Other strategies have been introduced to deal with this inefficiency, in particular Upper Confidence Bound (UCB) [BCB12]. In UCB strategies, the player chooses the arm to play, based on the statistical information it has so far, and favors the estimated best arm while ensuring that the statistical information gathered for all arms is sufficiently precise. In UCB strategies (like in the vast majority of existing MAB strategies), the reward of an arm is known by the player each time this arm is played. These strategies cannot be used for our problem, where obtaining the reward requires probing the arm, which, as explained above, is a costly procedure. For the aforementioned reasons, we introduce a new algorithm, HyMAB, in Section 4.4.1. It uses UCB strategies as a subroutine.

HyMAB is proven optimal under static conditions in Section 4.4. Nevertheless (see Sections 4.5 and 4.6), HyMAB is efficient under dynamic conditions. This adaptability stems from the nature of the MAB framework: Exploration and probing are useful not only to find the best multipath and optimal rates, but also to continuously adapt to dynamic conditions. There is a tradeoff between optimality and adaptability: Under static conditions (Section 4.4), the probing probability needs to go to zero to ensure optimality; under dynamic conditions (Section 4.5), the probing probability needs to stay away from zero to adapt continuously to dynamic environments. This tradeoff is studied in Section 4.5.1. Many works study MAB when the rewards vary dynamically [ACBFS95, BGZ14], but as with UCB strategies, they are valid only when the player knows the reward each time the arm is played, i.e., when no probing is required to get the rewards. D-MAB [DFSS08] is one of the few algorithms that could apply in our problem, but the empirical solution that it offers, evaluated by simulations, assumes that changes in the rewards of the arms happen simultaneously for all arms, which is usually not the case in our scenario.

Finally, the MAB strategies described above are defined for a single player (in our setting, a single flow), whereas in practice, several flows with different sources are present. Some papers have studied the problem of multiple players with dependent [GKJ10] or independent [LZ10, KNJ14] arms, but most of them assume that arms are shared by the players (i.e., they play the same arms). In our setting, arms for different players are not shared because the flows, and thus the multipaths, are necessarily distinct; but they might be correlated (the multipaths of the distinct flows share links or have links interfering with one another). Wilhelmi et al. [WCN+17] study the case where the arms are different and dependent, but their solution requires one arm per possible sending rate, which explodes the size of the search space for arms (the sending rates are continuous). Our extension to several flows (presented in Section 4.4.3) reduces the problem with F different flows to F independent problems with a single flow and N arms.

4.4 An Optimal Algorithm in Static Networks

4.4.1 Optimal Strategy with a Single Flow

Algorithm 1 defines HyMAB, the strategy for finding the best multipath and achieving optimal throughput. It is divided in two stages: in Stage 1, it decides the multipath, and in Stage 2, it decides the sending rate. In Stage 1, HyMAB chooses an arm (i.e., a multipath) according to the UCB1 strategy, introduced by Auer et al. [ACBF02]. We adopt this strategy because it is easy to implement and, when the reward is obtained each time an arm is played, it is shown to be optimal (in the sense that the *regret*, defined as the difference between the rate achieved and the rate that could have been achieved by always playing the best arm, is asymptotically optimal). But as we have seen in Section 4.3, a probing phase is required to obtain the reward of an arm, and therefore UCB strategies alone lead to sub-optimal results. For this reason, in Stage 2, HyMAB chooses between probing the arm chosen at Stage 1 or exploiting the best arm found so far. UCB1 assumes that the rewards are in $[0,1]$; for this reason, the rate vectors $\boldsymbol{x}_{\mathscr{P}}$ are scaled so that for all $\mathscr{P}$ and t, $\|\boldsymbol{x}_{\mathscr{P}}(t)\|_1 \leq 1$. Similarly to the ϵ_n-greedy strategy, the probing probability is a decreasing function of time; to achieve optimality, it must tend to zero and ensure that each arm is explored an infinite number of times almost surely. However, as opposed to the ϵ_n-greedy strategy that requires knowing a bound d on the reward difference, we do not want the exploration probability to depend on a quantity that is a priori unknown, and we set the exploration probability to be $\epsilon_{\mathscr{P}}(t) = 1/n_{\mathscr{P}}^{\lambda}(t-1)$, with a parameter λ that controls the rate of convergence. We study the effects of λ in the next paragraph. As opposed to the ϵ_n-greedy strategy, where the parameter c needs to be finely tuned and has a strong impact on the performance, we show that HyMAB is robust against the choice of λ, as it does not impact drastically the performance.

Theorem 3. *With*

$$\epsilon_{\mathscr{P}}(t) = \frac{1}{n_{\mathscr{P}}^{\lambda}(t-1)}, \tag{4.3}$$

HyMAB converges to achieving optimal throughput for any $\lambda > 0$.

Proof. The proof is given in Section B.2 in the Appendix. □

Algorithm 1 – HyMAB: strategy for optimal throughput.

```
Input: trial duration $D$, set of multipaths $\mathscr{S}$ with $|\mathscr{S}| = N$.
Initialize: for all $\mathscr{P} \in \mathscr{S}$, $n_{\mathscr{P}}(0) = 0$, $T_{\mathscr{P}}(0) = 0$.
for each trial $t$ do
    for all $\mathscr{P} \in \mathscr{S}$ do
        $n_{\mathscr{P}}(t) = n_{\mathscr{P}}(t-1)$, $T_{\mathscr{P}}(t) = T_{\mathscr{P}}(t-1)$
    end for
    Stage 1 Choose one arm (i.e., one multipath) $\mathscr{P}_t \in \mathscr{S}$ according to the UCB1 strategy:
    ◇ if there are non-explored arms, choose one among them,
    ◇ otherwise ($t > N$, $n_{\mathscr{P}} > 0$), choose the arm that maximizes

        $$V_{\mathscr{P}}(t) \doteq \left\|\overline{\boldsymbol{x}}_{\mathscr{P},n_{\mathscr{P}}(t-1)}\right\|_1 + \sqrt{\frac{2\ln(t-1)}{n_{\mathscr{P}}(t-1)}}. \qquad (4.2)$$

    ◇ Set $T_{\mathscr{P}_t}(t) \leftarrow T_{\mathscr{P}_t}(t) + 1$.
    Stage 2 Choose one of the following:
    ◇ with probability $\epsilon_{\mathscr{P}_t}(t)$, probe $\mathscr{P}_t$ by following the procedure described in Section 4.2, i.e.,
        • set $n_{\mathscr{P}_t}(t) \leftarrow n_{\mathscr{P}_t}(t) + 1$, and
        • choose $M$ rate vectors $\boldsymbol{x}^{(i)}$, $i \in \{1,\ldots,M\}$, that are admissible and linearly independent, and
        • send traffic at rates $\boldsymbol{x}^{(i)}$ during $D/M$ seconds, in turn for $i \in \{1,\ldots,M\}$, and
        • compute the reward $\boldsymbol{x}_{\mathscr{P}_t,n_{\mathscr{P}_t}(t)}$ and the current rate estimation $\overline{\boldsymbol{x}}_{\mathscr{P}_t,n_{\mathscr{P}_t}(t)}$.
    ◇ or with probability $1-\epsilon_{\mathscr{P}_t}(t)$, exploit the current best arm $\mathscr{P}_t^*$ maximizing $\left\|\overline{\boldsymbol{x}}_{\mathscr{P},n_{\mathscr{P}}(t-1)}\right\|_1$, i.e.,
      send traffic at rates $\overline{\boldsymbol{x}}_{\mathscr{P}_t^*,n_{\mathscr{P}_t^*}(t-1)}$ on $\mathscr{P}_t^*$ during $D$ seconds.
end for
```

In a real implementation, the trial duration D depends on the number M of paths in a multipath: For the probing phase to be precise, D/M needs to be one order of magnitude higher than the round-trip time on the paths. In practice, M is small: As we have seen in Section 2.3, the optimal number of paths can be limited to the number of technologies K without harming the performance. In the experiments of Section 4.6, we use $K = 2$ technologies (WiFi and PLC), hence $M = 2$.

4.4.2 Evaluation via Simulations in a Static Network

Here we study by simulation the convergence time of HyMAB in a static network. The purpose of the simulation is merely to capture the performance of the different MAB algorithms; for this reason, we only choose the rewards (i.e., the optimal rates) and ignore the underlying network and multipaths. Before running the algorithms, we pick the true optimal rates $\|\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}\|_1$ for the multipaths $\mathscr{P} \in \mathscr{S}$ uniformly at random in $[0.5, 1]$. The rates are assumed to be at least 0.5 because the multipaths returned by the routing algorithm are assumed to be good enough, as mentioned in Section 4.3. When exploring, we assume that the received rate is equal to 1/4 of the current estimation for the multipath, close to what we observe in our testbed experiments

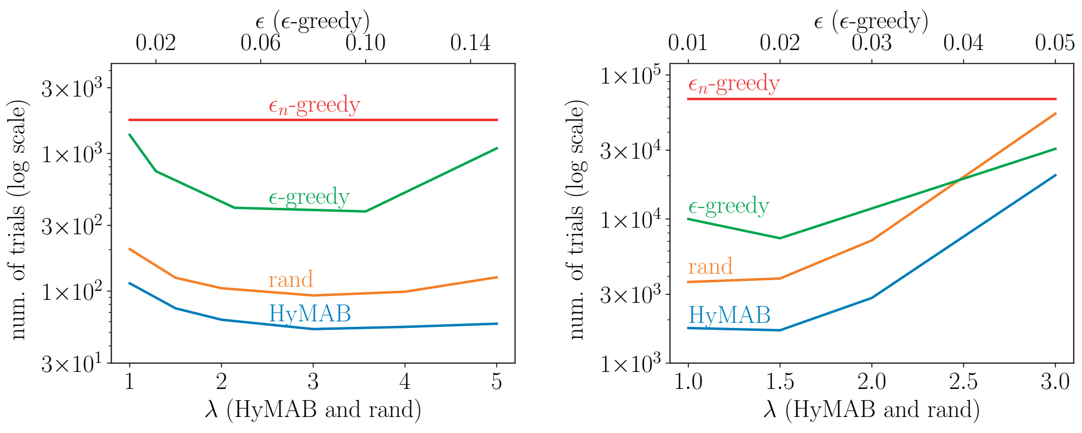


Figure 4.2 – Number of trials needed to reach 85% (left) and 95% (right) of the optimal rate (log scale).

presented in Section 4.6. An estimation $\boldsymbol{x}_{\mathscr{P}_t, n_{\mathscr{P}_t}(t)}$ is computed by adding to the true value $\|\boldsymbol{x}_{\mathscr{P}}^{\text{opt}}\|_1$ a random gaussian noise of mean 0 and standard deviation $0.2\|\boldsymbol{x}_P^{\text{opt}}\|_1$, close to what we observe in our testbed experiments. We study the number of trials needed for the averaged sending rate to reach 85% (Figure 4.2, left) and 95% (Figure 4.2, right) of the optimal rate $\|\boldsymbol{x}_{\mathscr{P}^*}^{\text{opt}}\|_1$, computed by averaging over 10 000 runs with different random seeds. We first evaluate the choices made at Stage 1 and Stage 2, and then compare HyMAB with other MAB algorithms.

Using an optimal strategy such as UCB1 at Stage 1 is not required to converge to the optimal throughput. Convergence can be shown for any strategy such that each arm is chosen at Stage 1 an infinite number of times almost surely, for example by choosing uniformly at random an arm among the others. However, using UCB1 strategy makes the convergence faster. We compare HyMAB (with UCB1 at Stage 1) with a modified HyMAB with uniform random selection at Stage 1 (denoted by *rand*). Figure 4.2 shows that choosing UCB1 strategy in Stage 1 decreases the convergence time of HyMAB by about 50% over rand (note the logarithmic scale for the y-axis). By using UCB1, HyMAB spends less time exploring the arms that are less good, making the overall exploration probability decrease faster than when choosing the arms uniformly at random. In practice, the faster convergence of HyMAB is very important in dynamic networks where the optimal multipath changes due to varying conditions (see also Section 4.5.1).

Figure 4.2 also shows the effects of the parameter λ in the exploration probability $\epsilon_{\mathscr{P}}(t)$ at Stage 2: When λ is increased, less time is wasted exploring, and HyMAB and rand converge faster. However, this is true only up to a certain value: If λ is increased too much, it takes a longer time to correctly estimate the optimal rate because of the noisy measurements of the busy times, and the convergence deteriorates. Nevertheless, Figure 4.2 (left) shows that HyMAB is robust against the choice of λ, as it does not impact drastically the performance: In the first 1 000 trials, the total amount of data sent varies between 0.896 and 0.906 of the best possible amount when λ is varied between 1 and 3, i.e., the maximal difference in the amount of data sent is about 1%. The maximal amount of data sent is found for $\lambda = 2$, a value that we use in the remainder of the chapter.

Finally, we compare HyMAB with two other algorithms, ϵ-greedy [SB98] and ϵ_n-greedy [ACBF02], described in Section 4.3. For ϵ_n-greedy, we assume the reward difference d to be known, and we try several values of c and present the results for the best one. Note that rand is equivalent to a modified version of ϵ_n-greedy where the exploration probability $\epsilon(t)$ in Equation (4.1) is replaced with $\epsilon_{\mathscr{P}}(t)$ defined by Equation (4.3). Figure 4.2 shows that HyMAB outperforms ϵ-greedy by an order of magnitude for any ϵ: With a high ϵ, the cost of exploration is too high, and with a low ϵ, it takes a long time to converge, because of noisy measurements. HyMAB also outperforms ϵ_n-greedy by more than an order of magnitude: When two arms have very close optimal values, ϵ_n-greedy spends most of its time exploring, thus sending at a sub-optimal rate.

4.4.3 Extension to Several Flows

HyMAB is optimal when a single flow is present, but it should also handle several contending flows. The goal is to converge to a fair rate-allocation, in a distributed and scalable way: The only information that the source of a flow needs in our implementation is the feedback from the destination of this flow (and *not* from sources or destinations of other flows).

Let $\mathscr{F}$ be the set of flows, i.e., of source-destination pairs. Each flow $f \in \mathscr{F}$ employs its own set of multipaths, denoted by $\mathscr{S}_f$. For each link l, the number of *interfering flows* F_l is the number of flows that can be overheard by this link, which means that it can be computed by each node for all its outgoing links l with only local measurements. Formally,

$$F_l = \#\{f \in \mathscr{F} \text{ s.t. } \exists P \in \mathscr{S}_f, l' \in \Lambda_P \text{ with } l \in \mathscr{I}_{l'}\}. \tag{4.4}$$

Instead of System (2.4), each source of a flow f solves the system:

$$\begin{aligned} &\max_{\boldsymbol{x}} \ \mathbf{1}^T \cdot \boldsymbol{x} \\ &\text{subject to } \ \boldsymbol{A}_{\mathscr{P}} \cdot \boldsymbol{x} \preceq \mathbf{1}/\boldsymbol{F}_l \text{ and } \boldsymbol{x} \succeq \mathbf{0}, \end{aligned} \tag{4.5}$$

where $\mathbf{1}/\boldsymbol{F}_l$ is the vector whose entries are $1/F_l$ for each link l. The constraint means that on the links where several flows contend, each flow gets an equal time-proportion of the resources.

When several flows are present, computing $\boldsymbol{A}_{\mathscr{P}}$ is more complex, because the busy times are now generated by all flows: $\mu_l = \sum_{f \in \mathscr{F}} \boldsymbol{\alpha}^T_{\mathscr{P}_{t,f},l} \cdot \boldsymbol{x}_{\mathscr{P}_{t,f}}$ where $\mathscr{P}_{t,f}$ is the multipath currently used by flow f. To compute $\boldsymbol{A}_{\mathscr{P}}$, i.e., to compute $\boldsymbol{\alpha}_{\mathscr{P},l}$ for each link $l \in \Lambda_{\mathscr{P}}$, we assume that during an exploration trial for a flow $f_0 \in \mathscr{F}$, the rates of all other flows $f \neq f_0$ are constant (i.e., all other flows are in an exploitation phase). This assumption is discussed in Section 4.5.2. Before an exploration of multipath $\mathscr{P}_0 \in \mathscr{S}_{f_0}$, the source of f_0 does not send traffic during a short time-slot (*silent slot*); each link $l \in \Lambda_{\mathscr{P}_0}$ measures the busy time during this time slot $\mu^{(0)}_{l,f_0} = \sum_{f \in \mathscr{F} \setminus f_0} \boldsymbol{\alpha}^T_{\mathscr{P}^*_{t,f},l} \cdot \boldsymbol{x}_{\mathscr{P}^*_{t,f}}$, where $\mathscr{P}^*_{t,f}$ is the current estimated best multipath for flow f. Then, the source of f_0 performs a regular exploration by sending traffic at M different rate vectors that form a full-rank matrix $\boldsymbol{X}_{\mathscr{P}_0}$, as described in Section 4.2; each link measures the busy-time vector $\boldsymbol{\mu}_l$. If the media are not saturated during the exploration trial, $\mu_l - \mu_{l,f_0}^{(0)} = \boldsymbol{\alpha}_{\mathscr{P}_0,l}^T \cdot \boldsymbol{x}_{\mathscr{P}_0}$ because for all $f \neq f_0, \mathscr{P}_{t,f} = \mathscr{P}_{t,f}^*$, and $\boldsymbol{A}_{\mathscr{P}}$ can be computed by solving $\boldsymbol{\alpha}_{\mathscr{P},l} = \boldsymbol{X}_{\mathscr{P}_0}^{-1} \cdot (\boldsymbol{\mu}_l - \mu_{l,f_0}^{(0)} \mathbf{1})$: When we subtract the busy time $\mu_{l,f_0}^{(0)}$ due to all other flows, which we assume to remain constant along the exploration trial, the linear relationship given by Equation (2.1) ensures that we get the busy time that f_0 would generate in the absence of transmission from other flows. In Section 4.5.2, we discuss how we guarantee that the media stay unsaturated during an exploration trial in the presence of multiple flows. System (4.5) is equivalent to System (2.4); this means that each source can apply HyMAB and converge to the rate allocation that maximizes System (4.5). With this method, each source runs HyMAB independently from the other sources: it computes the rate allocation maximizing its own throughput, while ensuring that the media are shared fairly.

4.4.4 Discussion on Non-Saturated Flows

HyMAB maximizes throughput and is therefore designed for saturated flows. A flow that requires low throughput does not need to use HyMAB. External flows are naturally supported by HyMAB, because their traffic is included in the busy-time measurements. Nevertheless, non-saturated flows are supported by HyMAB. During an exploration phase, HyMAB requires traffic to be sent at a rate that is not too small in order for the optimal rate to be estimated precisely enough; if the flow does not have enough data packets to send, HyMAB sends dummy packets to reach half of the estimated optimal-rate. During an exploitation phase, the source can send traffic at a rate below the optimal rate without any impact on HyMAB. When there are multiple flows, if a HyMAB flow is not saturated, some resources remain unused. It would be possible to apply a progressive filling algorithm to converge to a max-min fair allocation. Because low-throughput flows do not need to use HyMAB, a detailed study for such an algorithm is out of the scope of this dissertation. With external low-throughput flows, HyMAB converges to an optimal and fair utilization of the remaining resources, without affecting the external flows.

4.5 Practical Algorithm Under Dynamic Conditions

In Section 4.4, HyMAB is shown to be optimal under static conditions. In reality, network conditions change: Link capacities are not constant, flows come and go, nodes move. We discuss how to make HyMAB operational under these practical constraints. For short-term variability, HyMAB can be used with TCP (see Figure 4.1) that naturally deals with such variations; in Section 4.6.4, we discuss the interaction of HyMAB with TCP. However, we also want HyMAB to be able to adapt to longer term dynamics that would require to switch multipath, such as major capacity-changes (because of mobility or channel-condition changes) or traffic changes (flows coming or leaving).

4.5.1 Capacity Changes

In Section 4.4, we assumed static conditions; to ensure optimality, the probing probability of a multipath $\mathscr{P}$, $\epsilon_{\mathscr{P}}(t)$, must go to zero. However, these probing phases also enable HyMAB to adapt to dynamic conditions. For practical reasons, in order to continuously estimate the optimal rate and to adapt it to new conditions, we replace $\epsilon_{\mathscr{P}}(t)$ in Theorem 3 by $\epsilon_{\mathscr{P}}(t) = \max(\epsilon_{\min}, 1/n_{\mathscr{P}}^{\lambda}(t-1))$, so that $\epsilon_{\mathscr{P}}(t)$ never goes below a threshold $\epsilon_{\min}$. This differs from the ϵ-strategy described in Section 4.3 because the probing probability is higher at the beginning, which enables a fast convergence even with a small $\epsilon_{\min}$. In addition, because measurements carried long ago are not valid if the conditions have changed, we compute the average rate vector $\overline{\boldsymbol{x}}_{\mathscr{P}}$ over the last 10 measurements, instead of averaging over all measurements. Finally, similarly as the D-MAB algorithm [DFSS08], we reset the indicators $n_{\mathscr{P}}$ and $\boldsymbol{x}_{\mathscr{P}}$ of an arm $\mathscr{P}$ if three consecutive large variations of $\boldsymbol{x}_{\mathscr{P}}$ are observed (if $\|\boldsymbol{x}_{\mathscr{P}}(t)\|_1$ differs from the current average $\|\overline{\boldsymbol{x}}_{\mathscr{P}}\|_1$ by more than 40% three times consecutively). As opposed to the D-MAB algorithm that resets the indicators of all arms, only the indicators of $\mathscr{P}$ are reset, because a capacity change for $\mathscr{P}$ does not yield that all other multipaths are impacted: For example, if the capacity of a single link is modified, multipaths that do not use this link are not affected.

With the threshold $\epsilon_{\min}$, HyMAB converges to a proportion $1-\epsilon_{\min}$ of the optimal rate: There is a tradeoff between converging closer to the optimal rate, or exploring more often. To be more flexible, it is possible to change dynamically the threshold $\epsilon_{\min}$: When the capacities are stable, we use a low value, and increase it when the capacities change. Specifically, whenever the total rate of an estimation $\|\boldsymbol{x}_{\mathscr{P}}(t)\|_1$ differs from the current average $\|\overline{\boldsymbol{x}}_{\mathscr{P}}\|_1$ by more than 40%, $\epsilon_{\min}$ is set to a maximum value of 0.15. If $\|\boldsymbol{x}_{\mathscr{P}}(t)-\overline{\boldsymbol{x}}_{\mathscr{P}}\|_1$ is within 20% of $\|\overline{\boldsymbol{x}}_{\mathscr{P}}\|_1$, $\epsilon_{\min}$ is divided by 2 (with a minimum value of 0.05). This strategy is called *variable ϵ_{min}*. It is robust because when a false positive occurs, it increases only temporarily the probing probability, which therefore degrades the throughput only very slightly. Note that the values chosen for the variable $\epsilon_{\min}$ (minimum and maximum, thresholds for deciding when to change) depend on the conditions of the network (e.g., precision of the busy-time measurements, variability of the link capacities) and need to be set depending on the goals of the source (e.g., react faster or converge closer to the optimal).

We evaluate the variable $\epsilon_{\min}$ strategy by comparing it through simulations with Rexp3 [BGZ14], a MAB strategy defined for dynamically changing rewards; Rexp3 works by defining a *batch size* Δ_T and by resetting the weights it gives to each arm every Δ_T trials. As explained in Section 4.3, and similarly as with UCB1, Rexp3 works only when the player knows the reward each time the arm is played, which is not the case here. We can however evaluate a strategy where UCB1 is replaced in Stage 1 of Algorithm 1 by the Rexp3 strategy, with a fixed probing-probability $\epsilon_{\mathscr{P}}$ (Stage 2 of Algorithm 1). We use the same simulation scenario as in Section 4.4.2, except that at time $t = 25\,000$, the rate of two arms randomly chosen among the $N = 5$ arms are drawn again uniformly at random in $[0,1]$ (i.e., the rates of these two arms change). Figure 4.3 shows the throughput experienced at each trial (averaged over 10 000 random instances) for HyMAB with the variable $\epsilon_{\min}$ strategy and $\lambda = 2$, and for Algorithm 1 with Rexp3 (denoted by Rexp3)

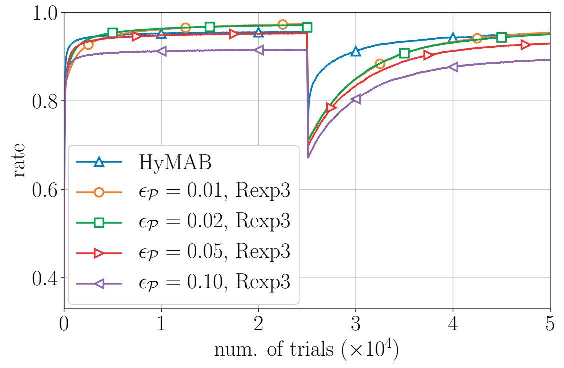


Figure 4.3 – Rate per trial for HyMAB and Rexp3 (simulations).

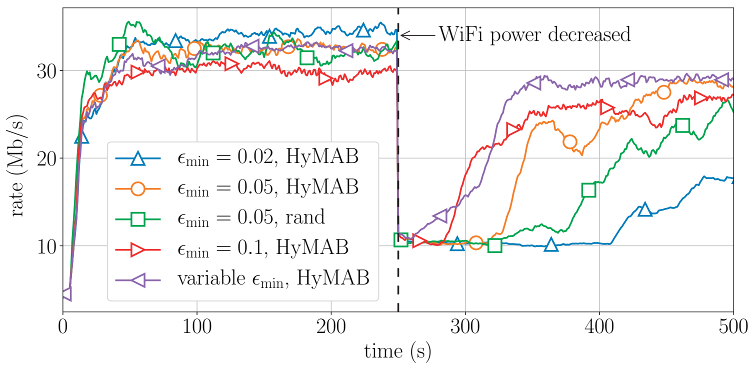


Figure 4.4 – Effect of ϵ_{min} with dynamic conditions (testbed experiments).

with different fixed probing-probabilities $\epsilon_{\mathcal{P}}$. We try several batch sizes Δ_T between 5 and 1 000 and always show the results for the best Δ_T. HyMAB with variable ϵ_{min} re-converges faster than Rexp3 for all probing probabilities.

We also evaluate the different strategies in HyMAB (different fixed ϵ_{min} and variable ϵ_{min}) with testbed experiments. To compare the different values of ϵ_{min} under a controlled environment, we repeat the same experiment with $M = 2$. We send UDP traffic between Nodes 19 and 22 (see Figure A.1 in the Appendix for a map of our testbed) and we force link-capacity changes by reducing the transmit power of WiFi at $t = 250$ s. Such long-term capacity changes happen unpredictably in practice, both for PLC (when appliances are switched on and off [VHT15]) and WiFi (due to varying signals [GC04]). The experiment is repeated five times for each value of ϵ_{min}, and we present averaged results. In Figure 4.4, we show the received throughput for different fixed values of ϵ_{min}: 0.02, 0.05, and 0.1. Here the best multipath uses WiFi-WiFi and PLC-WiFi paths, hence the power reduction causes throughput to drop suddenly; HyMAB adapts by using another multipath, with WiFi-PLC and PLC-PLC paths, which can be observed by the throughput re-increase. When the conditions are stable, a small value of ϵ_{min} (0.02, blue line) achieves a better throughput than a large value (0.1, red line). However, when the capacities change, it takes longer when ϵ_{min} is small to reset the indicators of the arms and to converge again.

Therefore, the smaller $\epsilon_{\min}$ is, the larger the convergence time is. The throughput obtained with a variable $\epsilon_{\min}$, as described above, is shown by the purple line in Figure 4.4. Before the capacities change, it converges to the same throughput as the fixed value $\epsilon_{\min} = 0.05$ (the minimum value of the variable $\epsilon_{\min}$). When the capacities change, it adapts faster than all fixed-value strategies do.

Figure 4.4 also shows the benefits of using the UCB1 strategy in HyMAB instead of a uniform random selection (rand): the received throughput is depicted when both strategies use $\epsilon_{\min} = 0.05$. Using UCB1 significantly improves the convergence rate of HyMAB, because less time is wasted probing the least-good arms. Similarly to what was shown in Section 4.4.2, this confirms the gains provided by employing a MAB strategy such as UCB1, rather than a simpler scheme such as uniform random selection.

The time needed to converge to the new multipath is of the order of a few tens of seconds. Because HyMAB does not target short-term variability but focuses instead on the adaptability to long term variations, this convergence time remains practical. Moreover, the performance after the capacities change and before HyMAB switches to the new best multipath is the performance on the multipath that was used before the conditions have changed, i.e., this is the performance that would have been observed without HyMAB (see also Figure 4.1, where we observe that between the moment when the capacities change and the moment when HyMAB switches multipath, the performance of HyMAB is similar to that of TCP and MPTCP without HyMAB).

4.5.2 Traffic Changes

HyMAB must also handle flows that come and go. The strategy for several flows (described in Section 4.4.3) is optimal for static conditions. Even though the sources do not know the total number of flows in the network, the nodes know the number of flows locally at each link, which makes possible to react very rapidly to flow arrivals/departures: Once per trial, nodes along a multipath $\mathscr{P} \in \mathscr{S}_f$ notify the source of flow f of the maximal number of interfering flows $F_{\mathscr{P}} = \max_{l \in \mathscr{P}} F_l$, with F_l given by Equation (4.4), along with a *flow hash*, a hash value of all the other flows that the nodes on $\mathscr{P}$ overhear. If $F_{\mathscr{P}}$ is increased, the flows, which had a share $1/F_{\mathscr{P}}^{\text{old}}$ of the time-resources, now have only $1/F_{\mathscr{P}}^{\text{new}}$: The source immediately reacts by scaling down the current sending rate vector $\overline{\boldsymbol{x}}_{\mathscr{P}}$ by a factor $F_{\mathscr{P}}^{\text{old}}/F_{\mathscr{P}}^{\text{new}}$. For example, if there was a unique flow and if a second flow appears, the rate vector is divided by 2. When the new flow shares a bottleneck link with the other flows, this scaling yields that the new rate vector is the vector that maximizes System (4.5). Otherwise, this scaling is overly conservative, but it ensures that the medium is unsaturated during the probing phases (no flow uses more resources than its share); the source can then employ the strategy described in Section 4.4.3 and converge to the vector that maximizes System (4.5). This strategy also requires that two flows never probe at the same time, which is likely but not certain, as the probing probability $\epsilon_P(t)$ is low at steady-state but non-zero. In practice, when a flow f_1 probes a multipath $\mathscr{P}_1$, control messages are sent on each path of $\mathscr{P}_1$: Nodes in the interference domain of $\Lambda_{\mathscr{P}_1}$ can overhear these control messages, and thus know that f_1 is probing. If another flow f_2 wants to probe a multipath $\mathscr{P}_2$, nodes belonging to $\mathscr{P}_2$ send a message to the source of f_2 if they know that f_1 is probing (i.e., if they overheard control messages), and the source of f_2 delays the probing phase. If no such message is sent, it means that no other interfering flow is probing.[2] This guarantees that the strategy with several flows converges to the optimal value.

If $F_{\mathscr{P}}$ is decreased, the source can determine, based on the flow hash, if the situation reverses to conditions seen earlier (same active flows); in which case, it restores the rate vectors it had computed. Otherwise, it initializes HyMAB again to find the optimal rates in this unknown configuration. This enables HyMAB to react very fast and to support efficiently short flows that disappear briefly after having appeared. We present experiments with multiple flows in Section 4.6.3.

4.6 Experimental Evaluation

We first describe the implementation of HyMAB. We then present the results of its testbed evaluation, first with a single UDP flow, then with several UDP flows. We then compare MPTCP to HyMAB and TCP combined.

4.6.1 Implementation Details

We implement HyMAB with the *Click Modular Router*[3] [KMC+00] on the 22-node testbed described in Section A in the Appendix. We use $K = 2$ technologies, WiFi and PLC. Our implementation is meant to run as a Linux module in kernel-space, but due to some incompatibility between ath9k and Click, our results are obtained in user space. Handling all packets in userspace incurs significantly more processing delay, and the performance and convergence speed could be improved further if run in kernel-space.

To compute the set of multipaths $\mathscr{S}$, we use the multipath-routing protocol described in Section 2.4 slightly modified to return the set $\mathscr{S}$ of N multipaths in decreasing order of the estimated multipath capacities, i.e., the N multipaths that maximize $C(\mathscr{P})$ as defined by Equation (2.9). Note that returning N multipaths instead of only one does not increase the complexity of the protocol. The multipath-routing protocol requires a complete view of the network: Each node estimates the capacity of its outgoing links by sending unicast frames at low rates and using link-quality information present in the packet headers [VHT15]. It broadcasts a list of its neighbors with the link capacities, and uses this information to compute the interference domains. Because HyMAB employs the N best paths of the routing protocol, this method is robust against estimation errors in the link capacities or interference domains. Once the multipath set $\mathscr{S}$ is chosen, HyMAB does *not* require the knowledge of the link capacities nor of the interference

[2] An architecture such as or similar to software-defined networks would help here, as a centralized controller would decide which flow probes when.

[3] The source code is available at c4science.ch/diffusion/6591/hymab.git.

domains, because it only uses the busy-time measurements to compute the optimal rates of the multipaths. When a capacity change is detected (see Section 4.5.1), $\mathscr{S}$ is computed again.

In Section 2.3, we have shown that the number of paths in the best multipath $\mathscr{P}^*$ is equal to at most the number of technologies. Because we use $K = 2$ technologies, we limit in our experiments the number of paths per multipath to $M = 2$. We set the number of multipaths in $\mathscr{S}$ to $N = 5$: This value is small enough to remain practical, and it is large enough to offer enough multipath diversity so that it is very likely that $\mathscr{P}^* \in \mathscr{S}$, as can be observed in the following experiment: We randomly choose 50 flows (i.e., source-destination pairs) and for each flow, we compute a large number of multipaths to ensure that with very high probability, the best multipath is among them (here, we compute the best 20 multipaths). For each of these 20 multipaths, the optimal rate is found by a brute-force approach: The source sends traffic on all paths of the multipath at all possible rates (with a granularity of 1 Mb/s), and keeps the best experienced throughput. For different values of N, we compute the ratio between the maximal rate among the 20 multipaths $\|\boldsymbol{x}_{20}^{\text{opt}}\|_1$ and the maximal rate among the N best multipaths $\|\boldsymbol{x}_N^{\text{opt}}\|_1$. The maximum and mean of this ratio among the 50 flows are reported in Table 4.1 for different values of N.

In the current version, all nodes run HyMAB. It would be possible to modify HyMAB so that the destination does not need to run HyMAB without impacting the performance: Indeed, because computations are done by the source of the flow and measurements of the busy-time by the source of the links, the last-edge routers can replace the destination in particular for sending the acknowledgements, described later in this section. In a real implementation, this means that for downlink traffic, user nodes can enjoy the benefits of HyMAB *without having to run it*. Uplink traffic is much less likely to require high throughput, hence to need HyMAB. Still, HyMAB can be used between the last-edge router and the gateway to avoid that the client has to run it.

HyMAB works at layer 2.5, between the IP and MAC layers. Figure 4.5 summarizes the main components of HyMAB. When launched, the program creates a virtual tun/tap interface that, with a local IP address, is transparently used by the applications. HyMAB uses source routing, i.e., the path is fully determined at the source and it is set in a layer-2.5 header that is used by the intermediate nodes to forward the packets to the next hop. The path is represented as a list of short hashes (2 bytes) of the MAC addresses of the interfaces along the path. The layer-2.5 header is 17 bytes long: 12 bytes are reserved for the path, limited to 6 hops; 4 bytes are reserved for a sequence number, used to reorder at destination the packets coming from different paths, before delivering them to higher layers; the final byte is reserved to indicate in which phase (probing or exploitation) the packet is sent. Time is split in slots of $D = 400$ ms. At each slot t, the source chooses a multipath $\mathscr{P}$ and either to probe $\mathscr{P}$ (with probability $\epsilon_{\mathscr{P}}(t)$), or to exploit

Table 4.1 – Ratio $\|\boldsymbol{x}_{20}^{\text{opt}}\|_1 / \|\boldsymbol{x}_N^{\text{opt}}\|_1$ for 50 random flows.

N	1	2	4	5
max ratio	1.62	1.50	1.50	1.02
mean ratio	1.08	1.03	1.02	1.00

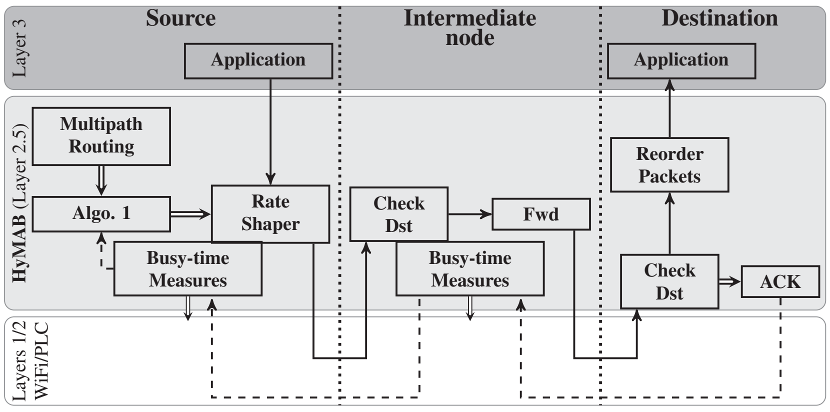


Figure 4.5 – The main components of HyMAB at layer 2.5. Plain-line arrows represent the data flow, dashed-line arrows represent the acknowledgements, and double arrows represent actions on the component. The source determines the multipath set $\mathscr{S}$ (Multipath Routing). $\mathscr{S}$ is used by Algo. 1 that sets the Rate Shaper to the desired vector rate. Using the header of our HyMAB protocol, intermediate nodes check whether they are the destination (Check Dst) and, if needed, forward packets to the next hop (Fwd). Finally, the destination reorders the packets based on a sequence number included in the HyMAB header. Upon reception of a control message sent at each probing phase, the destination sends acknowledgements on the paths (ACK), updated by each intermediate node with the busy-time measurements (Busy-time Measures).

the best multipath found so far, depending on the outcome of Stage 2 in Algorithm 1.

If the source chooses to exploit at time t, it shapes the traffic to send at the current best rate $\overline{\boldsymbol{x}}_{\mathscr{P}_t^*,t}$ on the estimated best multipath $\mathscr{P}_t^*$ during the entire slot. During an exploitation phase, the source does *not* send traffic by batches; the batches are needed only during probing phases. In practice, delays tend to increase rapidly when the busy time approaches 1: We replace the constraint $\boldsymbol{A}_{\mathscr{P}} \cdot \boldsymbol{x} \le \boldsymbol{1}$ in System (2.4) by a slightly more conservative constraint $\boldsymbol{A}_{\mathscr{P}} \cdot \boldsymbol{x} \le (1-\delta)\cdot \boldsymbol{1}$, with a small *constraint margin* $0 \le \delta \ll 1$. Increasing δ decreases the delays, but also decreases the throughput.

If instead the source chooses to probe multipath $\mathscr{P}$ at time t, it follows the procedure described in Section 4.2. For the probing phases, the source uses rate vectors $\boldsymbol{x}_{\mathscr{P}}^{(i)}$ such that all vector components of $\boldsymbol{x}_{\mathscr{P}}^{(i)}$ are equal to zero, except the i-th component, equal to $0.75 \cdot x_i^{\text{sp}}$, where x_i^{sp} is the current best rate for P_i *when used alone* (i.e., single-path). x_i^{sp} is computed for free when computing the optimal rate for the multipath with the busy-time measurements, as described in Section 4.2. This choice of rate vectors $\boldsymbol{x}_{\mathscr{P}}^{(i)}$ is justified by an empirical observation that we made during our experiments: The estimation of the optimal rate is more precise and more robust against measurement imprecisions due to noise when the matrix $\boldsymbol{X}_{\mathscr{P}}$ in Section 4.2) is diagonal, i.e., when the M measurements are carried out on a single path. This choice also ensures that traffic is sent at a sufficient rate even during a probing period (close to be the best single-path rate on each path). The factor 0.75, used only when probing, ensures that no link is close to saturation, which is required in practice to compute the optimal rate on $\mathscr{P}$. If a rate vector is non-admissible (e.g., because of dynamic conditions), i.e., if a $\mu_l^{(i)}$ is measured close to 1, the source removes the measurement and repeats the trial with the rate vector divided by 2. At the beginning of a probing phase, the source waits for a short time τ (of the order of the delay from source to destination) so that all packets sent during the previous trial reach destination. Control messages are then sent on all paths of $\mathscr{P}$. These control messages enable all nodes along the paths to initialize and measure the busy times $\mu_l^{(i)}$ for all links $l \in \Lambda_{\mathscr{P}}$, as described in Section 4.2. After the silent slot (described in Section 4.4.3), the source starts by sending traffic at rate $\boldsymbol{x}_{\mathscr{P}}^{(1)}$ during D/M, by batches of 100 packets for WiFi, 200 for PLC. The destination then sends an acknowledgement back on all the paths of the multipath $\mathscr{P}$. Nodes along the paths update this acknowledgement with the measured busy times: When it reaches the source of the flow, the acknowledgement contains all busy-time measurements $\mu_l^{(1)}$. When $M \geq 2$, the procedure is repeated with all rate vectors $\boldsymbol{x}_{\mathscr{P}}^{(i)}$. At the end of the trial, the source knows all measurements $\mu_l^{(i)}$ and can compute all $\boldsymbol{\alpha}_{\mathscr{P},l}$, as described in Section 4.2. Finally, to compute an estimation of the optimal rate, the source reduces $\boldsymbol{A}_{\mathscr{P}}$ by removing all vectors $\boldsymbol{\alpha}_{\mathscr{P},l}$ such that $\boldsymbol{\alpha}_{\mathscr{P},l} \preceq \boldsymbol{\alpha}_{\mathscr{P},l'}$ for another link l', and solves System (2.4) with the reduced $\boldsymbol{A}_{\mathscr{P}}$. In our experiments, the number of rows in the reduced $\boldsymbol{A}_{\mathscr{P}}$ is always less than 5, and the computation of the optimal rate, including that of the multipath-impact vectors, is done in less than 2 ms.

4.6.2 Testbed Results with a Single Flow

To evaluate the performance of HyMAB, we first compute the optimal rate by the brute-force approach described in Section 4.6.1. This approach is not practical, as it requires sending traffic at a large number (quadratic in the number of paths) of non-optimal rates and yields long convergence times. We compare the optimal rate obtained by this brute-force approach to the rate achieved by HyMAB. In the following, we use a small constraint margin $\delta = 0.05$ (value that reduces the delays significantly while decreasing only slightly the throughput) and $\lambda = 2$. In Section 4.6.2, where we study the convergence of HyMAB to the optimal value, we use a fixed $\epsilon_{\min} = 0.02$; in the following sections, we use a variable $\epsilon_{\min}$, as described in Section 4.5.1.

Flow A-B denotes a flow between Node A and Node B. We first show an example of how HyMAB works for Flow 17-6. The (imperfect) routing protocol returns $N = 5$ multipaths $\mathscr{P}_1, \ldots, \mathscr{P}_5$, ordered by decreasing estimated-capacity. The rates obtained by brute-force approach ($\mathscr{P}_1$: 23 Mb/s, $\mathscr{P}_2$: 30 Mb/s, $\mathscr{P}_3$: 29.5 Mb/s, $\mathscr{P}_4$: 20 Mb/s, and $\mathscr{P}_5$: 15 Mb/s) indicate that the best path for the routing protocol, $\mathscr{P}_1$, is *not* the actual best multipath (this is the case for 44% of our 50 experiments presented below), which shows that exploring several multipaths is indeed useful. Figure 4.6 shows the experiment for Flow 17-6 with HyMAB. $\mathscr{P}_2$ consists of two paths, denoted by P_1 and P_2. $\mathscr{P}_3$ uses P_1 and another path, denoted by P_3. $\mathscr{P}_1$, $\mathscr{P}_4$, and $\mathscr{P}_5$ use other paths that are used only during probing phases. The probing probability $\epsilon_{\mathscr{P}}(t)$ rapidly decreases, and at the end, the source spends most of its time exploiting the best multipath. HyMAB converges to the true best multipath $\mathscr{P}_2$; because $\mathscr{P}_2$ and $\mathscr{P}_3$ are very close, it takes some time to converge: Until $t \approx 650$ s, the estimated best arm $\mathscr{P}_t^*$ is $\mathscr{P}_3$. Nevertheless, it sends traffic at a rate very close to the best one (30 Mb/s) throughout the experiment.

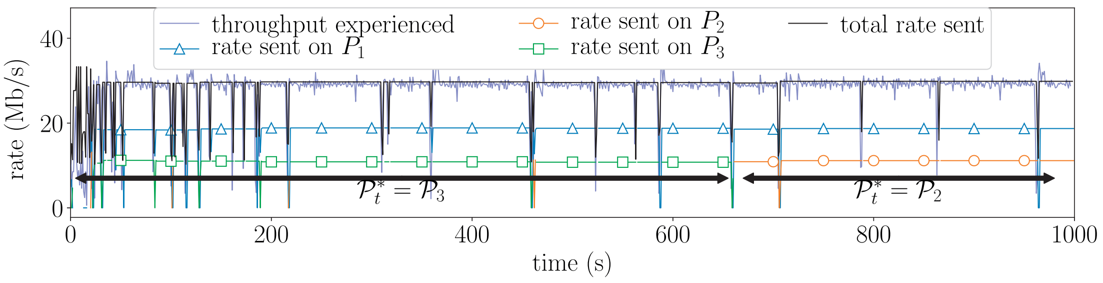


Figure 4.6 – Single experiment for Flow 17-6. The throughput experienced is shown along with the rate sent on the different paths and the total rate sent. Throughput drops correspond to probing phases, more frequent for small t ($t \leq 50$ s, when optimal rates are unknown). After 50 s, exploitation of the best found multipath dominates.

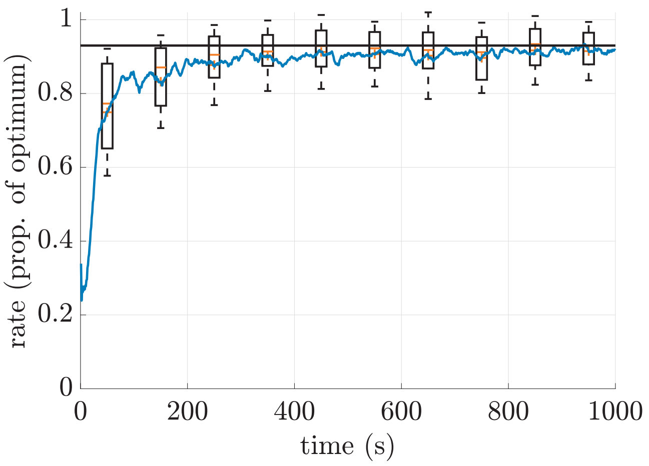


Figure 4.7 – Comparison of HyMAB with optimal brute-force rate, UDP traffic, for 50 flows.

We now compare HyMAB to the optimal brute-force rate on 50 randomly selected source-destination pairs. Figure 4.7 shows the rate received as a proportion of the optimal rate obtained by brute-force, averaged over all 50 runs. The box plot shows the median (red bar), the 25^{th} and 75^{th} percentiles (box), and the standard deviation (whiskers). With $\delta = 0.05$ and $\epsilon_{\min} = 0.02$, we expect to converge on average to 93% of the optimal throughput (black line). Indeed, it converges very close to this value; this shows that, in a home hybrid mesh network, HyMAB succeeds in finding the optimal multipath and the optimal rate at which traffic is sent on this multipath.

4.6.3 Testbed Results with Several Flows

We describe an experiment with two contending flows; all use HyMAB. We first run Flow 9-8; after 500 s, we run Flow 7-13 that interferes with Flow 9-8; after 1 000 s, we shut down Flow 9-8, and finally run it again after 1 500 s. In Figure 4.8, we show the rates sent and received for the two flows. The flow sources react extremely rapidly (a few hundreds of milliseconds): Following

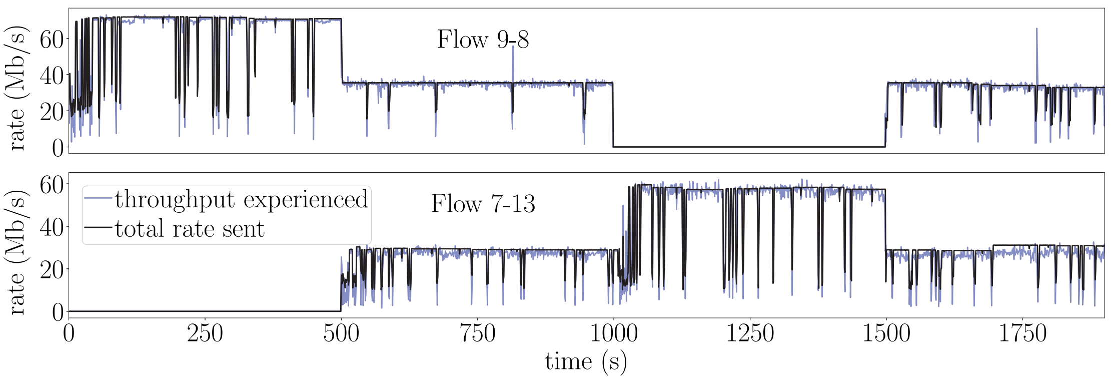


Figure 4.8 – Experiment with two flows. HyMAB fairly shares the resources and reacts very fast, in particular with conditions learned earlier ($t = 1\,500$ s).

the strategy described in Section 4.5.2, Node 9 divides the sending rate by 2 as soon as it detects Flow 7-13. When the flows contend, Nodes 9 and 7 continue to explore, and they converge to the solution of System (4.5), with fairly shared resources. When Flow 9-8 is shut down, Node 7 initializes HyMAB again for Flow 7-13 in isolation and converges to the optimal solution. When Flow 9-8 is run again, Nodes 9 and 7 immediately switch back to the rates last computed when the two flows were present. They continue to explore to adapt to dynamic conditions. This experiment shows that HyMAB efficiently handles contending flows, with a very fast reaction and a fair sharing of the resources.

4.6.4 Interaction of HyMAB with TCP

Here, we study the interaction of HyMAB with TCP. In any exploratory multipath protocol, such as HyMAB, the probing phases cause throughput drops, either because a sub-optimal multipath is explored, or because probing requires sending traffic at a sub-optimal rate. These drops are interpreted by TCP as a congestion signal. Consequently TCP decreases the congestion window, hence the sending rate. After a probing phase, it takes some time (up to a few seconds) for the current versions of TCP to converge back to the rate supported by an exploitation phase. In HyMAB, we alleviate this problem by buffering packets at the source during probing phases, which smoothes the effects of probing. When the source of a TCP flow is in the home network, (i.e., when we control it), it would be easy to completely solve the issue by implementing a specific version of TCP that, transparent for the destination, would use different congestion windows during probing and exploitation phases. Implementing this modified version of TCP is left for future work, and the results here are obtained with a classic version of TCP; they are only a lower bound of the performance that can be reached. We compare MPTCP to HyMAB and simple TCP combined. MPTCP uses the *best* multipath (that can consist in one or two paths), i.e., the multipath found optimal by HyMAB. This choice favors MPTCP: In reality, MPTCP typically uses the multipath returned by a multipath-routing protocol that, as we have seen in Section 4.6.2, is not necessarily the optimal one.

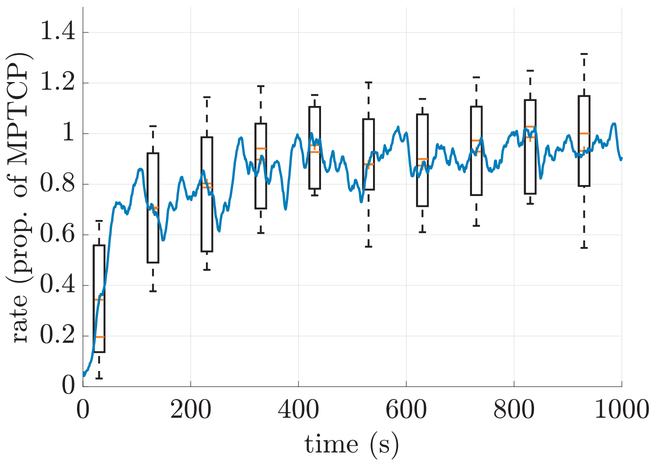


Figure 4.9 – Comparison of MPTCP with HyMAB and TCP combined for 20 random flows. HyMAB achieves performance very close to MPTCP after convergence.

Figure 4.9 shows the rate achieved by HyMAB, as a proportion of the rate achieved by MPTCP, averaged over 20 randomly selected isolated flows. On average, HyMAB and MPTCP are very close. The continuous exploration, which enables HyMAB to adapt much better to dynamic conditions, causes a slight variability increase. But overall, its cost is small compared to its advantages, among which is the better adaptability to dynamic conditions (see Figure 4.1) but also the absence of multi-homing requirements (see also Section 4.7). In addition, because of some incompatibility between ath9k and Click, the current version of HyMAB is implemented in userspace, which induces high processing delays: The RTT with HyMAB is about 10 times higher than with MPTCP (for a single-hop flow, it is about 30 ms with HyMAB, and about 3.5 ms with MPTCP). An implementation in kernel-space could improve significantly the performance of HyMAB with TCP by reducing the end-to-end delays, to which TCP is extremely sensitive [PFTK98].

4.7 Related Work

MAB has been widely studied in many contexts after the seminal works by Thompson [Tho33], Lai and Robbins [LR85], and Auer et al. [ACBF02]. Routing was identified early on as a potential application for MAB strategies [ACBFS95]. However, only a few papers specifically investigate this application; these works address the problem of finding the shortest path [AK04] or the path with minimal delay [BL94], [AK08], and not the problem of maximizing throughput, or that of multipath routing. In addition, they are not validated experimentally on a testbed. Quite a few MAB strategies are proposed [ACBF02], most of them when the rewards of the arms are stochastic. In this work, we consider the case where exploration is costly. More importantly, we implement this strategy on a testbed, showing its practical usability. As opposed to other multipath routing protocols [e.g., GGSE01, LC02, AKK04, LLZ06, TM06, TT07, THT08, TTAE09, GRS11, DQZ$^+$15] that use heuristics to build a single multipath and consequently do not guarantee the optimality of the result, our approach with MAB enables our algorithm to explore *N* different multipaths to find the best one. In addition, HyMAB can easily adapt to dynamic conditions that would require switching multipath.

4.8 Summary

In this chapter, we have proposed HyMAB, an algorithm that finds the best multipath in hybrid mesh networks with shared-medium technologies. HyMAB exploits the MAB framework to successfully address the tradeoff between exploitation and exploration, in contrast to current protocols that typically keep the same multipath as long as it is valid. It also finds the optimal rate at which traffic is sent on each path without having to measure link capacities or interference domains. It works efficiently when several flows are present. We have implemented HyMAB on a testbed of WiFi and PLC nodes, thus showing in practice its optimality and adaptability to dynamic conditions. To the best of our knowledge, this is the first implementation of a MAB strategy in the context of routing and congestion control.

The MAB foundations presented in this chapter could be employed for other communication technologies and contexts, such as IoT or vehicular networks. The specifics of HyMAB design are technology independent. To tackle dynamic conditions and network intricacies, we have proposed guidelines on how to adjust our design.

Latency Part II

5 Latency in Hybrid Time-Varying Networks

5.1 Introduction: Time Variability in Networks

In the first part of this dissertation, we have shown that hybrid networks and multipath can significantly improve the throughput experienced by users. But throughput is rarely the only metric that needs to be optimized, and hybrid networks open interesting perspectives in terms of latency, power consumption, reliability, privacy, etc. In this chapter,[1] we focus on latency in hybrid networks. Latency in today's networks is paramount and is acknowledged as being fundamental in tomorrow's networks, as illustrated by the standardization efforts towards ultra-reliable low-latency communications (URLLC) in wireless networks [vis15].

The time variability of the service rate has a strong impact on latency: If the service rate of a link (i.e., the link capacity) decreases, packets accumulate in the queue of the network interface, which increases the packet delay, i.e., the time interval between the moments when the packet is sent by the source and received by the destination (in the following, latency and delay are used interchangeably). Because of varying signals (fading, multi-path effects, etc.), WiFi typically presents a behavior with time-varying service rates. This is illustrated by Figure 5.1, where we show the instantaneous physical rate, obtained by the modulation and coding scheme (MCS) index employed, averaged over ten packets, in a 25 seconds WiFi trace, with no other traffic. Clearly, the MCS index is not constant; it varies between (mostly) four areas with two dominant states, a *high-rate* state (about 80 Mb/s) and a *low-rate* state (about 60 Mb/s). These service rate variations with high-rate and low-rate states can be observed in many contexts. They happen for example when several users employ the same channel and interfere with each other: If Alice is streaming a video or downloading a file with WiFi or LTE and no other user is active, Alice has a high throughput: she is in a high-rate state. If another user Bob is active (e.g., he is surfing the Web), Alice's throughput decreases: she is in a low-rate state. These variations can also be observed on WAN paths if one WAN router alternates between high-traffic loads (low-rate state) and low-traffic loads (high-rate state); or in data-centers where servers typically alternate between high-rate and low-rate states, that can be caused by a higher load of the server, but

[1]This chapter is based on the paper by Henri et al. [HST18].

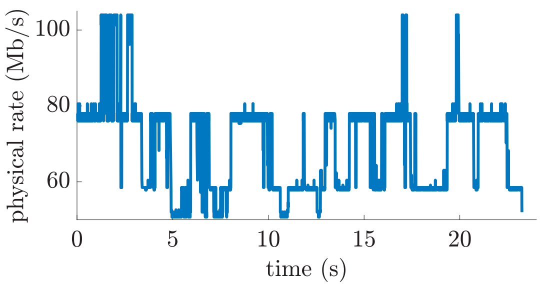


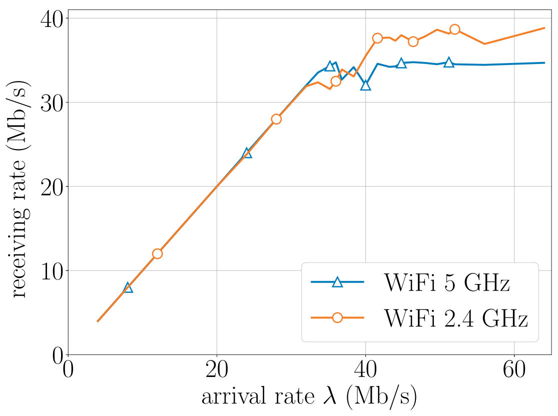
Figure 5.1 – Physical rate of a WiFi link, averaged over ten packets.

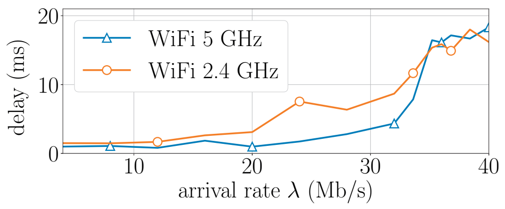
also by many external reasons such as garbage collection, network interrupts, or background work [GHBSW$^+$17]; or at home with PLC, because switching on and off appliances changes the electrical impedance and can cause link-capacity drops [VHT15].

To illustrate the effect that variability has on packet delays in hybrid wireless networks, we carry out the following uncontrolled experiment. We run traffic between two nodes that each have two WiFi interfaces; the first WiFi interface operates in the 2.4 GHz band, the second interface operates in the 5 GHz band, with a 20 MHz band for each. Both WiFi interfaces use the 802.11n protocol and there is a single antenna per interface. The nodes and the WiFi interfaces are described in details in Section A in the Appendix. The two nodes are approximately 15 meters apart in two different offices, with two walls between them. Lower frequencies are known to be less attenuated by walls than higher frequencies, and we observe that the maximum instantaneous rate between the two nodes is about 35 Mb/s in the 5 GHz band, whereas it is about 45 Mb/s in the 2.4 GHz band. However, in the building where the experiments take place, the 2.4 GHz band is also used by the WiFi network of the university, whereas no other node uses the 5 GHz band. When other nodes use the university WiFi network, the throughput on the 2.4 GHz link decreases. Consequently, the variability is larger in the 2.4 GHz band.

We run our experiments on a weekday, when the WiFi network of the university is more loaded. We send UDP traffic during 30 seconds at various rates with `iperf`, first in the 2.4 GHz band, then in the 5 GHz band, and we measure the one-way delay of the packets by using `tcpdump` on each interface.[2] The experiment is repeated five times for each interface and each rate, and we present averaged results. Figure 5.2 (left) shows the receiving rate; the average throughput achieved in the 2.4 GHz band (about 38 Mb/s) is slightly higher than that achieved in the 5 GHz band, but it is below the maximum instantaneous throughput of 45 Mb/s. This shows that the variability in the 2.4 GHz band is indeed quite large. In contrast, the throughput achieved in the 5 GHz is close to its maximum instantaneous throughput. Figure 5.2 (right) shows the average packet delay (top) and jitter (bottom). Even though the average throughput is slightly higher in the 2.4 GHz band, delay and jitter both are smaller in the 5 GHz band (delay is up to 4.5x smaller when the arrival rate is 24 Mb/s). This illustrates that, as expected, variability has a high impact on the delays; we study this impact in this chapter.

[2]The two nodes are synchronized with the Precision Time Protocol (PTP) [PTP08] that offers a precision of a few microseconds (minimum delays are of the order of the millisecond).





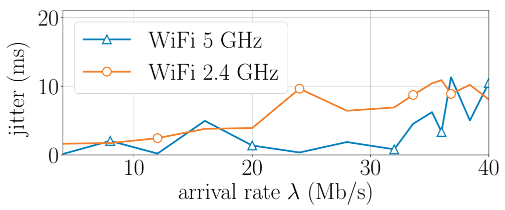


Figure 5.2 – Throughput achieved (left), delays (top right), and jitter (bottom right) on a testbed with 2.4 GHz and 5 GHz WiFi.

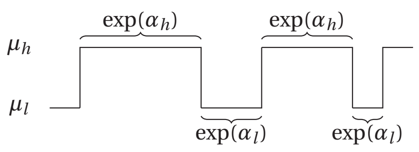


Figure 5.3 – Sample path of the service rate.

Outline of the Chapter This chapter is structured as follows. In Section 5.2, we describe a model that captures the variability of the service rate and that enables us to study the effect on delays of this variability, in two different settings: (*i*) For a given arrival rate, we want to choose between two paths (i.e., in the context of hybrid networks, between two technologies). Increasing the average service rate clearly decreases the average delays; to isolate the effect of time variability, we assume, in the first setting, that the two technologies have the same *average* service rate, but different *variances* of the service rate, and we want to find which technology yields the smallest average delays. This first setting is studied in Section 5.3. (*ii*) We next study the second setting where two technologies, potentially with different average service rates, can be used simultaneously, and traffic be split between them (i.e., we consider multipath routing). We want to study the effect of the splitting scheme. This second setting is studied in Section 5.4. We discuss related work in Section 5.5 and close the chapter with a summary in Section 5.6.

5.2 Queueing Model with Time-Varying Service Rates

We model a network interface by a queue with i.i.d. exponentially-distributed arrivals with rate parameter λ. The server operates in two different regimes: A *low-rate* state in which the packets are sent (or the jobs are completed) with exponential service rate μ_l, and a *high-rate* state in which the packets are sent with exponential service rate μ_h, with $0 < \mu_l < \mu_h$. From now on, we simply write low state and high state for low-rate state and high-rate state. The sojourn times in low and high states are exponentially distributed, with parameter α_l when in low state and α_h

when in high state. The service model is thus a continuous-time Markov chain, with a sample path illustrated in Figure 5.3. If $(Q(t), S(t))$ denotes the number of packets and the state of the server at time t, and if $r((q_1, s_1), (q_2, s_2))$ denotes the transition rates from state (q_1, s_1) to state (q_2, s_2), then the only non-zero transition rates are

$$\begin{aligned} r((q,s),(q+1,s)) &= \lambda, && \text{for all} \quad q \geq 0, \quad s \in \{h,l\}, \\ r((q,s),(q-1,s)) &= \mu_s, && \text{for all} \quad q \geq 1, \quad s \in \{h,l\}, \\ r((q,h),(q,l)) &= \alpha_h, && \text{for all} \quad q \geq 0, \\ r((q,l),(q,h)) &= \alpha_l, && \text{for all} \quad q \geq 0. \end{aligned}$$

This model with heterogeneous time-varying service rates was studied as early as 1971 by Yechiali and Naor [YN71]. However, the expression of the average delays is quite complex; it involves computing the root of a cubic equation (see also in the Appendix). Even though it can be computed in principle, its explicit expression is very complex (it would take dozens of line to write it explicitly), and few works give analytical results that can be intuitively understood. Ross conjectured in 1978 that increased variability leads to increased averaged delays [Ros78]. Variability was expressed by the sole parameter $\alpha = \alpha_h + \alpha_l$. A lower α means longer sojourn times in the “high” and “low” states: It leads to a higher heterogeneity of the service rates, i.e., a larger variability. In contrast, with very short sojourn times in each of the states (i.e., very frequent transitions and large α), the queue with heterogeneous service rates performs close to a homogeneous M/M/1 queue with an averaged service rate [GHBWY06]. Ross’s conjecture was proven in 1981 by Rolski [Rol81] in the particular case $\mu_h = \mu_l$, with different arrival rates in high and low states: The average delay in this scenario is a decreasing function of α. The general case, with different service rates ($\mu_h \neq \mu_l$), was studied in 2006 by Gupta et al. [GHBWY06]. They show that the average queue size is always monotonic in α, but not always decreasing. In our setting where the arrival rate is the same in both states, the average queue size, hence, the average delay is a decreasing function of α, as conjectured by Ross.

5.3 Time-Varying Hybrid Networks: Choosing the Best Technology

In this section, we consider a hybrid network with two technologies, and we study the single-path case, where the client wants to choose the best technology (with respect to delays). We model the two technologies as two queues that follow the model described in Section 5.2, with respective parameters $\mu_{l,i}$, $\mu_{h,i}$, $\alpha_{l,i}$, and $\alpha_{h,i}$ for $i = 1, 2$. The model of this section is illustrated by Figure 5.4. In the remainder of this chapter, we study only average delay, and sometimes refer to it simply as delay; the delay of Queue i as a function of the arrival rate λ is denoted by $D_i(\lambda)$. We want to compare the two queues and to find out which one yields the lowest delay.

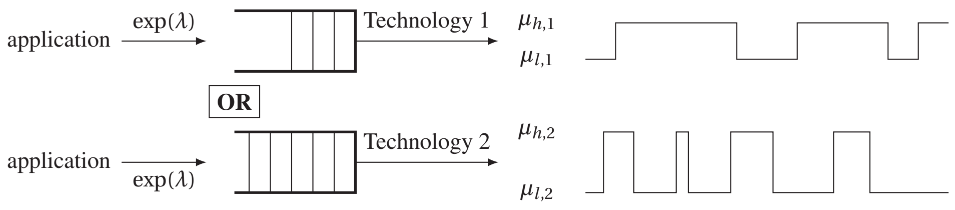


Figure 5.4 – Model with two technologies and single-path.

5.3.1 Analysis

In this section, we are interested in studying how the variability of the service rates, rather than the average service rate itself, affects the average delays. Indeed, it is obvious that for a fixed level of variability, an increased average service rate yields lower delays. Intuitively, we expect that a larger variability should yield larger delays. This is what happens for an M/G/1 queue, where variability is expressed by the variance of the service times: The well-known Pollaczek–Khinchine formula states indeed that when the two queues have the same average service rate $\hat{\mu}$ and the same arrival rate $\lambda \in [0, \hat{\mu})$, the queue with the largest delays is always the queue with the largest variance of the service times. Our model with heterogeneous time-varying service rates is different: In an M/G/1 queue, the service times are i.i.d., contrary to our model where they are drawn from two different exponential distributions, depending on the state (low or high) the system is in. We show in this section that, although this is often the case, larger variability does *not* always yield larger delays; we also show that for certain values of $\mu_{l,i}$, $\mu_{h,i}$, $\alpha_{l,i}$, and $\alpha_{h,i}$, the queue with the largest delays is not the same for all arrival rates λ.

Let R_i denote the random variable for the service rate, taking values in $\{\mu_{l,i}, \mu_{h,i}\}$ and distributed according to the stationary distribution of the process illustrated in Figure 5.3. The average service rate of Queue i for $i \in \{1,2\}$ is given by

$$\hat{\mu}_i = \mathbb{E}[R_i] = \frac{\mu_{h,i}/\alpha_{h,i} + \mu_{l,i}/\alpha_{l,i}}{1/\alpha_{h,i} + 1/\alpha_{l,i}} = \frac{\alpha_{l,i}\mu_{h,i} + \alpha_{h,i}\mu_{l,i}}{\alpha_{h,i} + \alpha_{l,i}},$$

For Queue i to be stable, we must have $\lambda < \hat{\mu}_i$ [YN71]. Because we assume in this section that the two queues have the same average service rate, we have $\hat{\mu}_1 = \hat{\mu}_2 \doteq \hat{\mu}$. In particular, the two queues have the same stability region $\lambda \in [0, \hat{\mu})$. Previous work has studied the effect on the delays of the parameter $\alpha_i \doteq \alpha_{h,i} + \alpha_{l,i}$ when μ_h and μ_l are fixed [GHBWY06, Rol81, Hey82]. For this reason, we assume in this section that $\alpha_1 = \alpha_2 \doteq \alpha$ and our goal is to study the effect of the other parameters. We express variability by the variance V_i of the service rate of Queue i, which can be written as

$$V_i = \mathrm{Var}[R_i] = \frac{\alpha_{l,i}\mu_{h,i}^2 + \alpha_{h,i}\mu_{l,i}^2}{\alpha_{h,i} + \alpha_{l,i}} - \hat{\mu}^2. \tag{5.1}$$

Note that $V_i = 0$ if and only if Queue i is homogeneous ($\mu_{h,i} = \mu_{l,i}$).

The delay for Queue i for $i \in \{1,2\}$ is determined by five parameters: $\mu_{h,i}$, $\mu_{l,i}$, $\alpha_{h,i}$, $\alpha_{l,i}$, and λ. When $\hat{\mu}$ and α are fixed, there are still three degrees of freedom, and the delay depends both on the service rates (μ's) and on the transition rates (α's). The first natural question that we want to answer is what happens when the variance V is fixed (i.e., $V_1 = V_2$). Theorem 4 shows that determining which queue has the largest delays now only depends on the service rates and not on the transition rates, and that the best queue is the same for all arrival rates. The proofs of the theorems of this section are given in the Appendix.

Theorem 4. *Let us assume that $\hat{\mu}_1 = \hat{\mu}_2$ and $\alpha_1 = \alpha_2$. For $i \in \{1,2\}$, let*

$$\pi_i = \mu_{h,i}\mu_{l,i}. \tag{5.2}$$

If $V_1 = V_2$, then for all arrival rates $\lambda \in (0,\hat{\mu})$, $D_1(\lambda) \geq D_2(\lambda)$ if and only if $\pi_1 \leq \pi_2$, with equality if and only if $\pi_1 = \pi_2$. Conversely, if $\pi_1 = \pi_2$, then for all arrival rates $\lambda \in (0,\hat{\mu})$, $D_1(\lambda) \geq D_2(\lambda)$ if and only if $V_1 \geq V_2$, with equality if and only if $V_1 = V_2$.

With $\hat{\mu}$ and α fixed, the delay for Queue i is fully determined by V_i (defined by Equation (5.1)), π_i (defined by Equation (5.2)), and λ. We now want to determine the queue with the lowest delays when neither the variance V nor the product of the two service rates π are fixed, i.e., when $V_1 \neq V_2$ and $\pi_1 \neq \pi_2$. Theorem 4 shows that for all arrival rates λ, the average delay is an increasing function of V_i when $\pi_1 = \pi_2$, and a decreasing function of π_i when $V_1 = V_2$. For this reason, we expect that the average delay increases when V_i increases and π_i decreases, and that it decreases when V_i decreases and π_i increases. Theorem 5 shows that this is indeed true.

Theorem 5. *Let us assume that $\hat{\mu}_1 = \hat{\mu}_2$ and $\alpha_1 = \alpha_2$. If $V_1 > V_2$ and $\pi_1 < \pi_2$, then for all arrival rates $\lambda \in (0,\hat{\mu})$, $D_1(\lambda) > D_2(\lambda)$. Conversely, if $V_1 < V_2$ and $\pi_1 > \pi_2$, then for all arrival rates $\lambda \in (0,\hat{\mu})$, $D_1(\lambda) < D_2(\lambda)$.*

When μ_h is fixed ($\mu_{h,1} = \mu_{h,2}$), V_i is a decreasing function of $\mu_{l,i}$ and π_i is an increasing function of $\mu_{l,i}$. (Note that because $\hat{\mu}$ and μ_h are fixed, increasing $\mu_{l,i}$ requires increasing the sojourn times in the low state.) A corollary of Theorem 5 is consequently that if $\mu_{h,1} = \mu_{h,2}$, the delay is an increasing function of V_i (and a decreasing function of $\mu_{l,i}$).

The situation becomes more complex when both V_i and π_i decrease, or both V_i and π_i increase. For certain values of V_i and π_i, one queue yields lower delays for all arrival rates, and in that case, we show that this must be the queue with the lowest variance V_i. However, we show that for certain values of V_i and π_i (precise conditions are given in the Appendix), determining which queue yields lower delays depends on the arrival rate λ. In particular, for certain arrival rates, the queue with the lowest delays is the queue with the largest variance V_i (Corollary 2).

Theorem 6. *Let us assume that $\hat{\mu}_1 = \hat{\mu}_2$ and $\alpha_1 = \alpha_2$. There are values of V_i and π_i such that $D_1(\lambda) - D_2(\lambda)$ changes sign in $(0,\hat{\mu})$.*

Theorem 7. *Let us assume that $\hat{\mu}_1 = \hat{\mu}_2$ and $\alpha_1 = \alpha_2$. If for all arrival rates $\lambda \in (0,\hat{\mu})$, $D_1(\lambda) \neq D_2(\lambda)$, then for all $\lambda \in (0,\hat{\mu})$, $D_1(\lambda) > D_2(\lambda)$ if and only if $V_1 > V_2$.*

Corollary 2. *For certain values of V_i and π_i, there is $\lambda_0 \in (0, \hat{\mu})$ such that for all arrival rates $\lambda \in (0, \lambda_0)$, $V_1 > V_2$ and $D_1(\lambda) < D_2(\lambda)$, or $V_1 < V_2$ and $D_1(\lambda) > D_2(\lambda)$.*

In Section 5.3.2, we present numerical and testbed evidences that show that by using the queue with the largest variance, the delay gain provided can be significant.

The next step is to understand why the queue with the lowest variance can sometimes yield larger delays. We make the following conjecture, that appears to hold numerically.

Conjecture 1. *Let us assume that $\hat{\mu}_1 = \hat{\mu}_2$ and $\alpha_1 = \alpha_2$. If $V_1 > V_2$ and $\mu_{l,1} < \mu_{l,2}$, then for all arrival rates $\lambda \in (0, \hat{\mu})$, $D_1(\lambda) > D_2(\lambda)$. Conversely, if $V_1 < V_2$ and $\mu_{l,1} > \mu_{l,2}$, then for all arrival rates $\lambda \in (0, \hat{\mu})$, $D_1(\lambda) < D_2(\lambda)$.*

This would mean that a necessary condition for the queue with the largest variance to yield the smallest delays would be to have its service rate in the low state ($\mu_{l,i}$) be larger than that of the queue with the lowest variance. (Note that this is *not* a sufficient condition.) Therefore, in low state, the size of the queue with the smallest variance would increase faster than the size of the queue with the largest variance, which can cause larger average delays despite a smaller variance. This conjecture has another direct consequence: If $\alpha_{h,1} = \alpha_{h,2} \doteq \alpha_h$ and $\alpha_{l,1} = \alpha_{l,2} \doteq \alpha_l$, then for all $\lambda \in (0, \hat{\mu})$, $D_1(\lambda) > D_2(\lambda)$ if and only if $V_1 > V_2$: If we fix the transition rates α_h and α_l, then the average delay is an increasing function of the variance. If α_h and α_l are fixed, we can already prove that if $\alpha_l \mu_{h,i} \geq \alpha_h \mu_{l,i}$, then $V_1 \geq V_2$ if and only if $\pi_1 \leq \pi_2$ (see Lemma 11 in the Appendix), and it follows therefore from Theorem 5 that for all $\lambda \in (0, \hat{\mu})$ $D_1(\lambda) > D_2(\lambda)$ if and only if $V_1 > V_2$. This is in particular the case if $\alpha_l \geq \alpha_h$, i.e., if the queues spend more time in the high state.

Note that if $\alpha_1 \neq \alpha_2$, it is possible to have $D_1(\lambda) < D_2(\lambda)$ with $V_1 > V_2$ and $\mu_{l,1} < \mu_{l,2}$ (e.g., when α_1 is large and α_2 is small).

5.3.2 Experimental Results

In contrast to the experiments of Section 5.1 (presented in Figure 5.2), where we did not have control over the sources of variability, we want in the following to be able to control the parameters $\mu_{h,i}$, $\mu_{l,i}$, $\alpha_{h,i}$, and $\alpha_{l,i}$, in order to compare the experimental results with the analytical results. This is achieved with WiFi by setting the modulation and coding scheme (MCS) used by the interface. The ath9k driver enables us to set the MCS to the desired value. To avoid external sources of variability, we run all the experiments of this section in the 5 GHz band, not used by any other user. The analysis assumes infinite queues. In practice, the queues of the network interfaces are finite. We run our experiments in two modes: Either packets are queued in an infinite queue at the application level, by using the Click Modular Router [KMC$^+$00]; or we use the default finite queues of the network interfaces.

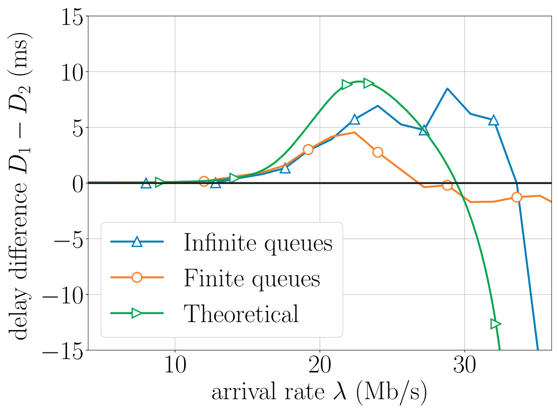


Figure 5.5 – Delay difference $D_1 - D_2$ between Queue 1 (smaller variance) and Queue 2. Analytical and testbed results.

We use the following values, with the service rates (μ) in packets/s, and the transition rates (α) in transitions/s: $\mu_{h,1} = 4196$ (MCS 5, about 46 Mb/s), $\mu_{l,1} = 1562$ (MCS 2, about 17.5 Mb/s), $\mu_{h,2} = 4687$ (MCS 6, about 52 Mb/s), $\mu_{l,2} = 2089$ (MCS 3, about 23 Mb/s), $\alpha_{h,1} = 2.92$, $\alpha_{l,1} = 7.01$, $\alpha_{h,2} = 4.84$, and $\alpha_{l,2} = 5.16$. These values are chosen randomly among some for which the smaller delays are achieved by the queue with larger variance for certain arrival rates, as shown by Corollary 2. More precisely, we choose values such that Equation (B.21) in the Appendix is verified. We have $\hat{\mu}_1 = \hat{\mu}_2 = 38.4$ Mb/s, $V_1 = 1.33 \times 10^6$, and $V_2 = 1.67 \times 10^6$: Queue 1 is the queue with smallest variance. The theoretical delay difference $D_1 - D_2$ obtained from Equation (B.13) in the Appendix is shown in Figure 5.5. For arrival rates lower than about 30 Mb/s, the queue with smallest delays is Queue 2, the queue with largest variance.

We carry experiments with two nodes (described in Section A in the Appendix), located in the same office. We send UDP traffic during 20 s on each queue and at various rates, and we measure the packet delays. Sending traffic on Queue i means that the WiFi interface switches between the MCS that corresponds to $\mu_{h,i}$ and $\mu_{l,i}$, with exponentially-distributed sojourn times with respective parameters $\alpha_{h,i}$ and $\alpha_{l,i}$. For all the testbed experiments of this chapter (Section 5.3.2 and Section 5.4.2), no assumption is made on the distribution of the service times of the packets, and only the average service rate is set; as opposed to the analytical part where they were assumed to be exponentially distributed, the service times depend on the wireless interface and their distribution is unknown. For the packet arrivals, we set the average arrival rate and we try exponential and deterministic distributions, but the choice has no effect on the delays. The results are shown with a deterministic distribution (i.e., the inter-arrival times between packets are constant). The experiments are repeated five times and we present averaged results. We first check that the average receiving rate is the same for both queues, which is the case (see Figure 5.6, left). As expected, it converges to approximately $\hat{\mu} = 38.4$ Mb/s for the infinite queues. The experimental average delay-difference is shown in Figure 5.5. It matches very well the analytical results, which shows that the queue with largest variance can indeed offer the smallest delays for certain arrival rates. This also shows that the best queue in terms of delays depends on the arrival

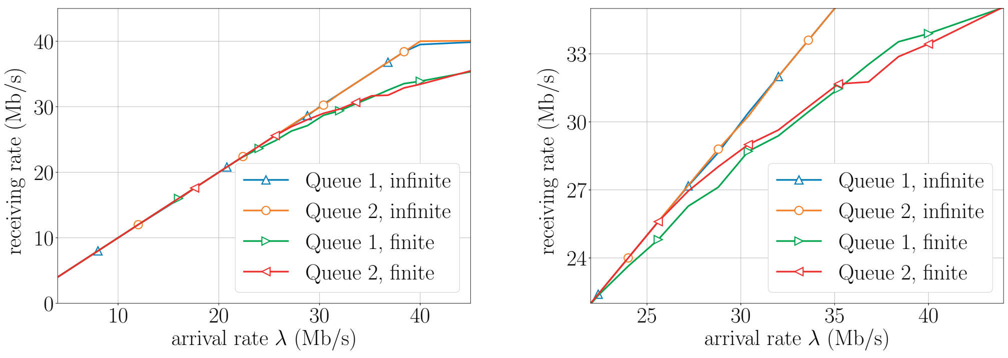


Figure 5.6 – Experimental receiving rate for the two queues on a wireless testbed, with finite and infinite queues. Left: All arrival rates. Right: Zoom on some arrival rates.

rate λ. We note that the delay difference is, as expected, larger when the queues are infinite, and is significant for certain arrival rates: When $\lambda = 22.4$ Mb/s, the delay is 3x larger when using Queue 1, the queue with smallest variance (8.7 ms vs. 2.9 ms).

When the queues are finite, some packets are discarded when the queues are too long. In this case, variability has two consequences: larger delays and lower receiving rate. Delays can be observed in Figure 5.5: For example, when the arrival rate is around 23 Mb/s, Queue 1, the queue with the smallest variance, has a larger average delay (7.7 ms vs. 3.1 ms). For low arrival rates, Queue 1 not only has larger delays, but it also has lower receiving rates. This can be observed in Figure 5.6 (right), that shows a zoom on some arrival rates of Figure 5.6 (left). When the arrival rate is around 25 Mb/s, Queue 1 has a receiving rate of 24 Mb/s, vs. 25 Mb/s for Queue 2. When the arrival rate is around 35 Mb/s, the order gets reversed: Queue 1 has smaller delays and a higher receiving rate.

5.4 Time-Varying Hybrid Networks and Multipath

We now move to the second setting, where the two technologies, potentially with different average service rates, can be used simultaneously (i.e., multipath routing). We first describe the model, then we analyze theoretically the average delay. We present numerical and experimental results that support our analytical results.

5.4.1 Model and Analysis

Model for Multipath

The model for our multipath scenario is illustrated in Figure 5.7. As described in Section 5.2, the service rate of Queue i for $i \in \{1,2\}$ is modelled by low and high states, with respective service rates $\mu_{l,i}$ and $\mu_{h,i}$ ($\mu_{h,i} > \mu_{l,i} > 0$). Packets arrive in the system as a Poisson process with rate parameter λ, and are routed to the queues based on a Bernoulli trial: With probability p, a packet

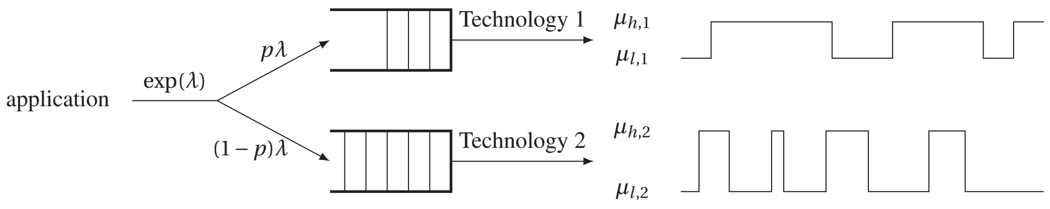


Figure 5.7 – Model with two technologies and multipath.

goes to Queue 1, with probability $1-p$, it goes to Queue 2. As opposed to most works that study queues in parallel [Hai58, Hyy13, IM14], we assume that the routing decision is made without knowledge of the current size of the queues. Several reasons justify this choice. First, it might be impossible to obtain the size of the internal queue of a network interface (e.g., most PLC devices do not give such an information). Second, even when it is possible, it can be quite complex in practice. For example, in OpenWrt (the system that we use in our experiments), the networking stack consists in three different queues [ope]; accessing each of them on a packet-per-packet basis might cause significant overload. Third, to reduce the overhead due to MAC protocols, recent technologies employ frame aggregation. This is for example the case in IEEE 802.11n and 802.11ac, the most recent standards for WiFi, and in IEEE 1901, the most recent standard for PLC. This means that a queue might be non-empty while the channel is idle, which makes the model where the routing decision is based solely on queue size inadequate. Protocols that send feedback for (almost) every packet, such as MPTCP, offer the possibility to use indirect information on the queues, such as the round-trip time. However, this information is delayed, whereas classic models assume immediate information. In addition, feedback cannot be employed for a protocol such as UDP, whereas UDP might be preferred for delay-sensitive applications [KR09]. In Section 5.4.2, we compare experimentally the delays obtained with our model with UDP and those obtained with TCP/MPTCP.

One purpose of our analysis is to study the effect on delays of the parameter p, i.e., of the traffic splitting between the two paths. We want to find the value of p, denoted in the following by p^*, that yields the smallest average delay. When the packets are routed to the queues by using a Bernoulli trial, the two queues are independent, with respective arrival rates $p\lambda$ and $(1-p)\lambda$. With $N_i(\lambda_i)$ being the average size of Queue i as a function of the arrival rate λ_i at Queue i, the average total delay as a function of the splitting probability p is simply, using Little's law,

$$D(p) = \frac{N_t(p)}{\lambda}, \tag{5.3}$$

where $N_t(p)$ is the total average queue length as a function of the splitting probability $p \in [0,1]$ and reads

$$N_t(p) \doteq N_1(p\lambda) + N_2((1-p)\lambda). \tag{5.4}$$

For a given λ, minimizing the average total size of the queues and minimizing the average delay is thus equivalent.

Remember that $\hat{\mu}_i$ is the average service rate of Queue i (in this section, we can have $\hat{\mu}_1 \neq \hat{\mu}_2$). The natural *static* splitting probability (i.e., constant for all arrival rates λ) to use is

$$p_{\text{lim}} = \frac{\hat{\mu}_1}{\hat{\mu}_1 + \hat{\mu}_2}. \tag{5.5}$$

It is easy to show that p_{lim} is the optimal static p, as it is the only static value of p that maintains the two queues stable for all arrival rates $\lambda < \hat{\mu}_1 + \hat{\mu}_2$. If $\lambda \geq \hat{\mu}_1 + \hat{\mu}_2$, no p maintains both queues stable. We now study the optimal splitting probability $p^*(\lambda)$, potentially different for each arrival rate $\lambda \in (0, \hat{\mu}_1 + \hat{\mu}_2)$.

Analysis of the Optimal Splitting Probability

We start by noting that N_i is a strictly convex function of λ_i [LS92]. From Equation (5.3), we know that, for a given λ, working with the average delays D and with the average total queue size N_t is equivalent, and that the delay is minimized when $N_t(p)$ given by Equation (5.4) is minimized. When λ is given, N_t is minimized, either when $p = 0$ or $p = 1$, or when

$$N_t'(p) = \lambda N_1'(p\lambda) - \lambda N_2'((1-p)\lambda) = 0. \tag{5.6}$$

Let us assume first that $N_1'(0) < N_2'(0)$. Because $N_1(\lambda)$ has a vertical asymptote for $\lambda = \hat{\mu}_1$, $N_1'(\lambda) \to \infty$ when $\lambda \to \hat{\mu}_1$, i.e., there is a $\lambda_0 \in (0, \hat{\mu}_1)$ such that $N_1'(\lambda_0) = N_2'(0)$. Because N_1 is strictly convex and consequently N_1' is strictly increasing, λ_0 is unique. Then for all $\lambda \leq \lambda_0$, $N_t'(p) \leq 0$ for all $p \in [0,1]$, i.e., $N_t(p)$ is decreasing and $p^*(\lambda) = 1$. For $\lambda \in (\lambda_0, \hat{\mu}_1 + \hat{\mu}_2)$, Equation (5.6) has a solution in (0,1) because $N_t'(0) = \lambda(N_1'(0) - N_2'(\lambda)) < \lambda(N_1'(0) - N_2'(0)) < 0$ and $N_t'(1) = \lambda(N_1'(\lambda) - N_2'(0)) > 0$, and because $N_t'(p_{\text{lim}})$ is finite. This solution is unique because N_1 and N_2, and thus N_t, are strictly convex. For a given $\lambda \in [\lambda_0, \hat{\mu}_1 + \hat{\mu}_2)$, $p^*(\lambda)$ is then the unique solution of

$$N_1'(p^*\lambda) = N_2'((1-p^*)\lambda). \tag{5.7}$$

The optimal splitting probabiliy $p^*(\lambda)$ is therefore the function equal to 1 for each $\lambda \in (0, \lambda_0]$, and that associates to each $\lambda \in (\lambda_0, \hat{\mu}_1 + \hat{\mu}_2)$ the solution of Equation (5.7). Note that $\lim_{\lambda \to \hat{\mu}_1 + \hat{\mu}_2} p^*(\lambda) = p_{\text{lim}}$.

If $N_2'(0) < N_1'(0)$, everything is similar, except that $p^*(\lambda) = 0$ for $\lambda \in (0, \lambda_0]$. We have thus shown the following theorem.

Theorem 8. *If $N_1'(0) \neq N_2'(0)$, there is a $\lambda_0 \in (0, \hat{\mu}_1 + \hat{\mu}_2)$ such that for all $\lambda \in (0, \lambda_0]$, using a single queue yields smaller average delays than any traffic splitting between the two queues.*

For homogeneous service rates (i.e., when $\mu_{h,i} = \mu_{l,i} = \hat{\mu}_i$), $N_1'(0) = N_2'(0)$ if and only if the queues are identical ($\hat{\mu}_1 = \hat{\mu}_2$). For heterogeneous service rates, we can show with the notations of Section 5.3.1 that $N_i'(0) = \frac{\alpha_i \hat{\mu}_i + \pi_i + V_i}{\hat{\mu}_i(\alpha_i \hat{\mu}_i + \pi_i)}$. Although two non-identical queues can in theory have same value $N'_i(0)$, the set of parameters that meet $N'_1(0) = N'_2(0)$ has measure zero in the space of possible parameters for Queues 1 and 2. For this reason, two non-identical queues with parameters chosen at random are very unlikely to have $N'_1(0) = N'_2(0)$.

We first assume homogeneous service rates ($\mu_{h,i} = \mu_{l,i} = \hat{\mu}_i$) for both queues. Without loss of generality, we assume that Queue 1 is the queue with largest average service rate, i.e., $\hat{\mu}_1 > \hat{\mu}_2$. In the homogeneous case, we can obtain an explicit expression for $p^*(\lambda)$. We know that $N_i(\lambda) = \lambda/(\mu_i - \lambda)$, and a simple computation gives

$$\lambda_0 = \hat{\mu}_1 - \sqrt{\hat{\mu}_1\hat{\mu}_2}. \tag{5.8}$$

Then, using Equation (5.7), p^* is given for $\lambda \in (0, \hat{\mu}_1 + \hat{\mu}_2)$ by

$$p^*(\lambda) = 1 \textbf{ if } \lambda \in (0, \lambda_0],$$

$$p^*(\lambda) = \frac{\hat{\mu}_1(\lambda - 2\hat{\mu}_2) + (\hat{\mu}_1 + \hat{\mu}_2 - \lambda)\sqrt{\hat{\mu}_1\hat{\mu}_2}}{\lambda(\hat{\mu}_1 - \hat{\mu}_2)} \textbf{ if } \lambda \in (\lambda_0, \hat{\mu}_1 + \hat{\mu}_2).$$

When the service rates are heterogeneous ($\mu_{h,i} \neq \mu_{l,i}$), the expression for N_i is too complex to provide an explicit expression of p^*. However, we have proven (Theorem 8) that there exists a $\lambda_0 > 0$ such that when $\lambda \leq \lambda_0$, $p^*(\lambda) = 0$ or $p^*(\lambda) = 1$: For low arrival rates, it is better in terms of delays to use only one path. In addition, p^* can be computed numerically using Equation (5.7).

5.4.2 Numerical and Experimental Results

Numerical Results

In this section, we show numerically that in time-varying networks, the choice of p has a strong impact on the delays. In particular, we compare the delays obtained with $p^*(\lambda)$ and with p_{lim}. Queue 1 has an average service rate $\hat{\mu}_1 = 30$ Mb/s, and Queue 2 an average service rate $\hat{\mu}_2 = 5$ Mb/s. Packets have size 1 400 B, so that $\hat{\mu}_1 = 2\,678$ packets/s and $\hat{\mu}_2 = 446$ packets/s. Queue 1 has homogeneous service rates ($\mu_{h,1} = \mu_{l,1}$), and we study how the variability of Queue 2 affects delays. We set $\alpha_{h,2} = \alpha_{l,2} = 1$ transition/s, and we use three different sets of values for $\mu_{h,2}$ and $\mu_{l,2}$ with same average service rate $\hat{\mu}_2$: $\mu_{h,2} = \mu_{l,2} = 5$ Mb/s; $\mu_{h,2} = 6$ Mb/s and $\mu_{l,2} = 4$ Mb/s; $\mu_{h,2} = 7$ Mb/s and $\mu_{l,2} = 3$ Mb/s.

Figure 5.8 shows the optimal splitting p^* for the three sets of values for $\mu_{h,2}$ and $\mu_{l,2}$. When $\mu_{h,2} = \mu_{l,2} = 5$ Mb/s, both queues are homogeneous, and, as shown above, it is optimal to send all the traffic on Queue 1 when $\lambda < \lambda_0$ with $\lambda_0 \approx 17.8$ Mb/s given by Equation (5.8). We see on Figure 5.8 that the effect of the variability ($\mu_{h,2} \neq \mu_{l,2}$) on λ_0 is quite small. However, the impact of the variability on the delays is very large. We show the ratio (Figure 5.9, left) and difference (Figure 5.9, right) of the packet delays when the traffic is split either with probability $p^*(\lambda)$ for an arrival rate λ, or with probability $p_{\text{lim}} \approx 0.86$ for all arrival rates. For example, when $\mu_{h,2} = 7$ Mb/s and $\mu_{l,2} = 3$ Mb/s and for an arrival rate $\lambda = 19$ Mb/s, $p^* = 1$, and the delay

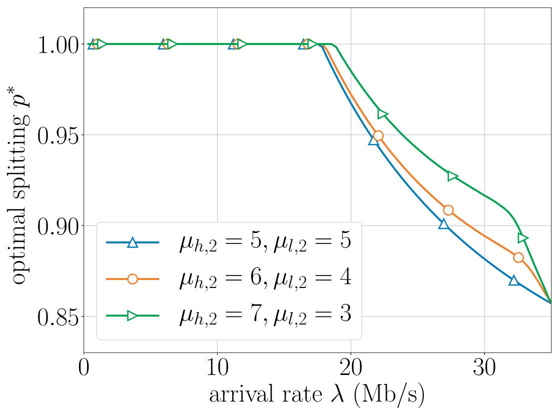


Figure 5.8 – Optimal splitting p^* as a function of the arrival rate λ.

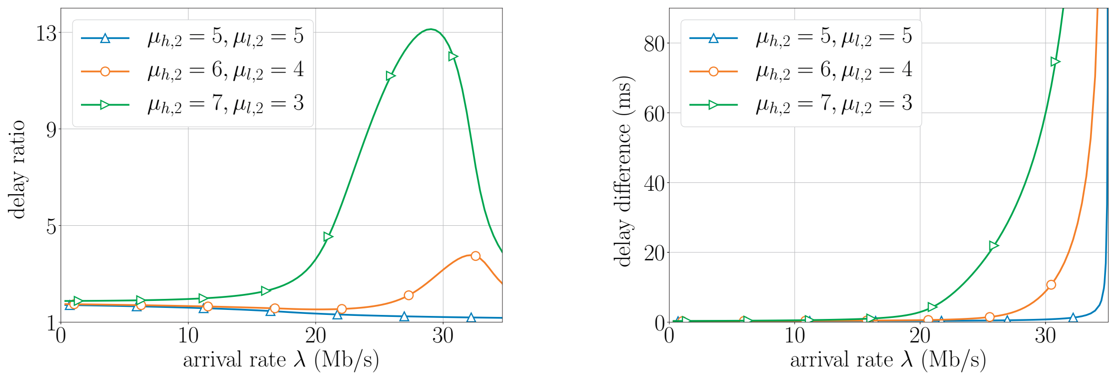


Figure 5.9 – Ratio $D(p_{\text{lim}})/D(p^*)$ (left) and difference $D(p_{\text{lim}}) - D(p^*)$ (right) of the delays with static (p_{lim}) and optimal (p^*) splitting.

obtained by sending everything on Queue 1 is about 1 ms; when traffic is split with probability p_{lim}, the delay is 2.8 ms. For higher arrival rates, the delay ratio can be up to 13: For $\lambda = 29$ Mb/s, the delay with p^* is 3.9 ms, and the delay obtained with p_{lim} is 51 ms. We next present evidence of this behavior on a wireless testbed.

In our analysis, we do not take reordering into account. A careful study of reordering is out of the scope of this dissertation, but reordering is likely to further increase the rate λ_0 before which using a single queue is preferable. Indeed, sending at a low rate on the second queue might harm delays (because an additional delay is required for packets to be ordered) more than the gains it offers.

Experimental Results

We now study experimentally the impact of the splitting probability p. Our hybrid network consists in Queue 1, a WiFi interface in the 5 GHz band, and Queue 2, a WiFi interface in the 2.4 GHz band. In this section, to avoid external sources of variability, especially in the 2.4 GHz band used by the WiFi network of the university, we run all the experiments at night. We consider a simple and realistic case where the two queues have different average service rates. Queue 1 has homogeneous service rates: $\mu_{h,1} = \mu_{l,1} = 5000$ packets/s (MCS 7, about 56 Mb/s). Queue 2 has variable service rates: $\mu_{h,2} = 2053$ (MCS 3, about 23 Mb/s) and $\mu_{l,2} = 1071$ (MCS 1, about 12 Mb/s). The transition rates for Queue 2 are $\alpha_{h,2} = \alpha_{l,2} = 3$ transitions/s. We send UDP traffic during 20 seconds for various arrival rates λ and for all probabilities p between 0 and 1, with 0.05 increments. The experiments are repeated three times for each λ and p. Figure 5.10 (left) shows the theoretical and experimental values for p^*. For the infinite queues, the experimental p^* is found for each arrival rate as the p that yields the smallest average delays. For the finite queues, there might be packet losses as shown in Section 5.3.2, and we measure both the splitting probability p that yields the smallest average delays and the splitting probability p that yields the maximal received throughput (we note that the two are equal, except when the arrival rate is very close to the saturation rate). The experimental values are close to the theoretical ones. The optimal static p, as defined by Equation (5.5), is $p_{\text{lim}} \approx 0.75$. Figure 5.10 (right) shows the ratio of the delays obtained with p_{lim} over the delays obtained with $p^*(\lambda)$. For certain arrival rates, the negative impact of a static splitting probability is quite strong: With infinite queues, for $\lambda = 64$ Mb/s, the delay is 6 ms with $p = p^*$, whereas it is 25 ms with $p = p_{\text{lim}}$. The impact is strong with finite queues as well: For $\lambda = 48$ Mb/s, the delay is 2 ms with $p = p^*$, whereas it is 9 ms with $p = p_{\text{lim}}$.

In contrast, when we set $\mu_{h,2} = \mu_{l,2} = 1562$ (MCS 2, about 17.5 Mb/s), i.e., when the two queues have homogeneous service rates, using p_{lim} instead of p^* has a small effect on delays: The maximum delay-ratio is around 1.5x for analytical and experimental results, and the difference is always less than a millisecond (the results are not shown due to lack of space).

Finally, we compare the results obtained with our model with the results obtained with single-path TCP and MPTCP. Because TCP and MPTCP are by default designed for favoring throughput, we do some modifications in order to favor delays, which reduces the delay at low throughput from about 2 ms to about 0.7 ms. This is the same value as with UDP traffic. We keep using $\mu_{h,1} = \mu_{l,1} = 5000$, $\mu_{h,2} = 2053$, $\mu_{l,2} = 1071$, and $\alpha_{h,2} = \alpha_{l,2} = 3$. TCP is used on Queue 1 only, the queue with higher average service rate (about 50 Mb/s with TCP). For MPTCP, the default scheduler is used. For each rate, the experiment is repeated five times and we present averaged results. Figure 5.11 (left) shows the probability that traffic with MPTCP is sent on Queue 1, along with p^* found experimentally with UDP traffic. For low arrival rates, MPTCP sends more traffic on the second queue when compared to p^*, the optimal splitting in our model. Figure 5.11 (right) shows the delays obtained with the different protocols. UDP is shown for reference, but a fair comparison with TCP or MPTCP is difficult, because the performance of TCP/MPTCP in terms of delays is very dependent on the configuration (scheduler, congestion window, slow start, etc.). Nevertheless, results indicate that MPTCP schedulers can be improved to favor latency. It is interesting to compare TCP and MPTCP, because they use the same configurations. For low arrival rates, TCP (i.e., single-path) yields slightly smaller delays, whereas MPTCP reduces the delays for higher arrival rates. This confirms our analysis and previous experiments: For low arrival rates, it is better in terms of delays to use a single path.

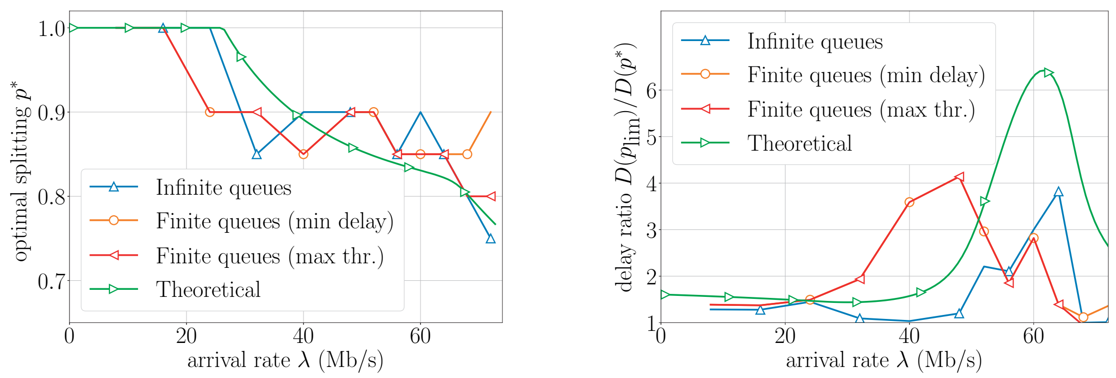


Figure 5.10 – Left: optimal splitting probability $p^*(\lambda)$. Right: ratio $D(p_{\text{lim}})/D(p^*)$ of the delays obtained with the static probability p_{lim} and with the optimal splitting p^*.

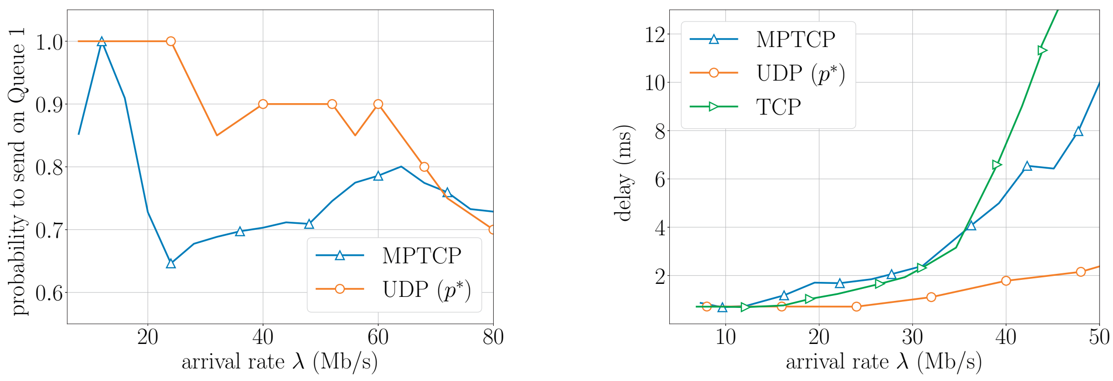


Figure 5.11 – Left: probability to send on Queue 1. Right: delays obtained with TCP, MPTCP, and UDP (p^*).

5.5 Related Work

Queueing Models

Time-varying queueing models were first studied in 1956 by Clarke [Cla56]. The model described in Section 5.2 was introduced and solved for the first time by Yechiali and Naor [YN71]. It was then studied with more general assumptions [HZ04] or solved with different techniques [Neu77, MG05]. These works give powerful tools to derive numerical results, but only a few give intuitive insights and fundamental properties of the delays generated by this queue model. The works that do so study the effect on delays of the transition rates α [GHBWY06, Rol81, Ros78, Hey82]. Other works have studied M/G/1 or MAP/G/1 queues with correlated service times [AK03, MPB77, LVHB06]; they find recursive equations for the delays and solve them numerically, but they do not give intuitive insights and fundamental properties of the average delays. To the best of our knowledge, our work is the first that shows that with the same average service rate, a largest variance can yield lower average delays.

A model with two queues in parallel was introduced in 1958 by Haight [Hai58]. A large number of works study models that assume identical servers with homogeneous service rates ($\mu_h = \mu_l$), when the packets are routed based on the current queue sizes [ARW11, Win77]. These models were extended to support non-identical servers, still with homogeneous service rates [Lar81, Hyy13]. They show that the optimal decision is threshold-based, i.e., packets are routed to the fastest queue, unless the queue-size difference is above some threshold. To the best of our knowledge, ours is the first work that studies queues in parallel with heterogeneous time-varying services rates. Here, we assume that packets are routed based on a Bernoulli trial. Only a few models study Bernoulli routing, either with identical servers and homogeneous service rates [CCP90], or by only looking at the case where the parameter p is the same for all arrival rates [Hyy13].

Delays in Hybrid Networks

With the recent development of hybrid networks, much attention has been recently given on knowing if multipath, in particular MPTCP, could help reduce delays. When the characteristics of the two paths are very different, in particular in terms of RTT (e.g., with LTE and WiFi), MPTCP is found to increase slightly the delays [CLG$^+$13, CT14]. When the characteristics of the two paths are close, MPTCP can reduce delays significantly [YFD$^+$16]. We find in addition that it depends on the arrival rates: For low arrival rates, it is usually better to use a single path, whereas it is better to use two paths for larger arrival rates. Finally, MPTCP can also be used by sending redundant messages on the two paths [FEB$^+$16]. This reduces the delays, but at the cost of higher utilization. This model with redundant messages is out of the scope of our dissertation.

5.6 Summary

In this chapter, we have investigated delays in time-varying hybrid networks. The variability of the service rate severely impacts packet delays, as illustrated by the experiments we carried out on a wireless testbed. First, we have studied the setting where the user wants to choose between two technologies with the same average service rate but different service rate variabilities. Using a queue model with heterogeneous time-varying service rates, we have shown that for a given average service rate, the technology that offers the smallest delays is not necessarily the same for all arrival rates, and that the technology with the largest variance sometimes yields the smallest delays. These results obtained with a time-varying queuing model are supported by experiments carried out on a wireless testbed with two WiFi channels; they have shown that the delay gain provided by using the technology with the largest variance can be significant. Then, we have studied the conditions under which multipath (for example, with two different technologies, e.g., WiFi and LTE or PLC, or two WiFi channels), reduces the delays, compared to using a single path. We have shown through analysis and testbed experiments that the optimal splitting between the two paths is not easy to find as it depends on the arrival rate, and that for low arrival rates, using a single path is usually better than using two paths. Numerically and experimentally, we have shown that the larger the variability, the higher the impact of the splitting decision on the delays.

Privacy Part III

6 Improving Privacy with Multipath and Hybrid Networks

6.1 Introduction

In the first and second parts of this dissertation, we have shown that hybrid networks and multipath can significantly improve the throughput and latency experienced by users. The recent revelations about mass surveillance have also shed light on the privacy risks that go with the use of Internet. In this chapter,[1] we hence study the effect of multipath and hybrid networks in terms of privacy.

6.1.1 Website Fingerprinting

The web activity of users, i.e., which websites they visit, is known to be a sensitive data as it discloses a large amount of information. Such information can be used by repressive states against citizens who try to go against country-level censorship and by marketers (e.g., it has been revealed that in the USA, Internet service providers (ISPs) sell the Internet browsing records to marketers [ISP07]). Consequently, being able to keep a person's web activity private is of the utmost importance. Typical solutions include the use of encryption protocols that, such as TLS (used by HTTPS), hide only the content of the packet; and of anonymous communication tools that, such as Tor, also hide the destination of the packets from a local adversary (e.g., a curious or state-controlled ISP, or the curious administrator of a public WiFi access-point). Anonymous communication tools like Tor, which we study more particularly in this dissertation, are supposed to make it impossible for a local adversary to detect which website a client visits.

Recent works [CA98, HWF09, PNZE11, WCN$^+$14, PLP$^+$16, HD16, WG16], however, have shown that traffic analysis techniques, such as *website fingerprinting* (denoted in this chapter by WF), enable a local adversary to identify with high accuracy which website is visited (around 90% accuracy in a list of 100 monitored websites), even if the client uses an anonymous communication tool (i.e., she hides the final destination and content of the packets from the local adversary). The adversary can successfully identify the website (carry out a WF attack) by looking at only

[1]This chapter is based on the paper by Henri et al. [HGAS$^+$18].

packet metadata such as timestamp, size and direction (incoming or outgoing) of the packets. WF attacks typically extract features on the observable metadata (such as total number of packets, timings of packets, ratio between outgoing packets and incoming packets). These attacks are formulated as a classification problem and rely on supervised machine-learning techniques to learn associations between features and websites.

A large number of defenses has been proposed to defeat these attacks. With Tor, the packets are fragmented in so-called *Tor cells* of fixed size, which means that only the timestamp and direction of the packets are available to the adversary; but it has been shown to be ineffective [PNZE11, WCN$^+$14, PLP$^+$16, HD16, WG16]. More complex defenses rely on two main techniques: (i) *link padding*, which takes advantage of the inability of the adversary to see the content of the packets and inserts dummy packets to the packet flow to confuse the adversary; and (ii) *packet delaying*, which delays packets to modify the trace. Both of these techniques incur a performance overhead: Link padding causes a traffic overhead, because more packets are sent; and packet delaying causes a loading-time overhead, because delaying packets makes the website loading time larger. This yields a tradeoff between privacy and performance.

6.1.2 Multipath and Hybrid Networks

We consider the scenario where a client is multihomed, i.e., (i) she is connected to the Internet through multiple networks, and (ii) an adversary can only observe the traffic sent through one of these networks — which, as we argue, is very likely in several use cases. Multihoming has been studied for quite some years as a solution for improving reliability and performance. It has long been used by enterprises [AMS$^+$03], but rarely by individual clients due to the lack of natively-multihomed devices and because it relied on multipath solutions, such as SCTP [Ste07, IAS06], which are difficult to use in today's Internet. Multihoming has recently gained popularity for two reasons. First, multihomed devices have become omnipresent in the last few years. This is in particular due to the emergence of hybrid networks. Today, virtually all smartphones are natively multihomed and support both WiFi and cellular. All laptops have a WiFi interface and can easily be made multihomed by the addition of a lightweight and low-cost component (e.g., 3G/4G USB dongles) or with a smartphone sharing its connection through Bluetooth or USB. Dual-SIM cellphones have also been launched onto the market and are gaining popularity. Second, multipath solutions compatible with the protocols dominant in today's Internet have been recently developed to exploit hybrid networks. The most popular solution is multipath TCP (MPTCP) [FRH$^+$11, FRHB13]: it has been shown to bring significant performance gains [RPB$^+$12, CLG$^+$13, LAPJ15, FEB$^+$16].

The availability of multiple networks makes it possible to choose, for each packet, through which one of these networks it should be sent, which is referred hereafter as *splitting*. Splitting traffic among the networks makes it possible to *remove* packets from one network by sending them through another — whereas current defenses can only add and/or delay packets, which creates performance (traffic and loading-time) overhead, as explained above. This enables us to design a defense with no traffic overhead. Inserting dummy packets and/or delaying packets remains of course possible in multihoming and can be combined with splitting to further improve privacy, as we also show.

To provide security and privacy guarantees, exploiting the existence of several non-colluding entities is a standard and successful technique used in other contexts (e.g., secret sharing [Sha79] or multi-cloud storage [SS13]). However, splitting traffic through multihoming is not straightforward as it raises several challenges. The first challenge that needs to be addressed is to know how to split the traffic between the networks to improve the resistance against WF attacks. We show that this is non-trivial, as deterministic schemes (e.g., round-robin) do not significantly improve privacy. This makes necessary the development of a specific multipath scheduler for increasing the resistance against WF. Our first contribution is the design and extensive evaluation against state-of-the-art WF attacks of HyWF, a novel multipath scheduler for protecting against WF. We show that HyWF achieves the same level of accuracy as the best existing practical defenses, but does so without adding any performance overhead. Note that HyWF is compatible with other defenses that rely on link padding or packet delaying, and combining HyWF with another defense further improves privacy by combining the gains brought by the two defenses. We demonstrate this with the description and evaluation of HyWF-AP, an extension of HyWF with adaptive padding (a state-of-the-art defense that relies on link padding) and our second contribution.

The second challenge comes from the fact that, to achieve significant privacy gains, the defense needs to be used for both incoming and outgoing traffic. If only the client implements the scheduler, privacy is expected to just slightly increase, and we demonstrate this through experiments. But it would be challenging, if not impossible, to make every server use the defense. As an alternative, we consider that a Tor proxy is available. This approach has also been employed by previous work [JIP$^+$16, WG17]. Our third contribution is a proof-of-concept implementation of HyWF for the client and the proxy. Our implementation does not require modifying Tor or the application. It enables us to evaluate the performance of our splitting scheme, and we show that HyWF does not add significant loading-time overhead.

Outline of the Chapter This chapter is structured as follows. In Section 6.2, we discuss related work. We present our system and adversarial model in Section 6.3 and describe our methodology in Section 6.4. In Section 6.5, we introduce HyWF, a defense with no traffic overhead. We evaluate it through extensive simulations in Section 6.6. In Section 6.7, we describe the implementation of HyWF and show that it does not add significant loading-time overhead. In Section 6.8, we introduce and evaluate HyWF-AP, an extension of HyWF with adaptive padding. We conclude in Section 6.9.

6.2 Related Work

We first discuss traffic analysis attacks, in particular, WF. We also present other contexts in which splitting the data between non-colluding entities is used to improve privacy.

6.2.1 Traffic Analysis

Traffic analysis has been extensively studied in the last 20 years. In 1996, Wagner and Schneier were the first to show that valuable information could be extracted from data encrypted with SSL [WS96]. In 1998, Cheng and Avnur studied WF more specifically and were able to show that the website accessed by clients could be successfully identified when observing encrypted data [CA98]. In 2009, Herrmann et al. showed that WF attacks could be successfully applied to anonymous communication tools [HWF09]. They achieved an accuracy of 20% for the JAP network, but the accuracy against Tor remained quite low (around 3%). Since then, many WF attacks and defenses have been proposed for anonymous communication tools. Several attacks have been shown to be efficient for Tor as well, as detailed below.

Existing Attacks

WF Attacks against Tor After the first work by Herrmann et al., many other attacks were proposed. The basic idea is the same for all of them. The attacks are formulated as a classification problem: They first extract meaningful features from the observable metadata; then, to associate these features to the websites, they train a supervised machine-learning model. The first attack against Tor proved to be successful was proposed by Panchenko et al. [PNZE11] and relied on support vector machines (SVM). Since then, many attacks that employ different features and machine-learning models have been proposed [SM09, CZJJ12, WG13, WCN+14, PLP+16, HD16, FL16]. In this chapter, we evaluate our defenses against the most relevant attacks:

- k-NN [WCN+14]: Wang et al. use a k-nearest neighbors model with $N = 1\,225$ features (total transmission time, number of packets, ordering and concentration of outgoing packets, burst length, etc.).
- CUMUL [PLP+16]: Panchenko et al. propose an attack that uses an SVM model with, as features, the cumulative sum of the packet sizes (negative for outgoing packets, positive for incoming packets). With Tor, the absolute value of the packet sizes is constant.
- k-fingerprinting [HD16]: Hayes et al. propose an attack that extracts $N = 175$ features out of the traces (e.g., total number of packets, number of packets per second, concentration of incoming/outgoing packets, etc.) and uses a random forest classifier.

These attacks are very general and are not bound to any specific defenses. Recent works [CHJ17, OJA+17] consider them to be the three most advanced and effective WF attacks. We also evaluate an attack that combines the three attacks:

- Ensemble [OJA+17]: Overdorf et al. propose to combine k-NN, CUMUL and k-fingerprinting by using the prediction score computed by the attacks for each website in the set of monitored websites. Each individual attack predicts the best website (i.e., the website with highest prediction score); Ensemble predicts the website that has, among the three attacks, the highest difference between the best website and the second best website.

We build on the code provided by the authors.[2] Hereafter, unless specified otherwise, we present only the results obtained with the most dangerous attack (i.e., the one reaching the highest accuracy) among k-NN, CUMUL, k-fingerprinting and Ensemble.

Juarez et al. [JAA+14] and Wang and Goldberg [WG16] study the practicality of WF attacks. Juarez et al. critically evaluate the typical assumptions made on the adversary (assumptions that we also make, as explained in Section 6.3), and show a rapid decrease in the accuracy of the attacks when the data gets older, or when multiple websites are accessed at the same time. Wang and Goldberg show that by carefully building the datasets on which the attacks train their model, it is possible to achieve a practical WF attack against Tor.

Other Traffic Analysis Attacks Here, we focus on WF attacks against the Tor network, in which only the timestamp and direction of packets, and not their size, are available to the adversary. Other traffic analysis attacks that use also the packet sizes are studied. Danezis [Dan10] and Miller et al. [MHJT14] study attacks against HTTPS traffic. With HTTPS, the adversary has information about the website that is accessed (the IP address of the server is sent in clear), and the authors show that it can with very good accuracy infer which webpage of the website the client visits. Traffic-analysis attacks also enable an adversary to compromise the privacy of encrypted voice-over-IP calls [ZF11].

Existing Defenses

In parallel, researchers study mechanisms to protect against these WF attacks. The large majority relies on link padding (i.e., inserting dummy packets[3] to confuse the adversary that is unable to distinguish them from real packets), and/or on packet delaying. These defenses incur traffic and/or loading-time overhead.

The most secure defense consists in ensuring that all traces look exactly the same; this is the basis for constant-rate padding, such as BuFLO proposed by Dyer et al. [DCRS12]. With BuFLO, real packets are delayed and dummy packets are inserted so that the inter-arrival times (time interval between two consecutive packets) stay constant. However, this might still leak some information, because the total loading time might be different for different websites (even with constant-rate padding, short traces will end before long traces). This enables an adversary to use this information to infer which website is accessed. For this reason, Cai et al. propose

[2]The entire code used for the attacks and defenses presented in this chapter is available at https://www.dropbox.com/s/716j4pvxuyjiu5j/code_final.zip.

[3]Note that breaking packets in Tor cells also requires some padding so that all cells have the same size. Because this padding is done with Tor with or without another specific defense against WF attacks, we only consider the insertion of dummy packets to compute the traffic overhead.

CS-BuFLO [CNJ14] and Tamaraw [CNW$^+$14] to improve the performance and to reduce the overhead of BuFLO. With these improved techniques that rely on constant-rate padding, the traces for different websites look the same, and the adversary is virtually unable to distinguish between them. However, sending packets at a constant rate requires both inserting dummy packets and delaying real packets, which causes very high traffic and loading-time overhead. Tamaraw offers the best performance, but the authors still report a loading-time overhead of 320%. The loading-time overhead is particularly harmful, as it directly affects the quality of experience for the client, and we would like to find defenses that do not add significant loading-time overhead. For this reason, several other defenses are proposed, as detailed below.

One defense is proposed that does not rely on link padding or on packet delaying (i.e., that does not insert dummy packets nor delay real packets): randomized pipelining (RP) [Per]. It works at the application layer. It enables HTTP pipelining (i.e., sending multiple HTTP requests in parallel) and it randomizes the number and the order of parallel HTTP requests. However, this technique is shown to be inefficient in practice by Cai et al. [CZJJ12] and Wang and Goldberg [WG13]. Cherubin et al. propose LLaMA [CHJ17], a defense that improves RP by combining link padding and additional delay for some packets, but this incurs traffic and loading-time overhead. Another defense, which loads a decoy page each time the client wants to access a website, is proposed [PNZE11], but Hayes et al. [HD16] show that this defense performs worse than other defenses such as adaptive padding (described below), with smaller privacy and larger overhead. Other defenses are proposed at the application layer. Cherubin et al. introduce ALPaCA [CHJ17], a server-side defense that pads the content of a webpage to alter its characteristics (in particular its size) without modifying how it looks to the client. The authors show that ALPaCA significantly improves privacy (the accuracy of the attack is reduced to about 15%), but it incurs a traffic overhead of about 90% and a loading-time overhead of more than 50%. In addition, because this defense needs to be implemented at the server side, it might be difficult to deploy in practice.

Other defenses are proposed: they do not try to make all traces look exactly the same (by sending packets at constant rate), rather aim at making the traces be statistically similar. Wright et al. propose traffic morphing [WCM09], a defense that relies on padding and ensures that the packet-size distribution is similar for all traces. This method is not effective with Tor, which already sends packets with a fixed size, and it is shown to be inefficient in practice even when the packet sizes are different [DCRS12, HD16]. *Adaptive padding* (denoted in this dissertation by AP) [SW06] employs a similar idea, but it works on the inter-arrival times (i.e., on the time intervals between two consecutive packets) rather than on the packet sizes: As opposed to constant-rate padding, AP does not send packets at a constant rate; instead, it tries to make all traces statistically similar and undistinguishable from each other by ensuring that the distribution of the inter-arrival times is the same for all traces. The packets are never delayed, i.e., there is no loading-time overhead. Juarez et al. propose WTF-PAD [JIP$^+$16], an implementation of AP for WF attacks. AP is the defense that is found to offer the best tradeoff between privacy and performance [HD16]. Importantly, AP is the defense currently under consideration for addition to the Tor project [Per15]. This is why we use AP as a benchmark in this chapter. However, traffic overhead remains significant, and implementing this scheme requires knowing in advance the statistics on the website traces, which might prove challenging in practice.

Note that the above-mentioned defenses can be implemented along with HyWF, the defense that we propose and that does not add overhead. In Section 6.8, we present and evaluate HyWF-AP, the extension of HyWF with AP, and show that it significantly improves privacy — compared to AP alone.

6.2.2 Privacy-Protecting Data-Splitting Schemes

Splitting data between several non-colluding entities has been proved to be a successful technique. Shamir [Sha79] and Blakley [Bla79] first invented secret sharing in 1979, as a way to protect information by sharing it between several participants. Since then, data-splitting schemes have been used in many other contexts that include cloud oblivious storage [SS13], privacy-protecting cloud computing [JLWW13], secure data deduplication [LCL+14], vehicular ad-hoc networks [RH07], and secure sharing of personal health records [LYZ+13], to name a few.

6.3 System and Adversarial Model

In this section, we describe our system and adversarial model.

6.3.1 System Model

We consider a Tor client who is multihomed, i.e., she has access to multiple networks. In this chapter, we consider that the client uses two networks, which is by far the most frequent case in practice; our defense can easily be extended to more networks. Some real-world examples of a multihomed client are presented in Figure 6.1: A client with a smartphone connected to a WiFi access-point and to a cellular network (left); a client with a laptop connected to its home access-point with Ethernet or WiFi and to a cellular network through a smartphone sharing its connection with Bluetooth or USB (center left); a client with a dual-SIM smartphone connected to two cellular networks with different ISPs (center right); and a client with a laptop connected to two WiFi access-points (home and public) in two WiFi bands (right). The client uses multipath, i.e., she is able to decide on a packet-per-packet basis which network she uses, through a dedicated scheduler implemented in the client's phone or laptop.

6.3.2 Adversarial Model

We consider a scenario where the client wants to protect against curious adversaries snooping on her traffic in order to know which website she accesses. The curious adversaries are local (between the client and the first Tor node). They can be an ISP, an entity that controls the ISP

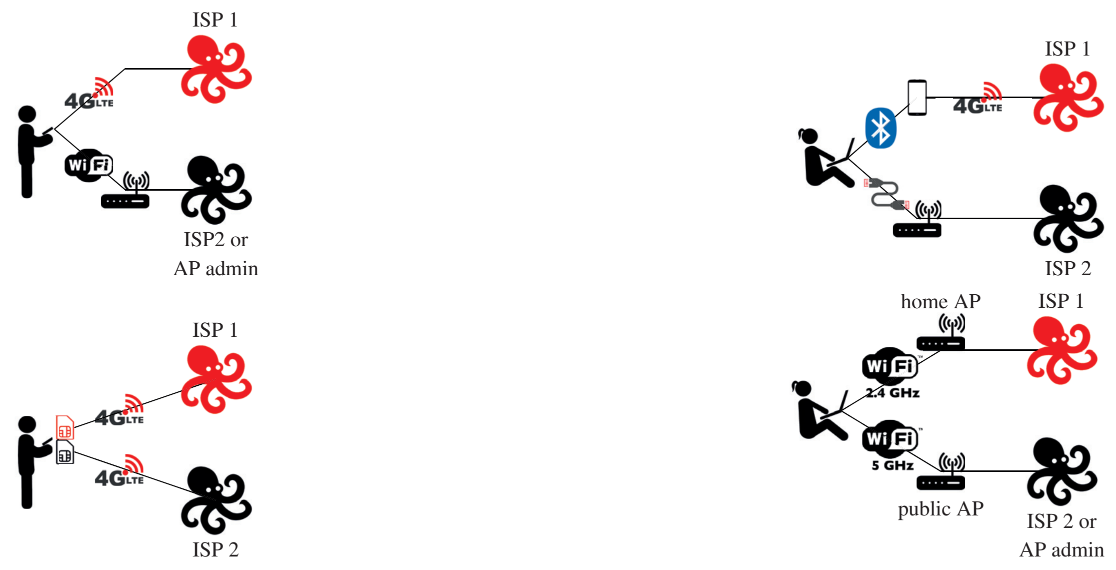


Figure 6.1 – Different real-world examples of a multihomed client. The adversary can be an ISP (*ISP1* and *ISP2* in the figure) or the administrator of a public WiFi access point (*AP admin* in the figure). Top left: client with a smartphone connected to a WiFi access-point (home or public) and to a cellular network. Top right: client with a laptop connected to its home access-point with Ethernet and to a cellular network through a smartphone sharing its connection with Bluetooth. Bottom left: client with a dual-SIM smartphone connected to two cellular networks with different ISPs. Bottom right: client with a laptop connected to two WiFi access-points (home and public) in two WiFi bands (2.4 GHz and 5 GHz).

(e.g., a state), or the curious administrator of a public WiFi access-point. The adversaries are passive, i.e., they do not modify the transmissions.[4] Because the client uses Tor, the adversaries observe only the timestamp and direction of the packets. The observable metadata for one website access is called a *trace*.

The key assumption that we make is that the adversaries are not able to correlate the traffic sent through one network and the traffic sent at the same time through the other network, i.e., they cannot reconstruct the original trace. In particular, a single adversary cannot snoop on the two networks at the same time. This happens when the client uses two technologies and the adversaries can snoop on a single technology (e.g., when the adversary is the curious administrator of a WiFi access-point). This also happens when the adversaries are different ISPs, which is an important use case: For example, it has been revealed that in the USA, ISPs sell the Internet browsing records to marketers [ISP07], which is cited as one reason for using the Tor network [Tor15]. Browsing records can typically be inferred by WF attacks. To be protected against such attacks, the client can use two different ISPs; for example, she can use hybrid networks, with two

[4]The defense that we present relies on multipath solutions, such as MPTCP. MPTCP packets have a special TCP field set, and a powerful adversary could drop all MPTCP packets to prevent our defense from working. Doing so would block all MPTCP traffic (sensitive or not) and the adversary would become active, which is out of the scope of this dissertation.

technologies (wired or wireless) that have different ISPs. She can also connect to two ISPs with the same technology. These cases are illustrated in Figure 6.1.

We consequently assume that there is one adversary per network. In most cases, the two adversaries do not collude and are consequently unable to reconstruct the original trace. Even if an external entity (e.g., a state or a state-controlled entity) can make the two adversaries (e.g., two ISPs) collude, it would also prove difficult and challenging to reconstruct the original trace from the two different traces, for the following reasons. First, when a device is multihomed, it has two different identifiers for the two ISPs (different IP and MAC addresses), and it can be non-trivial to associate the two identifiers with a same device.[5] Second, the infrastructures for the two different ISPs are necessarily different, which means that the paths taken by the packets are distinct. This means that even if the colluding ISPs are able to associate the two identifiers of a device, the reconstruction of the original trace cannot be achieved synchronously but is necessarily asynchronous. Consequently, the traces for each ISP would need to be stored to be reconstructed later, i.e., the two ISPs would need to store all the per-packet metadata sent through their network for all users they want to attack, until they can reconstruct the complete traces. This would incur significant practical difficulties and storage overhead. Third, the two ISPs should be tightly synchronized in order to reconstruct the trace, which would also be challenging in practice. In contrast, when the client uses a single ISP, the attack can be performed on-the-fly along the single path, hence there is no need for storage and synchronization. In repressive countries where the state is known to spy on the users and can force ISPs to collude, spying on a client who uses two different ISPs would consequently be much more difficult.

In a direct consequence of our hypothesis, an adversary can only use the partial trace sent through a single network to infer the website. In the following, we consequently consider a single adversary that tries to infer the website that a client visits by observing the traffic sent through a single network.

We also make the following assumptions that are extensively made in the literature [SM09, CZJJ12, HWF09, PNZE11, WG13, WCN$^+$14] and that all favor the adversary:

Page Load Parsing The adversary is able to detect the beginning and the end of different website accesses. We discuss further this question in Section 6.7.

No Background Traffic The adversary is able to distinguish one trace for a specific website access from other packets sent by other applications or for the access of another website. This can prove challenging in practice, because multiple applications might be used simultaneously.

Replicability The adversary is able to train its machine-learning model under the same conditions as the client. In practice, the trace that an adversary wants to classify is not obtained with the same method as the traces used for training the machine-learning model (e.g., they do not use

[5]It is trivial if the client is registered with the two ISPs under the same name, but this is not always the case, e.g., if she uses a friend's Internet access.

the same device, the device is not at the same distance of the WiFi access-point and the cellular base-station, etc.).

Failing to verify one of these three assumptions has been shown to have negative effect on the attack [JAA+14]. This means that the reported accuracy of the attacks against the defenses that we propose, HyWF and HyWF-AP, is a conservative value and that in a practical scenario, it is very likely that our defenses would achieve better privacy.

6.4 Data and Methodology

In this section, we describe the datasets and methodology used for the experiments of this chapter.

6.4.1 Datasets

`Wang` The primary dataset are traces gathered in 2014 by Wang et al. [WCN+14]. This dataset, denoted by `Wang`, has been extensively used in the literature [WCN+14, PLP+16, HD16, WG16, AG16]. It contains 90 different traces for each of 100 *monitored* websites (the total number of monitored traces is $n_{\text{mon}} = 9\,000$), and one trace for each of $n_{\text{unmon}} = 9\,000$ *unmonitored* websites. The monitored websites come from a list of websites blocked in China, the United Kingdom, and Saudi Arabia. The unmonitored websites are drawn from Alexa's top 10 000 list. There is no intersection between the monitored and unmonitored websites. 6 000 of the monitored and 6 000 of the unmonitored websites are randomly chosen as the training set, i.e., the set of traces used by the adversary to train the machine-learning model; the remaining 3 000 monitored and 3 000 unmonitored are used as our testing set, i.e., the set of traces used to test the accuracy of the attack.

`Hayes` To verify that our conclusions can be generalized to different datasets, we also use the dataset gathered in 2016 by Hayes et al. [HD16]. This dataset is denoted by `Hayes`. It contains 100 different traces for 85 monitored websites (i.e., $n_{\text{mon}} = 8\,500$), and one trace for 100 207 unmonitored websites. 55 of the monitored websites are the 55 top Alexa websites and the remaining 30 monitored websites are 30 popular Tor hidden services. The unmonitored websites are drawn from Alexa's top list, excluding the top 55. There is no intersection between the monitored and unmonitored websites. The training set always contains two-thirds of the monitored traces, i.e., 5 667 monitored traces, and the testing set contains the remaining third of the monitored traces. For the unmonitored traces, the Hayes dataset is used in two scenarios: with $n_{\text{unmon}} = n_{\text{mon}} = 8\,500$ (because we have $n_{\text{unmon}} = n_{\text{mon}}$ in the `Wang` dataset), and two-thirds of them in the training set and one third in the testing set, scenario denoted hereafter by *8500*; and, similarly as what is done by Hayes et al. [HD16], with all the unmonitored websites ($n_{\text{unmon}} = 100\,207$), and 5% of them in the training set (i.e., 5 010) and 95% of them in the testing set, scenario denoted hereafter by *all*.

The raw traces of these two datasets are hereafter called *original* traces. When a defense is applied on an original trace (i.e., packets are split between the two networks and/or dummy packets are inserted), the resulting trace is called *protected*.

6.4.2 Methodology and Experiments

We assume that the adversary knows the defense (i.e., it knows the splitting scheme used by the client and by the proxy), and that it is able to train a machine learning model on protected traces. In practice, an adversary does not know whether a specific trace is protected (sent through two networks) or not (sent through a single network).[6] Consequently, we assume that it knows that the client has the possibility to use two networks, but that it does not know if the client is actually using one or two networks. We will show that this assumption does not weaken the adversary, as the performance of the WF attacks remains extremely close to that obtained if the adversary knows beforehand that the trace is protected (in other words, the adversary is able to train a model that distinguishes protected traces from original traces).

To evaluate the attacks, we use the true positive rate (TPR) as the main accuracy measure, defined as the probability that a monitored website is classified as the correct monitored website. The lower the TPR is, the more secure the scheme is. We do closed-world and open-world experiments.

Closed-World In the closed-world experiments, the adversary tries to predict which website is visited out of the different monitored websites. Only the monitored websites are used for both training and testing.

Open-World In the more practical open-world experiments, the adversary tries to predict whether the client visits a monitored or an unmonitored website, and if it is a monitored website, which one it is. In the open-world experiments, it is also important to measure if the adversary makes a prediction error when observing the trace of an unmonitored website: In addition to using the TPR, we use the false positive rate (FPR) as a second accuracy measure, defined as the probability that an unmonitored website is incorrectly classified as a monitored website. The higher the FPR is, the more secure the scheme is.

6.5 Designing HyWF: A Defense Without Traffic Overhead

We first study defenses with no overhead: No packet is added to the trace (i.e., no link padding) and no packet is delayed by the client. The only choice that the client and the proxy make is about how packets are split between the two networks. In this section, we evaluate our design decisions in the closed-world experiment with the primary dataset (`Wang`). We will evaluate our defense more broadly in Section 6.6. We want to compare our defense to AP [SW06, JIP$^+$16] because it is the defense that is found to offer the best tradeoff between privacy and performance [HD16]

[6]The adversary is able to know if the client uses MPTCP by looking at the specific TCP field, but this is not enough to characterize the use of the defense, as it is possible to use MPTCP on a single path or only for performance improvement.

and that is currently being considered for addition to the Tor project [Per15]. As opposed to AP, the schemes described in this section do not incur any traffic overhead and do not require statistics on the traces. With the `Wang` dataset in the closed-world experiment, AP reduces the TPR of the attack to around 40% (see Section 6.8 for more details on AP). Our goal is to achieve at least the same accuracy. We first show that off-the-shelf schedulers do not reach this level of privacy. We then show that a random splitting scheme can achieve a level of privacy equivalent to that of AP.

6.5.1 Off-the-Shelf Schedulers

Split Outgoing and Incoming Traffic

The first solution, appealing due to the simplicity of its implementation, is to divide outgoing and incoming traffic, and to send the former through one network, and the latter through the other network. With the `Wang` dataset, this means that 90% of the traffic (the incoming traffic) is sent through one network, and 10% (the outgoing traffic) through the other. Such a scheme reduces to 67% the TPR of the attack against the network through which the incoming traffic is sent, and to 55% against the network through which the outgoing traffic is sent. Even though the privacy improvement is significant, this does not achieve our goal to reach the level of privacy offered by AP. Consequently, we move to schemes where both networks can be used for both incoming and outgoing traffic.

Round-Robin Scheduler

The simplest multipath scheduler is a round-robin scheduler, i.e., a scheduler that sends packets alternatively through the two networks. This scheduler is, for example, proposed in the default Linux kernel MPTCP implementation.[7] The number of consecutive packets sent through one network is denoted by $n_{\text{cons}} \in \mathbb{N}$. We now evaluate whether this scheduler can be used to improve the resistance against WF attacks.

Figure 6.2 shows the accuracy of the WF attack for various values of the number of consecutive packets n_{cons}. The TPR of the attack is almost not reduced, compared to the baseline (indicated by the black horizontal line), going from 91% to 85%. This is because with the round-robin scheduler, the splitting scheme is deterministic, and the adversary is able to learn the characteristics of the protected traces. To increase the level of privacy to the desired goal, off-the-shelf deterministic schedulers are not sufficient. We now study the use of a random splitting strategy.

6.5.2 Fixed Splitting Probability

Here, we assume that the client wants the same level of privacy in the two networks, i.e., she sends on average the same amount of traffic through both networks. In Section 6.6.5, we discuss

[7] See https://multipath-tcp.org/pmwiki.php/Users/ConfigureMPTCP.

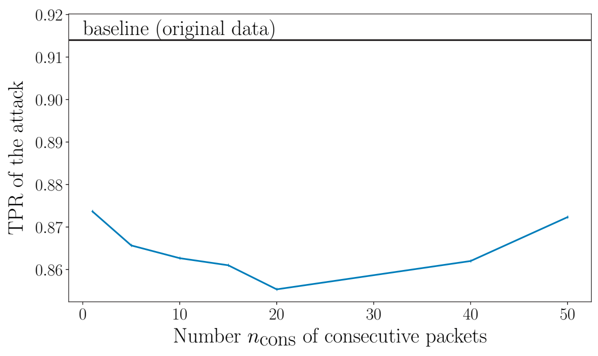


Figure 6.2 – Performance of the attack on traces protected with a round-robin scheme, for different values of the number of consecutive packets n_{cons}. Closed-world, `Wang` dataset.

the case where she wants to send a smaller fraction of her traffic through one of the networks, e.g., because this network is more costly (this naturally degrades privacy in the other network).

We first study the simplest random splitting strategy: For each packet, the client and the proxy randomly send it through Network 1 (with probability $p = 0.5$) or through Network 2 (with probability $1 - p = 0.5$). The accuracy of this very simple strategy depends heavily on the assumption made for the adversary. If the adversary does not know that the client is using a second network (i.e., it learns a model only on original traces), the TPR of the attack is reduced to virtually 0 (around 3%, whereas random guessing gives an accuracy of 1%). But this corresponds to a very weak adversary, hence is too optimistic. If the adversary knows that a second network is used (i.e., it learns a model on protected traces), then the TPR of the attack is again high (around 80%). More importantly, if the adversary learns a model with both protected and original traces (it only knows that the client has the possibility to use two networks, but it does not know if the client is actually using one or two networks), the accuracy is also around 80% for protected traces and around 90% for original traces. This means that the adversary is able to learn if the traces are protected or not. Clearly, this simple random strategy is insufficient.

6.5.3 One Splitting Probability Per Website Access

Alternatively, the client and the proxy can employ the following more complex defense: At the beginning of a website access, they choose a probability p uniformly at random in $[0,1]$, and for this website access, they send packets with probability p through Network 1 and with probability $1 - p$ through Network 2. This means that for each access of a website, the splitting probability is different. In particular, two different accesses of a same website will look different. On average along all website accesses, 50% of the traffic is sent through each network.

The probability used by the client and by the proxy are different, because they are chosen independently; they are respectively denoted by p_c and p_p. The adversary has no access to p_c or p_p, chosen locally for each website access. We assume, however, that it knows the strategy

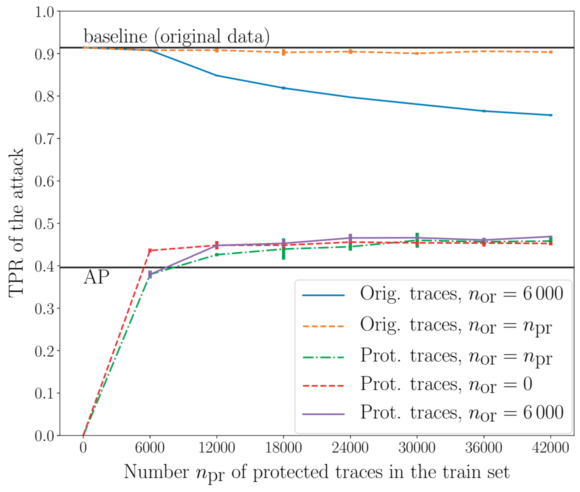


Figure 6.3 – Performance of the attack on original traces and traces protected with the splitting scheme described in Section 6.5.3, for different sizes of the training set. Closed-world, `Wang` dataset.

(i.e., that p is chosen uniformly in $[0,1]$) and it is able to train a model on traces protected with this strategy. With the $n_{or} = 6\,000$ traces in the training set of original traces, the adversary can build a larger training set of n_{pr} protected traces by computing several times the protected trace of any original trace (the protected trace is different each time because p_c and p_p are different each time). Because we assume that the adversary does not know if a trace is protected or not, it must train a model with a training set that consists of both protected and original traces. We evaluate the effect of the defense when the adversary tries to attack original traces and when it tries to attack protected traces.

Effect on Original Traces We start by studying the effect of the defense for an attack against original traces. The results are shown in Figure 6.3 (*Orig. traces*) for different values of n_{pr}; for every n_{pr}, the attack is repeated five times: The results might be different for each attack because the splitting probability p are different. Consequently, the traces in both the training set and the testing set are different. We present averaged results with a bar indicating the standard deviation. Adding more protected traces in the training set and using only the $n_{or} = 6\,000$ original traces tends to decrease the accuracy (plain blue line). However, this comes only from the fact that adding more protected traces biases the training set towards the protected traces, hence reduces the accuracy for the original traces. If the training set contains as many original traces as the number of protected traces ($n_{or} = n_{pr}$, dashed orange line), then the accuracy stays very close to the baseline. Note that, as opposed to the protected traces for which protecting several times the same original trace gives different results, adding each original trace several times does not add any information (as the trace is always the same) but only removes the bias towards protected traces. Because we assume that the adversary does not know whether a trace is protected or not and needs to be able to attack both protected and original traces, we use (unless specified otherwise) $n_{or} = n_{pr}$.

Effect on Protected Traces We then study the effect of the defense on protected traces. The results are shown in Figure 6.3 (*Prot. traces*) in three scenarios: when only protected traces are used to train the model ($n_{\text{or}} = 0$); when the 6 000 original traces are used once to train the model along with the protected traces ($n_{\text{or}} = 6000$); and, to remove the bias towards protected traces, when the number of original traces is the same as the number of protected traces ($n_{\text{or}} = n_{\text{pr}}$). As expected, adding several protected traces for each original trace in the training set helps the adversary: The TPR increases from 37.9%±0.8 when they are not repeated to 45.8%±0.8 when they are repeated seven times. This is because adding several protected traces per original trace enables the attack to implicitly infer the splitting probability p used for one website access, because it becomes more likely that a protected trace for the same website exists in the training set with a splitting probability close to that used for the website access. But the TPR plateaus once n_{or} reaches 36 000/42 000. We also note again that the TPR of the attack is similar when the adversary knows beforehand that the trace is protected ($n_{\text{or}} = 0$) and when the adversary does not know it ($n_{\text{or}} = n_{\text{pr}}$): The adversary is able to distinguish protected traces from original traces.

6.5.4 Consecutive Packets to One Network

Figure 6.3 also shows that this random splitting scheme does not attain our goal of reaching a TPR of the attack below 40%. We next show that the number of consecutive packets sent through one network has an impact on the TPR of the attack. We try two different settings. In the first setting, a number $n_{\text{cons}} \in \mathbb{N}$ is fixed in advance for all traces; when one network is chosen randomly for sending one packet, the source sends n_{cons} packets through this network, before choosing again randomly one of the two networks. In the second setting, the average number of consecutive packets $n_{\text{cons}} \in \mathbb{N}$ is fixed in advance for all traces; when one network is chosen randomly, the source draws randomly a number $c \in \mathbb{N}$ from a geometric distribution with average n_{cons}, and sends c packets through this network, before choosing again randomly one of the two networks and drawing a new value c.[8] Note that n_{cons} is the average number of consecutive packets sent *each time a network is chosen*. The average number of consecutive packets per network for an entire trace is different and depends on p: If p is close to 1, then the probability that Network 1 is chosen several times in a row will be high, and the average number of consecutive packets in Network 1 will be larger than in Network 2.[9]

The results are shown in Figure 6.4. Sending consecutive packets through each network improves the privacy of the defense scheme (it decreases the TPR of the attack). Choosing randomly the number of consecutive packets to send (second scheme, with a geometric distribution) further improves privacy. In Section 6.6.3, we evaluate further the effect of n_{cons} with the `Hayes` dataset.

[8]When c is drawn from a geometric distribution, this forms a two-state Markov chain, and it is equivalent to do the following for each packet: When the last packet was sent through Network 1, the packet is sent through Network 2 with probability $(1-p)/n_{\text{cons}}$ and through Network 1 with probability $1-(1-p)/n_{\text{cons}}$; when the last packet was sent through Network 2, the packet is sent through Network 1 with probability p/n_{cons} and through Network 2 with probability $1-p/n_{\text{cons}}$.

[9]The average numbers of consecutive packets for an entire trace are $n_{\text{cons}}/(1-p)$ in Network 1 and n_{cons}/p in Network 2.

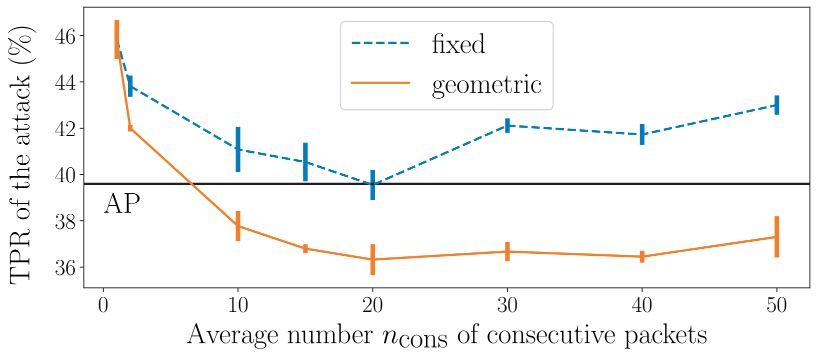


Figure 6.4 – Performance of the attack on traces protected with the splitting scheme described in Section 6.5.4, for different values of the average number of consecutive packets n_{cons}. Closed-world, `Wang` dataset.

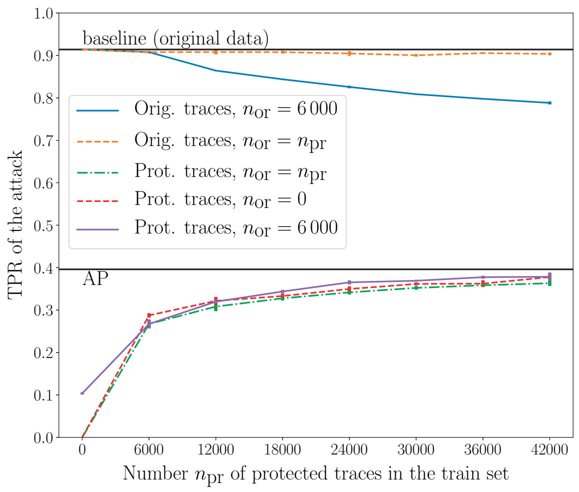


Figure 6.5 – Performance of the attack on original traces and traces protected with HyWF, for different sizes of the training set. Closed-world, `Wang` dataset.

Algorithm 2 – Pseudo-code of HyWF.

```
$n_{\text{cons}} = 20$
**for each** website access **do**
    Draw $p$ from Unif$([0,1])$, $n \leftarrow 0$, $c \leftarrow 0$
    **for each** packet **do**
        $n \leftarrow n + 1$
        **if** $n > c$ **then**
            Draw $c$ from $\mathrm{G}_0\,(1/n_{\text{cons}})$ and $i$ from Bern$(p)$
            $n \leftarrow 0$
        **end if**
        Send packet through Network $i$
    **end for**
**end for**
```

6.5.5 Dynamic Splitting Probability

In Sections 6.5.3 and 6.5.4, the splitting probability p remains the same for the duration of a website access. But p can also be changed along each website access. However, if p is drawn each time uniformly at random in [0,1], experiments show that the privacy is significantly degraded (i.e., the TPR of the attack is higher). The reason is that when redrawing each time p in [0,1], some features (e.g., the total number of packets) are close to those obtained with a fixed splitting probability $p = 0.5$, shown to be inefficient (Section 6.5.2).

Hence, we evaluate a strategy where an average splitting probability p_d is chosen uniformly at random in [0,1] at the beginning of the website access. Then, the splitting probability p is changed along each website access and regularly drawn uniformly at random in an interval such that the average value is p_d.[10] We try several strategies: changing the splitting probability p after a fixed or geometrically-distributed duration, after a fixed or geometrically-distributed number of packets, with different average values. None of these configurations improves the resistance against the state-of-the-art attacks considered in this chapter (the TPR stays the same): The variability added by the scheme described in Section 6.5.4 (switching network after a geometrically-distributed number of packets) is sufficient against existing attacks. Adding more randomness, by changing p sufficiently often, could however help against an attack that, targeting specifically HyWF, would infer the splitting probability used for each website access — with a scheme more efficient than the one we used (repeating the protected traces several times in the training set). As no such attack currently exists, this is out of the scope of this dissertation.

6.5.6 Definition of HyWF

The scheme defined in Section 6.5.4 is the simplest scheme that achieves our goal of reaching a level of privacy equivalent to that of AP. It choses a splitting probability uniformly at random for each website access and uses a number of consecutive packets drawn from a geometric distribution with average $n_{\text{cons}} = 20$. The TPR of the attack with $n_{\text{pr}} = 42\,000$ is decreased to 36.3%±0.6, i.e., it performs even a bit better than AP. We denote this scheme by HyWF. Algorithm 2 presents the pseudo-code for HyWF, with G_0, Unif and Bern denoting, respectively, the geometric, uniform and Bernoulli distributions. The TPR of the attack against original traces and traces protected with HyWF as a function of the number of protected traces n_{pr} is shown in Figure 6.5. Similarly as in Section 6.5.3 (Figure 6.3), the TPR clearly plateaus (near 36%).

6.6 Evaluation of HyWF

We now evaluate HyWF with different WF attacks and different datasets, and with open-world experiments. We also compare HyWF with AP.

[10]Formally, p is drawn in $[0, 2p_d]$ if $p_d \le 0.5$ and in $[2p_d - 1, 1]$ if $p_d \ge 0.5$.

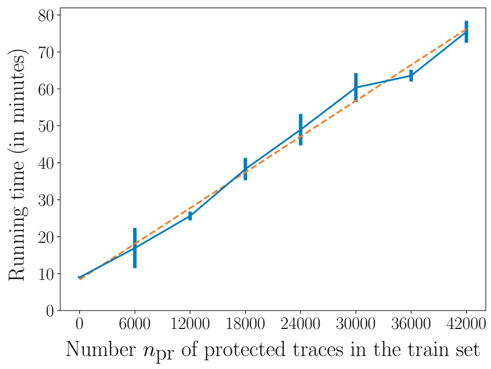


Figure 6.6 – Running time to perform the attack, for different sizes of the training set. Experimental values are in blue; the orange dashed line is the linear fit. Closed-world, *k*-fingerprinting attack against HyWF with `Wang` dataset.

6.6.1 Complexity of the Attack against HyWF

To increase the accuracy of the attack against HyWF by implicitly inferring the splitting probability, the protected traces are repeated several times in the training set. Adding more traces increases the TPR of the attack, but it also significantly increases the complexity of the learning phase of the model.

Closed-World In the closed-world experiment, the running time necessary for the attack increases linearly and reaches more than one hour (see Figure 6.6). In contrast, when attacking only original traces ($n_{\mathrm{pr}} = 0$, i.e., when the client uses a single network), performing the attack requires only eight minutes. Learning the model against protected traces also incurs significant costs in terms of memory: When $n_{\mathrm{pr}} > 42\,000$, a machine with 8 GB of RAM cannot learn the model anymore, because it requires too much memory. We note that using more powerful machines does not further increase the TPR of the attack: the TPR clearly reaches a plateau (see Figure 6.3 and Figure 6.5). We also check this by computing the TPR with $n_{\mathrm{pr}} = 60\,000$ (i.e., the protected traces are repeated ten times), equal to 36.8%±0.8, within the standard deviation of the TPR with $n_{\mathrm{pr}} = 42\,000$, equal to 36.3%±0.6. Hereafter, for the closed-world experiments, the attack always uses $n_{\mathrm{or}} = n_{\mathrm{pr}} = 42\,000$ (unless specified otherwise). For the closed-world experiments, this means that each original trace of the training set is protected seven times, and the seven different protected versions are added to the training set.

Open-World In the open-world experiment, there are twice more original and protected traces, and the running time of the attack is further increased. It grows linearly from 45 minutes with $n_{\mathrm{pr}} = 0$ to 589 minutes with $n_{\mathrm{pr}} = 60\,000$ (each trace is repeated five times). When $n_{\mathrm{pr}} > 36\,000$, a machine with 8 GB of RAM cannot learn the model anymore, because it requires too much memory. Here also, we verify that using more powerful machines does not help the adversary: The average TPR with the `Wang` dataset is 14.7%±0.9 with $n_{\mathrm{pr}} = 36\,000$, and it is 14.9%±0.8 with $n_{\mathrm{pr}} = 60\,000$. In contrast, it is 9.9%±0.4 when the protected traces are not repeated ($n_{\mathrm{pr}} = 12\,000$). Hereafter, for the open-world experiment, the attack always uses $n_{\mathrm{or}} = n_{\mathrm{pr}} = 36\,000$.

6.6.2 Different WF Attacks

In the results presented above, we show only the highest accuracy among the attacks k-NN [WCN$^+$14], CUMUL [PLP$^+$16], k-fingerprinting [HD16] and Ensemble [OJA$^+$17]. The best results against protected traces are always obtained with k-fingerprinting. In Table 6.1, we also show the results obtained with the three other attacks, k-NN, CUMUL and Ensemble. Each attack is repeated five times and Table 6.1 also presents the standard deviation. The first line of the table (*Orig. traces*) corresponds to the case $n_{\text{or}} = 6\,000$ and $n_{\text{pr}} = 0$ (the model is trained only on the original traces). We also show the results with a model trained on original and protected traces for an attack against original traces (second line) and protected traces (third line); we compute the TPR for different values $n_{\text{or}} = n_{\text{pr}}$ between 6 000 and 42 000 and keep the best result. For HyWF, k-fingerprinting yields results significantly better than the other three attacks. Note in particular that with k-NN and CUMUL, the performance of the model that works for both protected and original traces (an assumption that is necessary for a practical attack) is decreased even when attacking original traces. Because it uses only the cumulative sum of the packet sizes as features, CUMUL has already been shown to perform less well than k-fingerprinting (that uses more diverse features) when attacking protected traces [HD16]. k-NN has already been shown to perform less good than k-fingerprinting and better than CUMUL against protected traces [HD16], which is consistent with our results. Because k-fingerprinting performs significantly better than k-NN and CUMUL, combining the three attacks (Ensemble) does not help. In the remainder of the chapter, we show results only of k-fingerprinting.[11]

6.6.3 Open-World Experiment and Different Datasets

HyWF

We obtained the above results with the `Wang` dataset in the closed-world experiment. We now study open-world experiments (a scenario that is more practical than closed-world experiments) with both the `Wang` and `Hayes` datasets. The two datasets and the two scenarios with which the `Hayes` dataset is used are described in Section 6.4.1. The performance achieved by HyWF is reported in Table 6.2, where we present the true positive rates (TPR, proportion of monitored sites

[11] All the attacks presented in this chapter have been performed with the four attacks (k-fingerprinting, k-NN, CUMUL, and Ensemble); k-fingerprinting always performed significantly better than the others.

Table 6.1 – Performance of different attacks with a model learned on original traces (*Orig. traces*), and with a model learned on original traces and traces protected with HyWF. Closed-world, `Wang` dataset.

	k-fingerprinting	k-NN	CUMUL	Ensemble
Orig. traces	90.7%±0.3	91.0%±0.3	90.2%±0.6	91.4%±0.6
HyWF, orig. traces	90.8%±0.3	82.6%±1.9	75.9%±1.3	90.6%±0.2
HyWF, prot. traces	36.3%±0.6	15.3%±1.7	12.4%±0.4	26.8%±0.4

correctly classified) and the false positive rates (FPR, proportion of unmonitored sites classified as monitored). The TPR of the attack is decreased significantly, compared to the closed-world experiments; and with the `Wang` dataset, the TPR goes down from 36.3% to 14.7%. Protecting the traces also increases the FPR. Note that, in practice, the unmonitored websites represent the vast majority of the visited websites, and even a slight increase of the FPR significantly increases the number of false positives. In the `Hayes` dataset, the number of monitored websites is smaller than in the `Wang` dataset (85 instead of 100), hence we expect the TPR of the attack to be higher. This is indeed what we observe, with a TPR around 28% with the `Hayes` dataset. Nevertheless, the defense decreases significantly the TPR of the attack, by almost a factor 3. Also, the FPR is higher with the `Hayes` dataset than with the `Wang` dataset.

Impact of n_{cons}

In Section 6.5.4, we evaluated the impact of the average number of consecutive packets n_{cons} with the `Wang` dataset in the closed-world experiment. To verify that HyWF generalizes to other datasets, we evaluate here the impact of n_{cons} with the `Hayes` dataset in the open-world experiment. In Figure 6.7, we show the TPR as a function of n_{cons}. Figure 6.4 and Figure 6.7, obtained with two different datasets, show that the effect of variations of n_{cons} is small as long as n_{cons} is in the range 15-40. This is an important feature for the generalization of HyWF, as it means that the parameter n_{cons} does not need to be fine tuned for a specific dataset.

6.6.4 Comparison with AP

In Table 6.2, we also show the TPR and FPR of the attack when the traces are protected with AP, the favored state-of-the-art option if link padding were to be implemented in practice [HD16, Per15] (see Section 6.8 for more details on AP). We observe that the TPR with HyWF is similar or lower to that of AP, and that the FPR is higher with HyWF, i.e., it improves privacy. Note that, as opposed to AP, HyWF achieves such a performance without adding any traffic overhead. The traffic overhead of AP is reported in Table 6.2 and denoted by *ov.* (a traffic overhead of 65% means that 65 dummy packets are inserted for a trace of 100 real packets). In addition, as opposed to AP, HyWF does not require any a priori knowledge of the traces statistics (see Section 6.8).

Table 6.2 – Performance of the attack with a model learned on original traces and traces protected with HyWF, and with a model learned on original traces and traces protected with AP, for different datasets. Open-world.

datasets	`Wang`			`Hayes` (8 500)			`Hayes` (all)		
	ov.	TPR	FPR	ov.	TPR	FPR	ov.	TPR	FPR
HyWF, orig. traces	0%	88.7%	0.2%	0%	81.3%	1.1%	0%	83.1%	1.6%
HyWF, prot. traces	0%	14.7%	1.0%	0%	27.7%	1.8%	0%	28.0%	1.7%
AP	65%	20.6%	0.6%	47%	29.2%	1.1%	47%	27.8%	1.2%

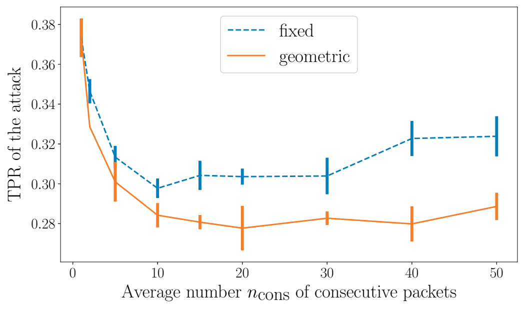


Figure 6.7 – Performance of the attack on traces protected with the splitting scheme described in Section 6.5.4, for different values of the average number of consecutive packets n_{cons}. Open-world, `Hayes` dataset.

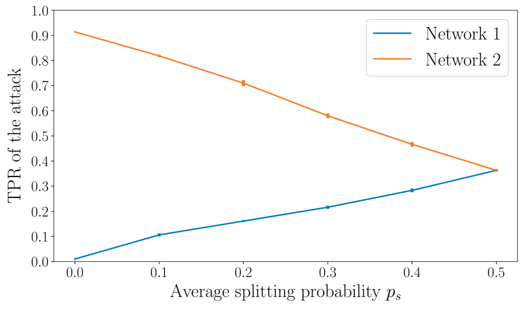


Figure 6.8 – Performance of the attack when the client sends a fraction p_s of its traffic through Network 1, for different values of p_s. Closed-world, `Wang` dataset.

6.6.5 Asymmetric Networks

We evaluate the scenario where the client wants to send a smaller fraction $p_s < 0.5$ of her traffic through one of the networks (denoted by Network 1), for example because this network is more costly (e.g., WiFi is typically cheaper to use than cellular). We evaluate the same splitting scheme as HyWF, except that p is chosen uniformly at random in $[0, 2p_s]$, so that the average value of p along all traces is p_s. Figure 6.8 shows the TPR of the attack against Network 1 and Network 2 as a function of p_s. We observe that the relationship between cost (i.e., amount of data sent on the costly network) and privacy is linear, which makes it simple for a client to decide how to make the tradeoff between the two.

6.6.6 Deployability of HyWF

It would be very challenging, if not impossible, to make all servers implement HyWF, especially when the benefit for the server would be inexistent. But if only the client uses HyWF, the privacy is significantly degraded: If the server uses the off-the-shelf MPTCP round-robin scheduler, the TPR goes from 36% (when both the client and the proxy use HyWF) up to 72% (when the client uses HyWF and the server just MPTCP), which is an unacceptable privacy loss. To

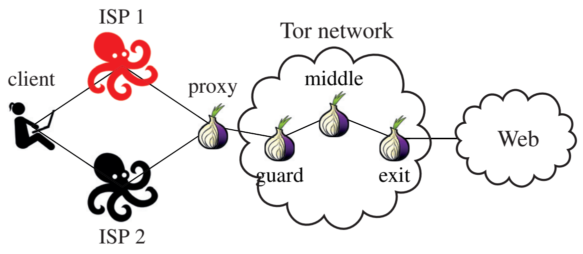


Figure 6.9 – Model where the client is multihomed, with two different curious ISPs, and connects to a Tor proxy.

protect against WF with an easily deployable solution, we use a proxy-based architecture, as depicted in Figure 6.9. This approach with a Tor proxy has also been employed by previous work [JIP$^+$16, WG17]. The key advantage is that it can be easily deployed by installing standard Tor solutions, along with the modified MPTCP modules that implement HyWF scheduling.

6.7 Implementation of HyWF

We now describe the proof-of-concept implementation of HyWF and evaluate its performance.

6.7.1 Hardware and Software Setup

We prototype a proof-of-concept implementation of HyWF and assess its performance with the testbed shown in Figure 6.10. It consists of an Intel-based PC equipped with an i7-6700 (3.40GHz) CPU and 16 GB of RAM, running Ubuntu 16.04, and placed in the DMZ of one of our institutions, to ensure a proper connectivity to the Tor network. This computer acts as a virtualization server running KVM, and it instantiates two Ubuntu 16.04 machines, one acting as client, and the other acting as Tor proxy. Each machine is provided with two different interfaces and IP addresses that are connected with the virtual links represented in Figure 6.10 (denoted as *Link 1* and *Link 2*). Their delay is artificially controlled throughout our experiments by using the Linux `tc` command. To ensure repeatability and ease of scripting, we develop a Python-based command-line web client that requests `.onion` resources via a SOCKS5 proxy provided by the Tor software. Both the client and the server are provided with two additional control interfaces in order to control the execution of the experiments (not shown in the figure). The proxy runs Tor v.0.3.3, and both the client and the server run MPTCP v.0.9.1. To implement HyWF, we do not need to modify the Tor software and we implement only a novel MPTCP scheduler, as described next.

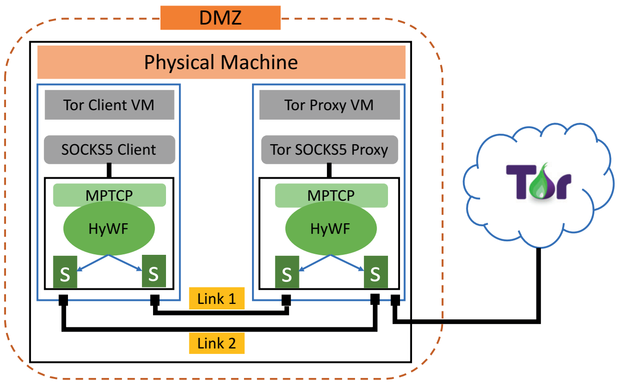


Figure 6.10 – Deployment used for the implementation and performance evaluation of HyWF. (Figure courtesy of Ginés García Avilés.)

6.7.2 HyWF Scheduler

Our HyWF scheduler builds on the round-robin scheduler provided with the MPTCP implementation and configured with the `full-mesh` mode of operation. With this mode, MPTCP generates one subflow for each IP address pair (source, destination). Therefore, we set up some `iptables` rules to ensure that only the two subflows that correspond to the two links of Figure 6.10 are available (this is queried with the `mptcp_rr_is_available` method). Instead of operating in "bursts" of segments, i.e., picking one of the paths at random with probabilities p for Link 1 and $(1-p)$ for Link 2, then drawing the consecutive numbers of segments from a geometric distribution with average n_{cons}, we operate for simplicity on a segment-by-segment basis and use the equivalent algorithm described in Footnote 8 (Section 6.5.4). We code this algorithm inside the `mptcp_write_xmit` method. More specifically, the `next_segment` method returns a pointer to the next link to use; the next link is determined based on the last link used (stored in a `*sock` pointer inside an `mptcp_cb` structure) as follows:

- If the last segment was transmitted over Link 1, the next segment is transmitted over the same link with probability $1-(1-p)/n_{\text{cons}}$, and over Link 2 with probability $(1-p)/n_{\text{cons}}$
- If the last segment was transmitted over Link 2, the next segment is transmitted over the same link with probability $1-p/n_{\text{cons}}$, and over Link 1 with probability p/n_{cons}.

The parameters n_{cons} and p are also stored in the `mptcp_cb` structure, with p randomly chosen on a per-download basis.[12] Due to the kernel programming constraints (in particular, the lack of `float` variables), p is chosen as a random integer in the range $[0,255]$ using the `get_random_bytes()` function with an `unsigned char`. The algorithm is

[12]Our proof-of-concept implementation chooses a new splitting probability p for each new MPTCP connection. It has been shown that it is possible to efficiently split different website accesses by using a time-based splitting with a fixed threshold of 1 second [WG16].

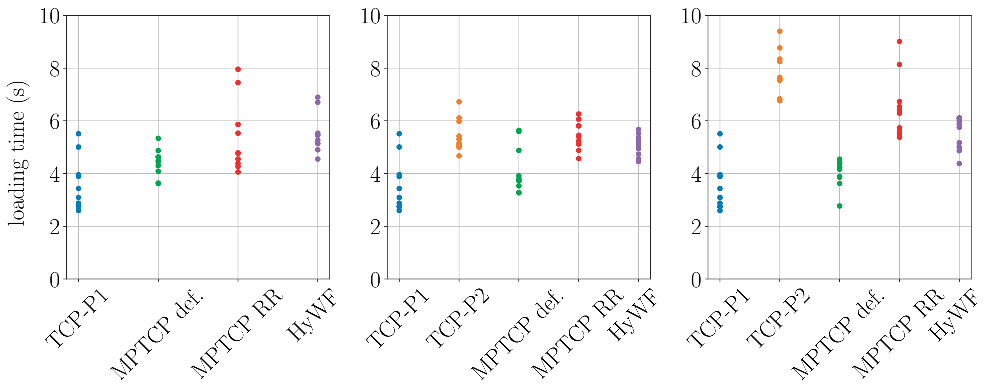


Figure 6.11 – 90-th percentile of the loading times for different configurations of the link delays: 50 ms for Link 1 and Link 2 (left), 50 ms for Link 1 and 150 ms for Link 2 (center) and 50 ms for Link 1 and 300 ms for Link 2 (right). The different transport mechanisms are single path TCP (TCP-P1 and TCP-P2, respectively on Link 1 and Link 2), and MPTCP with default scheduler (def.), round-robin scheduler (RR) and HyWF.

then updated accordingly. Finally, we follow recommendations to improve latency with MPTCP [CT14, YFD+16], and we disable the idle restart functionality and the Nagle algorithm (`tcp_low_latency=1` and `tcp_slow_start_after_idle=0`).

6.7.3 Performance Evaluation

The effectiveness of HyWF in improving privacy has been thoroughly analyzed in Sections 6.5 and 6.6. Here we analyze if using HyWF results in any costs in terms of user experience. To this aim, we focus on the *loading time* of resources over the Tor network when using HyWF, and we compare it against a number of other transport mechanisms (discussed next). Our methodology is as follows: Given a configuration of the delays for Link 1 and Link 2 and a transport mechanism, we download a set of 40 resources from the Tor network (20 being the most popular websites according to Alexa, and the other 20 selected from the various `.onion` resources accessible following our institution's policies). We compute the 90-percentile of the loading times. For each configuration, we repeat the experiment 10 times to gain statistical information.

We consider three configurations for the pair of delays for Link 1 and Link 2, namely, (a) {50,50} ms, (b) {50,150} ms, and (c) {50,300} ms. This enables us to understand the impact of the link delays on performance. These settings are representative for various cases of interest: For instance, configuration (a) can represent the case where the client uses two WiFi links or two cellular links from two different ISPs (Figure 6.1, right and center right), and configurations (b) and (c) can represent the case where the client uses WiFi and cellular (Figure 6.1, left and center left). For each scenario, we compare the performance of HyWF against that obtained with the following transport mechanisms: (i) MPTCP with the default configuration (denoted as def.), (ii) MPTCP using the round-robin scheduler (denoted as RR), and (iii) TCP over a single link (denoted as TCP-P1 and TCP-P2 for the case of Link 1 and Link 2, respectively). The results are shown in Figure 6.11, in which each subplot considers a different configuration of the link delays, each dot is the 90-percentile of the loading time of the 40 resources, and each transport alternative is represented with a different color.

From the results, we observe that HyWF provides performance comparable to the other approaches in terms of loading times. When compared to single-path TCP, the performance of HyWF stays between TCP on the short path (TCP-P1) and TCP on the long path (TCP-P2), which is expected given that HyWF employs both paths. When compared to the other MPTCP approaches that employ both paths, HyWF provides a comparable performance. Thus, we conclude that HyWF is able to provide a high level of privacy without paying a significant price in performance.

6.8 Designing HyWF-AP: A Defense with Adaptive Padding

We now show that HyWF is compatible with other defenses against WF. To improve privacy, the client can employ link padding, i.e., insert dummy packets to confuse the adversary. Because it does not see the content of the packets, the adversary is not able to distinguish real packets from dummy packets. When the client uses a single network and wants to avoid any significant loading-time overhead, link padding is the only possible defense, because she cannot remove packets (by sending them through the other network). Consequently, link padding is the method used by the vast majority of existing defenses (see Section 6.2.1). Here, we use AP [JIP+16, SW06], because it is the defense that is under consideration for addition to Tor if link padding were to be implemented [Per15]. Our defense would also be compatible with most of the other state-of-the-art defenses described in Section 6.2.1.

6.8.1 Adaptive Padding (AP)

We briefly describe the AP defense. As mentioned in Section 6.2.1, AP works on ensuring that the distribution of the inter-arrival times is the same for all traces. The packets are never delayed, which means that there is no loading-time overhead. The goal of AP is to disrupt statistically unlikely delays between packets. In particular, if an unusually large gap between two packets is found, AP adds dummy packets to hide this large gap and to prevent it from being used as a distinguishing feature. Because the authors noticed that bursts of packets play an important role in identifying websites, AP mimicks bursts of packets when filling the gaps. Details on AP can be found in the related literature [JIP+16, SW06]. Here, we use the implementation of AP for WF, provided by Juarez et al. [JIP+16].

With AP, both the client and the proxy use as a parameter some probability distribution for the inter-arrival times; which distribution is used has a strong impact on the performance of AP. Here, we try several distributions. First of all, as Juarez et al. do, we fit a normal distribution on all the inter-arrival times for the entire dataset (training set and testing set). This distribution is denoted by *dataset distribution*. This is the most favorable scenario for AP, because the distribution is computed on the traces that are used for the experiments. But this would be challenging to obtain in practice, as the client would need to gather in advance a large number of traces to compute the distribution, and the proxy would need to know this distribution for each client. As explained by Juarez et al. [JIP$^+$16], this original normal distribution can be tuned to favor shorter inter-arrival times (denoted by *tuned+*), which improves privacy but also increases the traffic overhead. It can also be tuned to favor longer inter-arrival times (denoted by *tuned-*), which decreases the traffic overhead and degrades privacy. We also compare with a more practical scenario where a default distribution (i.e., a distribution that is not computed based on the traces in the dataset) is used, denoted by *default distribution*. We use the default normal distribution provided by Juarez et al. with their code. In Table 6.3, we present the results obtained for the different distributions. We report the traffic overhead and the TPR of k-fingerprinting.

With HyWF, the defense described in Section 6.5, the TPR of the attack is decreased to a level similar to that of AP, and it adds no traffic overhead. In addition, as opposed to AP that requires knowing the distribution of inter-arrival times, HyWF does not require any prior knowledge on this distribution. We note that when AP is used with a default distribution, the privacy achieved is much worse than with HyWF.

6.8.2 Designing HyWF-AP

Despite its limitations (traffic overhead, requires knowing the distribution of the inter-arrival times), AP is able to improve privacy significantly when only one network is available. Therefore, we study a solution where HyWF, the defense we describe in Section 6.5, is used along with AP.

Splitting Strategy

We use HyWF, the splitting strategy described in Section 6.5.6. We also try the strategy with a dynamic splitting probability, as defined in Section 6.5.5: An average splitting probability p_d is chosen uniformly at random in [0,1] at the beginning of a website access, and the actual splitting probability p is drawn again in a uniform distribution with average value p_d after a number of packets drawn from a geometric distribution with average n_{chp}.

Table 6.3 – Performance of AP and HyWF. Closed-world, `Wang` dataset.

	traffic overhead	TPR
AP [SW06, JIP$^+$16] (dataset distribution)	65%	39.6%
AP (dataset distribution, tuned+)	105%	34.0%
AP (dataset distribution, tuned-)	37%	47.9%
AP (default distribution)	28%	63.8%
HyWF (Section 6.5)	0%	36.3%

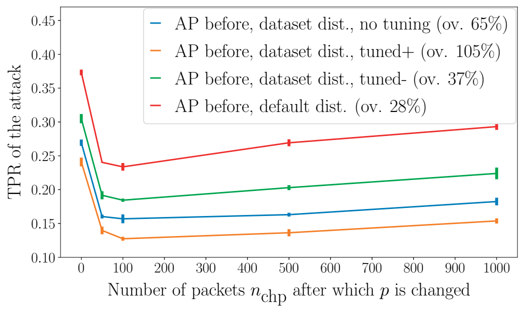


Figure 6.12 – Performance of the attack on traces protected with AP before, for different values of the average number of packets n_{chp} after which the splitting probability p is changed along one website access. Closed-world, `Wang` dataset.

AP Strategy

As explained above, AP requires setting the distribution of the inter-arrival times. We try the four distributions described in Section 6.8.1 (dataset, dataset tuned-, dataset tuned+ and default). We also try two different strategies: Either AP is added to the original trace, then the packets are split between the two networks (denoted by *AP before*); or the packets are split between the two networks, then AP is added (denoted by *AP after*). When AP is added after splitting the traffic, we do not want to use the same distribution as when AP is added before, because it would yield an overhead twice larger (AP is now done independently for both networks). Instead, when traffic is sent through Network 1 with probability p, each inter-arrival time drawn for the distribution is multiplied by $1/p$. Note that AP and splitting are both done at the same place (either the client or the proxy), thus AP knows which probability p is used for the splitting.

Impact of n_{chp} and AP Strategy

The results with AP before are shown in Figure 6.12, for different values of the average number of packets n_{chp} after which the probability is changed along one website access, and for different distributions (dataset distribution with or without tuning, and default distribution). The traffic overhead is indicated in the legend. We observe that with AP, changing the number of packets along one website access significantly decreases the TPR of the attack, whereas it was not the case without AP (see Section 6.5.5). The best level of privacy is found when $n_{\text{chp}} = 100$. The

Table 6.4 – Comparison of AP before and AP after with $n_{\text{chp}} = 100$. Closed-world, `Wang` dataset.

	traffic overhead	TPR of the attack
HyWF-AP, dataset distribution (AP after)	108%	18.3%
HyWF-AP, dataset distribution (AP before)	65%	15.7%
HyWF-AP, default distribution (AP after)	34%	24.5%
HyWF-AP, default distribution (AP before)	28%	23.4%

results with AP after are worse than with AP before (larger overhead and degraded privacy), as shown by the results with $n_{\text{chp}} = 100$ presented in Table 6.4 for the dataset and the default distributions (it is true as well for the tuned dataset distributions). Hereafter, we always employ AP before. The strategy with AP before and $n_{\text{chp}} = 100$ is denoted by HyWF-AP.

6.8.3 Evaluation of HyWF-AP

Closed-World

With HyWF-AP and for a traffic overhead of 65%, the TPR of the attack in the closed-world experiment decreases to 15% when the splitting probability is changed every 100 packets. Whereas with the same overhead, the TPR of the attack against AP alone is 39.6%, greater than the TPR of HyWF, where no traffic overhead is added. Compared to AP alone and for the same traffic overhead, HyWF-AP significantly improves privacy against WF attacks, as shown in Table 6.5.

Table 6.5 – Performance of AP and HyWF-AP. Closed-world, `Wang` dataset.

	traffic overhead	TPR of AP	TPR of HyWF-AP
dataset distribution	65%	39.6%	15.7%
dataset distribution, tuned+	105%	34.0%	12.7%
dataset distribution, tuned-	37%	47.9%	18.4%
default distribution	28%	63.8%	23.4%

Table 6.6 – Performance of the attack with a model learned on original traces (*Orig. traces*), with a model learned on original traces and traces protected with HyWF-AP, and with a model learned on original traces and traces protected with AP, for different datasets and different AP distributions. Open-world.

datasets	`Wang`			`Hayes` (8 500)			`Hayes` (all)		
	ov.	TPR	FPR	ov.	TPR	FPR	ov.	TPR	FPR
Orig. traces	0%	81.9%	0.34%	0%	80.0%	0.94%	0%	82.7%	0.98%
AP, dataset	65%	20.6%	0.60%	47%	28.3%	1.36%	47%	27.8%	1.24%
HyWF-AP, dataset	65%	1.9%	0.93%	47%	10.0%	1.48%	47%	10.2%	1.26%
AP, dataset tuned+	105%	8.4%	0.20%	69%	28.3%	1.21%	69%	26.7%	1.35%
HyWF-AP, dataset tuned+	105%	1.2%	0.82%	69%	9.7%	1.28%	69%	10.3%	1.25%
AP, dataset tuned-	37%	23.2%	0.67%	33%	30.4%	1.33%	33%	29.5%	1.22%
HyWF-AP, dataset tuned-	37%	3.4%	1.10%	33%	10.4%	1.26%	33%	10.2%	1.35%
AP, default	28%	29.9%	0.17%	19%	48.1%	1.37%	19%	47.5%	1.61%
HyWF-AP, default	28%	3.9%	0.82%	19%	17.8%	1.74%	19%	17.3%	1.82%

Open-World

We now compare AP with HyWF-AP in the open-world experiment. In Table 6.6, we report the results with the `Wang` and `Hayes` datasets, described in Section 6.4.1. Even though there are some differences in the results between the two datasets, we observe the following facts that hold in all cases.

First, when they use the same distribution, HyWF-AP always performs significantly better than AP, with a TPR divided by around 3 at least. With the dataset distribution, the TPR drops to between 1% and 3% with the `Wang` dataset, and around 10% with the `Hayes` dataset. This means that the adversary is practically unable to detect that the client visits a monitored website.

Second, the TPR of the attack against HyWF-AP with the default distribution (that is not computed based on the statistics of either of the two datasets) is 10% smaller than the TPR of the attack against AP with the dataset distribution (that requires knowing statistics on the datasets), with a much smaller traffic overhead. This means that employing HyWF-AP with a default distribution enables a better privacy than AP with the dataset distribution, is easier to implement in practice (because it does not require computing in advance statistics on the inter-arrival times) and reduces the traffic overhead by more than 2x.

Third, it is possible to reach a TPR below 10%, with a reasonable overhead of around 30%. This is better than all state-of-the-art defenses [HD16, WG17].

6.9 Summary

In this chapter, we have presented HyWF, a novel defense against website fingerprinting attacks. HyWF exploits multihoming and multipath, enabled in particular by hybrid networks, to split the traffic between two networks. We have shown that a high level of privacy cannot be reached with off-the-shelf multipath schedulers. We have designed an algorithm based on random splitting that achieves a privacy similar to that of state-of-the-art defenses — without any traffic overhead. We have presented a proof-of-concept implementation of HyWF and showed that it does not add any significant loading-time overhead. HyWF is compatible with other defenses that rely on link padding or randomized pipelining. Combining HyWF with another defense further improves privacy. We have illustrated this by introducing and evaluating HyWF-AP, an extension of HyWF with adaptive padding. HyWF-AP decreases the accuracy of the website-fingerprinting attacks below 10% with a reasonable traffic overhead of 30% and no significant loading-time overhead, which is better than all state-of-the-art defenses.

7 Conclusion

Mesh networks, hybrid networks and multipath routing are increasingly employed as a means to improve performance of local networks. But they pose several challenges that we have studied in this dissertation. We have focused on three performance metrics, throughput, latency and privacy.

In a first part, we have studied throughput in hybrid mesh networks. In Chapter 2, we have shown that with shared-medium and orthogonal technologies, multipath routing and hybrid networks are intrinsically related, and that the optimal number of paths is tightly linked with the number of technologies. We have shown analytically that for certain classes of networks, these two numbers are equal. We have verified and extended our analytical results through extensive simulations and testbed experiments. We have then described a heuristic-based interference-aware multipath-routing protocol that focuses on maximizing throughput. In Chapter 3, we have described a multipath interference-aware congestion controller that, although entirely distributed, optimizes a global utility function over all flows of the network. We have evaluated EMPoWER, the algorithm that combines the multipath-routing protocol and the congestion controller, through extensive simulations and testbed experiments. This evaluation shows that hybrid PLC/WiFi networks increase the spatial diversity compared to hybrid multi-channel WiFi networks hence bring further throughput improvements (up to 10x). In Chapter 4, we have described an algorithm that finds the best multipath and the optimal throughput in a dynamic hybrid mesh network. This algorithm is based on the multi-armed bandit framework and is able to switch multipaths when the network conditions change due to mobility or to link capacity variations.

In a second part, we have studied latency in time-varying hybrid networks. In Chapter 5, we have applied a queueing model where the service rate varies between two states (a high-rate state and a low-rate state) to model variations of the link capacities (due for example to variations of the signal or users contending for a network medium). We have studied the impact of these variations on latency in hybrid networks with two technologies. We have shown that when the average service rate is the same for the two technologies, the variance of the service rate plays, as expected, an important role, but that, surprisingly, the technology with the largest variance sometimes yields the smallest delays. We have studied the queueing model with multipath routing and shown that when the arrival rate is smaller than a certain threshold, it is better to use a single path (i.e., a single technology). When the arrival rate is above the threshold, it is better to simultaneously use the two technologies; but the optimal splitting scheme is not easy to find, and we have shown that using the same splitting probability for all arrival rates has a strong negative impact on delays when the service rates are varying.

Finally, in a third part, we have studied privacy. In Chapter 6, we have shown that when a user is connected to two different networks (typically in hybrid networks, in which the user is connected to two different technologies), she can significantly improve her privacy and successfully defend against so-called website fingerprinting attacks by adequately splitting the traffic between the two networks. We have extensively evaluated our splitting scheme and shown that it does not add any significant performance overhead. We have combined our splitting scheme with state-of-the-art defenses against website fingerprinting attacks and improved further privacy at the cost of a slight performance overhead.

Possible Extensions of this Work

In this dissertation, we have validated the performance gains enabled by mesh networks, hybrid networks and multipath routing, which opens interesting perspectives. We have shown that there is a tradeoff between reaching the optimal throughput and adapting rapidly to dynamic conditions. The algorithms for improving throughput, which we propose in the first part of this dissertation, are shown to be optimal and take seconds to adapt when conditions change. For certain applications or certain conditions, this adaptation time could be too long, and other algorithms that would converge faster at the cost of a smaller steady-state throughput would be required. We have shown that hybrid networks and multipath routing can greatly improve latency, but also that achieving these gains is not easy, because keeping the same splitting probability for all arrival rates is significantly sub-optimal in dynamic networks. This calls for the development of specific multipath schedulers that optimize latency and are practical. Finally, we have proposed an algorithm for protecting against website fingerprinting attacks with hybrid networks and multipath routing. It is, to the best of our knowledge, the first defense that exploits these novel solutions. Other attacks have been shown to impinge upon users' privacy: By successfully identifying which webpage of an HTTPS website the user visits, an adversary can expose personal details such as medical conditions, financial affairs and sexual orientation [MHJT14], or it can compromise encrypted voice-over-IP calls [ZF11], to name just a few examples. Hybrid networks and multipath routing can help protect against such attacks, and specific defenses need to be designed and evaluated.

Hybrid Networks: Perspectives and Future Work

More broadly speaking, there remain further open research questions about hybrid networks. In this dissertation, we have focused mostly on high-rate networks. With the emergence of the Internet of things, of smart-home and smart-city solutions, and of solutions for the home automation, to name a few examples, future networks will also include many low-rate and low-power devices. Several technologies for these applications exist, such as HomePlug Green PHY for power-line communications, and Bluetooth Low Energy, WiFi 802.11ah HaLow and Zigbee for wireless communications. All these technologies consume less energy at the cost of a smaller achievable rate. Hybrid networks have the potential to improve performance (reliability, coverage, latency, privacy, etc.) for low-rate applications, but this requires specific algorithms and is one open area of research.

Future networks will consist of a very large number of devices with very different applications (broadband download, home automation, medical diagnosis, etc.), forming the so-called heterogeneous Internet of things (H-IoT). This means that low-rate applications should in addition be compatible with high-rate applications, which might prove difficult with today's standards: For example, home-automation applications might require very low latency, whereas high-rate flows using the same medium and spectrum would interfere with the home-automation applications and might cause a latency too high for them. Enabling interoperation between devices with very different requirements and different technologies while enjoying the gains that hybrid networks offer is a direction of research that could help shape future networks and satisfy stringent performance demands.

A Experimental Testbed

To obtain the experimental results presented in this dissertation, we built a hybrid testbed with PLC and two-channels WiFi. It comprises 22 nodes spread over the second floor of the EPFL BC building. A map of the testbed is presented in Figure A.1. In this appendix, we provide some technical details about our testbed.

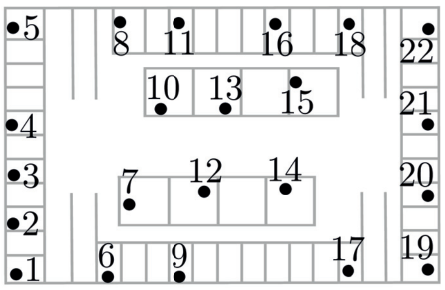


Figure A.1 – Testbed with PLC and two orthogonal WiFi channels (65×40 m).

A.1 Hardware

The nodes are PCEngines APU.1D boards.[1] These boards have AMD G series T40E CPU with a 1 GHz dual Bobcat core, 2 GB of RAM, and three Ethernet interfaces. All the nodes have two WiFi interfaces (Atheros AR9280), connected to the boards with miniPCI express. The first WiFi interface is connected to a channel in the 2.4 GHz band, the second to a channel in the 5 GHz; consequently, they do not interfere. The nodes have two antennas and support up to 2×2 MIMO with 802.11n. 2×2 MIMO is supported only when a single WiFi interface is used; if the two interfaces are used, each interface uses one antenna. When the interface uses two antennas, it is configured to use a bandwidth of 20 MHz; when it uses one antenna, it is configured to use a bandwidth of 40 MHz. This enables that the maximal theoretical PHY-layer data rate is similar in the two configurations, around 150 Mb/s [wif09]. The nodes have one

[1] https://www.pcengines.ch/apu1d.htm

HomePlug AV200 PLC interface (QCA 7420) connected to the electrical network of the building, and connected to the board through gigabit Ethernet, with a Realtek Ethernet driver. The maximal theoretical PHY-layer data rate is 200 Mb/s. With these configurations, the link capacities of WiFi and PLC are on average very close to each other (see Table A.1). There are two PLC subnetworks connected to two different electrical panels: one consists in Nodes 1 to 13, the other consists in Nodes 14 to 22. One link of one subnetwork does not interfere with one link of the other subnetwork, and one link of one subnetwork interferes with every other link of the same subnetwork. The nodes are connected through Ethernet to a central control server. This Ethernet link is used only for control and for the nodes to receive their kernel from the control server at boot time, by using PXE.

A.2 Software

The nodes run an OpenWrt Linux distribution with the open-source ath9k wireless drivers. The OpenWrt distribution is patched for MPTCP.[2] We implemented Python scripts to easily control the nodes (modifying the kernel, running the experiments, gathering the results, etc.). The algorithms are implemented with the *Click Modular Router* [KMC⁺00]. Click can run either in userspace or as a kernel module. Due to some incompatibility between ath9k and Click, our implementations are done only in userspace.

A.3 Capacity and Busy-Time Measurements

For the simulations of this dissertation, the link capacities have been chosen from a distribution based on real measurements conducted on our hybrid testbed. We have fit linearly our real measurements and have computed the variance of the residuals. The simulated capacity have then been computed by adding a normally distributed noise with the corresponding variance to the fitted value. This has been done independently for WiFi and PLC. Estimated capacities for 100 random links are shown as function of the link distance in Figure A.2 for WiFi and PLC, together with the actual measurements. Table A.1 also presents the statistics of the capacities of all the links of our testbed. The average capacity of PLC is very close to that of WiFi.

The link capacities and busy-time (as defined in Section 2.2.1) can be measured directly by the nodes. The physical rate of any link can be obtained by using the modulation and coding

[2]The distribution is available at https://multipath-tcp.org/pmwiki.php/Users/OpenWRT.

Table A.1 – Statistics of the capacities of all links of the testbed, for WiFi and PLC.

	WiFi	PLC
max	102 Mb/s	86 Mb/s
mean	31 Mb/s	30 Mb/s
std dev	30 Mbps	20 Mbps

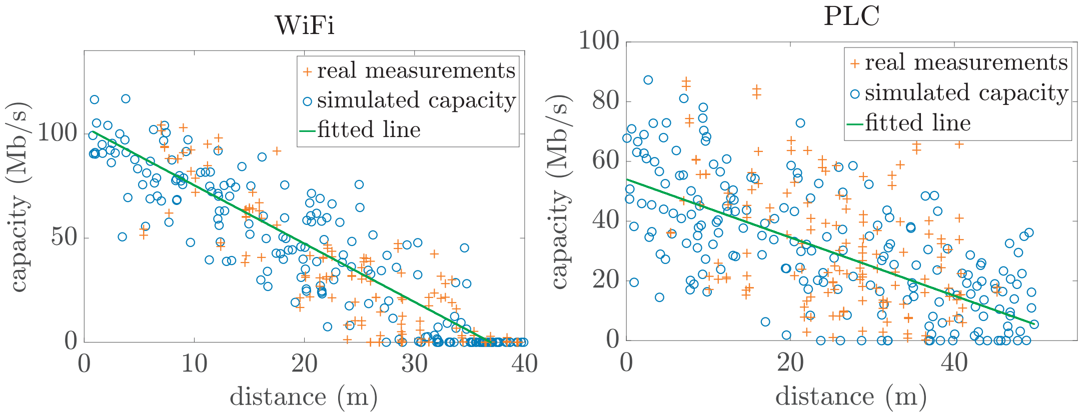


Figure A.2 – Capacity estimation together with real testbed measurements, for WiFi (left) and PLC (right).

scheme (MCS) for WiFi and the bit loading estimate (BLE) for PLC. For WiFi, ath9k drivers expose directly the statistics on the MCS used for each packet that can be accessed using netlink sockets [Hor04]. For PLC, the node sniffs all packets using `faifa` [CF08] and uses the BLE field of every PLC packet to compute the physical rate. The physical rate of the link is directly proportional to the real link-capacity (i.e., the maximum rate that can be sent on the link), as shown by Vlachou et al. [VHT15]. The busy time can be measured similarly. For WiFi, ath9k drivers expose directly the measured busy times that can also be accessed using netlink sockets. For PLC, the node also sniffs all packets using `faifa` and uses the duration field of every PLC packet to compute the busy time.

B Proofs

B.1 Proofs of Section 2.3.2

Before proving the theorems of Section 2.3.2, we define the following term.

Definition 3. *Given a multipath $\mathscr{P} = (P_1, \dots, P_M)$, two multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l_1} \in \mathbb{R}^M$ and $\boldsymbol{\alpha}_{\mathscr{P},l_2} \in \mathbb{R}^M$ are* link-independent *if there is no link of the multipath $\mathscr{P}$ that interferes with both l_1 and l_2, i.e., if $\mathscr{I}_{l_1} \cap \mathscr{I}_{l_2} \cap \Lambda_{\mathscr{P}} = \emptyset$.*

Proof of Theorem 1. Given a multipath $\mathscr{P}$ with M paths, if two links l and l' have same technology, then in a multi-complete network, $\boldsymbol{\alpha}_{\mathscr{P},l} = \boldsymbol{\alpha}_{\mathscr{P},l'}$ (all links of the technology interfere with each other). Consequently, the $L_{\mathscr{P}} \times M$ matrix $\boldsymbol{A}_{\mathscr{P}}$ given by (2.3) can be reduced to a $K \times M$ matrix $\widetilde{\boldsymbol{A}}_{\mathscr{P}}$ without changing the solution of System (2.4) ($\widetilde{\boldsymbol{A}}_{\mathscr{P}}$ has one row per technology). If $M > K$, we next show that for each rate vector $\boldsymbol{x}_{\mathscr{P}} \in \mathbb{R}^M$, it is possible to build a rate vector $\boldsymbol{x}'_{\mathscr{P}} \in \mathbb{R}^M$ that uses $M-1$ paths (i.e., there is an index i such that $x'_i = 0$) and such that $\mathbf{1}^T \cdot \boldsymbol{x}_{\mathscr{P}} \leq \mathbf{1}^T \cdot \boldsymbol{x}'_{\mathscr{P}}$, which proves the claim.

If there is an index i such that $x_i = 0$, then the result is trivially proven with $\boldsymbol{x}'_{\mathscr{P}} = \boldsymbol{x}_{\mathscr{P}}$. Let us now assume that $x_i > 0$ for all $1 \leq i \leq M$, and let $\boldsymbol{c}_1, \dots, \boldsymbol{c}_M \in \mathbb{R}^K$ be the columns of $\widetilde{\boldsymbol{A}}_{\mathscr{P}}$. Then, (2.2) can be written as

$$\sum_{i=1}^{M} x_i \boldsymbol{c}_i = \boldsymbol{\mu}_{\boldsymbol{x}_{\mathscr{P}}}. \tag{B.1}$$

Because $M > K$ and $\boldsymbol{c}_i \in \mathbb{R}^K$ for $1 \leq i \leq M$, there is $(\gamma_1, \dots, \gamma_M) \neq (0, \dots, 0)$ such that

$$\sum_{i=1}^{M} \gamma_i \boldsymbol{c}_i = \mathbf{0}. \tag{B.2}$$

Because the $\boldsymbol{c}_i$ only take positive values, some γ_i are positive and other are negative. Because for all $1 \leq i \leq M$, $x_i > 0$, $\epsilon^+ \doteq \min_{i \text{ s.t. } \gamma_i > 0} \frac{x_i}{\gamma_i} > 0$ and $\epsilon^- \doteq \max_{i \text{ s.t. } \gamma_i < 0} \frac{x_i}{\gamma_i} < 0$; let i^+ and i^- be the indices where the extremum is reached. For all $\epsilon \in [\epsilon^-, \epsilon^+]$ and all $1 \leq i \leq M$, $-x_i \leq \epsilon\gamma_i \leq x_i$;

also, $x_{i^+} - \epsilon^+ \gamma_{i^+} = 0$ and $x_{i^-} - \epsilon^- \gamma_{i^-} = 0$. Clearly, due to (B.1) and (B.2), if $\boldsymbol{x}_{\mathscr{P}} = (x_1, \dots, x_M)$ is an admissible rate vector, then so is $\boldsymbol{x}'_{\mathscr{P}}(\epsilon) = (x_1 + \epsilon\gamma_1, \dots, x_M + \epsilon\gamma_M)$. Let $\sigma_\gamma = \gamma_1 + \cdots + \gamma_M$. If $\sigma_\gamma \geq 0$, then $\mathbf{1}^T \cdot \boldsymbol{x}_{\mathscr{P}} \leq \mathbf{1}^T \cdot \boldsymbol{x}'_{\mathscr{P}}(\epsilon^+)$, whereas if $\sigma_\gamma \leq 0$, then $\mathbf{1}^T \cdot \boldsymbol{x}_{\mathscr{P}} \leq \mathbf{1}^T \cdot \boldsymbol{x}'_{\mathscr{P}}(\epsilon^-)$. Because $x'_{i^+}(\epsilon^+) = x'_{i^-}(\epsilon^-) = 0$, this proves the result with $\boldsymbol{x}'_{\mathscr{P}} = \boldsymbol{x}'_{\mathscr{P}}(\epsilon^+)$ if $\sigma_\gamma \geq 0$ and $\boldsymbol{x}'_{\mathscr{P}} = \boldsymbol{x}'_{\mathscr{P}}(\epsilon^-)$ if $\sigma_\gamma \leq 0$. □

Proof of Lemma 2. To give an intuition, we begin by the proof for the particular case $M = 1$ (i.e., when the multipath $\mathscr{P}$ is a single-path, denoted by P). If traffic is sent on P at an admissible rate x that saturates a technology on a link l_1, then $\sum_{l' \in \mathscr{I}_{l_1} \cap \Lambda_P} x/c_{l'} = 1$. If l_2 is a link using another technology, then $\mathscr{I}_{l_1} \cap \Lambda_P \neq \mathscr{I}_{l_2} \cap \Lambda_P$, i.e., from (2.7), $\sum_{l' \in \mathscr{I}_{l_2} \cap \Lambda_P} \frac{x}{c_{l'}} \neq \sum_{l' \in \mathscr{I}_{l_1} \cap \Lambda_P} \frac{x}{c_{l'}} = 1$. Because the rate x is admissible, $\sum_{l' \in \mathscr{I}_{l_2} \cap \Lambda_P} \frac{x}{c_{l'}} < 1$. This means that l_2 is not saturated.

Lemma 2 is trivial if $M = K$, hence we assume $M < K$. We move to $M \geq 2$. We first show the following two results.

Lemma 3. *If S is a d-dimension affine subspace with $1 \leq d < M$, spanned by $d+1$ link-independent multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_{d+1}}$, then a multipath-impact vector $\boldsymbol{\alpha}_{\mathscr{P},l_0}$, link-independent with all the others, cannot belong to S, unless all vectors in S have the same $M-d$ entries that are zero, i.e., there are (at least) $M-d$ paths $P_1, \dots, P_{M-d}$ such that for all i and j with $0 \leq i \leq d+1$ and $1 \leq j \leq M-d$, $\alpha_{P_j,l_i} = 0$.*

Proof. For $0 \leq i \leq d$, we denote by $\widetilde{\boldsymbol{\alpha}}_i \in \mathbb{R}^M$ the vector $\widetilde{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_{\mathscr{P},l_i} - \boldsymbol{\alpha}_{\mathscr{P},l_{d+1}}$. The M elements of $\widetilde{\boldsymbol{\alpha}}_i$ are denoted by $\widetilde{\alpha}_{ij}$ for $1 \leq j \leq M$. Because S is of dimension d, we know that the $\widetilde{\boldsymbol{\alpha}}_i$'s for $1 \leq i \leq d$ are linearly independent. The vector $\boldsymbol{\alpha}_{\mathscr{P},l_0}$ belongs to S if and only if $\widetilde{\boldsymbol{\alpha}}_0$ is a linear combination of the $\widetilde{\boldsymbol{\alpha}}_i$'s for $1 \leq i \leq d$, i.e., if and only if there is $(\gamma_1, \dots, \gamma_d)$ such that

$$\widetilde{\boldsymbol{\alpha}}_0 = \sum_{i=1}^{d} \gamma_i \widetilde{\boldsymbol{\alpha}}_i. \tag{B.3}$$

With $\boldsymbol{\gamma} = (\gamma_i)_{i \in \{1,\dots,d\}}$ and $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{d \times M}$ the matrix defined by $\widetilde{\boldsymbol{A}} = \begin{bmatrix} \widetilde{\boldsymbol{\alpha}}_1 & \dots & \widetilde{\boldsymbol{\alpha}}_d \end{bmatrix}^T$, this equation is equivalent to $\widetilde{\boldsymbol{A}}^T \cdot \boldsymbol{\gamma} = \widetilde{\boldsymbol{\alpha}}_0$. Let $k \in \{0, \dots, M\}$ be the number of paths such that for all $0 \leq i \leq d+1$ and all $1 \leq j \leq k$, $\alpha_{P_j,l_i} = 0$ or, equivalently, $\widetilde{\alpha}_{ij} = 0$. Without loss of generality, we assume that these k paths are the first k paths of $\mathscr{P}$. With respectively $\widetilde{\boldsymbol{A}}_{M-k} \in \mathbb{R}^{d \times M-k}$ and $\widetilde{\boldsymbol{\alpha}}_{0,M-k} \in \mathbb{R}^{M-k}$ the restrictions of respectively $\widetilde{\boldsymbol{A}}$ and $\widetilde{\boldsymbol{\alpha}}_0$ to their last $M-k$ columns (i.e., the multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l_i}$ are restricted to the last $M-k$ paths of $\mathscr{P}$), then (B.3) is equivalent to

$$\widetilde{\boldsymbol{A}}_{M-k}^T \cdot \boldsymbol{\gamma} = \widetilde{\boldsymbol{\alpha}}_{0,M-k}. \tag{B.4}$$

Let us assume that $k < M-d$, i.e., $M-k > d$. Then, the rank of $\widetilde{\boldsymbol{A}}_{M-k}^T$ is d (because $\widetilde{\boldsymbol{\alpha}}_i$ for $1 \leq i \leq d$ are linearly independent), and $(\gamma_1, \dots, \gamma_d)$ is uniquely defined by d rows of $\widetilde{\boldsymbol{A}}_{M-k}^T$.

Without loss of generality, we assume that these d rows are the first d rows of $\widetilde{\boldsymbol{A}}^T_{M-k}$. For (B.3) to be true, the equality (B.4) also needs to be verified for the last row of $\widetilde{\boldsymbol{A}}^T_{M-k}$, i.e.,

$$\widetilde{\alpha}_{0M} = \sum_{i=1}^{d} \gamma_i \widetilde{\alpha}_{iM}. \tag{B.5}$$

But we know that there is $i \in \{0, \dots, d\}$ such that $\widetilde{\alpha}_{iM} \neq 0$, i.e., $\mathscr{I}_{l_i} \cap \Lambda_{P_M} \neq \emptyset$. Modifying the capacity of one link of $\mathscr{I}_{l_i} \cap \Lambda_{P_M}$ by any small $\epsilon > 0$ changes the value of $\widetilde{\alpha}_{iM}$ without changing the capacity of any other $\widetilde{\alpha}_{ij}$ (because of the link-independence). In particular, it does not change the γ_i's. This means that under Assumption 1, (B.5) cannot be true. Consequently, for $k < M - d$, there is no $(\gamma_1, \dots, \gamma_d)$ such that (B.3) is verified, i.e., $\boldsymbol{\alpha}_{\mathscr{P},0}$ does not belong to S, which shows the claim. □

Lemma 4. *Let $\mathscr{P}$ be a multipath with M paths, and $\boldsymbol{x}_{\mathscr{P}}$ an admissible rate vector. If $k < K$ links $l_1, \dots, l_k$ that use k different technologies are such that the corresponding multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l_i}$ for $1 \le i \le k$ are linearly independent and verify $\boldsymbol{\alpha}_{\mathscr{P},l_i} \cdot \boldsymbol{x}_{\mathscr{P}} = 1$ for $1 \le i \le k$ (i.e., the k links are saturated), then the multipath-impact vector $\boldsymbol{\alpha}_{\mathscr{P},l_{k+1}}$ of link l_{k+1} using a technology different of the k others is either linearly independent with $\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_k}$, or it verifies $\boldsymbol{\alpha}_{\mathscr{P},l_{k+1}} \cdot \boldsymbol{x}_{\mathscr{P}} \neq 1$.*

Proof. Let us assume that neither of the two is verified, i.e., $\boldsymbol{\alpha}_{\mathscr{P},l_{k+1}} \cdot \boldsymbol{x}_P = 1$ and there are $(\gamma_1, \dots, \gamma_k)$ such that $\boldsymbol{\alpha}_{\mathscr{P},l_{k+1}} = \sum_{i=1}^k \gamma_i \boldsymbol{\alpha}_{\mathscr{P},l_i}$. Because for all $i \in \{1, \dots, k+1\}$, $\boldsymbol{\alpha}_{\mathscr{P},l_i} \cdot \boldsymbol{x}_{\mathscr{P}} = 1$, we have that $\sum_{i=1}^k \gamma_i = 1$, i.e., $\boldsymbol{\alpha}_{\mathscr{P},l_{k+1}}$ belongs to the affine subspace of dimension $d = k-1$ spanned by the multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l_i}$ for $1 \le i \le k$. But because they use different technologies, all the multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_{k+1}}$ are necessarily link-independent which, using Lemma 3, means that all the vectors $\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_k}$ must have $M - k + 1$ common zero-components, i.e., $\dim \ker(\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_k}) \ge M - k + 1$. Because the dimension of the space is M, the inequality is in contradiction with the fact that the vectors $\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_k}$ are linearly independent, i.e., $\operatorname{rank}(\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_k}) = k$, which shows the result. □

In particular, Lemma 4 shows that if we have $k = M$ links $l_1, \dots, l_M$ of different technologies (the $\boldsymbol{\alpha}_{\mathscr{P},l_i}$ are thus link-independent) that all are saturated, i.e., if $\boldsymbol{\alpha}_{\mathscr{P},l_i} \cdot \boldsymbol{x}_{\mathscr{P}} = 1$ for $1 \le i \le M$, then they are necessarily linearly independent, which means that they form a basis of $\mathbb{R}^M$. Consequently, for any other link l_{M+1} using another technology ($\boldsymbol{\alpha}_{\mathscr{P},l_{M+1}}$ is link-independent with the other multipath-impact vectors $\boldsymbol{\alpha}_{\mathscr{P},l_i}$), $\boldsymbol{\alpha}_{\mathscr{P},l_{M+1}}$ cannot be linearly independent with $\boldsymbol{\alpha}_{\mathscr{P},l_1}, \dots, \boldsymbol{\alpha}_{\mathscr{P},l_M}$, thus it must verify $\boldsymbol{\alpha}_{\mathscr{P},l_{M+1}} \cdot \boldsymbol{x}_{\mathscr{P}} \neq 1$. This proves Lemma 2. □

Proof of Theorem 2. Let $\mathscr{P}$ be a multipath with $M < K$ paths, i.e., $M = K - k$ for some $k \in \{1, \dots, K-1\}$. From Lemma 2, any rate vector $\boldsymbol{x}_{\mathscr{P}}$ saturates at most $K - k$ technologies. We therefore construct a K-path multipath that is strictly better by adding k paths using each only one of the (at least) k technologies that are not saturated (these paths exist because the network is multi-connected). Therefore, $M^{\text{opt}} \ge K$. □

B.2 Proof of Section 4.4

Proof of Theorem 3. We need to prove that each arm is probed an infinite number of times almost surely, which has two consequences. First, the strong law of large numbers then yields that the estimation of the optimal rate converges to the true value, and thus that HyMAB finds the best multipath. Second, this means that the probing probability $\epsilon_{\mathscr{P}}(t)$ given in Theorem 3 goes to zero and thus that HyMAB ends up exploiting the best multipath almost surely. Therefore, this shows that HyMAB converges to achieving optimal throughput. In the following, we show that probing each arm an infinite number of times is equivalent to having each arm chosen an infinite number of times at Stage 1, and that this is true almost surely.

For any arm $\mathscr{P}$, let $T_{\mathscr{P}}(t)$ be the number of times $\mathscr{P}$ is chosen at Stage 1 of the algorithm during the first t trials, and $n_{\mathscr{P}}(t)$ be the number of times the arm is probed, i.e., it is chosen at Stage 1 and probing is chosen at Stage 2. The best arm, i.e., the one maximizing $\|\boldsymbol{x}_{\mathscr{P}}\|_1^{\text{opt}}$, is denoted by $\mathscr{P}^*$. First, $n_{\mathscr{P}}(t)$ is unbounded almost surely if and only if $T_{\mathscr{P}}(t)$ is unbounded almost surely: If $n_{\mathscr{P}}(t)$ is unbounded, $T_{\mathscr{P}}(t)$ is obviously unbounded because $T_{\mathscr{P}}(t) \geq n_{\mathscr{P}}(t)$. If $T_{\mathscr{P}}(t)$ is unbounded, a probing phase will happen eventually almost surely because the probing probability is always strictly positive, which means that $n_{\mathscr{P}}(t)$ is unbounded almost surely.

Let us now assume that there is an arm $\mathscr{P}_0$ for which $T_{\mathscr{P}_0}(t)$ is bounded with non-zero probability. It means that with non-zero probability, $\mathscr{P}_0$ is not chosen anymore after some time, which means that with non-zero probability, at each time t for t large enough, there is an arm $\mathscr{P}(t)$ for which $T_{\mathscr{P}(t)}(t)$ is unbounded almost surely and such that $V_{\mathscr{P}(t)}(t) \geq V_{\mathscr{P}_0}(t)$. Using (4.2), this means that with non-zero probability, the following is true for each t for t large enough:

$$\left\|\overline{\boldsymbol{x}}_{\mathscr{P}(t),n_{\mathscr{P}(t)}(t-1)}\right\|_1 - \left\|\overline{\boldsymbol{x}}_{\mathscr{P}_0,n_{\mathscr{P}_0}(t-1)}\right\|_1 \geq \sqrt{2\ln t - 1}\left(\frac{1}{\sqrt{n_{\mathscr{P}_0}(t-1)}} - \frac{1}{\sqrt{n_{\mathscr{P}(t)}(t-1)}}\right).$$

But because for each estimation $\boldsymbol{x}_{\mathscr{P}}(t)$, $\|\boldsymbol{x}_{\mathscr{P}}(t)\|_1 \in [0,1]$, the left-hand side is finite; whereas the right-hand side is positive and goes to infinity: $n_{\mathscr{P}_0}(t)$ is bounded whereas $n_{\mathscr{P}(t)}(t)$ goes to infinity, i.e., the difference is strictly positive for t large enough. This is a contradiction, consequently, for any arm $\mathscr{P}$, $T_{\mathscr{P}}(t)$ is unbounded almost surely, and hence, $n_{\mathscr{P}}(t)$ is unbounded almost surely.

We now denote by $\boldsymbol{x}^r(t)$ the rate vector that is sent after t trials. Because at each trial, the source sends at most the current best estimate, we have

$$\mathbb{E}\left[\boldsymbol{x}^r(t)\right] \leq \mathbb{E}\left[\overline{\boldsymbol{x}}_{\mathscr{P}_t^*,n_{\mathscr{P}_t^*}(t)}\right]. \tag{B.6}$$

Each time exploitation is chosen at Stage 2, whatever the arm $\mathscr{P}_t$ chosen at Stage 1, the source sends the current best estimate, we thus have $\boldsymbol{x}^r(t) \geq \overline{\boldsymbol{x}}_{\mathscr{P}_t^*,n_{\mathscr{P}_t^*}(t)}\mathbf{1}_{\{\text{exploit at } t\}}$. The inequality is because when probing, the source still sends traffic at non-zero rate. The current rate estimation $\overline{\boldsymbol{x}}_{\mathscr{P}_t^*,n_{\mathscr{P}_t^*}(t)}$ at time t does not depend on the choice made at Stage 2 at time t, we consequently have

$$\mathbb{E}\left[\boldsymbol{x}^r(t)\right] \geq \mathbb{E}\left[\overline{\boldsymbol{x}}_{\mathscr{P}_t^*, n_{\mathscr{P}_t^*}(t)}\right] \mathbb{P}\left[\text{exploit at } t\right]. \tag{B.7}$$

But we have $\mathbb{P}\left[\text{exploit at } t\right] = 1 - \mathbb{P}\left[\text{explore } \mathscr{P}_t \text{ at } t\right]$. We know that for any $\mathscr{P}$, $n_{\mathscr{P}}(t)$ goes to infinity almost surely, consequently for any $\mathscr{P}$ and any $\lambda > 0$, $\mathbb{P}\left[\text{explore } \mathscr{P} \text{ at } t\right] = 1/n_{\mathscr{P}}^{\lambda}(t)$ goes to 0 almost surely. In particular, $\mathbb{P}\left[\text{explore } \mathscr{P}_t \text{ at } t\right]$ goes to 0, i.e.,

$$\mathbb{P}\left[\text{exploit at } t\right] \underset{t\to\infty}{\longrightarrow_{\text{a.s.}}} 1. \tag{B.8}$$

Using (B.6), (B.7), and (B.8), we have

$$\mathbb{E}\left[\boldsymbol{x}^r(t)\right] \sim \mathbb{E}\left[\overline{\boldsymbol{x}}_{\mathscr{P}_t^*, n_{\mathscr{P}_t^*}(t)}\right]. \tag{B.9}$$

The almost sure convergence of $n_{\mathscr{P}}(t)$ to infinity and the strong law of large numbers ensures that for any $\mathscr{P}$,

$$\overline{\boldsymbol{x}}_{P, n_{\mathscr{P}}(t)} \underset{t\to\infty}{\longrightarrow_{\text{a.s.}}} \boldsymbol{x}_P^{\text{opt}}. \tag{B.10}$$

In particular,

$$\overline{\boldsymbol{x}}_{\mathscr{P}^*, n_{\mathscr{P}^*}(t)} \underset{t\to\infty}{\longrightarrow_{\text{a.s.}}} \boldsymbol{x}_{\mathscr{P}^*}^{\text{opt}}. \tag{B.11}$$

Because $\mathscr{P}^*$ is defined as the arm maximizing $\|\boldsymbol{x}_{\mathscr{P}}\|_1^{\text{opt}}$, Equation (B.10) also means that for t large enough, $\mathscr{P}^*$ will be the arm maximizing the rate estimation $\left\|\overline{\boldsymbol{x}}_{\mathscr{P}, n_{\mathscr{P}}(t)}\right\|_1$, which means that

$$\mathbb{P}\left[\mathscr{P}_t^* = P^*\right] \underset{t\to\infty}{\longrightarrow_{\text{a.s.}}} 1. \tag{B.12}$$

Equations (B.9), (B.11), and (B.12) then ensure that

$$\mathbb{E}\left[\boldsymbol{x}^r(t)\right] \underset{t\to\infty}{\longrightarrow_{\text{a.s.}}} \boldsymbol{x}_{\mathscr{P}^*}^{\text{opt}},$$

which completes the proof. □

B.3 Proofs of Section 5.3.1

We study which queue has the largest average delays, i.e., the sign of $D_1 - D_2$. From Little's law, we know that working with the average delays D_i and with the average queue size N_i is equivalent, because $N_i = \lambda D_i$, hence, $D_1 - D_2$ and $N_1 - N_2$ have same sign. The average queue size of a queue with heterogeneous service rates is given by Yechiali and Naor [YN71]. After some manipulations, it can be rewritten for Queue i for $i \in \{1,2\}$ and for any $\lambda < \hat{\mu}_i$ as

$$N_i(\lambda) = \frac{\lambda}{\hat{\mu}_i - \lambda} + \frac{\lambda}{\alpha_i(\hat{\mu}_i - \lambda)} f_i(\lambda) V_i, \tag{B.13}$$

where we use the notations of Section 5.3.1 and where

$$f_i(\lambda) = \frac{1 - z_i(\lambda)}{\hat{\mu}_i - \lambda z_i(\lambda)}$$

with $z_i(\lambda)$ defined as the only root in (0,1) of

$$g_i(z) = \alpha_i z(\hat{\mu}_i - \lambda z) - (1-z)(\mu_{h,i} - \lambda z)(\mu_{l,i} - \lambda z). \tag{B.14}$$

We also define $\sigma_i = \mu_{h,i} + \mu_{l,i}$ and

$$M = \frac{\pi_2 - \pi_1}{\sigma_2 - \sigma_1}. \tag{B.15}$$

We start with useful lemmas. Remember that $\alpha_1 = \alpha_2 = \alpha$ and $\hat{\mu}_1 = \hat{\mu}_2 = \hat{\mu}$.

Lemma 5. *$z_i(\lambda)$ is a decreasing function of λ in $(0, \hat{\mu})$.*

Proof. z_i is the only root of $g_i(z)$ in (0,1). We write $g_i(z, \lambda)$ for the function g_i, making its dependency on λ explicit. Let $\epsilon > 0$. After some manipulations using $g_i(z_i, \lambda) = 0$, we have $g_i(z_i, \lambda + \epsilon) = \epsilon z_i k_i(z, \epsilon)$, where

$$k_i(z, \epsilon) = \sigma_i - (\alpha + \epsilon + 2\lambda + \sigma_i) z_i + (\epsilon + 2\lambda) z_i^2.$$

If $k_i(z_i, 0) > 0$, then for $\epsilon > 0$ small enough, $k_i(z_i, \epsilon) > 0$, and $g_i(z_i, \lambda + \epsilon) = \epsilon z_i k(z_i, \epsilon) > 0$, i.e., increasing λ increases g_i around z_i, and thus increasing λ decreases the root z_i, because g_i is increasing near z_i as $g_i(0) = -\pi_i < 0$ and $g_i(1) = \alpha(\hat{\mu} - \lambda) > 0$. Now, $k_i(z_i, 0) = (1 - z_i)(\sigma_i - 2\lambda z_i) - \alpha z_i$, which for $z_i \in (0,1)$ has same sign as

$$\begin{aligned} Q_i \doteq \frac{k(z_i, 0)}{1 - z_i} &= (\sigma_i - 2\lambda z_i) - \frac{\alpha z_i}{1 - z_i} \\ &= (\sigma_i - 2\lambda z_i) - \frac{(\mu_{h,i} - \lambda z_i)(\mu_{l,i} - \lambda z_i)}{\hat{\mu} - \lambda z_i}, \end{aligned}$$

where for the last equality we used (B.14) and $g_i(z_i, \lambda) = 0$. If $\mu_{h,i} = \mu_{l,i}$, then $\mu_{h,i} = \hat{\mu}$ and we have $Q_i = \hat{\mu} - \lambda z_i > 0$. Otherwise, necessarily $\mu_{h,i} > \hat{\mu}$ and $\mu_{l,i} < \hat{\mu}$. Then

$$\begin{aligned} Q_i &= (\mu_{h,i} - \lambda z_i) + (\mu_{l,i} - \lambda z_i) - \frac{(\mu_{h,i} - \lambda z_i)(\mu_{l,i} - \lambda z_i)}{\hat{\mu} - \lambda z_i} \\ &= (\mu_{h,i} - \lambda z_i)\left(1 - \frac{\mu_{l,i} - \lambda z_i}{\hat{\mu} - \lambda z_i}\right) + (\mu_{l,i} - \lambda z_i). \end{aligned}$$

Using (B.14) again, we know that $\mu_{l,i} - \lambda z_i > 0$. Also, $\mu_{l,i} < \hat{\mu}$, consequently $k(z_i, 0) > 0$, which concludes the proof. □

Lemma 6. *If $\sigma_1 = \sigma_2$ and $\pi_1 = \pi_2$, then $g_1 = g_2$. If $\sigma_1 = \sigma_2$ and $\pi_1 \neq \pi_2$, the only root of $g_1(z) - g_2(z)$ is 1. If $\sigma_1 \neq \sigma_2$, the only roots of $g_1(z) - g_2(z)$ are 1 and M/λ.*

Proof. This follows directly from (B.14). □

Lemma 7. *If there is no root of $g_1(z) - g_2(z)$ in (0,1), then $z_1 > z_2$ if and only if either $\pi_1 > \pi_2$, or $\pi_1 = \pi_2$ and $\sigma_1 < \sigma_2$.*

Proof. If $g_1(z) - g_2(z)$ has no root in (0,1) and because, g_i is increasing near z_i, we have $z_1 > z_2$ if and only if $\forall z \in (0,1)$, $g_1(z) < g_2(z)$. If $\pi_1 \neq \pi_2$, $g_1(z) < g_2(z)$ if and only if $\pi_1 > \pi_2$, because $g_i(0) = -\pi_i$. If $\pi_1 = \pi_2$, we have $g_1 - g_2 = \lambda z(1-z)(\sigma_1 - \sigma_2)$, i.e., $g_1(z) < g_2(z)$ if and only if $\sigma_1 < \sigma_2$. □

Lemma 8. *For all $\lambda \in (0, \hat{\mu})$, $f_1(\lambda) < f_2(\lambda)$ if and only if $z_1 > z_2$.*

Proof. Easy computation with $\lambda < \hat{\mu}$ and $0 < z_i < 1$. □

Lemma 9. $(V_1 - V_2)(\pi_1 - \pi_2) < 0$ *if and only if either $\sigma_1 = \sigma_2$, or $\hat{\mu}/M < 1$. Also, $V_1 = V_2$ if and only if either $M = \hat{\mu}$, or $\sigma_1 = \sigma_2$ and $\pi_1 = \pi_2$.*

Proof. After some manipulations, we can rewrite

$$V_i = \hat{\mu}\sigma_i - \pi_i - \hat{\mu}^2, \tag{B.16}$$

and the cases of equality become clear. It is also clear with simple manipulations that if $V_1 < V_2$ and $\pi_1 > \pi_2$, or $V_1 > V_2$ and $\pi_1 < \pi_2$, then $\hat{\mu}/M < 1$.

Reciprocally, if $\hat{\mu}/M < 1$, then either $\hat{\mu} < M$ and $M > 0$, or $\hat{\mu} > M$ and $M < 0$. In the first case, we have

$$\hat{\mu} < \frac{\pi_2 - \pi_1}{\sigma_2 - \sigma_1}. \tag{B.17}$$

Either $\sigma_1 > \sigma_2$ and we have $\pi_1 > \pi_2$ (because $M > 0$) and $V_1 < V_2$ (because of (B.17)) and thus $(V_1 - V_2)(\pi_1 - \pi_2) < 0$; or $\sigma_1 < \sigma_2$ and we have similarly $\pi_1 < \pi_2$ and $V_1 > V_2$. Similar manipulations show the result in the second case. □

Lemma 10. *If $V_1 \neq V_2$, there exists a $\lambda_m \in [0, \hat{\mu})$ such that for all $\lambda \in (\lambda_m, \hat{\mu})$, $N_1(\lambda) > N_2(\lambda)$ if and only if $V_1 > V_2$.*

Proof. We have

$$\frac{f_1(\hat{\mu})}{f_2(\hat{\mu})} = 1, \tag{B.18}$$

consequently there is a $\lambda_m \in [0, \hat{\mu})$ such that for all $\lambda \in (\lambda_m, \hat{\mu})$, $\left|1 - \frac{f_1(\lambda)}{f_2(\lambda)}\right| < \left|\frac{V_1 - V_2}{2}\right|$, which shows that for $\lambda \in (\lambda_m, \hat{\mu})$, $N_1(\lambda) - N_2(\lambda)$ has same sign as $V_1 - V_2$. □

Lemma 11. *Let $\beta_i = \alpha_{l,i}/\alpha$. If $\beta_1 = \beta_2 \doteq \beta$ and if $\beta\alpha_{h,i} \geq (1-\beta)\alpha_{l,i}$, then $V_1 \geq V_2$ if and only if $\pi_1 \leq \pi_2$.*

Proof. Let $v_i = \mu_{h,i} - \mu_{l,i}$. We have after easy computations $V_i = \beta(1-\beta)v_i^2$, thus $V_1 > V_2$ if and only if $v_1 > v_2$. After computations, we get $\pi_i = \hat{\mu}^2 + (1-2\beta)\hat{\mu}v_i - \beta(1-\beta)v_i^2$, hence

$$\pi_1 - \pi_2 = -(v_1 - v_2)(\beta(1-\beta)(v_1 + v_2) - (1-2\beta)\hat{\mu}).$$

If

$$\beta(1-\beta)(v_1 + v_2) \geq (1-2\beta)\hat{\mu}, \tag{B.19}$$

then $V_1 \geq V_2$ if and only if $\pi_1 \leq \pi_2$. After simplifications, we see that (B.19) is equivalent to $\beta(\mu_{h,1} + \mu_{h,2}) > \hat{\mu}$. This is true if, for $i \in \{1,2\}$, $\beta\alpha_{h,i} \geq (1-\beta)\alpha_{l,i}$, which concludes the proof. □

Lemma 12. *If*

$$(V_1 - V_2)(\pi_1 - \pi_2) > 0. \tag{B.20}$$

and

$$\left|1 - \frac{V_2}{V_1}\right| < \left|1 - \frac{\pi_2 + \alpha\hat{\mu}}{\pi_1 + \alpha\hat{\mu}}\right| \doteq B_0, \tag{B.21}$$

then $N_1(\lambda) - N_2(\lambda)$ changes sign in $(0, \hat{\mu})$, and there is a $\lambda_0 \in (0, \hat{\mu})$ such that for all $\lambda < \lambda_0$, $V_1 > V_2$ and $N_1(\lambda) < N_2(\lambda)$, or $V_1 < V_2$ and $N_1(\lambda) > N_2(\lambda)$.

Proof. A simple computation using (B.14) with $\lambda = 0$ shows that

$$z_i(0) = \frac{\pi_i}{\pi_i + \alpha\hat{\mu}} \text{ and } \frac{f_1(0)}{f_2(0)} = \frac{\pi_2 + \alpha\hat{\mu}}{\pi_1 + \alpha\hat{\mu}}.$$

So, when λ is close to 0, the difference $N_1(\lambda) - N_2(\lambda)$ has same sign as $f_1(0)V_1 - f_2(0)V_2$, i.e., it has same sign as

$$Q \doteq \frac{f_1(0)}{f_2(0)} - \frac{V_2}{V_1} = \frac{\pi_2 + \alpha\hat{\mu}}{\pi_1 + \alpha\hat{\mu}} - \frac{V_2}{V_1}.$$

If (B.20) and (B.21) hold, $Q > 0$ if and only if $V_1 < V_2$: When λ is close to 0, $N_1(\lambda) > N_2(\lambda)$ if and only if $V_1 < V_2$. From Lemma 10, we know that when λ is close to $\hat{\mu}$, $N_1(\lambda) > N_2(\lambda)$ if and only if $V_1 > V_2$, which means that $N_1 - N_2$ changes sign in $(0, \hat{\mu})$. In addition, if λ_0 is the first $\lambda \in (0, \hat{\mu})$ where $N_1 - N_2$ changes sign, then for all $\lambda \in (0, \lambda_0)$, $V_1 > V_2$ and $N_1(\lambda) < N_2(\lambda)$, or $V_1 < V_2$ and $N_1(\lambda) > N_2(\lambda)$. □

We can now prove the theorems.

Proof of Theorem 4. From (B.13), we write

$$N_1(\lambda) - N_2(\lambda) = \frac{\lambda}{\alpha(\hat{\mu} - \lambda)} \big(f_1(\lambda) V_1 - f_2(\lambda) V_2\big), \tag{B.22}$$

If $V_1 = V_2$ then from Lemma 9, $M = \hat{\mu}$, or $\sigma_1 = \sigma_2$ and $\pi_1 = \pi_2$. If $\pi_1 = \pi_2$, then $\sigma_1 = \sigma_2$ and thus $g_1 = g_2$ and $N_1 = N_2$. Otherwise, $M = \hat{\mu}$ and because $\lambda < \hat{\mu}$, Lemma 6 shows that there is no root of $g_1 - g_2$ in (0,1), and thus Lemma 7 and 8 along with (B.22) show that $N_1 > N_2$ if and only if $\pi_1 < \pi_2$.

If $\pi_1 = \pi_2$, then $V_1 = V_2$ is equivalent to $\sigma_1 = \sigma_2$ because of (B.16). If $\sigma_1 = \sigma_2$, we have proved $N_1 = N_2$. If $V_1 \neq V_2$, then $\sigma_1 \neq \sigma_2$ and $M = 0$, consequently there is no root of $g_1 - g_2$ in (0,1) (Lemma 6), and thus from Lemmas 7 and 8, $f_1(\lambda) > f_2(\lambda)$ for all $\lambda \in (0, \hat{\mu})$ if and only if $\sigma_1 > \sigma_2$, which happens if and only if $V_1 > V_2$. Thus $N_1 > N_2$ if and only if $V_1 > V_2$. □

Proof of Theorem 5. From Lemma 9, $(V_1 - V_2)(\pi_1 - \pi_2) < 0$ if and only if $\sigma_1 = \sigma_2$ or $\hat{\mu}/M < 1$. In both cases, Lemma 6 shows that there is no root of $g_1 - g_2$ in (0,1), and consequently, from Lemmas 7 and 8, $f_1(\lambda) > f_2(\lambda)$ for all $\lambda \in (0, \hat{\mu})$ if and only if $\pi_1 < \pi_2$, which happens if and only if $V_1 > V_2$. Thus for all $\lambda \in (0, \hat{\mu})$, $N_1(\lambda) > N_2(\lambda)$ if and only if $V_1 > V_2$. □

Proof of Theorem 6. Theorem 6 follows directly from Lemma 12. □

Proof of Theorem 7. Theorem 7 follows directly from Lemma 10. □

B.3.1 Additional Remarks on the Bounds

We can prove the following sufficient condition for having $N_1(\lambda) \neq N_2(\lambda)$ for all arrival rate λ, and consequently (because of Theorem 7), for having $N_1(\lambda) > N_2(\lambda)$ if and only if $V_1 \geq V_2$.

Lemma 13. *If (B.20) is verified and*

$$\frac{\max(V_1, V_2)}{\min(V_1, V_2)} > 1 + \frac{\max(\pi_1, \pi_2)}{\alpha\hat{\mu}}, \tag{B.23}$$

then for all $\lambda \in (0, \hat{\mu})$, $N_1(\lambda) \neq N_2(\lambda)$.

Proof. With simple manipulations, it is easy to see that (B.23) is equivalent to the inequality $\frac{f_1(0)}{f_2(\hat{\mu})} \leq \frac{V_2}{V_1} \leq \frac{f_1(\hat{\mu})}{f_2(0)}$ being false. Lemmas 5 and 8 prove that f_i is an increasing function of λ, (B.23) consequently implies that for all $\lambda \in (0, \hat{\mu})$, $\left|1 - \frac{V_2}{V_1}\right| > \left|1 - \frac{f_1(\lambda)}{f_2(\lambda)}\right|$, i.e., $N_1(\lambda) - N_2(\lambda)$ always has same sign as $V_1 - V_2$. $\square$

When (B.20) is verified but neither (B.21) nor (B.23) are verfified, then numerical results show that both cases can happen: Either for all $\lambda \in (0, \hat{\mu})$, $N_1(\lambda) \neq N_2(\lambda)$ (and in that case, $N_1(\lambda) > N_2(\lambda)$ if and only if $V_1 \geq V_2$); or $N_1(\lambda) - N_2(\lambda)$ changes sign. In that case, $N_1 - N_2$ changes sign at least twice in $(0, \hat{\mu})$. One open question is to determine the precise bound B as a function of π_i, α, and $\hat{\mu}$, such that $N_1(\lambda) \neq N_2(\lambda)$ for all $\lambda \in (0, \hat{\mu})$ if and only if $\left|1 - \frac{V_2}{V_1}\right| > B$. For that, one needs to find the extrema of $f_1(\lambda)/f_2(\lambda)$ in $(0, \hat{\mu})$. We conjecture that B is close to B_0 defined in (B.21). In fact, numerically, it seems that $\left|1 - \frac{V_2}{V_1}\right| > 2B_0$ already ensures that for all $\lambda \in (0, \hat{\mu})$, $N_1(\lambda) \neq N_2(\lambda)$.

Bibliography

[ABL05] Mansoor Alicherry, Randeep Bhatia, and Li Erran Li. Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[ACBF02] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 2002.

[ACBFS95] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert Schapire. Gambling in a Rigged Casino: The Adversarial Multi-armed Bandit Problem. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995.

[AG16] Kota Abe and Shigeki Goto. Fingerprinting Attack on Tor Anonymity using Deep Learning. *Proceedings of the Asia-Pacific Advanced Network*, 2016.

[AK03] Ivo Adan and Vidyadhar Kulkarni. Single-Server Queue with Markov-Dependent Inter-Arrival and Service Times. *Queueing Systems*, 2003.

[AK04] Baruch Awerbuch and Robert Kleinberg. Adaptive Routing with End-to-End Feedback: Distributed Learning and Geometric Approaches. In *ACM Symposium on Theory of Computing (STOC)*, 2004.

[AK08] Baruch Awerbuch and Robert Kleinberg. Online Linear Optimization and Adaptive Routing. *Journal of Computer and System Sciences*, 2008.

[AKK04] Jamal Al-Karaki and Ahmed Kamal. Routing Techniques in Wireless Sensor Networks: a Survey. *IEEE Wireless Communications*, 2004.

[AMS$^+$03] Aditya Akella, Bruce Maggs, Srinivasan Seshan, Anees Shaikh, and Ramesh Sitaraman. A Measurement-Based Analysis of Multihoming. In *ACM SIGCOMM Conference*, 2003.

[ARW11] Osman T. Akgun, Rhonda Righter, and Ronald Wolff. Multiple-server System with Flexible Arrivals. *Advances in Applied Probability*, 2011.

[BCB12] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *arXiv:1204.5721*, 2012.

[BGZ14] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic Multi-Armed-Bandit Problem with Non-stationary Rewards. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[BL94] Justin Boyan and Michael Littman. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. In *Advances in Neural Information Processing Systems (NIPS)*, 1994.

[Bla79] George Blakley. Safeguarding Cryptographic Keys. In *National Computer Conference (NCC)*, 1979.

[BSC$^+$13] Mehdi Bennis, Meryem Simsek, Andreas Czylwik, Walid Saad, Stefan Valentin, and Merouane Debbah. When Cellular Meets WiFi in Wireless Small Cell Networks. *IEEE Communications Magazine*, 2013.

[CA98] Heyning Cheng and Ron Avnur. Traffic Analysis of SSL Encrypted Web Browsing. Technical report, 1998. https://pdfs.semanticscholar.org/1a98/7c4fe65fa347a863dece665955ee7e01791b.pdf.

[CCP90] Cheng-Shang Chang, XiuLi Chao, and Michael Pinedo. A Note on Queues with Bernoulli Routing. In *IEEE Conference on Decision and Control (CDC)*, 1990.

[CF08] Xavier Carcelle and Florian Fainelli. FAIFA: A First Open Source PLC Tool. https://events.ccc.de/congress/2008/Fahrplan/events/2901.en.html, 2008. [Online; accessed 13-September-2018].

[Che17] Brian Chen. Why Your Next Wi-Fi Setup Should Be a Mesh Network. https://www.nytimes.com/2017/04/26/technology/personaltech/mesh-network-vs-router.html, April 2017. [Online; accessed 13-September-2018].

[CHJ17] Giovanni Cherubin, Jamie Hayes, and Marc Juárez. Website Fingerprinting Defenses at the Application Layer. In *Privacy Enhancing Technologies Symposium (PETS)*, 2017.

[Cla56] Bruce A. Clarke. A Waiting Line Process of Markov Type. *The Annals of Mathematical Statistics*, 1956.

[CLCD06] Lijun Chen, Steven Low, Mung Chiang, and John C. Doyle. Cross-Layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks. In *IEEE INFOCOM*, 2006.

[CLG$^+$13] Yung-Chih Chen, Yeon-sup Lim, Richard J. Gibbens, Erich M. Nahum, Ramin Khalili, and Don Towsley. A Measurement-based Study of Multipath TCP Performance over Wireless Networks. In *ACM Internet Measurement Conference (IMC)*, 2013.

[CNJ14] Xiang Cai, Rishab Nithyanand, and Rob Johnson. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2014.

[CNW$^+$14] Xiang Cai, Rishab Nithyanand, Tao Wang, Rob Johnson, and Ian Goldberg. A Systematic Approach to Developing and Evaluating Website Fingerprinting Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, 2014.

[coa13] North America Coaxial Cable Market - Industry Analysis, Size, Share, Growth, Trends and Forecast 2012 - 2018. https://www.transparencymarketresearch.com/coaxial-cable-market.html, 2013. [Online; accessed 13-September-2018].

[CT14] Yung-Chih Chen and Don Towsley. On Bufferbloat and Delay Analysis of Multipath TCP in Wireless Networks. In *IFIP Networking Conference*, 2014.

[CWY13] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In *International Conference on Machine Learning (ICML)*, 2013.

[CZJJ12] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a Distance: Website Fingerprinting Attacks and Defenses. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[Dan10] George Danezis. Traffic Analysis of the HTTP Protocol over TLS, 2010.

[DCB17] Quentin De Coninck and Olivier Bonaventure. Multipath QUIC: Design and Evaluation. In *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2017.

[DCRS12] Kevin P Dyer, Scott E Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE Symposium on Security and Privacy*, 2012.

[Del18] John Delaney. The Best Wi-Fi Mesh Network Systems of 2018. https://uk.pcmag.com/amped-wireless-ally-plus-whole-home-smart-wi-fi-system/87178/guide/the-best-wi-fi-mesh-network-systems-of-2018, August 2018. [Online; accessed 13-September-2018].

[DFSS08] Luis DaCosta, Alvaro Fialho, Marc Schoenauer, and Michèle Sebag. Adaptive Operator Selection with Dynamic Multi-armed Bandits. In *ACM Genetic and Evolutionary Computation Conference (GECCO)*, 2008.

[DPZ04] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2004.

[DQZ+15] Ping Dong, Huanyan Qian, Kai Zhou, Weidi Lu, and Shaohua Lan. A Maximally Radio-disjoint Geographic Multipath Routing Protocol for MANET. *Annals of Telecommunications*, 2015.

[ECM+08] Pedro Miguel Esposito, Miguel Elias Campista, Igor Moraes, Luis Henrique Costa, Otto Carlos Duarte, and Marcelo Rubinstein. Implementing the Expected Transmission Time Metric for OLSR Wireless Mesh Networks. In *IFIP Wireless Days*, 2008.

[ES06] Atilla Eryilmaz and R. Srikant. Joint Congestion Control, Routing, and MAC for Stability and Fairness in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 2006.

[FEB+16] Alexander Frömmgen, Tobias Erbshäußer, Buchmann, Torsten Zimmermann, and Klaus Wehrle. ReMP TCP: Low Latency Multipath TCP. In *IEEE International Conference on Communications (ICC)*, 2016.

[FL16] Saman Feghhi and Douglas J Leith. A Web Traffic Analysis Attack Using Only Timing Information. *IEEE Transactions on Information Forensics and Security*, 2016.

[FRH+11] Alan Ford, Costin Raiciu, Mark Handley, Sébastien Barré, and Janardhan Iyengar. Architectural Guidelines for Multipath TCP Development. *RFC 6182*, 2011.

[FRHB13] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. *RFC 6824*, 2013.

[GC04] Gregor Gaertner and Vinny Cahill. Understanding Link Quality in 802.11 Mobile Ad Hoc Networks. *IEEE Internet Computing*, 2004.

[GGSE01] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-Resilient, Energy-Efficient Multipath routing in Wireless Sensor Networks. *ACM SIGMOBILE*, 2001.

[GHBSW+17] Kristen Gardner, Mor Harchol-Balter, Alan Scheller-Wolf, Mark Velednitsky, and Samuel Zbarsky. Redundancy-d: The Power of d Choices for Redundancy. *Operations Research*, 2017.

[GHBWY06] Varun Gupta, Mor Harchol-Balter, Alan Wolf, and Uri Yechiali. Fundamental Characteristics of Queues with Fluctuating Load. In *ACM Sigmetrics*, 2006.

[GKJ10] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Learning Multiuser Channel Allocations in Cognitive Radio Networks: A Combinatorial Multi-Armed Bandit Formulation. In *IEEE Symposium on New Frontiers in Dynamic Spectrum (DYSPAN)*, 2010.

[GKJ12] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial Network Optimization With Unknown Variables: Multi-Armed Bandits With Linear Rewards and Individual Observations. *IEEE/ACM Transactions on Networking*, 2012.

[GNT06] Leonidas Georgiadis, Michael Neely, and Leandros Tassiulas. Resource Allocation and Cross-layer Control in Wireless Networks. *Foundations and Trends in Networking*, 2006.

[GRS11] Juan Gálvez, Pedro Ruiz, and Antonio Skarmeta. Multipath Routing with Spatial Separation in Wireless Multi-hop Networks Without Location Information. *ACM Computer Networks*, 2011.

[GS08] Manolis Genetzakis and Vasilios Siris. A Contention-Aware Routing Metric for Multi-Rate Multi-Radio Mesh Networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2008.

[GSK04] Violeta Gambiroza, Bahareh Sadeghi, and Edward Knightly. End-to-end Performance and Fairness in Multihop Wireless Backhaul Networks. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2004.

[GSK08] Anastasios Giannoulis, Theodoros Salonidis, and Edward Knightly. Congestion Control and Channel Assignment in Multi-radio Wireless Mesh Networks. In *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2008.

[Hai58] Frank A. Haight. Two Queues in Parallel. *Biometrika*, 1958.

[HD16] Jamie Hayes and George Danezis. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique. In *USENIX Security Symposium*, 2016.

[Hey82] Daniel P. Heyman. On Ross's Conjectures about Queues with Non-stationary Poisson Arrivals. *Journal of Applied Probability*, 1982.

[HGAS$^+$18] Sébastien Henri, Ginés García Avilés, Pablo Serrano, Albert Banchs, and Patrick Thiran. Protecting against Website Fingerprinting with Multihoming. Technical report, 2018.

[Homa] HomePlug Powerline Alliance. http://www.homeplug.org. [Online; accessed 13-September-2018].

[Homb] HomePlug Technology Overview. http://www.homeplug.org/tech-resources/techoverview/. [Online; accessed 13-September-2018].

[Hor04] Neil Horman. Understanding and Programming With Netlink Sockets. https://people.redhat.com/nhorman/papers/netlink.pdf, 2004. [Online; accessed 13-September-2018].

[HST18] Sébastien Henri, Seva Shneer, and Patrick Thiran. On the Delays in Time-Varying Networks: Does Larger Service-Rate Variance Imply Larger Delays? In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2018.

[HT18] Sébastien Henri and Patrick Thiran. Optimal Number of Paths with Multipath Routing in Hybrid Networks. In *IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2018.

[HVHT16] Sébastien Henri, Christina Vlachou, Julien Herzen, and Patrick Thiran. EMPoWER Hybrid Networks: Exploiting Multiple Paths over Wireless and ElectRical Mediums. In *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2016.

[HVT18] Sébastien Henri, Christina Vlachou, and Patrick Thiran. Multi-armed Bandit in Action: Optimizing Performance in Dynamic Hybrid Networks. *IEEE/ACM Transactions on Networking*, 2018.

[HWF09] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *ACM Workshop on Cloud Computing Security (CCSW)*, 2009.

[Hyb13] IEEE 1905.1-2013: IEEE Standard for a Convergent Digital Home Network for Heterogeneous Technologies. https://standards.ieee.org/findstds/standard/1905.1-2013.html, 2013. [Online; accessed 13-September-2018].

[hyb18] Huawei's New Hybrid Routers Blend Mesh and Powerline Connectivity. https://www.engadget.com/2018/01/09/huawei-wifi-q2-hybrid-mesh-router-plc/, January 2018. [Online; accessed 13-September-2018].

[Hyy13] Esa Hyytiä. Optimal Routing of Fixed Size Jobs to Two Parallel Servers. *INFOR: Information Systems and Operational Research*, 2013.

[HZ04] Peter G. Harrison and Harf Zatschler. Sojourn Time Distributions in Modulated G-queues with Batch Processing. In *IEEE Conference on Quantitative Evaluation of Systems (QEST)*, 2004.

[IAS06] Janardhan Iyengar, Paul Amer, and Randall Stewart. Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 2006.

[IM14] Ane Izagirre and Armand Makowski. Light traffic performance under the power of two load balancing strategy: The case of server heterogeneity. *ACM Sigmetrics Performance Evaluation Review*, 2014.

[ISP07] ISPs Sell Clickstreams For $5 A Month. https://seekingalpha.com/article/29449-compete-ceo-isps-sell-clickstreams-for-5-a-month, 2007. [Online; accessed 13-September-2018].

[JAA+14] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A Critical Evaluation of Website Fingerprinting Attacks. In *ACM Conference on Computer and Communications Security (CCS)*, 2014.

[JIP+16] Marc Juárez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an Efficient Website Fingerprinting Defense. In *European Symposium on Research in Computer Security (ESORICS)*, 2016.

[JLWW13] Taeho Jung, Xiang-Yang Li, Zhiguo Wan, and Meng Wan. Privacy Preserving Cloud Data Access with Multi-Authorities. In *IEEE INFOCOM*, 2013.

[JPPQ05] Kamal Jain, Jitendra Padhye, Venkata Padmanabhan, and Lili Qiu. Impact of Interference on Multi-hop Wireless Network Performance. *Wireless Networks*, 2005.

[KMC+00] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click Modular Router. *ACM Transactions on Computer Systems*, 2000.

[KMT98] Frank Kelly, Aman Maulloo, and David Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 1998.

[KNJ14] Dileep Kalathil, Naumaan Nayyar, and Rahul Jain. Decentralized Learning for Multiplayer Multiarmed Bandits. *IEEE Transactions on Information Theory*, 2014.

[KR09] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-down Approach*. 2009.

[KTZ04] Pejman Khadivi, Terence Todd, and Dongmei Zhao. Handoff Trigger Nodes for Hybrid IEEE 802.11 WLAN/Cellular Networks. In *International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE)*, 2004.

[KV06] Pradeep Kyasanur and Nitin Vaidya. Routing and Link-layer Protocols for Multi-channel Multi-interface Ad-hoc Wireless Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2006.

[LAPJ15] Igor Lopez, Marina Aguado, Christian Pinedo, and Eduardo Jacob. SCADA Systems in the Railway Domain: Enhancing Reliability Through Redundant Multipath TCP. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2015.

[Lar81] Ronald Leroy Larsen. *Control of Multiple Exponential Servers with Application to Computer Systems*. PhD thesis, University of Maryland, 1981.

[LC02] Gyu Myoung Lee and Jin Seek Choi. A Survey of Multipath Routing for Traffic Engineering. *Information and Communications University, Korea*, 2002.

[LCL$^+$14] Jin Li, Xiaofeng Chen, Mingqiang Li, Jingwei Li, Patrick PC Lee, and Wenjing Lou. Secure Deduplication with Efficient and Reliable Convergent Key Management. *IEEE Transactions on Parallel and Distributed Systems*, 2014.

[LG01] Sung-Ju Lee and Mario Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad-hoc Networks. In *IEEE International Conference on Communications (ICC)*, 2001.

[LL99] Steven Low and David Lapsley. Optimization Flow Control: Basic Algorithm and Convergence. *IEEE/ACM Transactions on Networking*, 1999.

[LLZ06] Wenjing Lou, Wei Liu, and Yanchao Zhang. Performance Optimization using Multipath Routing in Mobile Ad Hoc and Wireless Sensor Networks. *Combinatorial Optimization in Communication Networks*, 2006.

[LR85] Tze Leung Lai and Herbert Robbins. Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 1985.

[LR07] Xiaojun Lin and Shahzada Rasool. A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-Hoc Wireless Networks. In *IEEE INFOCOM*, 2007.

[LS92] Liwan Liyanage and J. George Shanthikumar. Second-order Properties of Single-stage Queueing Systems. *Oxford Statistical Science Series*, 1992.

[LS04] Xiaojun Lin and Ness Shroff. Joint Rate Control and Scheduling in Multihop Wireless Networks. In *IEEE Conference on Decision and Control (CDC)*, 2004.

[LS06] Xiaojun Lin and Ness Shroff. Utility Maximization for Communication Networks with Multipath Routing. *IEEE Transactions on Automatic Control*, 2006.

[LVHB06] Joke Lambert, Benny Van Houdt, and Chris Blondia. Queues with Correlated Service and Inter-Arrival Times and Their Application to Optical Buffers. *Stochastic Models*, 2006.

[LYZ$^+$13] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[LZ10] Keqin Liu and Qing Zhao. Distributed Learning in Multi-armed Bandit with Multiple Players. *IEEE Transactions on Signal Processing*, 2010.

[MG05] Sai Rajesh Mahabhashyam and Natarajan Gautam. On Queues with Markov Modulated Service Rates. *Queueing Systems*, 2005.

[MHJT14] Brad Miller, Ling Huang, Anthony D. Joseph, and Doug Tygar. I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis. In *Privacy Enhancing Technologies Symposium (PETS)*, 2014.

[MoC] Multimedia over Coax Alliance (MoCA). http://www.mocalliance.org. [Online; accessed 13-September-2018].

[MPB77] C. R. Mitchell, Albert S. Paulson, and C. A. Beswick. The Effect of Correlated Exponential Service Times on Single Server Tandem Queues. *Naval Research Logistics*, 1977.

[MRW06] Hamed Mohsenian-Rad and Vincent Wong. Joint Optimal Channel Assignment and Congestion Control for Multi-channel Wireless Mesh Networks. In *IEEE International Conference on Communications (ICC)*, 2006.

[Neu77] Marcel F. Neuts. *The M/M/1 Queue with Randomly Varying Arrival and Service Rates*. PhD thesis, Delaware University, 1977.

[NML08] Michael Neely, Eytan Modiano, and Chih-Ping Li. Fairness and Optimal Stochastic Control for Heterogeneous Networks. *IEEE/ACM Transactions on Networking*, 2008.

[OJA$^+$17] Rebekah Overdorf, Mark Juarez, Gunes Acar, Rachel Greenstadt, and Claudia Diaz. How Unique is Your .onion?: An Analysis of the Fingerprintability of Tor Onion Services. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[ope] Networking in the Linux Kernel. https://openwrt.org/docs/guide-developer/networking/praxis. [Online; accessed 13-September-2018].

[PB] Christoph Paasch and Sébastien Barré. Multipath TCP in the Linux Kernel. https://www.multipath-tcp.org. [Online; accessed 13-September-2018].

[PD11] Parth Pathak and Rudra Dutta. A Survey of Network Design Problems and Joint Design Approaches in Wireless Mesh Networks. *IEEE Communications Surveys & Tutorials*, 2011.

[PDD$^+$12] Christoph Paasch, Gregory Detal, Fabien Duchêne, Costin Raiciu, and Olivier Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *ACM SIGCOMM Workshop on Cellular Networks*, 2012.

[Per] Mike Perry. Experimental Defense for Website Traffic Fingerprinting, Tor project Blog. https://blog.torproject.org/experimental-defense-website-traffic-fingerprinting. [Online; accessed 13-September-2018].

[Per15] Mike Perry. Padding Negotiation, Tor Proposal 254. https://github.com/torproject/torspec/blob/master/proposals/254-padding-negotiation.txt, 2015. [Online; accessed 13-September-2018].

[PFTK98] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *ACM SIGCOMM Conference*, 1998.

[Plc10] IEEE 1901-2010: IEEE Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications. https://standards.ieee.org/standard/1901-2010.html, 2010. [Online; accessed 13-September-2018].

[PLP$^+$16] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website Fingerprinting at Internet Scale. In *Network and Distributed System Security Symposium (NDSS)*, 2016.

[PNZE11] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2011.

[PTP08] IEEE 1588-2008: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. https://standards.ieee.org/standard/1588-2008.html, 2008. [Online; accessed 13-September-2018].

[RFP10] Benjamin Recht, Maryam Fazel, and Pablo Parrilo. Guaranteed Minimum-rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *SIAM review*, 2010.

[RGGK08] Božidar Radunović, Christos Gkantsidis, Dinan Gunawardena, and Peter Key. Horizon: Balancing TCP over Multiple Paths in Wireless Mesh Network. In *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2008.

[RH07] Maxim Raya and Jean-Pierre Hubaux. Securing Vehicular Ad-Hoc Networks. *Journal of Computer Security*, 2007.

[Rol81] Tomasz Rolski. Queues with Non-stationary Input Stream: Ross's Conjecture. *Advances in Applied Probability*, 1981.

[Ros78] Sheldon M. Ross. Average Delay in Queues with Non-stationary Poisson Arrivals. *Journal of Applied Probability*, 1978.

[Ros18] Dan Rosenbaum. A fine mesh we're in: A guide to Wi-Fi mesh networks. https://www.hpe.com/us/en/insights/articles/a-fine-mesh-were-in-a-guide-to-wi-fi-mesh-networks-1807.html, July 2018. [Online; accessed 13-September-2018].

[RPB$^+$12] Costin Raiciu, Christoph Paasch, Sébastien Barré, Alan Ford, Michio Honda, Fabien Duchêne, Olivier Bonaventure, and Mark Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2012.

[SB98] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. 1998.

[Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 1979.

[SM09] Yi Shi and Kanta Matsuura. Fingerprinting Attack on the Tor Anonymity System. In *International Conference on Information and Communications Security (ICICS)*, 2009.

[Sob01] João Luís Sobrinho. Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet. In *IEEE INFOCOM*, 2001.

[Sri04] R. Srikant. *The Mathematics of Internet Congestion Control*. 2004.

[SS13] Emil Stefanov and Elaine Shi. Multi-Cloud Oblivious Storage. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[Ste07] Randall Stewart. Stream Control Transmission Protocol. *RFC 4960*, 2007.

[SW06] Vitaly Shmatikov and Ming-Hsiu Wang. Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In *European Symposium on Research in Computer Security (ESORICS)*, 2006.

[SZ10] Chengqi Song and Qian Zhang. Cooperative Spectrum Sensing with Multi-Channel Coordination in Cognitive Radio Networks. In *IEEE International Conference on Communications (ICC)*, 2010.

[TE92] Leandros Tassiulas and Anthony Ephremides. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multi-hop Radio Networks. *IEEE Transactions on Automatic Control*, 1992.

[Tho33] William R. Thompson. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 1933.

[THT08] Jenn-Yue Teo, Yajun Ha, and Chen-Khong Tham. Interference-Minimized Multipath Routing with Congestion Control in Wireless Sensor Network for High-Rate Streaming. *IEEE Transactions on Mobile Computing*, 2008.

[TM06] Jack Tsai and Tim Moors. A review of Multipath Routing Protocols: From Wireless Ad Hoc to Mesh Networks. In *ACoRN ECR Workshop on Multihop Wireless Networks*, volume 30, 2006.

[Tor15] Tor: Inception. https://www.torproject.org/about/torusers.html.en, 2015. [Online; accessed 13-September-2018].

[TT07] Wai-Hong Tam and Yu-Chee Tseng. Joint Multi-Channel Link Layer and Multi-path Routing Design for Wireless Mesh Networks. In *IEEE INFOCOM*, 2007.

[TTAE09] Mohammed Tarique, Kemal Tepe, Sasan Adibi, and Shervin Erfani. Survey of Multipath Routing Protocols for Mobile Ad-hoc Networks. *Journal of Network and Computer Applications*, 2009.

[VHT13] Christina Vlachou, Julien Herzen, and Patrick Thiran. Fairness of MAC protocols: IEEE 1901 vs. 802.11. In *IEEE International Symposium on Power Line Communications and its Applications (ISPLC)*, 2013.

[VHT15] Christina Vlachou, Sébastien Henri, and Patrick Thiran. Electri-Fi Your Data: Measuring and Combining Power-Line Communications with WiFi. In *ACM Internet Measurement Conference (IMC)*, 2015.

[vis15] IMT-Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. *Recommendation ITU-R*, 2015.

[Vla16] Christina Vlachou. *Measuring, Modeling and Enhancing Power-Line Communications*. PhD thesis, EPFL, 2016.

[WCM09] Charles Wright, Scott Coull, and Fabian Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *Network and Distributed System Security Symposium (NDSS)*, 2009.

[WCN$^+$14] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium*, 2014.

[WCN$^+$17] Francesc Wilhelmi, Cristina Cano, Gergely Neu, Boris Bellalta, Anders Jonsson, and Sergio Barrachina-Muñoz. Collaborative Spatial Reuse in Wireless Networks via Selfish Multi-Armed Bandits. *arXiv:1710.11403*, 2017.

[WG13] Tao Wang and Ian Goldberg. Improved Website Fingerprinting on Tor. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2013.

[WG16] Tao Wang and Ian Goldberg. On Realistically Attacking Tor with Website Fingerprinting. In *Privacy Enhancing Technologies Symposium (PETS)*, 2016.

[WG17] Tao Wang and Ian Goldberg. Walkie-Talkie: An Efficient Defense Against Passive Website Fingerprinting Attacks. In *USENIX Security Symposium*, 2017.

[wif09] IEEE 802.11n-2009-Amendment 5: Enhancements for Higher Throughput. http://standards.ieee.org/findstds/standard/802.11n-2009.html, 2009. [Online; accessed 13-September-2018].

[wif13] IEEE 802.11ac-2013-Amendment 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz. https://standards.ieee.org/findstds/standard/802.11ac-2013.html, 2013. [Online; accessed 13-September-2018].

[wif15] Consumer Interest in Wired Solutions to Wireless Home Networking Problems: Quantitative Findings. http://www.mocalliance.org/news/Parks-Associates-Final-Report.pdf, 2015. [Online; accessed 13-September-2018].

[wif18] IEEE Draft Standard 802.11ax: Enhancements for High Efficiency WLAN. https://standards.ieee.org/develop/project/802.11ax.html, 2018. [Online; accessed 13-September-2018].

[Win77] Wayne Winston. Optimality of the Shortest Line Discipline. *Journal of Applied Probability*, 1977.

[WPL03] Wei-Hua Wang, Marimuthu Palaniswami, and Steven Low. Optimal Flow Control and Routing in Multi-path Networks. *Performance Evaluation*, 2003.

[WS96] David Wagner and Bruce Schneier. Analysis of the SSL 3.0 Protocol. In *USENIX Workshop on Electronic Commerce*, 1996.

[YFD$^+$16] Kiran Yedugundla, Simone Ferlin, Thomas Dreibholz, Özgü Alay, Nicolas Kuhn, Per Hurtig, and Anna Brunstrom. Is Multi-path Transport Suitable for Latency Sensitive Traffic? *Computer Networks*, 2016.

[YN71] Uri Yechiali and Pinhas Naor. Queuing Problems with Heterogeneous Arrivals and Service. *Operations Research*, 1971.

[YWK05] Yaling Yang, Jun Wang, and Robin Kravets. Interference-aware Load Balancing for Multihop Wireless Networks. Technical report, 2005.

[ZF11] Ye Zhu and Huirong Fu. Traffic Analysis Attacks on Skype VoIP Calls. *Computer Communications*, 2011.

[ZWT$^+$10] Liang Zhou, Xinbing Wang, Wei Tu, Gabriel-Miro Muntean, and Benoît Geller. Distributed Scheduling Scheme for Video Streaming over Multi-channel Multi-radio Multi-hop Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 2010.

Sébastien Henri

✉ *sebastien.henri@epfl.ch*
people.epfl.ch/shenri
linkedin.com/in/shenri

Education

2013 – 2018	**PhD**, *EPFL*, Lausanne, Switzerland Thesis: Improving Throughput, Latency and Security with Hybrid Networks and Multipath Routing
2008 – 2013	**Engineer's degrees (*Diplôme d'ingénieur*)**, *École polytechnique and Télécom ParisTech*, France Majors in Computer Science and Networking ◦ *Diplôme d'ingénieur de l'École polytechnique* obtained in 2011 ◦ *Diplôme d'ingénieur de Télécom ParisTech* obtained in 2013

Professional Experience

Sept. 2013 – Nov. 2018 **Research and Teaching Assistant (PhD candidate)**, *EPFL*, Lausanne, Switzerland

Surpervised by Prof. Patrick Thiran

Performance of Hybrid PLC/WiFi Networks

- Research on hybrid networks
  - performance evaluation, modeling, protocol design and implementation in local high-throughput hybrid networks, in particular with powerline communication (PLC) and WiFi [1, 2, 5]
  - analysis of delays in hybrid networks with varying throughputs [4]
  - multipath routing and multipath congestion-control [2, 3, 5]
  - machine learning techniques applied to networking (on-going)
  - security of PLC and hybrid networks (on-going)
- Developed and maintained a PLC/WiFi testbed of 22 OpenWrt APU routers, used for experimental results
- Teaching assistant for mathematics and computer science courses (Bachelor and Master)
- Supervised research projects of ∼10 bachelor and master students

Mar. - Aug. 2013 **Engineer**, *Qualcomm*, Paris, France

Time Synchronization in Ad-hoc Networks

Protocol design and simulation for time synchronization in ad-hoc LTE-Direct networks

Aug. 2012 - Jan. 2013 **Engineer (internship)**, *Qualcomm*, Bridgewater, NJ, USA

Direct Device-to-Device Communications and Proximity-Aware Networking

- Modeling and simulation of very-high-throughput WiFi (802.11ac and 802.11ad) in dense local networks
- Protocol design and simulation for time synchronization in ad-hoc LTE-Direct networks

Sep. 2011 - June 2012 **Engineer**, *Technicolor*, Paris, France

Development of Network Tomography Modules

Development of C/C++ and Java modules for computing link loss-rates in WANs (part-time)

Apr. - June 2011 **Engineer (internship)**, *Safran Electronics & Defense (formerly Sagem Défense Sécurité)*, Massy, France

Study and Integration of Quality of Service Protocols Using a Network Emulation Environment

Integration of IntServ and DiffServ protocols in a network emulation environment, as part of a team of four engineers

Publications

Published articles

[1] C. Vlachou, S. Henri and P. Thiran, “Electri-Fi Your Data: Measuring and Combining PLC with WiFi,” in *ACM Internet Measurement Conference (IMC)*, 2015, doi: 10.1145/2815675.2815689.

[2] S. Henri, C. Vlachou, J. Herzen and P. Thiran, “EMPoWER Hybrid Networks: Exploiting Multiple Paths over Wireless and ElectRical Mediums,” in *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2016, doi: 10.1145/2999572.2999574.

[3] S. Henri and P. Thiran, “Optimal Number of Paths with Multipath Routing in Hybrid Networks,” in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2018, short paper.

[4] S. Henri, S. Shneer and P. Thiran, “On the Delays in Time-Varying Networks: Does Larger Service-Rate Variance Imply Larger Delays?” in *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2018, doi: 10.1145/3209582.3209603.

[5] S. Henri, C. Vlachou and P. Thiran, “Multi-Armed Bandit in Action: Optimizing Performance in Dynamic Hybrid Networks,” *IEEE/ACM Transactions on Networking*, 2018, doi: 10.1109/TNET.2018.2856302.

Patents

[6] N. Khude, S. Tavildar, S. Henri, N. Abedini, J. Li and V. Park, “Method and Apparatus for Dissemination of Timing Information in Distributed Synchronization Device to Device Networks,” US Patent 20 150 078 369, 2015, A1, filed by Qualcomm, n° US 14/029,752.

[7] N. Abedini, N. Khude, S. Tavildar, S. Henri, J. Li and V. Park, “Distributed Device-to-device Synchronization,” 2015, A1, filed by Qualcomm, n° US 14/286,354.

[8] N. Khude, S. Henri, V. Park and J. Li, “Method and Apparatus for Timing Source Selection and Deselection Distributed Device to Device Synchronization,” US Patent 9 888 449, 2015, B2, filed by Qualcomm, granted Feb. 2018, n° US 14/158,701.

[9] N. Khude, S. Henri, V. Park and J. Li, “Method and Apparatus for Resource Allocation for Distributed Device to Device Synchronization,” US Patent 9 467 957, 2015, B2, filed by Qualcomm, granted Oct. 2016, n° US 14/160,381.

Professional Services

Reviewer for IEEE Infocom 2018 and IEEE Transactions on Mobile Computing (TMC)

Technical Skills

Programming	Python, C/C++, Java, Matlab

Networking	◦ TCP/IP stack, IEEE 1901 (PLC) and 802.11n/ac/ad (WiFi) standards ◦ Design, analytical modeling and performance evaluation of network protocols ◦ Experimental and simulation platforms: *Click Modular Router*, NS3 ◦ Development and maintenance of a PLC/WiFi testbed of 22 OpenWrt APU routers ◦ Development with two other research assistants of a Python simulator for IEEE 1901 and 802.11 standards ◦ Security of PLC and WiFi
Data Science	Machine Learning (models and implementations)

Languages

French	Native language	English	Fluent
German	Working knowledge	Spanish	Basic knowledge

Extracurricular Activities

Sports	◦ Soccer (member of École polytechnique's and Télécom ParisTech's soccer teams, 20+ years of practice) ◦ Ice hockey (member of EPFL's ice hockey team) ◦ Skiing, cycling, tennis, running (half-marathon)