EMPoWER Hybrid Networks: Exploiting Multiple Paths over Wireless and ElectRical Mediums

Sébastien Henri†, Christina Vlachou†, Julien Herzen‡*, Patrick Thiran†
† EPFL, Switzerland ‡ Swisscom, Switzerland
† firstname.lastname@epfl.ch ‡ julien.herzen@swisscom.com

ABSTRACT

Several technologies, such as WiFi, Ethernet and power-line communications (PLC), can be used to build residential and enterprise networks. These technologies often co-exist; most networks use WiFi, and buildings are readily equipped with electrical wires that can offer a capacity up to 1 Gbps with PLC. Yet, current networks do not exploit this rich diversity and often operate far below the available capacity.

We design, implement, and evaluate EMPoWER, a system that exploits simultaneously several potentially-interfering mediums. It operates at layer 2.5, between the MAC and IP layers, and combines routing (to find multiple concurrent routes) and congestion control (to efficiently balance traffic across the routes). To optimize resource utilization and robustness, both components exploit the heterogeneous nature of the network. They are fair and efficient, and they operate only within the local area network, without affecting remote Internet hosts. We demonstrate the performance gains of EMPoWER, by simulations and experiments on a 22-node testbed. We show that PLC/WiFi, benefiting from the diversity offered by wireless and electrical mediums, provides significant throughput gains (up to 10x) and improves coverage, compared to multi-channel WiFi.

Keywords

Hybrid networks; Power-line communications; Multipath routing; Multipath congestion-control; Multi-channel WiFi.

1. INTRODUCTION

Today's residential and enterprise networks often rely on heterogeneous link technologies. WiFi is used nearly everywhere as a means of providing easy access to mobile clients. It is typically completed by wired links that serve as a backbone, provide access to fixed clients, or are used as range-extenders for WiFi. Traditional wired architectures usually rely on Ethernet, but power-line communications (PLC) are becoming increasingly popular as a means of providing wired connections with no additional cabling infrastructure. In fact, as of 2014, more than 100M PLC devices have been sold worldwide, and the branch is expecting a growth of over 30% by 2017.[1] This trend is well embodied in commercial solutions, such as Qualcomm HyFi[2] that proposes products that combine WiFi and PLC. In parallel, significant efforts have been devoted to the development of new standards for hybrid networks. In particular, the IEEE 1905.1 standard [2] was proposed to provide an abstraction layer (between the data link and network layers) that combines several link technologies, which brings numerous benefits: When the technologies do not interfere with each other, it is possible to aggregate their capacity, thus enabling immediate throughput improvements. Moreover, using different technologies such as PLC and WiFi, as opposed to 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. In Section 6, we show that compared to using two non-interfering WiFi channels, using hybrid PLC/WiFi yields significant throughput improvements. Achieving these benefits is not trivial and calls for the careful development of new algorithms.

Despite the commercial success and standardization efforts, there has been little research on how to design such efficient algorithms in practice.[3] Moreover, to the best of our knowledge, there is no published work on the algorithms used in commercial products. In Section 7, we review related work on distributed and centralized algorithms, and on

*Work done while at EPFL.

CoNEXT '16, December 12 - 15, 2016, Irvine, CA, USA

 ISBN 978-1-4503-4292-6/16/12…$15.00

DOI: http://dx.doi.org/10.1145/2999572.2999574

[1] http://www.qca.qualcomm.com/products/qualcomm-powerline

[2] http://www.qca.qualcomm.com/networking/connected-home/hybrid

[3] The IEEE 1905.1 standard and the corresponding *nVoy* certification focus on establishing an abstraction layer, without specifying routing or load-balancing algorithms.

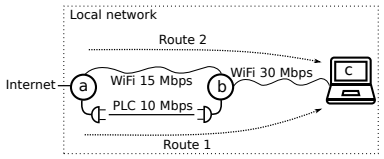

Figure 1: Scenario with a PLC/WiFi gateway (a), a PLC/WiFi range extender (b), and a WiFi client (c). Optimal load balancing yields 10 Mbps on Route 1 and 6.6 Mbps on Route 2.

hybrid networks. The design of efficient and practical algorithms for hybrid networks raises the following questions:

1. How can these algorithms optimally exploit all mediums?
2. How can they be implemented in practice?
3. Which features of the underlying PHY and MAC layers should and can they take into account?

We propose EMPoWER, a system that includes a *multipath-routing algorithm* and a *congestion-control algorithm*, and that achieves gains close to optimum. EMPoWER is technology independent and can be deployed in any hybrid network, in particular, in PLC/WiFi networks, which are the topic of this paper. Our algorithms are based on existing works designed for a single path in wireless networks or for wired networks. We extend these works to achieve practical multipath-routing and congestion-control with self-interfering technologies. One of the main contributions of our work is the real-world implementation of EMPoWER in a hybrid PLC/WiFi network. We show that combining PLC with WiFi can yield significant performance improvements.

We now describe how we solve the above questions.

Challenge 1: (i) Hybrid algorithms should take interference into account: The absence of interference across different links (e.g., using non-interfering technologies) means that multiplexing gains are possible, whereas the presence of interference can reduce throughput. In both cases, interference needs to be accounted for when choosing the routes and when load-balancing traffic. Similarly to WiFi, PLC links are subject to interference,[4] which therefore needs to be accounted for in different technologies, and not only in wireless links. (ii) For performance to be maximized and technologies to be fully exploited, hybrid algorithms should be able to simultaneously employ multiple paths.

For example, take the network of Figure 1, with a hybrid PLC/WiFi gateway (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 Mbps can be sent over the hybrid PLC-WiFi Route 1, as per requirement (i) above. This is what would happen with a typical PLC/WiFi extender, but would consume only $1/3$ of WiFi resources. To maximize throughput, *some* traffic can be sent over the two-hop WiFi Route 2, to meet requirement (ii) above. The amount of traffic needs to be carefully calibrated, as it is well known that saturating multihop paths is inefficient and can lead to congestion collapse with packet losses and instabilities [11, 33]. This amount depends on the characteristics of the two WiFi links $a \rightarrow b$ and $b \rightarrow c$ (these links cannot transmit simultaneously and need to share the capacity). In this simple case (see Section 4 for general cases), a back-of-the-envelope computation gives us the rate x to send over Route 2 as the solution of $x/15 + x/30 = 2/3$, i.e., $x \simeq 6.6$ Mbps. Hence using both Route 1 and Route 2 provides a 66% improvement, compared to using Route 1 alone. Even in this simple example, optimally balancing traffic is not trivial.

[4]To avoid collisions, IEEE 1901 (the *HomePlug* MAC/PHY standard used by the vast majority of PLC devices) employs a CSMA/CA scheme relatively similar to that of 802.11 [40].

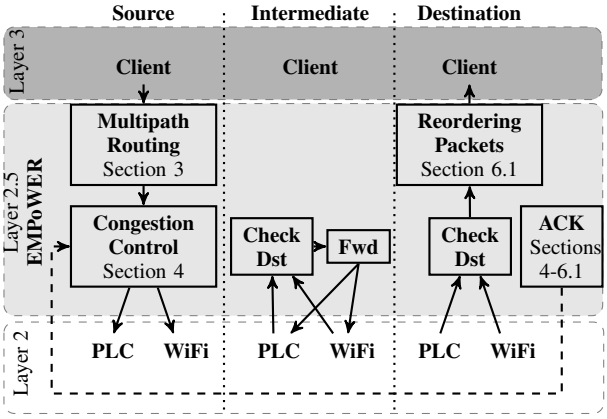

Figure 2: The main components of EMPoWER at layer 2.5. The source determines the routes (Multipath Routing) and the rates at which data is sent on each route (Congestion Control). Congestion control is separated from route selection for the sake of practical implementation. Using the header of our EMPoWER 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.

Challenge 2: Algorithms should operate in a distributed fashion, be transparent to other network protocols, and make an optimal use of the network in terms of throughput and fairness among the different flows.

EMPoWER follows a scheme (described in Sections 3 and 4) that maximizes a global utility function. Practical and distributed, it achieves a global performance close to optimal-but-impractical schemes. Practicality is achieved, in particular, by performing congestion control on preselected routes, thus separating routing from congestion control. Figure 2 summarizes the main components of EMPoWER. EMPoWER operates at layer 2.5, between the MAC and IP layers; it is hence confined to the local network and transparent to other IP hosts and protocols. In contrast, solutions such as Multipath TCP (MPTCP) act at layer 4 and target end-to-end paths. Both end-hosts have to handle MPTCP connections. In practice, if a client has only one interface (e.g., the laptop in Figure 1), MPTCP is not supported. We

further discuss MPTCP in Section 7. We evaluate the performance of EMPoWER: in Section 5, by simulations; and in Section 6, with an implementation on a testbed with both hybrid PLC/WiFi and multi-channel WiFi.

Challenge 3: Although the algorithms should be oblivious of underlying technologies, they should take into account crucial features of the PHY and MAC layers to make their implementation practical. For instance, link metrics for routing and queue-occupancy measurements for congestion control need to consider recent amendments of communication technologies (e.g., frame aggregation); recent works on WiFi and PLC link-metrics point out this requirement [31, 38]. EMPoWER employs accurate capacity link-metrics that are specific to each technology, making the estimation of link-metrics the only technology-dependent feature. We introduce a network model and a routing algorithm that use only capacities in Sections 2 and 3, and a technology-independent congestion controller in Section 4.

In summary, our contributions are the following:

- A multipath-routing algorithm that finds efficient paths in terms of total throughput. Its goal is to find paths that yield the highest total throughput when used simultaneously. Its novelty is to optimize specifically *total throughput* for *multiple self-interfering technologies*.
- A distributed congestion-control algorithm designed for *multipath use* and *self-interfering technologies*, that performs very close to an optimal, centralized scheduler, with significantly shorter convergence times.
- To the best of our knowledge, *the first published implementation and evaluation of congestion-control and multipath-routing algorithms in hybrid PLC/WiFi networks*, provably improving performance of both single-path and routing without congestion control.
- Evidence that, despite comparable aggregate capacities, *PLC/WiFi can improve throughput and coverage significantly compared to multi-channel WiFi* that has been widely studied and deployed, due to medium diversity.

2. NETWORK MODEL

In this section, we provide some necessary definitions and modeling assumptions. As explained in Section 1, we express our model only in terms of link capacities.

We consider a setting where N nodes compose a local network with K different technologies. This network can be seen as a *multigraph* $G(V, \{E_1, \ldots, E_K\})$, where V is the set of nodes, and E_k the set of links available with technology k. A link is present whenever its two endpoints can communicate with nonzero capacity with the corresponding technology. $\mathcal{L}$ is the set of all links ($\mathcal{L} = \cup_k E_k$). For a link $l \in \mathcal{L}$, we write c_l for the capacity of l, and we define $d_l = 1/c_l$. We write I_l for the *interference domain* of link l. It is defined as the set that contains l itself and all the links that cannot transmit simultaneously with l (because doing so would cause a collision at one of the links). A *route* (or, equivalently, a *path*) from a source $s \in V$ to a destination $d \in V$ is a set of links that join (without loop) s to d.

We define the *airtime* μ_l of a link l as the *fraction of time* during which l is active. When a link l is not saturated, we assume that its airtime is given by

$$\mu_l = \frac{x_l}{c_l} = x_l \cdot d_l, \tag{1}$$

where x_l denotes the traffic rate at link l. In Section 3, we show how this formulation can be exploited for finding efficient combinations of paths to be employed simultaneously. Then, in Section 4, we show how expressing interference in terms of the airtimes of the links can yield a simple constraint that can be used for distributed load balancing.

3. MULTIPATH ROUTING

In this section, we present our novel multipath-routing algorithm for hybrid networks. Its purpose is to obtain *combinations of paths* that can be efficiently employed *simultaneously* by a given flow. The multipath procedure is described in Section 3.2. It aims at finding the combination of paths yielding the highest total throughput in the presence of interference, as opposed to procedures that only focus on finding maximally disjoint paths [e.g., 6, 10, 21, 36]: In our protocol, a same link can, and should if useful, be used by several paths. As this multipath procedure is fairly computation-intensive, it needs in order to be practical to be applied on a limited number of paths. These paths are computed by an efficient single-path procedure, described in Section 3.1. This single-path procedure is based on an algorithm for multi-channel wireless networks, proposed by Yang et al. [44].

3.1 Single-Path Procedure

The single-path procedure applies an efficient shortest-path algorithm (e.g., Dijkstra's algorithm, that we use for EMPoWER) on the multigraph. A weight $W(l)$ is assigned to each link. In addition, because contiguous links using 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. [44], we add a *channel-switching cost* (CSC). The CSC is an *extra* weight assigned to each node u, 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. Requiring $w_s(u) < w_{ns}(u)$ at each node u favors paths with alternating technologies. To make the CSC compatible with Dijkstra's algorithm, we perform the shortest-path computations on the virtual graph of the network *interfaces* [44].

In EMPoWER, 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 [7], used by many schemes [7, 20]. If the weights w_{ns} and w_s can be chosen differently for each path, it is easy to show (see our Tech. Report [15]) that the optimal CSC for adjacent links is $w_{ns} = 0$ and $w_s = -\min(d_{l_i}, d_{l_e})$, with l_i the ingress link and l_e the egress link. But to guarantee that Dijkstra's algorithm converges to the shortest path, w_{ns} and w_s need to be chosen globally for a node and to be non-negative. For these reasons, we choose at any node u, based on the optimal CSC

and because a link is more likely to be used in a path if it has high capacity (i.e., small d_l), $w_{ns}(u) = \min_{l \in L(u)} d_l$ and $w_s(u) = 0$, with $L(u)$ the set of egress links for node u. The link-metric W is different from the metric, called IRU, of Yang et al.: Our routing protocol focuses on intra-flow interference (*via* the CSC), whereas IRU also accounts for inter-flow interference. With EMPoWER, the congestion controller is responsible for handling inter-flow interference.

To guarantee that Dijkstra's algorithm returns the shortest path, the link-metric W must be isotone [32], which explains our choices above. Although the single-path procedure is not necessarily optimal, as the shortest path is not always the route with highest throughput, the evaluation in Section 5 nevertheless shows that the procedure succeeds in finding good routes. In addition, we account for this non-optimality in the multipath procedure presented next.

3.2 Finding Efficient Combinations of Paths

We now introduce our novel multipath-routing protocol. We start with a useful lemma that follows from the model described in Section 2.

LEMMA 1. *If λ links $l_1, \ldots, l_\lambda$ are all contending for the same medium in a single collision domain ($\forall i, j, I_{l_i} = I_{l_j}$), 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 $R_{max} = \left(\sum_{i=1}^{\lambda} d_{l_i}\right)^{-1}$.*

PROOF. This is equivalent to solving $\sum_{i=1}^{\lambda} \mu_{l_i} = 1$ (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 yields the result. □

We now describe our procedure for finding efficient combinations of hybrid paths. To quantify the effect of employing several paths simultaneously, we define a procedure $\widetilde{G} = \texttt{update}(P, G)$ for a multigraph G and a path P. $\widetilde{G}$ 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 $\widetilde{G}$ 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 $\left(\sum_{l' \in I_l \cap P} d_{l'}\right)^{-1}$. A traffic rate R is supported by P if and only if $R \leq R(l, P)$ for all $l \in P$, hence $R(P) = \min_{l \in P} R(l, P)$, or, equivalently,

$$R(P) = \left(\max_{l \in P} \sum_{l' \in I_l \cap P} d_{l'} \right)^{-1}.$$

We use (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

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

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.

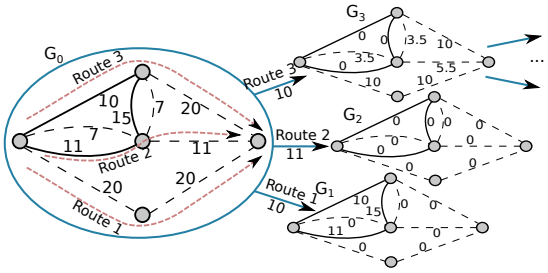

Figure 3: Illustration of the tree construction, with $n = 3$. G_0 consists of two mediums (resp. solid and dashed lines); all links using the same medium interfere. Link capacities are in Mbps. G_1, G_2 and G_3 are multigraphs with capacities updated to reflect the consumption of resources by resp. Routes 1, 2, and 3. Only G_3 has some spare capacity, therefore, the construction process continues only from G_3, after which all links have 0 capacity.

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

For each link $l \in \bigcup_{l' \in P} I_{l'}$, 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 topologies like the typical example of Figure 1. 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 Section 5.2). To compute efficient combinations of paths, we recursively apply the procedure `update`, along with the single-path procedure of Section 3.1. Observe that the best path is not necessarily part of the best *combination* of paths: Consider for example the network G_0 of Figure 3. Route 2 can accommodate a traffic rate of 11 Mbps, whereas Routes 1 and 3 can both accommodate only 10 Mbps: Route 2 is the best *isolated* route. However, the best *combination* of two routes is given by Routes 1 and 3: With Lemma 1, we compute that they can accommodate together $5 + 10 = 15$ Mbps. To avoid being constrained to always using the best isolated route, we compute in G the n shortest paths of our single-path procedure described above. We denote this step by $n\texttt{-shortest}(G)$. 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* $\mathcal{T}$ 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 tree is built recursively; its construction is illustrated in Figure 3. To a vertex G of $\mathcal{T}$, we add $j \leq n$ edges that are the j non-empty paths $(P_i)_{i \leq j}$ returned by $n\texttt{-shortest}(G)$, and j children vertices that are the multigraphs $\texttt{update}(P_i, G)$, for which link capacities have been updated to reflect the consumption of resources by P_i. After $\texttt{update}(P, G)$, we know that at least one link $l_0 \in P$ has a zero capacity, thus the procedure for building $\mathcal{T}$ eventually terminates; but the depth of $\mathcal{T}$ 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 could be limited by stopping the `update` process along a branch of $\mathcal{T}$, once this branch has reached a certain depth. In practice, this limitation is not necessary for local networks, where paths are limited to a few hops at most. Because we use shared-medium technologies, after a call to `update`, multiple links of the network get a zero-capacity, naturally limiting the depth of $\mathcal{T}$. In our simulations and testbed experiments, the depth of the tree is never more than 3, and almost always equal to 1 or 2.

To each edge P out of a vertex G, we associate a weight equal to the capacity $R(P)$ of P in G. For each leaf G_L, let $\mathcal{B}(G_L)$ be the set of edges (i.e., paths of the network) from G_0 to G_L. Then $C_{\mathcal{B}}(G_L) = \sum_{P \in \mathcal{B}(G_L)} R(P)$ is the total achievable capacity when using simultaneously all the network paths in $\mathcal{B}(G_L)$. The final paths returned by the whole routing procedure are the paths in $\mathcal{B}(G_L^M)$ of maximum associated capacity, i.e., $G_L^M = \arg\max_{\text{leafs } G_L} C_{\mathcal{B}}(G_L)$. With this method, the number of paths returned by the routing algorithm depends on the cardinality of $\mathcal{B}(G_L^M)$. This is a desirable feature, as it means that the number of paths depends on the network topology, and that additional paths are considered only if they provide additional gain.

Note that if we limit our routing protocol to returning only one route, we do *not* necessarily find the route returned by the single-path procedure described in Section 3.1 (see our Tech. Report [15] for an example). In Section 6.3, we show that our multipath-routing protocol performs much better than the single-path procedure. This comes at the cost of an increased complexity. However, our protocol remains practical: On our routers (described in Section 6), with $n = 5$, the routes are computed in about 50 ms. The routing protocol is not responsible for dealing with short-term variability, handled by the congestion controller (described in Section 4); the routes need to be recomputed only when there is a link failure or a large capacity variation, which occurs infrequently (order of minutes or hours), hence this computation time is not an issue in practice.

4. CONGESTION CONTROL

In this section, we assume that the routes for any source-destination pair are known by the source. The congestion-control algorithm decides the amount of traffic to inject over each route. Its goal is to maximize aggregate network utility (i.e., a chosen performance/fairness tradeoff), while avoiding congestion (thus keeping the queues stable). We begin by addressing the single-path case, where there is exactly one route between any source-destination pair. We then present the extension to multiple paths. To the best of our knowledge, this is the first *practical* and *interference-aware* congestion controller using link capacities designed for *multiple paths*.

4.1 Notation and Interference Constraint

We define $\mathcal{F}$ as the set of flows, where each flow is a source-destination pair (and can potentially employ several routes), and $\mathcal{R}$ as the set of all routes (across all flows). With each route $r \in \mathcal{R}$, we associate x_r, the traffic rate that the source of r sends over route r. We define the vector $\boldsymbol{x} = (x_r)_{r \in \mathcal{R}}$. The congestion controller decides the value of x_r for each r.

To avoid congestion, the rate injected on each route 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 the previous section, the airtime μ_l consumed by an unsaturated link l is given by $\mu_l = d_l \sum_{r: l \in r} x_r$. Hence a sufficient constraint to keep the queues finite is to require that the traffic intensities satisfy

$$\sum_{l' \in I_l} d_{l'} \sum_{r: l' \in r} x_r \leq 1 \quad \forall l \in \mathcal{L}. \tag{2}$$

This requires that the *airtime demand* does not exceed 100% in each interference domain. In practice, when the airtime approaches 1, the delays increase rapidly; for this reason, we might want to place a more conservative constraint

$$\sum_{l' \in I_l} d_{l'} \sum_{r: l' \in r} x_r \leq 1 - \delta \quad \forall l \in \mathcal{L}, \tag{3}$$

with a *constraint margin* δ in $[0, 1]$. Constraints (2) and (3) are conservative: The airtime could be reused by some links of an interference domain if there is no pairwise interference between them. However, (2) and (3) have the crucial advantage of being formulated locally (w.r.t. 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 5, we study the effect of this conservative policy and compare our scheme with optimal solutions based on perfect centralized scheduling.

4.2 Congestion Control on a Single Path

We consider here that a single route is used by each flow: There is a one-to-one mapping between routes and flows ($\mathcal{F} = \mathcal{R}$). To each route $r \in \mathcal{R}$ we attach an increasing and strictly concave utility function $U_r : \mathbb{R}^+ \to \mathbb{R}^+$; it describes the benefit that the source of r gets by sending traffic at rate x_r. We formulate an optimization problem similar to what was proposed for wired networks [17, 25]. 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 (2), and we re-

place the constraints on the link *capacities* with constraints on the *airtimes* of the interference domains. The problem then reads

$$\max_{\boldsymbol{x}} \sum_{r\in\mathcal{R}} U_r(x_r) \tag{4}$$

$$\text{subject to} \sum_{l'\in I_l} d_{l'} \sum_{r:l'\in r} x_r \leq 1 \qquad \forall l \in \mathcal{L}, \tag{5}$$

$$x_r \geq 0 \qquad \forall r \in \mathcal{R}. \tag{6}$$

Constraints (5) and (6) are linear: The problem is the maximization of a strictly concave function over a convex set. Assuming a slotted time, this yields (see our Tech. Report [15] for computation details) the following controller for the route rates x_r, with auxiliary variables γ_l, y_l, and q_r:

$$y_l[t] = \sum_{l'\in I_l} d_{l'} \sum_{s:l'\in s} x_s[t], \tag{7}$$

$$\gamma_l[t+1] = \left[\gamma_l[t] + \alpha_t(y_l[t]-1)\right]^+, \tag{8}$$

$$q_r[t] = \sum_{l\in r} d_l \sum_{i\in I_l} \gamma_i[t], \tag{9}$$

$$x_r[t+1] = U_r'^{-1}(q_r[t]), \tag{10}$$

with $\alpha_t > 0$ a sequence of step sizes, and $[y]^+ = y$ if $y > 0$ and 0 otherwise. This controller belongs to a class of controllers with linear constraints for which it can be shown [23] that if $\alpha_t \to 0$ as $t \to \infty$, and $\sum_t \alpha_t = \infty$, then the rate allocation vector $[x_r[t], r \in \mathcal{R}]$ converges to the optimal solution of the problem given by (4)-(6). In EMPoWER, we use a fixed *step size* $\alpha_t = \alpha$ in order to continuously adapt to changes in the network. Again, it can be shown [23] that if α is small enough, the rate allocation converges to a small neighborhood of the optimizer of Problem (4)–(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 $d_l \sum_{r:l\in r} x_r$ 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 over all egress links employing k, and (ii) the sum of the dual variables γ_l for each egress link l employing k. The node periodically broadcasts a packet that contains these two values over the medium k. 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 k, and (ii) the airtime demands received by all their neighbors, as described by (7). Knowing y_l, the nodes then update γ_l using (8). Finally, when forwarding a packet on link l, the nodes add the current value of $d_l \sum_{i\in I_l} \gamma_i$ to a dedicated field in the packet's 2.5-layer header (see Section 6 for more details). At the destination of route r, the value of this field is thus equal to q_r (given by (9)), and the destination can send back q_r to the source, via an acknowledgement. Upon reception of q_r, the source updates the rate x_r for route r using (10). This mechanism requires only local measurement 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 6.

4.3 Multipath Congestion-Control

We present the extension of our interference-aware congestion controller to multiple paths. We now differentiate between *flows* and *routes*: Each flow can potentially employ several routes. To express the availability of route r to flow f, we write $r \in f$. The flow of route r is denoted by $f(r)$. The utility obtained by each flow f is now given by $U_f(x_f)$ with $x_f = \sum_{r\in f} x_r$. 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 $\boldsymbol{x} = (x_r)_{r\in\mathcal{R}}$ anymore. To overcome this problem, we adopt the approach of *proximal optimization*, introduced by Wang et al. [41] when no interference is present, and we maximize another objective function, which has the same optimizer as the original, but is strictly concave in $\boldsymbol{x}$:

$$\max_{\boldsymbol{x},\bar{\boldsymbol{x}}} \sum_{f\in\mathcal{F}} \left(U_f\left(\sum_{r\in f} x_r\right) - \frac{1}{2}\sum_{r\in f}(x_r-\bar{x}_r)^2 \right) \tag{11}$$

$$\text{subject to} \sum_{l'\in I_l} d_{l'} \sum_{r:l'\in r} x_r \leq 1, \quad \forall l \in \mathcal{L} \tag{12}$$

$$x_r \geq 0, \quad \forall r \in \mathcal{R},$$

where $\bar{x}_r$ is an auxiliary variable for route r and $\bar{\boldsymbol{x}} = (\bar{x}_r)_{r\in\mathcal{R}}$. Subtracting a quadratic term strictly convex in $\boldsymbol{x}$ in (11) makes the objective function strictly concave in $\boldsymbol{x}$ [41]. The optimization is now performed over the two variables $\boldsymbol{x}$ and $\bar{\boldsymbol{x}}$. The new objective function (11) has the same optimizer $\boldsymbol{x} = \bar{\boldsymbol{x}}$ as the original one (4). We obtain the following discrete-time multipath congestion controller:

$$x_r[t+1] = \left[(1-\alpha)x_r[t] + \alpha\left(\bar{x}_r[t] + U_{f(r)}'\left(\sum_{h\in f(r)} x_h[t]\right) - q_r[t]\right)\right]^+$$

$$\bar{x}_r[t+1] = (1-\alpha)\bar{x}_r[t] + \alpha x_r[t],$$

with $y_l[t]$, $\gamma_l[t+1]$, and $q_r[t]$ given by (7), (8), and (9).

Accounting for interference does not change the linearity of (12), and the convergence of this controller can be established using similar techniques as Lin and Shroff [24]. This controller employs the same update rules as the single-path controller for γ_l, y_l, and q_r. In addition, the update rule for x_r depends only on q_r and $\bar{x}_r$. Thus, it can be implemented in a distributed way, exactly as the single-path controller.

This controller is able to account for external interference: Nodes can measure traffic from external nodes and add the corresponding airtimes in (7); however, because it has *no control* over external nodes, EMPoWER converges to the optimal allocation *under this external load*, which means that, except during a short transition phase, non-EMPoWER clients are not affected by EMPoWER clients. Although this

allocation does not affect external nodes, it is not necessarily fair: for example, if one external node saturates WiFi, EMPoWER converges to an allocation that never uses WiFi. Extending EMPoWER to support external interference in a fair way is left for future work.

The main downside of this controller is that it takes some time to converge (see Sections 5 and 6). For this reason, it is designed mostly for long-run best-effort flows (e.g., large-file download or video streaming). Yet, in Section 6.3, we show that EMPoWER helps also for short flows.

5. EVALUATION VIA SIMULATIONS

Before validating the performance gains of EMPoWER via testbed experiments, 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 an optimal algorithm based on backpressure, proposed by Neely et al. [27]. Implementing this algorithm would be very challenging, as it includes an optimal NP-hard [13] scheduling algorithm, that (i) requires solving a centralized optimization problem at each time step, and (ii) requires modifying the underlying MAC layer.

5.1 Simulation Settings

We write a packet-level simulator in Matlab.[5]. We simulate two different network topologies. The first topology is a typical residential network: On a 50×30 m rectangle, we drop uniformly at random 10 nodes. 5 nodes have both WiFi and PLC and can typically be PLC/WiFi gateways or extenders, desktop computers, connected televisions, etc.; 5 nodes have only single-channel WiFi and can typically be mobile phones, laptops, etc. The second topology is a typical enterprise network (e.g., company, hospital): On a 100×60 m rectangle, we drop 20 nodes. 10 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 both scenarios, the source of a flow is chosen among the PLC/WiFi nodes, and the destination is chosen among all the nodes, 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.

A link between two nodes u and v exists if the distance between u and v is smaller than a given connection radius R. Based on measurements from our indoor testbed (see Section 6) and reported in our Tech. Report [15], 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* [1], 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. Hence, for the 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 our Tech. Report [15]. As previously noted [38], the capacities for WiFi and PLC with the technologies used, respectively 802.11n and HPAV 200,[6] are similar. When employing two non-interfering WiFi channels, we consider that the two channels have the same bandwidth, consequently the same link capacities.

In this section and Section 6, 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 Sections 5 and 6 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.

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 3.1.[7] 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 3.2. We use $n = 5$ for n-`shortest`, which enables route diversity while limiting the number of possible combinations to be explored.

We then compare EMPoWER with the optimal backpressure algorithm [27].

5.2 Simulation Findings

We consider scenarios with one flow in Sections 5.2.1 and 5.2.2, and with several contending flows in Section 5.2.3.

[5]The source code of our simulator and of our Click implementation described in Section 6 is available at https://c4science.ch/diffusion/1252/empower.git

[6]HPAV 200 is specified by *HomePlug*, the leading alliance in PLC. It offers rates up to 150 Mbps.

[7]We also implemented other single-path procedures employing different metrics, such as IRU [44], ETT [7], and CATT [12]; all gave worse results in our experiments.

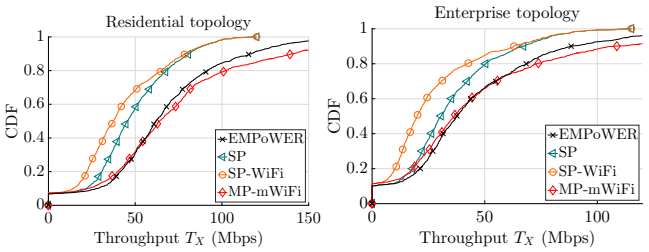

Figure 4: Distribution of T_X for different schemes X. Residential (left) and enterprise (right) topologies.

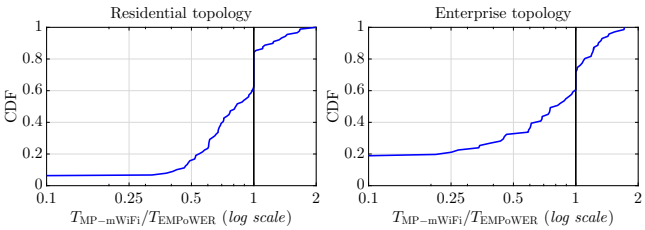

Figure 5: Distribution of $T_{\text{MP-mWiFi}}/T_{\text{EMPoWER}}$, worst flows. Residential (left) and enterprise (right) topologies.

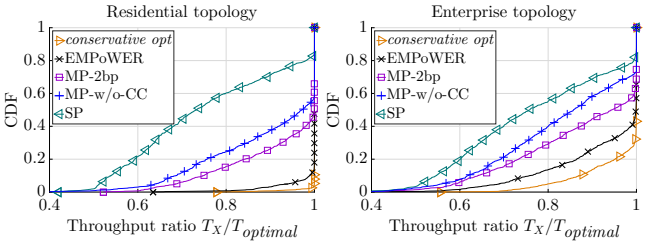

Figure 6: Distribution of $T_X/T_{optimal}$. Residential (left) and enterprise (right) topologies.

5.2.1 *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 routes returned by the routing algorithm (one for SP, two for MP). The results are presented in Figure 4, where we show the empirical cumulative distribution of the flow throughput T_X achieved by each scheme X for both residential (left) and enterprise (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 shows that multipath improves the throughput, only if there are two or more non-interfering technologies. Hybrid networks and multipath routing both contribute significantly to performance gains: In the residential (resp., enterprise) topology, the average gain compared to WiFi alone is 59% (resp., 68%), and it is 39% (resp., 31%) compared to single-path hybrid.

We now compare PLC/WiFi with multi-channel WiFi (two non-interfering channels). Because the two WiFi channels have the same capacities, $T_{\text{MP-mWiFi}} = 2T_{\text{SP-WiFi}}$. Figure 4 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 [38]. 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 5, we present the cumulative distribution of the ratio $T_{\text{MP-mWiFi}}/T_{\text{EMPoWER}}$ for the *worst flows*: We call worst flows the bottom-20% of the flows w.r.t. 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 3x (residential topology) or 4x (enterprise topology); our experiments on a real testbed (see Section 6) 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. For about 6% of the worst flows in the residential topology, 19% in the enterprise topology, PLC/WiFi brings connectivity where multi-channel WiFi does not ($T_{\text{MP-mWiFi}} = 0$ whereas $T_{\text{EMPoWER}} > 0$).

5.2.2 *EMPoWER vs. Optimal Schemes*

We evaluate the performance of EMPoWER by comparing it with a backpressure scheme, shown to be optimal [27]. EMPoWER differs from the optimum for two reasons: (i) It employs preselected routes, whereas the optimal scheme finds the optimal routes using backpressure, and (ii) it uses the conservative constraint (2). For this reason, we show results of this backpressure scheme in two different scenarios

- with an optimal centralized scheduler yielding 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 giving the optimal result under the condition (2) 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 (2).

For baselines, we use SP, MP-w/o-CC, and MP-2bp. In Figure 6, 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 (resp., enterprise) topology, in 98% of the cases (resp., 85%), the performance loss of EMPoWER is less than 10%. This shows that our multipath-routing protocol succeeds in finding good routes. The performance differences with SP and MP-2bp unveil that multipath routing is beneficial, and that finding the good routes is not trivial; in addition, the performance of multipath routing strongly depends on the congestion controller. The penalty of using the condition (2) is not severe in the vast majority of the cases.

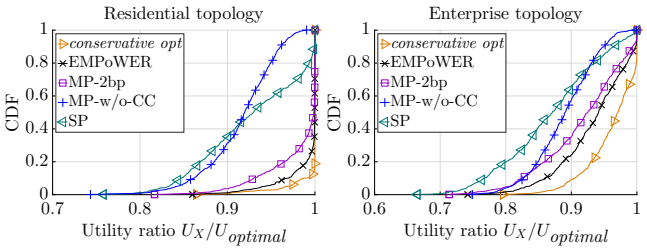

Figure 7: Distribution of $U_X/U_{optimal}$. Residential (left) and enterprise (right) topologies.

In total, in the residential topology, EMPoWER achieves optimal throughput in 88% of the cases, and in 99% of the cases, the performance loss w.r.t. *optimal* is less than 15%. In the enterprise topology, EMPoWER outperforms both SP and MP-2bp; the penalty of using (2) is a bit higher, but EMPoWER still achieves optimality in 60% of the cases, and in 83% of the cases, the performance loss w.r.t. *optimal* is less than 15%. Furthermore, EMPoWER converges drastically faster: In the residential (resp., enterprise) topology, it requires 90 time slots[8] (resp., 77) on average to reach steady-state ("steady" meaning that the throughput is within 1% of the final throughput), whereas *optimal* and *conservative opt* require more than 3 000 time slots (resp., 10 000). In fact, *optimal* and *conservative opt* suffer from symptoms of backpressure-based routing; although they are throughput-optimal at steady-state, good routes are employed only after the queues on the bad routes start to fill up.

5.2.3 *Utility Maximization and Proportional Fairness*

We now evaluate how EMPoWER distributes network resources among several competing flows. In Figure 7, we show the distribution of the total network utility when there are three 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 $1 \leq f \leq 3$ gets a 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 w.r.t. to MP-2bp when several flows are competing. Due to space constraints, results for the convergence time are not presented here; they are very similar to those of the single-flow case (see our Tech. Report [15]).

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

[8]In EMPoWER, a time slot is the interval between two acknowledgements, and it is 100 ms in our implementation; in *optimal*, it is the time between two calls to the centralized scheduler, which causes much higher communication overhead and computation time.

6. EVALUATION ON HYBRID TESTBED

In this section, we evaluate EMPoWER on a real testbed, and demonstrate its practical usability. We first describe our testbed and 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.

6.1 Experimental Settings

We implement EMPoWER on a testbed of 22 nodes spread over an entire floor of an office building (see Figure 8). All the nodes have two WiFi interfaces (Atheros AR9280), and a HomePlug AV PLC interface (QCA 7420) connected to the electrical network of our building. The nodes are APU1D boards running an OpenWrt Linux distribution with the open-source ath9k wireless drivers. The PLC interfaces use a Realtek Ethernet driver. We implement EMPoWER by using the *Click Modular Router* [19] in user space. 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), which we call Channel 1, and one from 2.412 GHz to 2.452 GHz, which we call Channel 2. To evaluate fairly our algorithm, we avoid external interference: Because Channel 2 is used by the WiFi network of the university, we run all multi-channel WiFi experiments at night, and we verify that there is no external traffic. For hybrid PLC/WiFi experiments, PLC and Channel 1 are used.[9] Link capacities with WiFi and PLC are comparable [15, 38] (the maximum link capacity is about 100 Mbps in both cases), which means that PLC/WiFi and multi-channel WiFi have comparable aggregate capacities.

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 routes. If several routes exist, each packet is sent over route r with a probability proportional to the rate x_r. We employ source routing: Route r 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 interfaces' MAC addresses 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 route (2 bytes are used to identify each ingress interface along the route, and the total length is limited to 6 hops). 4 bytes are used to store the variable q_r along route r, as described in Section 4.2. 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 routes. We do not use timeouts for missing packets. To identify a lost packet (because of chan-

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

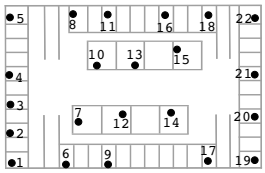

Figure 8: Map of the testbed (65×40 m).

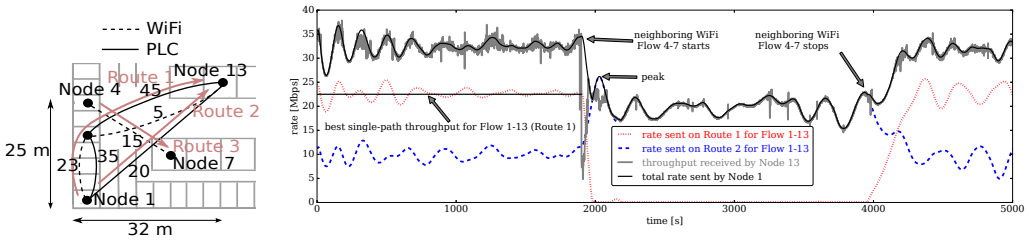

Figure 9: Left: scenario with two flows; the numbers on the figure indicate the measured link capacities in Mbps at the time of the experiment. Right: time evolution of the injected rates and measured throughput on Flow 1.

nel errors or congestion), the destination stores the last sequence number received from each route: A packet with a sequence number S is lost when it has received packets with sequence number greater than S on all routes 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. In previous work, we show that these two metrics are extremely accurate when traffic is sent at a high rate [38, 39]. 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) [38, 39], 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 route is two-hop; and by 4 when the longest route 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. We leave the careful investigation of finding the optimal α for future work.

6.2 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 capacities of the links involved are depicted in Figure 9 (left). Our routing algorithm selects, for Flow 1-13, the two routes shown on the figure: Route 1, a two-hop WiFi-PLC route, and Route 2, a single-hop PLC route. Flow 4-7 has a single-hop WiFi route, Route 3. During the first 1950 seconds, we send UDP saturated traffic with `iperf` on Flow 1-13. In Figure 9 (right), we show the traffic rate injected over each of the two routes 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 route (horizontal line), which is Route 1. Our congestion controller uses 100% of the capacity of Route 1. As this consumes only about 50% of the capacity of Route 2, it also injects a rate roughly equal to 50% of the available capacity on this route. Using two routes simultaneously provides an important gain in throughput (about 45% in this example) compared to a single route.

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 Route 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 routes: Observe the difference between the injected rate and the received throughput; when the injected rate is too large, the variance of the throughput increases (e.g., see the indicated "peak" on Figure 9). Although quite simple, this example shows that EMPoWER finds efficient routes and dynamically load-balances traffic in a way that explicitly accounts for interference.

6.3 Extensive Performance Evaluation

We now show the performance gains EMPoWER yields for an isolated flow, by broadly evaluating it on 50 randomly selected pairs of stations. As opposed to the simulations of Section 5, PHY layer conditions are not ideal: We use a small constraint margin $\delta = 0.05$ in (3). 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 with a brute-force approach, i.e., by

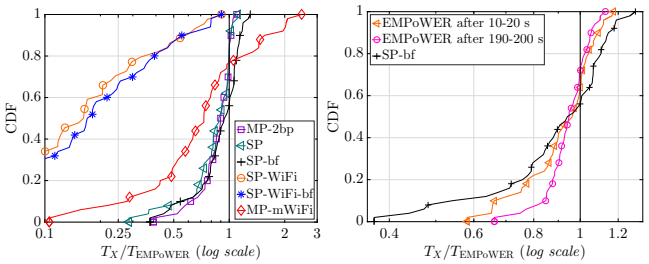

Figure 10: Distribution of T_X/T_{EMPoWER} (log scale).

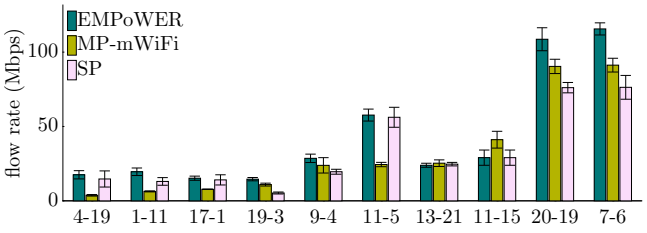

Figure 11: Average rate and standard deviation of throughput measurements during the last 100 seconds for EMPoWER, MP-mWiFi, and SP.

sending rates from 0 to the maximum possible rate with 0.25 MBps 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 10 (left), 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 naturally 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 5, 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. It 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.
- 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 routes; in the remaining 30% cases, our multipath-routing protocol returns only one route, 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 route as SP-bf, which means that the difference between EMPoWER and SP-bf does not come from using multiple routes, 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 10 (right), 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 we observe in our example of Section 6.2: 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 routes: 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 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).

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

Table 1: Download times for the four experiments.

Finally, in Figure 11, 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 routes. In general, multipath does not cause variations larger than single-path (see e.g., the results for Flows 20-19 and 7-6). The figure also further confirms our findings of Section 5: Compared to multi-channel WiFi, EMPoWER extends coverage by boosting performance especially for flows with poor connectivity (e.g., Flows 4-19

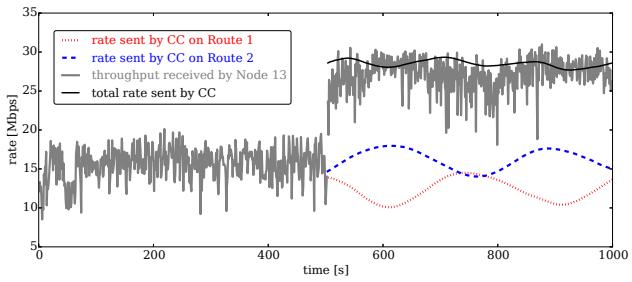

Figure 12: TCP for Flow 9-13. SP-w/o-CC from 0 s to 500 s; EMPoWER from 500 s to 1000 s.

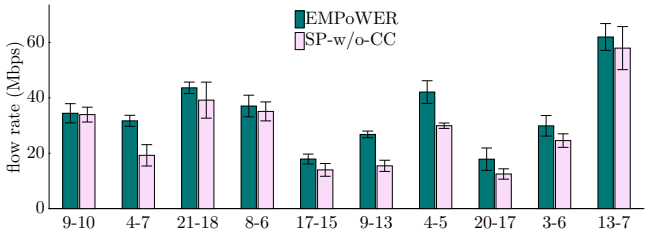

Figure 13: Average TCP rate, with standard deviation.

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.

6.4 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 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 (3); 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 routes. 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 route has delays much smaller than the other one, and packets sent on the fast route timeout while waiting for packets sent on the slow route. To improve performance, we add some delay on the fast route at the destination, so that both routes have approximately the same delays (see our Tech. Report [15] for implementation details). 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 12, we show how EMPoWER works with TCP for Flow 9-13. Traffic is sent during the first 500 s on Route 2 with TCP alone (SP-w/o-CC); during the next 500 s, our congestion controller is active and both Route 1 and Route 2 are used (EMPoWER). TCP acks are always sent on the best reversed route. Route 1 is a two-hop WiFi-WiFi route through Node 12. Route 2 is a three-hop PLC-PLC-WiFi route through Nodes 6 and 12. This is a "critical" case, as it uses routes 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 routes 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 13).

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 broadcasted price messages (described in Section 4.2): TCP and UDP flows can be handled efficiently at the same time.

7. RELATED WORK

We review related work on general routing and congestion control, multipath, and hybrid networks.

Congestion Control and Routing. From a theoretical point of view, several works propose routing and scheduling, building on the backpressure idea first proposed by Tassiulas and Ephremides [35]. In particular, some works propose utility maximizing schemes for wireless multi-hop networks [e.g., 5, 8, 23]. However, backpressure scheduling is NP-hard [13] 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* [29], 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 in this paper. *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. Furthermore, *Horizon* assumes that the paths are fixed and given in advance, whereas we also propose a routing algorithm that discovers efficient combinations of hybrid paths. Neely et al. [27] 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.

Congestion control and routing have been well studied in wireless networks. Many works propose routing algorithms for multi-channel ad-hoc wireless networks [e.g., 4, 7, 12, 44], but they do not consider the use of multiple paths. Other works have considered multiple paths for multi-channel wireless networks [e.g., 6, 10, 21, 36, 43], but they merely try to build maximally disjoint paths, whereas it can be more efficient to use a same link for several paths. There is also a large body of work for congestion control in multi-hop networks. EMPoWER borrows concepts from works that study wired networks [17, 25, 41]. 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., 14, 28], 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 [3, 22, 34, 45] are challenging to implement in practice (in particular, they require centralized decisions and time sychronization) and studied only through simulations. They also include channel assignment. In contrast, EMPoWER is fully distributed, which makes it amenable to implementation on a real testbed.

Multipath TCP. Multipath TCP (MPTCP) has attracted much attention as a means to efficiently spread TCP flows over several paths across the Internet [9]. A typical use case of MPTCP concerns mobile clients that have both a 4G and a WiFi connection to the Internet; promising results have also been obtained in datacenters [30]. MPTCP relies on a window-based multipath congestion-control to efficiently balance traffic and remain fair to TCP [18, 42]. However, MPTCP acts at layer 4 and targets end-to-end paths between end-hosts. As such, it requires both end-hosts to handle MPTCP connections. In addition, MPTCP requires that either end-hosts be multi-homed (i.e., have several network interfaces directly exposing different IP sub-stacks), or that all intermediate routers employ a solution such as equal-cost multipath routing [30], that does not capture the interference phenomena mentioned above. In practice, employing MPTCP means that two endpoints in the Internet have to handle several paths, even if these paths reside only within a local network. Moreover, the endpoints themselves are often not multi-homed (e.g., the laptop client in Figure 1), even though multiple paths exist. For example, in our testbed experiments presented in Section 6, 34% of source-destination pairs (the source playing the role of the gateway, the destination the role of the client) between which multiple paths exist would not support MPTCP, because the interface used by the client is common to the different paths. Finally, very few servers currently support MPTCP (less than 0.1% of the hosts in the Alexa top-1M list [26]). Contrary to MPTCP, EMPoWER acts at layer 2.5; it is therefore confined to the local network and transparent to other Internet hosts and protocols. It can be deployed wherever multiple paths exist, independently of the end-server. In addition, although it works distributively, EMPoWER provably maximizes a global utility function, contrary to any MPTCP-like solution.

PLC and Hybrid Networks. PLC networks and hybrid PLC/WiFi networks have received little attention from the research community, despite their commercial success. A few works explore the gains of introducing PLC in local networks. In previous works [38, 39], we compare WiFi and PLC in terms of coverage and capacity variability, and we introduce capacity estimation metrics and guidelines for PLC and WiFi. Tinnakornsrisuphap et al. [37] compare the coverage and capacity among hybrid WiFi/PLC, standalone WiFi, and standalone PLC networks. The authors show via simulations (employing measurements from actual houses to model PLC capacity, because of the challenging nature of PLC channel modelling) that hybrid networks significantly improve both coverage and capacity, and that the benefits of hybrid networks are high when multi-hop topologies are used. Finally, hybrid networks can also improve reliability by using cross-link replication; this is the basis for DiversiFi [16]. DiversiFi is implemented with multi-channel WiFi; due to medium diversity, such a solution in PLC/WiFi networks could improve further reliability.

8. CONCLUSION

Today's WiFi networks benefit from additional technologies, such as PLC, to eliminate blind spots and to relieve congestion. Although many commercial products combining diverse communication technologies exist on the market, the design and performance of hybrid solutions remain largely unexplored. We introduced EMPoWER: It comprises routing and congestion-control algorithms for exploiting multiple paths in hybrid networks. To fully exploit the gains enabled by the multiple technologies, we rely on a simple interference model that describes how links that employ the same technology share the available capacity. This model enables us to devise a congestion controller that converges to utility-optimal allocations in a distributed fashion. It also enables us to design a new multipath-routing algorithm that computes efficient combinations of paths for simultaneous use. As it is implemented at layer 2.5, EMPoWER can be deployed independently by local networks.

We 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 also substantiated the gains of introducing PLC in local networks. In particular, we found that due to medium diversity, hybrid PLC/WiFi improves throughput and coverage compared to multi-channel WiFi.

References

[1] IEEE Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications. *IEEE Std 1901-2010*, 2010.

[2] IEEE Standard for a Convergent Digital Home Network for Heterogeneous Technologies. *IEEE Std 1905.1-2013*, 2013.

[3] M. Alicherry, R. Bhatia, and L. E. Li. Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks. In *ACM MobiCom*, 2005.

[4] V. C. Borges, D. Pereira, M. Curado, and E. Monteiro. Routing Metric for Interference and Channel Diversity in Multi-Radio Wireless Mesh Networks. In *AdHoc-Now*, 2009.

[5] L. Chen, S. Low, M. Chiang, and J. Doyle. Cross-Layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks. In *IEEE INFOCOM*, 2006.

[6] P. Dong, H. Qian, K. Zhou, W. Lu, and S. Lan. A Maximally Radio-disjoint Geographic Multipath Routing Protocol for MANET. *Annals of Telecommunications*, 2015.

[7] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *ACM Mobicom*, 2004.

[8] A. 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.

[9] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses, 2013. RFC 6824.

[10] J. J. Gálvez, P. M. Ruiz, and A. F. Skarmeta. Multipath Routing with Spatial Separation in Wireless Multi-hop Networks Without Location Information. *ACM Computer Networks*, 2011.

[11] V. Gambiroza, B. Sadeghi, and E. W. Knightly. End-to-end Performance and Fairness in Multihop Wireless Backhaul Networks. In *ACM Mobicom*, 2004.

[12] M. Genetzakis and V. Siris. A Contention-Aware Routing Metric for Multi-Rate Multi-Radio Mesh Networks. In *IEEE SECON*, 2008.

[13] L. Georgiadis, M. J. Neely, and L. Tassiulas. Resource Allocation and Cross-layer Control in Wireless Networks. *Foundations and Trends in Networking*, 2006.

[14] A. Giannoulis, T. Salonidis, and E. Knightly. Congestion Control and Channel Assignment in Multi-radio Wireless Mesh Networks. In *IEEE SECON*, 2008.

[15] S. Henri, C. Vlachou, J. Herzen, and P. Thiran. EMPoWER Hybrid Networks: Exploiting Multiple Paths over Wireless and ElectRical Mediums. Technical report, Infoscience EPFL, 2016. https://infoscience.epfl.ch/record/221483/files/tech_report.pdf.

[16] R. Kateja, N. Baranasuriya, V. Navda, and V. N. Padmanabhan. DiversiFi: Robust Multi-link Interactive Streaming. In *ACM CoNEXT*, 2015.

[17] F. P. Kelly, A. K. Maulloo, and D. K. Tan. Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability. *Journal of the Operational Research Society*, 1998.

[18] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec. MPTCP is Not Pareto-optimal: Performance Issues and a Possible Solution. In *ACM CoNEXT*, 2012.

[19] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 2000.

[20] P. Kyasanur and N. H. Vaidya. Routing and Link-layer Protocols for Multi-channel Multi-interface Ad-hoc Wireless Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 2006.

[21] S. J. Lee and M. Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad-hoc Networks. In *IEEE ICC*, 2001.

[22] X. Lin and S. Rasool. A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-Hoc Wireless Networks. In *IEEE Infocom*, 2007.

[23] X. Lin and N. Shroff. Joint Rate Control and Scheduling in Multihop Wireless Networks. In *IEEE CDC*, 2004.

[24] X. Lin and N. Shroff. Utility Maximization for Communication Networks with Multipath Routing. *IEEE Transactions on Automatic Control*, 2006.

[25] S. H. Low and D. E. Lapsley. Optimization Flow Control: Basic Algorithm and Convergence. *IEEE/ACM Transactions on Networking*, 1999.

[26] O. Mehani, R. Holz, S. Ferlin, and R. Boreli. An Early Look at Multipath TCP Deployment in the Wild. In *Workshop on Hot Topics in Planet-Scale Measurement*, 2015.

[27] M. J. Neely, E. Modiano, and C.-P. Li. Fairness and Optimal Stochastic Control for Heterogeneous Networks. *IEEE/ACM Transactions on Networking*, 2008.

[28] A. H. M. Rad et al. Joint Optimal Channel Assignment and Congestion Control for Multi-channel Wireless Mesh Networks. In *IEEE ICC*, 2006.

[29] B. Radunović, C. Gkantsidis, D. Gunawardena, and P. Key. Horizon: Balancing TCP over Multiple Paths in Wireless Mesh Network. In *ACM Mobicom*, 2008.

[30] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *ACM SIGCOMM*, 2011.

[31] R. K. Sheshadri and D. Koutsonikolas. Comparison of Routing Metrics in 802.11n Wireless Mesh Networks. In *IEEE INFOCOM*, 2013.

[32] J. Sobrinho. Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet. 2001.

[33] R. Srikant. *The Mathematics of Internet Congestion Control*. 2004.

[34] W.-H. Tam and Y.-C. Tseng. Joint Multi-Channel Link Layer and Multi-Path Routing Design for Wireless Mesh Networks. In *IEEE INFOCOM*, 2007.

[35] L. Tassiulas and A. Ephremides. Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks. *IEEE Transactions on Automatic Control*, 1992.

[36] J. Y. Teo, Y. Ha, and C. K. Tham. Interference-Minimized Multipath Routing with Congestion Control in Wireless Sensor Network for High-Rate Streaming. *IEEE Transactions on Mobile Computing*, 2008.

[37] P. Tinnakornsrisuphap, P. Purkayastha, and B. Mohanty. Coverage and Capacity Analysis of Hybrid Home Networks. In *IEEE ICNC*, 2014.

[38] C. Vlachou, S. Henri, and P. Thiran. Electri-Fi Your Data: Measuring and Combining Power-Line Communications with WiFi. In *ACM IMC*, 2015.

[39] C. Vlachou, S. Henri, and P. Thiran. Electri-Fi Your Data: Measuring and Combining Power-Line Communications with WiFi. Technical report, Infoscience EPFL, 2015. https://infoscience.epfl.ch/record/210617/files/main_1.pdf.

[40] C. Vlachou, J. Herzen, and P. Thiran. Fairness of MAC protocols: IEEE 1901 vs. 802.11. In *IEEE ISPLC*, 2013.

[41] W.-H. Wang, M. Palaniswami, and S. H. Low. Optimal Flow Control and Routing in Multi-path Networks. *Performance Evaluation*, 2003.

[42] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. In *USENIX NSDI*, 2011.

[43] B. Yan and H. Gharavi. Multi-path Multi-channel Routing Protocol. In *IEEE International Symposium on Network Computing and Applications*, 2006.

[44] Y. Yang, J. Wang, and R. Kravets. Interference-aware Load Balancing for Multihop Wireless Networks. Technical report, 2005.

[45] L. Zhou, X. Wang, W. Tu, G.-M. Muntean, and B. Geller. Distributed Scheduling Scheme for Video Streaming over Multi-channel Multi-radio Multi-hop Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 2010.