Multi-Armed Bandit in Action: Optimizing Performance in Dynamic Hybrid Networks

Sébastien Henri, Christina Vlachou, and Patrick Thiran, *Fellow, IEEE*

***Abstract*— Today's home networks are often composed of several technologies such as Wi-Fi or power-line communication (PLC). Yet, current network protocols rarely fully harness this diversity and use each technology for a specific, pre-defined role, for example, wired media as a backbone and the wireless medium for mobility. Moreover, a single path is generally employed to transmit data; this path is established in advance and remains in use as long as it is valid, although multiple possible paths offer more robustness against varying environments. We introduce HyMAB, an algorithm that explores different multipaths and finds the best one in a mesh hybrid network, while keeping congestion under control. We employ the multi-armed-bandit framework and prove that HyMAB achieves optimal throughput under a static scenario. HyMAB design also accounts for real-network intricacies and dynamic conditions; it adapts to varying environments and switches multipaths when needed. We implement HyMAB on a PLC/Wi-Fi test bed. This is, to the best of our knowledge, the first implementation on a real test bed of multi-armed-bandit strategies in the context of routing. Our experimental results confirm the optimality of HyMAB and its ability to adapt to dynamic network conditions, as well as the gains provided by employing multi-armed-bandit strategies.**

***Index Terms*— Multi-armed bandit, dynamic networks, hybrid networks, wireless, power-line communications (PLC).**

I. Introduction

OVER the last few years, home networking has been facing several challenges due to the increasing number of mobile devices and the disruptive appearance of the Internet of Things (IoT) in everyday life. As a result, efforts have been made to improve coverage, throughput, and robustness in home networks. We focus our attention in particular on mesh networking and hybrid networks. First, mesh networking is gaining momentum, as it can effectively improve performance, and many commercial solutions are proposed. Mesh networking comes at the cost of an increased complexity compared to the infrastructure mode, because several paths can now be employed with potentially several hops. Second, it is possible to exploit the different technologies available, wired (e.g., PLC or Ethernet) and wireless (e.g., WiFi), which are referred to as hybrid networks. As a result of this trend, interoperation between different technologies is being standardized by IEEE 1905 [1], which takes place in layer 2.5, between IP and MAC layers. Hybrid networks do not only increase throughput; by exploiting the different characteristics of the underlying technologies, they can also provide substantial gains in terms of robustness against spatial and temporal variability.

In this paper, we consider specifically mesh networks with WiFi and PLC,[1] because they are very appealing solutions for home networks: They do not need any additional infrastructure to set up a network, and they provide medium diversity that enables better performance [2]–[4]. WiFi is now used in virtually all home networks. PLC is becoming increasingly popular in home networking, as it provides simple and high data-rate connectivity. By PLC, we refer to its most popular version, standardized by IEEE 1901 [5]. WiFi and PLC do not interfere with each other, but they both self-interfere,[2] a challenge that must be taken into account when designing reliable networking architectures.

Despite the prevalence of hybrid networks and the standardization attempts, these networks often operate far below their optimal performance in terms of throughput and latency, for the following three reasons: (i) Currently, the different technologies are often used for a dedicated and pre-determined roles, such as backbone or mobile scenarios. (ii) Current networks usually employ only one active path between two nodes and rarely take advantage of the benefits that multipath routing (i.e., using several paths *simultaneously*) can provide in certain scenarios, even for small-scale networks [2]. (iii) Whether they use single-path or multipath routing, current networks suffer from the static pre-establishment of the multipath, i.e., the set of paths (one or more) used simultaneously for a given flow, that remains active as long as it is valid. They cannot react to dynamic conditions that would require switching multipaths.

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: *Self-interfering 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,

Manuscript received November 24, 2017; revised May 24, 2018; accepted June 18, 2018; approved by IEEE/ACM Transactions on Networking Editor B. Krishnamachari. This work was supported by the Smart-World Project of the Hasler Foundation, Bern, Switzerland. *(Corresponding author: Sébastien Henri.)*

S. Henri and P. Thiran are with INDY, EPFL, 1015 Lausanne, Switzerland (e-mail: sebastien.henri@epfl.ch).

C. Vlachou is with HPE Labs, Palo Alto, CA 94304 USA.

Digital Object Identifier 10.1109/TNET.2018.2856302

[1]Our solution would still work with any combination of mesh media.

[2]IEEE 1901 employs a CSMA/CA scheme similar to that of 802.11; both are subject to interference.

the player chooses 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* (playing each machine to improve the estimates of the reward distributions) and *exploitation* (maximizing 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 [6], [7] or not [8], [9]. When applied to multipaths, MAB strategies accommodate I. However, to the best of our knowledge, no existing strategy has actually been implemented in 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 challenging [10]. Most protocols used in current networks are based on heuristics and are not guaranteed to be optimal.

The document is structured as follows: In Section II, we address the above-mentioned problem (i) and present a measurement-based method for computing the optimal rate on a multipath. This method accommodates self-interfering technologies and diverse interference graphs, thus addressing I 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 for finding the optimal rate in a mesh hybrid network. In Section III, we prove that HyMAB is optimal under static conditions. In Section IV, we show how HyMAB can also naturally adapt to dynamic conditions, thus confronting I, 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 V, showing the practical usability and performance gains of HyMAB. We discuss related work in Section VI and conclude in Section VII.

As a preliminary example, Figure 1 presents testbed experiments illustrating two typical use cases for HyMAB in dynamic conditions: capacity degradation (top, with a multipath of two paths) and a 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 MPTCP (MultiPath TCP [11], the most popular multipath solution today). In the bottom figure, HyMAB outperforms TCP protocols that were 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.

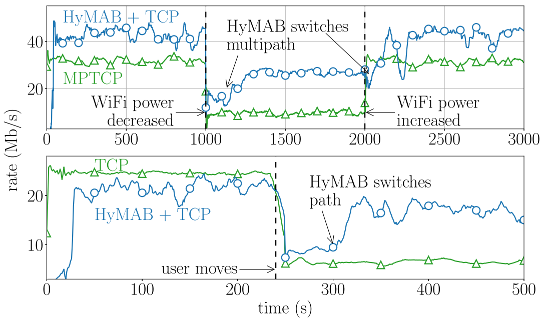

Fig. 1. HyMAB in action under dynamic conditions due to capacity drops (top, with a multipath of two paths) or mobility (bottom, with a single path). Results from two different flows in our hybrid WiFi/PLC testbed.

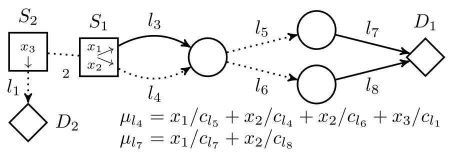
S_2
S_1
x_3
x_1
x_2
l_1
2
l_3
l_4
l_5
l_6
l_7
l_8
D_1
D_2
$\mu_{l_4} = x_1/c_{l_5} + x_2/c_{l_4} + x_2/c_{l_6} + x_3/c_{l_1}$
$\mu_{l_7} = x_1/c_{l_7} + x_2/c_{l_8}$

Fig. 2. Illustration of a multigraph, with two flows. Dotted lines represent WiFi, plain 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 $\Lambda_{P_3} = \{l_1\}$).

II. Optimal Rate of a Multipath

We describe our network model. It is applicable to every self-interfering technology and does not require knowing the specificities of the underlying physical layer. We then present a method for computing the optimal rate on a given multipath.

A. Network Model

We consider a home network with K different self-interfering technologies that do not interfere with each other (e.g., PLC, WiFi, LTE). 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. $\mathcal{E}$ is partitioned into K sets $\mathcal{E}_k$, $k \in \{1, \ldots, K\}$, 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 shows an example of a multigraph with $K = 2$ technologies, e.g., 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. c_l is a random variable with fixed mean $\overline{c}_l$ (in the following, $\overline{x}$ denotes the mean value of random variable x). 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, $\mathcal{I}_{l_4} = \{l_1, l_2, l_4, l_5, l_6\}$ and $\mathcal{I}_{l_7} = \{l_7, l_8\}$.

If a node transmits data to another node, we call the source-destination pair a *flow*. 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 III-E). A *path* is a self-avoiding path of the multigraph G that connects two nodes. The source of the flow can simultaneously use M paths $P_1, \ldots, P_M$; the set $\mathcal{P} = (P_1, \ldots, P_M)$ is called a *multipath*. When $M = 1$, the multipath is a single path. The set of links belonging to a path P_i is denoted by Λ_{P_i}, with $\Lambda_{P_i} \subseteq \mathcal{E}$; for a multipath $\mathcal{P} = (P_1, \ldots, P_M)$, 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, a possible multipath with $M = 2$ paths from source 1 to destination 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 $\mathcal{I}_l$, 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 [12] and PLC [5], 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:

$$\mu_l = \frac{x_l}{c_l}. \tag{1}$$

The validity of this assumption is discussed in Section II-C. When the link is saturated, then $\mu_l = 1$. Figure 2 illustrates the busy time with interfering links.

B. Computing the Optimal Rate

We present how the optimal rate on a given multipath can be computed. The results of this section are valid when a single flow is present; the extension to multi-flow is described in Section III-D.

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

$$\mu_{l,\boldsymbol{x}_{\mathcal{P}}} = \sum_{l' \in \mathcal{I}_l} \mu_{l'} = \sum_{i=1}^{M} x_i \sum_{l' \in \mathcal{I}_l \cap \Lambda_{P_i}} \frac{1}{c_{l'}} = \sum_{i=1}^{M} x_i \alpha_{P_i,l}, \tag{2}$$

where the parameter $\alpha_{P_i,l} = \sum_{l' \in \mathcal{I}_l \cap \Lambda_{P_i}} 1/c_{l'}$ is a metric that quantifies the impact of path P_i on the busy time for link l: If $\alpha_{P_i,l}$ is large, then a rate vector $\boldsymbol{x}_{\mathcal{P}}$ with a small rate x_i for P_i and $x_j = 0$ for $j \neq i$ yields a high busy time $\mu_{l,\boldsymbol{x}_{\mathcal{P}}}$, which means that P_i has a strong impact on the busy time for l (e.g., if P_i has a link with low capacity that interferes with l). A rate $\boldsymbol{x}_{\mathcal{P}}$ sent on a multipath $\mathcal{P}$ is *admissible* if for all $l \in \Lambda_{\mathcal{P}}$, $\mu_{l,\boldsymbol{x}_{\mathcal{P}}} \leq 1$ (the busy times of all links do not exceed 100%).

Writing $\boldsymbol{\alpha}_{\mathcal{P},l} \in \mathbb{R}^M$ for the vector $[\alpha_{P_i,l}]_{i \in \{1,\ldots,M\}}$, Equation (2) can be recast as $\mu_{l,\boldsymbol{x}_{\mathcal{P}}} = \boldsymbol{\alpha}_{\mathcal{P},l}^T \cdot \boldsymbol{x}_{\mathcal{P}}$ with T denoting transposition. $\boldsymbol{\alpha}_{\mathcal{P},l}$ is called the *multipath-impact vector* of $\mathcal{P}$ on l; it depends only on the network topology and on

TABLE I
NOTATION

$\mathcal{E}_k$; $\mathcal{E}$	set of links of technology k; $\bigcup_{k=1}^{K} \mathcal{E}_k$
c_l, I_l, μ_l	capacity, interference domain, busy-time of link l
$\mathcal{P} = (P_1, \ldots, P_M)$	multipath with M paths $P_1, \ldots, P_M$
Λ_P; $\Lambda_{\mathcal{P}}$	links that constitute path P; $\bigcup_{i=1}^{M} \Lambda_{P_i}$
$\alpha_{P,l}$	$\sum_{l' \in \mathcal{I}_l \cap P} 1/c_{l'}$
$\boldsymbol{\alpha}_{\mathcal{P},l} \in \mathbb{R}^M$	multipath-impact vector $[\alpha_{P_i,l}]_{i \in \{1,\ldots,M\}}$
$\boldsymbol{A}_{\mathcal{P}} \in \mathbb{R}^{L_{\mathcal{P}} \times M}$	matrix $[\boldsymbol{\alpha}_{\mathcal{P},l}^T]_{l \in \Lambda_{\mathcal{P}}}$
$\boldsymbol{x}_{\mathcal{P}} \in \mathbb{R}^M$	vector of rates sent on multipath $\mathcal{P}$
$\boldsymbol{X}_{\mathcal{P}} \in \mathbb{R}^{M \times M}$	matrix $[\boldsymbol{x}_{\mathcal{P}}^{(i)T}]_{i \in \{1,\ldots,M\}}$
$\mathcal{S}$	set of available multipaths ($\lvert\mathcal{S}\rvert = N$)

the paths, and *not* on the rate vector $\boldsymbol{x}_{\mathcal{P}}$. For example, the multipath-impact vectors of the multipath $\mathcal{P} = (P_1, P_2)$ from S_1 to D_1 in Figure 2 are $\boldsymbol{\alpha}_{\mathcal{P},l_4}^T = \left[1/c_{l_5}\ 1/c_{l_4} + 1/c_{l_6}\right]$ and $\boldsymbol{\alpha}_{\mathcal{P},l_7}^T = \left[1/c_{l_7}\ 1/c_{l_8}\right]$. We denote by $\boldsymbol{A}_{\mathcal{P}} \in \mathbb{R}^{L_{\mathcal{P}} \times M}$ the matrix $[\boldsymbol{\alpha}_{\mathcal{P},l}^T]_{l \in \Lambda_{\mathcal{P}}}$ and by $\boldsymbol{\mu}_{\boldsymbol{x}_{\mathcal{P}}} \in \mathbb{R}^{L_{\mathcal{P}}}$ the vector $[\mu_{l,\boldsymbol{x}_{\mathcal{P}}}]_{l \in \Lambda_{\mathcal{P}}}$. Table I summarizes the main notations of this paper.

If the multipath-impact vectors $\boldsymbol{\alpha}_{\mathcal{P},l}$ are known for all links $l \in \Lambda_{\mathcal{P}}$, it is easy to find an optimal rate $\boldsymbol{x}_{\mathcal{P}}^{\text{opt}}$: It is a maximum rate (in the sense of the 1-norm) that is admissible. Because $\boldsymbol{A}_{\mathcal{P}} \cdot \boldsymbol{x}_{\mathcal{P}} = \boldsymbol{\mu}_{\boldsymbol{x}_{\mathcal{P}}}$, $\boldsymbol{x}_{\mathcal{P}}^{\text{opt}}$ is a solution of the following system:

$$\begin{aligned} &\max_{\boldsymbol{x}} \ \mathbf{1}^T \cdot \boldsymbol{x} \\ &\text{subject to } \boldsymbol{A}_{\mathcal{P}} \cdot \boldsymbol{x} \ \preceq \ \mathbf{1} \text{ and } \boldsymbol{x} \succeq \mathbf{0}. \end{aligned} \tag{3}$$

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

But directly computing the multipath-impact vectors $\boldsymbol{\alpha}_{\mathcal{P},l}$ would require knowing the link capacity c_l and the interference domain $\mathcal{I}_l$ of all links $l \in \Lambda_{\mathcal{P}}$, which is challenging or impractical [13], 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 $\mu_{l,\boldsymbol{x}_{\mathcal{P}}}$ when the source sends traffic on $\mathcal{P}$ at rate $\boldsymbol{x}_{\mathcal{P}}$: 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 II-C). In Section V-A, we describe how the source of $\mathcal{P}$ gathers the busy-time measurements $\mu_{l,\boldsymbol{x}_{\mathcal{P}}}$ for all $l \in \Lambda_{\mathcal{P}}$. Once the source knows the busy-time measurements, it is easy to get $\boldsymbol{\alpha}_{\mathcal{P},l}$. 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 (2) by $\alpha_{P,l} = \mu_{l,x_P}/x_P$. This result can easily be extended when there are $M \geq 2$ paths in the multipath $\mathcal{P}$. We choose M linearly independent rate vectors $\boldsymbol{x}_{\mathcal{P}}^{(i)}$ for $i \in \{1, \ldots, M\}$, and for each rate vector $\boldsymbol{x}_{\mathcal{P}}^{(i)}$ at which traffic is sent on multipath $\mathcal{P}$ and each link $l \in \Lambda_{\mathcal{P}}$, the corresponding busy time $\mu_l^{(i)}$ is measured. If $\boldsymbol{X}_{\mathcal{P}}$ denotes the $M \times M$ matrix of the rate vectors $\boldsymbol{X}_{\mathcal{P}} = [\boldsymbol{x}_{\mathcal{P}}^{(i)T}]_{i \in \{1,\ldots,M\}}$ 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 $\boldsymbol{X}_{\mathcal{P}} \cdot \boldsymbol{\alpha}_{\mathcal{P},l} = \boldsymbol{\mu}_l$. By construction, the $\boldsymbol{x}_{\mathcal{P}}^{(i)}$'s are linearly independent, hence $\boldsymbol{X}_{\mathcal{P}}$ is a $M \times M$ full-rank matrix, and we get $\boldsymbol{\alpha}_{\mathcal{P},l} = \boldsymbol{X}_{\mathcal{P}}^{-1} \cdot \boldsymbol{\mu}_l$ for all links $l \in \Lambda_{\mathcal{P}}$, i.e., we get the matrix $\boldsymbol{A}_{\mathcal{P}}$. We then solve the linear system (3) using standard techniques, which gives the optimal rate $\boldsymbol{x}_{\mathcal{P}}^{\text{opt}}$

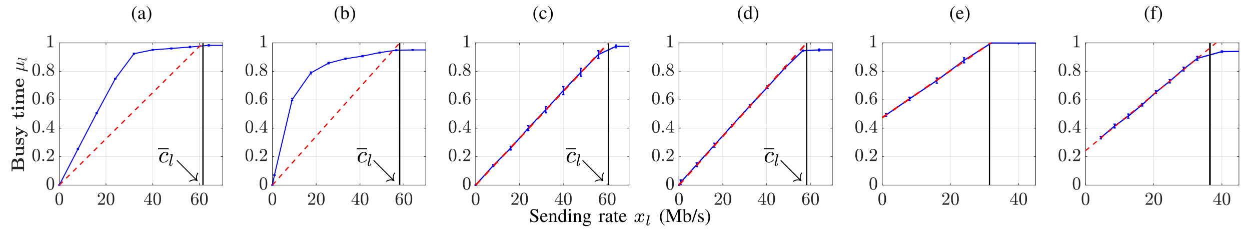

Fig. 3. Busy time μ_l versus rate x_l sent on link l. The experiments are repeated 50 times for each rate x_l; each point represents the average busy time μ_l, with the bars representing standard deviations. (a) WiFi, no batch. (b) PLC, no batch. (c) WiFi, $B = 100$. (d) PLC, $B = 200$. (e) WiFi, 3 links. (f) PLC, 3 links.

achievable on the multipath $\mathcal{P}$ without having to measure the link capacities or the interference domains.

C. Linearity of the Relation Busy Time vs. Rate

The key assumption made in the previous subsection for computing the optimal rate is the linear relationship (1). To evaluate its validity, we carry out the following experiment on our testbed (described further in Section V-A): 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. For WiFi, ath9k drivers expose directly the measured busy-times that can be accessed using netlink sockets [14]. For PLC, the node sniffs all packets using `faifa` [15], and uses the *duration* field of every PLC packet to compute the busy time. The experiment is repeated 50 times for each rate x_l, for both WiFi (Figure 3a) and PLC (Figure 3b). The link capacity $\overline{c}_l$, indicated by the black vertical line, is the maximum rate received by the destination. The dotted red line indicates the linear relationship (1), clearly not valid in this experiment.

The invalidity of the linear relationship (1) comes from the introduction, for increasing throughput, of frame aggregation in recent standards (e.g., IEEE 802.11n/ac for WiFi, IEEE 1901 for PLC). At low rates, frame aggregation is barely used, because the interval between two consecutive packets is too large for the network interface to wait for packets to aggregate. In contrast, it is fully used when sending saturated traffic. Having two different operating regimes invalidates the linear relationship (1). To make it valid, we 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 Figures 3c (WiFi) and 3d (PLC), we see that the linear relationship (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. Compared to Figures 3a and 3b, we see an increase of the standard deviation, in particular for WiFi: In order to be precise enough, the results need to be averaged on several measurements.

The relation busy time vs. rate is also linear when more than one node transmits, which justifies (2). We repeat the same experiment and send traffic at various rates on link l; two other nodes of $\mathcal{I}_l$ now contend and transmit traffic at constant rate. The transmitting nodes send traffic in batches of $B=100$ packets for WiFi, $B=200$ for PLC. In Figures 3e (WiFi) and 3f (PLC), we show 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.

Sending packets in batches 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 in batches is needed only during probing phases, and not during exploitation phases (see next section).

These results confirm that the method described in Section II-B can be used to compute the optimal rate of a path. But busy-time measurements are noisy, especially because packets need to be sent by batches during the probing phases. Therefore, they need to be repeated several times and averaged out. This means that one measurement is not enough, and this challenge motivates the use of a MAB strategy.

III. HyMAB Under Static Conditions

In this section, we discuss static conditions. We describe why MAB strategies are adapted for finding the best multipath and how they can be made practical. We present HyMAB for a single flow and show analytically that it achieves optimal throughput. We then extend HyMAB to efficiently accommodate several concurrent flows, and discuss the case of non-saturated flows.

A. Towards a Practical MAB Strategy

Finding the best multipath $\mathcal{P}^*$ is a difficult problem; in fact, it has been shown that in a network with interference, such as for WiFi and PLC, it is NP-hard [10]. 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 II-B, to compute the optimal rate of several multipaths, and to keep the best multipath as the one enjoying the maximum rate. However, as we have seen in Section II-C, the busy times (hence, the computation of the optimal rate) require several measurements to be precise (i.e., one measurement is not sufficient). Consequently, there is a conflict between exploring several multipaths to estimate with sufficient precision the optimal rate of each of them, 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 [8]. Here, we employ multipath routing, i.e., several paths can be employed simultaneously. Combinatorial MABs [16], [17] typically study the problem of routing. Gai et al. [16] find the shortest path in a graph with varying link capacities. However, with interfering links, finding the path with highest throughput is NP-hard [10], 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. [17] 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, in this work, we consider 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 V, 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 $\mathcal{S}$ 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 [7]–[9], 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 in 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 II-B. 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 [18] 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 [6], this issue is solved by introducing an exploration probability $\epsilon(t)$ that is a decreasing function of the time t, equal to

$$\epsilon(t) = \min\left(1, \frac{cN}{d^2 t}\right), \tag{4}$$

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 [6]. 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 Section V-B). The limitations of ϵ-greedy and ϵ_n-greedy are illustrated through simulations in Section III-C.

Other strategies have been introduced to deal with this inefficiency, in particular Upper Confidence Bound (UCB) [7]. 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 III-B. It uses UCB strategies as a subroutine.

HyMAB is proven optimal under static conditions in Section III-B. Nevertheless (see Sections IV and V), 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 III), the probing probability needs to go to zero to ensure optimality; under dynamic conditions (Section IV), the probing probability needs to stay away from zero to adapt continuously to dynamic environments. This tradeoff is studied in Section IV-A. Many works study MAB when the rewards vary dynamically [8], [9], 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 [19] 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 [20] or independent [21], [22] 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. [23] 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 (Section III-D) reduces the problem with F different flows to F independent problems with a single flow and N arms.

B. 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, 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. [6]. We adopt this strategy because it is easy to implement and because, 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 III-A, a probing phase is required to obtain the reward of an arm, 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 $\boldsymbol{x}_{\mathcal{P}}$ are scaled so that for all $\mathcal{P}$ and t, $\|\boldsymbol{x}_{\mathcal{P}}(t)\|_1 \le 1$. 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 $\epsilon_{\mathcal{P}}(t) = 1/n^{\lambda}_{\mathcal{P}}(t-1)$, with a parameter λ that controls the rate of convergence. We study the effects of λ in Section III-C. 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 in particular that HyMAB is robust against the choice of λ, in the sense that it does not impact drastically the performance.

Theorem 1: With $\epsilon_{\mathcal{P}}(t) = 1/n^{\lambda}_{\mathcal{P}}(t-1)$, HyMAB converges to achieving optimal throughput for any $\lambda > 0$.

Algorithm 1 HyMAB: Strategy for Optimal Throughput

```
Input: trial duration $D$, set of multipaths $\mathcal{S}$ with $|\mathcal{S}| = N$.
Initialize: for all $\mathcal{P} \in \mathcal{S}$, $n_{\mathcal{P}}(0) = 0$, $T_{\mathcal{P}}(0) = 0$.
For each trial t:
  For all $\mathcal{P} \in \mathcal{S}$, $n_{\mathcal{P}}(t) = n_{\mathcal{P}}(t-1)$, $T_{\mathcal{P}}(t) = T_{\mathcal{P}}(t-1)$.
  Stage 1 Choose one arm (i.e., one multipath) $\mathcal{P}_t \in \mathcal{S}$ according to UCB1 strategy:
      ◇ if there are non-explored arms, choose one among them,
      ◇ otherwise ($t > N$, $n_{\mathcal{P}} > 0$), choose the arm maximizing
            $V_{\mathcal{P}}(t) \doteq \left\|\overline{\boldsymbol{x}}_{\mathcal{P},n_{\mathcal{P}}(t-1)}\right\|_1 + \sqrt{\frac{2\ln(t-1)}{n_{\mathcal{P}}(t-1)}}.$   (5)
      ◇ Set $T_{\mathcal{P}_t}(t) \leftarrow T_{\mathcal{P}_t}(t) + 1$.
  Stage 2 Choose one of the following:
      ◇ with probability $\epsilon_{\mathcal{P}_t}(t)$, probe the arm $\mathcal{P}_t$ by following the procedure described in Section II-B, i.e.,
          • set $n_{\mathcal{P}_t}(t) \leftarrow n_{\mathcal{P}_t}(t) + 1$, and
          • choose $M$ rate vectors $\boldsymbol{x}^{(i)}$, $i \in \{1,\dots,M\}$, that are admissible and linearly independent, and
          • send traffic at rates $\boldsymbol{x}^{(i)}$ during $D/M$ seconds, in turn for $i \in \{1,\dots,M\}$, and
          • compute the reward $\boldsymbol{x}_{\mathcal{P}_t,n_{\mathcal{P}_t}(t)}$ and the current rate estimation $\overline{\boldsymbol{x}}_{\mathcal{P}_t,n_{\mathcal{P}_t}(t)}$.
      ◇ or with probability $1-\epsilon_{\mathcal{P}_t}(t)$, exploit the current best arm $\mathcal{P}^*_t$ maximizing $\left\|\overline{\boldsymbol{x}}_{\mathcal{P},n_{\mathcal{P}}(t-1)}\right\|_1$, i.e., send traffic at rates $\overline{\boldsymbol{x}}_{\mathcal{P}^*_t,n_{\mathcal{P}^*_t}(t-1)}$ on $\mathcal{P}^*_t$ during $D$ seconds.
```

Proof: The proof is given in Appendix. ■

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 discussed in Section V-A.

C. 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 $\left\|\boldsymbol{x}^{\text{opt}}_{\mathcal{P}}\right\|_1$ for the multipaths $\mathcal{P} \in \mathcal{S}$ 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 III-A. 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 presented in Section V. An estimation $\boldsymbol{x}_{\mathcal{P}_t,n_{\mathcal{P}_t}(t)}$ is computed by adding to the true value $\left\|\boldsymbol{x}^{\text{opt}}_{\mathcal{P}}\right\|_1$ a random gaussian noise of mean 0 and standard deviation $0.2\left\|\boldsymbol{x}^{\text{opt}}_{P}\right\|_1$, close to what we observe in our testbed experiments. We study the number of trials needed for the

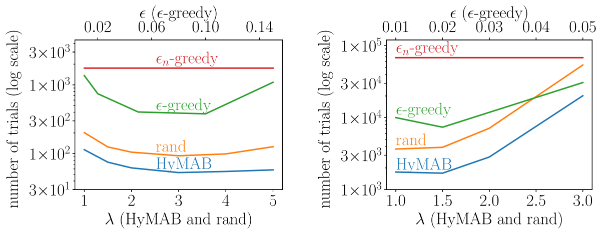

Fig. 4. Number of trials needed to reach 85% (left) and 95% (right) of the optimal rate (log scale).

averaged sending rate to reach 85% (Figure 4, left) and 95% (Figure 4, right) of the optimal rate $\left\|\boldsymbol{x}^{\text{opt}}_{\mathcal{P}^*}\right\|_1$, 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, e.g., 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 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 IV-A).

Figure 4 also shows the effects of the parameter λ in the exploration probability $\epsilon_{\mathcal{P}}(t)$ 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 (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 remaining of the paper.

Finally, we compare HyMAB with two other algorithms, ϵ-greedy [18] and ϵ_n-greedy [6], described in Section III-A. 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 (4) is replaced with $\epsilon_{\mathcal{P}}(t)$ defined in Theorem 1. Figure 4 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.

D. 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 $\mathcal{F}$ be the set of flows, i.e., of source-destination pairs. Each flow $f \in \mathcal{F}$ employs its own set of multipaths, denoted by $\mathcal{S}_f$. 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,

$$F_l = \#\{f \in \mathcal{F} \text{ s.t. } \exists P \in \mathcal{S}_f, l' \in \Lambda_P \text{ with } l \in \mathcal{I}_{l'}\}. \quad (6)$$

Instead of (3), each source of a flow f solves the system:

$$\begin{aligned} &\max_{\boldsymbol{x}} \ \mathbf{1}^T \cdot \boldsymbol{x} \\ &\text{subject to } \boldsymbol{A}_{\mathcal{P}} \cdot \boldsymbol{x} \preceq \mathbf{1}/\boldsymbol{F}_l \text{ and } \boldsymbol{x} \succeq \mathbf{0}, \end{aligned} \quad (7)$$

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 $\boldsymbol{A}_{\mathcal{P}}$ is more complex, because the busy times are now generated by all flows: $\mu_l = \sum_{f\in\mathcal{F}} \boldsymbol{\alpha}^T_{\mathcal{P}_{t,f},l} \cdot \boldsymbol{x}_{\mathcal{P}_{t,f}}$ where $\mathcal{P}_{t,f}$ is the multipath currently used by flow f. To compute $\boldsymbol{A}_{\mathcal{P}}$, i.e., to compute $\boldsymbol{\alpha}_{\mathcal{P},l}$ for each link $l \in \Lambda_{\mathcal{P}}$, we assume that during an exploration trial for a flow $f_0 \in \mathcal{F}$, 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 IV-B. Before an exploration of multipath $\mathcal{P}_0 \in \mathcal{S}_{f_0}$, the source of f_0 does not send traffic during a short time-slot (*silent slot*); each link $l \in \Lambda_{\mathcal{P}_0}$ measures the busy time during this time slot $\mu^{(0)}_{l,f_0} = \sum_{f\in\mathcal{F}\setminus f_0} \boldsymbol{\alpha}^T_{\mathcal{P}^*_{t,f},l} \cdot \boldsymbol{x}_{\mathcal{P}^*_{t,f}}$, where $\mathcal{P}^*_{t,f}$ 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 $\boldsymbol{X}_{\mathcal{P}_0}$, as described in Section II-B; each link measures the busy-time vector $\boldsymbol{\mu}_l$. If the media are not saturated during the exploration trial, $\mu_l - \mu^{(0)}_{l,f_0} = \boldsymbol{\alpha}^T_{\mathcal{P}_0,l} \cdot \boldsymbol{x}_{\mathcal{P}_0}$ because for all $f \neq f_0, \mathcal{P}_{t,f} = \mathcal{P}^*_{t,f}$, and $\boldsymbol{A}_{\mathcal{P}}$ can be computed by solving $\boldsymbol{\alpha}_{\mathcal{P},l} = \boldsymbol{X}^{-1}_{\mathcal{P}_0} \cdot (\boldsymbol{\mu}_l - \mu^{(0)}_{l,f_0}\mathbf{1})$: When we subtract the busy time $\mu^{(0)}_{l,f_0}$ due to all other flows, which we assume to remain constant along the exploration trial, the linear relationship (1) ensures that we get the busy time that f_0 would generate in the absence of transmission from other flows. In Section IV-B, we discuss how we guarantee that the media stay unsaturated during an exploration trial in the presence of multiple flows. Problem (7) is equivalent to (3); this means that each source can apply HyMAB and converge to the rate allocation that maximizes (7). 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.

E. 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 paper. With external low-throughput flows, HyMAB converges to an optimal and fair utilization of the remaining resources, without affecting the external flows.

IV. HyMAB Under Dynamic Conditions

In Section III, 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 1) that naturally deals with such variations; in Section V-D, 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).

A. Capacity Changes

In Section III, we assumed static conditions; to ensure optimality, the probing probability of a multipath $\mathcal{P}$, $\epsilon_{\mathcal{P}}(t)$, 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 $\epsilon_{\mathcal{P}}(t)$ in Theorem 1 by $\epsilon_{\mathcal{P}}(t) = \max(\epsilon_{\min}, 1/n_{\mathcal{P}}^{\lambda}(t-1))$, so that $\epsilon_{\mathcal{P}}(t)$ never goes below a threshold $\epsilon_{\min}$. This differs from the ϵ-strategy described in Section III-A 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 $\overline{\boldsymbol{x}}_{\mathcal{P}}$ over the last 10 measurements,

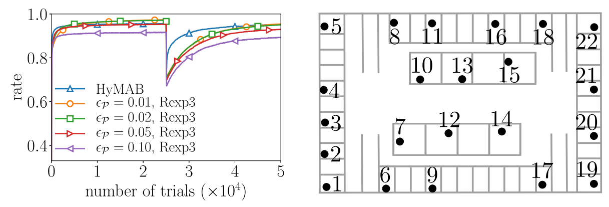

Fig. 5. Left: rate per trial for HyMAB and Rexp3 (simulations). Right: Map of our hybrid WiFi/PLC testbed (65×40 m).

instead of averaging over all measurements. Finally, similarly as the D-MAB algorithm [19], we reset the indicators $n_{\mathcal{P}}$ and $\boldsymbol{x}_{\mathcal{P}}$ of an arm $\mathcal{P}$ if three consecutive large variations of $\boldsymbol{x}_{\mathcal{P}}$ are observed (if $\|\boldsymbol{x}_{\mathcal{P}}(t)\|_1$ differs from the current average $\|\overline{\boldsymbol{x}}_{\mathcal{P}}\|_1$ 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 $\mathcal{P}$ are reset, because a capacity change for $\mathcal{P}$ 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. For more flexibility, it is possible to dynamically change 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 $\|\boldsymbol{x}_{\mathcal{P}}(t)\|_1$ differs from the current average $\|\overline{\boldsymbol{x}}_{\mathcal{P}}\|_1$ by more than 40%, $\epsilon_{\min}$ is set to a maximum value of 0.15. If $\|\boldsymbol{x}_{\mathcal{P}}(t) - \overline{\boldsymbol{x}}_{\mathcal{P}}\|_1$ is within 20% of $\|\overline{\boldsymbol{x}}_{\mathcal{P}}\|_1$, $\epsilon_{\min}$ 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 [9], 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 III-A, 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 $\epsilon_{\mathcal{P}}$ (Stage 2 of Algorithm 1). We use the same simulation scenario as in Section III-C, 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 5 (right) shows the throughput experienced at each trial (averaged over 10 000 random instances) for HyMAB with the variable $\epsilon_{\min}$ strategy

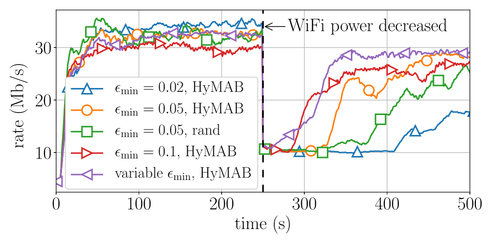

Fig. 6. Effect of ϵ_{min} with dynamic conditions (testbed experiments).

and $\lambda = 2$, and for Algorithm 1 with Rexp3 (denoted by Rexp3) 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 5, right 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 [4]) and WiFi (due to varying signals [24]). The experiment is repeated five times for each value of ϵ_{min}, and we present averaged results. In Figure 6, 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 ϵ_{min} is, the larger the convergence time is. The throughput obtained with a variable ϵ_{min}, as described above, is shown by the purple line in Figure 6. Before the capacities change, it converges to the same throughput as the fixed value $\epsilon_{\text{min}} = 0.05$ (the minimum value of the variable ϵ_{min}). When the capacities change, it adapts faster than all fixed-value strategies do.

Figure 6 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 $\epsilon_{\text{min}} = 0.05$. 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 III-C, 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 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).

B. Traffic Changes

HyMAB must also handle flows that come and go. The strategy for several flows (Section III-D) 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 at each link (local information), which makes it possible to react very rapidly to flow arrivals/departures: Once per trial, nodes along a multipath $\mathcal{P} \in \mathcal{S}_f$ notify the source of flow f of the maximal number of interfering flows $F_{\mathcal{P}} = \max_{l \in \mathcal{P}} F_l$, with F_l given by (6), along with a *flow hash*, a hash value of all the other flows that the nodes on $\mathcal{P}$ overhear. If $F_{\mathcal{P}}$ is increased, the flows, which had a share $1/F_{\mathcal{P}}^{\text{old}}$ of the time-resources, now have only $1/F_{\mathcal{P}}^{\text{new}}$: The source immediately reacts by scaling down the current sending rate vector $\overline{\boldsymbol{x}}_{\mathcal{P}}$ by a factor $F_{\mathcal{P}}^{\text{old}}/F_{\mathcal{P}}^{\text{new}}$. 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 maximizing (7). 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 III-D and converge to the vector maximizing (7). 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 $\mathcal{P}_1$, control messages are sent on each path of $\mathcal{P}_1$: Nodes in the interference domain of $\Lambda_{\mathcal{P}_1}$ can overhear these control messages, and thus know that f_1 is probing. If another flow f_2 wants to probe a multipath $\mathcal{P}_2$, nodes belonging to $\mathcal{P}_2$ 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.[3] This guarantees that the strategy with several flows converges to the optimal value.

If $F_{\mathcal{P}}$ 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 V-C.

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

TABLE II
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

V. EXPERIMENTAL EVALUATION

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

A. Experimental Framework and Implementation Details

We implement HyMAB with *Click*[4] [25] on a 22-node testbed located on one floor of an office building (see Figure 5, right). We use $K = 2$ technologies, WiFi and PLC. All the nodes have a WiFi interface (Atheros AR9280), and all but mobile nodes have a HomePlug AV PLC interface (QCA 7420) connected to the electrical network of our building. The nodes are APU1D boards with an OpenWrt distribution patched for MPTCP [26] and ath9k wireless drivers. 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 user-space 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 $\mathcal{S}$, we build on an existing single-path routing protocol for hybrid networks [27] and modify it to return multipaths of M or fewer paths. This multipath-routing protocol is described in our previous work [2]. It computes an estimate of the total capacity of multipaths and returns a set $\mathcal{S}$ of N multipaths, in decreasing order of the estimated multipath capacities. 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 by using link-quality information present in the packet headers [4]. 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 $\mathcal{S}$ is chosen, HyMAB does *not* require the knowledge of the link capacities or of the interference domains, because it uses only the busy-time measurements to compute the optimal rates of the multipaths. When a capacity change is detected (see Section IV-A), $\mathcal{S}$ is computed again.

In our testbed, we observe that the number of paths in the best multipath $\mathcal{P}^*$ is equal to at most the number of non-interfering technologies. We have proven this proposition for certain classes of networks [28]. 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 $\mathcal{S}$ 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 $\mathcal{P}^* \in \mathcal{S}$, as can be observed in the following experiment: We randomly choose 50 flows (i.e.,

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

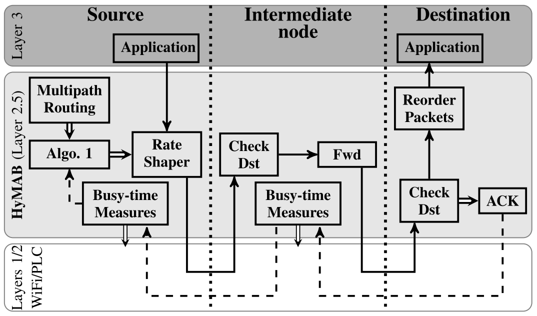

Fig. 7. The main components of HyMAB at layer 2.5. Plain-line arrows represent the data flow, dashed-line arrows represent the acknowledgements, and double arrows represent actions on the component. The source determines the multipath set $\mathcal{S}$ (Multipath Routing). $\mathcal{S}$ is used by Algo. 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).

source-destination pairs). 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 II 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 7 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

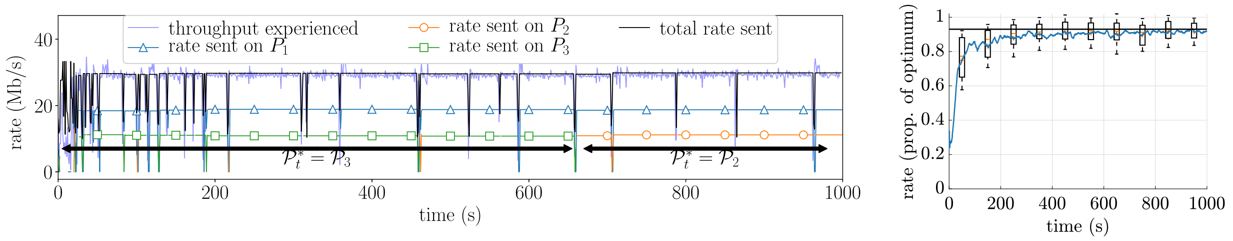

Fig. 8. Left: Single experiment for Flow 17-6. The throughput experienced is shown along with the rate sent on the different paths and the total rate sent. Throughput drops correspond to probing phases, more frequent for small t ($t \leq 50$ s, when optimal rates are unknown). After 50 s, exploitation of the best found multipath dominates. Right: Comparison of HyMAB with optimal brute-force rate, UDP traffic, for 50 flows.

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 $\mathcal{P}$ and either to probe $\mathcal{P}$ (with probability $\epsilon_{\mathcal{P}}(t)$), or to exploit 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 $\overline{\boldsymbol{x}}_{\mathcal{P}_t^*,t}$ on the estimated best multipath $\mathcal{P}_t^*$ during the entire slot. During an exploitation phase, the source does *not* send traffic in 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 $\boldsymbol{A}_{\mathcal{P}} \cdot \boldsymbol{x} \preceq \mathbf{1}$ in (3) by a slightly more conservative constraint $\boldsymbol{A}_{\mathcal{P}} \cdot \boldsymbol{x} \preceq (1-\delta) \cdot \mathbf{1}$, with a small *constraint margin* $0 \leq \delta \ll 1$. Increasing δ decreases the delays, but also decreases the throughput.

If instead the source chooses to probe multipath $\mathcal{P}$ at time t, it follows the procedure described in Section II-B. For the probing phases, the source uses rate vectors $\boldsymbol{x}_{\mathcal{P}}^{(i)}$ such that all vector components of $\boldsymbol{x}_{\mathcal{P}}^{(i)}$ 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 II-B. This choice of rate vectors $\boldsymbol{x}_{\mathcal{P}}^{(i)}$ 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 $\boldsymbol{X}_{\mathcal{P}}$ in Section II-B) 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 $\mathcal{P}$. 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 $\mathcal{P}$. These control messages enable all nodes along the paths to initialize and measure the busy times $\mu_l^{(i)}$ for all links $l \in \Lambda_{\mathcal{P}}$, as described in Section II-C. After the silent slot (described in Section III-D), the source starts by sending traffic at rate $\boldsymbol{x}_{\mathcal{P}}^{(1)}$ during D/M, in batches of 100 packets for WiFi, 200 for PLC. The destination then sends an acknowledgement back on all the paths of the multipath $\mathcal{P}$. 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 $\boldsymbol{x}_{\mathcal{P}}^{(i)}$. At the end of the trial, the source knows all measurements $\mu_l^{(i)}$ and can compute all $\boldsymbol{\alpha}_{\mathcal{P},l}$, as described in Section II-B. Finally, to compute an estimation of the optimal rate, the source reduces $\boldsymbol{A}_{\mathcal{P}}$ by removing all vectors $\boldsymbol{\alpha}_{\mathcal{P},l}$ such that $\boldsymbol{\alpha}_{\mathcal{P},l} \preceq \boldsymbol{\alpha}_{\mathcal{P},l'}$ for another link l', and solves (3) with the reduced $\boldsymbol{A}_{\mathcal{P}}$. In our experiments, the number of rows in the reduced $\boldsymbol{A}_{\mathcal{P}}$ 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.

B. 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 V-A. 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. Hereafter, 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 V-B, where we study the convergence of HyMAB to the optimal value, we use a fixed $\epsilon_{\text{min}} = 0.02$; in the following sections, we use a variable ϵ_{min}, as described in Section IV-A.

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 $\mathcal{P}_1, \ldots, \mathcal{P}_5$, ordered by decreasing estimated-capacity. The rates obtained by brute-force approach ($\mathcal{P}_1$: 23 Mb/s, $\mathcal{P}_2$: 30 Mb/s, $\mathcal{P}_3$: 29.5 Mb/s, $\mathcal{P}_4$: 20 Mb/s, and $\mathcal{P}_5$: 15 Mb/s) indicate that the best path for the routing protocol, $\mathcal{P}_1$, 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 8 (left) shows the experiment for Flow 17-6 with HyMAB. $\mathcal{P}_2$ consists of two paths, denoted by P_1 and P_2. $\mathcal{P}_3$ uses P_1 and another path, denoted by P_3. $\mathcal{P}_1$, $\mathcal{P}_4$, and $\mathcal{P}_5$ use other paths that are used only during probing phases. The probing probability $\epsilon_{\mathcal{P}}(t)$ rapidly decreases, and at the end, the source spends most of its time exploiting the best multipath. HyMAB converges to

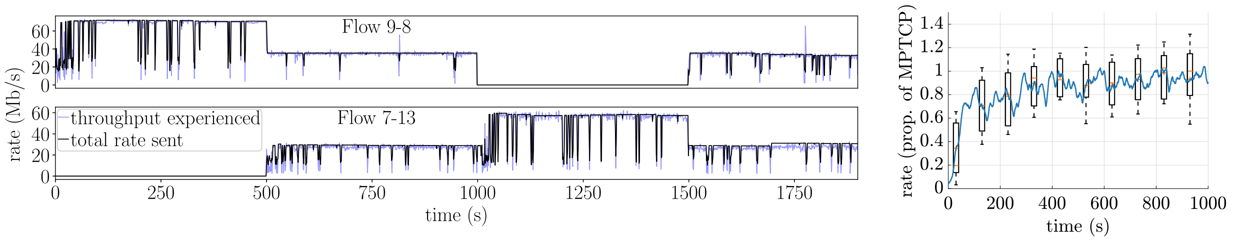

Fig. 9. Left: Experiment with two flows. HyMAB fairly shares the resources and reacts very fast, in particular with conditions learned earlier ($t = 1\,500$ s). Right: Comparison of MPTCP with HyMAB and TCP combined for 20 random flows. HyMAB achieves performance very close to MPTCP after convergence.

the true best multipath $\mathcal{P}_2$; because $\mathcal{P}_2$ and $\mathcal{P}_3$ are very close, it takes some time to converge: Until $t \approx 650$ s, the estimated best arm $\mathcal{P}_t^*$ is $\mathcal{P}_3$. Nevertheless, it sends traffic at a rate very close to the best one (30 Mb/s) throughout the experiment.

We now compare HyMAB to the optimal brute-force rate on 50 randomly selected source-destination pairs. Figure 8 (right) 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 25th and 75th 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.

C. 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 9 (left), we show the rates sent and received for the two flows. The flow sources react extremely rapidly (a few hundreds of milliseconds): Following the strategy described in Section IV-B, 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 (7), 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.

D. 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 V-B, is not necessarily the optimal one.

Figure 9 (right) 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 1) but also the absence of multi-homing requirements (see also Section VI). In addition, because of some incompatibility between ath9k and Click, the current version of HyMAB is implemented in user-space, 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 [29].

VI. Related Work

Routing and MAB: MAB has been widely studied in many contexts after the seminal works by Thompson [30], Lai and Robbins [31], and Auer et al. [6]. Routing was identified early on as a potential application for MAB [8]. However, only a few papers specifically investigate this application; they address the problem of finding the shortest path [32] or the path with minimal delay [33], [34], 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 [6], 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.

Multipath Routing: Multipath routing has been widely studied, mostly in three contexts: mobile ad-hoc networks (MANETs), wireless sensor networks (WSNs) and traffic engineering. In MANETs and WSNs [35], multipath-routing protocols have been shown to have several advantages, such as reduced delays and overhead, and better reliability and throughput [36]. But these works apply mostly on networks using only one technology. Tam et al. [37] present an algorithm valid in a multi-channel environment, but its implementation is challenging. Multipath routing has also been studied for traffic engineering [38], essentially to balance the load, which can yield lower delays and higher throughput. These techniques are valid for technologies that do not self-interfere, which is typically not the case when using shared-medium technologies such as WiFi or PLC. EMPoWER [2] is a multipath congestion-control and routing system for hybrid networks with shared-medium technologies. It optimizes throughput and controls the congestion on a *single* multipath. Independently of the context, all these protocols use heuristics to build the paths, and none of them, to the best of our knowledge, guarantees the optimality of the result. In HyMAB, we explore N different multipaths to find the best one, by using the MAB framework. We also consider the case of hybrid networks, with shared-medium technologies.

Layer 4 vs. Layer 2.5 Approaches: The most popular multipath-routing approach is MPTCP [11]. 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 significant limitation in practice: MPTCP, or any other layer-4 approach, limits the possibility of using several paths in a home network without multihoming. In addition, MPTCP can be inefficient in hybrid networks, especially with self-interfering technologies [39]. It requires knowing the paths in advance, thus it cannot adapt to dynamic conditions that would require switching multipath. 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, HyMAB operates at layer 2.5. It is confined to home networks and transparent to other Internet hosts.

VII. Conclusion and Discussion

Employing multipath in mesh hybrid networks is gaining momentum, especially with MPTCP, as a way to optimize performance under unreliable environments. We have proposed HyMAB that finds the best multipath in hybrid networks with self-interfering 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. It was implemented 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 paper 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.

Appendix

Proof of Theorem 1: 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 $\epsilon_{\mathcal{P}}(t)$ given in Theorem 1 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 $\mathcal{P}$, let $T_{\mathcal{P}}(t)$ be the number of times $\mathcal{P}$ is chosen at Stage 1 of the algorithm during the first t trials, and $n_{\mathcal{P}}(t)$ be the number of times the arm is probed, i.e., it is chosen at Stage 1 and probing is chosen at Stage 2. First, $n_{\mathcal{P}}(t)$ is unbounded almost surely if and only if $T_{\mathcal{P}}(t)$ is unbounded almost surely: If $n_{\mathcal{P}}(t)$ is unbounded, $T_{\mathcal{P}}(t)$ is obviously unbounded because $T_{\mathcal{P}}(t) \geq n_{\mathcal{P}}(t)$. If $T_{\mathcal{P}}(t)$ is unbounded, a probing phase will happen eventually almost surely because the probing probability is always strictly positive, which means that $n_{\mathcal{P}}(t)$ is unbounded almost surely. Let us now assume that there is an arm $\mathcal{P}_0$ for which $T_{\mathcal{P}_0}(t)$ is bounded with non-zero probability. It means that with non-zero probability, $\mathcal{P}_0$ 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 $\mathcal{P}(t)$ for which $T_{\mathcal{P}(t)}(t)$ is unbounded almost surely and such that $V_{\mathcal{P}(t)}(t) \geq V_{\mathcal{P}_0}(t)$. Using (5), this means that $\|\overline{\boldsymbol{x}}_{\mathcal{P}(t),n_{\mathcal{P}(t)}(t-1)}\|_1 - \|\overline{\boldsymbol{x}}_{\mathcal{P}_0,n_{\mathcal{P}_0}(t-1)}\|_1$ is greater than $\sqrt{2\ln t - 1}(1/\sqrt{n_{\mathcal{P}_0}(t-1)} - 1/\sqrt{n_{\mathcal{P}(t)}(t-1)})$. But the first term is finite (for each estimation $\boldsymbol{x}_{\mathcal{P}}(t)$, $\|\boldsymbol{x}_{\mathcal{P}}(t)\|_1 \in [0,1]$), whereas the second term is positive and goes to infinity (because $n_{\mathcal{P}(t)}(t)$ goes to infinity). This is a contradiction, consequently, for any arm $\mathcal{P}$, $T_{\mathcal{P}}(t)$ is unbounded almost surely, and hence, $n_{\mathcal{P}}(t)$ is unbounded almost surely, which completes the proof. ■

References

[1] *IEEE 1905.1-2013: Convergent Digital Home Network for Heterogeneous Technologies*, IEEE, Piscataway, NJ, USA, 2013.

[2] S. Henri, C. Vlachou, J. Herzen, and P. Thiran, "EMPoWER hybrid networks: Exploiting multiple paths over wireless and ElectRical mediums," in *Proc. ACM CoNEXT*, 2016, pp. 51–65.

[3] P. Tinnakornsrisuphap, P. Purkayastha, and B. Mohanty, "Coverage and capacity analysis of hybrid home networks," in *Proc. IEEE ICNC*, Feb. 2014, pp. 117–123.

[4] C. Vlachou, S. Henri, and P. Thiran, "Electri-Fi your data: Measuring and combining power-line communications with WiFi," in *Proc. ACM IMC*, 2015, pp. 325–338.

[5] *IEEE 1901-2010: Broadband Over Power Line Networks*, IEEE, Piscataway, NJ, USA, 2010.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, nos. 2–3, pp. 235–256, 2002.

[7] S. Bubeck and N. Cesa-Bianchi. (2012). "Regret analysis of stochastic and nonstochastic multi-armed bandit problems." [Online]. Available: https://arxiv.org/abs/1204.5721

[8] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a rigged casino: The adversarial multi-armed bandit problem," in *Proc. IEEE FOCS*, Oct. 1995, pp. 322–331.

[9] O. Besbes, Y. Gur, and A. Zeevi, "Stochastic multi-armed-bandit problem with non-stationary rewards," in *Proc. NIPS*, 2014, pp. 199–207.

[10] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," *Wireless Netw.*, vol. 11, no. 4, pp. 471–487, 2005.

[11] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, *TCP Extensions for Multipath Operation With Multiple Addresses*, document RFC 6824, IETF, Fremont, CA, USA, 2013.

[12] *IEEE 802.11n-2009: Enhancements for Higher Throughput*, IEEE, Piscataway, NJ, USA, 2009.

[13] P. H. Pathak and R. Dutta, "A survey of network design problems and joint design approaches in wireless mesh networks," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 396–428, 3rd Quart., 2011.

[14] *Understanding and Programming With Netlink Sockets*. Accessed: Jul. 2018. [Online]. Available: https://people.redhat.com/nhorman/papers/netlink.pdf

[15] *The Faifa Open Source Project*. Accessed: Jul. 2018. [Online]. Available: https://www.openhub.net/p/faifa

[16] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1466–1478, Oct. 2012.

[17] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. ICML*, 2013, pp. 1–9.

[18] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[19] L. DaCosta, A. Fialho, M. Schoenauer, and M. Sebag, "Adaptive operator selection with dynamic multi-armed bandits," in *Proc. ACM GECCO*, 2008, pp. 913–920.

[20] Y. Gai, B. Krishnamachari, and R. Jain, "Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation," in *Proc. IEEE DySPAN*, Apr. 2010, pp. 1–9.

[21] K. Liu and Q. Zhao, "Distributed learning in multi-armed bandit with multiple players," *IEEE Trans. Signal Process.*, vol. 58, no. 11, pp. 5667–5681, Nov. 2010.

[22] D. Kalathil, N. Nayyar, and R. Jain, "Decentralized learning for multiplayer multiarmed bandits," *IEEE Trans. Inf. Theory*, vol. 60, no. 4, pp. 2331–2345, Apr. 2014.

[23] F. Wilhelmi *et al.* (2017). "Collaborative spatial reuse in wireless networks via selfish multi-armed bandits." [Online]. Available: https://arxiv.org/abs/1710.11403

[24] G. Gaertner and V. Cahill, "Understanding link quality in 802.11 mobile ad hoc networks," *IEEE Internet Comput.*, vol. 8, no. 1, pp. 55–60, Jan./Feb. 2004.

[25] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[26] *MultiPath TCP—Linux Kernel Implementation*. Accessed: Jul. 2018. [Online]. Available: https://multipath-tcp.org/pmwiki.php/Users/OpenWRT

[27] Y. Yang, J. Wang, and R. Kravets, *Interference-Aware Load Balancing for Multihop Wireless Networks*, document UIUCDCS-R-2005-2526, Univ. Illinois Urbana-Champaign, Champaign, IL, USA, 2005.

[28] S. Henri and P. Thiran, "Optimal number of paths with multipath routing in hybrid networks," in *Proc. IEEE WoWMoM*, 2018.

[29] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 303–314, Oct. 1998.

[30] W. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, nos. 3–4, pp. 285–294, 1933.

[31] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.

[32] B. Awerbuch and R. D. Kleinberg, "Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches," in *Proc. ACM STOC*, 2004, pp. 45–53.

[33] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 671–678.

[34] B. Awerbuch and R. Kleinberg, "Online linear optimization and adaptive routing," *J. Comput. Syst. Sci.*, vol. 74, no. 1, pp. 97–114, 2008.

[35] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: A survey," *IEEE Wireless Commun.*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[36] M. Tarique, K. E. Tepe, S. Adibi, and S. Erfani, "Survey of multipath routing protocols for mobile ad hoc networks," *J. Netw. Comput. Appl.*, vol. 32, no. 6, pp. 1125–1143, 2009.

[37] W.-H. Tam and Y.-C. Tseng, "Joint multi-channel link layer and multi-path routing design for wireless mesh networks," in *Proc. IEEE INFOCOM*, May 2007, pp. 2081–2089.

[38] *A Survey of Multipath Routing for Traffic Engineering*. Accessed: Jul. 2018. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.9976&rep=rep1&type=pdf

[39] S. C. Nguyen, X. Zhang, T. M. T. Nguyen, and G. Pujolle, "Evaluation of throughput optimization and load sharing of multipath TCP in heterogeneous networks," in *Proc. IEEE WOCN*, May 2011, pp. 1–5.

Sébastien Henri received the M.Sc. degree in engineering from the École polytechnique, France, in 2011, and the M.Sc. degree in engineering from Télécom ParisTech, France, in 2013. He is currently pursuing the Ph.D. degree with EPFL, Switzerland. His Ph.D. research focuses on the evaluation and analysis of throughput, latency, and security with hybrid networks. Before starting his Ph.D. research, he worked with Qualcomm for one year and with Technicolor for one year. His research interests include the performance evaluation and modeling of wireless and mobile networks, and the security of power-line communications and hybrid networks.

Christina Vlachou received the Diploma degree in electrical and computer engineering from the National Technical University of Athens in 2011 and the Ph.D. degree from EPFL in 2016. Her Ph.D. dissertation was on measuring, modeling, and enhancing power-line communications (PLC) performance. During her Ph.D. studies, she was an intern at Marvell and Qualcomm involved in wireless and PLC. She is currently a Researcher with Hewlett Packard Labs, Palo Alto, CA, USA, involved in next-generation wireless technologies. Her interests include wireless and mobile communications, multi-user performance, and design of hybrid networks. She was a recipient of the Best Paper Runner-Up Award for her work on PLC at the IEEE International Conference on Network Protocols 2014 Conference and the Papakyriakopoulos Award for excellence in Mathematics from the National Technical University of Athens in 2007.

Patrick Thiran (S'89–M'96–SM'12–F'14) received the Electrical Engineering degree from the Université Catholique de Louvain, Louvain-la-Neuve, Belgium, in 1989, the M.S. degree in electrical engineering from the University of California at Berkeley, USA, in 1990, and the Ph.D. degree from EPFL in 1996. From 2000 to 2001, he was with Sprint Advanced Technology Labs, Burlingame, CA, USA. He is currently a Full Professor with EPFL. He became an Adjunct Professor in 1998, an Assistant Professor in 2002, an Associate Professor in 2006, and a Full Professor in 2011. His research interests include networks, performance analysis, stochastic models, the analysis and design of wireless and PLC networks, and data-driven network science. He was a recipient of the 1996 EPFL Ph.D. Award and the 2008 Crédit Suisse Teaching Award. He served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1997 to 1999 and for the IEEE/ACM TRANSACTIONS ON NETWORKING from 2006 to 2010. He is currently serving on the Editorial Board of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS.