Understanding and Tackling the Root Causes of Instability in Wireless Mesh Networks

Adel Aziz, David Starobinski, and Patrick Thiran

***Abstract*—We investigate, both theoretically and experimentally, the stability of CSMA-based wireless mesh networks, where a network is said to be stable if and only if the queue of each relay node remains (almost surely) finite. We identify two key factors that impact stability: the network size and the so-called "stealing effect", a consequence of the hidden node problem and non-zero transmission delays. We consider the case of a greedy source and prove, by using Foster's theorem, that 3-hop networks are stable, but only if the stealing effect is accounted for. We also prove that 4-hop networks are, on the contrary, always unstable (even with the stealing effect) and show by simulations that instability extends to more complex linear and non-linear topologies. To tackle this instability problem, we propose and evaluate a novel, distributed flow-control mechanism, called EZ-flow. EZ-flow is fully compatible with the IEEE 802.11 standard (i.e., it does not modify headers in packets), can be implemented using off-the-shelf hardware, and does not entail any communication overhead. EZ-flow operates by adapting the minimum congestion window parameter at each relay node, based on an estimation of the buffer occupancy at its successor node in the mesh. We show how such an estimation can be conducted *passively* by taking advantage of the broadcast nature of the wireless channel. Real experiments, run on a 9-node testbed deployed over 4 different buildings, show that EZ-flow effectively smoothes traffic and improves delay, throughput, and fairness performance.**

I. Introduction

WIRELESS mesh networks (WMNs) promise to revolutionize Internet services by providing customers with ubiquitous high-speed access at low cost. Thus, several cities and communities have already deployed, or are about to deploy WMNs [1, 2, 5]. Nevertheless, several technical obstacles must be surmounted to allow for the widespread adoption of this technology. In particular, a key challenge is to ensure a smooth and efficient traffic flow over the *backhaul*, i.e., the multi-hop wireless links connecting the end-users to the Internet.

The Medium Access Control (MAC) protocol, used to manage contention and avoid packet collisions on the shared channel, plays a key role in determining the performance of the backhaul of a WMN. Most WMNs use the IEEE 802.11 standard [8] as their MAC protocol for the following reasons: (i) It is based on Carrier-Sense Multiple Access (CSMA), a mechanism that naturally lends itself to a distributed implementation; (ii) it has low control overhead; (iii) it is ubiquitous and (iv) it is inexpensive to deploy.

The IEEE 802.11 protocol, however, was initially designed to support single-hop, but not multi-hop, communication where multiple nodes must cooperate to efficiently transport one or multiple flows. In this paper, we show how and why 802.11-based wireless mesh networks are susceptible to turbulence that takes the form of the following: (i) buffer build up and overflow at relaying nodes; (ii) major end-to-end delay fluctuations; and (iii) reduced throughput. In Figure 1 we depict the consequence of this unstable behavior by using data collected from measurements on a real network with a greedy access point. The figure shows the instantaneous buffer occupancy at the relaying nodes for a (stable) 3-hop network and an (unstable) 4-hop network. In this scenario, the end-to-end throughput in the 4-hop case is about half that in the 3-hop case. The intrinsic instability of IEEE 802.11 mesh networks that are longer than 3 hops may explain why current implementations use only a few hops [3]. It is therefore critical to rigorously characterize the behavior of CSMA-like protocols in multihop scenarios and propose possible improvements when appropriate.

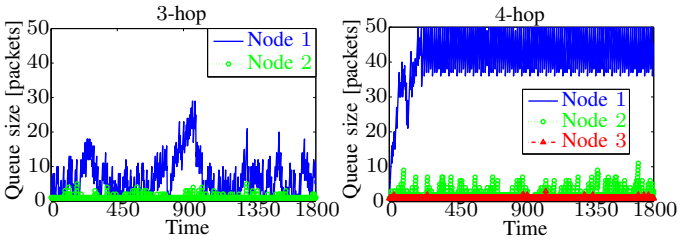

Fig. 1. Experimental results for the queue evolution of each relay node in 3- and 4-hop topologies. A 3-hop network is stable, whereas a 4-hop is unstable with the queue of its first relaying node (node 1) building up until it reaches the buffer hardware limit of 50 packets and starts overflowing.

We prove that the network is stable or unstable, depending on its size and a phenomenon referred to as a *stealing effect* that results from the hidden node problem and non-zero transmission delays. The likelihood of this phenomenon is captured by the stealing effect probability $0 \leq p \leq 1$, a parameter explained in detail in Section III.C.

After detailing the problem and reviewing related work in Section II, we introduce a discrete Markov chain model that captures the stealing effect phenomenon in Section III. We demonstrate in Section IV that in the case of a 3-hop network, the system is stable if and only if the stealing effect is present ($p > 0$). However, for larger linear K-hop topologies ($K > 3$) the network is always unstable, as proven in Section V for $K = 4$, and presumably so for larger K with a formal proof for the case $p = 0$. Even though 802.11 multihop networks are known to suffer from unfairness and starvation (see [16, 21, 36, 39]), to the best of our knowledge, to date the (in)stability of 802.11 multihop networks has not been demonstrated either

A. Aziz and P. Thiran are with the School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland; e-mail: adel.aziz@epfl.ch, patrick.thiran@epfl.ch.

D. Starobinski is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA; e-mail: staro@bu.edu

Part of the results in this paper appeared in [9] and [11].

experimentally or analytically.

After elucidating the sources of instability, we focus on the problem of devising distributed channel access mechanisms to ensure stability in multi-hop networks. This issue has received much attention since the seminal work of Tassiulas and Ephremides [41]. Most of the solid, analytical work on this problem [14, 22, 40, 44, 46] follow a "top-down" approach, i.e., they start from a theoretical algorithm that provably achieves stability and then try to derive a distributed version. The drawback of this approach is the difficulty of testing the proposed solution in practice by using existing wireless cards. Indeed, despite all the previous theoretical work, few solutions have been implemented and tested to date [11, 43]. To bridge this gap, we instead resort to a bottom-up approach, i.e., we start from the existing IEEE 802.11 protocol, identify the main causes of turbulence and instability, and then we derive a practical and decentralized mechanism to solve this problem.

In Section VIII, we propose and analyze a new, distributed flow-control mechanism, called *EZ-flow*, that solves the turbulent behavior of IEEE 802.11 WMNs. EZ-flow requires no modifications to the IEEE 802.11 protocol and is readily implementable with off-the-shelf hardware. EZ-flow runs as an independent program at each relaying node. By passively monitoring buffer occupancy at successor nodes, it adapts a parameter of IEEE 802.11, the minimum contention window CW_{min} (CW_{min} is inversely proportional to the channel access probability). The standard way to obtain the buffer occupancy information is via message passing. Message passing, however, may further exacerbate congestion and reduce resources available for sending useful data [44]. To avoid this drawback, EZ-flow takes advantage of the broadcast nature of the wireless medium to infer buffer occupancy at successor nodes. Obtaining this information without message exchanges is one of the major advantages of EZ-flow as it enables the network to achieve stability without any communication overhead and without requiring the knowledge of the capacity (which is time varying and hard to obtain in real implementations).

We end the paper by validating the stabilizing properties of EZ-flow experimentally in Section IX and summarizing our findings in Section X.

II. Background

A. Problem Statement

We consider the case of a wireless multi-hop topology such as the one in the backhaul of a mesh network. The backhaul is composed of three types of nodes: (i) a Wired Access Point (WAP) that plays the role of gateway and is connected to the Internet, (ii) Access Points (APs) that ensure the access part of the WMN by having the end-users connected to them (note that usually the backhaul and access part of a WMN run on independent channels to avoid interferences) and (iii) Transit Access Points (TAPs) that transport the data packets through multiple hops from the WAP to the AP and back.

We then focus on the stability of these multi-hop networks by analyzing the queue evolution at the relay nodes (TAPs), both analytically and experimentally.

B. Related Work

Much effort has been put into understanding how IEEE 802.11 behaves in a multi-hop environment. Previous works show the inefficiency of the protocol in providing optimal performance, as far as delay, throughput and fairness are concerned [19]. In [33], Nandiraju et al. propose a queue management mechanism to improve fairness. However, as they mention in their conclusion, a solution to the inherent unfairness of the IEEE 802.11 MAC layer is needed for their mechanism to work properly. In [26], Jindal and Psounis claim that the performance of IEEE 802.11 in multi-hop settings is not as bad as it could be expected. For instance, they show an example through simulation where IEEE 802.11 achieves a max-min allocation that is at least 64% of the max-min allocation obtained with a perfect scheduler. Our experiments, in Section IX, show that the performance may actually be much worse. We believe that the cause of the discrepancy is that [26] assumes that flows are rate-controlled at the source, whereas we do not make such an assumption. To tackle the inefficiency of IEEE 802.11, different approaches have been proposed and we regroup them into five categories.

1) Throughput-Optimal Scheduling with Message Passing: A first analytical solution to the stability problem in multi-hop networks is discussed in the seminal work of Tassiulas and Ephremides [41], which introduces a back-pressure algorithm. Their methodology uses a centralized scheduler that selects for transmission the link with the greatest queue difference, i.e. the greatest difference in buffer occupancy between the MAC destination node and the MAC source node. Such a solution works well for a wired network, but is not adapted to a multi-hop wireless network where decentralized schedulers are needed due to the synchronization problem. Toward this goal, Modiano et al. introduced the first distributed scheduling framework that uses control messages to achieve throughput optimal performances [32]. Further extensions to distributed scheduling strategies have been discussed in works such as [14], where Chapokar et al. propose a scheduler that attains a guaranteed ratio of the maximal throughput. Another effort to reduce the complexity of back-pressure is presented in [46], where Ying et al. propose to enhance scalability by reducing the number of queues that need to be maintained at each node. The interaction between an end-to-end congestion controller and a local queue-length-based scheduler is discussed by Eryilmaz and Srikant in [17]. The tradeoff that exists in each scheduling strategy between complexity, utility and delay is discussed in depth in [44] by Yi et al. One of the drawbacks of these previous methods is that they require queue information from other nodes. The usual solution is to use message passing, which produces costly overhead even if it is limited to the direct neighbors.

2) Throughput-Optimal Scheduling with CSMA and without Message Passing: Some recent works propose schedulers that do not require queue information from other nodes. In [22], Gupta et al. propose an algorithm that uses the maximal node degree in the network. Proutière et al. [34] propose another algorithm, where each node makes the scheduling decision based solely on its own queue. Similarly, Marbach

and Eryilmaz propose a throughput-optimal approach that uses a backlogged-based CSMA mechanism for scheduling and a congestion signal marking mechanism for source-rate control [31]. Shin et al. [40] have recently proposed an algorithm that achieves stability and where each node makes scheduling decisions on the basis of a logarithmic function of its own buffer occupancy. Nevertheless, even though their algorithm is throughput-optimal for the case of a perfect CSMA, it requires a very large buffer size (i.e., in the order of thousands of packets). Such a requirement presents two drawbacks: First, large buffers imply a large end-to-end delay; second, the requirement of such large buffers does not match with current hardware that usually have a standard MAC buffer of only 50 packets. A different approach was followed by Jiang and Walrand who introduced an adaptive CSMA algorithm that adjusts the transmission aggressiveness based on a differential between the arrival and service rate [24]. To sum up, significant theoretical progress has been recently made on algorithms that are based on variations of or around the MaxWeight algorithm, in order to provide queue stability and maximum throughput for a wide range of scenarios. Nevertheless, van de Ven et al. proved that this stability guarantee relies on the fundamental premise that the system consists in a fixed set of nodes with a fixed traffic demand. However, in case variability is accounted for in the system, MaxWeight policies may fail to provide stability [42]. There is therefore still a need to develop mechanisms that can cope with the network variability, as it is an inherent characteristic of a practical wireless network.

3) Practical Approaches at the MAC Layer: Despite this significant body of analytical work, almost all the existing solutions are still far from being compatible with the current IEEE 802.11 protocol, and require in general knowing the feasible capacity region. One possible solution is to estimate it before running the MAC algorithm and to then use an optimization-based rate control at the network layer [37]. Our approach differs from the previous works in the sense that we propose a practical solution that does not require estimating the capacity region. Our solution is implemented on off-the-shelf hardware and takes advantage of the broadcast nature of the wireless medium to derive the queue information of neighboring nodes. Another practical scheme, developed in parallel with our work, is the hop-by-hop congestion control mechanism DiffQ in [43], which implements a form of backpressure (i.e., prioritizing links with large backlog differential). To achieve this implementation, DiffQ lets each node inform its neighbors of its queue size by piggybacking this information on the data packet (i.e., modifying the packet structure by adding an additional header) and then it schedules the packets in one of the four MAC queues (each one with a different CW_{min} value) depending on the backlog difference. Our approach differs in two ways: (i) We use the next-hop queue information instead of the differential backlog, which results in an implicit congestion signal being pushed back more rapidly to the source; (ii) as opposed to DiffQ, we do not modify the packet structure in any way as we passively derive the next-hop buffer occupancy without any form of message passing. To the best of our knowledge, EZ-flow is the first implementation that solves the turbulence and instability problem in real 802.11-based multi-hop testbed without modifying the packets and without any form of message passing. We also point out that the novel passive queue derivation methodology of our BOE (Buffer Occupancy Estimation) module, detailed in Section VIII-C, is potentially compatible with new algorithms such as DiffQ; it could allow them to eliminate the need to piggyback the queue information (resulting in unmodified packet structure).

4) Practical Approaches at Upper Layers: Another line of research, parallel to ours, tackles congestion at the transport layer rather than the MAC (link) layer. In [35], Radunović et al. introduce a new practical system architecture called Horizon. Horizon uses back-pressure to perform load-balancing and multi-path routing in mesh networks using TCP. In [36], Rangwala et al. present limitations of TCP in mesh networks and propose a new rate-control protocol named WCP that achieves performances that are both more fair and efficient. Similarly, Shi et al. focus on the starvation that occurs in TCP when a one-hop flow competes with a two-hop flow and they propose a counter-starvation policy that solves the problem for this scenario [39]. Garetto et al. also tackle the starvation problem at an upper layer [21]. They propose a rate-limiting solution and evaluate it by simulation. Their main reason for not using the MAC-based approach is to ensure compatibility with 802.11-based mesh network currently deployed. EZ-flow is also fully compatible with the existing protocol because it only varies the contention window CW_{min}, a modification allowed by the standard. Our approach differs from previous work in the sense that we tackle the problem at the MAC layer without using any form of message passing. The work of Yi and Shakkottai showing that a hop-by-hop congestion control outperforms an end-to-end version further supports our approach [45].

5) Practical Approaches Exploiting Broadcast: Finally, another kind of work, which is similar to ours in the idea of exploiting the broadcast nature of the wireless medium, is found in cooperative diversity and network coding. In [27], Katti et al. propose that relay nodes listen to packets that are not necessarily targeted for them in order to code the packets together later on (i.e. XOR them together) and thus increase the channel capacity. In [12], Biswas and Morris present a routing mechanism named ExOR that takes advantage of the broadcast nature to achieve cooperative diversity and thus increase the achievable throughput. Note that EZ-flow can potentially work with routing solutions such as ExOR. Indeed, the fact that the forwarded packets are not all sent to the same successor node implies that the forwarding process may not be FIFO (First-In, First-Out) anymore and thus the information derived by the BOE becomes more noisy. Nevertheless, by using a larger averaging period to smoothen the noise, this information could still be useful for congestion control. Moreover, to perform congestion control, a node does not always need to know precisely which successor (i.e., which next-hop relay) gets its packets: It just needs to keep to a low value the *total* number of packets that are waiting at all of its successors to be forwarded. This could be done using a methodology similar to the one presented in this paper for the unicast

case. A similar extension of a congestion-control scheme from unicast to multicast is discussed by Scheuermann et al. in [38]. Finally, in [23] Heusse et al. also use the broadcast nature of IEEE 802.11 to improve the throughput and fairness of single-hop WLANs by replacing the exponential backoff with a mechanism that adapts itself according to the number of slots that are sensed idle. Our work follows the same philosophy of taking advantage of the "free" information given by the broadcast nature. Apart from this, our approach is different, because we do not use cooperation and network coding techniques at relay nodes. Instead, in a competitive context, we derive and use the next-hop buffer occupancy information to tackle the traffic congestion occurring in multi-hop scenarios.

III. Modeling the sources of instability

Figure 1 shows that a particular 3-hop network is stable, but not a 4-hop one. In order to understand these experimental results showing a drastic behavioral transition, we introduce an analytical model inspired from the behavior of CSMA/CA protocols (e.g., 802.11-like protocols) with some necessary simplifications for the sake of tractability. We emphasize that, given the mathematical assumptions, our analysis is exact.

A. MAC Layer Description

The first common assumption [14, 17, 30, 41, 46] is that of a slotted discrete time axis, in other words, each transmission takes one time slot and all the transmissions occurring during a given slot start and finish at the same time. We consider a greedy source model, i.e., the WAP (gateway) always has new packets ready for transmission. Assuming a K-hop system, the packets flow from the WAP to TAP_K, via TAP_1, TAP_2, ..., TAP_{K-1}. TAPs do not generate packets of their own. Each TAP is equipped with an infinite buffer.

We assume that the system evolves according to a two-phase mechanism: a *link competition phase* and a *transmission phase*. The link competition phase, whose length is assumed to be negligible, occurs at the beginning of each slot. During this phase, all the nodes with a non-empty buffer compete for the channel and a pattern of successful transmissions emerges, referred to as *transmission pattern* in this paper. Given the current state of buffers, the link competition process is assumed to be independent of competitions that happened in previous slots. This assumption is similar to the commonly used assumption of exponentially (memoryless) distributed backoffs. During this phase, non-empty nodes are sequentially chosen at random and added to the transmission pattern if and only if they do not interfere with already selected communications (with the notable exception of the *stealing effect* described below). The final pattern is obtained when no more nodes can be added without interfering with the others.

The second phase of the model is fairly straightforward as it consists in applying the transmission pattern from the previous phase in order to update the buffer status of the system. This buffer status information is of utmost importance for our analysis because it is the parameter that indicates whether the network remains stable (no buffer explodes) or suffers congestion (one or more buffers build up).

B. Discrete Markov Chain Model

We now formalize the model previously described mathematically. All packets are generated by the WAP (node 0), and are forwarded to the last TAP (node K) by successive transmissions via the intermediate nodes (TAPs) 1 to $K-1$. A time step $n \in \mathbb{N}$ corresponds to the successful transmission of a packet from some node i to its neighbor $i+1$, or if K is large enough, of a set of packets from different non-interfering nodes $i, j, \ldots$ to nodes $i+1, j+1, \ldots$, provided these transmissions overlap in time (the transmitters and receivers must therefore not interfere with each other). We assume that node 0 always has packets to transmit (infinite queue), and that node K consumes immediately the packets, as it is the exit point of the backbone (its queue is always 0). We are interested in the evolution of the queue sizes b_i of relaying nodes $1 \leq i \leq K-1$ over time, and therefore we adopt as a state variable of the system at time n the vector

$$\vec{b}(n) = [b_1(n)\ b_2(n)\ \ldots\ b_{K-1}(n)]^T,$$

with T denoting transposition. We also introduce a set of K auxiliary binary variables z_i, $0 \leq i \leq K-1$, representing the i^{th} link activity at time slot n: $z_i(n) = 1$ if a packet was successfully transmitted from node i to node $i+1$ during the n^{th} time slot, and $z_i(n) = 0$ otherwise. Observing that $b_i(n+1) = b_i(n) + z_{i-1}(n) - z_i(n)$, we can recast the dynamics of the system as

$$\vec{b}(n+1) = \vec{b}(n) + A * \vec{z}(n) \tag{1}$$

where

$$\vec{z}(n) = [z_0(n)\ z_1(n)\ z_2(n)\ \ldots\ z_{K-1}(n)]^T$$

$$A = \begin{bmatrix} 1 & -1 & 0 & \ldots & 0 \\ 0 & 1 & -1 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 & -1 \end{bmatrix}.$$

Finally, the activity of a link z_i depends on the queue sizes of all the nodes, which we cast as $z_i = g_i(\vec{b})$ for some random function $g_i(\cdot)$ of the queue size vector, or in vector form as

$$\vec{z}(n) = g(\vec{b}(n)). \tag{2}$$

The specification of $g = [g_0, \ldots, g_{K-1}]^T$ is the less straightforward part of the model, as it requires entering in some additional details of the CSMA/CA protocols, which we defer to the next sections. We will first expose it in Section IV for a $K = 3$ hops network, and then move to the larger networks with $K = 4$ and $K \geq 5$ in the subsequent section, as the specification of g comes with some level of complexity as K gets larger. Nevertheless, we can already mention here two simple constraints that g must verify: (i) node i cannot transmit if its buffer is empty, and therefore $z_i = g_i(\vec{b}) = 0$ if $b_i = 0$; (ii) nodes that successfully transmit in the same time slot must be at least 2 hops apart, as otherwise the packet from node i would collide at node $i+1$ with the packet from node $i+2$. Hence

$$z_i z_{i+k} = 0 \text{ for } k \in \{-2, -1, 1, 2\}. \tag{3}$$

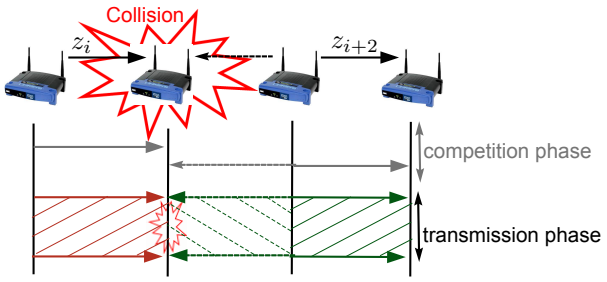

Fig. 2. Stealing effect scenario.

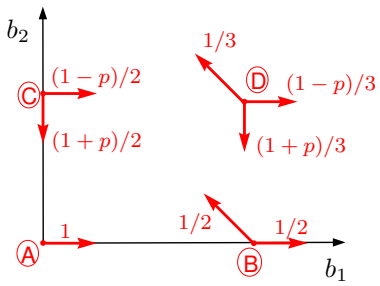

Fig. 3. Random walk in $\mathbb{N}^2$ modeling the 3-hop network. where the 4 regions are: (A) $\{0;0\}$, (B) $\{b_1>0;0\}$, (C) $\{0;b_2>0\}$ and (D) $\{b_1>0;b_2>0\}$.

We observe that (1) and (2) make the model a discrete-time, irreducible Markov chain. The (in)stability of the network coincides with its (non-)ergodicity.

C. Stealing Effect Phenomenon

The stealing effect phenomenon is a result of the well-known hidden node problem that occurs in multihop topologies. Indeed, the existence of directional multi-hop flows in the backbone of mesh networks, from node 0 to node K may induce unfairness in a way that does not arise in single-hop scenarios. Figure 2 illustrates an example where the stealing effect occurs. When node i first enters the link competition phase, node $i+2$ may be unaware of this transmission attempt. Because it senses the medium to be idle, node $i+2$ may therefore start a concurrent transmission to node $i+3$ even though it lost the competition phase (i.e. node $i+2$ selected a larger backoff than node i). As a collision occurs at node $i+1$ (due to the broadcast nature of the wireless medium), node i will experience an unsuccessful transmission, whereas the transmission from node $i+2$ will succeed. We refer to this unfairness artifact as the stealing effect, which differs from the classical capture effect. The latter pertains to packets transmitted to the same destination.

Definition 1 (Stealing Effect): The stealing effect occurs when a node $i+2$ successfully captures the channel from an upstream node i, even though it accesses the medium later. We define p to be the probability of the occurrence of the stealing effect.

In IEEE 802.11, the stealing effect corresponds to the event where node $i+2$ captures the channel, even though it has a larger backoff value than node i. The probability of this event depends on the specific protocol implementation. If the optional RTS/CTS handshake is disabled, then $p \to 1$. If RTS/CTS is enabled, then p is typically much smaller, but still non-zero because RTS messages may collide [39]. Indeed, the transmission time of a control message (e.g., the RTS transmission time at the 1Mb/s basic rate is $352\mu s$) is non-negligible compared to the duration of a backoff slot ($20\mu s$).

In our model, the stealing effect is captured by having the function $g(\cdot)$ in (2) depend on p. As revealed by our analysis, a positive and somewhat counterintuitive consequence of the stealing effect is the promotion of a laminar packet flow, namely, a smooth propagation of packets. Indeed, by favoring downstream links over upstream ones, it creates a form of virtual back-pressure that prevents packets from being pushed too quickly into the network.

D. Stability Definition

A buffer is stable when its occupancy does not tend to increase indefinitely. More formally, we adopt the usual definitions of stability (see e.g. Section 2.2 of [13]).

Definition 2 (Stability): A queue is stable when its evolution is ergodic (it goes back to zero almost surely in finite time). A network is stable when the queues of all forwarding nodes (i.e., all TAPs) are stable.

IV. 3-HOP NETWORKS STABILITY

Let us first analyze the 3-hop topology, which remains relatively simple because only one link can be active at a given time slot. Indeed, the only three possible transmission patterns $\vec{z}$ are $[1\ 0\ 0]^T$, $[0\ 1\ 0]^T$ and $[0\ 0\ 1]^T$. We can now complete the description of the function $g(\cdot)$, before analyzing the ergodicity of the Markov chain.

A. System Evolution

The role of the stochastic function $g(\cdot)$ is to map a buffer status $\vec{b}$ to a transmission pattern $\vec{z}$ with a certain probability.

First, in the case of an idealized CSMA/CA model without the stealing effect ($p=0$), all non-empty nodes have exactly the same probability of being scheduled. That is, if only node 0 and node 1 (or, respectively, node 2) have a packet to send, both patterns $[1\ 0\ 0]^T$ and $[0\ 1\ 0]^T$ (resp., $[0\ 0\ 1]^T$) happen with a probability of $1/2$. Similarly, when all three nodes have a packet to send, each of the three possible transmission patterns happens with a probability of $1/3$.

More generally, when we include the stealing effect, we capture the bias towards downstream links that are two hops away. When only node 0 and node 1 compete for the channel, nothing is changed and the probability of success remains $1/2$ as they are only separated by one single hop. However, when node 0 and node 2 compete together, there is a probability p that node 2 steals the channel.

This leads us to define function $g(\cdot)$ differently for each region of $\mathbb{Z}^2$ as shown in Figure 3. First, in region $A = \{b_1(n)=0, b_2(n)=0\}$, $g([b_1(n)\ b_2(n)]^T) = [1\ 0\ 0]^T$.

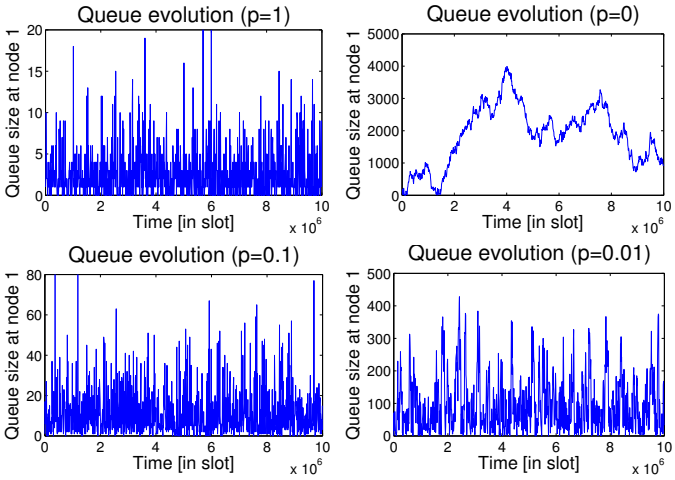

Fig. 4. Queue evolution for 3-hop with different p values.

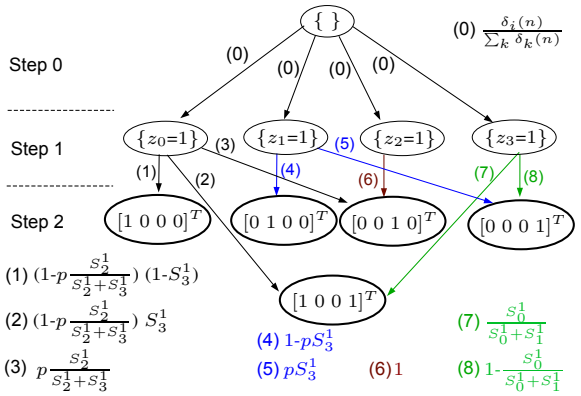

Fig. 5. *Decision tree* to obtain $\vec{z} = g(\vec{b})$ for the 4-hop model.

In region $B = \{b_1(n) > 0, b_2(n) = 0\}$ we have that

$$g([b_1(n)\ b_2(n)]^T) = \begin{cases} [1\ 0\ 0]^T & \text{with probability } 1/2 \\ [0\ 1\ 0]^T & \text{with probability } 1/2. \end{cases}$$

In region $C = \{b_1(n) = 0, b_2(n) > 0\}$,

$$g([b_1(n)\ b_2(n)]^T) = \begin{cases} [1\ 0\ 0]^T & \text{with probability } (1-p)/2 \\ [0\ 0\ 1]^T & \text{with probability } (1+p)/2. \end{cases}$$

Finally, in region $D = \{b_1(n) > 0, b_2(n) > 0\}$, all three nodes compete, and node 2 can still steal the channel from node 0, hence

$$g([b_1(n)\ b_2(n)]^T) = \begin{cases} [1\ 0\ 0]^T & \text{with probability } (1-p)/3 \\ [0\ 1\ 0]^T & \text{with probability } 1/3 \\ [0\ 0\ 1]^T & \text{with probability } (1+p)/3. \end{cases}$$

B. Stability Analysis

The queue evolution from (1) is a random walk in $\mathbb{N}^2$, as depicted in Figure 3. Theorem 1 shows the stabilizing influence of the stealing effect.

Theorem 1: A 3-hop network is unstable for the case $p = 0$ and it is stable for all $0 < p \leq 1$.

Proof: The instability of the case $p = 0$ is readily proved with the Non-ergodicity theorem ([18], p. 30) using the Lyapunov function

$$h(b_1, b_2) = b_1, \tag{4}$$

and setting the constants $c = d = 1$ in that theorem.

Next we prove the stability of the cases $0 < p \leq 1$ by using Foster's theorem (see Appendix) with the Lyapunov function

$$h(b_1, b_2) = b_1^2 + b_2^2 - b_1 b_2,$$

the finite set $F = \{0 \leq b_1, b_2 < 5/p\}$, the function $k = 1$ and the notations

$$\begin{aligned} \mu_{b_1,b_2}(n) &= \mathbb{E}\left[h(\vec{b}(n+1)) \mid h(\vec{b}(n)) = h(b_1, b_2)\right] \\ \epsilon_{b_1,b_2}(n) &= \mu_{b_1,b_2}(n) - h(b_1, b_2), \end{aligned}$$

where $\epsilon_{b_1,b_2}(n)$ can be interpreted as the drift of the random walk at time n. Then we verify Foster's theorem for all the three regions of $\mathbb{N}^2 \setminus F$. After some computations, we find that for Region $B \setminus F$, $\epsilon_{b_1,0}(n) = 2 - b_1(n)/2 < 0$. Likewise, for region $C \setminus F$, we get $\epsilon_{0,b_2}(n) = 1 - (3+p)b_2(n)/2 < 0$. Finally, for region $D \setminus F$, we have $\epsilon_{b_1,b_2}(n) = 5/3 - p(b_1(n) + b_2(n))/3 < 0$. Consequently, the two conditions of the theorem are satisfied and stability is proved. ■

Finally, in Figure 4 we present the effect of p on the queue evolution through a simulation of our model. We also mention, that our theoretical results give insight into monitoring the queue of node 1 in order to assess the stability of the system (the function of (4) only considers b_1 to prove instability).

V. 4-HOP NETWORKS INSTABILITY

The 4-hop system is relatively similar to the 3-hop, except that the function $g(\cdot)$ becomes more complex to derive. Indeed the five possible patterns $\vec{z}$ are now $[1\ 0\ 0\ 0]^T$, $[0\ 1\ 0\ 0]^T$, $[0\ 0\ 1\ 0]^T$, $[0\ 0\ 0\ 1]^T$ and $[1\ 0\ 0\ 1]^T$

A. System Evolution

The drastic difference when moving to 4-hop topologies is that nodes that can transmit concurrently will *reinforce* each other and will increase their transmission probability [15, 16].

This interdependence makes the determination of $g(\cdot)$ less straightforward than in the 3-hop case. We capture this complexity by a *decision tree*, depicted in Figure 5, which maps all the sequential events that can occur for the selection of the transmission pattern (one of the states in bold in Figure 5).

Before describing the exact mechanisms behind our decision tree, we introduce some necessary notations. First, we define the iteration step m that represents the step between two sequential events (an event corresponds to either the inclusion of a node in the transmission pattern or the removal of a node from the competition). As shown in Figure 5, the decision-tree process ends in two iterations ($m \in \{0, 1, 2\}$) and this is due to the fact that at most two links can be active concurrently in the transmission pattern of a 4-hop network.
Secondly, we introduce the two indicator vectors $\vec{\delta}(n)$ and $\vec{S}^m$. The four entries $\delta_i(n) = 1_{\{b_i(n)>0\}}$ indicate which buffers are occupied ($\delta_i(n) = 1$) or empty ($\delta_i(n) = 0$). The vector

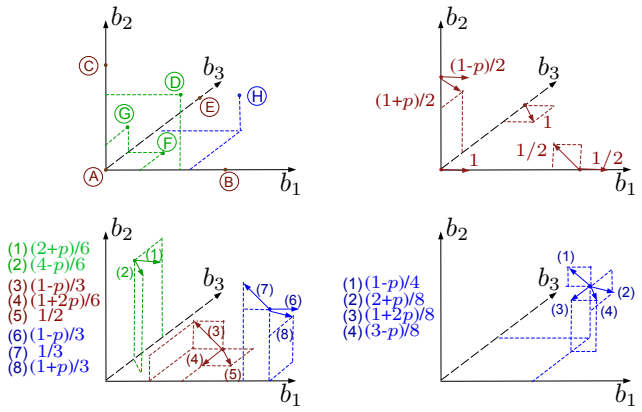

Fig. 6. Random walk in $\mathbb{N}^3$ for a 4-hop network.

$\vec{S}^m = [S_0^m \ \dots \ S_3^m]^T$, which is obtained through an iterative process, indicates the set of nodes that are still in competition for the channel at iteration step m. Initially, all the nodes with a non-empty buffer compete for the channel at step 0 and therefore $\vec{S}^0 = \vec{\delta}(n)$. Then the indicator vector at step m, $\vec{S}^m$, is obtained by removing from $\vec{S}^{m-1}$ the node that was selected at iteration step m and its direct neighbors. For example, if we start from the fully-occupied case $\vec{S}^0 = \vec{1}$ and follow the path where node 1 is selected (z_1 is set to 1), the nodes 0, 1 and 2 are removed from the competition and the new indicator vector becomes $\vec{S}^1 = [0\ 0\ 0\ 1]^T$ for this path.

The exact probabilities of each link of the decision tree are denoted in Figure 5. The intuition behind these probabilities is that at step m all nodes i that are still competing for the channel (i.e., $S_i^m = 1$) have an equal probability of being selected for transmission. Furthermore, if z_{i-2} is already set to 1 at step m, the selected node i has a probability p of successfully stealing the channel, in which case z_{i-2} is set to 0 and z_i is set to 1 instead. Otherwise, z_i is set to 0.

The computation of the different transmission pattern probabilities (i.e., the determination of the function $g(\cdot)$) is obtained by summing up the *path probability* of each of the paths leading to one of the five possible transmission patterns (state circled in bold in Figure 5). In other words, the probability of the pattern $[1\ 0\ 0\ 0]^T$ (resp., $[0\ 1\ 0\ 0]^T$) is the probability of having z_0 (resp. z_1) set to 1 at step 0, multiplied by the probability of keeping this selection at step 1 (i.e., no additional active link or stealing effect). Similarly, the probability of the pattern $[0\ 0\ 1\ 0]^T$ (resp., $[0\ 0\ 0\ 1]^T$) is obtained by adding: (i) the probability of having z_2 (resp. z_3) set to 1 at step 0, multiplied by the probability of having this selection maintained at step 1 and (ii) the probability of having z_0 (resp. z_1) set to 1 at step 0, multiplied by the probability of having the stealing effect at step 1. Finally, the probability of the pattern $[1\ 0\ 0\ 1]^T$ is obtained by adding: (i) the probability of having z_0 set to 1 at step 0 multiplied by the probability of having z_3 set to 1 at step 1 and (ii) the probability of having z_3 set to 1 at step 0 multiplied by the probability of having z_0 set to 1 at step 1. As in Figure 3, Figure 6 summarizes the transmission patterns probability (i.e., $g(\cdot)$) for each of the 8 regions of $\mathbb{Z}^3$: $A = \{0,0,0\}, \dots, H = \{b_1(n) > 0, b_2(n) > 0, b_3(n) > 0\}$.

B. Stability Analysis

Similarly to the 3-hop network, we model the queue evolution by the random walk in $\mathbb{N}^3$ depicted in Figure 6. However, contrary to the 3-hop case, the 4-hop case presents a structural factor that makes the system unstable either with or without the stealing effect as stated in Theorem 2.

Theorem 2: A 4-hop network is unstable for all $0 \leq p \leq 1$.

Proof: Starting with $p \neq 1$, we introduce the function

$$h(b_1, b_2, b_3) = b_1 + \frac{p}{1+p} b_3, \tag{5}$$

the constants $c = 3$, $d = 1$, $\epsilon = (1-p)/36$ and

$$k(i) = \begin{cases} 3 & \text{if } i \in \text{region } B \\ 2 & \text{if } i \in \text{region } D \\ 1 & \text{otherwise} \end{cases}, \tag{6}$$

Furthermore we introduce the notation

$$\begin{aligned} \mu_{k,b_1,b_2,b_3}(n) &= \mathbb{E}[h(\vec{b}(n+k)) | h(\vec{b}(n) = h(b_1,b_2,b_3))] \\ \epsilon_{k,b_1,b_2,b_3}(n) &= \mu_{k,b_1,b_2,b_3}(n) - h(b_1,b_2,b_3), \end{aligned}$$

where $\epsilon_{k,b_1,b_2,b_3}(n)$ is the drift of the k-step random walk, and verifies condition 2 of the Transience theorem (see Appendix) in Table I.

Region	ϵ-value
$A \cap S_c$	$\epsilon_{1,0,0,0} = 1 \geq \epsilon$
$B \cap S_c$	$\epsilon_{3,b_1,0,0} = \frac{1-p}{36} \geq \epsilon$
$C \cap S_c$	$\epsilon_{1,0,b_2,0} = \frac{1-p}{2} + \frac{1+p}{2}\frac{p}{1+p} = \frac{1}{2} \geq \epsilon.$
$D \cap S_c$	$\epsilon_{2,b_1,b_2,0} = \frac{1-p}{24} + \frac{p^2}{12} \geq \epsilon$ for $b_2 > 1$ $\epsilon_{2,b_1,1,0} = \frac{1-p}{18} \geq \epsilon$
$E \cap S_c$	$\epsilon_{1,0,0,b_3} = \frac{1}{1+p} \geq \epsilon$
$F \cap S_c$	$\epsilon_{1,b_1,0,b_3} = \frac{1-p}{6(1+p)} \geq \epsilon$
$G \cap S_c$	$\epsilon_{1,0,b_2,b_3} = \frac{4+p+p^2}{6(1+p)} \geq \epsilon$
$H \cap S_c$	$\epsilon_{1,b_1,b_2,b_3} = \frac{p^2+1}{8(1+p)} \geq \epsilon$

TABLE I
PROOF OF CONDITION 2 OF THE TRANSIENCE THEOREM FOR $p \neq 1$.

Consequently, as conditions 1 and 3 are trivially satisfied, the system is unstable for $p \neq 1$.

In the case $p = 1$, we prove the instability of the network by using the non-ergodicity theorem ([18], p. 30) with the Lyapunov function

$$h(b_1, b_2, b_3) = 2b_1 + b_3, \tag{7}$$

and setting the constants $c = d = 2$ in that theorem. Indeed, by computing the drift $\epsilon(\vec{b}(n)) = \epsilon_{1,b_1,b_2,b_3}(n)$, we obtain

$$\epsilon(\vec{b}(n)) = \begin{cases} 0 & \text{if } \vec{b}(n) \in \text{region } B, D, F \\ 1 & \text{if } \vec{b}(n) \in \text{region } C, E, G \\ 2/8 & \text{if } \vec{b}(n) \in \text{region } H. \end{cases} \tag{8}$$

Therefore, as we have non-negative values for all the regions of the space such that $h(\vec{b}(n)) > c$ and as the drift is upper-bounded by d, we end our proof for $p = 1$ by applying the non-ergodicity theorem.

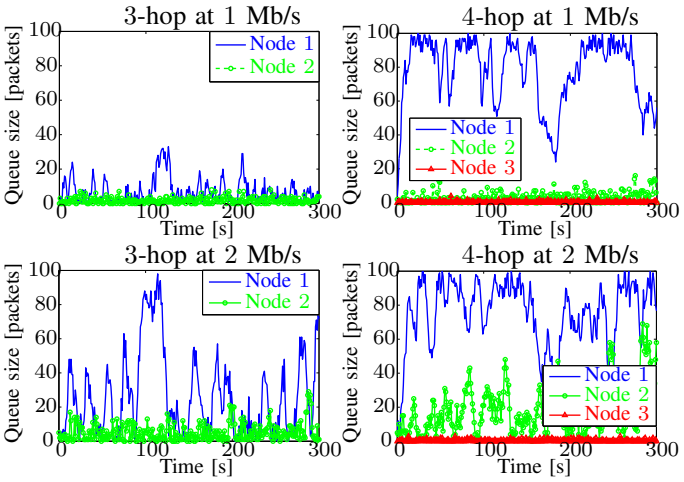

Fig. 7. Validation of the experimental results from Figure 1 on a different setup running at various data rate.

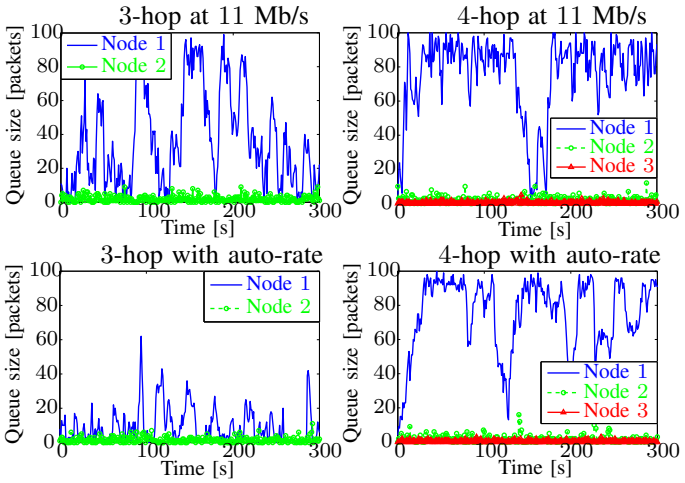

Fig. 8. Validation of the experimental results from Figure 1 on a different setup running at various data rate.

■

These results are fundamental for real networks as they reveal the tendency of CSMA to naturally produce instability for 4-hop topologies.

C. Extension to Larger K-hop Topologies

In the case without the stealing effect ($p = 0$), we can easily prove the network instability for $K = 2$, as we did in the previous sections for $K = 3, 4$. When $p = 0$, the instability of a K-hop topology for any $K > 4$ follows then from the following lemma.

Lemma 1 (K-hop Instability): If $p = 0$, a sufficient condition for a linear K-hop network to satisfy the conditions of the non-ergodicity theorem and thus to be unstable is that both the $(K-1)$ and $(K-3)$ hop networks satisfy the conditions of the non-ergodicity theorem.

Proof: Let us denote the next step expectation of a K-hop network by $\mu^K(n) = \mathbb{E}[h(\vec{b}(n+1)) \mid h(\vec{b}(n))]$. Here $h(\vec{b}) = b_1$ and therefore we can write

$$\mu^K(n) = \alpha\mu_0^K(n) + (1-\alpha)\mu_1^K(n) \tag{9}$$

where $\alpha = \mathbb{P}(z_{K-1}(n) = 0)$ and

$$\begin{aligned}\mu_0^K(n) &= \mathbb{E}\left[b_1(n+1) \mid b_1(n) = b_1, z_{K-1}(n) = 0\right] \\ &= \mu^{K-1}(n) \\ \mu_1^K(n) &= \mathbb{E}\left[b_1(n+1) \mid b_1(n) = b_1, z_{K-1}(n) = 1\right] \\ &= \mathbb{E}\left[b_1(n+1) \mid b_1(n) = b_1, z_{K-3}(n) = z_{K-2}(n) = 0\right] \\ &= \mu^{K-3}(n)\end{aligned}$$

where we have used (3) and the independence of $b_i(n+1) - b_i(n)$, $1 \le i \le K-3$, from $b_{K-2}(n)$ and $b_{K-1}(n)$, conditionally to $z_{K-3}(n) = z_{K-2}(n) = 0$. Therefore (9) becomes

$$\mu^K(n) = \alpha\mu^{K-1}(n) + (1-\alpha)\mu^{K-3}(n),$$

which implies that $\mu^K(n)$ verifies the inequalities of the non-ergodicity theorem if $\mu^{K-1}(n)$ and $\mu^{K-3}(n)$ do. ■

VI. Instability at Higher Rates

The analytical model of Section III allows us to explain why a stable 3-hop network becomes unstable when a 4^{th} hop is added (see Figure 1). Nevertheless, the results from Figure 1 are obtained with a fixed data rate of 1 Mb/s, a buffer size limit of 50 packets, and a small-scale testbed where the routers are used without their external antennas (better control on the experimental environment). In order to validate our results on a different setting, we modify the MadWifi driver to unlock the buffer size limit and to allow the modification of its value at run time through a simple command. We then set the buffer limit to 100 packets and repeat the experiment from Figure 1 on the real-scale deployment of Figure 9, with different data rate settings.

Figure 7 and 8 depict the queue evolution of a 3-hop network (from node 0 to node 3 in Figure 9) and a 4-hop network (from node 0 to node 4 in Figure 9) at data rates of: 1 Mb/s, 2 Mb/s, 11 Mb/s and auto-rate. Additionally, Table II presents the link throughputs and the end-to-end throughputs achieved at the different data rates.

Our results show that even though l_2 is the bottleneck link for all the data rates (i.e., the link with the smallest capacity when transmitting alone), the 3-hop network does not become unstable and this is because of the stealing effect described in Section III.C. Moreover, the simple addition of a 4^{th} hop turns the network from stable to unstable (i.e., the queue remains close to the buffer limit). We note that the queue size variations are larger than in Figure 1. This is because the

throughput\rate	1 Mb	2 Mb	11 Mb	auto-rate
l_0	894 kb/s	1.67 Mb/s	6.71 Mb/s	5.79 Mb/s
l_1	858 kb/s	1.52 Mb/s	5.82 Mb/s	2.03 Mb/s
l_2	754 kb/s	1.28 Mb/s	4.23 Mb/s	1.95 Mb/s
l_3	813 kb/s	1.6 Mb/s	5.98 Mb/s	5.49 Mb/s
3-hop	241 kb/s	493 kb/s	1.05 Mb/s	373 kb/s
4-hop	194 kb/s	354 kb/s	791 kb/s	260 kb/s

TABLE II
MEASUREMENTS OF THE LINKS THROUGHPUT AND THE END-TO-END THROUGHPUT OF A 3- AND 4-HOP LINEAR TOPOLOGY FOR DIFFERENT DATA RATES.

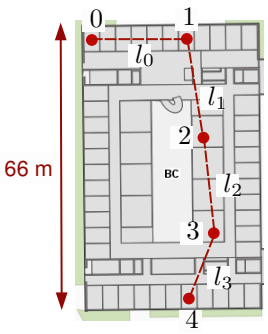

Fig. 9. Illustration of the deployment used in Section VI.

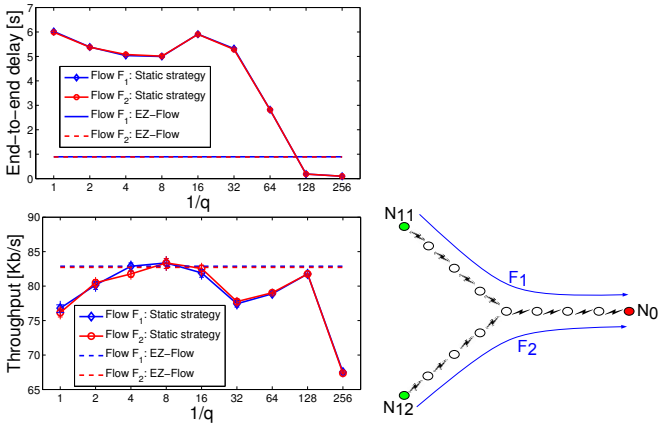

Fig. 10. Illustration of the evolution of the median of the delay and the averaged throughput (with confidence interval) depending on the throttling factor q. We note that the static value $q = 1/128$ stabilizes the network (i.e. low delay), while achieving a good throughput performance. As the optimal parameter q is topology dependent, we design a dynamical protocol, EZ-flow, that approaches the static performance.

real-scale deployment is a less controlled environment more prone to changing channel conditions. Nevertheless, we stress that, despite these variations, the change in stability between a 3-hop and 4-hop network is seen for all the different data rates that we tested, as predicted by our analytical model.

Our experimental results at higher rates also provide an interesting finding that is worth mentioning. Indeed, when observing the 3-hop results, we see that the queue variation of node 1 increases at higher rates.

To understand this finding, we need to recall that the transmission duration decreases at higher rates. This means that the period of vulnerability to the stealing effect decreases at higher rates and, therefore the probability of stealing effect p decreases as a function of the data transmission rate.

Finally, once we understand the relation between the data rate and the probability p, we note that our experimental results at higher rates confirm the simulation results of our analytical model presented in Figure 4. In other words, we see that the higher the rate (i.e., the smaller p), the closer the queue evolution gets to a null recurrent system.

VII. Simulations on multi-flows topologies

Up to this point in the paper, we have focused on single flow linear topologies as they are the building block of more general mesh topologies. However, to show that the stability problem also arises in more complex topologies, we present in this section the simulation results obtained with the ns-2 simulator. Moreover, we evaluate the *static stabilization strategy* proposed in [9] that uses a throttling factor q that reduces the channel access probability of the source, compared to the other nodes. This factor is defined as the ratio $q = cw_{src}/cw_{relay}$, where cw_{src} (cw_{relay}) is the CW_{min} contention window at the source (relay). We note that this strategy ensures that the first link becomes the bottleneck of the flow and Gao et al. show that in this situation offered load congestion control is not needed as it does not improve performance [20].

We analyze the multi-flow topology depicted in Figure 10, where two concurrent flows compete for the medium. We set the simulator to use the standard parameters of 802.11 ad-hoc networks (RTS/CTS disabled, Tx range: 250 m, Cs range: 550 m) and let the simulations run for $100,000$ s.

The two performance metrics we focus on are: (i) the end-to-end delay (low delays means that the network is stable, whereas high delay is a symptom of saturated buffers) and (ii) the throughput. Figure 10 shows the average performance achieved by the network as a function of the throttling factor q for the static stabilization strategy. We compute the throughput and the delay by measuring the average on disjoint 50 seconds intervals. Then we plot the median value with the 95%-confidence intervals. We note that standard 802.11 (i.e. $q = 1$) performs poorly as expected, with lower throughput and high end-to-end delays. Furthermore, using an appropriate throttling factor larger than for the single-flow case [9] (here $q = 1/128$), performance are significantly improved by achieving both negligible delay and higher global throughput due to a lower packet loss rate (as no buffer overflows in stable regime).

Nevertheless, the optimal throttling factor is hard to guess beforehand as it is topology dependent. Moreover, discovering it at run-time requires network-wide message passing in general topologies as the congestion might occur at any node of the network while only the source throttles itself. In order to avoid message passing, we designed EZ-flow, a dynamic hop-by-hop congestion control mechanism described in the next section; it automatically approaches the performance of the *static stabilization strategy* as depicted in Figure 10. EZ-flow does not require message passing, because all the nodes adapt their contention window, thus implicitly pushing back the congestion information to the source.

VIII. EZ-flow

A. System Requirements

In the design of our mechanism we focus on developing a practical, stabilizing solution that is compatible with current equipments and protocols used in IEEE 802.11 wireless mesh networks. Toward this goal, we set four main requirements:

- **Network stabilization**: EZ-flow is designed mainly to ensure network stability, where we define a network to be stable if all the relay nodes have their queue finite when equipped with infinite buffers. In practice, when buffers are finite, this means that no queue builds up. Furthermore, as the environment changes in real networks, we

require EZ-flow to automatically adapt itself to changes in the traffic matrix.

- **End-to-end delay reduction**: The first implication of network stability is a reduced end-to-end delay that should be maintained low with EZ-flow, compared with IEEE 802.11 alone. Such a requirement of low delays is of utmost importance in cases where a mesh network supports real-time, multimedia services such as VoIP, video-on-demand or online-gaming.
- **Unmodified MAC layer**: We require that the IEEE 802.11 MAC layer remains unmodified in order to ensure the compatibility of our solution with the mesh networks already deployed. To meet this objective, we propose to implement EZ-flow as a separate program that interacts with the MAC layer solely through the contention window CW_{min} parameter of IEEE 802.11.
- **Backward compatibility**: We ensure the backward compatibility of EZ-flow by having each node derive the needed information without message passing. This approach allows for the possibility of an incremental deployment of EZ-flow in an already existing mesh.

B. EZ-Flow Description

First, we introduce the notion of flow, where a flow is a directed communication between a source and a destination. In the multi-hop case, the intermediate nodes act as relays to transport the packets to the final destination. A node $i+1$ is the successor node of node i along a given flow if it is the next-hop relay in the multi-hop flow. We denote the buffer occupancy of node i by b_i and its minimal contention window (CW_{min}) by cw_i. In order, not to starve forwarded traffic, each node that acts both as a source and relay should maintain 2 independent queues: one for its own traffic and the other for the forwarded traffic. Furthermore, a node that has multiple successors should maintain 1 queue per successor (2 if it acts as source and relay). Indeed, different successors may encounter different congestion levels and thus EZ-flow performs best if it can adapt the channel access probability per successor. Note that, this requirement is scalable as EZ-flow does not need queuing per destination, but per successors and the number of successors is typically limited to a single digit in the case of a WMN.

Second, we describe the two modules forming EZ-flow: (i) a Buffer Occupancy Estimator (BOE) that derives the buffer status of the successor node along a flow and (ii) a Channel Access Adaptation (CAA) that uses the information from the BOE to adapt the channel access probability through cw_i.

C. Buffer Occupancy Estimation

One of the major novelties of EZ-flow lies in the BOE that passively derives the buffer occupancy at the successor node b_{i+1} without requiring any type of message passing. We emphasize that our BOE works differently than estimation approaches, such as [25], that sends probe packets to estimate the total queue size. Instead, in our approach each node i passively computes how many of its own packets are queued at node $i+1$. Using this information, instead of the total queue size, EZ-flow aims to keep the number of packets at a successor's queue small. This design choice prevents a node from starving itself due to non-cooperative neighbors (not performing congestion control).

To perform its task, the BOE keeps in memory a list L of the identifiers of the last 1000 packets it sent to a successor node. In our deployment we use the 16-bit checksum of the TCP or UDP packet as an identifier so as not to incur any computational overhead due to processing the packet. We note that this identifier, present in the packet header, could be used by any mesh network based on TCP/UDP and IP, and this is clearly the standard in currently deployed networks. Nevertheless, we stress that this design choice is used without any loss of generality. Even if, in the future, the standard would be to run IPsec or to use non-TCP/UDP packets, our mechanism would simply need to use a lightweight hash of the packet payload as an identifier instead.

The second information needed is the identifier of the packet that is actually forwarded by the successor node. This piece of information can be obtained by taking advantage of the broadcast nature of the wireless medium. Indeed, node i is on the range of $i+1$ and is thus able to hear most of the packets that are sent by node $i+1$ to $i+2$. In the usual settings, the MAC layer at each node transmits to the upper layer only the messages that are targeted to it and ignores the messages targeted to other nodes. However, by setting a node in the monitoring mode, it is possible to sniff packets that are targeted to other nodes through a raw socket (as tcpdump does [7]). Using such a methodology, it is then possible for a node to track which packets are being forwarded by its successor node without it requiring any message passing.

Finally, as the standard buffering policy is "First In, First Out" (FIFO), node i can accurately compute the number of its packets stored at node $i+1$ each time it hears a packet from node $i+1$. Indeed, it only needs to compare the identifier of the packet it hears with the identifiers of the sent packets it has in the list L. The number of packets between the corresponding match (the packet that node $i+1$ forwards) and the last packet that node i sent (the last entry in the list L) corresponds to b_{i+1}. It is important to note that the BOE module does not need to overhear all the packets forwarded by node $i+1$ in order to work. Instead, it is enough for it to be able to overhear some packets. Each time node i overhears a forwarded packet from node $i+1$ (which happens most of the time, experimentally), it can precisely derive the buffer occupancy and transmit it to the CAA that will react accordingly. Obviously, the more forwarded packets node i can overhear, the faster it can detect and react to congestion. Nevertheless, even in the hypothetical case where node i is unable to hear most of the forwarded packets, it will still adapt to the congestion and eventually set its contention window to the right value.

D. Channel Access Adaptation

The second module of EZ-flow is the CAA that adapts the channel access probability according to $\overline{b}_{i+1}$, which is the 50-sample average of the b_{i+1} derived by the BOE. The intuition behind EZ-flow is that in the case a successor node has already

Algorithm 1 EZ-flow mechanism at node i

BOE module:
if transmission of packet p to node $i+1$ **then**
  Store checksum of p in $PktSent[]$ (overwrite oldest entry if needed)
  $LastPktSent$ = checksum of p
else if sniffing of packet p from $i+1$ to $i+2$ **then**
  if checksum of $p \in PktSent[]$ **then**
    b_{i+1} = number of packets in $PktSent[]$ between p and $LastPktSent$
    return b_{i+1} to CAA module
  end if
end if

CAA module:
Require: Reception of 50 b_{i+1} samples from BOE
$\overline{b}_{i+1}$ = Average of 50 b_{i+1} samples
if $(\overline{b}_{i+1} > b_{max})$ **then**
  $count_{down} \leftarrow 0$; $count_{up} \leftarrow count_{up} + 1$
  if $(count_{up} >= log(cw_i))$ **then**
    $cw_i \leftarrow cw_i \cdot 2$; $count_{up} \leftarrow 0$
  end if
else if $(\overline{b}_{i+1} < b_{min})$ **then**
  $count_{up} \leftarrow 0$; $count_{down} \leftarrow count_{down} + 1$
  if $(count_{down} >= 15 - log(cw_i))$ **then**
    $cw_i \leftarrow cw_i/2$; $count_{down} \leftarrow 0$
  end if
else
  $count_{up} \leftarrow 0$; $count_{down} \leftarrow 0$
end if

many packets to forward, it is useless to send it more packets. Even worse, sending more packets degrades the performances. Indeed, every time node i sends a new packet to be forwarded, node $i+1$ looses a chance to transmit.

Following this result, we propose a simple policy for the CAA that uses solely two thresholds: (i) b_{min} and (ii) b_{max}. Then it adapts the channel access of each node by changing its value of the contention window cw_i. Indeed, every time node i needs to send a packet when the channel is not idle, it randomly chooses a backoff value that is inside the interval $[0, cw_i - 1]$ and it waits for this amount of time before retrying to transmit (see [8] for more details on how the backoff exactly works in IEEE 802.11). Therefore, we note that the higher the cw_i is, the lower the channel access probability is.

Our policy makes the decision based on a time average of the buffer occupancy at the successor node ($\overline{b}_{i+1}$). We set the time average parameter to be 50 samples and then one of three cases may occur:

- $\overline{b}_{i+1} < b_{min}$: the average queue at node $i+1$ is below the lower threshold. This shows that the buffer is underutilized. Thus node i should increase its channel access probability by dividing cw_i by a factor of two.
- $\overline{b}_{i+1} > b_{max}$: the average queue at node $i+1$ is above the upper threshold. This shows that the buffer is overutilized (or even overflows). Thus node i should decrease its channel access probability, which it does by doubling cw_i.
- $b_{min} < \overline{b}_{i+1} < b_{max}$: it is the desired situation as the buffer is correctly utilized by being neither empty most of the time nor saturated. In this case, node i concludes that it has a correct channel access probability and thus keeps cw_i unchanged.

Other policies than multiplicative-increase, multiplicative-decrease could be used to update cw_i in order to have a higher range of possible values. Yet, we chose this policy due to the hardware constraint that requires setting cw_i at powers of 2.

Furthermore, we provide a better inter-flow fairness in EZ-flow by using two parameters:

- $count_{up}$ counts the number of successive times the condition $(\overline{b}_{i+1} > b_{max})$ happens (overutilization).
- $count_{down}$ counts the number of successive times the condition $(\overline{b}_{i+1} < b_{min})$ happens (underutilization).

These two pieces of information are then used to update the contention window parameter according to the current cw_i value, where nodes with a high cw_i react both quicker to underutilization signals and slower to overutilization signals than nodes with a low cw_i react.

Finally, the selection of the parameters b_{min} and b_{max} can affect the reactivity and the speed of convergence of EZ-flow, depending on the topology. Indeed, the smaller the gap between these two values, the higher the reactivity of EZ-flow to slight variations, whether due to variations of the traffic load or not. These parameters can thus be fine tuned depending on the desired behavior, but fortunately the general values of b_{min} and b_{max} already significantly improve the situation compared to standard IEEE 802.11. Indeed, the most important parameter to set is b_{min}, which has to be very small (i.e., $\sim 10^{-1}$) in order to avoid that the nodes too often become too aggressive and reach unsupportable rates. The parameter b_{max} can then be set with more flexibility depending on the desired reactivity.

E. EZ-Flow Dynamical Model

Using the same notation as in Section III, the dynamics of a network using EZ-flow are captured by the recursive equations

$$cw_i(n+1) = f(cw_i(n), b_{i+1}(n)) \tag{10}$$

$$b_i(n+1) = b_i(n) + z_{i-1}(n) - z_i(n), \tag{11}$$

where $f(\cdot,\cdot)$ is defined by

$$f(cw_i(n), b_{i+1}(n)) = \begin{cases} \min(cw_i(n)\cdot 2, max_{cw}) & \text{if } (b_{i+1}(n) > b_{max}) \\ \max(cw_i(n)/2, min_{cw}) & \text{if } (b_{i+1}(n) < b_{min}) \\ cw_i(n) & \text{otherwise,} \end{cases}$$

with b_{max} and b_{min} being, respectively, the maximal and minimal threshold values for the buffer and $min_{cw} = 2^m$ and $max_{cw} = 2^M$ being the bounds between which the contention windows can evolve. Practical values are $m = 4$ and $M = 15$, thus we always take $M > m+1$. This discrete-time model is a Markov chain with the tuple $\{\vec{b}(n), c\vec{w}(n)\}$ as state, where $\vec{b}(n) \in \mathbb{N}^{K+1}$ and where $c\vec{w}(n)$ satisfies both $cw_i(n) \in \{2^m, 2^{m+1}, \cdots, 2^M\}$ and

$$cw_i(n) \geq 2^{m+\min(l, M-m)} \text{ when } b_{i+1}(n) > b_{max} + l, \tag{12}$$

Region	$\vec{z}$	$\mathbb{P}(\vec{z})$
A	$[1,0,0,0]$	1
B	$[1,0,0,0]$	$cw_1/(cw_0+cw_1)$
	$[0,1,0,0]$	$cw_0/(cw_0+cw_1)$
C	$[0,0,1,0]$	1
D	$[0,1,0,0]$	$\frac{cw_0cw_2}{\sum_{i=0,1,2}\prod_{j\neq i}cw_j}$
	$[0,0,1,0]$	$1-\frac{cw_0cw_2}{\sum_{i=0,1,2}\prod_{j\neq i}cw_j}$
E	$[1,0,0,1]$	1
F	$[0,0,0,1]$	$cw_0/(cw_0+cw_1)$
	$[1,0,0,1]$	$cw_1/(cw_0+cw_1)$
G	$[0,0,1,0]$	$cw_3/(cw_2+cw_3)$
	$[1,0,0,1]$	$cw_2/(cw_2+cw_3)$
H	$[0,0,1,0]$	$\frac{cw_0cw_1cw_3}{\sum_{i=0,1,2,3}\prod_{j\neq i}cw_j}+\frac{cw_1cw_2cw_3}{\sum_{i=0,1,2,3}\prod_{j\neq i}cw_j}\frac{cw_3}{cw_2+cw_3}$
	$[0,0,0,1]$	$\frac{cw_0cw_2cw_3}{\sum_{i=0,1,2,3}\prod_{j\neq i}cw_j}+\frac{cw_0cw_1cw_2}{\sum_{i=0,1,2,3}\prod_{j\neq i}cw_j}\frac{cw_0}{cw_0+cw_1}$
	$[1,0,0,1]$	$\frac{cw_1cw_2cw_3}{\sum_{i=0,1,2,3}\prod_{j\neq i}cw_j}\frac{cw_2}{cw_2+cw_3}+\frac{cw_0cw_1cw_2}{\sum_{i=0,1,2,3}\prod_{j\neq i}cw_j}\frac{cw_1}{cw_0+cw_1}$

TABLE III
PROBABILITY OF OCCURRENCE OF THE TRANSMISSION PATTERN $\vec{z}$ FOR THE DIFFERENT REGION OF THE SPACE $\mathbb{N}^3$.

where $l > 0$. The lower-bound condition (12) comes from the recursive application of (10) for the last l time slots ($b_{i+1}(k) > b_{max}$ for $n-l < k \leq n$ implies that $cw_i(k+1) = \min(cw_i(k)\cdot 2, 2^M)$). The state space is divided in 2^{K-1} regions, which differ by the entries of $\vec{b}$ that are zero and non-zero (i.e., the queues that are empty or not). Figure 6 illustrates these 8 regions for a 4-hop network (denoted A-H). In each region, one can compute first the possible outcomes of the back-off timers that depend on the contention values $c\vec{w}(n)$, and next the resulting transmission patterns that depend also on the possible collisions due to hidden terminals. The enumeration of all the possible outcomes is not included here for lack of space, but it follows the same reasoning as in Section V. It is summarized in Table III for the 4-hop network with a stealing effect $p = 1$ (i.e. no RTS/CTS).

F. Proof of Stability

Equipped with the model described above, we now formally prove the efficiency of EZ-flow in stabilizing the network. We give a proof, which holds when

$$b_{min} > M - m + 1. \tag{13}$$

This condition further reduces the state space of our model as, following a similar recursive argument than for (12), it implies that

$$cw_i(n) = 2^m \text{ when } b_{i+1}(n) = 0. \tag{14}$$

When $b_{min} \leq M - m + 1$, the proof uses computer-assisted computations, and is given in [11].

Theorem 3: EZ-flow stabilizes a 4-hop network by maintaining almost surely finite the queues of all the relaying nodes.

Proof: We apply Foster's theorem (see Appendix) with the Lyapunov function

$$h(b_1, b_2, b_3, cw_0, cw_1, cw_2, cw_3) = b_1 + b_2 + b_3,$$

and the finite set $S = \{cw_0, cw_1, cw_2, cw_3 \leq 2^M; 0 \leq b_1, b_2, b_3 \leq b_{max} + M - m + 3\}$. We need to verify that both conditions (15) and (16) of this theorem are verified for all points $\{\vec{b}(n), c\vec{w}(n)\}$ within the state space.

We note first that (15) is satisfied by the definition of h and the non-zero transition probabilities of the random walk.

It takes some more work to verify (16). One needs to compute

$$\epsilon_{k,\vec{b}}(n) = \mathbb{E}\left[h(\vec{b}(n + k(\vec{b}(n)))|\vec{b}(n)\right] - h(\vec{b}(n))$$

for all possible $c\vec{w}$ and with $\vec{b}(n)$ in each of the 7 regions B-H outside S, similarly to the proof of Theorem 2.

First, we note that the transition probabilities from Table III imply that:

$$\epsilon_{1,\vec{b}}(n) > 0 \text{ for } \vec{b}(n) \in B,$$

$$\epsilon_{1,\vec{b}}(n) < 0 \text{ for } \vec{b}(n) \in F \cup H,$$

$$\epsilon_{1,\vec{b}}(n) = 0 \text{ otherwise.}$$

Then, we find that after some computations that for all $c\vec{w}$, (16) is verified. In regions F and H, we directly have from Table III that

$$k(\vec{b}(n)) = 1 \text{ when } \vec{b}(n) \in F \cup H.$$

In regions D and E, we note that there is a strictly positive probability of having $\vec{b}(n+1) \in F \cup H$ and a zero probability of having $\vec{b}(n+1) \in B$. Therefore, we derive that

$$k(\vec{b}(n)) = 2 \text{ when } \vec{b}(n) \in D \cup E.$$

In region G, we see that there is a strictly positive probability of having $\vec{b}(n+1) \in D \cup H$ and a zero probability of having $\vec{b}(n+1) \in B$. Thus, this gives us that

$$k(\vec{b}(n)) = 3 \text{ when } \vec{b}(n) \in G.$$

In region C, there is a probability 1 of having $\vec{b}(n+1) \in G$. Hence, we conclude that

$$k(\vec{b}(n)) = 4 \text{ when } \vec{b}(n) \in C.$$

For region B, the demonstration is a little more complex. First, we use that for $\vec{b}(n) \in B \setminus S$, we have

$$b_1(n) > b_{max} + M - m + 3$$

and

$$b_2(n) = b_3(n) = 0.$$

Thus, it follows from (12) and (14) that

$$c\vec{w}(n) = [2^M, 2^m, 2^m, 2^m] \text{ for } \vec{b}(n) \in B \setminus S.$$

Next, we obtain $\epsilon_{3,\vec{b}}(n)$ by defining $E^x(\vec{b}(n))$ as the event that

$$h(\vec{b}(n+3)) - h(\vec{b}(n)) = x.$$

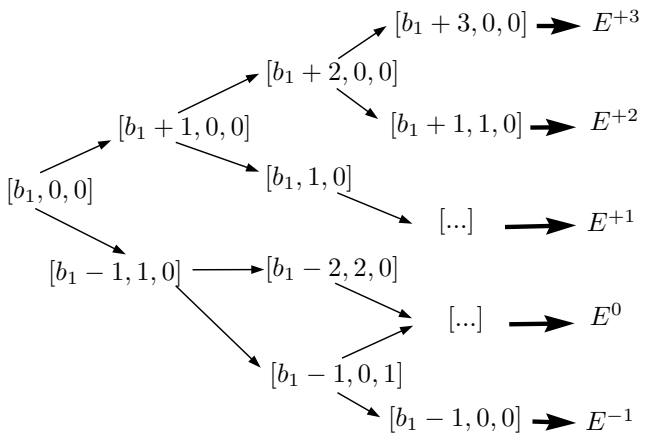

Fig. 11. Tree representing all possible transitions at steps $n+1$, $n+2$ and $n+3$ starting from $b(n) \in B \setminus S$. The five possible resulting events are are E^{+3}, E^{+2}, E^{+1}, E^0, E^{-1}; where $E^x = E^x(\vec{b}(n))$ is the event that $h(\vec{b}(n+3)) - h(\vec{b}(n)) = x$.

Then, we compute the probabilities for the five possible events $E^{+3}(\vec{b}(n))$, $E^{+2}(\vec{b}(n))$, $E^{+1}(\vec{b}(n))$, $E^0(\vec{b}(n))$, and $E^{-1}(\vec{b}(n))$ (see Figure 11). We obtain that

$$\begin{aligned}
\mathbb{P}(E^{+3}) &= 1/(1+2^{M-m})^3 \\
\mathbb{P}(E^{+2}) &= 1/(1+2^{M-m})^2 - 1/(1+2^{M-m})^3 \\
\mathbb{P}(E^{+1}) &= 1/(1+2^{M-m}) - 1/(1+2^{M-m})^2 \\
\mathbb{P}(E^{-1}) &= 2^{M-m}/(1+2^{M-m}) \cdot \\
&\quad (1-2^{M-m}/(2\cdot 2^{M-m}+1)) \cdot \\
&\quad 2^{M-m}/(1+2^{M-m}).
\end{aligned}$$

Then, we find that $\epsilon_{3,\vec{b}}(n) = 3 \cdot \mathbb{P}(E^{+3}(\vec{b}(n))) + 2 \cdot \mathbb{P}(E^{+2}(\vec{b}(n))) + \mathbb{P}(E^{+1}(\vec{b}(n))) - \mathbb{P}(E^{-1}(\vec{b}(n)))$, and because $M - m > 1$, we have that

$$\epsilon_{3,\vec{b}}(n) < 0.$$

Thus

$$k(\vec{b}(n)) = 3 \text{ satisfies (16) for } \vec{b}(n) \in B \setminus S.$$

Finally, as Region $A \subseteq S$, the conditions of Foster's theorem are satisfied in all $\{\vec{b}(n), c\vec{w}(n)\}$ within the state space, and it proves that EZ-flow stabilizes the network. ■

IX. Experimental validation

A. Hardware and Software Description

The testbed is composed of 4 laptops running Linux, which act as source and sink of the traffic, and 9 wireless nodes equipped with an omni-directional antenna that represent the multi-hop backhaul of a mesh network. The wireless routers are Asus WL-500gP, in which we change the mini-PCI WiFi card to an NMP-8602 Atheros card. Each router runs the OpenWRT firmware [6] with the MadWifi driver [4] modified to perform both buffer monitoring and the modification of the contention window. The wireless cards operate in 802.11b at a fixed transmission rate of 1 Mb/s and with the RTS/CTS mechanism disabled. Finally, we set the routing to be static.

We implement the two modules of EZ-flow, the BOE and CAA, in C code as described in Section VIII. Two practical constraints need to be accounted for. Both of them are not required in other implementations with different hardware.

1) **Sniffer constraint:** We initially intended to deploy both the BOE and CAA module within the same wireless card (i.e., the same router), but we had to reconsider our design. Indeed, the BOE acts mostly as a sniffer that collects the packets sent either by a node itself or its direct forwarder. The problem is that a WiFi card cannot transmit and receive at the same time and therefore is unable to really sniff its own packet on the air. Instead the best a sniffer can do is to capture the packet before it is sent to the MAC layer to be actually transmitted in the air. However, the drawback of this technique is that packets can be sniffed as sent by a node, even though they are dropped by the MAC layer (for example a buffer overflow), and thus are never really physically transmitted. To overcome this limitation, we use two WiFi interfaces per wireless node (i.e., two routers connected through an Ethernet cable). One interface is responsible for sending the traffic and running the CAA. The other interface does not transmit any packet and acts only as a sniffer that implements the BOE. We use this approach to simplify the practical deployment. EZ-flow does not require the use of two interfaces. Indeed, another approach could be to use only one interface and to directly implement EZ-flow at the kernel level of the wireless driver (and not the application level) in order for the BOE to capture only the packets that are truly sent at the physical layer.
2) **MadWifi constraint:** The second practical constraint comes from the *iwconfig* command of the Madwifi driver to increase the contention window CW_{min}. Indeed, it has no effect above 2^{10} (even though the driver allows the command to execute up to 2^{15}). We noticed this flaw in the implementation of the MadWifi command by checking a single-link capacity for different CW_{min} values and observing that it significantly varies up to 2^{10}, but it remains unchanged between 2^{10} and 2^{15}.

B. Topology Description

We deploy our testbed over 4 buildings of the university campus where at most 2 flows are concurrently active. Figure 12 presents the exact map of our mesh network deployment. On the one hand, the flow F_1 is a 7-hop flow for which the bottleneck link is l_2 as shown in Table IV. On the other hand, the flow F_2 is a shorter flow of 4 hops that shares the same path than F_1 and produces a typical parking-lot scenario. For the sake of comparability, we avoid the effect of interference from other networks by running our experiments on channel 12 during the night (1 am - 5 am), but we stress that the instability problem remains also during daytime as shown in our demo[1]. Finally, we use the values from Table IV to obtain the theoretical optima from Table V that assume a

[1]Demo available at: http://icawww1.epfl.ch/NetController/ (Video 2)

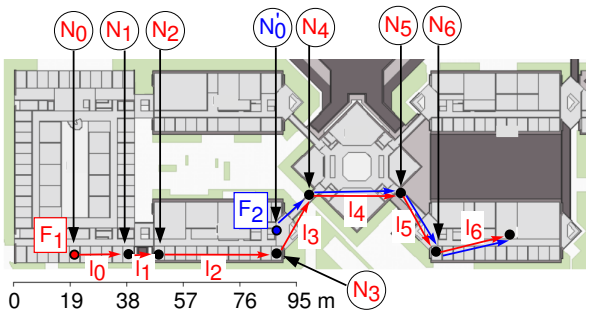

Fig. 12. Illustration of the testbed topology. The hardware used are Asus WL-500gP routers with an Atheros-based wireless card.

k-hop interference effect between the links with $k = 2$ and $k = 3$ (the experimental setup is somewhere between this two ranges). To do so, we compute the capacity of all paths of interfering links $C_j^{j+k} = 1/(\sum_{i=j}^{j+k} \frac{1}{C_i})$ for $0 \leq j \leq 6-k$, and where C_i is the capacity of link l_i. The theoretical optimum is then obtained by taking the capacity $C_{j'}^{j'+k}$ of the bottleneck path of interfering links within a flow.

C. Measurement Results

The first scenario we consider is when F_1 is alone in the network. Figure 13 shows the buffer evolution with standard IEEE 802.11 and with EZ-flow turned on. We note that for IEEE 802.11 both nodes N_1 and N_2 saturate and overflow, due to the bottleneck link l_2 (between N_2 and N_3), whereas all the other nodes have their buffer occupancy negligibly small, similarly to N_3. This results in an end-to-end throughput of 119 kb/s as shown in Table V (note that a similar throughput degradation for the backlogged case has been observed through simulation in [29]). In contrast, EZ-flows detects and reacts to the bottleneck at link l_2 by increasing cw_1 up to 2^8. This action stabilizes the buffer of N_2 by reducing the channel access of link l_1. Similarly, EZ-flow detects that the buffer of N_1 builds up and makes N_0 increase cw_0 until it reaches our hardware limit of 2^{10} (see Section 4.1). This hardware limitation prevents EZ-flow from reducing the buffer occupancy of N_1 to a value as low as N_2. However, we stress that despite this hardware limitation, EZ-flow still significantly improves the performance by reducing the turbulence of the flow and increasing the throughput to 148 kb/s (close to the 3-hop interference range theoretical optimum and mapping to a 41% reduction in the gap to the 2-hop optimum). Furthermore, we show through simulation in [11] that EZ-flow completely stabilizes the network once this limitation is removed.

In the second scenario, we consider F_2 alone. Similarly

	Mean throughput	Standard deviation
l_0	845 kb/s	23 kb/s
l_1	672 kb/s	49 kb/s
l_2	408 kb/s	67 kb/s
l_3	748 kb/s	42 kb/s
l_4	746 kb/s	28 kb/s
l_5	805 kb/s	27 kb/s
l_6	648 kb/s	43 kb/s

TABLE IV
ILLUSTRATION OF THE CAPACITY OF EACH LINK OF FLOW F_1. THE MEANS ARE OBTAINED THROUGH MEASUREMENTS OVER 1200 S.

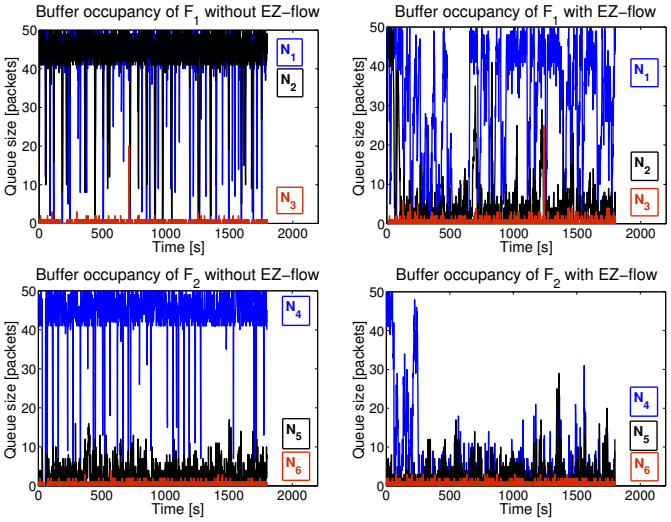

Fig. 13. Experimental results for the queue evolution of the relay nodes when flow F_1 or F_2 are active. The average number of buffered packets are: (i) without EZ-flow 41.6 (N_1), 43.1 (N_2) and 43.7 (N_4) and (ii) with EZ-flow 29.5 (N_1), 5.2 (N_2) and 5.3 (N_4). The remaining queues are very small.

to our mathematical analysis of Section V, we note that for IEEE 802.11 the buffer of the first relay node of F_2 (i.e., N_4) builds up and overflows, resulting in a throughput of 157 kb/s. However, EZ-flow completely stabilizes the network for all the relay nodes (no queue builds up) by making the source node $N_0^{'}$ increase $cw_0^{'}$ up to 2^8. Thus EZ-flow works even better in this scenario where it is not blocked by the hardware limitation and it achieves a throughput of 185 kb/s.

Finally the last scenario is a parking-lot scenario where both F_1 and F_2 are simultaneously active. Similarly to what is also reported in [39] between a 1- and 2-hop flow, Table V shows that IEEE 802.11 performs very poorly: the long flow F_1 is completely starved in favor of the short flow F_2, because $N_0^{'}$ is too aggressive (even for its own flow) and thus prevents the packets from the longer flow F_1 from being relayed by the intermediate nodes N_1, N_2, N_3. However, by its nature, EZ-flow solves the problem by making the two source nodes, $N_0^{'}$ and N_0, become less aggressive in order to stabilize their own flow. This approach thus solves the starvation problem and significantly increases both the aggregate throughput of F_1 and F_2 and Jain's fairness index.

Due to space limitation, we point to [10] for additional experimental and simulation results for dynamic flows with more complex topologies (e.g., bi-directional traffic).

	Mean throughput	Theoretical optima $k=3$	Theoretical optima $k=2$	Jain's Fairness
F_1	119 kb/s	151 kb/s	190 kb/s	
F_2	157 kb/s	183 kb/s	242 kb/s	
F_1 F_2	7 kb/s 143 kb/s			0.55
F_1^{EZ}	148 kb/s	151 kb/s	190 kb/s	
F_2^{EZ}	185 kb/s	183 kb/s	242 kb/s	
F_1^{EZ} F_2^{EZ}	71 kb/s 110 kb/s			0.96

TABLE V
MEASUREMENTS OVER 1800 S WITH AND WITHOUT EZ-FLOW. THE SUB-DIVISION IN THE TABLE SHOWS THE RESULTS FOR: (I) ONE SINGLE FLOW, AND (II) TWO SIMULTANEOUS FLOWS.

X. Conclusion

We addressed the problem of network stability in CSMA-based linear wireless mesh network and provided three main contributions. First, we identified two key factors impacting the stability: the network size and an artifact that we called stealing effect. Second, we proved analytically and showed experimentally that 3-hop networks are stable when we account for the stealing effect, but 4-hop networks (and presumably larger topologies) are not. Third, we proposed and designed EZ-flow, a new flow control mechanism for IEEE 802.11 WMNs. EZ-flow is fully backward compatible with the IEEE 802.11 standard and works without any form of message passing. EZ-flow is implemented in a distributed fashion as a simple program running at each relay node. It takes advantage of the broadcast nature of the wireless medium to passively estimate the buffer occupancy at a successor node. The minimum congestion window parameter is adapted at each relay node based on this estimation to ensure a smooth flow, specifically, each relay node adapts its contention window to avoid buffer build-up at its successor node.

We demonstrated by experiments the attendant benefits of EZ-flow on a testbed composed of 9 standard wireless mesh routers deployed over 4 different buildings. Our measurement results show that EZ-flow simultaneously improves throughput and fairness performance. To our knowledge, it is the first implementation of an algorithm addressing instability in a real multi-hop network. Moreover, we derived a Lyapunov function with which we analytically prove the stability of an 802.11-based linear K-hop topology implementing EZ-flow.

We conclude by noting that the methodology followed by EZ-flow is not limited to line topologies. One possible approach to dealing with more general topologies is to take advantage of IEEE 802.11e, which uses four different MAC-layer queues. This protocol was originally designed to support Quality of Service (QoS) by categorizing the traffic into four types of service: (i) Background (BK), (ii) Best Effort (BE), (iii) Voice (VO) and (iv) Video (VI). Yet to date, this service differentiation is not commonly used and almost all traffic is classified as BE and queued accordingly. Thus, the three other queues are mostly left idle. A node forwarding traffic to up to four successors could take advantage of the availability of these MAC-layer queues in order to use one different queue (thus one different CW_{min} value) per successor. This approach suits well the backhaul scenario this paper focuses on, as it usually follows a tree-based topology with a limited number of neighbors. In cases where EZ-flow needs to be deployed in networks with a higher neighbor density, a similar mechanism could be used with a slight modification. Here, multiple queues could be implemented at the routing layer (e.g. by using Click [28]). The BOE would remain unchanged; and the CAA would control the scheduling rate at which packets belonging to different routing queues are delivered to the MAC layer, instead of directly modifying the MAC contention window.

Appendix

Theorem 4 (Foster [18], p. 30): Let the transition probability matrix P on the state space $\mathbb{N}^K$ (with $K > 0$) be irreducible and suppose that there exists a positive function $h : \mathbb{N}^K \to \mathbb{R}$ such that for some finite set S, some $\epsilon > 0$ and some positive integer-valued function $k : \mathbb{N}^K \to \mathbb{R}$ where $\sup_{\vec{b}\in\mathbb{N}^K} k(\vec{b}(n)) < \infty$ the following conditions hold

$$\mathbb{E}\left[h(\vec{b}(n+1)) \mid \vec{b}(n) = \vec{i}\right] = \sum_{\vec{k}\in\mathbb{N}^2} p_{\vec{i}\vec{k}} h(\vec{k}) < \infty \qquad (15)$$

for all $\vec{i} \in S$ and

$$\mathbb{E}\left[h(\vec{b}(n + k(\vec{b}(n)))|\vec{b}(n) = \vec{i}\right] \leq h(\vec{i}) - \epsilon k(\vec{b}(n)) \qquad (16)$$

for all $\vec{i} \notin S$. Then the corresponding Homogeneous Markov Chain (HMC) is *ergodic*.

Theorem 5 (Transience [18], p. 31): For an irreducible Homogeneous Markov Chain (HMC) to be transient, it suffices that there exist a positive function $h(\vec{i}), \vec{i} \in \mathbb{Z}^3$, a bounded integer-valued positive function $k(\vec{i}), \vec{i} \in \mathbb{Z}^3$, and numbers $\epsilon, c, d > 0$, such that, setting $S_c = \{\vec{i} : h(\vec{i}) > c\} \neq 0$, the following conditions hold:

1) $\sup_{\vec{i}\in\mathbb{Z}^3} k(\vec{i}) = k < \infty$;
2) $\mathbb{E}[h(\vec{b}_{n+k(\vec{i})})|h(\vec{b}_n) = h(\vec{i})] - h(\vec{i}) \geq \epsilon$, $\forall n$, for all $\vec{i} \in S_c$;
3) for some $d > 0$, the inequality $|h(\vec{i}) - h(\vec{j})| > d$ implies $p_{\vec{i}\vec{j}} = 0$.

Acknowledgment

This work is supported in part by Deutsche Telekom Laboratories, in the framework of the Magnets project, by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 and by the US National Science Foundation under grants CCF-0729158 and CCF-0916892.

We are grateful to Julien Herzen for his help in implementing the first version of EZ-flow on the Asus routers and to Seva Shneer for his valuable feedbacks on the analytical model.

References

[1] *The Cloud*. http://www.thecloud.net/.
[2] *EarthLink*. http://www.earthlink.net/.
[3] *LUNAR project*. http://cn.cs.unibas.ch/projects/lunar/.
[4] *Madwifi/AtherosWireless Linux Driver Users Guide*.
[5] *MIT Roofnet*. http://pdos.csail.mit.edu/roofnet/.
[6] *OpenWRT firmware*. http://openwrt.org/.
[7] *The Tcpdump Manual Page*. http://www.tcpdump.org/.
[8] *IEEE 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Aug. 1999.
[9] A. Aziz, D. Starobinski, and P. Thiran. Elucidating the instability of random access wireless mesh networks. In *Proc. of SECON*, Rome, Italy, June 2009.
[10] A. Aziz, D. Starobinski, and P. Thiran. Understanding and tackling the root causes of instability in wireless mesh networks: (extended version). Technical Report EPFL-REPORT-151661, EPFL, 2010.
[11] A. Aziz, D. Starobinski, P. Thiran, and A. El Fawal. EZ-Flow: Removing turbulence in IEEE 802.11 wireless mesh networks without message passing. In *Proc. of CoNEXT*, Rome, Italy, Dec. 2009.
[12] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. In *Proc. of SIGCOMM*, Philadelphia, PA, Aug. 2005.
[13] S. Borst, M. Jonckheere, and L. Leskelä. Stability of parallel queueing systems with coupled service rates. *Discrete Event Dynamic Systems*, 18(4):447–472, 2008.

[14] P. Chapokar, H. Kar, X. Luo, and S. Sarkar. Throughput and fairness guarantees through maximal scheduling in wireless networks. *IEEE Transactions on Information Theory*, 54(2):572–594, Feb. 2008.
[15] O. Dousse. Revising buffering in CSMA/CA wireless multihop networks. In *Proc. of SECON*, San Diego, CA, June 2007.
[16] M. Durvy and P. Thiran. A packing approach to compare slotted and non-slotted medium access control. In *Proc. of INFOCOM*, Barcelona, Spain, Apr. 2006.
[17] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. In *Proc. of INFOCOM*, Miami, FL, Mar. 2005.
[18] G. Fayolle, V. A. Malyshev, and M. V. Menshikov. *Topics in constructive theory of countable Markov chains*. Cambridge University Press, 1995.
[19] V. Gambiroza, B. Sadeghi, and E. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *Proc. of MobiCom*, Philadelphia, PA, Sept. 2004.
[20] Y. Gao, D.-M. Chiu, and J. C. Lui. Determining the end-to-end throughput capacity in multi-hop networks: methodology and applications. In *Proc. of SIGMETRICS*, Saint-Malo, France, 2006.
[21] M. Garetto, T. Salonidis, and E. W. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. *IEEE/ACM Transactions on Networking*, 16(4):864–877, 2008.
[22] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. In *Proc. of INFOCOM*, Anchorage, Alaska, 2007.
[23] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle sense: An optimal access method for high throughput and fairness in rate diverse wireless lans. In *Proc. of SIGCOMM*, Philadelphia, PA, Aug. 2005.
[24] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *to appear in IEEE/ACM Transactions on Networking*.
[25] W. Jiang. Accurate queue length estimation in wireless networks. In *Proc. of PAM*, Louvain-la-Neuve, Belgium, 2007.
[26] A. Jindal and K. Psounis. The achievable rate region of 802.11-scheduled multihop networks. *IEEE/ACM Transactions on Networking*, 17(4):1118–1131, 2009.
[27] S. Katti, H. Rahul, H. Wenjun, D.Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking*, 16(3):497–510, 2008.
[28] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, Aug 2000.
[29] J. Li, C. Blake, D. S. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proc. of MobiCom*, 2001.
[30] W. Luo and A. Ephremides. Stability of n interacting queues in random-access systems. *IEEE Transactions on Information Theory*, 45(5):1579–1587, July 1999.
[31] P. Marbach and A. Eryilmaz. A backlog-based CSMA mechanism to achieve fairness and throughput-optimality in multihop wireless networks. In *Proc. of Allerton*, 2008.
[32] E. Modiano, D. Shah, and G. Zussman. Maximizing throughput in wireless networks via gossiping. In *Proc. of SIGMETRICS*, Saint-Malo, France, 2006.
[33] N. Nandiraju, D. Nandiraju, D. Cavalcanti, and D. Agrawal. A novel queue management mechanism for improving performance of multihop flows in IEEE 802.11s based mesh networks. *In Proc. of IPCCC*, 2006.
[34] A. Proutière, Y. Yi, and M. Chiang. Throughput of random access without message passing. In *Proc. of CISS*, Princeton, NJ, Mar. 2008.
[35] B. Radunović, C. Gkantsidis, D. Gunawardena, and P. Key. Horizon: balancing TCP over multiple paths in wireless mesh network. In *Proc. of MobiCom*, San Francisco, USA, 2008.
[36] S. Rangwala, A. Jindal, K.-Y. Jang, and K. Psounis. Understanding congestion control in multi-hop wireless mesh networks. In *Proc. of MobiCom*, San Francisco, CA, May 2008.
[37] T. Salonidis, G. Sotiropoulos, R. Guerin, and R. Govindan. Online optimization of 802.11 mesh networks. In *Proc. of CoNEXT*, Rome, Italy, Dec. 2009.
[38] B. Scheuermann, M. Transier, C. Lochert, M. Mauve, and W. Effelsberg. Backpressure multicast congestion control in mobile ad-hoc networks. In *Proc. of CoNEXT*, New York, NY, USA, Dec. 2007.
[39] J. Shi, O. Gurewitz, V. Mancuso, J. Camp, and E. Knightly. Measurement and modeling of the origins of starvation in congestion controlled mesh networks. In *Proc. of INFOCOM*, Phoenix, AZ, Apr. 2008.
[40] J. Shin, D. Shah, and S. Rajagopalan. Network adiabatic theorem: An efficient randomized protocol for contention resolution. In *Proc. of SIGMETRICS*, Seattle, WA, June 2009.
[41] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, Dec. 1992.
[42] P. van de Ven, S. Borst, and S. Shneer. Instability of maxweight scheduling algorithms. In *Proc. of INFOCOM*, Brazil, 2009.
[43] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee. Diffq: Practical differential backlog congestion control for wireless networks. In *Proc. of INFOCOM*, Rio de Janeiro, Brazil, 2009.
[44] Y. Yi, A. Proutière, and M. Chiang. Complexity in wireless scheduling: Impact and tradeoffs. In *Proc. of MobiHoc*, Hong Kong, China, 2008.
[45] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *IEEE/ACM Transactions on Networking*, 15(1):133–144, 2007.
[46] L. Ying, R. Srikant, and D. Towsley. Cluster-based back-pressure routing algorithm. In *Proc. of INFOCOM*, Phoenix, AZ, Apr. 2008.

Adel Aziz received his M.Sc. degree in Communication Systems from EPFL, Switzerland, in 2006 and his M.Sc. degree in Management from HEC Lausanne, Switzerland, in 2009. In 2004, he was a research intern with CPqD, Campinas, Brazil. In 2006, he worked on his master's thesis project at Qualcomm, San Diego, USA. Since 2006, he is a Ph.D. student in Communication Systems at EPFL and he works on a project in collaboration with Deutsche Telekom Laboratories, Berlin, Germany. His research interests include the analysis and design of medium access control mechanisms for multi-hop wireless mesh networks

David Starobinski received his Ph.D. in Electrical Engineering (1999) from the Technion-Israel Institute of Technology. In 1999-2000 he was a visiting post-doctoral researcher in the EECS department at UC Berkeley. In 2007-2008, he was an invited Professor at EPFL (Switzerland). Since September 2000, he has been with Boston University, where he is now an Associate Professor.

Dr. Starobinski received a CAREER award from the U.S. National Science Foundation (2002), an Early Career Principal Investigator (ECPI) award from the U.S. Department of Energy (2004), the best paper award at the WiOpt 2010 conference, and the 2010 BU ECE Faculty Teaching Award. He is on the Editorial Board of the IEEE/ACM Transactions on Networking. His research interests are in the modeling and performance evaluation of high-speed, wireless, and sensor networks.

Patrick Thiran received the Electrical Engineering degree from the Université Catholique de Louvain, Louvain-la-Neuve, Belgium, in 1989, the M.Sc. degree in Electrical Engineering from the University of California at Berkeley, USA, in 1990, and the Ph.D. degree from EPFL, in 1996. He is an Associate Professor at EPFL. He became an Adjunct Professor in 1998, an Assistant Professor in 2002 and an Associate Professor in 2006. From 2000 to 2001, he was with Sprint Advanced Technology Labs, Burlingame, CA. His research interests include communication networks, performance analysis, dynamical systems, and stochastic models. He is currently active in the analysis and design of wireless multihop networks and in network monitoring. Dr. Thiran served as an Associate Editor for the IEEE Transactions on Circuits and Systems in 1997-1999, and he is currently an Associate Editor for the IEEE/ACM Transactions on Networking. He was the recipient of the 1996 EPFL Ph.D. award and of the 2008 Crédit Suisse Teaching Award.