EZ-Flow: Removing Turbulence in IEEE 802.11 Wireless Mesh Networks without Message Passing

Adel Aziz[†], David Starobinski[‡], Patrick Thiran[†], Alaeddine El Fawal[†]
[†] School of Computer and Communication Sciences, EPFL, Switzerland
{adel.aziz, patrick.thiran, alaeddine.elfawal}@epfl.ch
[‡] Boston University, USA
staro@bu.edu

ABSTRACT

Recent analytical and experimental work demonstrate that IEEE 802.11-based wireless mesh networks are prone to turbulence. Manifestations of such turbulence take the form of large buffer build-up at relay nodes, end-to-end delay fluctuations, and traffic congestion. In this paper, we propose and evaluate a novel, distributed flow-control mechanism to address this problem, 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. We further corroborate these results with a mathematical stability analysis and extensive ns-2 simulations run for different traffic workloads and network topologies.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Design, Experimentation

Keywords

Congestion control, multi-hop, wireless, IEEE 802.11, EZ-flow

1. 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 within their boundaries [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 (see Figure 2).

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 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. Recent analytical and experimental work [9, 12, 15] show that 802.11-based wireless mesh networks are susceptible to turbulence that takes the form of buffer build up and overflow at relaying nodes, major end-to-end delay fluctuations, and reduced throughput. In fact, the work in [9] rigorously proves the inherent instability of IEEE 802.11 WMNs longer than 3-hops. We depict in Figure 1 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 almost twice smaller than in the 3-hop case. The intrinsic instability of IEEE 802.11 mesh networks that are longer than three hops may explain why current implementations use only a few hops [3].

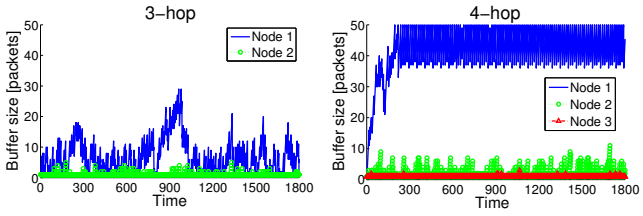


Figure 1: Experimental results for the buffer evolution of each relay node in 3- and 4-hop topologies. A 3-hop network is stable, whereas a 4-hop is turbulent with the buffer of its first relaying node building up until saturation.

CoNEXT'09, December 1–4, 2009, Rome, Italy.

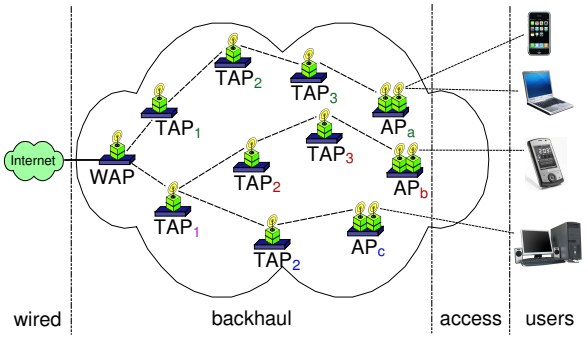


Figure 2: A wireless mesh network consists of a backhaul and an access part.

The problem of devising distributed channel access mechanisms to ensure stability in multi-hop networks has received much attention since the seminal work of Tassiulas et al. [30]. Most of the solid, analytical work on this problem [11, 17, 29, 32, 34] 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 using existing wireless cards. Indeed, despite all the previous theoretical work, no solution has been implemented and tested to date. 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 derive a practical and decentralized mechanism to solve this problem.

In this paper, we propose, analyze, and report experimental results for a new, distributed flow-control mechanism, called *EZ-flow*, that solves the turbulent behavior of IEEE 802.11 WMNs. EZ-flow requires no modification 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 an existing parameter of IEEE 802.11, the minimum contention window CW_{min}.

The standard way to obtain buffer occupancy information is via message passing. Message passing, however, may further exacerbate congestion and reduce resources available for sending useful data [32]. 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 represents one of the major advantages of EZ-flow as it enables the network to achieve stability without incurring any communication overhead.

The rest of this paper is organized as follows. After detailing the problem and stating the design requirements of EZ-flow, we review the related work in Section 2. The core of EZ-flow is based on two modules: (i) a Buffer Occupancy Estimator (BOE), which is responsible for inferring the buffer occupancy at the successor node and (ii) a Channel Access Adaptation (CAA), which uses this buffer information to appropriately adapt the channel access probability (i.e. the contention window CW_{min}). We thoroughly detail the intuition behind the mechanism and the technical implementation challenges for both modules in Section 3. We demonstrate the practical feasibility and performance gain achieved with EZ-flow in Section 4, where we present our experimental testbed and describe our measurement results. In Section 5, we extend our study to other topologies and traffic matrices through simulations. In Section 6, we mathematically prove the stabilization effect of EZ-flow for a linear topology, via a Lyapunov stability analysis. We summarize our findings in Section 7.

2. BACKGROUND

2.1 Problem Statement

We consider the case of a wireless multi-hop topology as the one existing in the backhaul of a mesh network. As depicted in Figure 2, the backhaul of a wireless mesh 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.

2.2 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 [9]. 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 to 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.

In addition to these requirements we add two properties that are not primary goals of EZ-flow, but that are still desired properties that appeared in all our simulations and experimental deployments:

- **Fairness improvement**: We take IEEE 802.11 as a baseline for EZ-flow and note that fairness is improved through a higher Jain's fairness index value

$$FI = \frac{(\sum x_i)^2}{(n_{flows} \cdot \sum x_i^2)} \quad (1)$$

where n_{flows} is the total number of flows and x_i is the throughput achieved by flow i.

- **Fair throughput improvement**: Low delays and high throughput are often seen wrongly as antagonist goals to be pursued. Of course, reducing the application throughput to a very low value will always ensure low delays. Nevertheless, we do not want our mechanism to limit itself to ensuring delay at the price of throughput. For the same level of fairness, we therefore require that EZ-flow achieves a global throughput that is higher than with IEEE 802.11 alone.

2.3 Related Work

Much effort has been put into understanding how IEEE 802.11 behaves in a multi-hop environment. Previous work show the inefficiency of the protocol in providing optimal performance, as far as delay, throughput and fairness are concerned [15]. In [24], 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 [19], Jindal et al. 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 4, show that the performance may actually be much worse. We believe that the cause of the discrepancy is that [19] assumes that flows are rate-controlled at the source, whereas we do not make such an assumption.

References [9, 28] propose analytical models to explain the causes of network instability and flow starvation occurring in WMNs. Our previous work [9] shows that IEEE 802.11 networks with more than 3 hops are intrinsically unstable. This work also proposes a penalty strategy that efficiently stabilizes a network through the use of a throttling factor $q \in [0, 1]$ for each flow, where q is the ratio between the contention window at the source node and at the relay nodes. Despite its efficiency, this penalty strategy has a drawback in that the parameter q is topology-dependent. Hence, we need to develop a self-adaptive algorithm such as EZ-flow to automatically discover a contention window distribution (i.e., q) that stabilizes the network independently of the topology.

A first analytical solution to the stability problem in multi-hop networks is discussed in the seminal work of Tassiulas et al. [30], which introduces a back-pressure algorithm. Their methodology uses a centralized scheduler that selects for transmission the link with the greatest buffer 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. Extensions from this work to distributed scheduling strategies have been discussed in works such as [11], 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 [34], 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 et al. in [13]. The tradeoff that exists in each scheduling strategy between complexity, utility and delay is discussed in depth in [32]. One of the drawbacks of these previous methods is that they require buffer information from other nodes. The usual solution is to use message passing, which produces an overhead and is thus costly even if it is limited to the direct neighbors.

Some recent work propose schedulers that do not require buffer information from other nodes. In [17], Gupta et al. propose an algorithm that uses the maximum node degree in the network. Proutiere et al. [25] propose another algorithm, where each node makes the scheduling decision based solely on its own buffer. Finally, most recently Shin et al. propose an algorithm that achieves stability and where each node uses its own buffer occupancy with a $\log \log$ function to make the scheduling decision [29]. Nevertheless, even though their algorithm is efficient 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, which usually have a standard MAC buffer of only 50 packets. To sum up, despite recent and significant progress on the theoretical side, almost all the existing solutions are still far from being compatible with the current IEEE 802.11 protocol. One exception, which was developed in parallel with our work, is the hop-by-hop congestion control scheme in [31]. In their paper, Warrier et al. propose and deploy DiffQ, which is a protocol implementing a form of backpressure (i.e., prioritizing links with large backlog differential). To achieve this implementation, DiffQ makes each node inform its neighbors of its queue size by piggybacking this information in the data packet (i.e., modifying the packet structure by adding an additional header) and then schedules the packets in one of the four MAC queues (each with different CW_{min} value) depending on the backlog difference. Our approach differs in two ways: (i) we use the next-hop buffer 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. Our approach differs from all the previous works in the sense that we propose a practical solution, implemented with off-the-shelf hardware, where we take advantage of the broadcast nature of the wireless medium to derive the buffer information of neighboring nodes. We also highlight that the novel passive buffer derivation methodology of our BOE module is potentially compatible with new algorithms such as DiffQ, and it could allow them to eliminate the need to piggy-back the buffer information (resulting in unmodified packet structure).

Another line of work, parallel to ours, tackles congestion at the transport layer rather than the MAC (link) layer. In [26], 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 [28]. Garetto et al. also tackle the starvation problem at an upper layer [16]. They propose a rate-limiting solution and evaluate it by simulation. Their major motivation for not using 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 since 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 and that our methodology solves the problem both for bi-directional traffic (e.g TCP) or uni-directional traffic that cannot count on feedbacks from the final destination to adapt its rate (e.g. UDP). The work of Yi et al. showing that a hop-by-hop congestion control outperforms an end-to-end version further motivates our approach [33].

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 [21], 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 [10], Biswas et al. present a routing mechanism named ExOR that takes advantage of the broadcast nature to achieve cooperative diversity and thus increase the achievable throughput. Furthermore, in [18] 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 that, our approach is different, because we do not use cooperation and network coding techniques at relay nodes, but instead in a competitive context we derive and use the next-hop buffer information to tackle the traffic congestion occurring in multi-hop scenarios.

To work in combination with routing solutions such as ExOR, our approach could be extended . Truly, 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 precisely need to know 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 to be forwarded at all of its successors. This could be done using a similar methodology to the one presented in this paper for the unicast case. A similar extension of a congestion-control from unicast to multicast is discussed by Scheuermann et al. in [27].

3. EZ-FLOW MECHANISMS

3.1 EZ-Flow Description

EZ-flow is a mechanism that interacts with the MAC layer in order to efficiently adapt the channel access parameter so as to match the requirement described in Section 2.

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 relay to transport the packets to the final destination. A node N_{k+1} is the successor node of N_k along a given flow if it is the next-hop relay in the multi-hop flow. We denote the buffer occupancy of N_k by b_k and its minimal contention window by cw_k. In order, not to starve forwarded traffic, each node that acts both as source and relay should maintain 2 independent queues: one for its own traffic and another 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 level 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 backhaul.

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_k.

3.2 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_{k+1} without requiring any type of message passing.

To achieve this performance, the BOE keeps in memory a list of the identifiers of the last 1000 packets it sends 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 highlight 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 just need to use a lightweight hash of the packet payload as an identifier instead.

Then 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 N_k is on the range of N_{k+1} and is thus able to hear most of the packets that are sent by N_{k+1} to N_{k+2}. In the usual settings, the MAC layer at each node only transmits to the upper layer 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 requiring any message passing.

Finally, as the standard buffering policy is "First In, First Out" (FIFO), N_k can estimate b_{k+1} each time it hears a packet from N_{k+1}. Indeed, it then compares the packet it hears with the identifiers of the sent packets it has in memory and the number of packets between the last packet node N_k sent and the last packet node N_{k+1} forwards correspond to b_{k+1} (as the intermediate packets are in the FIFO buffer). It is important to note that the BOE module does not need to overhear all the packets forwarded by N_{k+1} in order to work. Instead, it is enough for it to be able to overhear some packets. Each time N_k overhears a forwarded packet from N_{k+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 N_k can overhear, the faster it can detect and react to congestion. Nevertheless, even in the hypothetical case where N_k 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. This invulnerability of EZ-flow to forwarded packets that are not overhead is a crucial property, as some packets may be missed due to the variability of the wireless channel or hidden node situations.

3.3 Channel Access Adaptation

The second module of EZ-flow is the CAA that adapts the channel access probability according to $\overline{b}_{k+1}$, which is the 50 samples average of the b_{k+1} derived by the BOE. The intuition behind EZ-flow is that in the case a successor node has already many packets to forward, it is useless to send it more packets. Even worse, sending more packets degrades the performances. Indeed, every time node N_k sends a new packet to be forwarded, N_{k+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

Algorithm 1 EZ-flow mechanism at node N_k

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

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

contention window cw_k. Indeed, every time the node N_k needs to send a packet when the channel is not idle, it randomly chooses a backoff value that is inside the interval $[0, cw_k - 1]$ 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_k, the lower the channel access probability.

Our policy makes the decision based on a time average over 50 samples of the buffer occupancy at the successor node ($\bar{b}_{k+1}$) and thus one of three cases may occur:

- $\bar{b}_{k+1} < b_{min}$, where the average buffer occupancy at N_{k+1} is below the lower threshold. This shows that the buffer is underutilized. Thus N_k should increase its channel access probability, which it does by dividing cw_k by a factor two.
- $\bar{b}_{k+1} > b_{max}$, where the average buffer occupancy at N_{k+1} is above the upper threshold. This shows that the buffer is overutilized (or even overflows). Thus N_k should decrease its channel access probability, which it does by doubling its cw_k value.
- $b_{min} < \bar{b}_{k+1} < b_{max}$, which is the desired situation as the buffer is rightly utilized by neither being empty most of the time or being saturated. In this case, N_k concludes that it has a correct channel access probability and thus keeps its cw_k unchanged.

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

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

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

These two pieces of information are then used to update the contention window parameter according to the current cw_k value, where nodes with a high cw_k react both quicker to underutilization signals and slower to overutilization signals than nodes with a low cw_k 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 variation 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 become too often too aggressive and reach unsupportable rates. The parameter b_{max} can then be set with more flexibility depending on the desired reactivity level.

4. EXPERIMENTS

We implement EZ-flow on a testbed composed of 9 off-the-shelf wireless nodes. First, we describe the environment and hardware used in our experiment, and we discuss the practical details for the implementation of EZ-flow. Then, we present the real measurement results that confirm the efficiency of EZ-flow in improving the performance in a real WMN testbed environment.

4.1 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 version *Kamikaze 7.07* of 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 IEEE 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 Section 3. Two practical constraints need to be taken into account in the testbed. 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

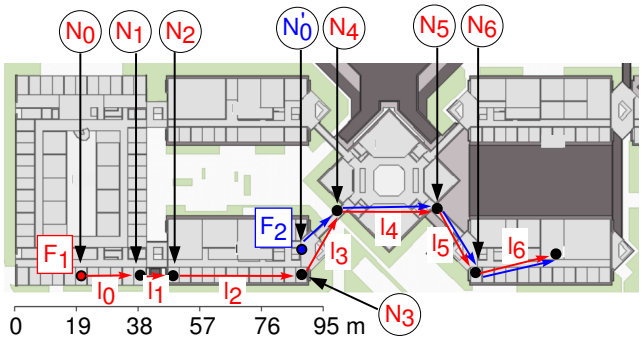


Figure 3: Illustration of the testbed topology. The hardware used in our experimental testbed are Asus WL-500gP routers with an Atheros-based wireless card.

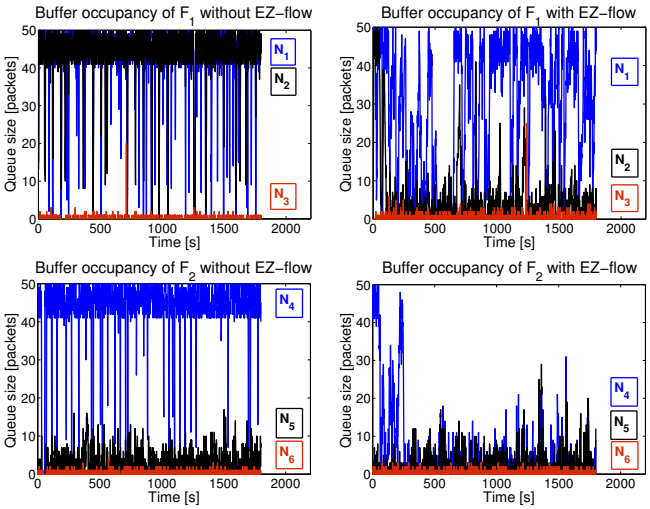


Figure 4: Experimental results for the buffer evolution of the relay nodes when flow F_1 or F_2 are active. The average number of buffered packets are: (i) without EZ-flow 41.6 **(N_1),** 43.1 **(N_2) and** 43.7 **(N_4) and (ii) with EZ-flow** 29.5 **(N_1),** 5.2 **(N_2) and** 5.3 **(N_4). The remaining buffers are very small.**

node even though they are dropped by the MAC layer (for example a buffer overflow), and thus are never really transmitted physically. 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 notice 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 that it remains unchanged for values between 2^{10} and 2^{15}.

4.2 Topology Description

We deploy our testbed over 4 buildings of the university campus where at most 2 flows are concurrently active. Figure 3 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 1. On the other hand, the flow F_2 is a shorter flow of 4 hops

	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 1: Illustration of the capacity of each link of flow F_1. The means are obtained through measurements over 1200 **s.**

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).

4.3 Measurement Results

The first scenario we consider is when F_1 is alone in the network. Figure 4 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 at the link l_2 (i.e., 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 2. Whereas, 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 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 highlight that despite this hardware limitation, EZ-flow still significantly improves the performance by reducing the turbulence of the flow and increasing the end-to-end throughput to 148 kb/s. Furthermore, we show through simulation in Section 5 that EZ-flow completely stabilizes the network once this limitation is removed.

In the second scenario, we consider F_2 alone. Similarly to [9], 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, turning EZ-flow on completely stabilizes the network for all the relay nodes (no queue builds up) by making the source node $N_0^{'}$ increase its $cw_0^{'}$ up to 2^8. Thus EZ-flow works even better in this scenario where it is not blocked by the hardware limitation and 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 [28] between a 1- and 2-hop flow, Table 2 shows that

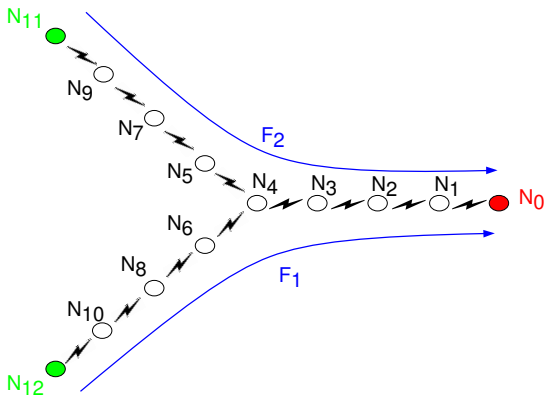


Figure 5: Scenario 1: 2-flows topology.

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 the fairness index (defined in Eq. (1)).

5. SIMULATION

We present in this section some simulation results on two different scenarios with varying traffic loads to confirm our statement that the EZ-flow mechanism successfully achieves network stability and adapts to changing traffic matrices.

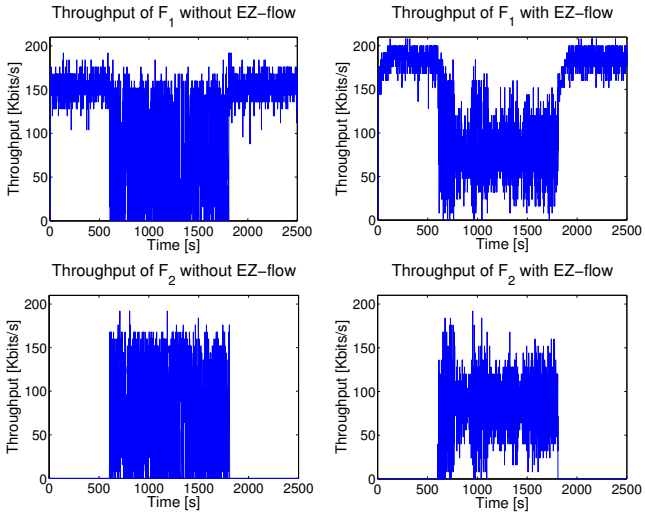
5.1 System Description

We implemented the two modules of EZ-flow, the BOE and CAA, in ns-2 simulator version 2.33 [23]. Our implementation closely follows the description of Section 3, where each node does not use any global information, but only uses the information it can hear by sniffing the channel.

Beside the inclusion of EZ-flow, we kept the standard parameters of IEEE 802.11. Therefore we use a transmission range of 250 m, a sensing range of 550 m and the RTS/CTS mechanism turned off.

	Mean throughput	Standard dev.	FI (Eq. (1))
F_1	119 kb/s	25 kb/s	
F_2	157 kb/s	29 kb/s	
F_1 F_2	7 kb/s 143 kb/s	15 kb/s 34 kb/s	0.55
F_1^{EZ}	148 kb/s	28 kb/s	
F_2^{EZ}	185 kb/s	26 kb/s	
F_1^{EZ} F_2^{EZ}	71 kb/s 110 kb/s	31 kb/s 35 kb/s	0.96

Table 2: Mean throughput, standard deviation and Jain's fairness index (FI) for measurements over 1800 s with and without EZ-flow. The sub-division in the table present the results for: (i) one single flow, or (ii) two simultaneous flows in the network.



(i) (ii)

Figure 6: Throughput statistics for the flows F_1 and F_2 in scenario 1: (i) with standard IEEE 802.11 and (ii) with EZ-flow turned on.

The reasons we do not use RTS/CTS are twofold: (i) the current implementations of the protocol disable the mechanism by default and (ii) enabling the RTS/CTS is useless in the standard case we consider where the area covered by the sensing range (550 m) is larger than the maximal area covered by RTS and CTS ($2 \cdot 250$ m). We also kept the default data rate of 1 Mb/s and the propagation model to be two-ray ground. To ensure that the systems run in saturated mode, we generate at the source a Constant Bit Rate (CBR) traffic at a rate of 2 Mb/s. Finally we use the NOAH routing agent [20], which is a static routing agent, in order to focus on the influence of the MAC layer and to remove from our study the effect of route link failure and the overhead of routing messages. The parameters of EZ-flow are $b_{min} = 0.05$, $b_{max} = 20$ and $max_{cw} = 2^{15}$.

5.2 Scenario 1: 2-Flows Topology

The topology we study in our first scenario is depicted in Figure 5 and corresponds to two 8-hop flows that merge together to access a gateway. This situation corresponds to the uplink scenario happening in the backhaul of WMNs, where different flows merge together to reach the gateway that delivers the access to the Internet.

The flow F_1 is active for the entire duration of the simulation, i.e., from 5 s to 2504 s. Flow F_2 is active between 605 s and 1804 s. The throughput and delay results are shown respectively in Figures 6 and 7.

During the first period, the flow F_1 is alone in the network ($5 - 604$ s). We note that in the case of standard IEEE 802.11 without EZ-flow, the network already suffers from congestion. Indeed, the end-to-end delay reaches a value of 4.1 s, which is unacceptable for delay-sensitive traffic, and the throughput only reaches 153.2 kb/s. But when EZ-flow is turned on, the network is stabilized. Indeed, the end-to-end delays drop at a value as low as 0.2 s. Interestingly, this reduction in delay does not happen at the cost of a reduced throughput as it increases up to an average of 183.9 kb/s, which corresponds to a throughput gain of 20% over standard IEEE 802.11. To understand why EZ-flow achieves this performance, Figure 8 shows how the contention windows are automatically adapted at the different nodes. The stable regime is reached once the relay

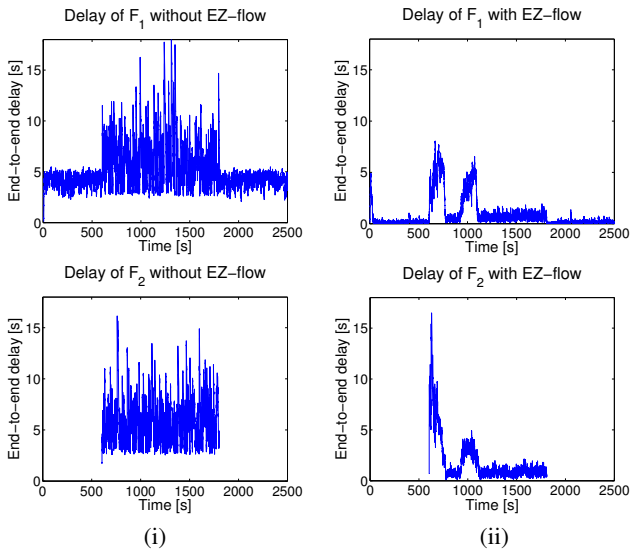


Figure 7: Delay statistics for the flows F_1 and F_2 in scenario 1: (i) with standard IEEE 802.11 and (ii) with EZ-flow turned on.

nodes set their contention window to the minimal value of 2^4 and the source node, N_{12}, sets it to $cw_{12} = 2^7$. Therefore, we highlight that for the single-flow topology, EZ-flow reaches distributively the static solution that was proven to be stable (proposed in [9]).

During the second period, both flows F_1 and F_2 are concurrently active ($605 - 1804$ s). Obviously, for IEEE 802.11 the congestion problem becomes worse with average delays as high as 5.8 s, an average throughput reduced to 76.5 kb/s and a high throughput variation. Enabling EZ-flow improves once again these three metrics, and most importantly solves the problem of congestion. Indeed, the end-to-end delay rapidly drops to negligible values, which shows no buffer builds up in the network. Furthermore, the average throughput is also increased to 82.1 kb/s.

The explanation for the two peaks in delay at around 600 s and 1000 s is found in Figure 8. The first peak corresponds to the transient incurred by the arrival of flow F_2. Up to 605 s only flow F_1 exists in the network, and EZ-flow adapted the contention windows to stabilize the network for a single-flow topology. At 605 s the second flow F_2 appears in the network and therefore the previous contention windows are too small for this new traffic load. Thus, the buffer starts to build up at some nodes and this is reflected by the sudden increase in end-to-end delay. Fortunately, EZ-flow rapidly adapts the contention windows to solve the problem and converges once again to a stable state. However, we note that after this first

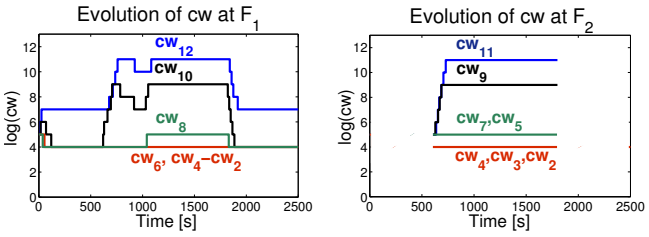


Figure 8: Illustration of how EZ-flow modifies the CW_{min} values at the different nodes of the network.

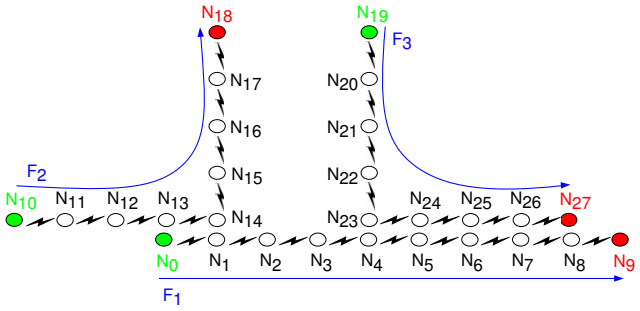


Figure 9: Scenario 2: 3-flows topology.

peak, the contention windows of the nodes in F_1 and F_2 are different as $cw_8 = 2^4$, whereas $cw_7 = 2^5$. This difference is the cause of the second peak. Indeed, due to the small cw_8, N_{10} and then N_{12} sense their successor node underutilized and thus become more aggressive. Unfortunately, this increase leads to a rate that is not supportable at the junction node N_4, and the buffers of N_5 and N_6 start to build up. Both N_7 and N_8 detect this increase, but following the algorithm of the CAA, N_8 is more likely to react as $cw_8 < cw_7$. Therefore N_8 increases its cw_8, N_{10} and N_{12} react to it and reach a steady state. Interestingly, once the stable regime is reached, the source nodes set cw_{11} and cw_{12} at the value of 2^{11}, which is once again similar to the optimal static solution proposed in [9] ($q = 2^4/2^{11} = 1/128$).

During the last period, the flow F_1 is again alone in the network ($1805 - 2504$ s). As expected, IEEE 802.11 achieves performances similar to the first period. More importantly, the results show a particularly interesting property of EZ-flow: its adaptability to changes in the traffic load. Indeed, as soon as the flow F_1 leaves the network the buffer of some nodes becomes under-utilized. EZ-flow detects this and becomes more aggressive by decreasing the cw_{12}, cw_{10} and cw_8 until it reaches the same stable state as in the first period. Therefore improvements in throughput and delay similar to the first period are found for this last period.

5.3 Scenario 2: 3-Flows Topology

The second scenario we consider is a 3-flow topology as depicted in Figure 9. This situation corresponds to the multi-hop sce-

	Mean throughput	Standard dev.	FI (Eq. (1))
F_1	145.6 kb/s	27.4 kb/s	0.75
F_2	39.9 kb/s	36.7 kb/s	
F_1	129.9 kb/s	45.3 kb/s	0.64
F_2	31.0 kb/s	32.5 kb/s	
F_3	27.3 kb/s	39.9 kb/s	
F_1	150.0 kb/s	13.0 kb/s	
F_1^{EZ}	89.9 kb/s	41.3 kb/s	1.00
F_2^{EZ}	100.3 kb/s	42.6 kb/s	
F_1^{EZ}	29.5 kb/s	22.9 kb/s	0.80
F_2^{EZ}	139.7 kb/s	23.0 kb/s	
F_3^{EZ}	135.4 kb/s	26.9 kb/s	
F_1^{EZ}	179.9 kb/s	13.5 kb/s	

Table 3: Mean throughput, standard deviation and Jain's fairness index (FI) with and without EZ-flow for the three periods: (i) F_1 alone, (ii) F_1 and F_2 active and (iii) all three flows active.

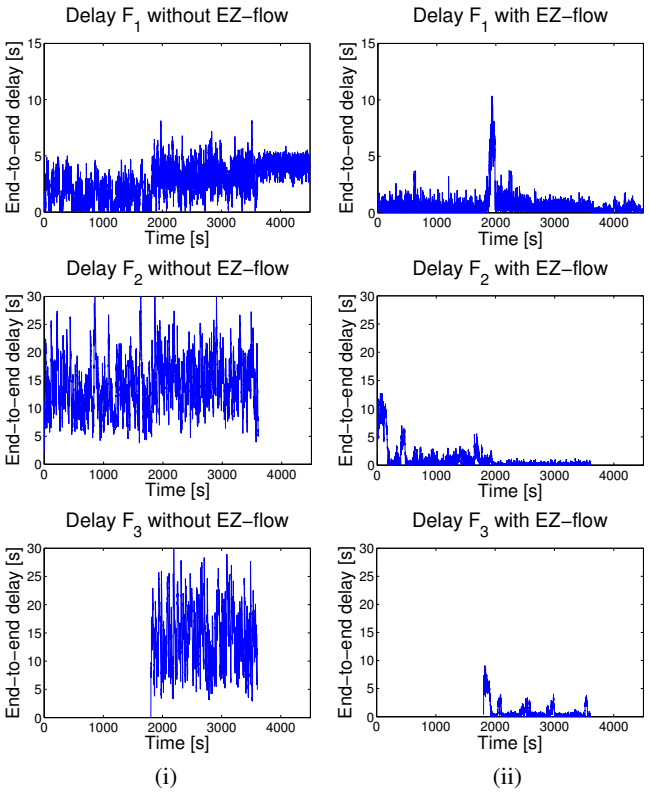


Figure 10: Delay statistics for the flows F_1, F_2 and F_3 in scenario 2: (i) with standard IEEE 802.11 and (ii) with EZ-flow turned on.

nario where multiple sources have to reach different destinations, but share the wireless resource with other flows on some parts of their paths. Furthermore, this topology illustrates what happens when the source of one flow (i.e., N_0) is a hidden node from another source (i.e., N_{10}). The simulation starts with flows F_1 and F_2 present in the network from 5 s to 1805 s. Then flow F_3 is added and the three flows share the resources from 1805 s to 3605 s. Finally, we remove flows F_2 and F_3 and let F_1 alone in the network from 3605 s to 4500 s, in order to check that the system stabilizes once again to a performance similar to what we find in the single-flow topology of scenario 1. The throughput and delay statistics are shown respectively in Figure 10 and Table 3. Furthermore, Figure 11 illustrates how EZ-flow adapts the contention windows over time.

During the first period, $[5, 1805)$, we see that IEEE 802.11 drastically suffers from the hidden node situation, with F_2 experiencing a particularly high delay ($\sim$ 15 s) and low throughput. The fairness index is of 0.75. On the contrary, when EZ-flow is turned on, the contention window of the source of F_2 cw_{10} is increased up to a value of 2^{10} to provide a smooth flow. We note that this increase delivers negligible delays to both flows and does not penalize F_2 as it has a throughput that is even slightly higher than F_1. The reason F_2 achieves a higher throughput while having a larger contention window ($cw_{10} = 2^{10}$ and $cw_0 = 2^5$) is that N_{10} only directly competes with two nodes (N_{11} and N_{12}), whereas N_0 competes with seven other nodes.

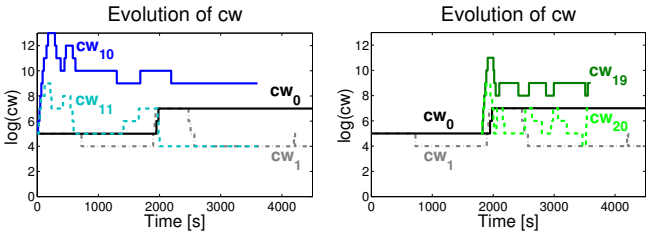


Figure 11: Illustration of how EZ-flow modifies the CW_{min} values at the two first node of each flow.

During the second period, $[1805, 3605)$, we see that IEEE 802.11 starves flow F_2 and F_3 in favor of F_1 and that all flows suffer from high delays. The reason that F_1 shows better performances than F_3 is that N_0 has many neighbors and it naturally reduces the source access rate and thus the buffer building-up problem. IEEE 802.11 achieves a cumulative throughput of 188.2 kb/s and a fairness index of 0.64. In contrast, EZ-flow increases the cumulative throughput to 304.6 kb/s (a 62% throughput gain over the standard IEEE 802.11), increases the fairness index to 0.8, and drastically reduces the end-to-end delay by an order of magnitude at least. We note that F_1 has its throughput reduced even though the source of F_1, N_0, has cw_0 that is lower than cw_{10} and cw_{19} ($cw_0 = 2^7$ and $cw_{10} = cw_{19} = 2^9$). This reduction is due to the higher competition that F_1 experiences and it allows both F_2 and F_3 to have higher throughputs and all the flows to have negligible delays and thus, a stable network.

Finally, during the last period we see that once again EZ-flow successfully detects the variation in traffic load and adapts the contention windows to achieve results similar to those in the single-flow case of scenario 1.

6. MATHEMATICAL ANALYSIS

We now formally prove the stabilization property of EZ-flow for a single flow, 4-hop network topology, which is the smallest and simplest topology that has been shown to be unstable with standard IEEE 802.11 protocols [9]. We use the same discrete-time model as in [9], which captured this instability when EZ-flow was not deployed. The result can also be extended for a general K-hop network, with $K \geq 4$.

6.1 Model Notations

We analyze a K-hop flow where the node 0 sends packets to the final destination node K with the intermediate node i forwarding the packet to node $(i+1)$. Each node i is characterized by a given contention window value cw_i and a buffer occupancy b_i. As we are interested in the stability of the queues of the relay nodes (the source being saturated), we propose a slotted-time model where a slot n corresponds to the occurrence of one transmission pattern. Furthermore, we use as state variable of the system at time n the vectors

$$\vec{b}(n) = [b_0(n)\ b_1(n)\ \ldots\ b_{K-1}(n)]^T\,,$$

$$\vec{cw}(n) = [cw_0(n)\ cw_1(n)\ \ldots\ cw_{K-1}(n)]^T\,,$$

where T denotes transposition and $b_0(n) = \infty$ due to the saturated source. We also introduce the link activation vector

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

representing the successful link activities at time slot n: $z_i(n) = 1$ if a packet is transmitted from node i to node $i+1$ during the n^{th} time slot, and $z_i(n) = 0$ otherwise. We note that to have $z_i(n) = 1$, we must have $\sum_{k\in\Lambda_i} z_k(n) = 0$, where Λ_i is the

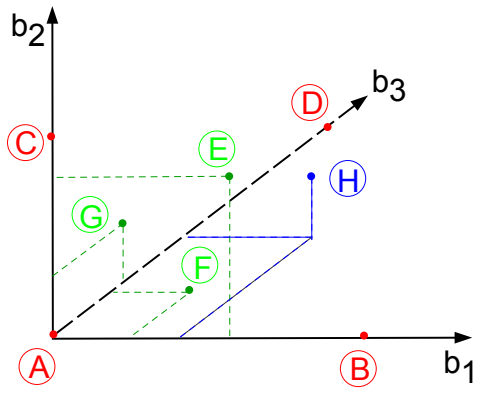


Figure 12: A 4-hop network seen as a random walk on the positive orthant of $\mathbb{Z}^3$. The eight regions A to H correspond to all the different combination of zero and nonzero entries of $\vec{b}$.

set of nodes that are in the interference range with node i. The standard interference model that is used is the 2-hop interference model that requires that all the nodes that are in the 2-hop vicinity remain silent in order for a node to transmit successfully.

6.2 EZ-Flow Dynamics

The dynamics of a network using EZ-flow is captured by the recursive equations

$$cw_i(n+1) = f(cw_i(n), b_{i+1}(n)) \quad (2)$$

$$b_i(n+1) = b_i(n) + z_{i-1}(n) - z_i(n), \quad (3)$$

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 $min_{cw} = 2^4$ and $max_{cw} = 2^{15}$ being the bounds between which the contention windows can evolve. The activation vector is computed as the result of a random function $g(\cdot,\cdot)$ as

$$\vec{z}(n) = g(\vec{b}(n), c\vec{w}(n)). \quad (4)$$

Function $g(\cdot,\cdot)$ is obtained by a set of rules, detailed in [9], and which outputs is a transmission pattern as a function of the current buffer state (only nodes with non empty buffers can transmit) and of the random sequence of backoff timers and possible collisions generated by hidden terminal effects (this makes function $g(\cdot,\cdot)$ random). This discrete-time model maps the buffer evolution of a K-hop network to a random walk on the positive orthant of $\mathbb{Z}^{K-1}$, which is divided in 2^{K-1} regions differing by the entries of $\vec{b}$ which are zero and non zero (i.e., the queues that are empty or not). Figure 12 illustrates the $2^{4-1} = 8$ regions of the $K = 4$ hop network. In each region, one can compute first the possible outcomes of the back-off timers, which depend on the contention values $c\vec{w}(n)$, and next the resulting transmission patterns which 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 follows the same reasoning as in [9]. It is summarized in Table 4 for the 4 hop network.

Region	$\vec{z}$	$\mathbb{P}(\vec{z})$
A	$[1,0,0,0]$	1
B	$[1,0,0,0]$	$\frac{cw_1}{cw_0+cw_1}$
	$[0,1,0,0]$	$\frac{cw_0}{cw_0+cw_1}$
C	$[0,0,1,0]$	1
D	$[1,0,0,1]$	1
E	$[0,1,0,0]$	$\frac{cw_0cw_2}{\sum_{i=0,1,2}\prod_{j\neq i} cw_j}$
	$[0,0,1,0]$	$1 - \frac{cw_0cw_2}{\sum_{i=0,1,2}\prod_{j\neq i} cw_j}$
F	$[0,0,0,1]$	$\frac{cw_0cw_3}{\sum_{i=0,1,3}\prod_{j\neq i} cw_j} + \frac{cw_0cw_1}{\sum_{i=0,1,3}\prod_{j\neq i} cw_j}\frac{cw_0}{cw_0+cw_1}$
	$[1,0,0,1]$	$\frac{cw_1cw_3}{\sum_{i=0,1,3}\prod_{j\neq i} cw_j} + \frac{cw_0cw_1}{\sum_{i=0,1,3}\prod_{j\neq i} cw_j}\frac{cw_1}{cw_0+cw_1}$
G	$[0,0,1,0]$	$\frac{cw_0cw_3}{\sum_{i=0,2,3}\prod_{j\neq i} cw_j} + \frac{cw_2cw_3}{\sum_{i=0,2,3}\prod_{j\neq i} cw_j}\frac{cw_3}{cw_2+cw_3}$
	$[1,0,0,1]$	$\frac{cw_0cw_2}{\sum_{i=0,2,3}\prod_{j\neq i} cw_j} + \frac{cw_2cw_3}{\sum_{i=0,2,3}\prod_{j\neq i} cw_j}\frac{cw_2}{cw_2+cw_3}$
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 4: Probability of occurrence of the transmission pattern $\vec{z}$ for the different region of the space $\mathbb{Z}^3$.

6.3 Proof of Stability

Equipped with the model described above, we now formally prove the efficiency of EZ-flow in stabilizing the network. In order to maintain the number of regions small, we detail the proof for 4 hops. A similar methodology can be used to show the stability of a K-hop network, for any given value of K by using the generalized version of the Lyapunov function used in the proof $h(\vec{b}) = \sum_{i=1}^{K-1} b_i$.

The queue dynamics of a 4-hop network is thus a random walk on $\mathbb{Z}^3$, where each dimension represents the queue size of the node i, with $i \in \{1,2,3\}$ (cfr Figure 12). We do not include the buffer occupancy of either the source b_0 that we assume to be always saturated, nor of the final destination b_4 that we assume to be always empty as the received packets are immediately removed from the buffer.

THEOREM 1. *EZ-flow stabilizes a 4-hop network by maintaining the queue size of all the relaying nodes almost surely finite.*

PROOF. As depicted in Figure 12, we divide the positive orthant of $\mathbb{Z}^3$ into 8 regions (denoted A-H), which depend on the buffer occupancy status (empty or not) of each node. The probability of each of the transmission patterns for each these 8 regions is listed in Table 4. We can then apply Foster Theorem 2 (see Appendix) with the Lyapunov function

$$h(b_1, b_2, b_3) = b_1 + b_2 + b_3,$$

and the finite set $S = \{0 \leq b_1, b_2, b_3 < B\}$, where $B > b_{max} + \log_2(max_{cw})$. We need to verify that both conditions (5) and (6) of this theorem are verified.

We note first that (5) is satisfied by the definition of h and the nonzero transition probabilities of the random walk.

It takes some more work to verify (6). One need to compute $\mathbb{E}\left[h(\vec{b}(n+1))|\vec{b}(n)\right] - h(\vec{b}(n))$ with $\vec{b}(n)$ in each of the 7 regions B-H outside S, similarly to the proof of Theorem 2 in [9]. After some computations, we find that (6) is verified with $k(\vec{b}(n)) = 1$ when $\vec{b}(n) \in F \cup H$, $k(\vec{b}(n)) = 2$ when $\vec{b}(n) \in D \cup E$, $k(\vec{b}(n)) = 3$ when $\vec{b}(n) \in G$, $k(\vec{b}(n)) = 4$ when $\vec{b}(n) \in C$ and finally $k(\vec{b}(n)) = 25$ when $\vec{b}(n) \in B$ (the latter region being handled by a computer-assisted computation). Therefore, as Region $A \subseteq S$, the conditions of Theorem 2 are satisfied in all the positive orthant of $\mathbb{Z}^3$, which proves that EZ-flow stabilizes the network. □

7. CONCLUSION

In this paper, we have 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 have 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.

We have also thoroughly evaluated the dynamic behavior of EZ-flow by using ns-2 simulation. The results show that EZ-flow quickly adapts to changing traffic loads and ensures end-to-end delays much lower than standard IEEE 802.11 WMNs.

Finally, we have derived a Lyapunov function with which we analytically prove the stability of an IEEE 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 the current IEEE 802.11e protocol, 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 [22]). 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. The extension of the Lyapunov stability analysis to more general topologies and traffic matrices remains an important area left for future work.

Acknowledgment

This work is supported in part by Deutsche Telekom Laboratories, in the framework of the Magnets project, and by the US National Science Foundation under grants CCF-0729158 and CCF-0916892.

We are very grateful to Julien Herzen for his great help in implementing the first version of EZ-flow on the Asus routers, which showed the practical feasibility of our mechanism.

Appendix

THEOREM 2 (THEOREM 2.2.4 [14], P. 30). *Let the transition probability matrix P on the state space $\mathbb{Z}^2$ be irreducible and suppose that there exists a positive function $h : \mathbb{Z}^2 \to \mathbb{R}$ such that for some finite set S, some $\epsilon > 0$ and some positive integer-valued function $k : \mathbb{Z}^2 \to \mathbb{R}$ where $\sup_{\vec{b}\in\mathbb{Z}^2} k(\vec{b}(n)) < \infty$ the following conditions hold*

$$\mathbb{E}\left[h(\vec{b}(n+1)) \mid \vec{b}(n) = \vec{i}\right] = \sum_{\vec{k}\in\mathbb{Z}^2} p_{\vec{i}\vec{k}} h(\vec{k}) < \infty \qquad (5)$$

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

$$\mathbb{E}\left[h(\vec{b}(n + k(\vec{b}(n)))|\vec{b}(n) = \vec{i}\right] \leq h(\vec{i}) - \epsilon k(\vec{b}(n)) \qquad (6)$$

for all $\vec{i} \notin S$. Then the corresponding HMC is ergodic.

8. REFERENCES

[1] *The Cloud*. http://www.thecloud.net/.

[2] *EarthLink*. http://www.earthlink.net/.

[3] *LUNAR- Lightweight Underlay Network Ad hoc Routing project*. available at 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] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. In *Proc. of SIGCOMM*, Philadelphia, PA, Aug. 2005.

[11] 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.

[12] O. Dousse. Revising buffering in CSMA/CA wireless multihop networks. In *Proc. of SECON*, San Diego, CA, June 2007.

[13] 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.

[14] G. Fayolle, V. A. Malyshev, and M. V. Menshikov. *Topics in constructive theory of countable Markov chains*. Cambridge University Press, 1995.

[15] V. Gambiroza, B. Sadeghi, and E. Knightly. End-to-end performance and fairness in multihop wireless backhaul networks. In *Proc. of ACM MobiCom*, Philadelphia, PA, Sept. 2004.

[16] 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.

[17] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. In *Proc. of INFOCOM*, Anchorage, 2007.

[18] 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.

[19] A. Jindal and K. Psounis. Achievable rate region of wireless multi-hop networks with 802.11 scheduling. *to appear in Transactions on Networking*.

[20] J.Widmer. *NO Ad-Hoc Routing Agent (NOAH)*. http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/.

[21] 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.

[22] 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.

[23] S. McCanne and S. Floyd. *ns Network Simulator*. http://www.isi.edu/nsnam/ns/l.

[24] 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. *IPCCC*, 2006.

[25] A. Proutiere, Y. Yi, and M. Chiang. Throughput of random access without message passing. In *Proc. of CISS*, Princeton, NJ, Mar. 2008.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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, Apr 2009.

[32] Y. Yi, A. Proutiere, and M. Chiang. Complexity in wireless scheduling: Impact and tradeoffs. In *Proc. of MobiHoc*, Hong Kong, China, May 2008.

[33] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *IEEE Transactions on Networking*, 15(1):133–144, 2007.

[34] L. Ying, R. Srikant, and D. Towsley. Cluster-based back-pressure routing algorithm. In *Proc. of INFOCOM*, Phoenix, AZ, Apr. 2008.