# Measurement and Modeling of the Origins of Starvation of Congestion-Controlled Flows in Wireless Mesh Networks

Omer Gurewitz, *Member, IEEE*, Vincenzo Mancuso, *Member, IEEE*, Jingpu Shi, *Member, IEEE*, and Edward W. Knightly, *Fellow, IEEE*

***Abstract*—Significant progress has been made in understanding the behavior of TCP and congestion-controlled traffic over CSMA-based multihop wireless networks. Despite these advances, however, no prior work identified severe throughput imbalances in the basic scenario of mesh networks, in which a one-hop flow contends with a two-hop flow for gateway access. In this paper, we demonstrate via real network measurements, testbed experiments, and an analytical model that starvation exists in such a scenario; i.e., the one-hop flow receives most of the bandwidth, while the two-hop flow starves. Our analytical model yields a solution consisting of a simple contention window policy that can be implemented via standard mechanisms defined in IEEE 802.11e. Despite its simplicity, we demonstrate through analysis, experiments, and simulations that the policy has a powerful effect on network-wide behavior, shifting the network's queuing points, mitigating problematic MAC and transport behavior, and ensuring that TCP flows obtain a fair share of the gateway bandwidth, irrespective of their spatial location.**

***Index Terms*—Experimental, fairness, IEEE 802.11, mesh, TCP.**

## I. Introduction

MESH deployments are expected to provide broadband low-cost mobile access to the Internet. The prevailing architecture for large-scale deployments is a multitier architecture in which an access tier connects end-user PCs and mobile devices to mesh nodes and a backhaul tier forwards traffic to and from a few high-speed gateway nodes. Different from WLANs, the mesh backhaul tier topology is *multihop*; i.e., some of the traffic traverses more than one wireless link before reaching the wired network. Clearly, for mesh networks to be successful, it is critical that the available bandwidth be distributed fairly among users, irrespective of their spatial location and regardless of their hop distance from the wired gateway.

Significant progress has been made in understanding the behavior of TCP and congestion-controlled traffic over wireless networks. Moreover, previous work showed that severe unfairness and even complete starvation can occur in multihop wireless networks using CSMA-based MAC (e.g., IEEE 802.11a/b/g MAC), and solutions have been proposed correspondingly (see Section VI for a detailed discussion of related work). However, despite these advances, no prior work has identified the basic scenario in which congestion-controlled flows contending for a shared gateway yields starvation.

In this paper, we analytically and experimentally show that starvation (i.e., long-term and severe throughput imbalance) occurs in a scenario in which two-hop flows share the same gateway with one-hop flows. Interestingly, we also show that the starvation phenomenon is not significantly affected by the number of TCP flows involved, either one-hop or two-hop flows, therefore resulting in a dramatic performance impairment of *all* two-hop flows as soon as *at least one* one-hop flow comes into play. Because the occurrence of such a combination of flows cannot be avoided in a mesh network, we refer to this fundamental scenario as the *basic scenario*. Moreover, this scenario exists with both single-radio and multiradio architectures (see the discussion in Section III). Note that starvation of two-hop flows precludes the use of the mesh architecture, which employs multihop paths by definition. Our contributions are as follows.

First, we describe the protocol origins of starvation as a compounding effect of three factors: 1) the MAC protocol induces bistability in which pairs of nodes alternate in capturing system resources; 2) despite the inherent symmetry of MAC bistability, the transport protocol induces *asymmetry* in the time spent in each state and favors the one-hop flow; and 3) most critically, the multihop flow's transmitter often incurs a high penalty in terms of loss, delay, and consequently, throughput, in order to recapture system resources.

Second, we perform experiments in the technology for all (TFA) mesh network, an operational network deployed in a densely populated urban area. We demonstrate the existence of starvation under saturation conditions and show that only a one-hop TCP flow in competition with a two-hop TCP flow is sufficient to induce starvation.

Manuscript received March 25, 2008; revised November 06, 2008; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Kumar. First published July 14, 2009; current version published December 16, 2009. This research was supported by NSF Grants CNS-0331620 and CNS-0325971 and by the Cisco Collaborative Research Initiative.

O. Gurewitz is with the Department of Communication Systems Engineering, Ben Gurion University of the Negev, Beer Sheva 84105, Israel (e-mail: gurewitz@cse.bgu.ac.il; gurewitz@gmail.com).

V. Mancuso is with the Department of Electrical, Electronic and Telecommunication Engineering, Università di Palermo, Palermo 90133, Italy (e-mail: vincenzo.mancuso@dieet.unipa.it).

J. Shi was with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA. He is now with Quantlab Financial LLC, Houston, TX 77006 USA (e-mail: jingpushi@yahoo.com).

E. W. Knightly is with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: knightly@ece.rice.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2009.2019643

Third, we develop an analytical model both to study starvation and to devise a solution to counter starvation. The model omits many intricacies of the system (TCP slow start, fading channels, channel coherence time, etc.) and instead focuses on the minimal elements needed such that starvation manifests. Namely, the model uses a discrete-time Markov chain embedded over continuous time to capture a *fixed* end-to-end congestion window, a carrier sense protocol with or without RTS/CTS, and all end-point and intermediate queues.

The model enlights *counter-starvation policy*, in which only the gateway's directly connected neighbors should increase their minimum contention window to a value significantly greater than that of other nodes. This policy can be realized via mechanisms in standard protocols such as IEEE 802.11e [2]. The model also characterizes *why* the policy is effective in that it forces all queuing to occur at the gateway's one-hop neighbors rather than elsewhere. Because these nodes have a perfect channel view of both the gateway and their neighbors that are two hops away from the gateway, bistability is eliminated such that the subsequent penalties are not incurred.

Finally, we experimentally demonstrate that the counter-starvation policy completely solves the starvation problem. In particular, we realize this policy by employing the IEEE 802.11e mechanism that allows policy-driven selection of contention windows. We redeploy a manageable set of MirrorMesh nodes on site (mirroring a subset of the TFA mesh nodes) and perform extensive experiments. We extend our investigation to a broader set of scenarios and show that the counter-starvation policy enables TCP flows to fairly share the gateway bandwidth in more general scenarios.

In the remainder, we present an experimental demonstration of starvation in Section II, an analysis of starvation's cross-layer protocol origins in Section III, an analytical model and a counter-starvation policy in Section IV, the experimental evaluation of such a policy in Section V, related work in Section VI, and a conclusion in Section VII.

## II. Starvation in Urban Mesh Networks

In this section, we describe the *basic* topology for mesh networks and experimentally demonstrate the existence of starvation in this basic topology.

### A. Basic Topology

The basic topology of any mesh network is shown in Fig. 1, in which two mesh nodes, $A$ and $B$, are located two and one hops away from the gateway, $GW$, respectively. Mesh nodes $A$ and $GW$ do not sense each other's transmission— i.e., they are hidden—and node $B$ forwards all the traffic between nodes $A$ and $GW$. In particular, we consider the case of upstream TCP traffic, in which both $A$ and $B$ transmit a TCP flow to $GW$. Since downstream traffic is expected to be at least as important as upstream traffic, we will show in Section V-D that similar results as the ones shown for upstream traffic also apply to downstream traffic, i.e., the same basic topology in which the gateway transmits two TCP flows to A and B.

Note that this topology is necessarily embedded in any larger mesh network topology given that mesh networks are defined as multihop wireless networks with gateways. In general, nodes within a two-hop distance according to the adopted routing protocol, e.g., $A$ and $GW$ in Fig. 1, can be either in transmission range or not. In this paper, we consider the relevant case in which those nodes cannot coordinate their transmissions; i.e., neither $A$ nor $GW$ defers its transmission when the other is transmitting. Throughout this paper whenever conducting measurements on the basic topology (TFA network and MirrorMesh network), we verified that nodes $A$ and $GW$ are indeed hidden. The set of experiments that we performed included the scan of wireless signals detected by both node $A$ and node $GW$ to verify that neither one of them could see the other. We also verified before and after setting each experiment that nodes $A$ and $GW$ can transmit to two different receivers simultaneously and achieve about the same throughput that can be achieved by each flow while transmitting alone; therefore, $A$ and $GW$ do not interfere with each other—i.e., they are hidden nodes.

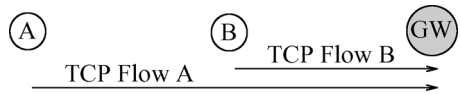


Fig. 1. The traffic matrix in the basic topology. Mesh nodes $A$ and $GW$ do not sense each other's transmission. Packet exchanges between mesh nodes $A$ and $GW$ are forwarded by mesh node $B$.

### B. Measurements in TFA

Here, we experimentally demonstrate the potential for starvation in the TFA network. TFA network is an operational mesh network that provides Internet access in a densely populated urban neighborhood in Houston, TX [1]. For each scenario experimentally examined in TFA network, we selected relevant nodes that complied with the topology studied, artificially generated the required traffic (TCP or UDP) using *Iperf v.1.7.0*, and measured the achieved throughput on each of the observed nodes. Since all experiments on TFA took place in the presence of the network's normal user traffic, and in order to minimize the interference with TFA users, we performed the experiments during off-peak hours (3:00–6:00 a.m), when TFA user traffic was negligible. Moreover, before and after each experiment, we ensured that the links under investigation were fully operational and that full throughput could be achieved when each link was used *alone*; e.g., we generated traffic only from one node and measured the end-to-end throughput achieved (*achievable TCP throughput*). Throughout this paper, we *only* show experimental results in which all participating links can reach about the same TCP (UDP) throughput when isolated. In order to further understand the channel activity throughout each experiment, we used *tcpdump v.3.4* and *Kismet v.2006.04.R1* to collect MAC-level traces at selected network nodes.

In the set of results presented in this section, the measurement intervals used are 120 s, the maximum PHY rate is 11 Mbps, and the radio band is channel 6 of the 2.4-GHz ISM band. Information regarding TFA network, including the connectivity map, and the specific nodes used for each experiment can be found in [19].

In the basic set of experiments, we chose two TFA nodes $(A, B)$, which in addition to the gateway $(GW)$, complied with the *basic topology* as described in Section II-A. As explained in

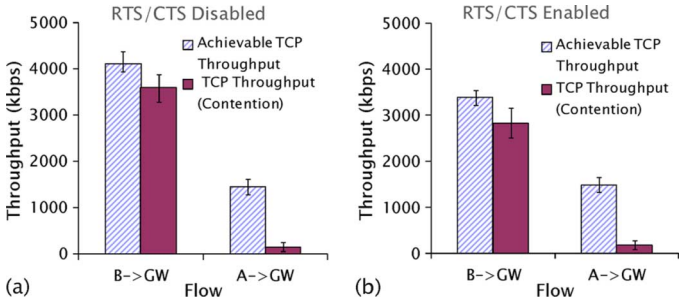


Fig. 2. TCP behavior in the basic topology, with (a) RTS/CTS disabled and with (b) RTS/CTS enabled. Each pair of bars represents the achievable TCP throughput and the TCP throughput resulting from flow contention for the one-hop flow ($B \rightarrow GW$) and the two-hop flow ($A \rightarrow GW$), respectively.

Section II-A, we experimentally verified that nodes $A$ and $GW$ were hidden from one another. Furthermore, by observing the routing table throughout the experiments, we verified that all of $A$'s packets to and from the gateway were forwarded by node $B$, and no other node was involved in the data forwarding. We simultaneously generated a TCP flow from the two-hop node $A$ and a TCP flow from the one-hop node $B$ to the gateway $GW$, and measured the TCP throughput attained by each flow.

Fig. 2 depicts the throughput of the two flows with and without contention. As can be seen in the figure, the achievable throughputs on links $A \rightarrow B$ and $B \rightarrow GW$ are about the same; hence, the two-hop flow's achievable TCP throughput is about half of the one-hop flow's one. Nonetheless, although the two-hop flow can receive considerable throughput when singly active, severe starvation occurs when the RTS/CTS mechanism is off [Fig. 2(a)] as well as when the RTS/CTS is enabled [Fig. 2(b)]. In particular, the one-hop TCP flow from node $B$ dominates, whereas the two-hop TCP flow from node $A$ receives nearly zero throughput in all experiments. Since we verify that other network activities during our experiment are negligible—i.e., we measured only a few kbps of control and data traffic—the starvation observed in Fig. 2 can be only due to the activity of nodes $A$, $B$, and $GW$, i.e., due to the high collision probability experienced by $A$'s TCP DATA and $GW$'s TCP ACKs (or by their RTS frames).

A comprehensive measurement study was conducted in TFA including diverse combinations of user activity and protocol set. Due to space limitations, those experiments are omitted from this manuscript and are fully reported in [19]. Nonetheless, the outcome of this set of experiments verifies that the basic topology starvation is neither solely due to topology and MAC behavior (hidden terminal problem), nor a straightforward consequence of the traffic matrix, but rather the compounding effect of topology, medium access mechanisms, and the behavior of a connection-oriented transport protocol, that are required to induce starvation.

## III. Starvation's Protocol Origins

Here, we describe how the protocol mechanisms of medium access and congestion control mechanisms interact to cause starvation in the basic scenario shown in Fig. 1. We analytically model this scenario in Section IV.

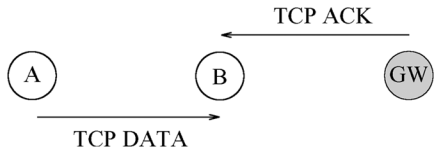


Fig. 3. TCP DATA and TCP ACK contending for channel access. Nodes $A$ and $GW$ cannot sense one another. Hence, collisions are possible at node $B$, either involving the MAC frames carrying TCP packets or the respective RTS frames, if the RTS/CTS handshake is enabled.

### A. Protocol Origins

*Medium Access and Bistability:* The collision avoidance mechanism in CSMA/CA causes bistability, in which node pairs $(A, B)$ and $(B, GW)$ alternate in transmission of multiple packet bursts. In particular, the system alternates between a state in which $A$ and $B$ jointly capture the system resources for multiple transmissions while $GW$ is idle, and a state in which $GW$ and $B$ transmit while $A$ is idle.

In order to understand the bistability, we first examine the behavior of two flows in the scenario shown in Fig. 3, where the gateway node $GW$ and two-hop node $A$ contend for transmitting TCP ACK and TCP DATA, respectively.

Assume the transmission queues of $A$ and $GW$ are backlogged at a given time, and both nodes are in the minimum contention stage. Since the two senders, namely $A$ and $GW$, are hidden from each other, a transmission from one sender succeeds only when it fits within the other sender's backoff interval. Note that when the packet size of one sender is comparable to or larger than the contention window of the other sender, the probability of collision between the two senders is very high. For example, in IEEE 802.11b with default parameters, the collision probability between two RTS transmissions from the two senders is 0.7, assuming that both transmitters are in the first backoff stage. The collision probability for data packets with RTS/CTS off is even higher (e.g., nearly 1 for packets larger than 750 bytes in 802.11b). Thus, when both nodes are in an early backoff stage, the system is likely to experience collisions. After a series of collisions, the backoff window of both nodes will become sufficiently large such that one of the nodes will successfully transmit a packet, as shown in Fig. 4(a).

Assume, without loss of generality, that node $GW$ finally succeeds in transmitting a packet. After this successful transmission, node $GW$ resets its contention window back to its minimum size, while node $A$ keeps a high contention window. In order for node $A$ to succeed in its next transmission attempt, it must fit its packet in a small backoff interval of node $GW$, which is an unlikely event. After a resulting collision, the probability to succeed for each node is asymmetric because the contention window of $GW$ is much smaller than that of $A$. This process can recur many times such that only node $GW$ manages to transmit packets, while node $A$ keeps increasing its contention window. When the contention window of $A$ is high, $GW$ can transmit multiple packets between two consecutive transmission attempts by $A$, as depicted in Fig. 4(b).

To summarize, when mesh node $GW$ ($A$) wins the channel, it enters a *success state* in which it transmits a burst of packets, while $A$ ($GW$) enters a *fail state* in which it does not succeed in transmitting any packets. The success state can terminate for

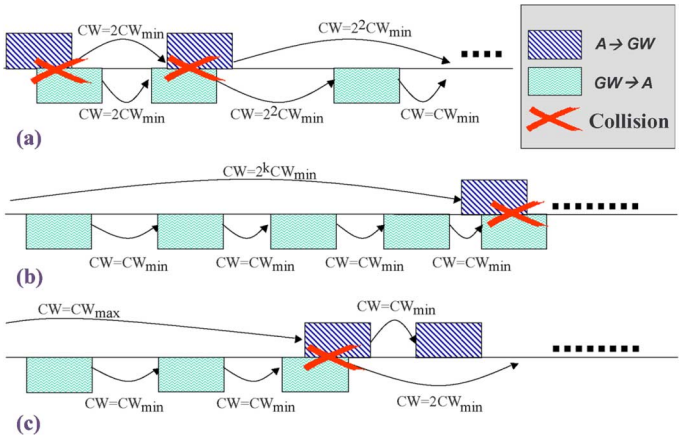


Fig. 4. Illustration of the multipacket capture of the channel by either node $A$ or $GW$. (a) Small contention window results in collision with high probability. (b) Node $GW$ succeeds to transmit a packet and resets its contention window. It may transmit multiple packets due to the high contention window of node $A$. (c) When node $A$ reaches its maximum retry limit, it still collides with high probability due to the minimum contention window of node $GW$; hence, it drops the packet and resets its contention window. Note that $GW$ increases its contention window due to the collision, which leaves high probability to node $A$ to succeed in its next transmission.

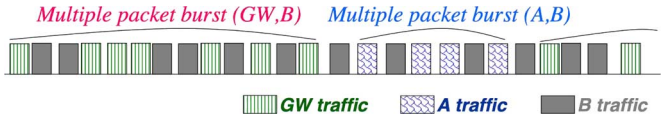


Fig. 5. Illustration of bistability with alternation of $(A, B)$ and $(B, GW)$ transmissions. Whenever $A$ $(GW)$ enters in the *success state*, a burst of packets is transmitted by $A$ $(GW)$ *and* $B$. The length of the burst depends on the value of the MAC maximum retry limit and on the backlog of the transmission queue on $A$ $(GW)$.

three reasons: 1) the probability of the node with higher contention window to win is low but not zero; 2) the losing node drops the packet and resets its contention window after it reaches its maximum retry limit, as illustrated in Fig. 4(c); 3) the transmission queue of the winning node is emptied.

Note that since node $B$ is in sensing range of both $A$ and $GW$, it contends fairly with the node that is in the success state and interleaves its packets with the burst generated from this node. This bistability is depicted in Fig. 5.

*Asymmetry Induced by Sliding Window:* TCP causes the system to spend dramatically different times in the two stable states. Specifically, TCP's sliding window mechanism creates a closed-loop system between each sender–receiver pair in which the transmission of new packets is triggered by the reception of acknowledgments. The *basic scenario* contains two nested transport loops, one for each flow. We term the one-hop and the two-hop loops as the inner loop and outer loop, respectively, as depicted in Fig. 6(a). When in the stable state reported in Fig. 6(b), in which $(A, B)$ bursts and $GW$ is in the fail state, both the outer and inner loops are broken, and hence, $(A, B)$'s burst length is upper-bounded by $A$'s TCP congestion window. Conversely, when $(B, GW)$ bursts, as in Fig. 6(c), only the outer loop is broken, and the inner loop is self-sustaining due to the loop's own ACK generation. Consequently, the duration for $GW$ and $B$ to jointly capture the channel is not bounded. As a result, the system spends much more time in the state in

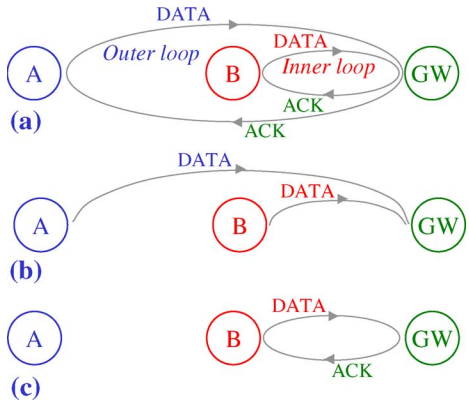


Fig. 6. Illustration of multiple control loops and a shared medium. (a) Two overlapping TCP congestion control loops are formed by TCP flows generated by $A$ (*outer loop*), and $B$ (*inner loop*). (b) When $A$ enters the *success state*, mesh nodes $A$ and $B$ can transmit TCP DATA, but they cannot receive TCP ACKs from the destination $GW$. Hence, both control loops are open. (c) When $B$ enters the *success state*, only the *outer loop* is open, and the *inner loop* is self-sustaining thanks to the TCP ACKs transmitted by node $GW$.

which $(B, GW)$ captures the channel than in the state in which $(A, B)$ captures the channel.

In order to demonstrate the asymmetry between the two states, we positioned two sniffers next to nodes $A$ and $B$. We used *Kismet* to capture all transmission attempts by the two nodes. We distinguish between transmissions initiated by transport-layer (TCP) and link-layer transmissions originated by the MAC. Accordingly, TCP transmissions include all new as well as retransmitted segments due to TCP timeout expiration, which are passed from the TCP layer to the MAC for transmission. MAC transmissions include all transmission attempts (successful and unsuccessful) by the MAC. In the following figures, each TCP segment transmission will be represented by the first MAC attempt to transmit that segment.

Fig. 7(a) shows the progress of TCP segment transmission (new and retransmitted segments) from nodes $A$ and $B$, over a 120-s experiment. The $y$-axis depicts the segment sequence number, and the $x$-axis describes the corresponding time each segment was transmitted. It can be seen in the figure that new segments from flow $B$ are continuously transmitted over time, while segments from flow $A$ are intermittently transmitted, including few long idle intervals. Fig. 7(b) depicts solely TCP retransmissions from the two nodes. As can be seen in the figure, flow $A$ suffers from frequent TCP retransmissions, while flow $B$ experiences only three TCP retransmissions within the 2-min experiment. Note that since node $B$ is within transmission range of both nodes $A$ and $GW$, all three retransmissions are due to TCP ACK-drop at $GW$'s MAC. The asymmetry between the two flows in terms of both successful as well as unsuccessful segment transmissions is clearly depicted by the two figures.

*Severe Transition Penalties:* Due to asymmetric bistable states, nodes $A$ and $GW$ experience different fail-state duration, leading to a severe penalty only for the TCP flow originating from node $A$. We described the three possible ways a node can exit its fail state. However, when $GW$ is in the fail state, node $A$'s limited burst is not likely to drive $GW$ to drop a packet. Hence, $GW$ will most likely exit its fail state by case 3); i.e., the transmission queue of $A$ is emptied. The penalty that node

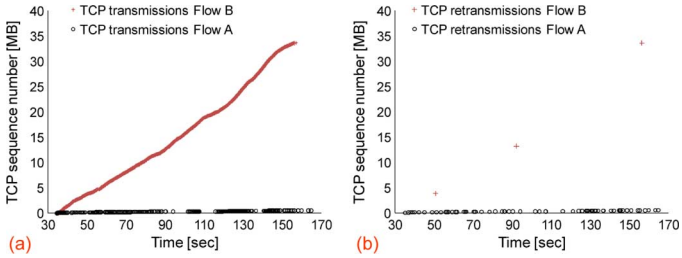


Fig. 7. TCP segment transmissions (new segment transmissions plus TCP retransmissions due to TCP timeout expiration) as captured by the sniffers next to nodes $A$ and $B$. MAC retransmissions (due to MAC timeouts expiration) are not reported in the figures. (a) All TCP segment transmissions and retransmissions. (b) Only TCP retransmissions.

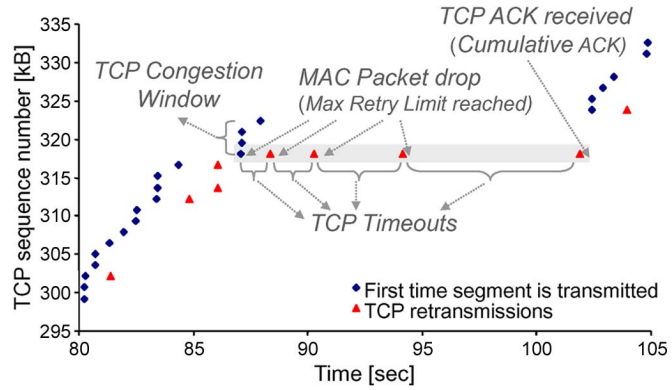


Fig. 8. A sample of node $A$'s TCP (re)transmissions as captured by the sniffer next to node $A$. The severe penalty incurred by node $A$, due to MAC packet drop, can be seen in long idle periods due to long TCP timeouts for every TCP retransmission. Note that this idle period is exponentially increased for multiple drops of the same TCP segment because TCP timeouts are doubled after each drop.

$GW$ incurs is small due to short duration of its fail state. Furthermore, this penalty is shared by both TCP flow $A$ and TCP flow $B$. On the other hand, when node $A$ is in the fail state, the inner loop is self-sustaining; hence, the gateway queue is rarely empty. Consequently, node $A$ most likely exits its fail state by case 2), i.e, by dropping the packet. The penalty node $A$ incurs is high, including both the long duration of its fail state (MAC penalty) and TCP timeout, a duration that grows exponentially with multiple drops of the same TCP segment. This penalty is only paid by TCP flow $A$.

Fig. 8 presents a sample of the TCP segment transmissions and retransmissions (excluding retransmissions initiated by MAC layer due to MAC timeouts). The severe penalty incurred by node $A$ due to MAC packet drop can be observed in the figure. For example, segment 318048 was retransmitted by the transport layer four times, which induced long TCP timeouts that resulted in long idle periods (in the order of seconds) due to small TCP congestion window.

### B. Broader Topology

A variation of the *basic topology* is shown in Fig. 9(a), where $A$ and $C$ transmit a two-hop TCP flow and a one-hop TCP flow to the gateway node $GW$, respectively. In this case, although node $C$ does not forward traffic for node $A$, the same reasoning of starvation origins applies. The gateway $GW$ and $A$ are out of

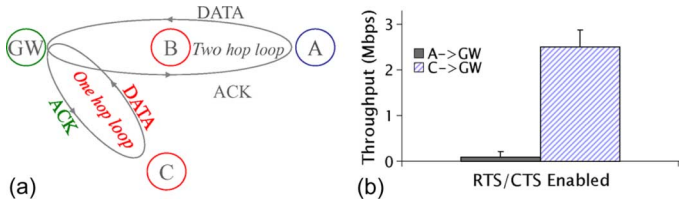


Fig. 9. Two-branch scenario and experimental TCP throughput with contending flows. (a) The scenario is composed of two branches: $A \rightarrow B \rightarrow GW$ and includes a two-hop loop, and $C \rightarrow GW$, characterized by a one-hop control loop. (b) Despite the RTS/CTS handshake mechanism, a severe TCP throughput imbalance occurs between a two-hop flow on the two-hop branch and the one-hop flow on the other branch.

carrier sense range, yielding bistable behavior. When $GW$ and $C$ obtain the channel, the one-hop loop is self-sustaining. When $A$ and $B$ obtain the channel, $GW$ is in fail state, and both loops are broken. Consequently, the burst size of $A$ is limited by its congestion window.

To verify starvation in the scenario shown in Fig. 9(a), in TFA, we select another one-hop node $C$ besides nodes $A$, $B$, and $GW$ [19]. As depicted in Fig. 9(a), two TCP flows are active on the two branches, $A \rightarrow B \rightarrow GW$ and $C \rightarrow GW$. Fig. 9(b) depicts the result of the experiment and shows that starvation does persist in this two-branch topology. As expected, the behavior of the TCP flow pair $A \rightarrow B \rightarrow GW$ and $C \rightarrow GW$ is strictly analogous to the behavior of the pair $A \rightarrow B \rightarrow GW$ and $B \rightarrow GW$ discussed above.

### C. Discussion

In mesh networks, the basic topology shown in Fig. 1 or its variation shown in Fig. 9(a) is necessarily embedded in larger scenarios such as long-chain and broad-tree topologies. In these larger scenarios, although there are other factors that affect the behavior of the contending flows, since all flows finally converge to the gateway, the embedded basic scenario plays an important role in determining the throughput of each flow. Indeed, as shown later in Section V, our extensive experiments demonstrate it in a large set of scenarios, where one-hop flows starve multihop flows.

Finally, we comment on the number of radios used in each mesh node. In our work, we consider one backhaul radio with or without a second access radio, thereby covering commercial architectures of Tropos, Cisco, Nortel, and others. Nevertheless, with multiple radios, if the number of radios is not sufficient to allocate orthogonal channels to every interfering wireless link, the results of this work are still pertinent. In fact, based on the previous subsection, whenever a two-hop transmitter is assigned the same channel with a one-hop transmitter, starvation can occur.

## IV. Analytical Model and Starvation Solution

Our proof of starvation in the basic topology is tiered. In Section II, we experimentally demonstrated the existence of TCP starvation in the basic topology. A more thorough measurement study of the starvation occurrence can be found in [19], where we experimentally isolate the originating starvation factors by eliminating alternate explanations such as background

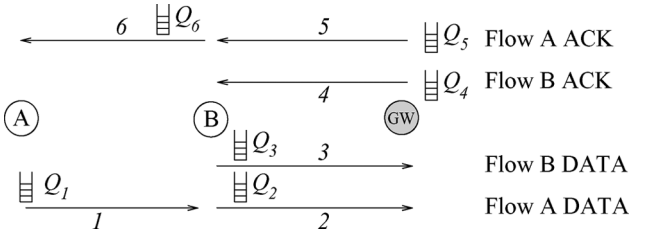


Fig. 10. Queues at different mesh nodes and links in the *basic topology*. Node $A$ only transmits TCP DATA to $B$ using queue $Q_1$ and link 1. Node $B$ maintains two different queues for the TCP DATA packets generated by $A$ (queue $Q_2$, forwarding on link 2), and by $B$ (queue $Q_3$, transmitting on link 2), plus a queue for TCP ACKs to be forwarded to node $A$ (queue $Q_6$, forwarding on link 6). Node $GW$ uses two different queues, $Q_4$ and $Q_5$, to transmit TCP ACKs on links 4 and 5 to nodes $B$ and $A$, respectively.

congestion and topology (hidden terminals with UDP traffic will not lead to starvation). In Section III, we logically explained the origins of starvation in the basic topology. In this section, we complete the validation of the starvation causes by analytically proving the compounding effects of medium access and congestion control on TCP starvation in the same scenario in which UDP has been shown to behave fairly [9]. More specifically, we employ a simplified system model that isolates the root causes of starvation under the simplest conditions in which they arise. Finally, driven by the model, we propose a counter-starvation policy based on the modification of the MAC contention window.

### A. System Model

As described in Section III, the DATA-ACK control loop in the transport layer is a key factor in starvation. Consequently, we model only one aspect of congestion control, the sliding window. In particular, we consider a *fixed* congestion control window and analytically show that the combination of CSMA MAC and transport-layer sliding window congestion control *alone* is sufficient to induce severe throughput imbalance. We show experimentally that dynamic congestion control windows with adaptive timeouts (e.g., TCP) amplify the throughput imbalance, which can result in complete starvation of some multihop flows, e.g., the two-hop flow in Fig. 1.

As for the medium access, we consider an idealized physical layer in which node pairs $(GW, B)$ and $(A, B)$ can communicate without channel errors. We do not consider physical-layer capture effect; i.e., we assume that overlapped transmissions fail. We consider the binary exponential backoff scheme, as defined in the 802.11 standard [11], and we denote by $CW_i$ the current contention window of node $i$, and by $CW_{\min,i}$ and $CW_{\max,i}$ the initial and maximum contention window for a generic node $i$, respectively. We model the transmission attempt of a backlogged node during an idle slot as a Bernoulli trial, yielding a geometric distribution of the backoff interval as used in other analytical works (e.g., [9], [15], and [18]). Consequently, at the beginning of each idle mini-slot, a backlogged node $i$ with contention window $CW_i$ attempts a new transmission with probability $e_i = 2/(CW_i - 1)$. Note that the resulting average backoff window exhibits the same average as the one defined by the 802.11 standard.

We assume that a node maintains a separate queue for each subflow; e.g., node $B$ maintains three queues: two separate queues for uplink TCP DATA originating from $A$ and locally generated by $B$ and a third queue for downlink TCP ACKs to node $A$. The resulting queuing system in the basic topology is shown in Fig. 10. We modeled a round-robin scheduling discipline without memory constraints by assuming that each time a node gains channel access, backlogged queues are served with equal probability. We will show that while this system model omits many aspects of our experimental system, it nonetheless captures the behavior of the system and predicts the severe fairness imbalance between one-hop flows and two-hop flows.

### B. Model Description

With the assumptions stated in Section IV-A, the system evolution can be modeled as a Markov chain embedded over continuous time at the mini-slot boundaries in which the channel is idle from any transmission. Note that these time epochs characterize the beginning of eight different *channel activity states*, including three *DATA transmission states* occupied by an upstream transmission on link 1, 2, or 3; three *ACK transmission states* occupied by a TCP ACK transmission on link 4, 5, or 6; one *collision state* for collisions between frames transmitted by $A$ and $GW$; and finally one *idle state* in which all the nodes count down their backoff counters and no transmission occurs. A schematic illustration of different channel activity states is given in Fig. 11, where the embedded time epochs in which we sample the system are pointed by arrows placed below the temporal axis. Note that the idle states last exactly one mini-slot as the next mini-slot defines a new state. Also, note that our model captures the behavior of CSMA with RTS/CTS handshake enabled as well as without RTS/CTS; the difference between the two will be in the duration the system spends in each state.

We label a transmission state using the index of the link on which this transmission occurs. For example, channel activity state 4 refers to transmission on link 4. We denote the duration of the transmission states, the collision state, and the idle state by $T_i (1 \le i \le 6)$, $T_c$, and $T_\delta$, respectively.

We denote the length of the transmission queue for link $i$ as $Q_i$. Let $Q_{GW} = Q_4 + Q_5$ denote the aggregate queue length at node $GW$, and let $W_A$ and $W_B$ be the *fixed* congestion windows for flows $A \rightarrow GW$ and $B \rightarrow GW$, respectively. Note that $W_A$ and $W_B$ are constant values. Because the middle node is in radio range of the two other nodes, the collision probability between the middle node and one of the other two nodes is very small, compared to the collision probability between $A$ and $GW$.[1] We therefore assume that the middle node never doubles its backoff counter; i.e., $CW_B = CW_{\min,B}$. Finally, the length of $Q_4$, $Q_5$, and $Q_6$ can be expressed with $(Q_1, Q_2, Q_3, Q_{GW}, W_A, W_B)$ as follows:

$$\begin{aligned} Q_4 &= W_B - Q_3 \\ Q_5 &= Q_{GW} - (W_B - Q_3) \\ Q_6 &= W_A + W_B - (Q_1 + Q_2 + Q_3 + Q_{GW}). \end{aligned} \tag{1}$$

[1]To collide, the middle node has to send the first packet of a data transmission within the propagation delay of one of the outer nodes, given that both nodes choose to transmit on the same mini-slot.

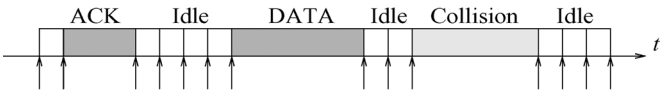


Fig. 11. Illustration of *channel activity states*. States represented by the label "DATA" include three *DATA transmission states* (transmissions on links 1,2, and 3 in Fig. 10). States represented by the label "ACK" include three *ACK transmission states* (transmissions on links 4,5, and 6 in Fig. 10). "Collision" state relates to the packet collisions between data (or RTS) frames generated by nodes $A$ and $GW$. In "Idle" state, no transmitter is active.

Therefore, we represent the system state as a six-dimensional array $\mathbf{S} = \{Q_1, Q_2, Q_3, Q_{GW}, \Omega_A, \Omega_{GW}\}$, where $\Omega_A, \Omega_{GW}$ denote the current backoff stage of nodes $A$ and $GW$, respectively.

### C. Transition Probability Computation

In order to compute the transmission probability matrix, we need to compute the probability of each possible contention outcome from each state. In the following subsection, we will only show a representative example of how to compute the transmission probability matrix. The rest of the transmission probabilities can be similarly computed.

Consider a system state $(Q_1, Q_2, Q_3, Q_{GW}, \Omega_A, \Omega_{GW})$ in which $Q_i > 0 \forall i$; i.e., each queue of Fig. 10 is backlogged. Note that this state is the most complex due to the fact that all nodes are backlogged; hence, all three nodes contend for channel access at the next state-switching time. Accordingly, each node $i \in \{A, B, GW\}$ attempts to transmit a packet with probability $e_i = 2/(CW_i - 1)$. Let $f$ denote the duration of this contending packet expressed in the number of mini-slots (RTS or data packet, depending on the handshake mechanism used). The next state is a successful packet transmission by node $A$ *iff*: 1) $A$ attempts to transmit in the next mini-slot, 2) the middle node does not attempt to transmit in the next mini-slot, and 3) the gateway $GW$ does not attempt to transmit in the next $f$ mini-slots. Thus, the successful transmission probability of the two-hop node is given by

$$e_A(1 - e_B)(1 - e_{GW})^f$$

which is the transition probability from the current state to $(Q_1 - 1, Q_2 + 1, Q_3, Q_{GW}, 0, \Omega_{GW})$.

All of the possible next states and their transition probabilities can be computed similarly and are summarized in Table I. When collision occurs, both the two-hop node and the gateway increase their backoff to the next stage, e.g., after $k$ collisions $CW_i = 2^k CW_{\min,i}$ for binary exponential backoff. If the backoff stage reaches the maximum retry limit $R_L$, it is reset to 0, which explains the modulus operator. When a node with more than one nonempty queue wins contention, these queues have equal probability to transmit their head-of-line packet, which explains the division operator.

### D. Throughput Computation

After computing all transition probabilities for matrix $\mathbf{P}$, we can numerically solve the Markov chain and obtain the stationary distribution $\mathbf{\Pi} = \mathbf{\Pi P}$, where $\mathbf{\Pi} = \{\Pi_i, 1 \le i \le H\}$ and $H$ is the total number of system states given by

TABLE I
TRANSITION PROBABILITIES WITH ALL QUEUES BACKLOGGED

| link # | to state | probability |
|---|---|---|
| link 1 | $(Q_1 - 1, Q_2 + 1, Q_3, Q_{GW}, 0, \Omega_{GW})$ | $e_A(1 - e_B)(1 - e_{GW})^f$ |
| link 2 | $(Q_1, Q_2 - 1, Q_3, Q_{GW} + 1, \Omega_A, \Omega_{GW})$ | $\frac{(1-e_A)e_B(1-e_{GW})}{3}$ |
| link 3 | $(Q_1, Q_2, Q_3 - 1, Q_{GW} + 1, \Omega_A, \Omega_{GW})$ | $\frac{(1-e_A)e_B(1-e_{GW})}{3}$ |
| link 4 | $(Q_1, Q_2, Q_3 + 1, Q_{GW} - 1, \Omega_A, 0)$ | $\frac{(1-e_A)^f(1-e_B)e_{GW}}{2}$ |
| link 5 | $(Q_1, Q_2, Q_3, Q_{GW} - 1, \Omega_A, 0)$ | $\frac{(1-e_A)^f(1-e_B)e_{GW}}{2}$ |
| link 6 | $(Q_1 + 1, Q_2, Q_3, Q_{GW}, \Omega_A, \Omega_{GW})$ | $\frac{(1-e_A)e_B(1-e_{GW})}{3}$ |
| colliding | $(Q_1, Q_2, Q_3, Q_{GW}, (\Omega_A + 1)\%R_L, (\Omega_{GW} + 1)\%R_L)$ | $(1 - e_B)[e_A + e_{GW} + -e_A e_{GW} - e_A(1 - e_{GW})^f + -e_{GW}(1 - e_A)^f]$ |
| none | $(Q_1, Q_2, Q_3, Q_{GW}, \Omega_A, \Omega_{GW})$ | otherwise |

$$H = R_L^2(W_A + 1)^2(W_B + 1)(W_A + W_B + 1). \quad (2)$$

Following the classification of channel activity states into transmission states on link $l \in \{1, 2, \ldots, 6\}$, collision state $(c)$, and idle state $(\delta)$, we now compute the binary matrices $\mathbf{\Phi_i}$ for the channel activity state $i \in \{1, 2, \ldots, 6, c, \delta\}$. These matrices have the same dimensions as the transition matrix $\mathbf{P}$ and can be computed as follows. Suppose the system makes a transition from state $m$ to state $n$, where $m$ and $n$ are indices of the system state. When making this transition, if a successful data transmission on link $l = 1, 2, \ldots, 6$ occurs, we set $[\mathbf{\Phi_l}]_{m,n} = 1$; otherwise, it is $[\mathbf{\Phi_l}]_{m,n} = 0$. Similarly, when making this transition, if a collision occurs, we set $[\mathbf{\Phi_c}]_{m,n} = 1$. If none of the nodes attempts a new transmission, we set $[\mathbf{\Phi_\delta}]_{m,n} = 1$. Let $\mathbf{M_i}$, $i \in \{1, 2, \ldots, 6, c, \delta\}$, be the $H \times H$ matrix obtained by multiplying element-by-element the matrices $\mathbf{P}$ and $\mathbf{\Phi_i}$, i.e., $[\mathbf{M_i}]_{k,l} = [\mathbf{P}]_{k,l} \cdot [\mathbf{\Phi_i}]_{k,l}$ $\forall 1 \le k, l \le H$. Note that the generic matrix element $[\mathbf{M_i}]_{k,l}$ denotes the transition probability from system state $k$ to $l$ due to an event related to the channel activity state $i$; e.g., $[\mathbf{M_c}]_{k,l}$ denotes the probability of moving from system state $k$ to $l$ due to collision $(i = c)$. The occurrence probability of each channel activity state can be computed as

$$p_{ch,i} = \sum_{k=1}^{H} [\mathbf{\Pi M_i}]_k, \qquad i \in \{1, 2, \ldots, 6, c, \delta\}. \quad (3)$$

The throughput of the two flows originating from nodes $A$ and $B$ is then expressed in packets per second as

$$\lambda_{A \to GW} = \frac{p_{ch,6} \cdot T_6}{\Delta}, \qquad \lambda_{B \to GW} = \frac{p_{ch,4} \cdot T_4}{\Delta} \quad (4)$$

in which $\Delta$ is the average duration of the channel activity states, and $T_6$ and $T_4$ are the average duration of channel activity states 6 and 4, which represent the *transmission states* for links 6 and 4, respectively. To compute the duration of the collision state, we assume that, on average, the colliding packet starts in the middle of the packet that is transmitted first.

### *E. Model Evaluation*

As previously explained, the model captures the system behavior under static sliding window congestion control mechanism. In this subsection, in order to evaluate the contribution of each factor to the starvation phenomenon, we compare the results obtained by the model with three other platforms.

First, analogously to the model, we fix the TCP congestion window in the ns-2 simulator and run the same setup. Note that even though the TCP-congestion window is fixed to the same value as the model, the simulator mimics all other TCP mechanisms, such as timeouts and cumulative ACKs, which are not included in our model. For instance, whenever a packet is dropped, the simulated system will experience a TCP timeout before retransmitting the packet. Note that this timeout grows with each packet drop. Also, note that in ns-2 the transmission attempt of each backlogged node is uniformly distributed over the current MAC contention window, according to the IEEE 802.11 standard [11], and not geometrically distributed as assumed in the model. Second, we run legacy TCP New Reno over ns-2. Note that this platform captures the complete TCP suite, including the dynamic congestion window. Third, we compare the model to measurements taken in TFA with artificial backlogged traffic injected to both nodes $A$ and $B$, where TCP New Reno adaptive congestion control is used. Note that in TFA, besides the impact of nonmodeled factors of TCP, we also evaluate MAC and PHY influences on starvation. We term the three platforms under comparison as ns-2-Fixed TCP Win., ns-2, and TFA, respectively.

The parameters used in both the model and the simulator are the default parameters of IEEE 802.11b. Because the six-dimensional Markov chain leads to a large state space as shown by (2), we numerically solve the model for $W_A = W_B = 3$; i.e., both flows are modeled as having a fixed congestion window of three packets. Accordingly, we fix the TCP congestion window in the ns-2-Fixed TCP Win. also to three.

Fig. 12 clearly shows starvation in all four platforms under investigation. The throughput of the two flows as predicted by the model is close to that obtained from the simulations and in TFA. The slight drift of throughput from flow $A \rightarrow GW$ to flow $B \rightarrow GW$ can be explained by accounting for additional parameters not included in the model or in the simulations. In fact, accounting for TCP parameters, such as adaptive TCP timeouts, when moving from the model to ns-2-Fixed TCP Win. degrades the throughput received by flow $A \rightarrow GW$ due to the higher penalty incurred by node $A$ after each packet drop. Moreover, the cumulative ACK mechanism of TCP prevents the one-hop flow to slow down as a consequence of TCP ACK losses (dropped by $GW$), while it cannot help the two-hop flow recover from TCP DATA losses (dropped by $A$), as exemplified in Fig. 8. This further magnifies the throughout imbalance between the one-hop and the two-hop flows. Allowing dynamic congestion window when moving from ns-2-Fixed TCP Win. to ns-2 further degrades the performance of flow $A \rightarrow GW$. Specifically, high packet loss induces node $A$'s flow to maintain the minimum congestion window (see Fig. 8), whereas node $B$'s flow can rapidly increase its congestion window to a high value. As a result, $B$ can push more packets than $A$ to the gateway, even

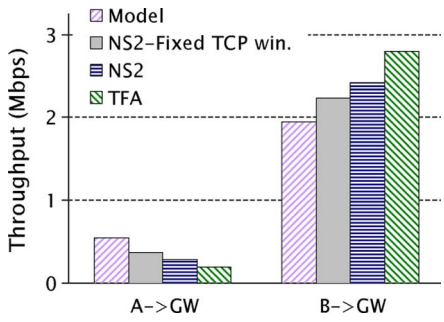


Fig. 12. Analytical model predictions compared to simulation and TFA. Unmodeled mechanisms in the transport, network, MAC, and PHY layers only aggravate the starvation problem.

when the inner control loop is broken. Finally, as can be seen in Fig. 12, in the TFA measurements, the addition of the unmodeled factors related to the MAC layer, e.g., slowing down transmissions due to EIFS and Aurotate Fallback, as well as factors related to the PHY layer, such as the fading channels, further amplify the starvation problem. In fact, since link quality and losses in $B-GW$ influence both flows, while link $A-B$ affects only flow $A \rightarrow GW$, the degradation due to channel quality or the transmission rate slowing-down due to MAC mechanisms affects flow $A \rightarrow GW$ much more than flow $B \rightarrow GW$.

### *F. Starvation Solution*

We now address improving fairness in wireless mesh networks. By reducing the backlog of nodes $A$ and $GW$, we can reduce the collisions probability between them and consequently reduce the severe penalty incurred by node $A$. Considering that $GW$'s load consists of TCP ACKs, by increasing the minimum contention window of node $B$, we reduce the rate of packets delivered to $GW$ while providing node $A$ more opportunities to forward its traffic to node $B$, consequently reducing the backlog of both nodes $A$ and $GW$.

Hence, we vary the minimum contention window $CW_{\min}$ of the middle node and evaluate its impact on throughput via both model and simulations. We observe in Fig. 13(a) that the model not only accurately predicts the throughput, but also confirms our analysis regarding node $B$'s minimum contention window. In particular, the figure shows that increasing the contention window of node $B$ has the desired effect of removing starvation and indeed providing fairness among the two flows. When the contention window is very high (e.g., 512), fairness is achieved at the unnecessary cost of throughput reduction. However, when node $B$'s minimum contention window is modestly increased to 64 or 128, fairness and high throughput are simultaneously achieved. Regardless, note that the sum of the throughput of the two flows is reduced when starvation is removed. This is necessarily the case because the two-hop flow consumes twice the resources of a one-hop flow in order to deliver the same amount of throughput. Consequently, we propose the following policy to counter starvation.

*Counter-Starvation Policy: All nodes that are directly connected to a gateway, or gateways in case of multiple gateways, should increase their minimum contention window to a value greater than that of all other nodes*. The exact window that should be selected is dependent on topology, traffic patterns, and

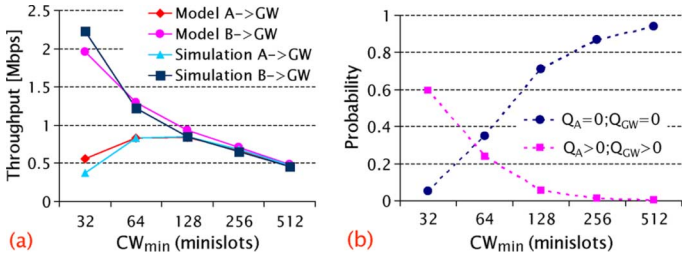


Fig. 13. System behavior versus the minimum contention window of node $B$: (a) analytical model predictions compared to simulation; (b) queue behavior.

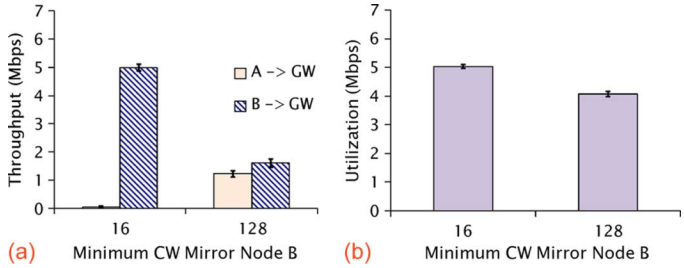


Fig. 14. RTS/CTS enabled. (a) Starvation with default $CW_{\min,B}$ and counter-starvation policy results in the basic scenario of MirrorMesh. (b) Aggregate network utilization with different values of $CW_{\min,B}$.

network objectives that can include, for example, different fairness objectives (e.g., MaxMin fairness versus proportional fairness versus combination of fairness and network utilization).

Analysis of the model's state probabilities further reveals the effect of the policy on the system queues. Fig. 13(b) shows that as the minimum contention window of the one-hop node $B$ increases, the probability that both $Q_1$ and $Q_{GW}$ are empty dramatically increases. Thus, the model indicates that the counter-starvation policy results in minimal queuing at the gateway and two-hop node for flows employing a sliding window protocol. Without these queues, the MAC protocol's bistable behavior is broken and, in turn, the "penalty to exit the fail state" is very rarely incurred.

## V. Evaluation of the Counter-Starvation Policy

In this section, we evaluate our contention window policy's ability to counter starvation. As described in Section IV, the policy sets the minimum contention window of the gateway's immediate neighbors to a value significantly larger than all other nodes. To evaluate our solution, we use an on-site deployment termed MirrorMesh.

### A. MirrorMesh Testbed

To implement our $CW_{\min}$ policy, we need to change the minimum contention window of all of the gateway's immediate neighbors. However, since this functionality is not supported in the current deployment of TFA, we deploy a few auxiliary nodes to experimentally evaluate the counter-starvation policy on the field. We refer to the platform as MirrorMesh, as we perform all experiments in the same area as TFA in order to inherit the TFA's propagation environment.

MirrorMesh nodes are desktop PCs with a Linux operating system (kernel 2.6) and Atheros wireless card (Madwifi v. 0.9.2 driver) that allows $CW_{\min}$ to be changed. Each desktop PC connects to an external omnidirectional antenna. The antennas used are the same as the ones used in TFA and are placed outdoors to emulate as closely as possible the real TFA network. Although different from the TFA nodes with respect to the wireless card and Linux kernel, they retain the behavior of network protocols such as TCP. All parameters for MAC and physical layer are according to IEEE 802.11b standard [11], except the minimum backoff window, which is set to 16 by default in Atheros chipset. MirrorMesh contains no user-generated background flows such that the only traffic is that generated by our tests.

The MadWifi driver adopts a different input direct memory access (DMA) buffer for each network device and also an additional input buffer for locally generated packets. Therefore, Atheros/Madwifi cards will classify the traffic originating at node $A$ and the traffic originating at node $B$ as two different types and will therefore operate a per-flow queuing. Furthermore, in order to concentrate on the minimal condition that can result with severe throughput imbalance and to prune any other effect that can further degrade the two-hop throughput, and since both links ($A \rightarrow B$ and $B \rightarrow GW$) can always employ the 11-Mbps physical-layer rate, we fix the transceivers to the highest modulation rate.

### B. Validation for the Basic Topology

Here, we experimentally validate our counter-starvation policy on MirrorMesh. In this set of experiments, we measure per-flow throughput and network utilization for the basic topology, both for the default $CW_{\min}$ and for increased $CW_{\min}$ as recommended by the counter-starvation policy. Each experiment lasts 120 s and the packet size is set to 1500 bytes unless stated otherwise.

We consider the scenario depicted in Fig. 1, in which nodes $A$ and $B$ both transmit packets to the gateway node, $GW$. As in TFA, we first verify that all links are operational and that $A$ and $GW$ are hidden nodes.

*RTS/CTS On:* In this experiment, we enable RTS/CTS and set $CW_{\min}$ of nodes $A$, $B$, and $GW$ to the default value of 16. Fig. 14(a) depicts a severe throughput imbalance and confirms that the system behavior for this scenario is consistent between MirrorMesh and TFA. We increase $CW_{\min}$ of node $B$ to 128 and repeat the experiment. The result is also shown in Fig. 14(a), which indicates significantly improved throughput for flow $A \rightarrow B \rightarrow GW$. In this case, $A$ and $B$ share the gateway bandwidth almost equally. Fig. 14(b) shows the aggregate utilization in which we observe that the increased $CW_{\min}$ of $B$ only leads to slightly dropped utilization. Note that when we compute the network utilization, we take into account the fact that some packets need to traverse multiple hops before reaching the gateway. For the scenario depicted in Fig. 1, we count $A$'s throughput twice because its transmitted packets need to traverse two links that cannot be active simultaneously.

The adoption of $CW_{\min} = 32$ as baseline for all nodes, as recommended by the IEEE 802.11 standard [11], leads to similar results, as shown in Fig. 15. Here, we first set $CW_{\min}$ to 32 for $A$, $B$, and $GW$ and collect the measurement results. Then, we increase $CW_{\min}$ of node $B$ to 128 and 256 and report the results for the three cases. We observe from the figure that the

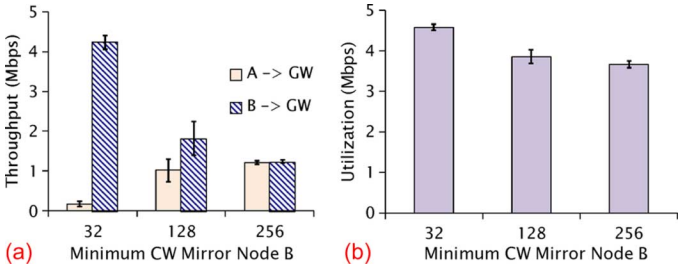


Fig. 15. Experiments in MirrorMesh with default minimum contention window set to $CW_{\min} = 32$, as for commercial devices. (a) Starvation and counter-starvation policy results in the basic topology. (b) Aggregate network utilization with different values of $CW_{\min,B}$.

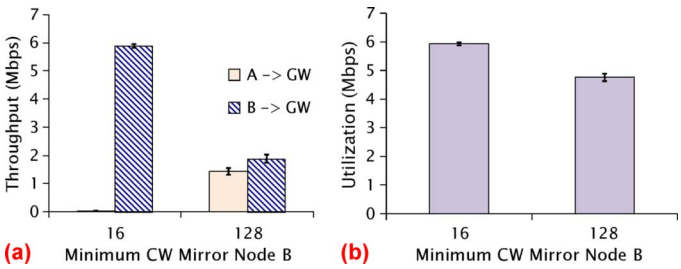


Fig. 16. RTS/CTS disabled. (a) Starvation with default $CW_{\min,B}$ and counter-starvation policy results in the basic scenario of MirrorMesh. (b) Aggregate network utilization with different values of $CW_{\min,B}$.

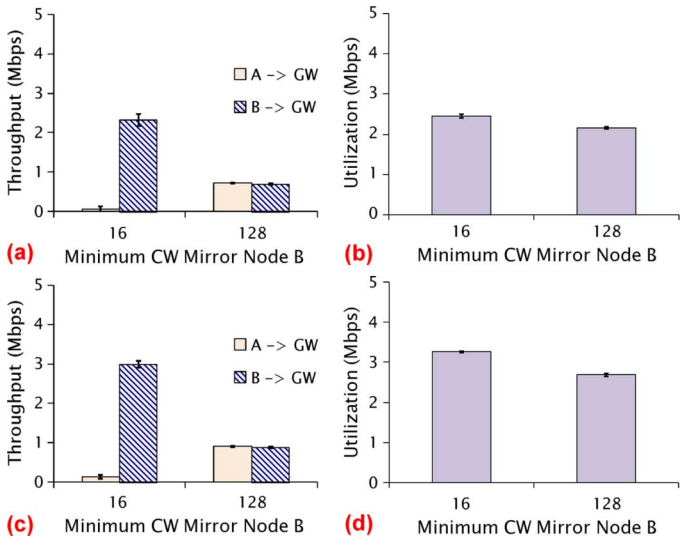


Fig. 17. Starvation and counter-starvation policy results with 500-byte TCP packets. (a) and (b) depict the results for MirrorMesh with RTS/CTS enabled, whereas (c) and (d) relate to MirrorMesh operating without RTS/CTS. Aggregate utilization is shown in (b) and (d).

nature of the starvation problem remains, yet our solution is equally effective.

*RTS/CTS Off:* Fig. 16 reports results for the case that RTS/CTS is disabled. We consider $CW_{\min} = 16$ for all nodes as well as $CW_{\min} = 128$ for node $B$. The results indicate that the counter-starvation policy is equally effective and allows equal throughput distribution among the two contending TCP flows, even without RTS/CTS. The reason is that, as discussed in Section IV, our solution results in having all queued packets at $B$. Consequently, the hidden nodes, $A$ and $GW$, are not backlogged such that the probability that both $A$ and $GW$ have packets to send simultaneously and collide is negligible, irrespective of the RTS/CTS mechanism.

*Adoption of Small Packet Size:* Because realistic traffic does not have only 1500-byte packets, we next test the impact of packet size on the starvation problem and on our solution. As shown in Fig. 17, the mere adoption of small packet sizes (500-byte packets) does not significantly affect the starvation problem. When RTS/CTS is on (upper part of the figure), contending packets are RTS control frames rather than data packets. However, a severe unfairness is experienced with the default $CW_{\min}$ configuration, which is completely eliminated by setting $CW_{\min,B}$ to 128. When RTS/CTS is disabled (lower part of Fig. 17), the system throughput is increased, but the unfairness between one-hop and two-hop flows is even more severe. Again, the adoption of $CW_{\min,B} = 128$ nearly equalizes the throughput of the two TCP flows.

Note that our analysis in Sections III and IV confirms that smaller data packets do not mitigate the starvation problem and that the counter-starvation policy is equally effective.

*Multi-TCP Streams Scenario:* Based on the explanation given in Section III, one might expect that the throughput distribution between the two mesh nodes might be more balanced if instead of a single TCP stream, each node sent multiple TCP streams. Specifically, since by sending multiple flows, the penalty paid by node $A$ is expected to be spread between the flows, the aggregate throughput of node $A$ is expected to suffer less from the causes discussed in Section III. In order to check if the throughput imbalance is indeed unique to injecting a *single* TCP stream per node, we generate various combinations of multiple TCP streams from each node $A$ and $B$, destined to $GW$. Note that generating multiple TCP streams from each mesh node emulates aggregate user traffic per node.

Thus, we repeated the basic experiment, only this time we generated multiple TCP streams from both nodes $A$ and $B$. Specifically we generated between one and ten streams from each node and checked different combinations. In Fig. 18, the labels $n$–$m$ on the horizontal axis denote the setup in which nodes $A$ and $B$ generate $n$ and $m$ TCP streams, respectively. Hence, results labeled as 1-0 and 0-1 represent the *achievable TCP throughput*, i.e., the maximum attainable TCP throughput, respectively for node $A$ and node $B$, whereas the other labels point to the contention of multiple flows generated by both $A$ and $B$. In the experiment, besides varying the number of TCP streams generated from each node, we also varied node $B$'s $CW_{\min}$; specifically, we ran the experiment for three different minimum contention windows, namely $CW_{\min,B} \in \{16, 32, 128\}$, which are the default minimum contention windows suggested by the Atheros chipset, IEEE 802.11 standard, and by our solution, respectively. Fig. 18(a)–(c) depict the results for the three different $CW_{\min,B}$ examined: 16,32,128, respectively. In the figure, for each $CW_{\min,B}$ examined, we plot both the aggregate throughput per node in the leftmost graphs (a.1, b.1, and c.1)

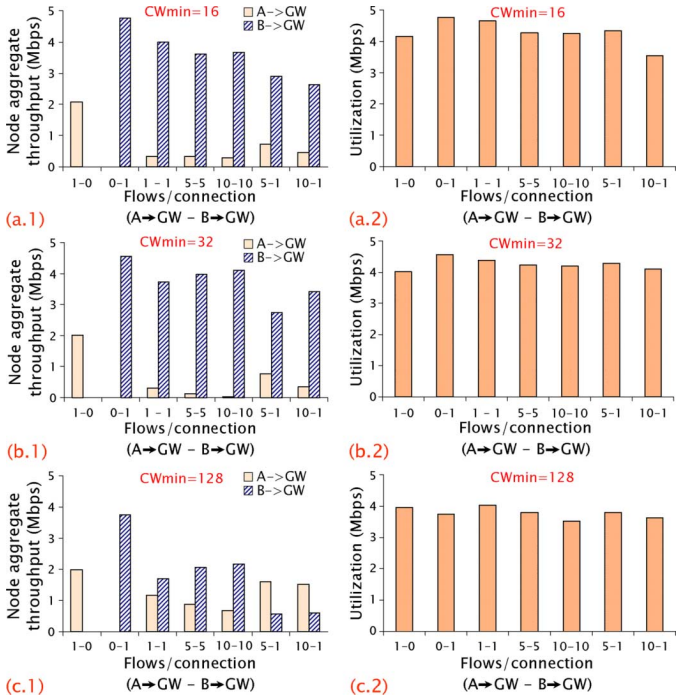


Fig. 18. Behavior of multiple contending TCP flows under different values of $CW_{\min,B}$. (a.1) and (a.2) respectively depict the per-node aggregate throughput and the network utilization when $CW_{\min,B} = 16$ (Atheros default). (b.1) and (b.2) report the case of $CW_{\min,B} = 32$ (IEEE 802.11 default). (c.1) and (c.2) show the effects of the counter-starvation policy, i.e., $CW_{\min,B} = 128$.

and the utilization, which is reported in the rightmost graphs (a.2, b.2, and c.2).

Fig. 18(a) and (b) clearly depict that under the default $CW_{\min}$ (Atheros chipset and IEEE 802.11 standard, respectively) deployed by node $B$, starvation persists regardless of the number of TCP streams generated by each node. Note that even though the asymmetric setups 5-1 and 10-1 slightly improve the throughput achieved by node $A$, the trend in which node $B$ receives most of the available throughput is kept; i.e., the throughput achieved by the single flow sent by node $B$ is much higher than the aggregate throughput attained by all the five or ten streams generated by node $A$. Fig. 18(c) shows that our suggested solution, which assigns node $B$ a $CW_{\min} = 128$, dramatically improves the throughput attained by node $A$ also for the multiflow setup [Fig. 18(c.1)], while hardly affecting the utilization [Fig. 18(c.2)]. Furthermore, Fig. 18(c.1) reveals that in the asymmetric scenarios, the aggregate throughput of node $A$ is much higher than $B$'s throughput, allowing a much fairer sharing of bandwidth on a per-flow basis rather than merely on a per-node basis.

The reason why even the multiflow setup suffers from severe throughput imbalance is twofold. First, note that as far as the aggregate throughput attained by node $A$ is concerned, there is gain by spreading the penalty between all the streams sent by node $A$ [e.g., Fig. 18(a.1)]; however, this gain only affects the transport layer and not the MAC layer. In fact, TCP parameters such as congestion window and TCP timeouts are individual per flow and will not directly affect other TCP flows. In contrast, MAC penalties related to long delays due to high backoff stages will affect all the flows passing through node $A$ regardless

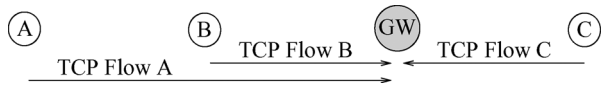


Fig. 19. Two-branch topology. TCP flow $A$ is a two-hop flow between hidden nodes $A$ and $GW$. TCP flows $B$ and $C$ are one-hop flows.

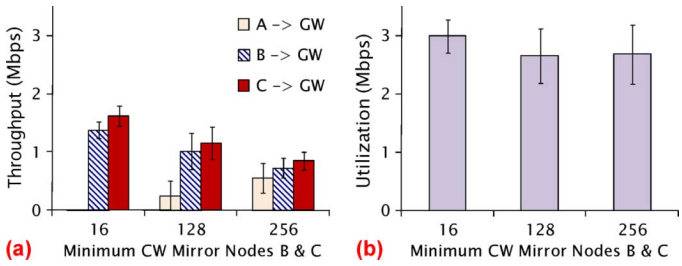


Fig. 20. Results for the two-branch topology. (a) Starvation and counter-starvation policy in MirrorMesh with default $CW_{\min,A} = 16$, and various $CW_{\min}$ for nodes $B$ and $C$. (b) Aggregate network utilization with different values of $CW_{\min,B} = CW_{\min,C}$.

of the stream currently being served. Second, even concerning the transport layer penalty, it is important to note that there is a tradeoff between the gain and the loss due to the multiflow setup. In particular, it is true that each stream will have to pay the penalty less frequently in order to gain channel access; however, due to the multiple flows generated by node $A$, the values of TCP parameters like RTT and timeouts are high even without contention and penalty due to packet drops. Hence, the penalty is much higher than for the case of single flow—e.g., each packet drop doubles a timeout that is high anyway and decreases a congestion window that is small in the first place.

### C. Extending the Basic Topology: Two Branches

Next, we use MirrorMesh to evaluate the counter-starvation policy in the two-branch topology as discussed in Section III. The scenario of this experiment is shown in Fig. 19, in which three flows are active on two branches. Results in Fig. 20 confirm that the two-hop flow from $A$ is starved, whereas one-hop flows from $B$ and $C$ almost equally share the bandwidth. We then invoke the counter-starvation policy by increasing $CW_{\min}$ for both of the gateway's one-hop neighbors, $B$ and $C$. As shown in Fig. 20, our solution dramatically improves the two-hop flow throughput. This is because increasing the $CW_{\min}$ of all one-hop nodes slows down all the first-hop nodes, giving more opportunities for node $A$ to forward its traffic to node $B$ with less interference from the ACKs sent by the $GW$.

### D. Downstream Traffic

So far, we have considered upstream data traffic. Here, we experimentally show the presence of the starvation problem and the effectiveness of our solution also for downstream flows. We reverse the direction of the flows in Fig. 1 such that DATA packets are transmitted from $GW$ to nodes $A$ and $B$ and TCP ACKs are transmitted from $A$ and $B$ to $GW$ (Fig. 20). Note that the compounding effect described in Section III-A, due to the MAC layer hidden terminals and the transport-layer control loops enforced by the sliding window congestion control, remain the same as in the uploading scenario despite the direction reversal.

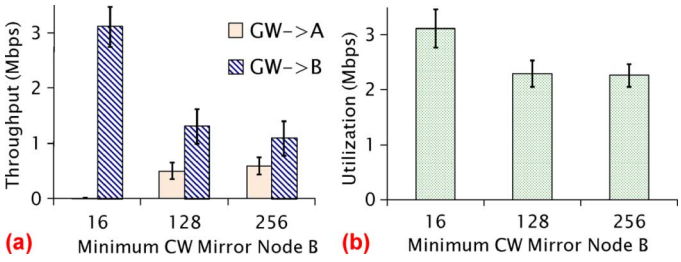


Fig. 21. Downstream flows in the *basic topology*. (a) Starvation and counter-starvation policy results in MirrorMesh. (b) Aggregate network utilization with different values of $CW_{\min,B}$.

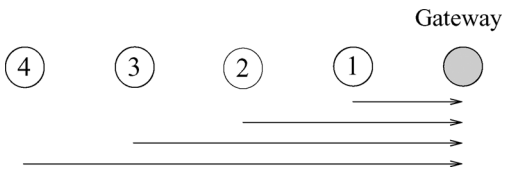


Fig. 22. TCP flows in four-hop chain topology.

Fig. 21 shows the throughput $A$ and $B$ receive when $CW_{\min}$ is 16 for all three nodes as well as when $CW_{\min,B}$ is set to 128 and 256. As anticipated, starvation indeed occurs in the download scenario, and enlarging the one-hop contention window allows the two-hop downstream TCP flow to receive significantly higher throughput than with the default window. Interestingly, even though the trend is the same as with the upstream results, the aggregate network utilization is affected (degraded) noticeably more than in the upstream case.

### *E. Larger Scenarios via Simulation*

We use the *ns-2* simulator to validate our solution in more general scenarios. We begin our simulations with a longer chain topology where spatial reuse is present. We then perform simulations on a topology in which three branches are connected to the gateway, with each branch further diverging. In all simulations, we use TCP New Reno for congestion control and IEEE 802.11b for medium access control. We use the default ns-2 MAC and PHY layer parameters, fixing the transmission rate to 11 Mbps.

*Four-Hop Chain Topology:* In this scenario, four mesh nodes are connected to a gateway in a chain topology. Each node $i = 1,2,3,4$ generates a long-lived TCP flow to the gateway, as depicted in Fig. 22. Each node can directly transmit to the node(s) located immediately on its right or left in the figure. Following the ns-2 terminology, each node is in *transmission range* with its immediate neighbor and out of *carrier sense range* with all other nodes. Hence, spatial reuse is possible in the topology depicted in Fig. 22; e.g., nodes 1 and 4 can transmit simultaneously to the gateway and to node 3, respectively. In these simulations, we explore whether this factor changes the nature of the starvation problem and the effectiveness of the proposed counter-starvation policy.

Fig. 23 depicts the simulation results for two cases: when all nodes have $CW_{\min} = 32$ and when $CW_{\min,1} = 128$—i.e., the minimum congestion window of the gateway's one-hop neighbor is increased to 128 following the counter-starvation

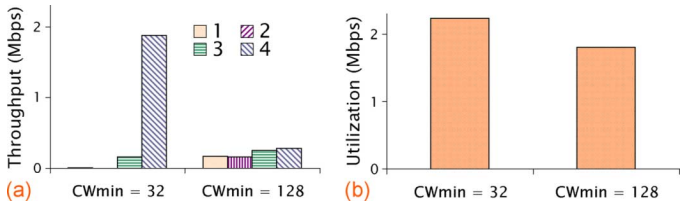


Fig. 23. ns-2 Simulation in a four-hop chain topology. (a) TCP throughput with $CW_{\min} = 32$ for all nodes and with the one-hop changed to 128. (b) Aggregate network utilization with different values of $CW_{\min,1}$.

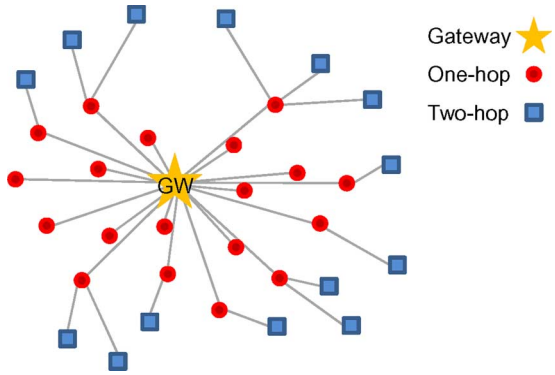


Fig. 24. A dense random topology used for ns-2 simulations. A single gateway GW is connected to 19 one-hop nodes. In turn, the one-hop nodes offer connectivity to 14 randomly located two-hop nodes. All nodes but GW are establishing fully backlogged TCP connections to the gateway.

policy. We observe that with all nodes having the same minimum contention window, the one-hop node receives an order of magnitude larger throughput than the sum of the throughput received by all other nodes. In contrast, by changing $CW_{\min}$ for the gateway's neighbor to 128, all mesh nodes receive similar throughput.

Note that in a longer chain topology, though spatial reuse is possible, nodes farther away from the gateway have less forwarding responsibility and are more lightly loaded. In contrast, nodes that are one and two hops away from the gateway still share the medium with all flows and, consequently, are the bottleneck. Thus, the starvation problem in a longer chain has the same nature as in the two-hop chain topology, and our solution is just as effective in eliminating starvation.

*Dense Topology:* Finally, we consider a scenario in which multiple nodes on several branches are connected to the gateway such as depicted in Fig. 24. The key issue is whether in a dense network the one-hop nodes could be silenced by the transmissions on other branches, thus leaving more transmission opportunities to their successor nodes, i.e., the multihop nodes, and subsequently eliminating starvation *without* requiring the counter-starvation policy.

We test the scenario in Fig. 24, in which each mesh node is simultaneously transmitting TCP traffic to the gateway. Fig. 25(a) and (b) show the cumulative distribution function (cdf) of the throughput obtained by the one- and two-hop nodes, respectively. The figure depicts the throughput CDF results for different $CW_{\min}$ assigned to all the one-hop nodes.

As expected, based on the explanation given in Section III-B, using the default contention window ($CW_{\min} = 32$ for all

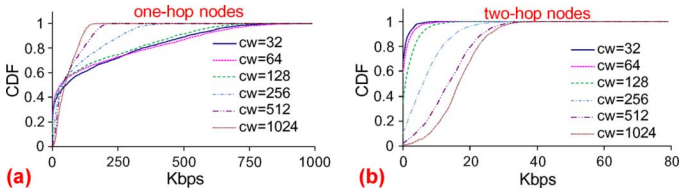


Fig. 25. Cumulative distribution function of (a) one-hop nodes and (b) two-hop nodes, i.e., the fraction of nodes that receives not more than $x$ kbps throughput in the network topology depicted in Fig. 24.

nodes), starvation persists in this scenario; i.e., Fig. 25(b) shows that no two-hop node attains more than 12 kbps and more than 90% of the two-hop nodes attains less than 2 kbps.

Based on our counter-starvation policy, we increase the minimum backoff window $CW_{\min}$ of all one-hop nodes. We observe in Fig. 25(b) that with this policy, the bigger the $CW_{\min}$ assigned to all the one-hop nodes, the higher the throughput attained by the two-hop traffic. Hence, for example, with $CW_{\min}$ equal to 128 and 256, 2% and 20% of the nodes respectively receive more than 12 kbps, which is more than what any node received with $CW_{\min} = 32$. Increasing the $CW_{\min}$ to all the one-hop nodes to 512 will allow 57% of the nodes to receive more than 12 kbps. The average throughput attained by all two-hop nodes is 0.67, 2.25, 7.52, and 14.26 kbps for $CW_{\min} = 32$, 128, 256, and 512, respectively. This gain, which is more than 300%, 1100%, and 2100%, respectively, was obtained at the expense of the one-hop nodes, which on average attained 164, 144, 90, and 63 kbps for $CW_{\min} = 32$, 128, 256, and 512, respectively. Nonetheless, even though the two-hop flows gained hundreds percent more throughput than the default flow, due to the limited capacity of a single GW to support many nodes, the effectiveness of the solution is limited—i.e., the bandwidth of the two-hop flows is still relatively low.

## VI. Related Work

*One-Hop Flows:* We refer to a flow as a one-hop flow if the source of the flow can reach its destination within one hop. One-hop flows can exist in both one-hop topologies, in which all nodes sense each other's transmission, and multihop topologies, in which they do not. One-hop flow studies showed both analytically as well as by simulation that in a fully backlogged scenario without flow control mechanisms (e.g., UDP traffic), network resources can be shared unevenly between contending flows. It was shown that MAC mechanisms ranging from binary exponential backoff to the use of carrier sense itself can cause unfairness [3], [4], [8], [9]. Moreover, MAC-level solutions to unfairness among one-hop flows have been previously proposed, including suggested modifications to exponential backoff [4], [22] and the handshake mechanism [4]. Likewise, in the context of 802.11e (which addresses QoS and service differentiation), some proposals allow different system parameters (Contention Window, $SIFS$ and $DIFS$, etc.) for different traffic classes [16], [17], [20], thereby achieving performance differentiation.

In contrast, we consider *multihop* flows, which yield a significant difference from one-hop flows. For example, the memory introduced due to receipt and subsequent forwarding of the same packet adds multiple dimensions to the modeling problem as we describe in Section IV.

*Multihop Flows:* Poor performance of multihop TCP flows has been previously established [10], [21]. Furthermore, severe unfairness has been observed when multiple TCP flows compete for the same wireless medium [12], [23]–[25].

To improve performance of congestion control in multihop wireless networks, proposals include hop-by-hop distributed congestion control [26] and joint redesign of congestion control and medium access [5], [7]. Transport-level counter-starvation policies have also been proposed in which the TCP protocol is modified by adaptively slowing down the transmission rate [6], [13], limiting the TCP transmission window [6], [21] or modifying RED [6], [25]. Finally, a simplified model of IEEE 802.11 MAC and TCP features for multihop flows can be found in [14], where a single TCP flow is modeled over a two-hop chain assuming that the TCP transmission window is fixed and neglecting MAC collisions (and hence, neglecting binary exponential backoff issues).

Differently from prior work, this paper shows that it is the sliding window congestion control and IEEE 802.11 MAC that jointly induce unfairness. Even with the TCP window fixed to its optimal value suggested in [6], TCP can still perform poorly and lead to unfairness. In addition, neither of these prior works identified nor modeled starvation in the basic topology discussed here, which is the minimum and fundamental topology that inherently exists in mesh networks. Moreover, our counter-starvation policy only modifies basic MAC protocol parameters and does not require any transport, network, or MAC protocol modifications, nor does it necessitate any control message exchange.

## VII. Conclusion

In this paper, we show that the interaction of one-hop TCP flows with two-hop TCP flows is sufficient to induce starvation. We measure starvation in an operational multitier urban mesh network and describe how the starvation's originating factors stem from interaction between the transport layer's congestion control and the MAC layer's collision avoidance. We analytically model the system and utilize the model to devise a simple counter-starvation policy in which nodes one hop away from the gateway increase their minimum contention window. We finally implement and empirically validate the solution not only via simulation, but also on MirrorMesh, a network redeployment within the same urban environment.

## References

[1] "TFA-Wireless," [Online]. Available: http://www.techforall.org/tfa_wireless.html
[2] "Wireless LAN medium access control (MAC) enhancements for quality of service (QoS)," IEEE 802.11e draft 8.1, May 2005.
[3] G. Anastasi, E. Borgia, M. Conti, and E. Gregori, "IEEE 802.11b ad hoc networks: Performance measurements," *J. Cluster Comput.*, vol. 8, no. 2–3, pp. 135–145, Jul. 2005.
[4] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," in *Proc. ACM SIGCOMM*, London, U.K., 1994, pp. 212–225.
[5] L. Chen, S. H. Low, and J. C. Doyle, "Joint congestion control and media access control design for ad hoc wireless networks," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, pp. 2212–2222.
[6] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 2003, vol. 3, pp. 1744–1753.

[7] V. Gambiroza, B. Sadeghi, and E. Knightly, "End-to-end performance and fairness in multihop wireless backhaul networks," in *Proc. ACM MobiCom*, Philadelphia, PA, 2004, pp. 287–301.
[8] M. Garetto, T. Salonidis, and E. Knightly, "Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks," presented at the IEEE INFOCOM, Barcelona, Spain, Apr. 2006.
[9] M. Garetto, J. Shi, and E. Knightly, "Modeling media access in embedded two-flow topologies of multi-hop wireless networks," in *Proc. ACM MobiCom*, Cologne, Germany, Aug. 2005, pp. 200–214.
[10] M. Gerla, K. Tang, and R. Bagrodia, "TCP performance in wireless multi-hop networks," in *Proc. WMCSA*, New Orleans, LA, Feb. 1999, pp. 41–50.
[11] *IEEE 802.11-2007: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Std. 802.11-2007 edition, IEEE 802.11 Working Group.
[12] H. Y. Hsieh and R. Sivakumar, "IEEE 802.11 over multi-hop wireless networks: Problems and new perspectives," in *Proc. VTC (Fall)*, Vancouver, BC, Canada, 2002, vol. 2, pp. 748–752.
[13] T. Jimenez and E. Altman, "Novel delayed ACK techniques for improving TCP performance in multihop wireless networks," in *Proc. PWC*, Venice, Italy, 2003, pp. 237–250.
[14] A. Kherani and R. Shorey, "Throughput analysis of TCP in multi-hop wireless networks with IEEE 802.11 MAC," in *Proc. IEEE WCNC*, Atlanta, GA, Mar. 2004, pp. 237–242.
[15] A. Kumar, E. Altman, D. Miorandi, and M. Goyal, "New insights from a fixed point analysis of single cell IEEE 802.11 WLANs," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, pp. 1550–1561.
[16] A. Nafaa, A. Ksentini, A. Mehaoua, B. Ishibashi, Y. Iraqi, and R. Boutaba, "Sliding contention window (SCW): Towards backoff range-based service differentiation over IEEE 802.11 wireless LAN networks," *IEEE Netw.*, vol. 19, no. 4, pp. 45–51, 2005.
[17] L. Romdhani, Q. Ni, and T. Turletti, "Adaptive EDCF: Enhanced service differentiation for IEEE 802.11 wireless ad hoc networks," in *Proc. IEEE WCNC*, New Orleans, LA, 2003, pp. 1373–1378.
[18] G. Sharma, A. Ganesh, and P. Key, "Performance analysis of contention based medium access control protocols," presented at the IEEE INFOCOM, Barcelona, Spain, Apr. 2006.
[19] J. Shi, O. Gurewitz, V. Mancuso, J. Camp, and E. W. Knightly, "Starvation in operational urban mesh networks: Compounding effects of congestion control and medium access," Rice Univ., Tech. Rep. TREE0709, Jun. 2007 [Online]. Available: http://www.ece.rice.edu/~og3888/Rice-TREE0709.pdf
[20] V. Siris and G. Stamatakis, "Optimal $CW_{\min}$ selection for achieving proportional fairness in multi-rate 802.11e WLANs: Test-bed implementation and evaluation," in *Proc. ACM WiNTECH*, Los Angeles, CA, 2006, pp. 41–48.
[21] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar, "ATP: A reliable transport protocol for ad-hoc networks," in *Proc. ACM MobiHoc*, Annapolis, MD, Jun. 2003, pp. 64–75.
[22] Y. Wang and J. J. Garcia-Luna-Aceves, "Channel sharing of competing flows in ad hoc networks," in *Proc. IEEE ISCC*, Kemer-Antalya, Turkey, Jun. 2003, pp. 189–196.
[23] K. Tang and M. Gerla, "Fair sharing of MAC under TCP in wireless ad hoc networks," in *Proc. IEEE MMT*, Venice, Italy, Oct. 1999, vol. 4, pp. 231–240.
[24] K. Xu, S. Bae, S. Lee, and M. Gerla, "TCP behavior across multihop wireless networks and the wired Internet," in *Proc. WOWMOM*, Atlanta, GA, Sep. 2002, pp. 41–48.
[25] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED," in *Proc. ACM MobiCom*, San Diego, CA, 2003, pp. 16–28.
[26] Y. Yi and S. Shakkottai, "Hop-by-hop congestion control over a wireless multi-hop network," in *Proc. IEEE INFOCOM*, Hong Kong, China, Mar. 2004, pp. 2548–2558.

**Omer Gurewitz** (S'00–M'05) received the B.Sc. degree in physics from Ben Gurion University, Beer Sheva, Israel, in 1991, and the M.Sc. and Ph.D. degrees from the Technion—Israel Institute of Technology, Haifa, in 2000 and 2005, respectively.

He is an Assistant Professor in the Department of Communication Systems Engineering, Ben-Gurion University. Between 2005 and 2007, he was a Post-Doctoral Researcher at the ECE Department, Rice University, Houston, TX. His research interests are in the field of performance evaluation of wired and wireless communication networks. His current projects include cross-layer design and implementation of medium access protocols for 802.11, as well as 802.16 (WiMax) standards.

**Vincenzo Mancuso** (S'02–M'06) received the Laurea degree in electronics and the Ph.D. in telecommunications from the University of Palermo, Palermo, Italy, in 2001 and 2005, respectively.

He is a Post-Doctoral Researcher at the DIEET, University of Palermo. He received a scholarship for his Ph.D. from the MIUR Ministery, Italy, a grant from the University of Roma "Tor Vergata" in the frame of the SATNEX network of excellence, and his position at the University of Palermo is currently funded by the MIUR. His research activities involve QoS support in access networks, QoS support for multimedia applications over the Internet, vehicular area networks, hybrid satellite and terrestrial networks, wireless mesh networks, and mesh deployments. He has participated and is currently participating in several projects funded by the European community and by the Italian government.

**Jingpu Shi** (S'96–M'07) received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, in 1996; the M.S. degree from The Ohio State University, Columbus, in 2004; and the Ph.D. degree from Rice University, Houston, TX, in 2007, all in electrical and computer engineering.

He is a Quantitative Analyst at Quantlab Financial LLC, Houston, TX. From 1996 to 2001, he was a Researcher at the Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China. His research interests are in the areas of wireless networks, high-performance MAC protocol design, and quantitative finance.

**Edward W. Knightly** (S'91–M'96–SM'04–F'09) received the B.S. degree from Auburn University, Auburn, AL, in 1991, and the M.S. and Ph.D. degrees from the University of California at Berkeley in 1992 and 1996, respectively.

He is a Professor of Electrical and Computer Engineering at Rice University, Houston, TX. His research interests are in the areas of mobile and wireless networks and high-performance and denial-of-service resilient protocol design.

Prof. Knightly is a Sloan Fellow and a recipient of National Science Foundation CAREER Award. He received the best paper award from ACM MobiCom 2008. He served as Technical Co-Chair of IEEE INFOCOM 2005, General Chair of ACM MobiHoc 2009 and ACM MobiSys 2007, and served on the program committee for numerous networking conferences, including ICNP, INFOCOM, MobiCom, and SIGMETRICS. He served as Associate Editor for numerous journals and special issues, including the IEEE/ACM TRANSACTIONS ON NETWORKING and the IEEE JOURNAL ON SELECTED AREAS OF COMMUNICATIONS Special Issue on Multi-Hop Wireless Mesh Networks.