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

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.

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

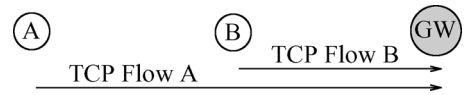

Fig. 1. The traffic matrix in the basic topology. Mesh nodes A and GW do not sense each other's transmission. Packet exchanges between mesh nodes A and GW are forwarded by mesh node B.

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.

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

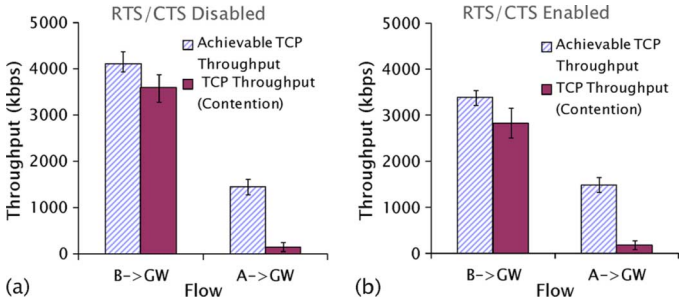

Fig. 2. TCP behavior in the basic topology, with (a) RTS/CTS disabled and with (b) RTS/CTS enabled. Each pair of bars represents the achievable TCP throughput and the TCP throughput resulting from flow contention for the one-hop flow ($B \rightarrow GW$) and the two-hop flow ($A \rightarrow GW$), respectively.

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 star-

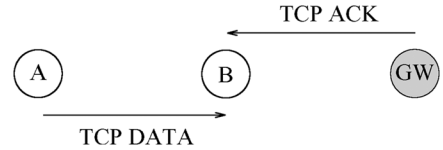

Fig. 3. TCP DATA and TCP ACK contending for channel access. Nodes A and GW cannot sense one another. Hence, collisions are possible at node B, either involving the MAC frames carrying TCP packets or the respective RTS frames, if the RTS/CTS handshake is enabled.

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.

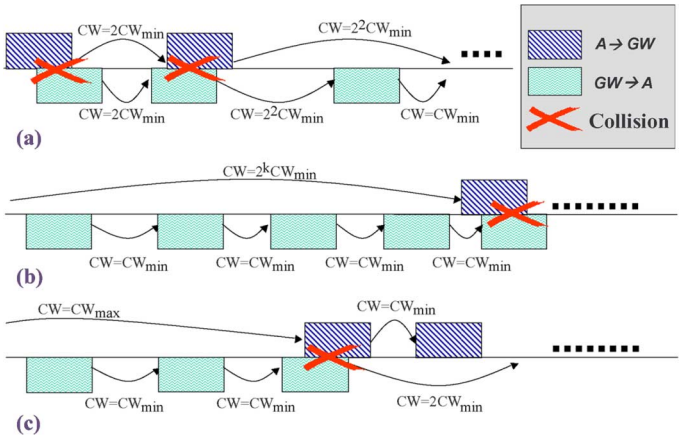

Fig. 4. Illustration of the multipacket capture of the channel by either node A or GW. (a) Small contention window results in collision with high probability. (b) Node GW succeeds to transmit a packet and resets its contention window. It may transmit multiple packets due to the high contention window 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.

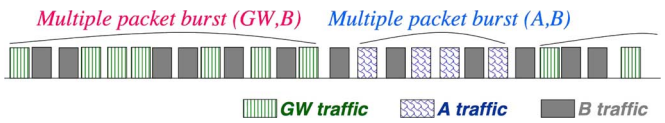

Fig. 5. Illustration of bistability with alternation of (A, B) and (B, GW) transmissions. Whenever A (GW) enters in the *success state*, a burst of packets is transmitted by A (GW) *and* B. The length of the burst depends on the value of the MAC maximum retry limit and on the backlog of the transmission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

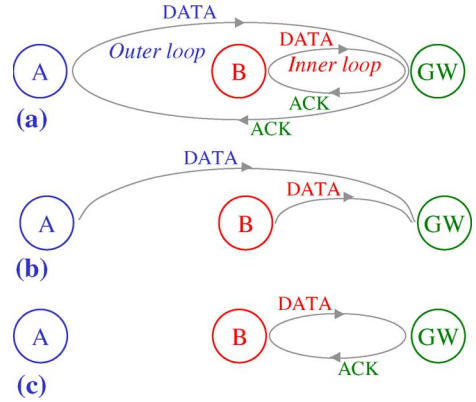

Fig. 6. Illustration of multiple control loops and a shared medium. (a) Two overlapping TCP congestion control loops are formed by TCP flows generated by A (*outer loop*), and B (*inner loop*). (b) When A enters the *success state*, mesh nodes A and B can transmit TCP DATA, but they cannot receive TCP 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,

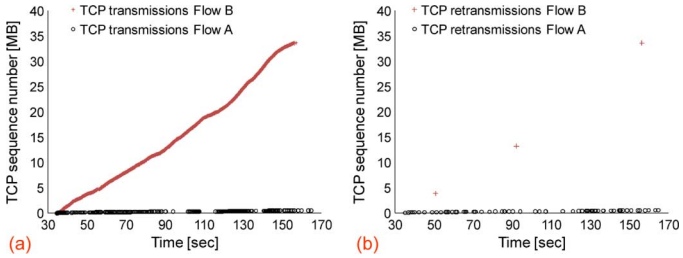

Fig. 7. TCP segment transmissions (new segment transmissions plus TCP retransmissions due to TCP timeout expiration) as captured by the sniffers next to nodes A and B. MAC retransmissions (due to MAC timeouts expiration) are not reported in the figures. (a) All TCP segment transmissions and retransmissions.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

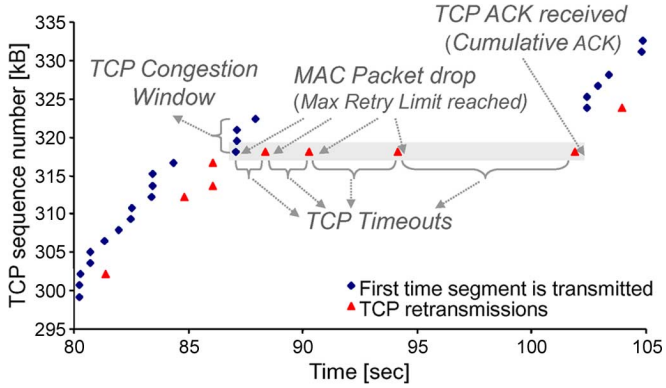

Fig. 8. A sample of node A's TCP (re)transmissions as captured by the sniffer next to node A. The severe penalty incurred by node A, due to MAC packet drop, can be seen in long idle periods due to long TCP timeouts for every TCP retransmission. Note that this idle period is exponentially increased for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

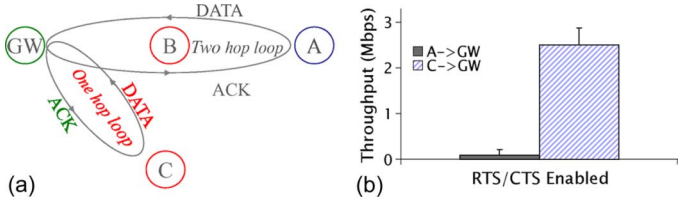

Fig. 9. Two-branch scenario and experimental TCP throughput with contending flows. (a) The scenario is composed of two branches: $A \rightarrow B \rightarrow GW$ and includes a two-hop loop, and $C \rightarrow GW$, characterized by a one-hop control loop. (b) Despite the RTS/CTS handshake mechanism,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

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.