Starvation in Operational Urban Mesh Networks: Compounding Effects of Congestion Control and Medium Access

Jingpu Shi[1], Omer Gurewitz[1], Vincenzo Mancuso[2], Joseph Camp[1], and Edward W. Knightly[1]
[1]ECE Department, Rice University, Houston, TX
[2]DIEET, Universita‘ di Palermo, Italy

Abstract

Using an operational urban mesh network serving nearly 2,000 users, we show that one-hop TCP flows can obtain high throughput whereas multi-hop TCP flows starve. We characterize the origins of starvation as compounding effects of mechanisms within the medium access protocol and end-to-end congestion control protocol. We develop an analytical model that captures this phenomenon and yields a solution: We propose a counter-starvation policy in which the gateway's one-hop neighbors increase their minimum contention window in comparison with other nodes. The policy therefore requires no change to TCP nor IEEE 802.11. Despite its simplity, we demonstrate through the model, experiments, and simulations, that the policy has a powerful effect on network-wide behavior, shifting the network's queueing points, mitigating problematic MAC behavior, and ensuring that TCP flows obtain a fair share of the gateway bandwidth, irrespective of their spatial locations.

1 INTRODUCTION

Large-scale mesh network deployments are planned and underway in cities across the world. According to In-Stat, the market will grow from 248 cities in 2005 to 1,500 cities in 2010, becoming a $1B industry in mesh access point sales alone. The prevailing architecture for large-scale deployments is a two-tier architecture in which an access tier connects end users' PCs and mobile devices to mesh nodes and a backhaul tier forwards traffic to and from high-speed gateway nodes. Directional antennas terminating at the gateway are also used to expand coverage and capacity.

We have deployed and are operating UrbanMesh,[1] a two-tier mesh network serving a user population of nearly 2,000 users in a 3 km^2 urban community. As most deployments are in the planning or early-deployment stages, to the best of our knowledge, this is the highest density urban mesh network operating to date.

Unfortunately, we have observed that under heavy load, flow *starvation* will occur in which one-hop flows obtain high throughput whereas competing multi-hop flows obtain near zero throughput. The phenomena occurs under two hardware/software platforms as well as in simulation. Clearly, for mesh networks to be successful, it is critical that network resources are distributed fairly among users, irrespective of their spatial location.

In this paper, we (i) perform extensive measurements in UrbanMesh to characterize starvation, (ii) study starvation's protocol origins, (iii) develop an analytical model to capture the interaction of medium access and congestion control that leads to starvation and (iv) design, justify, and evaluate a counter-starvation

1. We refer to the network as UrbanMesh for double-blind reviewing.

policy in which nodes one-hop away from the gateway increase their minimum contention window. In particular, our contributions are as follows.

First, we experimentally demonstrate the existence of starvation in UrbanMesh. Moreover, we design a set of experiments to isolate the factors that cause starvation. We identify that only a one-hop TCP flow coupled with a two-hop TCP flow is sufficient to induce starvation. Moreover, we demonstrate that starvation is not merely a hidden-terminal effect by showing that starvation does *not* occur if the TCP flows are replaced by UDP flows and IEEE 802.11's RTS/CTS handshake is used to counter hidden terminals; yet, the use of RTS/CTS is irrelevant for TCP flow starvation.

Second, we describe the protocol origins of starvation as a compounding effect of three factors. First, the medium access protocol induces bi-stability in which pairs of nodes alternate in capturing system resources. Second, the system's nested congestion control loops utilizing the wireless medium make it more likely that outer loops (multi-hop flows) are disrupted rather than inner loops (single-hop flows). Third, and most critically, the system incurs a high penalty when switching between the two states of the bi-stable system. In particular, a state is normally exited when either a flow "runs out of ACKs" and thus cannot transmit data because it cannot receive feedback, or when a DATA packet is dropped at the medium access layer.

Third, we develop an analytical model to both study starvation and to drive the 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 yields a *Counter-Starvation Policy* in which all of the gateway's neighbors should increase their minimum contention window to a value significantly greater than that of other nodes.[2] The model also characterizes *why* the policy is effective in that it forces all queueing 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, bi-stability is eliminated such that the subsequent penalties are not incurred.

Finally, we validate the Counter-Starvation Policy by re-deploying a manageable set of MirrorMesh nodes on-site (mirroring a subset of the UrbanMesh mesh nodes). The results demonstrate that our solution completely solves the starvation problem for TCP upstream and downstream traffic. We extend our investigation to a broader set of scenarios using simulations and show that our solution enables TCP flows to fairly share the gateway bandwidth in more general scenarios.

The rest of the paper is organized as follows. In Section 2, we introduce UrbanMesh. In Section 4, we demonstrate the existence of starvation experimentally and analyze its protocol origins. We develop the analytical model and Counter-Starvation Policy in Section 5. In Section 6, we evaluate the policy with measurements and simulations. We discuss related work in Section 7 and conclude in Section 8.

2 URBANMESH: AN OPERATIONAL ACCESS NETWORK

In this section, we describe the UrbanMesh network, its hardware and software platform, and our experimental setup.

2.1 Network Description

UrbanMesh is an operational two-tier mesh network that provides Internet access in a densely populated, single-family residential, urban neighborhood. The network is two-tier because it consists of a backhaul tier which wirelessly forwards data and an access tier which wirelessly provides access between end-users and the mesh infrastructure.

2. In existing mesh deployments, all nodes employ the same value as recommended by the IEEE 802.11 standard.

UrbanMesh backhaul nodes are predominantly deployed on single-story residences with the exception of two schools, two businesses, and a public library within the neighborhood. The current mesh infrastructure is composed of 18 backhaul nodes which coordinate to share the Internet bandwidth from a single fiber that is burstable to 100 Mbps. Fig. 1 depicts the spatial distribution of the UrbanMesh backhaul tier and the connectivity map. In the figure, nodes are depicted as connected if a direct transmission can occur between the two backhaul nodes. All links are omni-directional with the exception of a directional link (shown in black) which serves as an additional point of capacity for the network.

Each mesh node serves access nodes or clients wirelessly within its immediate proximity of approximately 200-300 m in radius. The radius is limited primarily by heavy tree foliage and densely packed homes with lot sizes averaging only 510 m^2.

At the time of our experiments, there are nearly 2,000 users in the network in an area of nearly 3 km^2, i.e., approximately 600 users and 6 backhaul nodes per km^2, which makes UrbanMesh one of the most dense mesh networks operating to date; for comparison, the St. Cloud mesh network currently serves 120 users per km^2 [1]. There are 4,760 residents per km^2 in the served neighborhood; as a point of reference, the average population density of the 20 largest U.S. cities is 2,800 residents per km^2. Planned and deployed commercial mesh networks also employ this two-tier architecture in which Internet gateways feed multiple multi-hop wireless paths and directional links are used to provide high-speed "short-cuts" to the gateway for capacity improvement. The process is then duplicated every 3-10 km^2 for larger coverage areas.

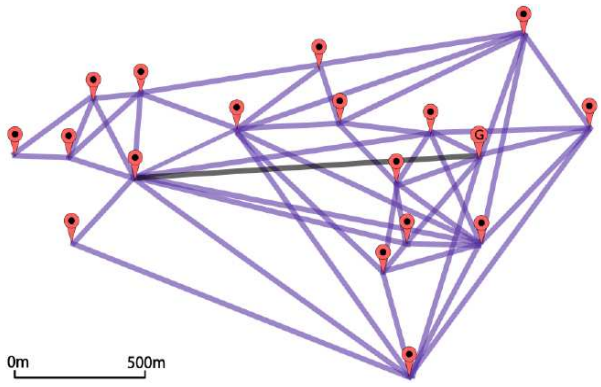

Fig. 1. Connectivity graph of the UrbanMesh backhaul topology with appropriate scaling for distance between nodes. There are nearly 2,000 residential users (not shown).

2.2 Hardware Platform

The UrbanMesh platform is programmable and observable. Each of the UrbanMesh nodes runs an open-source operating system. We perform extensive, non-intrusive, and privacy-respecting measurements consisting of detailed packet and signal measurements for network operations, modeling, and protocol design. The UrbanMesh nodes have much greater processing power (1GHz) and storage (4 GB) than most commercial mesh nodes to handle advanced protocol design and rapid data logging. The Linux operating system is derived originally from the open-source LocustWorld mesh networking software which uses AODV routing and HostAP drivers.

Each mesh node has a single, SMC 2532-B 802.11b wireless adapter with 200 mW transmission power to serve both backhaul and access traffic. The cards connect to a 15 dBi omni-directional antenna with a vertical beamwidth of 8 degrees. The mini-ITX motherboard is encased in a waterproof enclosure installed on the outside of building structures of backhaul deployment locations. The backhaul antennas are attached to the sides of homes at 10m height, and at slightly greater height (maximum of 20m) at the library, schools, and businesses. The client access node hardware is in many cases unknown to us. Yet,

it is clear that a wide variety exists, from PCs employing an external USB WiFi antenna placed near a window to laptops.

2.3 Experimental Setup

In each experiment of Section 4, we generate traffic (TCP and UDP) using *iperf* and measure the achieved throughput. Before each experiment, we measure the throughput when each of the flows is singly active to ensure good channel state. Unless stated differently, our measurement intervals are 120 seconds. The maximum PHY rate is 11 Mbps and the radio band adopted is channel 6 of the 2.4 GHz ISM band. All experiments on UrbanMesh take place in the presence of the network's normal user traffic.

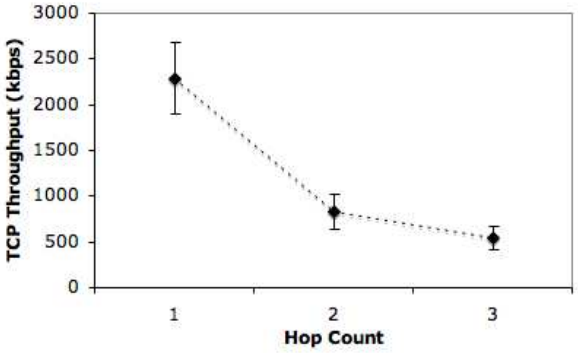

Fig. 2. TCP throughput of a singly-active flow from each hop of a three-hop route within UrbanMesh.

For the remainder of the paper, we explore the relationship between one-hop and two-hop flows simultaneously contending with one another. As a baseline, we first characterize the behavior of a single active flow over a multihop route. In Fig. 2, each point represents one singly-active TCP flow generated 1, 2, or 3 hops away from the gateway from the same route. In the measurements, RTS/CTS is disabled, and the experiment is run during the off-peak hours to minimize the effects of background traffic. The effect of TCP throughput vs. hop count and its reasons have been extensively studied in prior work, e.g., [22], and we do not review it here. Rather, we present this result as a baseline to study *multiple* contending flows and the resulting starvation phenomena.

3 Origins of Starvation

In this section, we first demonstrate the existence of starvation of two-hop TCP flows in UrbanMesh. We then describe the factors inherent in medium access protocols, congestion control protocols, and their joint behavior, that lead to starvation. Finally, we show that analysis of these factors predicts that starvation should also occur in a multi-branch scenario and we experimentally show that this is indeed the case.

3.1 Measurements in UrbanMesh

Here, we experimentally demonstrate the existence of starvation in UrbanMesh. Moreover, we experimentally isolate the originating starvation factors by eliminating alternate explanations such as background congestion and hidden terminals. Because we cannot directly control the system's offered load due to the large user population, we perform experiments at different times of the day and night to study this factor.

In all experiments, user traffic from community residents is originating from all nodes of the network. We select three nodes A, B, and the gateway GW depicted in Fig. 13 for our test workload and measurements. Although A is physically close to the gateway, it is not in radio range of the gateway due to the propagation environment, i.e., A and the gateway are not in transmission range nor interference range due to obstacles

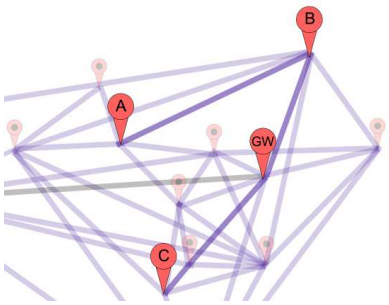

Fig. 3. Nodes used within the UrbanMesh for traffic generation and measurement. Background traffic exists from hundreds of users from a user population of near 2,000.

such as trees and houses. All of A's packets to and from the gateway are forwarded by node B, as verified by observing the routing table. Therefore, node A has a two-hop route whereas node B has a one-hop route to the gateway GW.

3.1.1 Starvation under Heavy User Load

We first explore heavy background load which occurs daily between the hours of noon and 10pm. In each trial of the experiment, we simultaneously generate a long-lived TCP flow from the two-hop node (A) and a TCP flow from the one-hop node (B) to the gateway (GW). Thus, in all experiments, all three nodes mutually contend for channel access in support of both uplink data and downlink acknowledgements.[3]

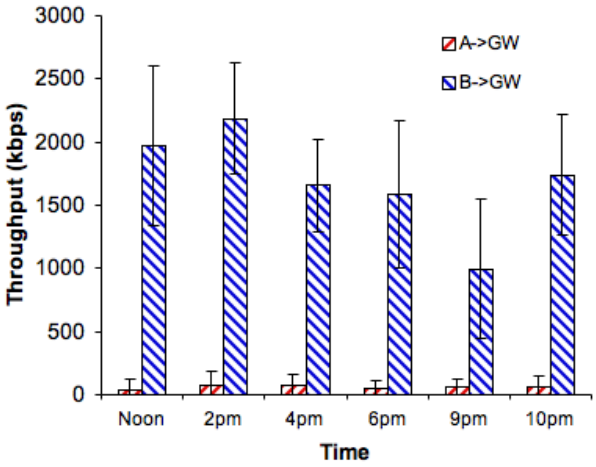

Fig. 4. Two-hop chain contention measured at six different times of the day.

Fig. 14 depicts the throughput of the two flows and illustrates that severe starvation occurs. 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 conducted at different times under different traffic load of the network. Moreover, before and after each experiment, we ensure that link A-B, as well as all other links, are fully operational and that full throughput can be achieved when each link is used *alone*.

3.1.2 Starvation under Light User Load

A potential explanation for the two-hop flow's starvation is that node A has high contention and interference due to its neighbors. We eliminate this hypothesis by repeating the above experiment at 3am when network

3. To ensure that our results are not unique to injecting a *single* flow from both nodes in the presence of many background flows, we also generate *aggregate* flows from both nodes and obtain nearly identical results which are not shown.

activity is at a minimum. Fig. 5 (left) shows that even when the background load is light, indicating low contention and interference, starvation persists. Thus, inter-flow contention between the two injected TCP flows alone is sufficient for starvation.

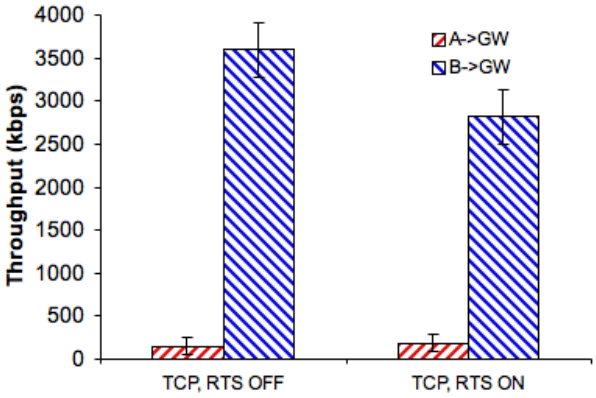

Fig. 5. Single Branch Upload: $A \rightarrow B \rightarrow GW$ is a chain of nodes and GW is the gateway. Congestion with other mesh and access nodes is minimized by running the experiments from 3am to 6am.

3.1.3 Starvation with RTS/CTS

In the above experiments, RTS/CTS is turned off. However, a second potential explanation for starvation is that nodes A and GW suffer from a hidden terminal effect when A transmits DATA and GW transmits ACKs. In particular, the second-hop node A and the gateway node GW are out of range. Consequently, without the RTS/CTS handshake, the two nodes are *hidden* [25] such that carrier sense is ineffective. Thus, RTS/CTS provides a mechanism to coordinate nodes that cannot carrier sense each other but can carrier sense a common node (B in this case).

Here, we invalidate the hypothesis that use of RTS/CTS would eliminate starvation. Fig. 5 (right) shows that despite the 2nd-hop throughput being slightly improved with RTS/CTS, the throughput imbalance is nearly as severe as that with RTS/CTS off.

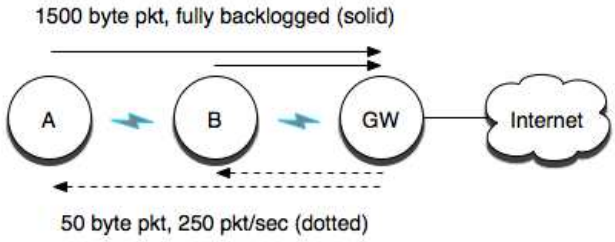

Fig. 6. UDP traffic matrix to compare to the bidirectional nature of TCP sending 1500 bytes upstream (DATA) and 50 bytes downstream (ACK).

3.1.4 No Starvation for UDP with RTS/CTS

Finally, we show that starvation is not simply due to the traffic matrix, i.e., simultaneously having two- and one-hop upload and download flows. Rather, it is the compound effects of TCP and medium access mechanisms that are required to induce starvation. We show this via experiments in which TCP's DATA uplink traffic is replaced with fully backlogged 1500 byte UDP DATA packets, and TCP's ACK downlink

traffic is replaced with 50 byte UDP DATA packets. This 50-byte packet rate is 250 packets per second, corresponding to the ACK rate of a saturated TCP flow. This scenario is illustrated in Fig. 17.

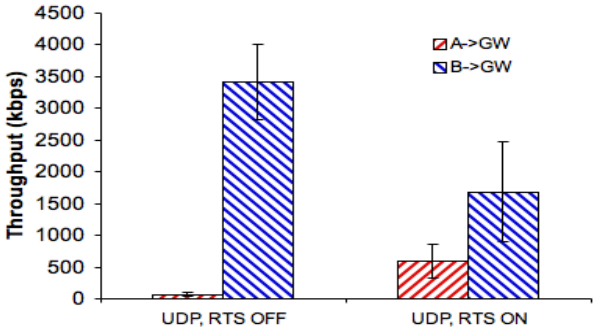

Fig. 7. Same two-hop chain without the windowing mechanism. Hidden terminals (RTS OFF) cause starvation, but without the windowing, traffic has much more equal distribution (considering the two-hop flow takes twice the resources as the one-hop flow).

The results of the experiment are depicted in Fig. 18. First, we revisit the issue of RTS/CTS. Comparing the cases of RTS/CTS off (left) and RTS/CTS on (right), it is clear that for UDP, the use of RTS/CTS yields a "fair" bandwidth distribution. In particular, when the RTS/CTS mechanism is enabled, the two throughputs reflect that the two-hop flow utilizes twice the resources at node B compared to the one-hop flow. Thus, for UDP traffic, RTS/CTS has the desired effect of countering hidden terminals.

Second, comparing TCP and UDP when both have RTS/CTS on, Fig. 5 (right) and Fig. 18 (right), we conclude that it is not a consequence of hidden terminals, but rather mechanisms within TCP that are necessary for starvation to occur.

3.2 Starvation's Protocol Origins

Here we describe how the protocol mechanisms of medium access and congestion control mechanisms interact to cause starvation.

Medium Access and Bi-stability. The collision avoidance mechanism in CSMA/CA, with or without RTS/CTS, causes node pairs (A, B) and (B, GW) to 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 the GW is idle, and a state in which A is idle while GW and B transmit. As depicted in Fig. 19, when mesh node A (or GW) wins the channel, it enters a success state in which it will transmit a burst of packets, while GW (or A) enters a fail state in which it will not succeed in transmitting any packets. Node B, which is in sensing range with the other two nodes, is always in a success state. Hence, it sends a steady flow of DATA to GW, and potentially, ACKs to A. In other words, node B contends fairly with the other node that is in a success state and interleaves its packets with the burst generated from either A or GW. Note that a resembling phenomenon has been previously illustated in [6], [20] for a simpler scenario consisting of two one-hop flows.

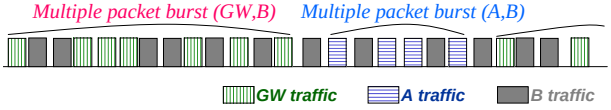

Fig. 8. Depiction of bi-stability with alternation of (A, B) and (B, GW) transmissions.

The Congestion Control Loops. TCP's congestion control mechanism creates a closed-loop system between each sender-receiver pair in which the transmission of new packets is triggered by the reception

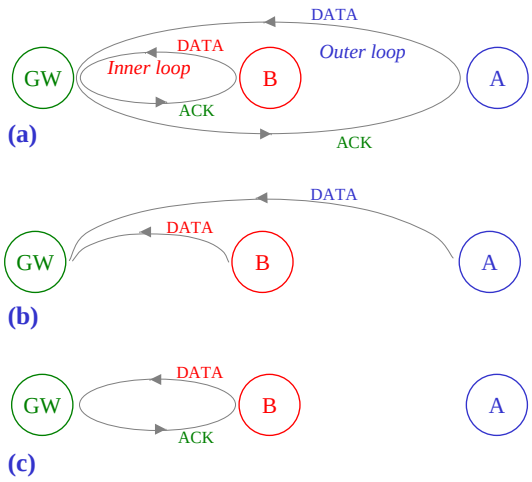

Fig. 9. Illustration of multiple control loops and a shared medium.

of acknowledgments. The two-hop scenario contains two nested transport loops, one for each upstream flow. We term the single-hop and the two-hop loops as the inner loop and outer loop respectively, as depicted in Fig. 20(a). The outer loop consists of four links, two upstream links and two downstream links. Because the inner loop only consists of two links which are a subset of the links of the outer loop, the outer loop is more likely to be broken due to any of its links not being able to transmit over a period of time.

Penalty to Switch States. When switching between the two states, the system incurs a severe penalty, namely, high delay and/or a dropped packet. Yet, the system's bi-stability and alternation between success and fail states has different effects on the two transport loops. In particular, when A bursts, the gateway is in a fail state, and both transport loops are broken (Fig. 20(b)). However, due to the lack of feedback (ACKs) from the gateway, A's burst length is limited by its congestion window.

Conversely, when GW bursts and A is in a fail state, only the outer loop is broken. In this case, the inner loop is self-sustaining due to the loop's own ACK generation (Fig. 20(c)). Consequently, the duration for GW and B to jointly capture the channel is not bounded in the way A's duration is bounded. Thus, the ratio between the average durations of the capture states for GW and A is large. Furthermore, when GW bursts, most of the packets transmitted belong to the inner loop, i.e., the one-hop flow, due to the fact A is in a fail state and therefore can not inject new packets to the network. To exit this state, A again pays a high cost: because of binary exponential backoff, A mostly likely drops a data packet in order to have an equal chance of capturing the medium to enter the success state. If A loses contention again, it will not have an equal chance to capture the medium until it drops a subsequent packet.

3.3 Prediction for Two Branches

In this section, we show that the starvation origins described in Section 3.2 are able to correctly predict that starvation will also occur in an alternate scenario.

According to Section 3.2, TCP starvation is caused by the joint effect of mechanisms in medium access contention resolution and closed loop congestion control. Moreover, if a scenario contains a one-hop loop sharing a common node with a two- or more- hop loop, starvation will occur.

In this experiment, the one-hop loop and the two-hop loop join at the gateway as depicted in Figure 23. 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 carrier sense range yielding bi-stable 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.

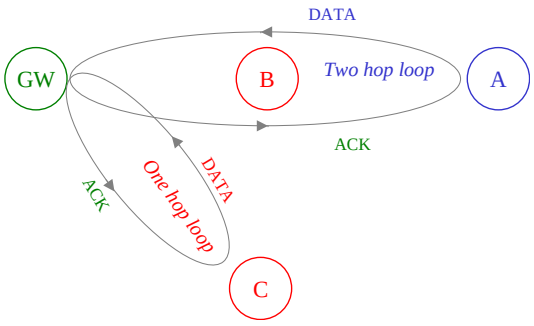

Fig. 10. Two Branch Loop

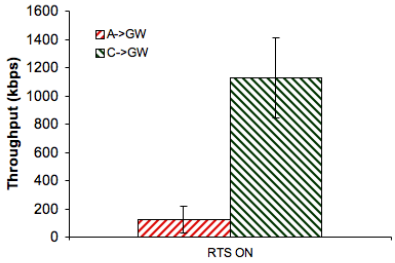

Fig. 11. Two Branch Upload: $A \rightarrow B \rightarrow GW$ is a chain of nodes, GW is the gateway, and $C \rightarrow GW$ is another branch of the mesh. The two-hop flow has slightly higher throughput, but starvation still occurs.

In UrbanMesh, we select another one-hop node C, as depicted in Fig. 13. Two TCP flows are active on the two branches $A \rightarrow B \rightarrow GW$ and $C \rightarrow GW$, respectively. Fig. 24 depicts the result of the experiment and shows that starvation persists 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.

4 Origins of Starvation

In this section, we first show TCP starvation observed in UrbanMesh, an operational two-tier urban mesh network. We then reveal the origin of starvation through analysis. Finally, we examine more general topologies within UrbanMesh to show that starvation persists beyond the two-hop chain.

4.1 Experimental Set-up

In order to measure goodput, we generate long-lived TCP flows using *iperf* and measure the achieved goodput. Unless stated differently, our measurement intervals are 120 seconds. As can be seen in Fig. 12, the UrbanMesh links are very stable during each run, and goodput does not vary much (low standard error), hence 120-second interval captures the behavior of the observed path with minimum interference with the UrbanMesh users. Unless stated differently, the mesh nodes are configured based on their default configuration. The RTS/CTS mechanism is disabled, the intra-mesh routing protocol is AODV, the maximum PHY rate is 11 Mbps and the radio band adopted is channel 6 of the 2.4GHz ISM band.

4.2 Measurements on UrbanMesh

In order to show starvation within an urban mesh network, We run an extensive set of measurements on the three nodes, A, B and GW,

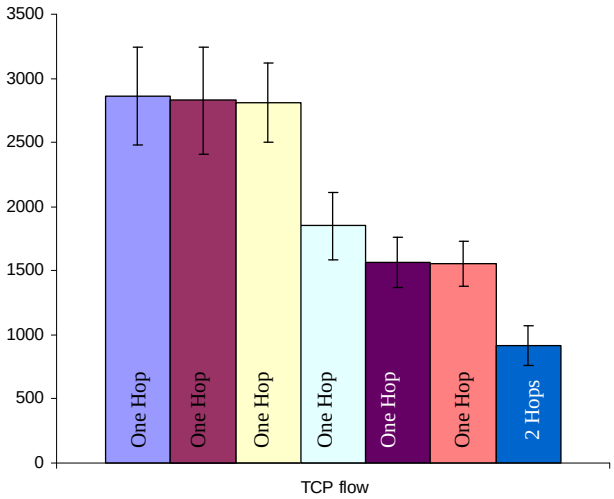

Fig. 12. Available TCP Goodput for selected UrbanMesh nodes under normal daytime load.

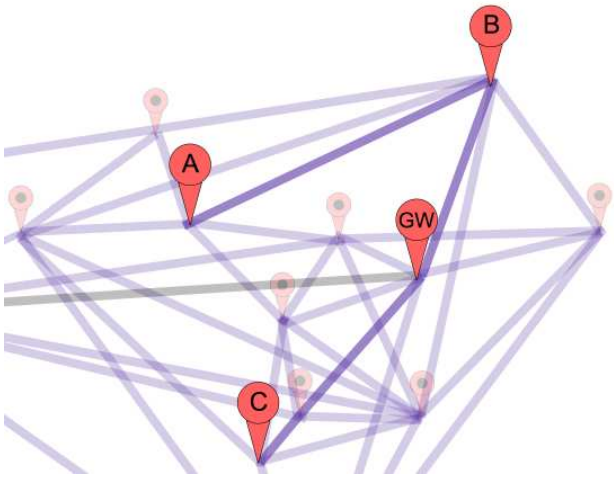

Fig. 13. Nodes used within the UrbanMesh network to create a three-node chain and a three-node chain with an additional branch from the gateway.

4.2.1 Starvation Under Normal Network Load

We select three nodes A, B, and the gateway GW depicted in Fig. 13. Although A is physically close the gateway, due to the propagation environment, it is not in radio range of the gateway GW[4]. According to the routing table of the network, A's packets to and from the gateway are all forwarded by node B. Therefore, node A has a two-hop route whereas node B has a one-hop route to the gateway GW.

We conducted a set of measurements in UrbanMesh between the hours of noon to 10pm under normal network operations. In each experiment, we simultaneously generate a TCP flow from the two-hop node (A) and a TCP flow from the one-hop node (B) to the gateway (GW), i.e., in all experiments, node A and B mutually contend for channel access.

We present the goodput of the two flows in Fig. 14, which shows severe starvation occurs. In all experiments being conducted at different times under different traffic load of the network, the one-hop TCP flow from node B dominates the channel while the two-hop TCP flow from node A receives nearly zero throughput.

4. A and the gateway GW is not in transmission range nor in interference range, due to obstacles between them such as trees and houses.

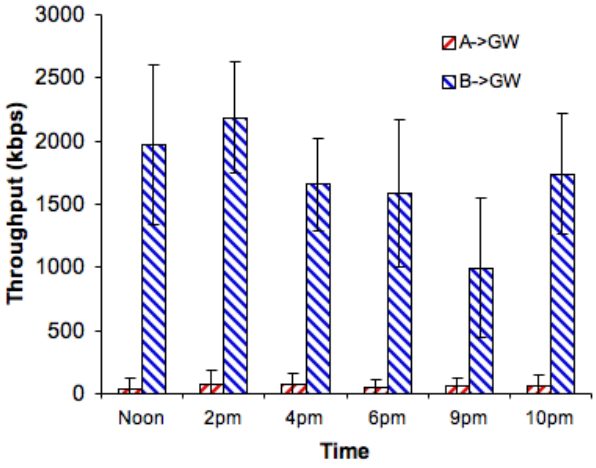

Fig. 14. Two-hop chain contention measured at six different times of the day.

4.2.2 Starvation on Clean Channel

Under normal network operations, the interference level for the two flows is unpredictable. If due to the activity of other users, the 2nd hop node receives much more interference than the 1st hop node, the two-hop TCP flow can be starved. To examine whether this hypothesis is true, we repeat the experiment at 3am when the network activity is minimum. Fig. 15 reports that even when the channel is relatively clean, starvation persists. Inter-flow contention between the two flows is sufficient for starvation.

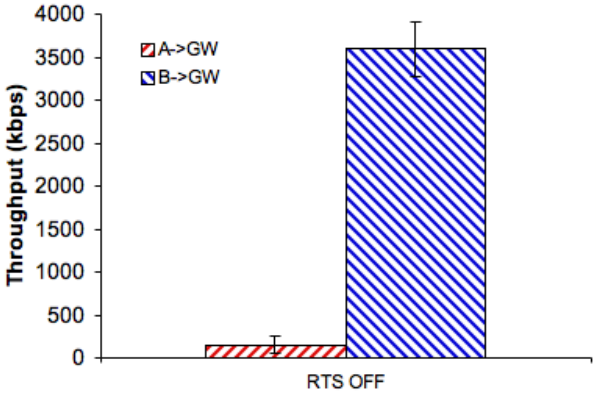

Fig. 15. Single Branch Upload: $A \rightarrow B \rightarrow GW$ is a chain of nodes and GW is the gateway. Congestion with other mesh and access nodes is minimized by running the experiments from 3am to 7am.

4.2.3 Starvation with RTS/CTS

In Fig. 14, severe starvation occurs with default network configurations, i.e., RTS/CTS is disabled. Note that TCP ack packets are transmitted from the gateway node GW, which also contend for channel access. Since the second-hop node A and the gateway node GW are hidden terminals with respect to each other, the well known hidden terminal effect is present in the network. we experimentally explore whether the hidden terminal effect with RTS/CTS off is the root cause of starvation. In this experiment, we turn RTS/CTS on to combat the hidden terminals, and again let first hop node A and second hop B all transmit TCP flows to the gateway simultaneously. The experiment result is reported in Fig. 16 shows that despite

of the 2nd-hop throughput being slightly improved with RTS/CTS, the throughput imbalance is as nearly the same as that with RTS/CTS off. Next we analyze the root cause of starvation in this two-hop chain topology and see why RTS/CTS does not solve the problem.

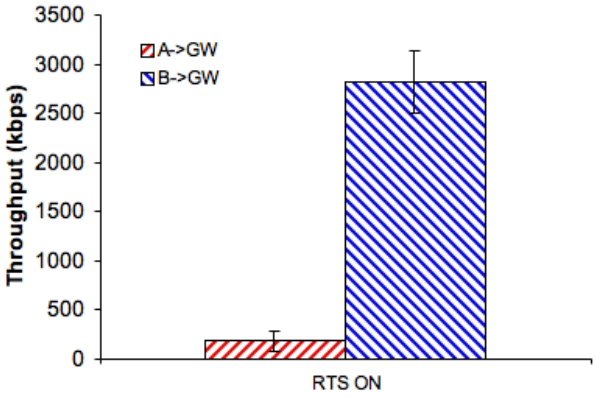

Fig. 16. Single Branch Upload: $A \rightarrow B \rightarrow GW$ is a chain of nodes and GW is the gateway. Congestion with other mesh and access nodes is minimized by running the experiments from 3am to 7am.

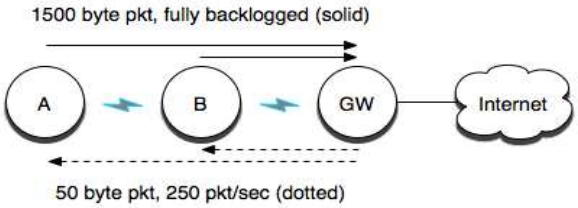

Fig. 17. UDP traffic matrix to compare to the bidirectional nature of TCP sending 1500 bytes upstream (DATA) and 50 bytes downstream (ACK). The upstream packets are fully backlogged and the downstream sent at a rate of 2k pkt/sec which would be approximately the rate of ACKs for a 3 Mbps link.

4.2.4 Removing Windowing

We finally consider UDP traffic on the same three nodes to remove the windowing mechanism. We have generated 1500 byte packets upstream to represent TCP data packets and 50 byte packets downstream to represent acknowledgment packets (see Fig. 17). We have fully backlogged the upstream packets and generate the downstream packets at 2000 pkt/sec (the expected rate of a 3 Mbps TCP connection considering there is a 30:1 ratio between the size of the two packets and that each DATA packet has a corresponding ACK). We find in Fig. 18 that with RTS/CTS disabled we still have starvation which can be attributed to the well-known hidden terminal problem [17]. However, when the RTS/CTS mechanism is enabled we see that there is a much more fair distribution of traffic considering that the two-hop flow takes twice the resources at node B that the one-hop flow. We conclude that the windowing mechanism of TCP combined with the innate IEEE 802.11 MAC protocol is the cause of the severe starvation that we have seen in the same scenario with TCP traffic without hidden terminals. We now elaborate on this relationship.

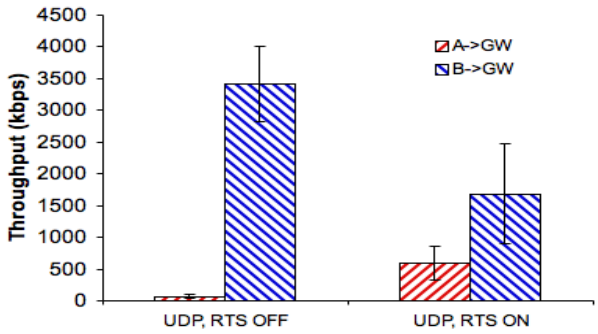

Fig. 18. Same two-hop chain without the windowing mechanism. Hidden terminals (RTS OFF) cause starvation, but without the windowing traffic has much more equal distribution (considering the two-hop flow takes twice the resources as the one-hop flow).

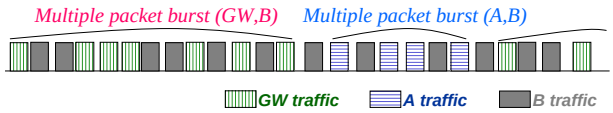

Fig. 19.

4.3 High Level Explanation

Here we show that the starvation experienced by the two-hop flow is a result of the coupling of two separate mechanisms located in two separate layers of the protocol stack. The first is the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), which is a contention-based protocol that manages the medium access of the wireless devices (MAC Layer). The second mechanism is the window based flow control operated at the transport layer, such as TCP congestion control. In this section, we explain the impact of the coupling between the two mechanism on throughput distribution. We start with high level description the role each factor contributes to the starvation, followed by detailed discussions. Later, in Section 5, we will suggest an analytical model that completes the understanding of the starvation.

The role of CSMA/CA MAC As we will show in this section, the collision avoidance mechanism in CSMA MAC causes nodes A and GW to alternate successful bursts of transmissions, which contributes to starvation when coupled with flow control in the transport layer. For the ease of presentation, we use an *Success/Fail* source model to describe this burst behavior of the two nodes: when mesh node A (GW) wins the channel, it enters a *Success* state in which it will transmit a burst of packets, while GW (A) enters its *Fail* state in which it will not succeed in transmitting any packets. Node B, which is in sensing range with the other two nodes, is always in *Success* state. Hence, it sends a steady flow of packets to the GW, i.e., the MAC will allow it to contend fairly with the node that is in *Success* state and interleaves its packets with the burst generated from either A or GW (see Fig. 19).

The role of transport Layer Sliding Window Transport layer sliding window congestion control creates a closed-loop system between each sender-receiver pair, meaning the transmission of new packets is triggered by the reception of acknowledgments. In our two-hop scenario, we have two nested transport loops, one for each upstream flow. We term the single-hop and the two-hop loops as *inner loop* and *outer loop* respectively (Fig. 20(a)). The *outer loop* consists of four links, two upstream links and two downstream links. The *inner loop* only consists of two links which are a subset of the links of the *outer loop*. Consequently, the *outer loop* is more likely to be broken due to any of its links not being able to transmit over a period of time.

Coupling of CSMA and Sliding Window The occurrence of burst alternation has different effects on the two transport loops. For example, when A bursts, the gateway GW is in fail state, and both transport loops are broken (Fig. 20(b)). However, due to the lack of feedbacks (ACKs) from the gateway GW,

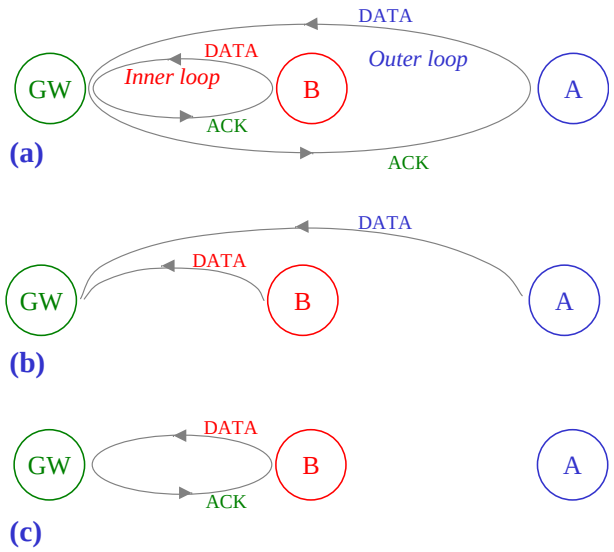

Fig. 20.

this burst length is limited by its congestion window. Conversely, in the case GW bursts (A is in fail state), only the outer loop is broken. In this case, the inner loop is self-sustaining due to ACK generation (Fig. 20(c)). Its burst length is therefore unbounded. As a result, the ratio between the burst lengths from GW and A is large with high probability. Furthermore, when GW bursts, most of the packets transmitted belong to the inner loop, i.e., the one-hop flow, due to the fact A is in its *Fail* state and therefore can not inject new packets to the network. Next we give a detailed explanation about the cause of starvation.

4.4 Detailed Explanation of Starvation Factors

We now describe in detail the burst nature of transmission in the MAC and how the MAC and window-based congestion control jointly cause starvation. For the ease of presentation, we use IEEE 802.11 as an example of a CSMA/CA protocol. Here we assume RTS/CTS is enabled. The same analysis can be extended to the case with RTS/CTS off: we only need to replace RTS packet with contending DATA packet.

We observe that Node B is in transmission range with both A and GW, and hence, the collision probability is negligible for its packets[5]. Therefore, its contention behavior is predictable: B contends fairly with either mesh node A or GW whoever bursts, and the packets from B will be interleaved with the packets from either A or B whoever in its *Success* state. Moreover, both A and the gateway GW perceive the transmission of B in the same way, as shown in Fig. 21. Due to these reasons, when we discuss the interaction between A and GW, for simplicity we omit transmissions of node B, which will not change the interaction between A and GW.

High Collision Probability at Middle Node Since A and GW are not in sensing range of each other, if one of them starts an RTS transmission while the other one is transmitting an RTS, both RTS packets will fail. In order to avoid collision, no RTS can start $T_{RTS} + T_{SIFS}$ seconds before, and T_{SIFS} seconds after the transmission of RTS of the other node, where T_{RTS} is the duration of an RTS transmission and T_{SIFS} is the duration of a $SIFS$ defined in the standard. The vulnerable interval for an RTS packet is therefore $2 \times T_{RTS} + T_{SIFS}$, comparable to the minimum value of the contention window. As a result of long vulnerable interval, if one of the hidden node (either GW or A) is contending with small window, whenever the other node transmits an RTS packet (even after a long backoff), with very high probability a collision will occur. Due to the vulnerable interval of this RTS packet, the node with a small contention window is very likely to win the contention.

5. Indeed, for them to collide, A and B has to start their transmissions within a propagation delay interval. The probability of this occurring is negligible.

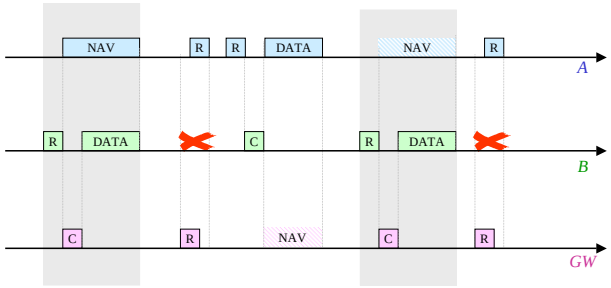

Fig. 21. Node A and GW perceive the same activity of node B

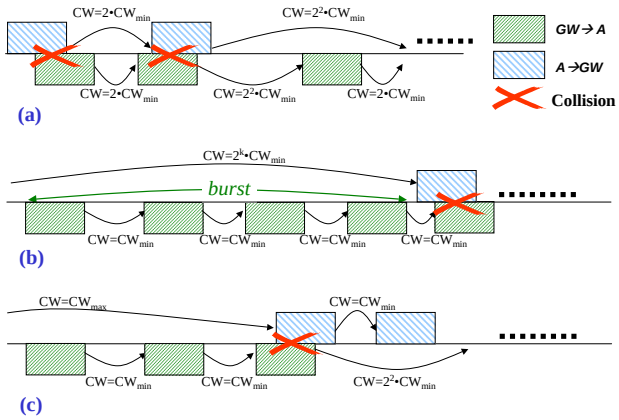

Fig. 22. MAC contention of two flows

Burstiness A successful transmission will eventually occur after one or more collisions, which will widen the two windows (Fig. 22(a)). Let us assume without loss of generality that GW is the mesh node that eventually wins the contention and successfully transmits an RTS. Obviously, due to MAC symmetry, the same holds for mesh node A. After GW transmits its RTS it will reset its contention window to CW_{min} and and transmit a burst of one or more packets before the mesh node A will try to transmit an RTS of its own (Fig. 22(b)). In order for A to win, it should place an RTS frame between two consecutive trails of GW. The probability of A successfully doing so is small, since GW is back to CW_{min} and the average idle channel between its two consecutive trails is small. With very high probability the RTS from A will collide with the RTS from GW (Fig. 22(b)). Each such collision causes both mesh nodes to increase their contention window; however GW goes to the second contention stage and A goes to a much higher contention stage, so that GW will likely transmit another burst. This burst will be longer than its previous one, due to the incremented contention window of A; e.g., with binary exponential backoff, the average of the burst will be doubled. GW is more and more likely to keep its winning streak and transmit longer and longer burst as A falls into deeper and deeper backoff stage. The process will repeat itself until the mesh node A eventually wins the channel and starts a burst of its own (Fig. 22(c)). The burst is terminated when (i) with low probability $A(GW)$ wins the contention while still in high backoff stage; (ii) the GW (A) has no transport-layer ACKs (no data packets) to transmit; (iii) A (GW) wins the contention with high probability after it reaches its maximum transmission limit and resets its contention window to CW_{min}.

4.4.1 Compound Effect of MAC and Sliding Window

As previously explained, node B keeps transmitting its packets during bursts of both A and GW. During the burst of the GW, it transmits ACKs to B. the transport layer sliding window mechanism then allows B to generate more traffic to GW upon receiving these ACKs. GW generates new ACKs upon receiving new traffic from B. The *Inner loop* is therefore self-sustaining. Due to this closed-loop feedback, the GW burst size is not upper-bounded, and most of the packets transmitted belong to the one-hop flow. On the

other hand during A's burst, the fact that GW is in *Fail* state breaks both both loops. Consequently, the number of ACKs fed back to A is limited to the number of ACKs backlogged at B at the beginning of the burst. Hence, the maximum burst size of A is bounded by the size of the transport layer congestion window.

Furthermore, when GW bursts, the queue at GW is backlogged because the *Inner loop* is self-sustaining. Then it is most likely that A has to reach its retry limit and drops its packet in order to exist its *Fail* state. As far as the transport layer adopts a sliding window mechanism such as TCP, the dropping costs a high a penalty. In fact, the next transmission opportunity will be completely wasted by retransmissions of previously dropped packets. Note that this process can be repeated several times for the same transport packet.

4.4.2 TCP Effect on Burstiness

TCP magnifies the negative impact of sliding window mechanisms at transport layer. TCP adapts the transmission windows with additive increments after ACK receptions and with multiplicative decrements after packet losses (e.g., by dropping). According to factor (iii), node A has to pay a multiplicative window decrease when it exists its fail state. As a result, its congestion window is small, making its burst short.

4.4.3 Summary

To summarize, the joint effect of window based congestion control in the transport layer and the MAC contention resolution mechanism causes starvation. A more rigorous explanation is given in Section 5, where the interaction between the two layers is modeled. This negative effect is magnified by TCP window dynamics. It is worth noting that, according to the famous formula for TCP throughput given by Towsley in [**?**], and also according to remarks given by Altman in [4] on the effects of end-to-end delay variations tipical of wireless networks, the TCP goodput is expected to roughly be inversely proportional to the Round Trip Time (RTT) and to the square root of loss probability ($\sqrt{p}$), which are exactly the parameters affected by the described coupling effect.

4.5 Validation on Two-branch Topology.

In this section, we first show that our analysis predicts much broader starvation scenario than the two-hop chain topology. We then validate our analysis by validating this prediction.

According to our analysis, TCP starvation is caused by the joint effect of the MAC contention resolution mechanism and the closed loop congestion control scheme. Burst traffic due to MAC contention broke different transmission links in the burst cycle, which has different effect on the one-hop loop (1st hop loop) and the longer loop (2nd hop loop). If in a scenario a one-hop loop shares a common node with a two- or more- hop loop, starvation is likely to occur.

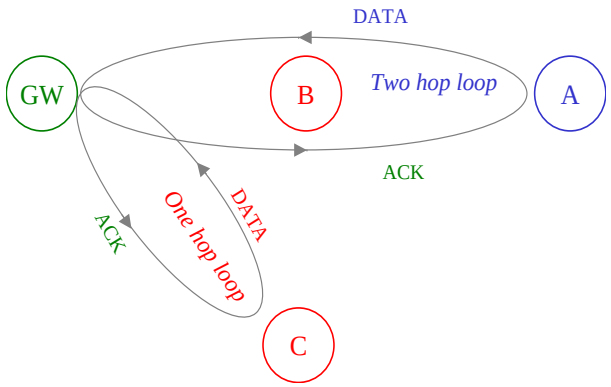

Fig. 23. Two Branch Loop

According to this analysis, we predict that in a two-branch topology shown in Fig. 23, where the one hop loop and the two hop loop joins at the gateway, starvation occurs. In this topology, although node C does not forward traffic for the node A, the same starvation analysis applies. The gateway GW and A are hidden terminals to each other and thus, the probability of collision between gateway node GW and node A is high, resulting in burst traffic. When GW bursts its traffic, the one-hop loop is self-sustaining. When A bursts its traffic, GW is silenced and both loops are broken. Consequently, the burst size of A is limited by its window.

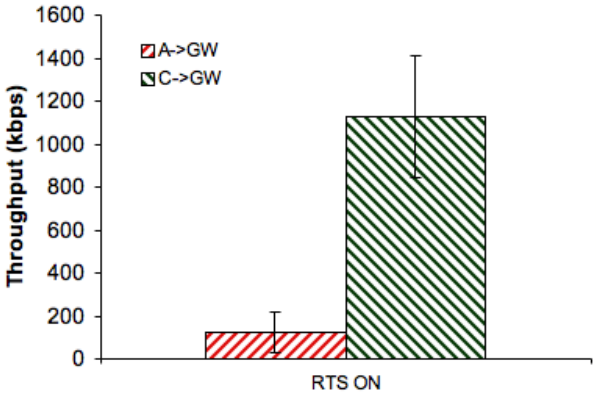

Fig. 24. Two Branches Upload: $A \rightarrow B \rightarrow GW$ is a chain of nodes, GW is the gateway, and $C \rightarrow GW$ is another branch of the mesh. The two-hop flow has slightly more goodput but starvation still occurs.

We now validate through UrbanMesh that starvation occurs in two branches shown in Fig. 13. Two TCP flows are active on the two branches $A \rightarrow B \rightarrow GW$ and $C \rightarrow GW$, respectively. Fig. 24 reports that starvation persists 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.

5 Analytical Model and Starvation Solution

In this section, we develop an analytical model to study the compounding effects of medium access and congestion control on starvation. We employ a highly simplified system model in order to isolate and study the root causes of starvation under the simplest conditions in which they arise. Finally, driven by the model, we propose a contention window policy to counter starvation.

5.1 System Model

As described in Section 3.2, the DATA-ACK control loop is a key factor in starvation. Consequently, we model only one aspect of congestion control, the sliding window, and in particular, we consider a *fixed* congestion control window. Consequently, when the corresponding analytical model predicts starvation, we can conclude that among congestion control's many mechanisms, the DATA-ACK control loop and a sliding window *alone* are sufficient to induce starvation.

For medium access, we also consider a highly simplified system model with an idealized physical layer in which GW and B, and A and B can communicate without channel errors. We consider a collision model in which even partially overlapped transmissions fail and require retransmission, e.g., overlapped transmission of RTS messages from A and GW or overlapped transmission of A's DATA with GW's ACK. We consider that the initial contention window of node i is given by $CW_{\min,i}$ and the contention window evolves according to the binary exponential backoff scheme. Moreover, we assume that the backoff

counter of each station is geometrically distributed over the current window. This assumption allows us to exploit the memoryless property of the geometric distribution and to avoid tracking the number of mini-slots already elapsed. This assumption is common and has been previously validated, e.g., [20], [23], [29]. Our model captures both RTS/CTS on as well as pure CSMA with RTS/CTS off.

In addition to medium access and end-to-end sliding window, we also model the queues at each node. We assume that a node contains a separate queue for each subflow, e.g., node B has a queue for downlink ACKs to node A, a queue for uplink DATA originating from A, and a queue for uplink DATA originating from B. Moreover, each time a node gains channel access, each of the node's queues receives service with equal probability. This assumption provides a memoryless property thereby aiding the model's tractability.

We will show that while this system model omits many aspects of our experimental system, it nonetheless captures starvation.

5.2 Model Description

As shown in Fig. 25, six sub-flows originated from the three mesh nodes need to be modeled. Included in the six sub-flows are three upstream DATA flows and three down-stream ACK flows, respectively, traversing to and from the gateway node. Correspondingly, we need to track the queue occupancy of the six flows as shown in the figure.

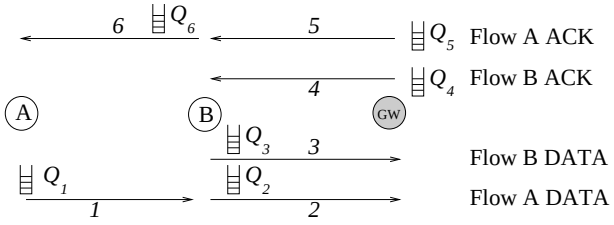

Fig. 25. Queues at different mesh points.

All possible channel states are illustrated in Fig. 26. Eight channel states are identified including three *DATA transmission states* occupied by upstream DATA transmissions on links 1, 2, and 3 in Fig. 25; three *ACK transmission states* occupied by ACK transmissions on links 4, 5, and 6; one *collision state* occupied by RTS (or DATA if the RTS/CTS mechanism is not used) collisions between the second-hop and gateway node; and one *idle state* occupied by an idle mini-slot to characterize when all nodes are counting down their backoff counters. The time instants of a possible channel state change are pointed by arrows placed below the temporal axis in Fig. 26. We label a transmission state using the index of the link on which this transmission occurs. For example, channel state 4 refers to transmission on link 4. We denote the duration of the transmission states, the collision state, and the idle state by $T_i (1 \leq i \leq 6)$, T_c and T_δ, respectively.

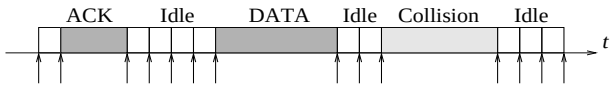

Fig. 26. Illustration of channel states.

We now construct a Markov chain model embedded over continues time at mini-slot boundaries in which all three nodes can (potentially) start transmitting the first packet of a new data exchange (either the RTS or the DATA packet), provided that their queue is not empty and their backoff counter reaches zero.

For the ease of presentation, in notations we use a, b and g to represent nodes A, B and GW, respectively. For node i, $(i \in \{a, b, g\})$, the parameter e_i of the geometric distribution that characterizes the backoff counter is given by $e_i = \frac{2}{CW_i}$, where CW_i is the current contention window of node i. With

e_i computed above, the mean backoff interval with geometric distribution is set to be the same as with the system's actual uniform distribution. Consequently, at any state-switching time epoch, a node with contention window CW_i attempts a new transmission with probability e_i.

We denote the length of queue i for link i as Q_i. Let $Q_g = Q_4 + Q_5$ be the aggregate queue length at gateway node GW and W_a, and W_b be the *fixed congestion window* for flow $A \rightarrow GW$ and flow $B \rightarrow GW$, respectively. 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,[6] compared to the collision probability between A and GW. We therefore assume that the middle node never collides and never doubles its backoff counter, i.e., $CW_B = CW_{\min,B}$.

In order to capture both the MAC contention status and the queue behavior, we represent the system state as $\mathbf{S} = \{Q_1, Q_2, Q_3, Q_g, CW_a, CW_g\}$. Although the length of some of the queues is not incorporated in the system state, they can all be expressed with (Q_1, Q_2, Q_3, Q_g) as follows.

$$\begin{aligned} Q_4 &= W_b - Q_3 \\ Q_5 &= Q_g - (W_b - Q_3) \\ Q_6 &= W_a + W_b - (Q_1 + Q_2 + Q_3 + Q_g) \end{aligned} \tag{1}$$

5.3 Transition Probability Computation

To compute the transition probabilities given a system state, we first use the queue occupancy to determine the set of nodes that are contending for channel access. Since the next state that the system switches to depends on the contention outcomes, we compute the probability that each of the possible contention outcomes occurs. The key to compute these probabilities is to handle hidden terminals, which is described below.

We consider the system state $(Q_1, Q_2, Q_3, Q_g, CW_a, CW_g)$, in which each queue of Fig. 25 has packets to send. This is the state in which the computation of the transition probability is most involved due to the fact that all nodes are contending. We therefore show the computation of the transmission probabilities through this example. For system states in which not all queues have packets to send, the transition probability can be similarly computed.

With all queues backlogged, all three nodes contend for channel access at the next state switching time, in which node $i, (i \in \{a, b, g\})$ transmits a packet (RTS or data packet depending on which hand-shake mechanism is used) with probability $e_i = \frac{2}{CW_i}$. Let f denote the duration of the first packet expressed in the number of mini-slots. The 2nd hop node A successfully transmits a packet only if (1) it 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 does not attempt to transmit in the next f mini-slots. Thus, the successful transmission probability of the 2nd hop node is given by

$$e_a(1 - e_b)(1 - e_g)^f,$$

which is the transition probability from the current state to $(Q_1 - 1, Q_2 + 1, Q_3, Q_g, 0, CW_g)$.

All of the possible next states and their transition probabilities can be computed similarly and are summarized in Table 1. When collision occurs, both the 2nd hop 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 denoted by R_L, it is reset to 0, which explains the modulus operator. When a node with more than 1 non-empty queue wins contention, these queues have equal probability to transmit their head-of-line packet, which explains the division operator.

6. 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.

which link	to state	probability
link 1	$(Q_1-1, Q_2+1, Q_3, Q_g, 0, CW_g)$	$e_a(1-e_b)(1-e_g)^f$
link 2	$(Q_1, Q_2-1, Q_3, Q_g+1, CW_a, CW_g)$	$\frac{(1-e_a)e_b(1-e_g)}{3}$
link 3	$(Q_1, Q_2, Q_3-1, Q_g+1, CW_a, CW_g)$	$\frac{(1-e_a)e_b(1-e_g)}{3}$
link 4	$(Q_1, Q_2, Q_3+1, Q_g-1, CW_a, 0)$	$\frac{(1-e_a)^f(1-e_b)e_g}{2}$
link 5	$(Q_1, Q_2, Q_3, Q_g-1, CW_a, 0)$	$\frac{(1-e_a)^f(1-e_b)e_g}{2}$
link 6	$(Q_1+1, Q_2, Q_3, Q_g, CW_a, CW_g)$	$\frac{(1-e_a)e_b(1-e_g)}{3}$
colliding	$(Q_1, Q_2, Q_3, Q_g, (CW_g+1)\%R_L, (CW_g+1)\%R_L)$	$(1-e_b)(e_a+e_g-e_ae_g-e_a(1-e_g)^f-e_g(1-e_a)^f)$
none	$(Q_1, Q_2, Q_3, Q_g, CW_a, CW_g)$	otherwise

TABLE 1
Some of the transition probabilities of the Markov Model, when one of the queues is empty.

5.4 Throughput Computation

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

$$H = R_L^2 W_a^2 W_b (W_a + W_b). \tag{2}$$

We also compute binary matrix φ_i, for the transmission channel state $i, (1 \leq i \leq 6)$, φ_c for the collision state, and φ_δ for the idle state. These matrices have the same dimension as the transition matrix and can be computed as follows. If the system makes a transition from state i to state j, and in making this transition, a transmission on link 1 occurs, we set $\varphi_1(i,j) = 1$; otherwise, we set $\varphi_1(i,j) = 0$. Similarly, when making this transition, if a collision occurs, we set $\varphi_c(i,j) = 1$. If none of the nodes attempt a new transmission, we set $\varphi_\delta(i,j) = 1$. Let M be the transition matrix. Then the occurrence probability of each channel state can be computed as

$$p_i = \sum(\mathbf{\Pi} \times (\varphi_i \cdot M)), \quad i \in \{1,2,3,4,5,6,c,\delta\}, \tag{3}$$

in which, the operator $\cdot$ denotes inner product, and the operator $\sum$ denotes the operation that adds all elements of a vector. The throughput of the two flows originating from node A and B is then expressed in pkts/s as,

$$\lambda_a = \frac{p_1 T_1}{\triangle}, \lambda_b = \frac{p_3 T_3}{\triangle}, \tag{4}$$

in which $\triangle$ is the average duration of the channel states, computed as the average of the duration of all channel states, weighted by their respective probabilities. Recall that T_1 and T_3 are the duration of *transmission state* 1 and 3, 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.

5.5 Model Validation

We first validate the analytical model of the two hop chain topology using both UrbanMesh and ns-2 simulations. The parameters used in both the model and the simulations are listed in Table 3. Because the six-dimensional Markov chain leads to a large state space as shown by Equation (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 3 packets. To simulate fixed-window congestion control, we set the congestion window of TCP in the simulator to 3. In UrbanMesh, TCP's window is allowed to adapt as normal. We vary the minimum contention window of $CW_{\min}$ of the middle node and evaluate the impact on throughput.

We make the following observations on the results depicted in Fig. 27. First, the model accurately predicts the trends measured via simulation, i.e., the impact of node B's minimum contention window

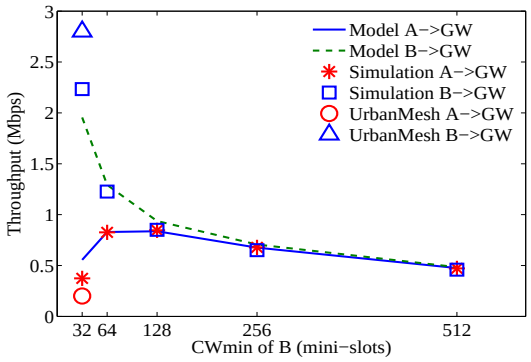

Fig. 27. Analytical model predictions compared to simulation and UrbanMesh.

on both flow's throughput. Second, for $CW_{\min,B} = 32$, the value employed by UrbanMesh, both the model and simulation underestimate the true extent of starvation. Thus, in the actual system, non-modeled factors of TCP such as time out and window dynamics, and non-modeled factors of medium access such as fading channels, have only aggrevated the starvation problem.

5.6 Starvation Solution

The most important observation about Fig. 27 is that it reveals the solution to starvation. 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 un-necessary 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 two flow throughputs is reduced when starvation is removed. This is necessarily the case because the two-hop flow consumes at least twice the resources of a one-hop flow. Consequently, we propose the following policy to counter starvation.

Counter-Starvation Policy: *All nodes that are directly connected to the gateway should increase their minimum contention window to a factor of at least two greater than all other nodes.*

Analysis of the model's state probabilities reveals the effect of the policy on the system queues. Fig. 28 shows that when the minimum contention window of the 1st-hop node B increases, the probability that both Q_1 and Q_g are empty dramatically increases. Recall that Q_1 is the queue at 2nd-hop node A and Q_g is the aggregate queue at gateway node GW. Having both of these queues empty indicates that most packets in the system are queued at B. Since B always contends fairly for the channel due to its ability to sense both A or GW (see Section 3.2), this is the ideal queueing point within the system. Consequently, collisions between A and GW are almost zero. Indeed, the model indicates that with large CW_{min} for the 1st-hop node, A and GW will rarely collide and rarely increases their backoff window.

Thus, the model indicates that the Counter-Starvation Policy results in minimal queueing at the gateway and two-hop node for flows employing a sliding window protocol. Without these queues, the MAC protocol's bi-stable behavior is broken. Without bi-stability, the "penalty to switch states" is very rarely incurred.

6 Evaluation of the Counter-Starvation Policy

In this section, we evaluate our contention window policy's ability to counter starvation. As described in Section 4, the policy sets the minimum contention window of the gateway's immediate neighbors to a value significantly larger than all other nodes. Here, we address issues such as the ideal setting, the impact on downstream traffic, and the effect of gateways with multiple branches. We use a combination of simulations and an on-site deployment termed MirrorMesh.

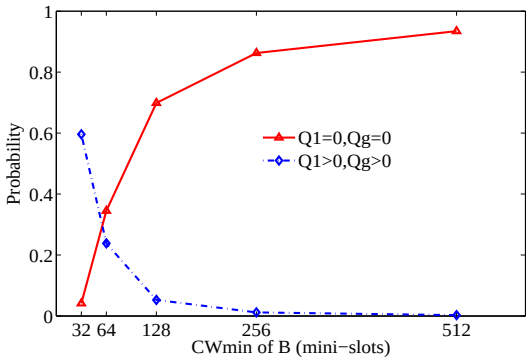

Fig. 28. Queue behavior as a function of the minimum contention window of B.

6.1 MirrorMesh Testbed

To implement our $CW_{\min}$ policy we need to change the minimum contention window of all of the gateway's immediate neighbors. In the current deployment of UrbanMesh, changing CW_{min} is not supported by the SMC wireless cards. As part of our network operations plans, we will be replacing the cards with ones that allow this parameter change.

Here, we describe the results of a set of experimental tests designed to validate the Counter-Starvation Policy in the field. We refer to the platform as MirrorMesh we perform all experiments in the same area as UrbanMesh in order to inherit the UrbanMesh's propagation environment. MirrorMesh nodes are desktop PCs with a Linux Operating System and Atheros wireless card that allows CW_{min} to be changed. Each desktop PC connects to an external omni-directional antenna. The technical details describing our replacement mesh nodes are listed in Table 2. Although different from the UrbanMesh 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 reported in Table 3, except the minimum backoff window, which is 16 by default in Atheros chip set. MirrorMesh contains no user-generated background flows such that the only traffic is that generated by our tests.

Operating system	Linux Red Hat, kernel 2.6
TCP version	TCP/Reno
Wireless Card	Atheros chipset
Wireless Driver	Madwifi v. 0.9.2 (modified)
MAC	IEEE 802.11b
Physical rate	Fixed to 11 Mbps
Physical channel	ch. 3
Mesh Point Connectivity	WDS

TABLE 2
System parameters for the MirrorMesh nodes.

6.2 One and Two Branches via MirrorMesh

Here, we experimentally validate our Counter-Starvation Policy on MirrorMesh. In this set of experiments, we test per-flow throughput and network utilization with various topologies and system parameters, both for the default CW_{min} and for increased CW_{min} as recommended by the Counter-Starvation Policy. Each experiment lasts 120s and the packet size is set to 1500 unless stated otherwise.

6.2.1 Experiments on a Single Branch

We first consider the scenario depicted in Fig. 13, in which nodes A and B both transmit packets to the gateway node, GW. As in UrbanMesh, we first verify that all links are operational and that A and GW are out of range.

RTS/CTS on. In this experiment, we enable RTS/CTS and set CW_{min} of node A, B, and GW to the default value of 16. Fig. 29(left) depicts a severe throughput imbalance and confirms that the system behavior for this scenario is consistent between MirrorMesh and UrbanMesh. Increase CW_{min} to 128 and repeat the experiment. The result is also shown in Fig. 29(left), 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. 29(right) 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 utilization, we count A's throughput twice, because its packets need to traverse two links to and from the gateway node GW.

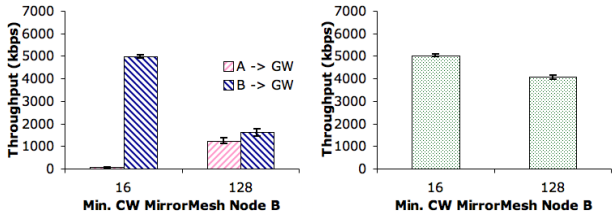

Fig. 29. Starvation with default CW_{min} and Couter-Starvation Policy result in a two-hop chain of MirrorMesh. Aggregate network utilization is shown in the rightmost graph. RTS/CTS mechanism is enabled.

RTS/CTS off. Fig. 30 report 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 4, 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.

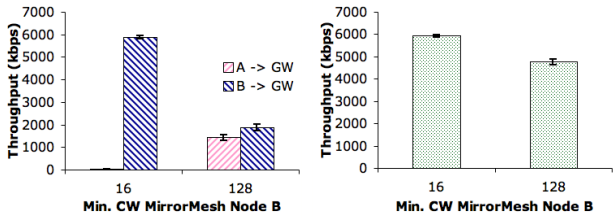

Fig. 30. Starvation with default CW_{min} and Couter-Starvation Policy result in a two-hop chain of MirrorMesh. Aggregate utilization is shown in the rightmost graph. RTS/CTS mechanism is disabled.

Baseline $CW_{min} = 32$. In the above experiments, the baseline CW_{min} is set to 16, the default value for Atheros. However, for most commercial wireless cards, including those deployed in UrbanMesh, CW_{min} is set to 32 as recommended by the IEEE 802.11 standard. Consequently, we evaluate our policy for $CW_{min} = 32$ on MirrorMesh. We first set CW_{min} to 32 for A, B and GW and collect the measurement results. Then we enlarge CW_{min} of node B to 128 and report the result for both cases in Fig. 31. We observe from the two figures that with the default value set according to the IEEE 802.11 recommendation, the nature of the starvation problem remains, yet our solution is equally effective.

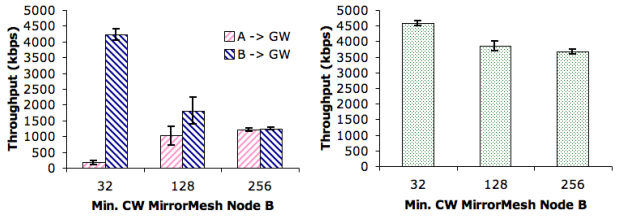

Fig. 31. Starvation and Counter-Starvation Policy result in a two-hop chain of MirrorMesh with the most common default minimum contention window setting for commercial devices (32). Aggregate utilization is shown in the rightmost graph.

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. 32, the mere adoption of small packet sizes does not significantly effect the starvation problem. When RTS/CTS is on, contending packets are RTS packets rather than data packets. When RTS/CTS is disabled, data packets that are usually larger than RTS packets contend for channel access. This results in more severe bi-stability effect than with RTS packets. Our analysis in Section 4 and 5 captures that smaller data packets do not mitigate the starvation problem, and that the Counter-Starvation Policy is equally effective.

6.2.2 Experiments on Two Branches

Fig. 24 in Section 3.3 demonstrated that starvation occurs not only on one branch in a mesh network, but also on two or more branches. In this experiment, we evaluate our solution in the scenario shown in Fig. 23 in which three flows are active on two branches. Fig. 33 reports that flow A is starved, whereas flow B and C almost equally split the bandwidth. We then invoke the Counter-Starvation Policy by increasing CW_{min} for both B and C, both of the gateway's one-hop neighbors. As shown in Fig. 33, with our solution, the throughput of the second hop flow is dramatically improved. This is because with increased CW_{min}, most of the packets of flow $A \rightarrow B \rightarrow GW$ are queued at node B, and therefore contend with node C more fairly.

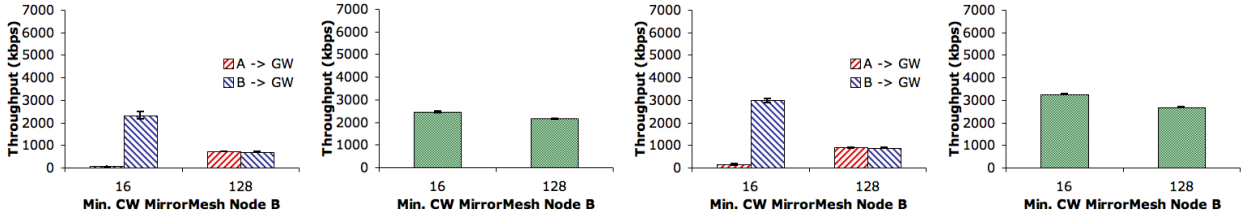

Fig. 32. Starvation and Counter-Starvation Policy result with 500 Byte TCP packets. Aggregate utilization is shown in the rightmost graphs.

6.2.3 Downstream Traffic

Thus far, we have considered upstream data traffic. In Fig. 23, we reverse the direction of the flows such that DATA packets are transmitted from GW to nodes A and B, respectively, and TCP ACKs are transmitted from A and B to GW. In this scenario, both the hidden terminal effect in the MAC layer and the flow loops enforced by the sliding window congestion control remain the same as in uploading scenario. In this experiment, we show the presence of the starvation problem and the effectiveness of our solution for downstream flows.

Fig. 34 shows the throughput A and B receive when CW_{min} is 16 for all three nodes and the throughput of A and B when CW_{min} is set to 32. As predicted, starvation indeed occurs in the download scenario,

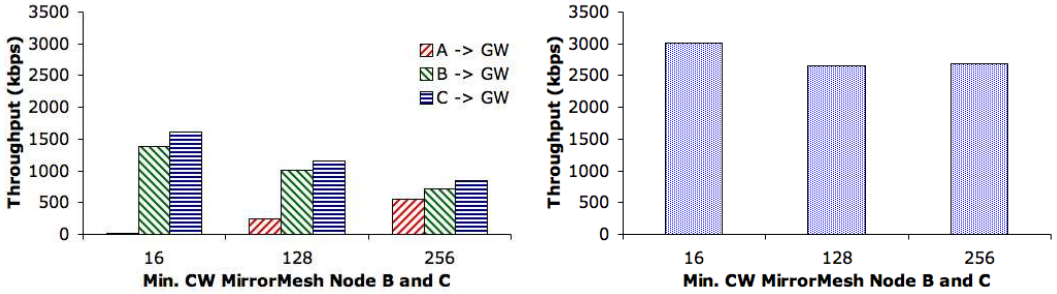

Fig. 33. Starvation in a two-hop chain of MirrorMesh with default contention windows in all nodes. Aggregate utilization is shown in the bottom graph.

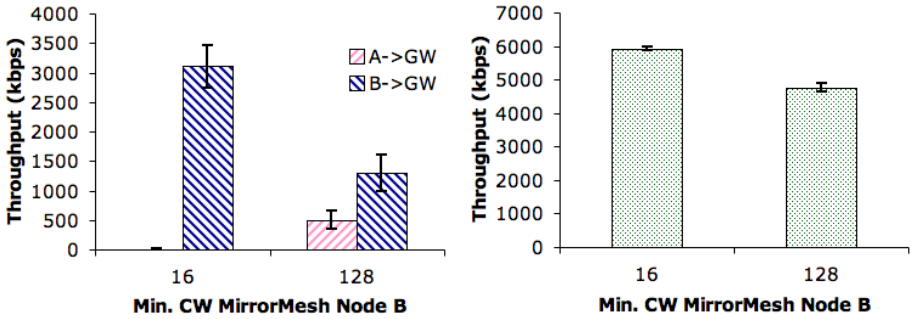

Fig. 34. Starvation and Counter-Starvation Policy result in a downstream two-hop chain of MirrorMesh nodes. Aggregate utilization is shown in the bottom graph.

and our solution allows the two-hop downstream TCP flow to receive significantly higher throughput than the default window.

6.3 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-Reno for congestion control and IEEE 802.11b for medium access control. We use the MAC and physical parameters of Table 3.

Four hop chain topology. In a four-hop chain topology as depicted inFig 35, spatial reuse is possible and there are an increase in the number of nodes out of carrier sense range. In these simulation, we explore whether these factors change the nature of the problem and solution. In this scenario, four mesh nodes simultaneously creat long-lived TCP connections to the gateway.

SIFS	10 μs
DIFS	50 μs
EIFS	364 μs
σ	20 μs
BasicRate	2 Mbps
DataRate	11 Mbps
PLCPRate	1 Mbps
(CW_{min}, CWmax)	(32,1024)
Short Retry Limit	7

TABLE 3
Parameters setting for the MAC and physical layers.

Fig. 36 depicts the simulation results for all nodes having $CW_{min} = 32$ and node 1 (the gateway's one-hop neighbor) having $CW_{min} = 128$ following the Counter-Starvation Policy. We observe that with all nodes having the same minimum contention window, the 1st 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} of the gateway's neighbor to 128, all mesh nodes receive equal throughput.

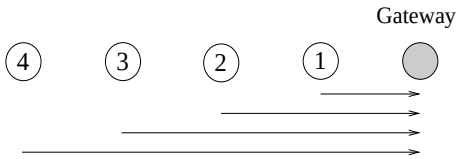

Fig. 35. TCP flows in four-hop chain topology.

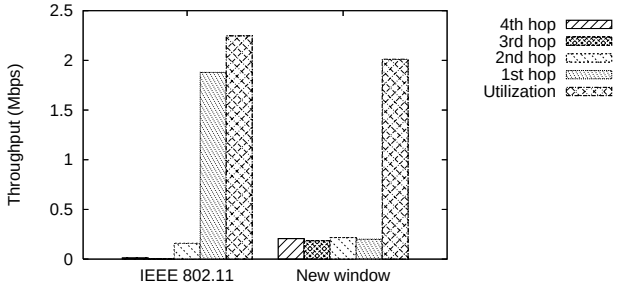

Fig. 36. TCP throughput in a four-hop chain topology, with minimum contention windows all 32 (left) and the one hop neighbor changed to 128 (right).

In a longer chain topology, while 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.

Three-branch tree topology Finally, we consider a scenario in which several branches are connected to the gateway such as depicted in Fig. 37. The key issue is whether the first-hop node on one branch is silenced by the transmissions on other branches, thus leaving more spare capacity to downstream nodes on the same branch to transmit, and subsequently eliminating starvation *without* requiring the Counter-Starvation Policy.

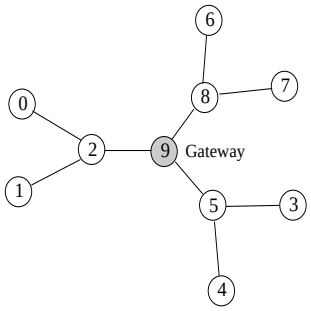

Fig. 37. TCP flows in a tree topology.

We test the scenario in Fig. 37, in which each mesh node is simultaneously transmitting TCP traffic to the gateway. Fig. 38 shows that starvation persists in this scenario, as the three one-hop nodes obtain most

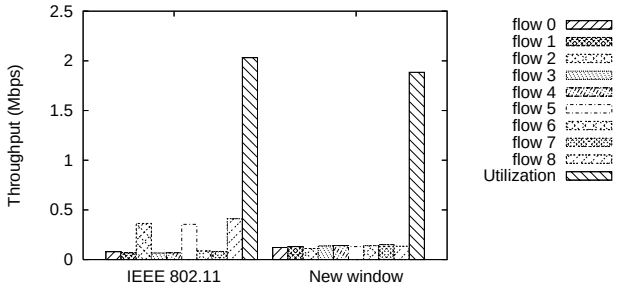

Fig. 38. TCP throughput in three-branch tree topology, with new window 32-128-32.

of the gateway bandwidth, whereas the six two-hop nodes receive significantly reduced throughput. Our Counter-Starvation Policy suggests to increase the minimum backoff window CW_{min} of all three one-hop nodes, and we select a value of 128. We observe in Fig. 38 that with this policy, all flows in the network equally share the gateway bandwidth, albeit with decreased utilization.

In this topology, indeed the one-hop flow on one branch can be silenced by the transmissions on other branches. However, downstream TCP flows on this branch cannot take advantage of this silenced flow, because their DATA transmissions to the gateway still collide with the TCP ACK from the gateway with high probability. Thus the root causes of starvation are not eliminated.

This confirms that the starvation phenomenon exists not only in two-hop chains, but also in a much broader scope beyond a single branch of a tree. Any one-hop mesh node(s) can starve any node(s) that are two or more hops away from the gateway whenever they transmit simultaneously, if no counter-starvation mechanism is present. This simulation also verifies that our solution is effective in a more general topology than a chain.

7 Related Work

There is significant prior work investigating TCP performance over wireless networks including mesh networks, e.g., [5], [14], [16], [26]. However, none of this work demonstrates TCP starvation via measurements, models, nor simulations. Moreover, no prior work exists for a large-scale urban mesh network. Likewise, there are several analytical models of TCP over wireless, e.g., [16], [21], yet, none are able to predict the existence of starvation.

Prior work on starvation focuses on MAC layer modeling without TCP effects [19] and 802.11s-like rate limiting to counter starvation [11], [18]. Moreover, starvation of *two-hop* flows was not identified and no measurements were performed in an operational environment.

Policies for determining the contention window have been studied extensively in the context of providing quality of service, e.g., [10], [13], [24], [27], [28], [30]. However, no prior work recognized the role of the contention window in countering starvation.

A number of techniques have been proposed for joint design of congestion control and medium access, e.g., [9], [12], [26], [31], [32]. In contrast, our approach requires no changes to TCP nor 802.11 and our solution is demonstrated experimentally.

The IEEE 802.11s draft [2] contains a mechanism for hop-by-hop congestion control such that nodes can request other nodes to ramp up or slow down. While this mechanism has the potential to alleviate starvation, the standard does not specify an algorithm (only the message structure), and to the best of our knowledge, no implementation, simulated nor experimental, yet exists.

Finally, there are a number of measurement studies on deployed mesh networks [3], [7], [8], [15]. Such networks differ architecturally from UrbanMesh in that they support a smaller user population with a single-tier architecture combined with the use of multiple residential DSL lines: such an architecture is often referred to as a community mesh network and is characterized by an "organic" topology. In contrast, UrbanMesh employs the two-tier architecture used by commercial deployments, with an access tier to

serve a large user population, and a planned backhaul tier with multiple branches and directional links feeding high-speed gateways. In any case, [7] focuses on single active TCP flows and therefore did not observe starvation.

8 Conclusion

In this paper, we measure starvation in an operational two-tier mesh access network, UrbanMesh, and identify the origins of starvation by isolating potential causes. We show that a one-hop TCP flow interacting with a two-hop TCP flow is sufficient to induce starvation. Furthermore, we describe how starvation's originating factors stem from interaction between the transport layer's congestion control and the MAC layer's collision avoidance to produce effects such as (i) bi-stability in which node pairs alternate between dominating system resources, (ii) multiple interacting congestion control loops with an unequal probability of disrupting the loops, and (iii) a high penalty incurred by the system when switching states of the bi-stable system. 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 implement and empirically validate the solution on MirrorMesh, a network redeployment within the same urban environment. Finally, we extend our validation to larger topologies and traffic matrices via simulation.

References

[1] http://www.stcloud.org/index.asp?NID=402.

[2] IEEE 802.11s draft. 802.11 TGs simple efficient extensible mesh (SEE-Mesh) proposal. January 2006.

[3] D Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11 mesh network. In *Proceedings of ACM SIGCOMM*, Portland, OR, 2004.

[4] E. Altman, C. Barakat, and V.M. Ramos Ramos. Analysis of AIMD protocols over paths with variable delay. In *Proc. IEEE INFOCOM*, Hong Kong, China, March 2004.

[5] G. Anastasi, E. Borgia, M. Conti, and E. Gregori. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of Cluster Computing*, 8(2-3):135–145, July 2005.

[6] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A Media Access Protocol for Wireless LANs. In *Proc. ACM SIGCOMM*, London, UK, 1994.

[7] J. Bicket, S. Biswas, D. Aguayo, and R. Morris. Architecture and evaluation of the MIT Roofnet mesh network. In *Proceedings of ACM MobiCom*, Cologne, Germany, September 2005.

[8] Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. In *Proceedings of HotNets-II*, Cambridge, MA, November 2003.

[9] M. Bottigliengo, C. Casetti, C.F. Chiasserini, and M. Meo. Short-term fairness for TCP flows in 802.11b WLANs. In *Proc. IEEE INFOCOM*, Hong Kong, China, March 2004.

[10] F. Cali, M. Conti, and E. Gregori. Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit. *IEEE/ACM Transactions on Networking*, 8:785–799, 2000.

[11] J. Camp, J. Robinson, C. Steger, and E. Knightly. Measurement-driven deployment of a two-tier urban mesh access network. In *Proceedings of ACM MobiSys*, Uppsala, Sweden, 2006.

[12] L. Chen, SH Low, and JC Doyle. Congestion control and media access control design for ad hoc wireless networks. In *Proceedings of IEEE INFOCOM*, Miami, FL, March 2005.

[13] M. Chen and A. Zakhor. Differentiation mechanisms for IEEE 802.11. In *Proc. IEEE INFOCOM*, Barcelona, Spain, 2001.

[14] M. Chen and A. Zakhor. Flow control over wireless network and application layer implementation. In *Proc. IEEE INFOCOM*, Barcelona, Spain, 2006.

[15] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of ACM MobiCom*, September 2003.

[16] 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, April 2003.

[17] C.L. Fullmer and J.J. Garcia-Luna-Aceves. Solutions to Hidden Terminal Problems in Wireless Networks . In *Proc. ACM SIGCOMM*, Cannes, France, September 1997.

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

[19] M. Garetto, T. Salonidis, and E. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. In *Proc. IEEE INFOCOM*, Barcelona, Spain, 2006.

[20] M. Garetto, J. Shi, and E. Knightly. Modeling media access in embedded two-flow topologies of multi-hop wireless networks. In *Proceedings of ACM MobiCom*, Cologne, Germany, September 2005.

[21] Y. Gong and P. Marbach. Interaction of rate and medium access control in wireless networks: The single cell case. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Florence, Italy, May 2006.

[22] A. Kherani and R. Shorey. Throughput analysis of TCP in multi-hop wireless networks with IEEE 802.11 MAC. In *Proceedings of IEEE Wireless Communications and Networking Conference*, Atlanta, GA, March 2004.

[23] 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, March 2005.

[24] W.K. Kuo and C.C. J. Kuo. Enhanced backoff scheme in CSMA/CA for IEEE 802.11. In *Proceedings of IEEE INFOCOM*, Orlando, FL, October 2003.

[25] J. Kurose and K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, 4th edition, 2007.

[26] K.Xu, M.Gerla, L.Qi, and Y.Shu. Enhancing TCP fairness in ad hoc wireless networks using Neighborhood RED. In *Proceedings of ACM Mobicom*, San Diego, CA, 2003.

[27] A. Nafaa and A. Ksentini an A. Mehaoua. Sliding contention window (SCW): towards backoff range-based service differentiation over IEEE 802.11 wireless LAN networks. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, New Orlean, LA, 2005.

[28] L. Romdhani, Q. Ni, and T. Turletti. Adaptive EDCF: Enhanced service differentiation for IEEE 802.11 wireless ad hoc networks. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, New Orlean, LA, 2003.

[29] G. Sharma, A. Ganesh, and P. Key. Performance analysis of contention based medium access control protocols. In *Proc. IEEE INFOCOM*, Barcelona, Spain, April 2006.

[30] Y. Wang and J.J. Garcia-Luna-Aceves. Enhanced backoff scheme in CSMA/CA for IEEE 802.11. In *IEEE Symposium on Computers and Communications (ISCC)*, Kemer-Antalya, Turkey, June 2003.

[31] H. Wu, Y. Peng, K. Long, S. Cheng, and J. Ma. Performance of reliable transport protocol over IEEE 802.11 Wireless LAN: Analysis and enhancement. In *Proceedings of IEEE INFOCOM*, New York City, NY, 2002.

[32] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. In *Proc. IEEE INFOCOM*, Hong Kong, China, March 2004.