Measurement and Modeling of the Origins of Starvation in Congestion Controlled Mesh Networks

Jingpu Shi, Omer Gurewitz, Vincenzo Mancuso, Joseph Camp, and Edward W. Knightly

***Abstract*—Significant progress has been made in understanding the behavior of TCP and congestion-controlled traffic over multi-hop wireless networks. Despite these advances, however, no prior work identified severe throughput imbalances in the basic scenario of mesh networks, in which one-hop flows contend with two-hop flows for gateway access. In this paper, we demonstrate via real network measurements, test-bed 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 mechanisms 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 behavior, and ensuring that TCP flows obtain a fair share of the gateway bandwidth, irrespective of their spatial locations.**

I. Introduction

Large-scale wireless mesh network deployments are planned and underway in cities across the world. Mesh deployments are expected to provide broadband low-cost mobile access to the Internet. The prevailing architecture for large-scale deployments is a multi-tier 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, 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. 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 multi-radio architectures (see the discussion in Section III). Note that starvation of two-hop flows precludes the use of the mesh architecture, which by definition employs multi-hop paths. Our contributions are as follows.

First, we perform experiments in an operational urban mesh network, Technology For All (TFA).[1] We demonstrate the existence of starvation under saturation conditions and show that only a one-hop TCP flow in competition with a two-hop TCP flow is sufficient to induce starvation.

Second, we describe the protocol origins of starvation as a compounding effect of three factors: *(i)* the medium access protocol induces bi-stability in which pairs of nodes alternate in capturing system resources; *(ii)* despite the inherent symmetry of MAC bi-stability, we show that the transport protocol induces *asymmetry* in the time spent in each state and favors the one-hop flow; and *(iii)* most critically, the multi-hop-flow's transmitter often incurs a high penalty in terms of loss, delay, and consequently, throughput, to re-capture system resources.

Third, we develop an analytical model 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 both end-point and intermediate queues.

The model yields a *Counter-Starvation Policy* in which only the gateway's one-hop neighbors should increase their minimum contention window to a value significantly greater than that of other nodes. This policy does not require any hardware and software modification. Conversely, it can be realized via standard mechanisms as IEEE 802.11e. 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, bi-stability is eliminated such that the subsequent penalties are not incurred.

Finally, we experimentally demonstrate that the Counter-Starvation Policy solves the starvation problem. In particu-

J.Shi, J. Camp and E. Knightly are with Rice University, Houston, TX (http://www.ece.rice.edu/networks). O. Gurewitz is with Department of Communication Systems Engineering, Ben Gurion University of the Negev, Beer Sheva, Israel. V. Mancuso is with DIEET, Università di Palermo, Italy. All authors were with Rice while performing the research for this work.

This research was supported by NSF grants CNS-0331620 and CNS-0325971 and by the Cisco Collaborative Research Initiative.

[1] http://tfa.rice.edu.

lar, we realize this policy by employing the IEEE 802.11e mechanism which allows policy-driven selection of contention windows. We re-deploy 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.

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 topology.

A. The Basic Topology

The basic topology we consider in this paper is shown in Fig. 1, in which two mesh nodes, A and B, are located two and one hop away from the gateway, GW, respectively. Mesh nodes A and GW do not sense each other's transmission, i.e., they are hidden from each other. Both A and B transmit a TCP flow to the gateway node GW. Note that this topology is necessarily embedded in any larger mesh network topology given that mesh networks are defined as multi-hop wireless networks with gateways.

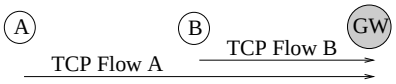

Fig. 1. Traffic matrix in the basic topology

B. Technology For All Network

The Technology For All (TFA) network is an operational mesh network that provides Internet access in a densely populated urban neighborhood in Houston. The network is multi-tier, including a backhaul tier which wirelessly forwards data and an access tier which provides access between end-users and the mesh infrastructure. At the time of our experiments, there were approximately 2,500 users in the network in an area of nearly 3 km^2. Each mesh node runs software derived from open-source LocustWorld mesh networking software. 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. Each wireless card connects to a 15 dBi omni-directional antenna with a vertical beamwidth of 8 degrees. The backhaul antennas are attached to the sides of homes at 10m height, and at slightly greater height (maximum of 20m) at libraries, schools, and businesses.

C. Experimental Setup

In each experiment, we generate TCP traffic using *Iperf* and measure the achieved throughput. Before each experiment, we measure the throughput when each flow 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 is channel 6 of the 2.4 GHZ ISM band. By default, the RTS/CTS mechanism is not used by the TFA mesh nodes. All experiments on TFA take place in the presence of the network's normal user traffic.

D. Measurements In TFA

Here, we experimentally demonstrate the potential for starvation in operational mesh networks under saturation conditions. For our measurements, we select three TFA nodes - A, B and GW - which form a *basic topology* as described in II-A. All of A's packets to and from the gateway GW are forwarded by node B, as verified by observing the routing table. Further, in order to minimize the impact of users' behavior on our measurements, we perform the experiments during the off-peak hour. In particular, 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. Thus, all three nodes mutually contend for channel access in support of both uplink data and downlink acknowledgments.[2]

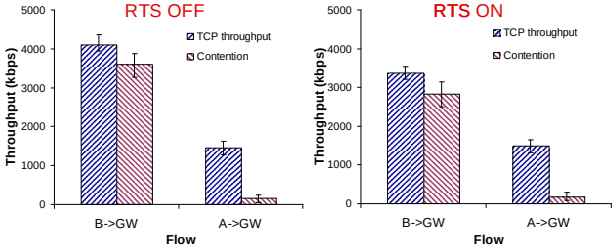

Fig. 2. TCP contention in the basic topology, with and without RTS/CTS.

Fig. 2 depicts the throughput of the two flows with and without contention. It illustrates that, even though 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 active (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 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).

Similar starvation occurs in scenarios characterized by different combinations of user activity and protocol set. Due to space limitation, those experiments are reported in [20].

III. Starvation's Protocol Origins

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

A. Protocol Origins

Medium Access and Bi-stability. The collision avoidance mechanism in CSMA/CA causes bi-stability, in which node pairs (A, B) and (B, GW) alternate in transmission of multiple packet bursts. In particular, the system alternates between

[2]To ensure that our results are not unique to injecting a *single* flow in the presence of many background flows, we also generate *aggregate* flows from both nodes and obtain similar results.

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.

To understand the bi-stability, we first examine the behavior of two flows in the scenario shown in Fig.3, where the gateway GW and two-hop mesh node A contend for transmitting TCP ACK and TCP DATA, respectively.

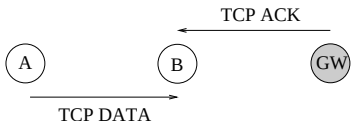

Fig. 3. TCP DATA and TCP ACK are contending for channel access.

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 respectively 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.

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 repeat 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.

To summarize, when mesh node GW (A) wins the channel, it enters a success state in which it transmits a burst of packets, while A (GW) enters a fail state in which it does not succeed in transmitting any packets. The success state can terminate for three reasons: *(i)* the probability of the node with the higher contention window to win is low but not zero; *(ii)* the losing node drops the packet and resets its contention window after it reaches its maximum retry limit; *(iii)* the transmission queue of the winning node is emptied.

Note that since node B is in sensing range with 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 bi-stability is depicted in Fig.4.

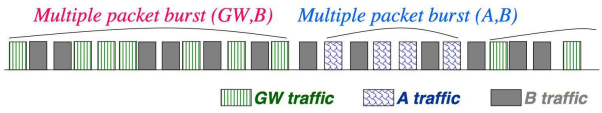

Fig. 4. Illustration of bi-stability with alternation of (A, B) and (B, GW) transmissions.

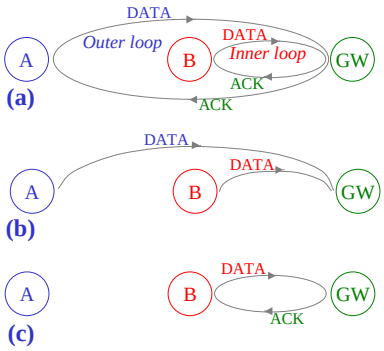

Fig. 5. Illustration of multiple control loops and a shared medium.

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. 5(a). When in the stable state in which (A, B) bursts and GW is in the fail state, both the outer and inner loops are broken (Fig. 5(b)), and hence, (A, B)'s burst length is upper bounded by A's TCP congestion window. Conversely, when (B, GW) bursts, only the outer loop is broken, and the inner loop is self-sustaining due to the loop's own ACK generation (Fig. 5 (c)). Consequently, the duration for GW and B to jointly capture the channel is not bounded. As a result, the system spends much more time in the state in which (B, GW) captures the channel than in the state in which (A, B) captures the channel.

Severe Transition Penalties. Due to asymmetric bi-stable states, node A and node GW experience different fail state durations, leading to a severe penalty only for the TCP flow originating from node A. Recall that a node exits its fail state in the three ways described above. When GW is in the fail state, node A's limited burst is not likely to drive GW to drop a packet. Hence, GW will most likely exit its fail state by case *(iii)*, i.e., the transmission queue of A is emptied. The penalty that node 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 *(ii)*, 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

which exponentially grows with multiple drops of the same TCP segment. This penalty is only paid by TCP Flow A.

B. Broader Topology

A variation of the basic topology is shown in Fig. 6 (left), where A and C respectively transmit a two-hop TCP flow and a one-hop TCP flow to the gateway node GW. 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.

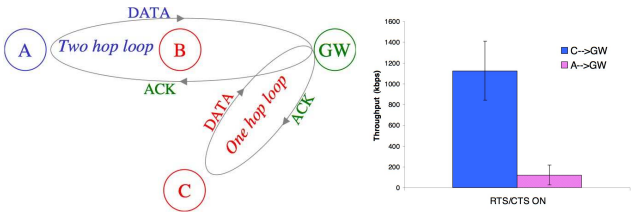

Fig. 6. Two-branch scenario and throughput. $A \rightarrow B \rightarrow GW$ is one branch. $C \rightarrow GW$ is another branch of the mesh.

To verify starvation in the scenario shown in Fig. 6 (left), in TFA we select another one-hop node C besides node A, B and GW. As depicted in Fig. 6 (left), two TCP flows are active on the two branches $A \rightarrow B \rightarrow GW$ and $C \rightarrow GW$, respectively. Fig. 6 (right) 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.6 (left) is necessarily embedded in larger scenarios such as long-chain and broad-tree topology. 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, our extensive experiments demonstrate that starvation occurs in a large set of scenarios, where one-hop flows starve multi-hop flows. The results of these experiments are reported in [20].

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 paper 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

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 counter starvation policy.

A. System Model

As described in Section III, 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. 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 simplified system model with an idealized physical layer in which node pair (GW, B) and node pair (A, B) can communicate without channel errors. We do not consider physical layer capture effect, i.e., we assume that overlapped transmissions fail. We consider 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., [8], [15], [19]. 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.

B. Model Description

As shown in Fig. 7, six sub-flows originating 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, traversing to and from the gateway node, respectively. Correspondingly, we need to track the queue occupancy of the six sub-flows as shown in the figure.

Eight channel states are identified including three *DATA transmission states* occupied by upstream DATA transmissions on links 1, 2, and 3; 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

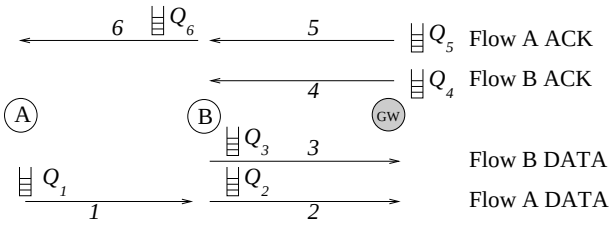

Fig. 7. Queues at different mesh points.

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 back-off counters. These channel states are illustrated in Fig. 8, where the time instants of a possible channel state switch are pointed by arrows placed below the temporal axis. We label a transmission channel 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.

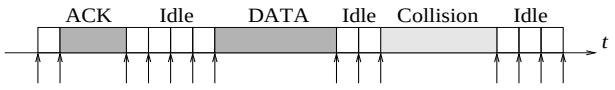

Fig. 8. Illustration of channel states.

The system evolution can be modeled as a Markov chain embedded over continuous time at mini-slot boundaries in which the channel is idle.

We use a, b and g to represent nodes A, B and GW, respectively. For node $i \in \{a, b, g\}$, the success probability 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 as above, the mean backoff interval 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 node GW, and W_a, and W_b be the *fixed congestion window* for flow $A \rightarrow GW$ and flow $B \rightarrow GW$, respectively. Note that W_a and W_b are constant values. Because the middle node is in radio range of the two other nodes, the collision probability between the middle node and one of the other two nodes is very small, compared to the collision probability between A and GW. [3] We therefore assume that the middle node 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 $\mathbf{S} = \{Q_1, Q_2, Q_3, Q_g, \Omega_a, \Omega_g\}$, where Ω_a, Ω_g denote the current backoff stage of node A and GW, respectively. Although the length of some of the queues is not incorporated in the system state, due to the fixed congestion window, they can all be expressed with (Q_1, Q_2, Q_3, Q_g) as follows:

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

[3]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.

C. 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 now consider system state $(Q_1, Q_2, Q_3, Q_g, \Omega_a, \Omega_g)$ in which each queue of Fig. 7 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 $i \in \{a, b, g\}$ attempts to transmit 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 this contending packet expressed in the number of mini-slots. The second 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 second hop node is given by

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

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

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

D. Throughput Computation

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

$$H = R_L^2 (W_a + 1)^2 (W_b + 1)(W_a + W_b + 1). \tag{2}$$

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

TABLE I
TRANSITION PROBABILITIES OF THE MARKOV MODEL, WHEN ALL QUEUES ARE BACKLOGGED.

We now compute binary matrix φ_i for the transmission channel state $i, (1 \le i \le 6)$, 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. Suppose the system makes a transition from system state m to system state n, where m and n are index of the system state. When making this transition, if a successful data transmission on link 1 occurs, we set $\varphi_1(m,n) = 1$; otherwise, we set $\varphi_1(i,j) = 0$. Similarly, when making this transition, if a collision occurs, we set $\varphi_c(m,n) = 1$. If none of the nodes attempts a new transmission, we set $\varphi_\delta(m,n) = 1$. Let M be the $H \times H$ transition matrix. Then the occurrence probability of each channel state can be computed as

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

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,

$$\lambda_a = \frac{p_6 T_6}{\triangle}, \lambda_b = \frac{p_4 T_4}{\triangle}, \qquad (4)$$

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_6 and T_4 are the duration of *transmission state* 6 and 4, respectively. To compute the duration of the collision state, we assume that, on average, the colliding packet starts in the middle of the packet that is transmitted first.

E. Model Evaluation

We now show the results for flow contention in the basic topology as predicted by the analytical model. We then compare the model with NS-2 simulations. We also show the difference that arises when considering an actual TCP implementation in the TFA network.

The parameters used in both the model and the simulations are default parameters of IEEE 802.11b. Because the six-dimensional Markov chain leads to a large state space as shown by Eq.(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. Analogously, we fix the TCP congestion window in the NS-2 simulator to 3. Even though the simulation keeps the behavior of TCP mechanisms such as timeouts and cumulative acks, Fig.9 reports that the throughput of the two flows predicted by the model is close to that obtained from simulation.

Through measurements in TFA, where TCP New Reno adaptive congestion control is used, we evaluate the impact of non-modeled factors of TCP, MAC and PHY on starvation. Fig.9 shows that both the model and simulation underestimate the true extent of starvation. Thus, in the actual system, non-modeled factors of TCP such as timeout and window dynamics, and non-modeled factors of medium access such as fading channels, have only aggravated the starvation problem.

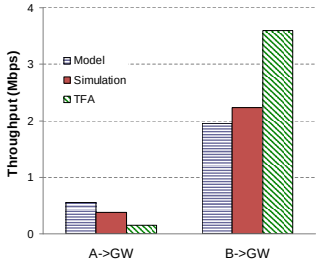

Fig. 9. Analytical model predictions compared to simulation and TFA.

F. Starvation Solution

We now address how to improve fairness in the basic topology. As our above analysis shows, collisions between node A and GW cause the system to spend much more time in one stable state than in the other. Since A and GW cannot sense each other transmission, if they both have packets to transmit at a given time, with high probability their transmissions (either the packet or the RTS, if RTS/CTS is enabled) would overlap in time and a collision would occur. To reduce the probability of collisions between A and GW, we can reduce the backlog of these two end nodes. Considering that GW's load consists of TCP ACKs, this can be done by reducing the rate of packets delivered from B to GW, e.g., by increasing the minimum contention window of node B. Also, with increased minimum contention window, node B contends less aggressively with the two end nodes, therefore the queuing of the system is shifted to B and thus the backlog of the two end nodes is reduced.

Hence, we vary the minimum contention window $CW_{\min}$ of the middle node and evaluate its impact on throughput, via both model and simulations. We observe in Fig.10(a) that the model not only accurately predicts the throughput, but it confirms our above analysis regarding node B's minimum

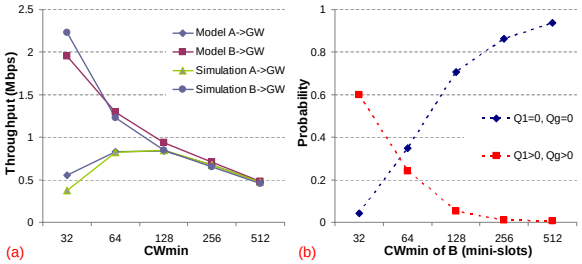

Fig. 10. System behavior vs. the minimum contention window of node B: (a) analytical model predictions compared to simulation; (b) queues behavior.

contention window. In particular, the figure shows that increasing the contention window of node B has the desired effect of removing starvation and indeed providing fairness among the two flows. When the contention window is very high, e.g., 512, fairness is achieved at the 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 throughput of the two flows is reduced when starvation is removed. This is necessarily the case because the two-hop flow consumes twice the resources of a one-hop flow in order to deliver the same amount of throughput. Consequently, we propose the following policy to counter starvation.

Counter-Starvation Policy: *All nodes that are directly connected to the gateway should increase their minimum contention window to a value greater than that of all other nodes.* In practice, the standard-defined binary exponential backoff mechanism yields a minimum congestion window increase of a factor of at least two.

Analysis of the model's state probabilities further reveals the effect of the policy on the system queues. Fig. 10(b) shows that when the minimum contention window of the first-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 second-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 queuing point within the system. Consequently, collisions between A and GW are almost zero. Indeed, the model indicates that with large CW_{min} for the first-hop node, A and GW will rarely collide and rarely increase their backoff window.

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

V. Evaluation of the Counter-Starvation Policy

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

A. MirrorMesh Testbed

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

MirrorMesh nodes are desktop PCs with a Linux Operating System (kernel 2.6) and Atheros wireless card (Madwifi v. 0.9.2 driver) that allows CW_{min} to be changed. Each desktop PC connects to an external omni-directional antenna. Although different from the TFA nodes with respect to the wireless card and Linux kernel, they retain the behavior of network protocols such as TCP. All parameters for MAC and physical layer are according to the IEEE 802.11b standard except the minimum backoff window, which is 16 by default in the Atheros chip set. MirrorMesh contains no user-generated background flows such that all traffic is generated by our tests.

B. Validation For Basic Topology

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

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

RTS/CTS on. In this experiment, we enable RTS/CTS and set CW_{min} of all nodes to the default value of 16. Fig.11 (left) depicts severe throughput imbalance and confirms that the system behavior for this scenario is consistent between MirrorMesh and TFA. We increase CW_{min} of node B to 128 and repeat the experiment. The result is also shown in Fig. 11(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. 11(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 the network utilization, we take into account the fact that some packets need to traverse multiple hops before reaching the gateway. For the scenario depicted in Fig. 1, we count A's throughput twice, because its transmitted packets need to traverse two links which can not be active simultaneously.

RTS/CTS off. Fig. 12 reports results for the case that RTS/CTS is disabled. We consider $CW_{min} = 16$ for all nodes as well as $CW_{min} = 128$ for node B. The results indicate that the Counter-Starvation Policy is equally effective and allows

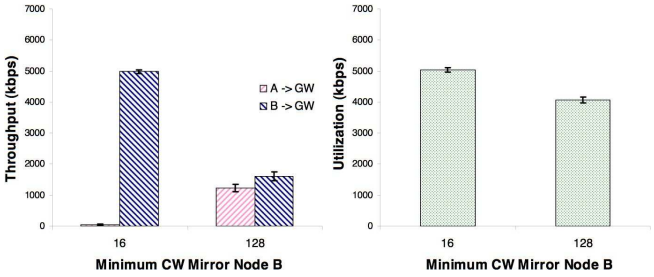

Fig. 11. Starvation with default CW_{min} and Counter-Starvation Policy result in the basic scenario of MirrorMesh. Aggregate network utilization is shown in the rightmost graph. RTS/CTS mechanism is enabled.

equal throughput distribution among the two contending TCP flows, even without RTS/CTS. The reason is that, as discussed in Section IV, our solution results in having all queued packets at B. Consequently, the hidden nodes, A and GW, are not backlogged such that the probability that both A and GW have packets to send simultaneously and collide is negligible, irrespective of the RTS/CTS mechanism.

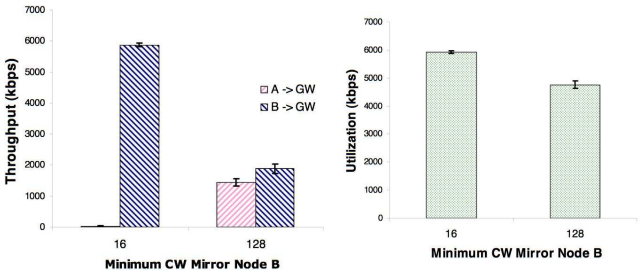

Fig. 12. Starvation with default CW_{min} and Counter-Starvation Policy result in the basic scenario of MirrorMesh. Aggregate utilization is shown in the rightmost graph. RTS/CTS mechanism is disabled.

C. Validation For Two Branches

In Section III, we showed that starvation occurs for flows not only on one branch in a mesh network, but also on two branches. In this experiment, we evaluate our solution in the scenario shown in Fig. 13 in which three flows are active on two branches. Fig. 14 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. 14, 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.

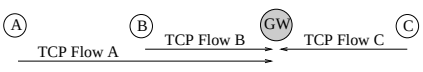

Fig. 13. Two branch topology. TCP flow A is a two-hop flow. TCP flow B and C are two one-hop flows.

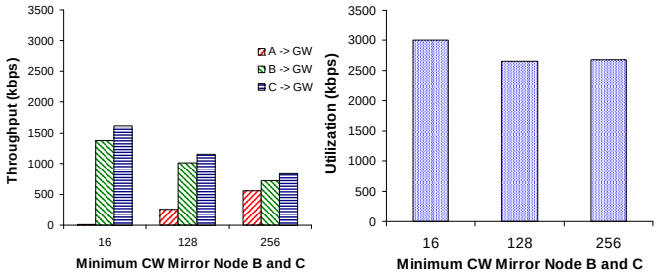

Fig. 14. Starvation and Counter-Starvation Policy in the two branches scenario. Aggregate utilization is shown in the rightmost graph.

D. Other Scenarios

More results for different CW_{min}, longer chain, different packet size, and download traffic are presented in [20]. These results confirm that the starvation phenomenon exists in a much broader scope beyond the basic scenario. In fact, any one-hop mesh node(s) can starve any node(s) that are two or more hops away from the gateway, if no counter-starvation mechanism is present. Furthermore, multiple backlogged one-hop flows can starve two- or more-hop flows, as long as these one-hop flows jointly saturate the channel.

These results also verify that our solution is effective in scenarios beyond the two-hop topology. This is because in these scenarios, nodes farther away from the gateway have less forwarding responsibility and are less loaded. In contrast, nodes that are one and two hops away from the gateway are the bottleneck, because all flows finally converge to the gateway. Thus, the starvation problem in more general scenarios has the same nature as in the basic two-hop scenario, and our solution is just as effective in eliminating starvation.

VI. Related Work

Single-hop flows. We refer to a flow as a single-hop flow if the source of the flow can reach its destination within one hop. Single-hop flows can exist in both single-hop topologies in which all nodes sense each other's transmission and multi-hop topologies in which they do not. Single-hop flow studies showed both analytically as well as by simulation that in a fully backlogged scenario without flow control mechanisms (e.g., UDP traffic), network resources can be shared unevenly between contending flows. It was shown that MAC mechanisms ranging from binary exponential backoff to the use of carrier sense itself can cause unfairness [2], [3], [7], [8]. Moreover, MAC-level solutions to unfairness among single-hop flows have been previously proposed including suggested modifications to exponential backoff [3], [23] and the handshake mechanism [3]. Likewise, in the context of 802.11e (which addresses QoS and service differentiation), some proposals allow different system parameters (Contention Window, $SIFS$ and $DIFS$, etc.) for different traffic classes [17], [18], [21], thereby achieving performance differentiation.

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

Multi-hop flows. Poor performance of multihop TCP flows has been previously established [9], [22]. Furthermore severe unfairness has been observed when multiple TCP flows compete for the same wireless medium [10], [12], [13], [16].

To improve performance of congestion control in multi-hop wireless networks, proposals include hop-by-hop distributed congestion control [1], [24] and joint re-design of congestion control and medium access [4], [6]. Transport-level counter-starvation policies have also been proposed in which the TCP protocol is modified by adaptively slowing down the transmission rate [5], [11], limiting the TCP transmission window [5], [22] or modifying RED [5], [16]. Finally, a simplified model of IEEE 802.11 MAC and TCP features for multi-hop flows can be found in [14], where a single TCP flow is modeled over a two-hop chain assuming that the TCP transmission windows is fixed and neglecting MAC collisions (and hence neglecting binary exponential backoff issues).

Differently from the prior work, this paper shows that it is the sliding window congestion control and IEEE 802.11 MAC that jointly induce unfairness. Even with the TCP window fixed to its optimal value suggested in [5], TCP can still perform poorly and lead to unfairness. In addition, none of these prior works identified nor modeled starvation in the basic topology discussed here, which is the minimum and fundamental topology that inherently exists in mesh networks. Moreover, our counter-starvation policy only modifies basic MAC protocol parameters and does not require any transport, network, nor MAC protocol modifications, nor does it necessitate any control message exchange.

VII. Conclusion

In this paper, we show that a one-hop TCP flow interacting with a two-hop TCP flow is sufficient to induce starvation. We measure starvation in an operational two-tier urban mesh network and describe how starvation's originating factors stem from interaction between the transport layer's congestion control and the MAC layer's collision avoidance. We analytically model the system and utilize the model to devise a simple counter-starvation policy in which nodes one-hop away from the gateway increase their minimum contention window. We finally implement and empirically validate the solution on MirrorMesh, a network re-deployment within the same urban environment.

References

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

[2] G. Anastasi, E. Borgia, M. Conti, and E. Gregori. Ieee 802.11b ad hoc networks: Performance measurements. *Journal of Cluster Computing*, 8(2-3):135–145, July 2005.

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

[4] L. Chen, S. H. Low, and J. C. Doyle. Joint Congestion Control and Media Access Control Design for Ad Hoc Wireless Networks . In *Proc. IEEE INFOCOM*, Miami, FL, USA, March 2005.

[5] 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 *INFOCOM*, San Francisco, CA, April 2003.

[6] 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, USA, 2004.

[7] 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, April 2006.

[8] M. Garetto, J. Shi, and E. Knightly. Modeling Media Access in Embedded Two-Flow Topologies of Multi-hop Wireless Networks. In *Proc. ACM MobiCom*, Cologne, Germany, August 2005.

[9] M. Gerla, K. Tang, and R. Bagrodia. Tcp performance in wireless multi-hop networks. In *WMCSA*, New Orlean, LA, USA, February 1999.

[10] H.Y. Hsieh and R. Sivakumar. Ieee 802.11 over multi-hop wireless networks: Problems and new perspectives. In *VTC (fall)*, Vancouver, BC, Canada, 2002.

[11] T. Jimenez and E. Altman. Novel delayed ACK techniques for improving TCP performance in multihop wireless networks. In *Proceedings of Personal Wireless Communications (PWC'03)*, Venice, Italy, September 2003.

[12] S. Lee K. Xu, S. Bae and M. Gerla. Fair sharing of mac under tcp n wireless ad hoc networks. In *IEEE MMT*, Venice, Italy, October 1999.

[13] S. Lee K. Xu, S. Bae and M. Gerla. Tcp behavior across multihop wireless networks and the wired internet. In *WOWMOM*, Atlanta, GA, USA, 2002.

[14] 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, USA, March 2004.

[15] A. Kumar, E. Altman, D. Miorandi, and M. Goyal. New insights from a fixed point analysis of single cell IEEE 802.11 WLANs. In *Proc. IEEE INFOCOM*, Miami, FL, March 2005.

[16] 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.

[17] A. Nafaa, A. Ksentini, A. Mehaoua, B. Ishibashi, Y. Iraqi, and R. Boutaba. Sliding contention window (SCW): towards backoff range-based service differentiation over IEEE 802.11 wireless LAN networks. *IEEE Network Magazine*, 19(4):45–51, 2005.

[18] L. Romdhani, Q. Ni, and T. Turletti. Adaptive EDCF: Enhanced service differentiation for IEEE 802.11 wireless ad hoc networks. In *WCNC*, New Orlean, LA, USA, 2003.

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

[20] J. Shi, O. Gurewitz, V. Mancuso, J. Camp, and E.W. Knightly. Starvation in operational urban mesh networks: Compounding effects of congestion control and medium access. In *Rice University Technical Report TREE0709*, http://www.ece.rice.edu/∼jingpu/doc/tcpmesh_tr.pdf, June 2007.

[21] V. Siris, , and G. Stamatakis. Optimal cwmin selection for achieving proportional fairness in multi-rate 802.11e wlans: Test-bed implementation and evaluation. Los Angeles, CA, USA, 2006.

[22] K. Sundaresan, V. Anantharaman, H. Hsieh, and R. Sivakumar. ATP: A Reliable Transport Protocol for Ad-hoc Networks . In *Proc. ACM MobiHoc*, Annapolis, MD, USA, June 2003.

[23] Y. Wang and J.J. Garcia-Luna-Aceves. Channel Sharing of Competing Flows in Ad Hoc Networks. In *IEEE Symposium on Computers and Communications (ISCC)*, Kemer-Antalya, Turkey, June 2003.

[24] 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.