Buffer Sizing in 802.11 Wireless Mesh Networks

Kamran Jamshaid*, Basem Shihada*, Li Xia*, and Philip Levis†
* MCSE Division, KAUST, Saudi Arabia. {kamran.jamshaid, basem.shihada, li.xia}@kaust.edu.sa
† Department of Computer Science, Stanford University, USA. pal@cs.stanford.edu

***Abstract*—We analyze the problem of buffer sizing for TCP flows in 802.11-based Wireless Mesh Networks. Our objective is to maintain high network utilization while providing low queueing delays. The problem is complicated by the time-varying capacity of the wireless channel as well as the random access mechanism of 802.11 MAC protocol. While arbitrarily large buffers can maintain high network utilization, this results in large queueing delays. Such delays may affect TCP stability characteristics, and also increase queueing delays for other flows (including real-time flows) sharing the buffer. In this paper we propose sizing link buffers collectively for a set of nodes within mutual interference range called the 'collision domain'. We aim to provide a buffer just large enough to saturate the available capacity of the bottleneck collision domain that limits the carrying capacity of the network. This neighborhood buffer is distributed over multiple nodes that constitute the network bottleneck; a transmission by any of these nodes fully utilizes the available spectral resource for the duration of the transmission. We show that sizing routing buffers collectively for this bottleneck allows us to have small buffers (as low as $2-3$ packets) at individual nodes without any significant loss in network utilization. We propose heuristics to determine these buffer sizes in WMNs. Our results show that we can reduce the end-to-end delays by $6\times$ to $10\times$ at the cost of losing roughly 5% of the network capacity achievable with large buffers.**

***Index Terms*—Wireless Mesh Networks, 802.11, Buffer size, TCP, Throughput, Delay**

I. Introduction

Packet-switched networks use buffers to accommodate transient traffic bursts. These buffers aim to prevent packet loss and maintain high output link utilization. Buffer sizing is an important network configuration parameter: under-buffered networks lead to frequent packet loss and subsequent under-utilization of network resources, while over-buffered networks lead to increased queueing delays.

We wish to study the impact of buffer sizing in 802.11-based Wireless Mesh Networks (WMNs) providing backhaul connectivity for last mile Internet access [1]. These multihop networks have stationary mesh routers affixed to rooftops and building structures, forming a multihop wireless backhaul. Client devices connect to their preferred mesh router via wire or an orthogonal wireless channel. Most of the traffic in this network is directed towards or away from the *gateway* mesh router (GW) that bridges traffic to the wired Internet.

With declining memory chip prices, system architects may be inclined to over-provision routing buffers. This, however, does not necessarily lead to improved network performance. IP networks, in general, should minimize buffering and queueing delays. While this is apparent for real-time flows, it also holds for scenarios where throughput has precedence over delay (*e.g.*, bulk file transfer with TCP). TCP's additive-increase multiplicative-decrease (AIMD) rate control algorithm is deliberately designed to fill any buffer and invoke an occasional packet loss so as to provide feedback to the sender [2]. Large network buffers increase the delay before this congestion information can be communicated to the TCP source. This may have a number of ramifications: first, it can potentially affect the stability characteristics of the TCP stream [3]; and second, it also impacts the delay characteristics for other flows (both real-time and TCP) sharing the buffer with this stream. These well-known problems have again recently been highlighted by the Bufferbloat project [4].

Buffer sizing has been extensively studied for wired networks. A widely used rule-of-thumb is to have buffers larger than the bandwidth-delay product (BDP) of the network [5], *i.e.*, $B \geq RTT \times C$, where C is the *bottleneck* link capacity along the path, and RTT is the effective round-trip propagation delay through the bottleneck link. This buffer size B reflects the number of packets queued at a bottleneck link to keep it fully utilized while a TCP source recovers from a loss-induced window reduction. This rule-of-thumb holds for a single TCP flow in a wired network. Its extension to multiple flows is discussed in Sect. II.

There is limited available work on the impact of buffer size on wireless network performance (see Sect. II for related work). Unsurprisingly, we found a significant disparity in the buffer size used in various research platforms. Many papers using the *ns-2* [6] simulator use a buffer size of 50 packets, the default size for the queue object in ns-2. The legacy open source MadWifi drivers [7] for Atheros chipset use a device driver ring buffer of 200 packets. The new ath5k drivers [8] for the same hardware divide this 200 packet buffer[1] equally among four queues representing the traffic access categories (ACs) as defined in Enhanced Distributed Channel Access (EDCA) mechanism. The ath9k [9] drivers for 802.11n Atheros cards use a buffer of 512 packets, again equally divided among the four ACs. In addition, the Linux network stack introduces other layers of buffers, including the network stack's transmit queue, typically set to a default value of 1000 packets.

There are a number of challenges in adapting the existing research literature on buffer sizing for wired networks into the wireless domain. In a wired network, buffer sizing is usually studied in the context of keeping the bottleneck link fully utilized. Wireless links, however, are just an abstraction

[1]Values corresponding to the 2.6.35 Linux kernel

for shared spectrum between communicating entities and their neighbors, only one of which can successfully transmit at a time. It is not clear how buffers can be associated with a distributed set of nodes, some of which relay traffic for other nodes in the network. Further, the capacity of this shared spectrum is not a known constant, but varies over time due to sporadic noise and interference from other wireless nodes. Finally, for the popular CSMA/CA networks, the packet inter-service time at each transmission hop varies due to the random scheduling by the MAC. This MAC behavior needs to be accounted for while sizing network buffers.

In this work we propose studying the buffer sizing problem in WMNs using the notion of distributed queues spread over a contention neighborhood. For a TCP flow traversing multihop wireless links, the end-to-end rate is determined by the contention neighborhood that allocates the minimum transmission time to member links. For optimal network utilization, this bottleneck contention neighborhood needs to be fully utilized. We thus map the buffer sizing problem as a distributed buffer management problem for nodes sharing the bottleneck contention neighborhood. A packet transmission by *any* of these nodes fully utilizes the bottleneck spectral resource for the duration of this transmission. While individual mesh nodes may always be assigned fixed, small buffers, these become suboptimal when network topology changes (*e.g.*, node addition, removal, etc.). In contrast, formulating this problem as sizing a distributed, neighborhood buffer allows us to fully utilize the available spectral resource without adding to the latency, as if sizing a single node. Using simulations and analysis, we show that our proposed approach allows us to have buffers as small as 2–3 packets at individual nodes. These small buffers reduce the end-to-end delay of a TCP flow by a factor of 6–10 while maintaining close to full network utilization.

Our main contributions in this paper are as follows:

1) We formulate the buffer sizing problem in WMNs as sizing a distributed queue spread over multiple nodes constituting the bottleneck radio neighborhood.
2) Using a simple cost function, we propose mechanisms for sizing the transmission buffer at individual nodes in the bottleneck neighborhood queue.
3) We verify using simulations that our approach achieves close to full network utilization with a significant reduction in the end-to-end delay for a flow.

The remainder of this paper is organized as follows. We first cover the background work, contrasting it with our approach. In Sect. III we describe the notion of a wireless network bottleneck spanning multiple, distributed nodes that limits the end-to-end rate of a multihop flow. In Sect. IV, we formulate a model for sizing the distributed buffer collectively and then allocating it among the nodes constituting the network bottleneck. We then show the efficacy of our model *via* a simulation study. We conclude by observing what issues remain open.

II. Related Work

TCP's window-based congestion control algorithm can trigger a burst of packet transmissions (*e.g.*, on receiving a cumulative ACK or several back-to-back ACKs). Sustained bursts can lead to packet losses with subsequent drop in throughput. TCP pacing [10] addresses this by spacing out the transmission of a `cwnd` worth of packets over the estimated RTT interval, *i.e.*, the packets are injected into the network at a rate `cwnd`/RTT. While this minimizes traffic burstiness (and subsequent queueing delays), it may reduce throughput compared to unmodified TCP because of delayed congestion signals and synchronized losses [11]. ElRakabawy *et al.* [12] observe that the inherent variability in the RTT of multihop wireless flows may offset these synchronization issues. They propose a rate-based transmission algorithm over TCP called TCP with Adaptive Pacing (TCP-AP): instead of spacing the transmission of `cwnd` worth of packets over the RTT of a flow, the transmission rate is computed using 4-hop propagation delay (determined by spatial reuse in a chain of wireless nodes) and coefficient of variation of recent RTTs (to identify incipient congestion along network path). We compare our proposed buffer sizing strategy against TCP-AP in Sect. V.

The BDP rule-of-thumb for buffer sizing has extensively been studied for core Internet routers with a large number of TCP flows. When these flows share a common bottleneck, the window size processes quickly synchronize in lockstep. As a result, the aggregate window size is still a sawtooth pattern, and the BDP guideline for sizing the bottleneck buffer still holds. However, when the flows are desynchronized and the window processes are independent, the buffer size B can be reduced to $B = RTT \times C/\sqrt{N}$ while still achieving near 100% utilization [2] . Enachescu *et al.* [13] suggest that B can further be reduced to $O(log\, W)$, where W is the window size of each flow, resulting in buffer sizes of only $10 - 20$ packets while achieving $85 - 90\%$ of link utilization.

Calculating the BDP of a multihop wireless network is different from a wired network because of coupling between the bandwidth and delay of a wireless link [14]. In long-haul wired links, the transmission delay is small compared to the propagation delay; thus a source can often inject multiple packets back-to-back into the pipe. The same is not true for CSMA/CA-based wireless links; first, the transmitter has to contend for channel access for *each* transmission, and second, the 802.11 transmitter requires an ACK before it can transmit the next frame. As a result, the delay of transmitting a packet is strongly coupled with the link's effective bandwidth. Chen *et al.* [14] show that the BDP of multihop wireless flow is tied to the number of round-trip hops along the path to the destination. They use this BDP value to define the TCP congestion window limit in order to avoid the window overshooting problem that may self-interfere with a flow's transmission.

There is only a limited amount of work on buffer sizing for wireless networks. For single-hop 802.11 WLANs, large Access Point (AP) buffers can improve fairness between upstream and downstream TCP flows [15]; a large buffer increases the queueing delay for TCP ACKs being transmitted back to the upstream wireless source, in effect rate limiting the upstream flows. This scheme, however, disturbs the tight feedback loop necessary for optimal operation of TCP. The AP buffer can be sized dynamically [16] to strike a balance between channel utilization and delay requirements of various flows.

For multihop wireless networks, a number of publications (*e.g.*, [17]) propose using queue sizes for detecting network congestion and enforcing subsequent rate adjustments. Xu *et al.* [18] have extended this notion to a distributed 'neighborhood queue'. Their NRED algorithm improves flow rate fairness by probabilistically dropping packets when the cumulative size of this distributed queue exceeds a threshold. This category of work, however, does not consider sizing link buffers at individual nodes to provide high utilization and low delay for TCP traffic.

Dousse [19] proposes essentially bufferless relay nodes for a MAC protocol that, amongst other characteristics, self-regulates the UDP traffic it injects in the network. We instead consider the 802.11 MAC protocol for WMNs and size the mesh router buffers for bursty TCP traffic.

III. Network Bottleneck and Distributed Buffers

We motivate our discussion by studying the impact of buffer sizing for two variants of a 4-hop TCP flow (Fig. 1) using *ns-2* [6]. A 4-hop flow is amenable to easier analysis, though similar results have been validated across larger topologies. We use TCP NewReno [20] as its behavior is close to the most common TCP variants used today on the Internet. Our results are summarized in Fig. 2 for a buffer size of 50 packets (default ns-2 queue size as used in many publications) at each node with 11 Mb/s wireless links. Fig. 2a shows our results when all nodes are within mutual carrier sense range; Fig. 2b is the corresponding result for a node placement in which node 4 is outside the carrier sense range of nodes 1 and GW. These two cases experience different sources of packet loss: queue overflows in the first topology and collisions in the latter. We further discuss these below.

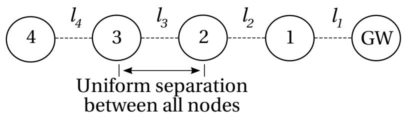


Fig. 1: 4-hop chain. We experiment with two variants: inter-node distance of 130 m. (all nodes within mutual carrier sense range) and 200 m. (node 4 outside the carrier sense range of 1 and GW)

The first subplot in Figs. 2a and 2b shows the congestion window size `cwnd` and RTT for a TCP stream in congestion avoidance phase over the 20–50 s. simulation interval. Fig. 2a experiences queue drop(s) around the 30 s. interval at the node 1 queue and 46 s. interval at the node 3 queue, resulting in reduction of `cwnd` as per TCP's AIMD control algorithm. However, there are no overflow buffers in Fig. 2b (see queue size subplots); instead, all drops in `cwnd` are due to collisions. While 802.11 uses positive acknowledgments with retransmissions, both the 1-hop and 4-hop nodes are outside carrier sense range, resulting in inopportunely scheduled retransmissions that may also collide. These frames are discarded when the retransmission count exceeds the short or large retry count threshold (corresponding to `dot11ShortRetryLimit` and `dot11LongRetryLimit` values of 7 and 4, respectively, per the IEEE 802.11 standard [21]).

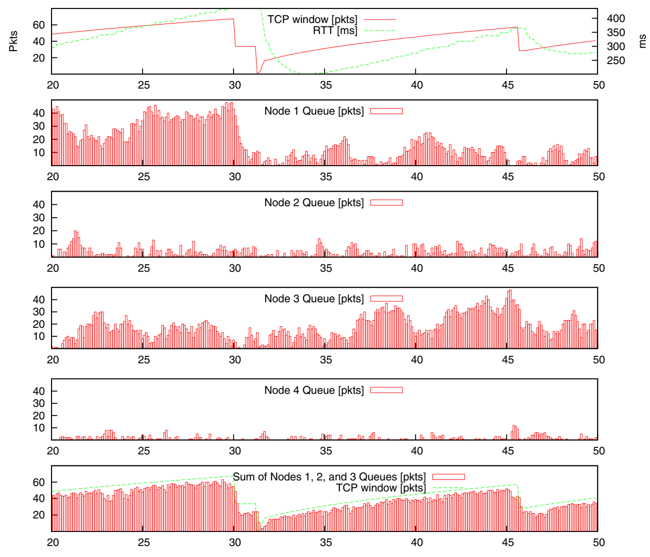
In general, we observe that while queues are more randomly distributed in Fig. 2a, they predominantly build-up on the 1-hop node in Fig. 2b; for the former, the average queue sizes are 16.6, 3.9, 17.6, and 1.4 packets for queues at nodes 1, 2, 3, and 4, respectively; for the latter, these average queue sizes are 14.6, 2.8, 5.2, and 1.9 packets for nodes 1, 2, 3, and 4, respectively. The skewed queueing with 200 m. inter-node separation is an artifact of our topology in which the 1-hop node experiences collisions due to concurrent transmissions with the 4-hop node. The last subplot in Fig. 2 shows the TCP window size against the sum of queue lengths at nodes 1, 2, and 3. It closely mimics the TCP window size, substantiating our view that there is only minimal queueing at the 4-hop node. The average goodput for the two topologies is almost the same (less than 2% difference), but the mean RTT for Fig. 2a is $1.6\times$ that of Fig. 2b because of larger queueing delay.

Based on this and a number of other experiments, we draw the following conclusions: while queueing in a wired network occurs primarily at the buffer interface at the bottleneck link, there is no such 'link' in a wireless network. Instead, the queues in a wireless network build up over a distributed set of nodes that contend for channel access. The queue size is larger for nodes that experience more or asymmetric [1] contention. The radio spectrum around these nodes forms the bottleneck that determines the carrying capacity of a wireless network.

We now first identify these bottlenecks in a multihop wireless network. We then show how distributed buffers are associated with this bottleneck.

A. Bottleneck collision domain

The wireless medium is a shared resource. This limits the set of nodes that can transmit concurrently. We use the notion of collision domain [22] to identify these interfering links. The *collision domain* of link l_i is defined as the set of all links that contend with link l_i. In general, identifying the interfering links is a hard problem, as links exhibit varying degree of interference [23]. Additional mechanisms requiring active or passive monitoring may have to be used [24]. In this work we use the two-hop interference model [25] to identify interfering links; two links interfere if they operate on the same channel and one endpoint of one link is within the transmission range of one endpoint of the other link. The two-hop model approximates the interference avoidance in a 802.11 network with RTS/CTS enabled. While this may not accurately capture all interference relationships in a network, it is sufficient for our study in this paper as our proposed


(a) All nodes within mutual carrier sense range

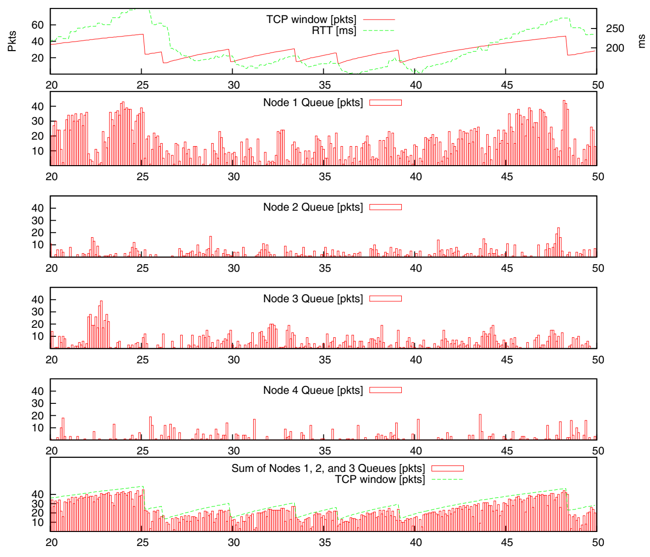

(b) End nodes outside mutual carrier sense range

Fig. 2: TCP congestion window size, RTT, and queue size distribution for a TCP flow in two variants of a 4-hop chain topology with 11 Mb/s wireless links. All nodes have a 50 packet buffer. The reported data is for 20–50 s. of simulation interval.

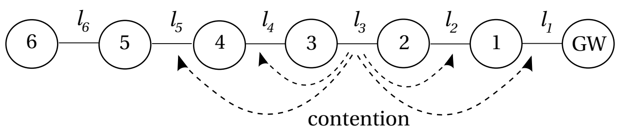


Fig. 3: With a two-hop interference model, the collision domain of link l_3 includes links $l_1, l_2, l_4,$ and l_5.

buffer sizing mechanism is independent of the interference model. In Fig. 3, the collision domain of link l_3 includes links $l_1, l_2, l_4,$ and l_5.

The *utilization* of a collision domain is the sum total of transmission times for all links in a collision domain [26]. The feasibility constraints on scheduling require that this utilization cannot exceed 1. Mathematically, we represent it as follows: Let $R_{(m,n)}$ be the link rate between neighboring nodes (m,n) and let $r_{(m,n)}$ be the traffic carried by this link. Let r_i be the end-to-end rate for flow f_i. Then $r_{(m,n)} = \sum_{i: f_i \text{ traverses } (m,n)} r_i$. Let $\mathbf{C} = \{C_1, C_2, ..., C_j\}$ be the set of j collision domains in this network. Ignoring physical and MAC layer headers, the feasibility constraints require

$$\sum_{\forall\,(m,n)\text{ in }C_p} \frac{r_{(m,n)}}{R_{(m,n)}} \leq 1,\ \forall p \in \{1, 2, ..., j\}$$

A *saturated* collision domain is defined as a collision domain which is fully utilized. A saturated collision domain becomes a *bottleneck* for a flow if that flow has a maximal rate amongst all other flows using this collision domain. A multihop flow may be part of one or more collision domains; its end-to-end rate is then bound by the collision domain that assigns it the lowest rate.

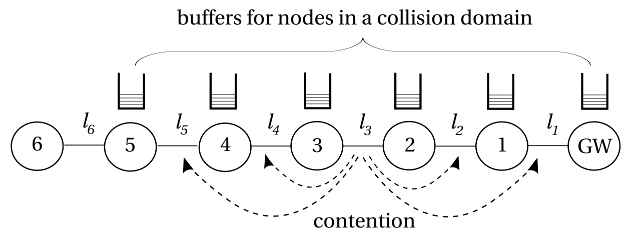


Fig. 4: The distributed neighborhood buffer of a collision domain includes the buffers associated with the constituent nodes in that collision domain.

B. Distributed neighborhood buffers

Since the bottleneck collision domain limits the end-to-end rate of a multihop flow, it is important that the spectral resource in this bottleneck be fully utilized. From a buffering perspective, this is achieved if the nodes in this bottleneck always have packets to send. A packet transmission by any of these nodes fully utilizes the available spectral resource for the duration of the transmission. Thus, instead of considering buffers at individual nodes, we propose sizing the collective buffer of the bottleneck collision domain so as to saturate its available spectral resource. This neighborhood buffer is distributed over the set of nodes constituting the bottleneck. Its size is the sum of the backlog at all the constituent nodes. Similarly, its packet arrival rate is the sum of the arrival rates at individual nodes. However, this neighborhood buffer does not exhibit the FIFO characteristics due to distributed queues and stochastic scheduling of 802.11 MAC protocol. In Fig. 4, we show the distributed FIFO buffer at nodes constituting the bottleneck collision domain for a 6-hop chain topology.

We note that our definition of distributed FIFO buffers includes all packets queued at a node. For Fig. 4 this includes, *e.g.*, the packets queued at node 5 for transmission to node 6 via link l_6. This can be mitigated by maintaining a *virtual* per-link queue at each node, and including only those virtual queues in the distributed buffer that directly correspond to the links in the collision domain. In this current work, however, we include the entire FIFO buffer at a node in the distributed queue. This allows node 6 to transmit to 5 (but not vice versa) when there is an active transmission on link l_3. Limiting node 5's transmission to 6 over-estimates the impact of interference when 3 transmits to 2, but may be necessary to prevent interference at 3 when it receives a packet from 2 (as interference range is typically larger than transmission range, a concurrent transmission from node 5 may corrupt reception at 3).

One simple (though naive) way to enhance the utilization of bottleneck spectrum is to make the bottleneck neighborhood buffer arbitrarily large, and then hope that it will always have sufficient packets in queue to saturate the bottleneck even when the TCP source recovers from a packet loss. However, it is easy to visualize that if this buffer is any larger than the rate supported by the network, it will only add to the queueing delay experienced by the flow. At the same time, if the bottleneck is under-buffered, it will not always be fully utilized. Our challenge, thus, is to determine the size of this neighborhood buffer that maintains the bottleneck near 100% utilization while minimizing the queueing delay.

IV. System Model

Consider a multihop wireless network with N nodes. Assume there are M nodes in the bottleneck collision domain, where $M \in N$, *i.e.*, a node in a bottleneck collision domain contends with $M-1$ other nodes for channel access. The node n_i has a buffer of size b_i packets. Let $B = \sum_{\forall\, i \text{ in } M} b_i$ be the cumulative distributed buffer for this collision domain.

In our analysis below, we first determine B, and then propose a model for distributing it as b_i units amongst the M contending nodes.

A. Neighborhood buffer size B

Consider a single multihop TCP stream. Assume our wireless network can support this stream at a maximum rate of λ packets/s.[2] Let the stream be in AIMD congestion avoidance phase. The TCP window size reaches a value of W_{max} packets before experiencing a loss. As a result, the sender halves the window size to $W_{max}/2$ packets. Since the window size limits the number of unacknowledged packets in flight, the sender, on average, waits for a time interval $\frac{W_{max}/2}{\lambda}$ before starting to retransmit. At the same time, the distributed buffer B takes B/λ s. to drain. For full utilization of the bottleneck spectrum, the source should start retransmitting before the buffer is fully drained, *i.e.*, $\frac{W_{max}/2}{\lambda} < B/\lambda$, or

$$B > \frac{W_{max}}{2} \tag{1}$$

[2]We use packets instead of bits for ease of exposition.

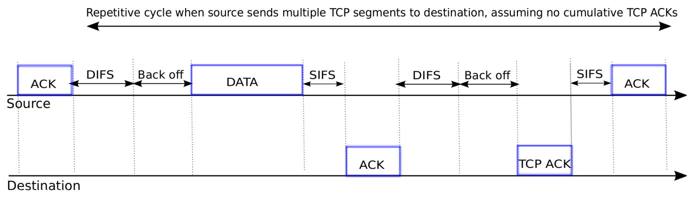

Repetitive cycle when source sends multiple TCP segments to destination, assuming no cumulative TCP ACKs
ACK
DIFS
Back off
DATA
SIFS
DIFS
Back off
SIFS
ACK
Source
ACK
TCP ACK
Destination

Fig. 5: 802.11 MAC overhead per TCP segment transmission.

When the TCP source starts retransmitting, its rate (*i.e.*, cwnd/RTT) should be higher than λ, as otherwise the network capacity is under-utilized. Thus $\frac{W_{max}/2}{RTT} \geq \lambda$, or,

$$\frac{W_{max}}{2} \geq RTT \cdot \lambda \tag{2}$$

From Eq. (1) and (2),

$$B \geq RTT \cdot \lambda \tag{3}$$

Eq. (3) is similar to that of wired networks [2]. The main difference is in λ, the maximum carrying capacity of the network. In wired networks, λ is determined by the bottleneck link capacity. In multihop wireless networks, it is limited by the bottleneck collision domain. The exact values of λ and RTT depend on the network topology and wireless channel characteristics. Per Eq. (3), however, we size our distributed buffer B higher than these instantaneously precise values. This is necessary to account for the time-varying capacity of the wireless channel, and non-deterministic 802.11 Distributed Coordination Function (DCF) scheduling. In this work we estimate loose upper bounds on λ and RTT for a given network. The resulting larger-than optimal buffer trades-off queueing delay for higher channel utilization. Our results in Sect. V confirm that the impact of this trade-off is minimal, and we can still maintain buffers as small as 2-3 packets at most mesh nodes.

For a given flow, the upper bound on λ is obtained when there is no competing traffic. Let the wireless link rates for the M nodes in the collision domain be represented by the vector $\mathbf{W} = \{w_1, w_2, ..., w_M\}$. The bottleneck bandwidth of the path is determined by w_{min} = min$(w_1, w_2, ..., w_M)$. We use $\lambda = w_{min}$ as a loose upper bound on the network carrying capacity.

The RTT in Eq. (3) should only include the propagation delay (and not the queueing delays) for the wireless links. The propagation delays are dominated by transmission delays in wireless networks. To compute this, consider the MAC overhead for transmitting a TCP segment and its associated ACK in Fig. 5. Let T_{d-DATA} and T_{d-ACK} represent the *total* transmission time for TCP segment and ACK, respectively.

$$\begin{aligned} T_{d-DATA} &= T_{BO} + T_{DIFS} + T_{DATA} + T_{SIFS} + T_{ACK} \\ T_{d-ACK} &= T_{DIFS} + T_{TCP-ACK} + T_{SIFS} + T_{ACK} \end{aligned}$$

where T_{BO} is the backoff interval. The average T_{BO}, $\overline{T_{BO}}$ = $CW \times T_{slot}/2$, where CW is the MAC contention window and T_{slot} is the slot duration. For the first transmission attempt, CW is randomly chosen between [0, CW_{min}].

Parameter	Value
T_{slot}	20 μs
T_{SIFS}	10 μs
T_{DIFS}	50 μs
CW_{min}	31
CW_{max}	1023

TABLE I: System parameters of IEEE 802.11b [21].

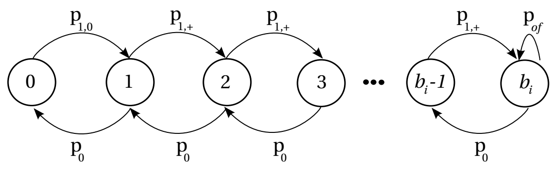


Fig. 6: Queue occupancy state transition for a wireless node n_i with a buffer of size b_i.

CW increases exponentially on a transmission failure. At the n^{th} transmission attempt, CW is randomly chosen between [0, min$\{2^n(CW_{min}+1)-1, CW_{max}\}$]. T_{SIFS} and T_{DIFS} are the SIFS and DIFS deferral, respectively. T_{DATA}, $T_{TCP-ACK}$, and T_{ACK} are the transmission times to transmit a TCP segment, ACK, and MAC-level ACK, respectively. We list the various 802.11b system parameters in Table I. For transmitting a TCP segment of size 1460 B and its ACK (per the exchange shown in Fig. 5), it takes approximately 15 ms at 1 Mb/s link rate and 2.7 ms at 11 Mb/s link rate.

RTT through the bottleneck collision domain can then be computed as follows:

$$\begin{aligned} RTT &= \sum_{i=1}^{M} T_{d-DATA} + T_{d-ACK} \\ &= M \cdot (T_{d-DATA} + T_{d-ACK}) \end{aligned}$$

B. Distributing the neighborhood buffer amongst nodes

Once the neighborhood buffer size B is computed, we distribute it amongst the set of nodes in the bottleneck collision domain. One simple strategy is to divide B equally amongst the nodes. However, this does not consider the fact that a queue drop closer to the source has consumed fewer network resources than a queue drop near the destination. In our model we introduce a cost function to capture this bias.

Fig. 6 shows the queue occupancy state transition for a mesh node. A node n_i can queue at most b_i packets corresponding to its allocated buffer. Let π_k represent the steady state probability that the node queue size is k, for $0 \leq k \leq b_i$. In particular, π_0 and π_{b_i} represent the probabilities that the buffer is empty or full, respectively. The node successfully transmits a packet with probability p_0, transitioning to a state with one less packet queued. With probability $p_{1,+}$, a relay node receives a packet from its parent in the routing tree, and then queues it locally for transmission to the next hop. The transition probability $p_{1,0}$ is a special case for packet reception when the current queue size is zero. Finally, probability p_{of} represents a packet drop due to buffer overflow.

A node can transmit a packet with probability 1 if all contending nodes have an empty buffer (*i.e.*, they are in state π_0); else it contends fairly for the channel with other nodes that also have packet(s) buffered for transmission. Thus,

$$\begin{aligned} p_0 &= 1 \cdot \pi_0^{M-1} + \frac{1}{2} \cdot \binom{M-1}{1} \cdot \pi_0^{M-2}(1-\pi_0) \\ &+ \frac{1}{3} \cdot \binom{M-1}{2} \cdot \pi_0^{M-3}(1-\pi_0)^2 + \ldots + \\ &+ \frac{1}{M} \cdot \binom{M-1}{M-1} \cdot (1-\pi_0)^{M-1} \\ &= \sum_{i=1}^{M} \frac{1}{i} \cdot \binom{M-1}{i-1} \cdot \pi_0^{M-i} \cdot (1-\pi_0)^{i-1} \qquad (4) \end{aligned}$$

The probability of receiving a packet from a parent is 0 if the parent buffer is empty. If it is not empty (*i.e.*, in state $1-\pi_0$), the parent contends fairly with the other nodes that also have packet(s) buffered for transmission.

$$\begin{aligned} p_{1,0} &= 0 \cdot \pi_0 + (1-\pi_0)\Big[1 \cdot \pi_0^{M-2} + \frac{1}{2} \cdot \binom{M-2}{1} \cdot \pi_0^{M-3}(1-\pi_0) \\ &+ \frac{1}{3} \cdot \binom{M-2}{2} \cdot \pi_0^{M-2}(1-\pi_0)^2 + \ldots + \\ &+ \frac{1}{M-1} \cdot \binom{M-2}{M-2} \cdot (1-\pi_0)^{M-2}\Big] \\ &= (1-\pi_0) \sum_{i=1}^{M-1} \frac{1}{i} \cdot \binom{M-2}{i-1} \cdot \pi_0^{M-1-i} \cdot (1-\pi_0)^{i-1} \qquad (5) \end{aligned}$$

If the child queue is not empty, it always contends for transmission with its parent. Thus $p_{1,+}$ is computed as follows:

$$\begin{aligned} p_{1,+} &= 0 \cdot \pi_0 + (1-\pi_0)\Big[\frac{1}{2} \cdot \pi_0^{M-2} + \frac{1}{3} \cdot \binom{M-2}{1} \cdot \pi_0^{M-3}(1-\pi_0) \\ &+ \frac{1}{4} \cdot \binom{M-2}{2} \cdot \pi_0^{M-4}(1-\pi_0)^2 + \ldots + \\ &+ \frac{1}{M} \cdot \binom{M-2}{M-2} \cdot (1-\pi_0)^{M-2}\Big] \\ &= (1-\pi_0) \sum_{i=1}^{M-1} \frac{1}{i+1} \cdot \binom{M-2}{i-1} \cdot \pi_0^{M-1-i}(1-\pi_0)^{i-1} \qquad (6) \end{aligned}$$

The buffer at a node overflows when a node in state π_{b_i} receives a packet. Thus,

$$p_{of} = \pi_{b_i} \cdot p_{1,+} \qquad (7)$$

The cost associated with this drop due to network resources already consumed by this packet increase with the hop distance of n_i from the source along the routing path, *i.e.*, packet dropped closer to source wastes fewer network resources compared to a packet dropped closer to destination. Assume the nodes n_i are ordered such that they represent an increasing hop distance from the source, *i.e.*, the hop count from the source for $n_i < n_{i+1}$. We can thus use the index i for $i = \{1, 2, ..., M\}$ as a simple cost function to represent the bias associated with a packet drop at node n_i.

Thus our optimization problem is as follows:

$$\begin{aligned} \min_{b_i} \quad & \sum_{i=1}^{M} \pi_{b_i} \cdot p_{1,+} \cdot i \qquad (8) \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{M} b_i = B \\ \text{and} \quad & b_i \geq 0, \forall i \in M \end{aligned}$$

The steady state flow balance leads to the following:

$$\begin{aligned} \pi_0 \cdot p_{1,0} &= \pi_1 \cdot p_0 \\ \pi_1 \cdot p_{1,+} &= \pi_2 \cdot p_0 \\ &\cdots \\ \pi_{b_{i-1}} \cdot p_{1,+} &= \pi_{b_i} \cdot p_0 \end{aligned} \qquad (9)$$

Therefore, we get

$$\pi_k = \pi_0 \left(\frac{p_{1,+}}{p_0}\right)^{k-1} \cdot \frac{p_{1,0}}{p_0}$$

when $k > 0$. Substituting this in $\sum_{k=0}^{b_i} \pi_k = 1$, we can calculate π_0 since $p_{1,0}, p_{1,+},$ and p_0 can also be represented by π_0.

The above analysis is based on the assumption that all nodes experience identical transition probabilities. This assumption holds, irrespective of the buffer b_i allocated to node n_i, as long as the probability of a node buffer being empty is small, *i.e.*, $\pi_0 \approx 0$. We can see that this allows p_0, $p_{1,0}$, and $p_{1,+}$ from Eqs. (4), (5), and (6) to converge to $\frac{1}{M}$ (for simplicity, we approximate $p_{1,0}$ from $\frac{1}{M-1}$ as $\frac{1}{M}$). This behavior is consistent with the 802.11 MAC that provides equal transmission opportunities (TXOPs) to all nodes in a collision domain.

From Eq. (9) we observe that $\pi_{b_i} = \frac{1}{b_i+1}$. Substituting into Eq. (7),

$$p_{of} = \pi_{b_i} \cdot p_{1,+} = \frac{1}{b_i+1} \cdot \frac{1}{M} \qquad (10)$$

Our optimization problem then becomes

$$\begin{aligned} \min_{b_i} \quad & \sum_{i=1}^{M} \frac{1}{b_i+1} \cdot \frac{1}{M} \cdot i \qquad (11) \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{M} b_i = B \\ \text{and} \quad & b_i \geq 0, \forall i \in M \end{aligned}$$

The objective function in (11) can be expanded as follows:

$$\frac{1}{M} \min_{b_1,b_2,\ldots,b_M} \left(\frac{1}{b_1+1} + \frac{2}{b_2+1} + \ldots + \frac{M}{b_M+1}\right)$$

Let

$$f \triangleq \left(\frac{1}{b_1+1} + \frac{2}{b_2+1} + \ldots + \frac{M}{b_M+1}\right) \qquad (12)$$

From analytical algebra, the optimal b_i satisfies

$$\frac{\partial f}{\partial b_i} = 0, \forall i \in M, \qquad (13)$$

or the optimal b_i are the boundary values, *i.e.*, $b_i = 0$ or B. Using substitution, we can verify that the boundary values do not minimize the cost function f. We then turn to Eq. (13). This is a set of M linear equations with M unknowns whose solution is as follows:

$$\begin{aligned} b_1 &= \frac{B+5}{1+\sqrt{2}+\sqrt{3}+\ldots+\sqrt{M}} - 1 \\ b_2 &= \frac{\sqrt{2}(B+5)}{1+\sqrt{2}+\sqrt{3}+\ldots+\sqrt{M}} - 1 \\ &\cdots \\ b_M &= \frac{\sqrt{M}(B+5)}{1+\sqrt{2}+\sqrt{3}+\ldots+\sqrt{M}} - 1 \end{aligned} \qquad (14)$$

When B is large, *i.e.*, $B >> 1$, we can approximate this as:

$$b_1 : b_2 : \cdots : b_M = 1 : \sqrt{2} : \ldots : \sqrt{M} \qquad (15)$$

We note that the results derived in Eq. (15) hold for any cost function that increases linearly with the hop count from the source.

Scheme	Goodput Kb/s	Mean RTT (Std. Dev.) ms
50 pkts buffer	930	316 (68)
TCP-AP	850	15 (0.5)
Buffer per Eq. (3) & (15)	878	22 (2)

TABLE II: Performance comparison of a 4-hop TCP flow with 11 Mb/s wireless links. Network layout corresponds to Fig. 2a

V. Performance Analysis

We benchmark our performance results against the following: (1) A 50-packets buffer at all mesh routers. This is the default ns-2 configuration commonly used in research publications; (2) TCP-AP [12], a TCP pacing mechanism for multihop wireless networks (see Sect. II for a brief description of TCP-AP). Pacing TCP traffic minimizes bursts that lead to large queueing delays. TCP-AP paces its traffic using 4-hop propagation delay, thus minimizing queueing along the interfering links. Consistent with [12], we use a 50-packets buffer for TCP-AP, though our results suggest that queues hardly build up beyond a few packets. We simulate backlogged TCP traffic with a large file transfer. The TCP segment size used in our simulations is 1460 bytes. Wireless link rates are 11 Mb/s, unless specified otherwise.

A. Single flow topologies

We first analyze the 4-hop chain topology from Fig. 2a. With 11 Mb/s wireless links, the neighborhood buffer size B computed from Eq. (3) is approximately 9 packets. Per Eq. (15), we distribute it as buffer of size 1, 2, 3, and 3 packets at 1, 2, 3, and 4-hop nodes from the source, respectively. Our results are shown in Table II. The CDF plots for the delay distribution are shown in Fig. 7.

Our results confirm that the 50-packet buffer yields the highest aggregate throughput, albeit with large RTT delays. This may be immaterial for large file downloads where the objective is to maximize throughput. Regardless, we believe that it is essential for a network to be able to provide low-delay services, especially for real-time flows. TCP-AP achieves the lowest delay amongst the three schemes: its RTT is $1/20^{th}$ that of the base case of TCP with 50-packet buffers, while it registers a 9% drop in goodput. In general, however, there

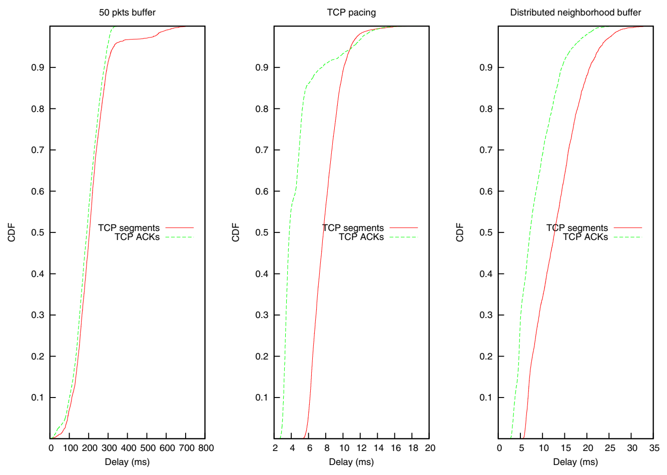


Fig. 7: CDF for delay distribution for the topology in Fig. 2a

Scheme	Goodput Kb/s	Mean RTT (Std. Dev.) ms
50 pkts buffer	910	200 (49)
TCP-AP	825	15 (0.5)
Buffer per Eq. (3) & (15)	830	23 (1.3)

TABLE III: Performance comparison of a 4-hop TCP flow with 11 Mb/s wireless links. Network layout corresponds to Fig. 2b

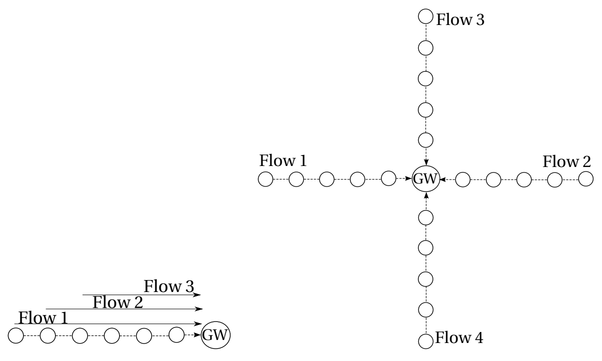
are practical limitations of deploying pacing in real networks, considering the large installed base of TCP in millions of client devices. Our proposed buffer sizing strategy lowers the base case RTT by a factor of 15, at the cost of about 6% drop in throughput. While the performance is comparable to TCP-AP, buffer sizing has a clear advantage from a deployment perspective: WMN service providers can easily configure the buffers on their mesh routers, but they have little or no control on the client devices.

We next consider the 4-hop flow corresponding to the topology in Fig. 2b. Here the losses are dominated by collisions due to asymmetric information about the channel state at different nodes [1]. Our results are tabulated in Table III. In general, all schemes experience a drop in goodput because of multiple transmissions required to resolve collisions between nodes that cannot carrier sense each others transmissions. As described in Sect. III, some of these collisions cannot be resolved by retransmissions, requiring the source to retransmit the packet and drop its `cwnd`. A smaller `cwnd` lowers the RTT by 50% even for our base case with a 50-packet buffer compared to Table II. Compared to the base case, our proposed buffer sizes improve RTT by a factor of 9, at the cost of 10% drop in goodput.

We have validated our results via simulations on multiple other topologies by varying the flow hop count, including topologies where packet losses occur due to both buffer overflows as well as collisions. In all simulations, the base case with the 50 packet buffer yielded the highest goodput, while TCP-AP yielded the least delay. Therefore, we normalize the goodput and RTT for a given simulation run to these values,

Scheme	Normalized Goodput	Normalized RTT
50 pkts buffer	1	20.3
TCP-AP	0.90	1
Buffer per Eq. (3) & (15)	0.96	2.2

TABLE IV: Performance statistics averaged over multiple topologies with varying hop count. Goodput results for a given simulation were normalized to the goodput achieved with the base case for a 50 packet buffer for that simulation. RTT results were normalized to the RTT measured with TCP pacing for that simulation. We present averages computed over multiple different topologies.



(a) Parking lot topology (b) Cross topology

Fig. 8: Multi-flow topologies

and then average those up over different topologies. Our results are tabulated in Table IV. In general, our proposed buffer sizing strategy achieves about 96% goodput of the base case with a 50 packet buffer, while reducing the RTT by a factor of 10.

B. Multi-flow topologies

Our problem formulation in Sect. IV considers a single TCP stream. We now investigate if the underlying behavior is also sustainable in a multi-flow network. Our motive here is not to determine the optimal cumulative neighborhood buffer in multi-flow environments (we defer this to future work); instead, we simply wish to evaluate whether small buffers work well in the presence of multiple flows. We consider two illustrative scenarios: (1) a parking lot topology where multiple flows pass through shared relay nodes, and (2) a cross topology when multiple flows share the bottleneck spectrum but not the relay nodes. We perform experiments in which each flow represents either (i) a large file transfer with TCP, or (ii) a bundle containing a large file transfer with TCP *and* a VoIP stream. We simulate VoIP streams with G.729 codec characteristics using constant bit rate (CBR) UDP traffic [27]: a stream bit rate of 8 Kb/s with a packet size of 40 bytes each. For toll quality service, VoIP streams should maintain a one-way latency (mouth to ear) of less than 150 ms and a delay jitter of less than 30 ms [28].

Parking lot topology We first experiment with the parking lot topology in Fig. 8a. It has data flows originating from nodes 4, 5, and 6-hop away from the gateway. These flows share a common bottleneck collision domain. Considering each flow in isolation, our buffer sizing strategy in Sect. IV can be used to compute a neighborhood buffer size B_1, B_2, and B_3 corresponding to each flow. We then distribute these buffers such that a mesh node in the collision domain has a *per-flow* buffer of size determined by Eq. (15). Per-flow queueing requires flow classification at relay nodes; we perform it using source node IP address and port numbers. We service these queues using fair queueing [29] on top of DCF scheduling. Our methodology considerably overestimates the cumulative buffer required for saturating the bottleneck spectrum; however, the per-flow buffer (of 1–3 packets) at each relay node is still small. Determining the optimal cumulative neighborhood buffer in such multi-flow networks is left for future work.

Scheme	Avg. Goodput (Std. Dev.) Kb/s	Mean RTT (Std. Dev.) ms.
50 pkts buffer	285 (68)	354 (35)
TCP-AP	264 (32)	38 (3)
Buffer per Eq. (3) & (15)	275 (22)	58 (3)

TABLE V: Parking lot topology. 3 FTP streams

Scheme	FTP		VoIP	
	Mean Goodput (Std. Dev.) Kb/s	Mean RTT (Std. Dev.) ms.	Mean Bit Rate Kb/s	Mean Delay (Jitter) ms.
50 pkts buffer	261 (58)	388 (32)	7.8	239 (8)
TCP-AP	240 (17)	54 (6)	8	37 (5)
Buffer per Eq. (3) & (15)	250 (33)	87 (6)	8	40 (6)

TABLE VI: Parking lot topology. 3 FTP and 3 VoIP streams

Our results for the three large file transfer streams are summarized in Table V. We list the mean and the standard deviation for goodput and RTT averaged over the three flows. TCP pacing lowers RTT by a factor of 9 at a cost of 8% drop in goodput. In contrast, our distributed buffering strategy improves RTT by a factor of 6 at the cost of less than 4% drop in average goodput. These improvements may appear relatively modest compared to our results for networks with single TCP streams. This is partly because multiple streams sharing a 50 packet buffer do not let TCP congestion window for individual streams grow quickly to unsustainable large values. Finally, our improvements with distributed buffers are obtained despite the suboptimal (though small) buffer sizes used in this experiment.

Table VI shows our results when each flow in Fig. 8a represents a bundle containing a large file transfer with TCP and a VoIP stream. For the VoIP traffic, we list the mean bit rate (stream goodput), one way delay, and delay jitter averaged over the three streams. With large network buffers, TCP streams can quickly build a large congestion window. This increases the queueing delay for the VoIP traffic that shares the buffer with the TCP streams. With 50 packet buffers, the average delay for the three VoIP streams far exceeds the 150 ms delay bound. In contrast, both TCP-AP and our proposed distributed buffer sizing mechanism limit this unchecked growth of the TCP congestion window. TCP-AP throttles the TCP streams more aggressively, roughly incurring a loss of 5% in goodput compared to our proposed buffer sizing mechanism. As a final observation, we note that the combined goodput of the FTP and VoIP streams in Table VI is less than the corresponding entries in Table V. This is because the 802.11 overhead is better amortized over the larger 1460 byte TCP segments used in our FTP simulations. Various optimizations for reducing this overhead have been discussed in prior work [27].

Scheme	Mean Goodput (Std. Dev.) Kb/s	Mean RTT (Std. Dev.) ms.
50 pkts buffer	465 (44)	260 (53)
TCP-AP	417 (24)	28 (5)
Buffer per Eq. (3) & (15)	455 (27)	56 (13)

TABLE VII: Cross topology. 4 FTP streams

Scheme	FTP		VoIP	
	Mean Goodput (Std. Dev.) Kb/s	Mean RTT (Std. Dev.) ms.	Mean Bit Rate Kb/s	Mean Delay (Jitter) ms.
50 pkts buffer	382 (17)	309 (72)	7.8	187 (30)
TCP-AP	339 (15)	33 (6)	7.9	24 (4)
Buffer per Eq. (3) & (15)	368 (5)	71 (8)	7.9	35 (4)

TABLE VIII: Cross topology. 4 FTP and 4 VoIP streams

Cross topology We next consider the cross topology in Fig. 8b. A source node 5-hops along each edge sends a data flow to the gateway in the center. Using a 2-hop interference model, the 1 and 2-hop nodes around the gateway along each edge constitute the shared bottleneck collision domain between the four flows. With uniform 11 Mb/s links, this distributed buffer size is $B = 18$ packets per Eq. 3. Since the number of hops (and consequently the cost of packet drop) are similar for each edge, we distribute B as 2 and 3 packets at the 3 and 4-hop nodes from source along each edge.

Our results for the FTP streams are summarized in Table VII. We list the mean and the standard deviation for the four flows. TCP pacing lowers RTT 10-fold at a cost of 10% drop in goodput. Our distributed buffering strategy improves RTT by a factor of 5 at the cost of 2% drop in average goodput of the four flows.

Table VIII shows our results when each flow in Fig. 8b represents a bundle containing a large file transfer with TCP and a VoIP stream. Consistent with our observations for the parking lot topology, TCP fills up available network buffers, leading to unacceptable VoIP delays when large network buffers are shared with the TCP stream. Right sizing the network buffers per our proposed mechanism reduces the VoIP delays by more than a factor of 5, at a cost of less than 3% loss in goodput for the TCP streams.

VI. Conclusion and Future Work

Buffer sizing is an important system configuration parameter for WMNs. We use the concept of a neighborhood buffer distributed over multiple nodes to determine the cumulative buffer size that will saturate the spectral resource constituting the network bottleneck. Through careful sizing of this buffer, our goal is to achieve high utilization of the bottleneck spectrum while maintaining low queueing delays. In this context, we propose heuristics for sizing the cumulative neighborhood buffer, and for distributing it amongst the contending nodes. Our proposed approach results in small buffers (as low as 2–3 packets) at each node. Simulation results over various topologies demonstrate that we can effectively maintain high network utilization with low delays.

Our current buffering strategy uses a loose upper bound on achievable network capacity to size the neighborhood buffer. We are working on an adaptive approach in which the nodes in a radio neighborhood measure the current flow rates to estimate the network carrying capacity and adjust their allocated buffer sizes in response. We anticipate that these adaptive schemes would fare better in multi-flow networks as well as in wireless networks susceptible to large variability in channel noise and interference. We are also working on evaluating the feasibility of this scheme on a 802.11 WMN testbed deployment at our university campus.

REFERENCES

[1] V. Gambiroza, B. Sadeghi, and E. Knightly, "End-to-end performance and fairness in multihop wireless backhaul networks," in *Proc. of the ACM MobiCom '04*, Sep. 2004, pp. 287–301.

[2] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proc. of the ACM SIGCOMM '04*, Sep. 2004, pp. 281–292.

[3] R. Johari and D. Tan, "End-to-end congestion control for the internet: delays and stability," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 818–832, Dec. 2001.

[4] Bufferbloat. http://www.bufferbloat.net/.

[5] C. Villamizar and C. Song, "High performance TCP in ANSNET," *SIGCOMM Computer Communications Review*, vol. 24, no. 5, pp. 45–60, 1994.

[6] The network simulator - ns-2. http://www.isi.edu/nsnam/ns.

[7] MadWifi: Multiband Atheros driver for WiFi. http://www.madwifi-project.org.

[8] ath5k FOSS drivers. http://linuxwireless.org/en/users/Drivers/ath5k.

[9] ath9k FOSS drivers. http://linuxwireless.org/en/users/Drivers/ath9k.

[10] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the dynamics of a congestion control algorithm: the effects of two-way traffic," in *Proc. of the ACM SIGCOMM '91*, Oct. 1991, pp. 133–147.

[11] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proc. of the IEEE INFOCOM '00*, Mar. 2000, pp. 1157–1165.

[12] S. M. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with adaptive pacing for multihop wireless networks," in *Proc. of the ACM MobiHoc '05*, May 2005, pp. 288–299.

[13] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," in *Proc. of the IEEE INFOCOM '06*, Apr. 2006.

[14] K. Chen, Y. Xue, S. Shah, and K. Nahrstedt, "Understanding bandwidth-delay product in mobile ad hoc networks," *Elsevier Computer Communications*, vol. 27, no. 10, pp. 923–934, June 2004.

[15] R. Bruno, M. Conti, and E. Gregori, "Analytical modeling of TCP clients in Wi-Fi hot spot networks," in *Proc. of the IFIP Networking '04*, May 2004, pp. 626–637.

[16] T. Li and D. J. Leith, "Buffer sizing for TCP flows in 802.11e WLANs," *IEEE Communication Letters*, vol. 12, no. 3, pp. 216–218, Mar. 2008.

[17] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proc. of the ACM SenSys '04*, Baltimore, MD, Nov. 2004.

[18] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED," in *Proc. of the ACM MobiCom '03*, Sep. 2003, pp. 16–28.

[19] O. Dousse, "Revising buffering in CSMA/CA wireless multihop networks," in *Proc. of the IEEE SECON '07*, Jun. 2007.

[20] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," Internet Engineering Task Force, RFC 3782, Apr. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3782.txt

[21] IEEE LAN/MAN Standards Committee, *IEEE 802.11 - Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*, IEEE, Jun. 2007.

[22] J. Jun and M. L. Sichitiu, "The nominal capacity of wireless mesh networks," *IEEE Wireless Communications*, pp. 8–14, Oct. 2003.

[23] J. Padhye, S. Agarwal, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill, "Estimation of link interference in static multi-hop wireless networks," in *Proc. of the ACM/USENIX IMC '05*, Oct. 2005, pp. 305–310.

[24] A. Kashyap, U. Paul, and S. R. Das, "Deconstructing interference relations in WiFi networks," in *Proc. of the IEEE SECON '10*, Jun. 2010, pp. 73–81.

[25] H. Balakrishnan, C. L. Barrett, V. S. A. Kumar, M. Marathe, and S. Thite, "The distance-2 matching problem and its relationship to the MAC-layer capacity of ad hoc networks," *IEEE Journal on Selected Areas in Communication*, vol. 22, no. 6, pp. 1069–1079, Aug. 2004.

[26] K. Jamshaid and P. Ward, "Gateway-assisted max-min rate allocation for wireless mesh networks," in *Proc. of the ACM MSWiM '09*, Oct. 2009, pp. 38–45.

[27] S. Ganguly, V. Navda, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. R. Das, "Performance optimizations for deploying VoIP services in mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2147 –2158, 2006.

[28] T. Szigeti and C. Hattingh, *End-to-end QoS network design: Quality of Service in LANs, WANs, and VPNs*, 1st ed. Cisco Press, 2004.

[29] A. J. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proc. of the ACM SIGCOMM '89*, Sep. 1989, pp. 1–12.