Neighborhood-centric congestion control for multi-hop wireless mesh networks

Sumit Rangwala, Apoorva Jindal, *Member, IEEE*, Ki-Young Jang, Konstantinos Psounis, *Senior Member, IEEE*, and Ramesh Govindan

***Abstract*—Complex interference in static multi-hop wireless mesh networks can adversely affect transport protocol performance. Since TCP does not explicitly account for this, starvation and unfairness can result from the use of TCP over such networks. In this paper, we explore mechanisms for achieving fair and efficient congestion control for multi-hop wireless mesh networks. First, we design an AIMD-based rate-control protocol called Wireless Control Protocol (WCP) which recognizes that wireless congestion is a neighborhood phenomenon, not a node-local one, and appropriately reacts to such congestion. Second, we design a distributed rate controller that estimates the available capacity within each neighborhood, and divides this capacity to contending flows, a scheme we call Wireless Control Protocol with Capacity estimation (WCPCap). Using analysis, simulations, and real deployments, we find that our designs yield rates that are both fair and efficient. WCP assigns rates inversely proportional to the number of bottlenecks a flow passes through while remaining extremely easy to implement. And, an idealized version of WCPCap is max-min fair, whereas a practical implementation of the scheme achieves rates within 15% of the max-min optimal rates, while still being distributed and amenable to real implementation.**

***Index Terms*—Congestion Control, Multi-hop, Mesh, Wireless, WCP, WCPCap.**

I. INTRODUCTION

Static multi-hop wireless mesh networks, constructed using off-the-shelf omnidirectional 802.11 radios, promise flexible edge connectivity to the Internet, enabling low-cost community networking in densely populated urban settings [2]. They can also be rapidly deployed to provide a communications backbone where none exists, such as in a disaster recovery scenario.

However, their widespread adoption has been limited by significant technical challenges. Finding high-quality routing paths was an early challenge addressed by the research community [14]. However, that alone is not sufficient to ensure good performance in mesh networks, where transport protocols like TCP can perform poorly because of complex interference among neighboring nodes.(We formally define a wireless neighborhood in Section III-A) In particular, TCP does not explicitly account for the fact that congestion in a mesh network is a neighborhood phenomenon. Consider the topology of Figure 1, in which links connect nodes which can exchange packets with each other, perhaps with asymmetric reception rates. In this topology, it is easy to show in simulation and actual experiments that the TCP connection in the middle is almost completely starved (gets extremely low throughput), since it reacts more aggressively to congestion than the two outer flows. As an aside, we note that research on TCP for last-hop wireless networks [7], [8] does not address this problem.

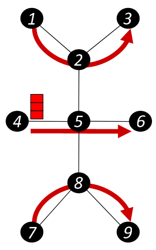


Fig. 1. Stack topology

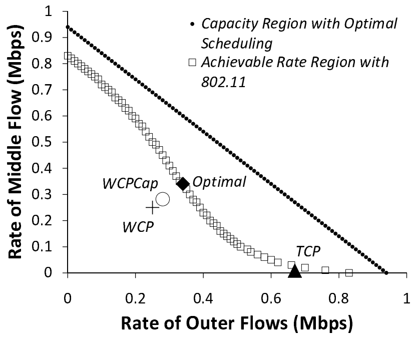


Fig. 2. The achievable rate region

To understand the properties of a desirable solution to this problem, consider Figure 2. The y-axis plots the rate achieved by the middle flow, and the x-axis for the outer two flows (by symmetry, these flows will achieve approximately the same rate for any scheme) of Figure 1. Now, with a perfect MAC scheduler that has the same overhead as 802.11, it is intuitively clear that the rates achievable lie on or below the straight line shown in the figure (since an optimal scheduler would either schedule the two outer flows simultaneously or the flow in the middle). With 802.11, there is some loss of throughput due to contention, and the corresponding *achievable-rate region* bounds the rates achievable by the flows on this topology (in Section IV-A, we describe a methodology to compute the achievable-rate region). TCP achieves rates that lie at one corner of this plot. We contend that, for this topology, a desirable solution is one that gets us close to the max-min fair rate allocation point, which corresponds to the intersection of the 45° line and the 802.11 achievable-rate curve.

In this paper, we explore mechanisms for achieving such a solution in wireless mesh networks. Three considerations inform our choice of mechanisms. First, we do not make any changes to the widely-used 802.11 MAC. It may well be that such changes can improve the performance of our

Sumit Rangwala, Ki-Young Jang, and Ramesh Govindan are with the Department of Computer Science, University of Southern California, Los Angeles, CA, 90089.
Apoorva Jindal is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 48109.
Konstantinos Psounis is with the Department of Electrical Engineering and has a joint appointment with the Department of Computer Science, University of Southern California, Los Angeles, CA, 90089.
Email: {srangwal, apoorvaj, kjang, kpsounis, ramesh}@usc.edu.
The work was supported by NSF grants CNS-0325875 and CNS-0520235, METRANS grant DTRS98-G0019 and a research grant from Cisco.

mechanisms, but we have deliberately limited the scope of our work to enable a clearer understanding of congestion control. Second, our approach is *clean-slate*. We conduct our explorations in the context of a *rate-based* protocol that incorporates some of TCP's essential features (such as ECN, and SACK), yet allows us to explore more natural implementations of the mechanisms for improving fairness and efficiency that we study in this paper. However, our work makes no value judgement on whether a clean-slate transport protocol is *necessary* for mesh networks; it may be possible to retrofit our mechanisms into TCP. Finally, we restrict our explorations to plausibly implementable mechanisms, in contrast to other work that has explored theoretical methods for optimizing (separately or jointly) scheduling and rate assignment in wireless networks [17], [34].

Contributions. We make two contributions in this paper. First, we design an AIMD-based rate-control protocol called WCP which explicitly reacts to congestion within a wireless neighborhood (Section III-A). We start by correctly identifying the precise set of nodes within the vicinity of a congested node that needs to reduce its rates under the assumption that interference range equals transmission range.Signaling these nodes is implemented using a lightweight *congestion sharing* mechanism. Interestingly, we find that congestion sharing alone is not enough, and that, to achieve fairness, sources also need to clock their rate adaptations at the time-scale of the highest RTTs of flows going through the congested region. This is implemented using a local mechanism for *RTT sharing*. Figure 2 shows that, for the topology of Figure 1, WCP avoids starving the middle flow (we discuss methodology and more detailed experimental results in Sections IV and V).

Our second contribution is the design of a distributed rate controller that estimates the available capacity within each neighborhood, and apportions this capacity to contending flows. This scheme, which we call WCPCap (Section III-B), has the property that it uses local information and can *plausibly* be implemented in a distributed fashion. Techniques that perform congestion control by estimating capacity in wired networks have been proposed before, *e.g.,* [29], but wireless capacity estimation is significantly harder. WCPCap is the first attempt in that direction that does not rely on heuristics, but instead uses a precise analytical methodology to accurately estimate the available capacity.

Using analysis, simulations, and real deployments, we find that our designs yield rates that are both fair and efficient. Analogous to TCP, WCP assigns rates inversely proportional to the number of bottlenecks, which, in our case, is the number of congested neighorhoods (defined in Section III-B) a flow passes through, while an idealized version of WCPCap is max-min fair, and a practical implementation of the scheme allocates to each flow a rate within 15% of the rate allocated to it by the max-min fair rate allocation. WCP achieves consistently good performance in the topologies we study while being extremely easy to implement. In fact, our experiments using five flows in a 14-node testbed show that, while TCP starves one or two of these flows in each run, WCP assigns fair rates to all the flows. Finally, in addition to good throughput performance, WCPCap exhibits low end-to-end delay and fast convergence.

II. Related Work

Extensive research has been done to understand the shortcoming and to improve the performance of TCP in wireless networks [10], [21], [23], [30], [35], [47], [51], [54]. We briefly discuss broad classes of research pertinent to our work while referring the interested reader to [36] for a more comprehensive survey of congestion control in wireless networks.

Early work on improving TCP performance in wireless networks focused on distinguishing between packet loss due to wireless corruption from loss due to congestion, in the context of last-hop wireless [7], [8] or wireless wide-area networks [45]. More recent work, however, has addressed congestion control for mobile ad-hoc wireless networks. One class of work concentrates on improving TCP's *throughput* by freezing TCP's congestion control algorithm during link-failure induced losses, especially during route changes [10], [23], [30], [35], [54]. However, unlike WCP, these proposals do not explicitly recognize and account for congestion within a neighborhood. As a result, they would exhibit the same shortcomings of TCP as discussed in Section I.

Another class of work related to WCP address TCP performance issues for ad-hoc networks with no link-failure induced losses using congestion metrics that includes average number of backoffs on a link [13], average number of retransmissions at the MAC layer [21] and the sum of the queuing and transmission delay at each intermediate node [47]. Even though these schemes do not recognize the need of congestion detection and signaling over a neighborhood, their congestion metric *implicitly* takes some degree of neighborhood congestion into account. However, congestion in wireless networks exhibits strong location dependency [51], *i.e.,* different nodes in a congested neighborhood *locally* perceive different degrees of congestion. In the above schemes, flows traversing different nodes in a single congested neighborhood would receive varying levels of congestion notification. In contrast, WCP explicitly shares congestion within a neighborhood, ensuring that each flow in a single congested neighborhood gets its fair share of the bottleneck bandwidth.

Three other pieces of work, however, have recognized the importance of explicitly detecting and signaling congestion over a neighborhood. NRED [51] identifies a set of flows which share channel capacity with flows passing through a congested node. But, it identifies only a subset of the contending flows: it misses flows that traverse two hop neighbors of a node without traversing its one hop neighbors (for example, the flow traversing $7 \rightarrow 9$ in Fig. 3, Section III). Moreover, the mechanism to regulate the traffic rates on these flows is quite a bit more complex than ours (it involves estimating a neighborhood queue size, and using RED [20]-style marking on packets in this queue). Finally, unlike WCP, NRED requires RTS/CTS, is intimately tied to a particular queue management technique (RED), might require special hardware for channel monitoring, and has not been tested in a real implementation. EWCCP [48] correctly identifies the set of flows that share channel capacity with flows passing through a congested node.

EWCCP is designed to be proportionally-fair, and its design as well as its proof of correctness assumes that the achievable rate region of 802.11 is convex. As Figure 2 shows, however, this is not necessarily true. Moreover, EWCCP [48] has also not been tested in a real implementation. Finally, COMUT [28] and our own work IFRC [42] propose rate control schemes designed for many-to-one communication. These designs take advantage of the tree-structured topology and many-to-one traffic pattern and cannot be trivially extended for general, many-to-many traffic settings.

As a final note, our AIMD-based scheme WCP borrows heavily from TCP's essential features such as ECN, SACK, and round-trip time estimation [18], [25], and uses some well established approaches from the active queue management literature [20], [32] to detect congestion at a node.

An alternative to AIMD-based schemes are schemes in which intermediate routers send explicit and precise feedback to the sources. XCP [29] and RCP [16] are examples of such schemes for wired networks. Such schemes cannot be directly extended to multi-hop wireless networks, since the available capacity at a wireless link depends on the link rates at the neighboring edges, and ignoring this dependence will overestimate the available capacity and lead to performance degradation [38] and eventually to instability. Prior schemes for wireless networks that involve sending precise rate feedback to the sources use heuristics based on indirect quantities like queue sizes and the number of link layer retransmissions [4], [46] to limit capacity overestimation. If, instead, one can directly estimate the exact capacity of a link as a function of the link rates at the neighboring edges using a distributed algorithm, then an accurate XCP-like scheme can be implemented for wireless multi-hop networks.

In 802.11-scheduled multi-hop networks, the complex interference among nodes makes it very hard to estimate the capacity of a link. Results have been known either for multi-hop networks that use perfect MAC schedulers [26], or for single-hop 802.11-scheduled networks under saturation traffic conditions [9]. We have recently developed an analytical methodology which characterizes the achievable rate region of 802.11-scheduled multi-hop networks [27]. Our second scheme, WCPCap, uses this prior work of ours to find the supportable per-flow rate in a neighborhood. Further, it uses a novel, decentralized mechanism that relies on message exchanges within local neighborhoods only, to calculate the end-to-end flow rates.

Related to WCPCap are interesting line of works that either explore theoretical methods for jointly optimizing scheduling and rate assignment in wireless networks [17], [34] or reply on a non-standardize MAC [52], [53]. Unlike this body of work, we restrict the scheduler to be 802.11. Explicit rate assignments for 802.11-scheduled MAC always use a centralized computation [33], [44], while our work explores distributed rate-control mechanisms. While optimized rate assignment through a distributed realization of back-pressure techniques [49] have been proposed, it still requires every node in the network to maintain separate queues for each possible network destination. More recent practical studies of the problem have not been able to relax this requirement [5], [50]. Horizon [41], another distributed realization of backpressure techniques, addresses the challenge of load balancing in multi-hop networks with *multi-path* routing, and unlike WCP and WCPCap, does not study congestion control.

Finally, there has been a growing interest in industry [6] and academia [31] in using multiple radios per node, in an effort to mitigate or nullify the complex interference found in multi-hop wireless networks. This line of work is orthogonal to our efforts. We believe that in dense deployments our work will be relevant even if multiple radios are used, since the large number of channels required to completely avoid interference, as well as the complexity associated with their scheduling, would be prohibitively expensive.

III. Design

In this section, we first discuss the design and implementation of WCP, an AIMD-based rate-control protocol that incorporates many of the features of TCP, but differs significantly in its congestion control algorithms. We then describe WCPCap which incorporates wireless capacity estimation in order to assign fair and efficient rates to flows.

A. WCP

WCP is a rate-based congestion control protocol for static multi-hop wireless mesh networks which use the 802.11 MAC. In this section, we assume that the link rates of all the links are equal and auto-rate adaptation is turned off. In Section IV-C, we discuss the impact of relaxing this assumption on our design. In WCP, for every flow, the source maintains a rate r which represents the long term sending rate for the flow. WCP is AIMD-based, so that the source additively increases r on every ACK reception and multiplicatively decreases r upon receiving a congestion notification from routers (intermediate forwarding nodes). Routers signal congestion by setting a congestion bit in the packet header of ongoing packets. Unlike existing congestion control techniques, WCP has novel algorithms for detecting and signaling congestion at the intermediate routers, as well as for adapting rates at sources in response to congestion signals.

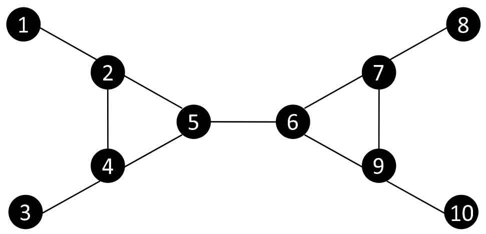


Fig. 3. Congestion neighborhood

Congestion in Multi-hop Wireless Networks. The central observation underlying the design of WCP is that the nature of congestion in a wireless network is qualitatively different from that in a wired network. In a wireless network, since neighboring nodes share the wireless channel, the available transmission capacity at a node can depend on traffic between its neighbors.

More precisely, congestion in wireless networks is defined not with respect to a node, but with respect to transmissions from a node to its neighbor. In what follows, we use the term *link* to denote a one-hop sender-receiver pair. (We use the terms sender and receiver to denote one-hop transmissions, and source and destination to denote the endpoints of a flow). Thus, in Figure 3, we say that a transmission from 5 to 6 is along the link $5 \longrightarrow 6$. Consider the following example. When 5 is transmitting to node 6 it shares the wireless channel with any transmission from node 7, say a transmission from node 7 to node 9, as that transmission can collide with a transmission from node 5 to node 6. However, when node 5 is transmitting to node 2 it *does not* share capacity with, for example, a transmission from node 7 to node 9. Thus, congestion in wireless networks is defined not with respect to a node i, but with respect to a link $i \longrightarrow j$.

What, then, are the set of links ($L_{i \rightarrow j}$) that share capacity with a given link ($i \longrightarrow j$)? Consider link $5 \longrightarrow 6$ in Figure 3. Clearly, all outgoing links from node 5 and node 6 share capacity with link $5 \longrightarrow 6$. Moreover, every outgoing link from a one-hop neighbor of node 5 shares capacity with link $5 \longrightarrow 6$ because any transmission from a neighbor of 5, say node 2, can be sensed by node 5 and would prevent node 5 from capturing the channel while node 2 is transmitting. Additionally, any incoming link to any neighbor of node 5, say $1 \longrightarrow 2$, also shares capacity with link $5 \longrightarrow 6$ as the link-layer acknowledgement from node 2 to node 1 would also prevent node 5 from capturing the channel for transmission. Similarly, any outgoing link from a neighbor of node 6 shares capacity with link $5 \longrightarrow 6$ as any transmission along the outgoing link of neighbor of 6 can collide with transmission along $5 \longrightarrow 6$. Finally, any incoming link into a neighbor of node 6, say $8 \longrightarrow 7$, also shares capacity with $5 \longrightarrow 6$ as the link-layer acknowledgement from node 7 to node 8 can collide with transmissions along $5 \longrightarrow 6$.

Thus, $L_{i \rightarrow j}$, for a mesh network using an 802.11 MAC is defined as:

> the set of all incoming and outgoing links of i, j, all neighbors of i, and all neighbors of j.

Note that $L_{i \rightarrow j}$ includes $i \longrightarrow j$. Moreover, this relationship is symmetric. If a link $i \longrightarrow j$ belongs to $L_{k \rightarrow l}$, $k \longrightarrow l$ also belongs to $L_{i \rightarrow j}$. Furthermore, this definition is valid even when RTS-CTS is used. However, there is an important limitation in our definition. If a node is outside another node's transmission range, but within its interference range, WCP cannot account for the reduction in channel capacity as a result of the latter's transmissions.

Congestion Detection and Sharing. In WCP, one key idea is *congestion sharing*: if link $i \longrightarrow j$ is congested, it shares this information with all links in $L_{i \rightarrow j}$. Packets traversing those links (as well as link $i \longrightarrow j$ itself) are marked with an explicit congestion notification, so that sources can appropriately adapt the rates of the corresponding flows. We now describe how routers detect congestion, and how they share their congestion state.

Congestion detection in WCP is deliberately simple. A router detects congestion on its outgoing link using a simple thresholding scheme. It maintains an exponentially weighted moving average (EWMA) of the queue size, $q^{avg}_{i \rightarrow j}$, for every *outgoing link* $i \longrightarrow j$ as

$$q^{avg}_{i \rightarrow j} = (1 - w_q) * q^{avg}_{i \rightarrow j} + w_q * q^{inst}_{i \rightarrow j}$$

where $q^{inst}_{i \rightarrow j}$ is the instantaneous queue size for link $i \longrightarrow j$ and w_q is the EWMA weight. A link is congested when its average queue size is greater than a congestion threshold K. Various other congestion detection techniques have been explored in wireless networks: channel utilization [51], average number of retransmissions [22], mean time to recover loss [40], among others. We choose queue size as a measure of congestion for two reasons: it has been shown to work sufficiently well in wireless networks [24], [42]; and is a more natural choice for detecting congestion per link compared to, say, channel utilization which measures the level of traffic around a node. Also, more sophisticated queue management schemes are possible (*e.g.*, RED or AVQ), but they are beyond the scope of this paper.

When a router i detects that $i \longrightarrow j$ is congested, it needs to share this information with nodes at the transmitting ends of links in $L_{i \rightarrow j}$ (henceforth referred to as nodes in $L_{i \rightarrow j}$). This is achieved by piggybacking congestion information on each outgoing packets which neighbors snoop. Specifically, each node maintains the congestion state of all its outgoing and incoming links. Nodes can locally determine (from queue size) the congestion state of their outgoing links. For nodes to obtain congestion state on their incoming links, every node during an outgoing packet transmission along a link includes congestion state of *that* link in the outgoing packet. In addition, every node also includes the following information in each outgoing packet: a bit indicating if any outgoing or incoming link from the node is congested and the identity of the link (sender and receiver of the link); and a bit indicating if any outgoing or incoming link from any of the node's neighbors is congested and the identity of the link. This latter bit is calculated from information obtained by snooping the former bit from neighbors and requires no additional information exchange. In the event of more than one link being congested at a node, it is sufficient for the node to select (and inform its neighbors) of only one of these links. Information shared in this manner is sufficient for any node in $L_{i \rightarrow j}$ to receive congestion notification when link $i \longrightarrow j$ is congested

Finally, when a node in $L_{i \rightarrow j}$ detects that link $i \longrightarrow j$ is congested, it marks all outgoing packets on that link with an explicit congestion indicator (a single bit).

Rate Adaptation. In WCP sources perform rate adaptation. While the principles behind our AIMD rate adaptation algorithms are relatively standard, our contribution is to correctly determine the timescales at which these operations are performed. The novel aspect of our contribution is that these timescales are determined by the RTTs of flows traversing a congested neighborhood; without our innovations (described below), flows do not get a fair share of the channel, and sometimes react too aggressively to congestion.

A source S in WCP maintains a rate r_f for every flow f originating at S. It linearly increases the rate r_f every t_{ai}

seconds, where t_{ai} is the control interval for additive increase:

$$r_f = r_f + \alpha$$

where α is a constant. The choice of t_{ai} is an important design parameter in WCP. In the above equation the rate of change of r_f, dr_f/dt is α/t_{ai}. Intuitively, for stable operation, dr_f/dt should be dependent on feedback delay of the network. Using the weighted average round-trip time of flow f, rtt_f^{avg}, of the flow seems an obvious choice for t_{ai} as it satisfies the above requirement. But consider three flows $1 \rightarrow 3$, $4 \rightarrow 6$, and $7 \rightarrow 9$ in Figure 1. Packets of flow $1 \rightarrow 3$ share the wireless channel with nodes 1 through 6 while packets from flow $4 \rightarrow 6$ share wireless channel with all the nodes in the figure. As the rate of all the flows in the network increases, flow $4 \rightarrow 6$ experiences more contention as compared to flow $1 \rightarrow 3$ and the average RTT of flow $4 \rightarrow 6$ increases much faster than the average RTT of flow $1 \rightarrow 3$. Thus, even if these flows were to begin with the same rate, their rates would diverge with the choice of $t_{ai} = rtt_f^{avg}$. This happens because a fair allocation of capacity using a AIMD scheme requires similar dr_f/dt for all the flows sharing the capacity [12].

To enable fairness, WCP introduces the notion of a *shared RTT*. Denote by $rtt_{i\rightarrow j}^{avg}$ the average RTT of all the flows traversing the link $i \rightarrow j$ (the average RTT of each flow is computed by the source, and included in the packet header) *i.e.*,

$$rtt_{i\rightarrow j}^{avg} = \sum_{\forall f \in F_{i\rightarrow j}} \frac{rtt_f^{avg}}{|F_{i\rightarrow j}|}$$

where $F_{i\rightarrow j}$ is the set of flows traversing link $i \rightarrow j$. For each link $i \rightarrow j$, node i computes the shared RTT, $rtt_{i\rightarrow j}^{Smax\_avg}$, as the maximum RTT among all links in $L_{i\rightarrow j}$ *i.e.*,

$$rtt_{i\rightarrow j}^{Smax\_avg} = \max_{\forall k\rightarrow l \in L_{i\rightarrow j}} (rtt_{k\rightarrow l}^{avg})$$

In words, this quantity measures the largest average flow RTT across the set of links that $i \rightarrow j$ shares the channel with. Why this particular choice of timescale? Previous work has shown that the average RTT of flows is a reasonable control interval for making congestion control decisions [29]. Our definition conservatively chooses the largest control interval in the neighborhood.

For all flows traversing link $i \rightarrow j$, the router includes $rtt_{i\rightarrow j}^{Smax\_avg}$ in every packet header only if it exceeds the current value of that field in the header. The source uses this value for t_{ai}: thus, t_{ai} is the largest shared RTT across all the links that the flow traverses. This ensures that the value of the control interval t_{ai} for a flow is no less than the highest RTT of any flow with which it shares a wireless neighborhood. Intuitively, with this choice of the control interval, all flows in the Stack topology will increase their rates at the same timescale.

Upon receiving a packet with a congestion notification bit set, a source reduces the rate r_f by half and waits for the control interval for multiplicative decrease t_{md} before reacting again to any congestion notification from the routers. t_{md} must be long enough so that the source has had time to observe the effect of its rate reduction. Moreover, for fairness, flows that traverse the congested region[1] must all react at roughly the same timescale. To ensure this, WCP also computes a quantity for each link that we term the *shared instantaneous RTT*, denoted by $rtt_{i\rightarrow j}^{Smax\_inst}$. This is computed in exactly the same way as the shared RTT, described above, except that the instantaneous RTT is used, rather than the average RTT. The former is a more accurate indicator of the current level of congestion in the network and is a more conservative choice of the timescale required to observe the effect of a rate change. As before, routers insert this shared instantaneous RTT into the packet header only if it exceeds the current value. Sources set t_{md} to be the value of this field in the packet header that triggered the multiplicative decrease. If a flow traverses multiple congested regions, its multiplicative decreases are clocked by the neighborhood with the largest shared instantaneous RTT.

Similar to congestion sharing, computation of $rtt_{i\rightarrow j}^{Smax\_avg}$ ($rtt_{i\rightarrow j}^{Smax\_inst}$) requires sharing "link RTTs", $rtt_{i\rightarrow j}^{avg}$ ($rtt_{i\rightarrow j}^{inst}$), among all nodes in $L_{i\rightarrow j}$ requiring significant overhead. However, its definition permits a natural optimization. Similar to congestion sharing discussed above, every node includes in each outgoing packet only the maximum of $rtt_{i\rightarrow j}^{avg}$ ($rtt_{i\rightarrow j}^{inst}$) over all incoming and outgoing links of the node and the maximum of $rtt_{i\rightarrow j}^{avg}$ ($rtt_{i\rightarrow j}^{inst}$) over all incoming and outgoing links of all the neighbors of the node. This allows for a low-overhead distributed calculation of shared RTT over all the nodes in $L_{i\rightarrow j}$.

Finally, we describe how the source uses the value r_f. WCP aims to assign fair *goodputs*. Naively sending packets at the rate r_f assigns fair throughputs, but packet losses due to channel error or interference can result in unequal goodputs. Instead, WCP sends packets at a rate r_f/p, when p is the empirically observed packet loss rate over the connection. Intuitively, this *goodput correction* heuristic sends more packets for flows traversing lossy paths, equalizing flow goodputs. Other rate-based protocols [39] use more sophisticated loss rate computation techniques to perform similar goodput correction. As we show in Section IV, our approach works extremely well for WCP. In that section, we also quantify the impact of turning off this "correction".

Implementation. We could have retrofitted congestion and RTT sharing in TCP. But, the complexity of current TCP implementations, and the fact that TCP performs error recovery, congestion control and flow control using a single window-based mechanism, made this retrofit conceptually more complex. Given that our goal was to understand the issues underlying congestion in mesh networks, incremental deployability was not paramount. So, at the cost of some additional packet header overhead, we decided to explore a clean-slate approach. Our implementation uses a rate-based protocol for congestion control (as described above), but uses an implementation of TCP SACK for error recovery, a window-based flow control mechanism exactly like TCP, and the same RTO estimation as in TCP. In a later section, we show that our implementation does not bias the results in any

[1]We use the terms congested region and congested neighborhood interchangeably in this paper.

way: if we remove our sharing innovations from WCP, its performance is comparable to TCP.

B. *WCPCap*

An alternative to WCP is a protocol in which the network sends explicit and precise feedback to the sources. In order to do this, it is important to be able to estimate the available capacity within a neighborhood, a non-trivial task. In this section, we describe WCPCap, a protocol that provides explicit feedback (in much the same way that XCP [29] and RCP [16] do for wired networks). An important goal in designing WCPCap is to explore the feasibility of capacity estimation using only local information, thereby making it amenable to distributed implementation.

Determining the Achievable Link-Rate Region. At the core of WCPCap is a technique to determine whether a given set of rates is *achievable* in an 802.11 network; using this technique, WCPCap estimates the available capacity and distributes this fairly among relevant flows. This technique is presented in detail in our prior work [27]. For completeness, we describe the main idea of the analytical methodology here, assuming IEEE 802.11 scheduling with RTS/CTS in the network.

The precise goal of the technique is as follows. Given a link $i \longrightarrow j$, and a set of candidate aggregate rates $r_{l\rightarrow m}$ over links $l \longrightarrow m$ belonging to $L_{i\rightarrow j}$ (link $i \longrightarrow j$ belongs to this set), we seek a decision procedure that will enable us to determine if these rates are achievable. The decision process assumes that the channel loss rates (losses not due to collisions) of links in $L_{i\rightarrow j}$, and the interference graph between links in $L_{i\rightarrow j}$ are known. Channel losses are assumed to be independent Bernoulli random variables. The interference model neglects some physical layer phenomena like the capture effect [11] (where a receiver can correctly decode data despite of interference), situations where transmitters send data despite of interference and carrier sensing, and situations where remote links in isolation do not interfere with the link under study, but their aggregate effect may cause loses on the link [15]. The interested reader is referred to [27] for a detailed description of all the assumptions, an extensive evaluation of their effect on the accuracy of the model, and a discussion on how to remove them.

The decision process first determines, for each link, the expected service time in terms of (a) the collision probability at the receiver and (b) the idle time perceived by the transmitter of that link. Given these service times, the given set of link-rates is achievable only if

$$\sum_{e\in O_v} \lambda_e E[S_e] \leq U, \forall v \in V$$

where V is the set of all nodes, O_v is the set of outgoing links from a node $v \in V$, λ_e is the packet arrival rate at link e, $E[S_e]$ is the expected service time of a packet at link e, and the *utilization factor* U is a fraction between 0 and 1 and reflects the desired utilization of the channel. In practice, U is usually set to less than 1 to keep the system stable. Otherwise, small non-idealities can drive the network beyond the capacity region.

The key challenge then, is to determine the collision and the idle time probabilities, made difficult because these values for a link depend on the rates at its neighboring links, which, in turn, depend on the rates at their neighboring links and so on. We use the following procedure: the sub-graph formed by the set of links in $L_{i\rightarrow j}$ is *decomposed* into a number of two-link topologies and the collision and idle probabilities for each two-link topology is derived. The net probability is found by appropriately *combining* the individual probabilities from each two-link topology. Combining these probabilities is quite complicated due to the interdependence among links. For brevity, we will omit the analytical formulas and their complete derivations. The interested reader is referred to [27] for details.

Estimating Available Bandwidth. WCPCap uses the achievable rate computation technique to estimate achievable bandwidth and give precise rate feedback to sources. Conceptually, each router maintains, for each outgoing link $i \longrightarrow j$, a rate $R_{i\rightarrow j}$ which denotes the maximum rate allowable for a flow passing through the link. However, a flow traversing $i \longrightarrow j$ is actually only allowed to transmit at the minimum (denoted $R^{min}_{i\rightarrow j}$) of all rates $R_{k\rightarrow l}$ such that $k \longrightarrow l$ belongs to $L_{i\rightarrow j}$ (intuitively, at the most constraining rate over all links that share channel capacity with $i \longrightarrow j$). The rate feedback is carried in the packet header. When a packet traverses $i \longrightarrow j$, the router sets the feedback field to $R^{min}_{i\rightarrow j}$ if $R^{min}_{i\rightarrow j}$ is lower than the current value of the field. This feedback rate is eventually delivered to the source in an end-to-end acknowledgement packet, and the source uses this value to set its rate. Thus the source sets its rate to the smallest allowable rate in the wireless neighborhoods that it traverses.

$R_{i\rightarrow j}$ for each link is updated every $k \cdot rtt^{Smax\_avg}_{i\rightarrow j}$, where $rtt^{Smax\_avg}_{i\rightarrow j}$ is the shared RTT defined in Section III-A and k is a parameter which trades-off the response time to dynamics for lower overhead. The duration between two successive updates of $R_{i\rightarrow j}$ is referred to as an epoch. During each epoch, transmitter i measures $x_{i\rightarrow j}$, the actual data rate over link $i \longrightarrow j$ and $n_{i\rightarrow j}$, the number of flows traversing link $i \longrightarrow j$. Using $x_{k\rightarrow l}$ and $n_{k\rightarrow l}$ for all $k \longrightarrow l$ in $L_{i\rightarrow j}$ transmitter i computes the new value of $R_{i\rightarrow j}$ (denoted by $R^{new}_{i\rightarrow j}$) to be used in the next time epoch, and broadcasts $x_{i\rightarrow j}$, $n_{i\rightarrow j}$, and $R^{new}_{i\rightarrow j}$ to all nodes in $L_{i\rightarrow j}$. (If the measured $x_{i\rightarrow j}$ in the previous epoch equals zero at a link $i \longrightarrow j$, it does not broadcast its current value of $R^{new}_{i\rightarrow j}$. Thus links which become inactive due to network dynamics will not contribute in determining the largest achievable flow-rate in a neighborhood.)

We now describe how $R^{new}_{i\rightarrow j}$ is determined (Figure 4). Note that the transmitter i has $x_{k\rightarrow l}$ and $n_{k\rightarrow l}$ for all links $k \longrightarrow l$ in $L_{i\rightarrow j}$. It uses this information, and the methodology described above, to determine the maximum value of δ such that the rate vector $\vec{x}$ shown in Figure 4 is achievable. (δ can have a negative value if the current rates in the neighborhood are not achievable.) Then, node i sets $R^{new}_{i\rightarrow j}$ to $R_{i\rightarrow j} + \rho\delta - \beta q^{inst}_{i\rightarrow j}/rtt^{Smax\_avg}_{i\rightarrow j}$ if δ is positive, else $R^{new}_{i\rightarrow j}$ is set to $R_{i\rightarrow j} + \delta - \beta q^{inst}_{i\rightarrow j}/rtt^{Smax\_avg}_{i\rightarrow j}$. We use a scaling factor ρ while increasing the rate to avoid big jumps, analogous to similar scaling factors in XCP and RCP. On the other hand, we remain conservative while decreasing the rate. $q_{i\to j}^{inst}$ denotes the instantaneous queue at link $i \to j$, $rtt_{i\to j}^{Smax\_avg}$ is the shared RTT defined in Section III-A, and β is a scaling parameter. To ensure that the rate goes down when the queue builds up, we subtract a fraction of the bandwidth required to drain the queue within one shared RTT ($q_{i\to j}^{inst}/rtt_{i\to j}^{Smax\_avg}$). Each node independently computes $R_{k\to l}$ for its links. These computations do not need to be synchronized, and nodes use the most recent information from their neighbors for the computation.

Next, we describe how the value of the parameter U (the maximum allowed utilization per queue) is determined. The analysis described at the start of this section to derive the achievable rate region does not incorporate losses at higher layers (that is, it incorporates only channel losses and collisions), and hence, assumes infinite buffer sizes and infinite MAC retransmit limits. Assuming no losses, operating very close to the capacity region will result in large delays and huge queues. However, in practice both the buffer sizes and MAC retransmit limits are finite. Hence, these huge queues can result in significant losses. Additionally, note that the procedure presented in [27] assumes global knowledge, while the information exchange in WCPCap is only between neighbors. Hence, the computation itself is approximate which can potentially lead to an overestimation of available capacity. For these two reasons, we operate the network well within the capacity region; the parameter U controls how far the network is from the boundary of the capacity region. How close to the boundary of the capacity region can we operate without overflowing buffers and without overestimating available capacity depends on the topology at hand. Hence, the value of U depends on the topology. Choosing a conservative value for U is inefficient as it leads to a low channel utilization in most topologies. So we use the following algorithm to set the value of U. If $q_{i\to j}^{avg}$ (the average queue size) is greater than 1, the value of U is reduced; and if $q_{i\to j}^{avg}$ remains less than 1 for 5 iterations of the algorithm described in Figure 4, the value of U is increased. Binary search is used to converge to the correct value of U. For example, in the Stack topology (Figure 1), this approach yields a value of $U = 0.85$.

The computational overhead of the WCPCap algorithm is very low. To determine $R_{i\to j}^{new}$, we perform a binary search to find the maximum value of δ such that the rate vector $\vec{x}$ is achievable. Each iteration decomposes $L_{i\to j}$ into two-link topologies, computes collision and idle probabilities for each two-link topology, and combines the results. Overall, the algorithm requires a logarithmic number of iterations whose complexity is polynomial in $|L_{i\to j}|$. In practical topologies the cardinality of $L_{i\to j}$ is small. For example, in our experiments (run on 3.06GHz Linux boxes) determining $R_{i\to j}^{new}$ takes as much time as it takes to send a data packet. Since each epoch consists of about 30 data packet transmissions and a single $R_{i\to j}^{new}$ computation, the computational overhead per epoch is very low.

Finally, we note that, if naively designed, WCPCap can impose significant communication overhead. For each link $i \to j$, the following information needs to be transmitted to all nodes in $L_{i\to j}$ once every epoch: the maximum RTT

```
Every $k \cdot rtt_{i\to j}^{Smax\_avg}$ sec
    Find max $\delta$ such that
    $\vec{x} \leftarrow ( x_{k\to l} + n_{k\to l}\delta$  for $k \to l \in L_{i\to j} )$
    is achievable
       $R_{i\to j}^{new} \leftarrow \begin{cases} R_{i\to j} + \rho\delta - \beta q_{i\to j}^{inst}/rtt_{i\to j}^{Smax\_avg} & \delta > 0 \\ R_{i\to j} + \delta - \beta q_{i\to j}^{inst}/rtt_{i\to j}^{Smax\_avg} & \delta \le 0 \end{cases}$
    Broadcast $R_{i\to j}^{new}$, $x_{i\to j}$ and $n_{i\to j}$
              to all links in $L_{i\to j}$
```

Fig. 4. Pseudo-code for rate controller at link $i \to j$

across the flows passing through the link, the actual data rate at the link, the number of flows passing through the link and $R_{i\to j}$. Assuming one byte to represent each variable, the overhead is equal to $4L_{i\to j}$ bytes. For practical topologies, where neighborhood sizes are expected to be less than 20, this overhead consumes less than 15% of the actual throughput. However, the overhead does increase linearly with $L_{i\to j}$. There are ways to optimize this, by quantizing the information or reducing the frequency of updates, but we have left these to future work. Instead, in our simulations, we assume that all the relevant information is available at each node without cost, since our goal has been to understand whether available bandwidth estimation using only local information is plausibly implementable in wireless networks.

Properties. To understand the design rationale of the WCPCap algorithm, we characterize the fairness properties of an idealized WCPCap algorithm. The idealized WCPCap algorithm assumes that all control message broadcasts are exchanged instantaneously and without loss, and each node has complete information about the entire network instead of just its neighborhood. Specifically, each node is aware of the data rate at each link in the network and the global network topology. The last assumption is needed because residual capacity at a link depends on the global topology and not merely the local neighborhood topology [27]. Hence WCPCap obtains an approximate value of the residual capacity while idealized WCPCap will obtain an exact value of the residual capacity. We prove fairness properties for idealized WCPCap here, and evaluate how the non-idealities impact the performance of WCPCap (without the additional assumptions of idealized WCPCap) through simulations in Section IV.

We will prove that the rates assigned by idealized WCPCap converge to the max-min rate allocation. The proof is constructed using two lemmas. The first lemma looks at the properties of the max-min allocation in wireless multi-hop networks while the second lemma studies the rates assigned by idealized WCPCap. Before presenting the lemmas, we define three concepts which will be extensively used in the proofs. At the max-min allocation, let the links whose queues are fully utilized (arrival rate = service rate) be referred to as congested links, and the neighborhood of congested links be referred to as congested neighborhoods. Note that each flow may pass through several congested neighborhoods. We define the most congested neighborhood a flow passes through to be the neighborhood which gets congested at the lowest throughput amongst the congested neighborhoods that flow traverses. Thus, there is a unique most congested neighborhood associated with each flow. The throughput achieved by a flow is dictated by the most congested neighborhood it passes through.

Lemma 1: A rate allocation which assigns the largest achievable equal rate to the flows which share the most congested neighborhood is the max-min rate allocation.

Proof: Let there be n flows passing through the congested neighborhood C_R. Additionally, let k of these n flows have C_R as the most congested neighborhood they pass through. Consider the following rate allocation. Fix the rate of the other $n-k$ flows as dictated by the most congested neighborhood they pass through, and then assign the maximum possible equal rate to the k flows. Label this rate r_{eq}. Then, by definition of the most congested neighborhood a flow passes through, the other $n-k$ flows have a rate smaller than r_{eq}.

Let $i \longrightarrow j$ denote the congested link in the congested neighborhood C_R. (That is, link $i \longrightarrow j$ is fully utilized.) Increasing the rate of any of the k flows will either increase the busy probability or the collision probability at link $i \longrightarrow j$, making its queue unstable [27]. To keep the rate allocation feasible, the rates of one of the other flows (which have either smaller or equal rates) will have to be reduced. Hence, by definition, allocating the maximum possible equal rate to the k flows sharing the same most congested neighborhood is the max-min rate allocation. ■

Let f_1 and f_2 be two flows which share the most congested neighborhood they traverse. The next lemma relates the rates allocated to f_1 and f_2 by idealized WCPCap.

Lemma 2: The rates allocated by idealized WCPCap to f_1 and f_2 converge to the largest achievable equal rate.

Proof: By design, f_1 and f_2 are allocated equal rates by idealized WCPCap. So, to prove this lemma, we will prove that this equal rate converges to the largest achievable equal rate. We first assume that the rate of flows f_1 and f_2 is initialized to 0. WCPCap calculates the maximum rate increase per flow which keeps the system stable. (Since, by assumption, idealized WCPCap calculates the exact residual capacity at each link, the flow-rate updates will not cause any link-rate to exceed its capacity,and hence, the flow rates will always increase.) This maximum rate increase is labelled δ. Then, the rate allocated to f_1 and f_2 is increased by $\rho\delta$. Since the system remains within the stable region, the queue size remains negligible. Hence, within $\frac{log(1-\theta)}{log(1-\rho)}$ iterations of the algorithm, the rate allocated to f_1 and f_2 is more than $\theta < 1$ of the largest achievable equal rate. Thus, the rates allocated by idealized WCPCap converge to the largest achievable equal rate.

Finally, now we comment on convergence of the algorithm if the rate of flows f_1 and f_2 is initialized to a non-zero value. Then, if the initialization is to a value such that the arrival rate on the bottleneck link is less than its service rate, by the same argument as before, within $\frac{log(1-\theta)}{log(1-\rho)}$ iterations of the algorithm, the rate allocated to f_1 and f_2 is more than $\theta < 1$ of the largest achievable equal rate. Now, if the initialization is to a value such that the arrival rate on the bottleneck link exceeds its service rate, in the next iteration, WCPCap will reduce the rate of the flows to a value such that the arrival rate reduces below the service rate as the reduction will not only be due to a lower estimated capacity but also due to non-negligible queue sizes. After this first iteration, the same argument for convergence applies. ■

Theorem 1: The rates assigned by idealized WCPCap converge to the max-min rate allocation.

Proof: Since Lemma 2 holds for all flows which share the most congested neighborhood they traverse, in conjuntion with Lemma 1, it implies that the rates allocated by idealzed WCPCap converge to the max-min rate allocation. ■

IV. Simulation Results

In this section we evaluate the performance of WCP and WCPCap in simulation, and in the next we report on results from real-world experiments of WCP.

A. Methodology

We have implemented WCP and WCPCap using the Qualnet simulator [3] version 3.9.5. Our WCP implementation closely follows the description of the protocol in Section III-A. Our WCPCap implementation, on the other hand, does not simulate the exchange of control messages at the end of each epoch; rather, this control information is made available to the relevant simulated nodes through a central repository. This ignores the control message overhead in WCPCap, so our simulation results overestimate WCPCap performance. This is consistent with our goal, which has been to explore the feasibility of a wireless capacity estimation technique.

All our simulations are conducted using an unmodified 802.11b MAC (DCF). We use default parameters for 802.11b in Qualnet unless stated otherwise. Auto-rate adaption at the MAC layer is turned-off and the rate is fixed at 11Mbps. Most of our simulations are conducted with zero channel losses (we report on one set of simulations with non-zero channel losses), although packet losses due to collisions do occur. However, we adjusted the carrier sensing threshold to reduce interference range to equal transmission range. This prevented MAC backoffs at a node due to transmissions from other nodes outside the transmission range but within the interference range, thus helping us create topologies on which performance of our schemes could be clearly studied.

On this set of topologies (described below), we run bulk transfer flows for 200 seconds for WCP, WCPCap, and TCP. Our TCP uses SACK with ECN, but with Nagle's algorithm and the delayed ACK mechanism turned off; WCP implements this feature set. (We have also evaluated TCP-Reno on our topologies. The results are qualitatively similar.) Congestion detection for TCP uses the average queue size thresholding technique discussed in Section III-A. Other parameters used during the runs are given in Table I. Note that our choice of α is conservative, ensuring small rate increases over the range of timescales we see in our topologies. This choice of α also works in our real-world experiments, but more experimentation is necessary to determine a robust choice of α. For each topology we show results averaged over 10 runs.

We measure the goodput achieved by each flow in a given topology by TCP, WCP, and WCPCap, and compare these goodputs with the optimal max-min rate allocations for each topology. To compute the max-min rate allocations, we

Parameter	Value
Congestion Threshold(K)	4
EWMA Weight (w_q)	0.02
Router Buffer size	64 packets
Packet Size	512 bytes
Additive Increase Factor (α)	0.1
WCPCap epoch duration constant (k)	10
WCPCap scale factors (ρ and β)	0.3 and 0.1

TABLE I
PARAMETERS USED IN SIMULATIONS

observe that the methodology in Section III-B, with $U = 1$, can be applied to the global network topology to characterize the achievable rate region for a collection of flows. Intuitively, we can view the service times and arrival rates on links, together with flow conservation constraints, as implicitly defining the achievable rate region for the topology [27]. (Essentially, this is how we derive the achievable rate region in Figure 2.[2]) The max-min allocations can then be found by searching along the boundary of the achievable rate region. [3] Using this methodology, we are also able to identify the links in a given topology that tend to be congested: we simply simulate the optimal max-min rate allocations, and identify congested links as those whose queues are nearly fully utilized (utilization of the queue is > 0.95). Note that we use this information in our intuitive discussion about the dynamics of each topology; this information is obviously not used in any way by neither WCP nor WCPCap.

To understand the performance of WCP and WCPCap, we examine four topologies,with associated flows, as shown in Figures 1, 5, 6, and 7. Nodes connected by a solid line can hear each others' transmissions. (Since, in our simulations, we equalize interference range and transmission range, only nodes that can hear each others' transmissions share channel capacity with each other.) Arrows represent flows in the network. Congested links (determined using the methodology described above) are indicated with a symbol depicting a queue. Each of these four topologies has qualitatively different congestion characteristics, as we discuss below.

Stack (Figure 1) consists of a single congested region. $4 \longrightarrow 5$ is the congested link, and all other links in the topology belong to $L_{4\rightarrow5}$. **Diamond** (Figure 5) contains two intersecting congested regions. $1 \longrightarrow 2$ and $7 \longrightarrow 8$ are both congested links. $L_{1\rightarrow2}$ includes all outgoing links from nodes 1 to 6 and $L_{7\rightarrow8}$ includes all outgoing link from node 4 to 9. **Half-Diamond** (Figure 6) contains two overlapping congested regions. $4 \longrightarrow 5$ and $7 \longrightarrow 8$ are congested, and $L_{7\rightarrow8}$ is a subset of $L_{4\rightarrow5}$. **Chain-Cross** (Figure 7) contains two congested regions, with four flows traversing one region, and two flows the other. $1 \longrightarrow 2$ is a congested link, but $6 \longrightarrow 7$ does not belong to $L_{1\rightarrow2}$. $4 \longrightarrow 5$ and $4 \longrightarrow 3$ are also congested, and $L_{4\rightarrow5}$ does include $6 \longrightarrow 7$.

Finally, since WCP uses a rate-based implementation, it is important to ensure that its *baseline* performance is comparable to that of TCP. To validate this, we ran TCP and WCP on a chain of 15 nodes. WCP gets 20% *less* throughput on this topology; it is less aggressive than TCP. We also disabled RTT and congestion sharing in WCP, and ran this on all our topologies. In general, this stripped-down version of WCP gets qualitatively the same performance as TCP. For example, Figure 8 shows the goodputs achieved by each flow for the Stack topology. As expected, WCP without congestion and RTT sharing starves the middle flow, just as TCP does, although to a lesser extent since its rate increases are less aggressive than that of TCP.

[2] In this section, we will assume the transport overhead to be equal to the overhead of WCP while Figure 2 assumes the transport overhead to be the same as that of TCP.

[3] We verified through simulations that the rates determined through this analytical methodology are achievable while these rates scaled up by 10% are not.

B. Performance of WCP and WCPCap

We now discuss the performance of WCP and WCPCap for each of our topologies. In what follows, we use the notation $f_{i\rightarrow j}$ to denote a flow from node i to node j.

Stack (Figure 9). The optimal (max-min) achievable rates for this topology are 300 kbps for all the flows. TCP, as described earlier, starves the middle flows ($f_{4\rightarrow6}$). Intuitively, in TCP, flows traversing links that experience more congestion ($4 \longrightarrow 5$) react more aggressively to congestion, leading to lower throughput. WCP identifies the single congestion region in this topology ($L_{4\rightarrow5}$) and shares the rate equally among all the flows assigning about 250 kbps to all the flows which is within 20% of the optimal achievable rate for this topology. WCPCap, with a more precise rate feedback, assigns slightly higher rates to all the flows and these allocated rates for each flow is within 15% of the rate allocated to it by the max-min fair rate allocation.

Diamond (Figure 10). The optimal achievable rates for this topology are 325 kbps for all the flows. TCP starves flows traversing the congested links in this topology. By contrast, WCPCap, assigns 300 kbps to all the flows. Hence, it achieves rates within 10% of the max-min optimal rates. WCP, however, assigns $f_{4\rightarrow6}$ approximately half the rate assigned to the other flows. This topology consists of two congested regions ($L_{1\rightarrow2}$ and $L_{7\rightarrow8}$) and $f_{4\rightarrow6}$ traverses both congested regions while the other two flows traverse only one. Roughly speaking, $f_{4\rightarrow6}$ receives congestion notification twice as often as the other flows, and therefore reacts more aggressively. Thus, WCP is *not* max-min fair. WCP appears to assign rates to flows in inverse proportion to the number of congested regions traversed.

Half-Diamond (Figure 11). The optimal max-min rates for this topology are 315 kbps for $f_{4\rightarrow6}$ and $f_{7\rightarrow9}$, and 335 kbps for $f_{1\rightarrow3}$; the asymmetry in this topology permits $f_{1\rightarrow3}$ to achieve a slightly higher rate. Relative to other topologies, TCP performs fairly well for this topology. WCPCap achieves rates within 14% of the max-min optimal rates. WCP assigns comparable rates to $f_{4\rightarrow6}$ and $f_{7\rightarrow9}$ as they traverse both congested regions $L_{4\rightarrow5}$ and $L_{7\rightarrow8}$. $f_{1\rightarrow3}$ achieves a higher rate as it traverses only one congested region ($L_{4\rightarrow5}$) but its rate is not twice the rate of the other flow. Thus WCP achieves a form of fairness in which the rate allocations depend not only on the number of congested regions a flow passes through, but also the intensity of congestion in those regions.

Chain-Cross (Figure 12). The optimal rates for this topology are 420 kbps for $f_{6\rightarrow7}$ and 255 kbps for all other flows. TCP

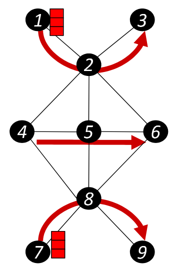


Fig. 5. Diamond topology

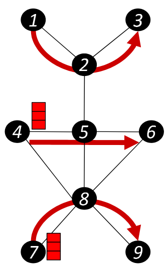


Fig. 6. Half-Diamond topology

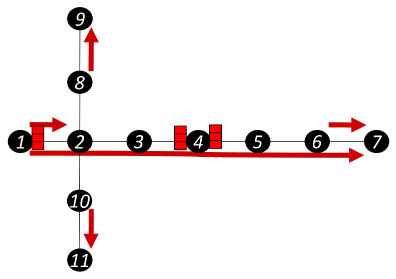


Fig. 7. Chain-Cross topology

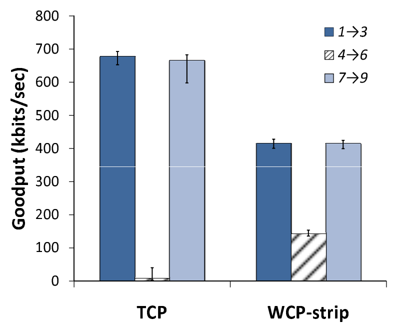


Fig. 8. WCP without congestion and RTT sharing, Stack

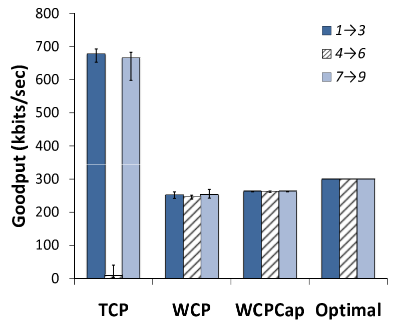


Fig. 9. WCP and WCPCap, Stack

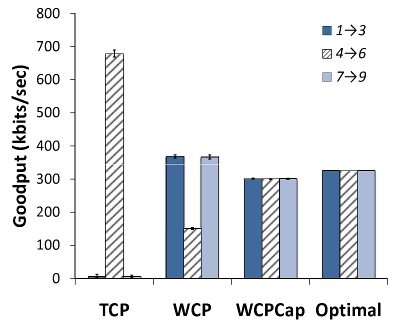


Fig. 10. WCP and WCPCap, Diamond

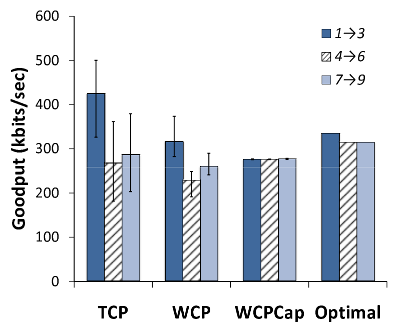


Fig. 11. WCP and WCPCap, Half-Diamond

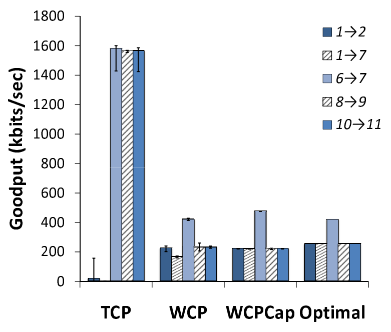


Fig. 12. WCP and WCPCap, Chain-Cross

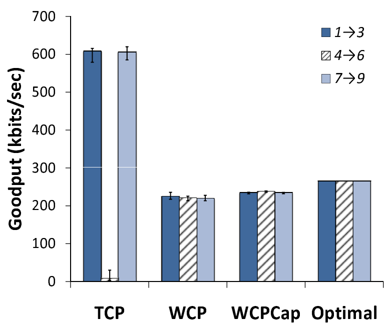


Fig. 13. WCP and WCPCap over lossy links, Stack

starves the flows traversing the most congested link $1 \rightarrow 2$. WCPCap achieves rates within 15% of the optimal max-min rates. WCP achieves rates that depend inversely on the number of congested regions traversed, with $f_{1\rightarrow 7}$ achieving lower goodput and $f_{1\rightarrow 2}$, $f_{10\rightarrow 11}$, $f_{8\rightarrow 9}$ achieving equal rates. WCP is able to utilize available network capacity efficiently; $f_{6\rightarrow 7}$ does not traverse $L_{1\rightarrow 2}$ and gets higher goodput.

Fairness Achieved with WCP . WCP assigns rates inversely proportional to the number of congested regions a flow passes through and the intensity of congestion in these regions. More the number of congested regions a flow passes through and higher the intensity of congestion in these regions, the more will be the congestion notifications received at the source, and lower will be the throughput. Note that the intensity of congestion depends on the local topology of the region and the number of flows passing through the region. As an example, consider Figure 14 where we plot the evolution of flow rates with WCP in the Chain-Cross topology. When the queue at $1 \rightarrow 2$ gets congested, the rate of flows $f_{1\rightarrow 2}$, $f_{1\rightarrow 7}$, $f_{8\rightarrow 9}$ and $f_{10\rightarrow 11}$ is cut by half (for example, at 10s, the rate of all these flows is reduced); and when the queue at $4 \rightarrow 5$ gets congested, the rate of flows $f_{1\rightarrow 7}$ and $f_{6\rightarrow 7}$ is cut by half (for example, at 30s, the rate of both the flows is reduced). Flow $1 \rightarrow 7$ receives congestion notifications from both congested regions, hence it receives a lower throughput than others. Since there are more flows passing through $L_{1\rightarrow 2}$, it has a higher congestion intensity. So, we see more notifications from $L_{1\rightarrow 2}$, and hence a lower throughput for $f_{1\rightarrow 2}$, $f_{8\rightarrow 9}$ and $f_{10\rightarrow 11}$.

The property of assigning rates inversely proportional to the number of bottlenecks a flow passes through is not unique to WCP. In wired networks, the AIMD algorithm of TCP was observed to assign rates inversely proportional to the number of *congested links* a flow passes through [19]. TCP does not retain this property in wireless networks as we observe in the simulation results presented for TCP. By making the AIMD algorithm neighborhood-centric, WCP achieves this property in wireless networks also.

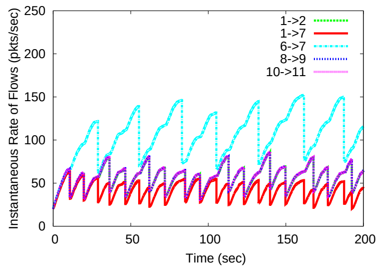


Fig. 14. Evolution of flow rates with WCP in Chain-Cross

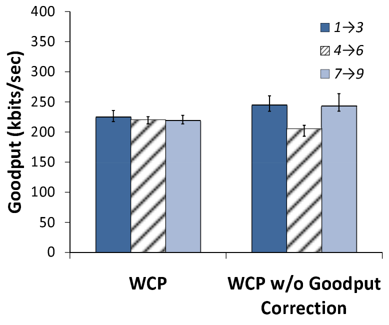


Fig. 15. WCP without goodput correction, Stack

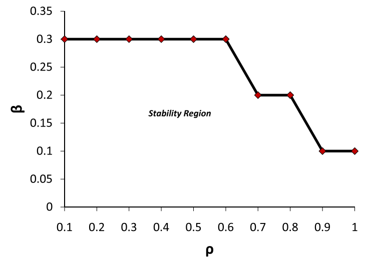


Fig. 16. Stability region of WCPCap. For the parameter values lying below the curve shown in the figure, WCPCap is stable.

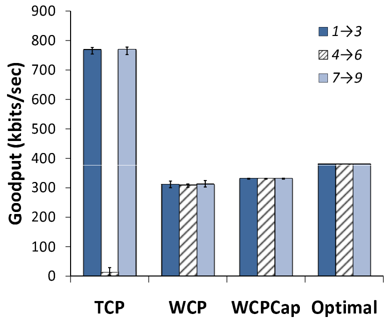


Fig. 17. WCP and WCPCap with no RTS/CTS, Stack

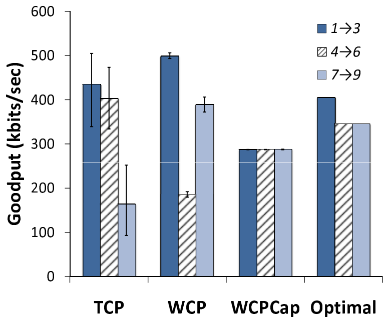


Fig. 18. WCP and WCPCap with no RTS/CTS, Half-Diamond

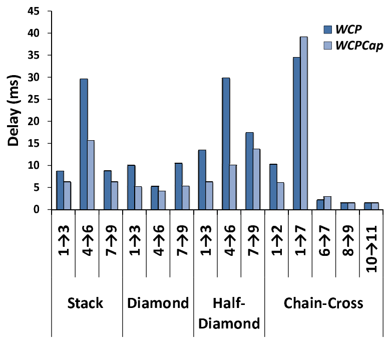


Fig. 19. Average end-to-end delay with WCP and WCPCap

C. Discussion

Impact of Physical Losses. Thus far, we have assumed perfect wireless links in our simulations (losses do occur in our simulations due to collisions, however). Figure 13 shows the performance of WCP and WCPCap for the Stack with a loss rate of 10% on each link. The results are qualitatively similar to Figure 9. As expected, the goodputs drop by about 10% for WCP and WCPCap, as do the optimal rates. We have conducted similar experiments for the other topologies, but omit their results for brevity. We also illustrate the efficacy of goodput correction in dealing with packet loss (Section III-A). Figure 15 shows the goodputs achieved in the Stack topology with 10% loss on all links, when goodput correction is disabled. Goodputs are no longer fair.

Choice of ρ and β in WCPCap . The stability of WCPCap depends on the values of ρ and β. Similar to the stability studies of XCP and RCP in wired networks [16], [29], we consider a network with a single bottleneck. Since a bottleneck in wireless networks corresponds to a congested region, we use the Stack topology which has only one congested region to understand the stability of WCPCap. We run WCPCap for the Stack topology for varying values of ρ and β and determine the values for which it remains stable. Figure 16 shows the stability region of WCPCap. For the parameter values lying below the curve shown in the figure, WCPCap is stable. For all the topologies studied in this paper, any value of ρ and β within the stability region yields nearly similar performance as WCPCap converges quickly. We verify using extensive simulations that a higher and lower value of ρ and β does not impact our results. We omit these results due to space limitations.

Without RTS/CTS Our simulation results have used RTS/CTS so far. Note that the design of both WCP and WCPCap is insensitive to the use of RTS/CTS. Both WCP and WCPCap without RTS/CTS (Figure 17) perform just as well as (and get higher goodputs than) with RTS/CTS (Figure 9). The goodputs increase as the overhead of exchanging RTS/CTS is removed. Other topologies show similar results, except for Half-Diamond (Figure 18). Without RTS/CTS, $1 \longrightarrow 2$ and $7 \longrightarrow 8$ become the most congested links in Half-Diamond, changing the dynamics of congestion in this topology. Qualitatively, this topology starts to resemble the Diamond, with two overlapping congested regions. A lower goodput for $f_{7\to 9}$ than $f_{1\to 3}$ results from additional links $4 \longrightarrow 8$ and $6 \longrightarrow 8$ in $L_{7\to 8}$ which reduces the capacity in the region.

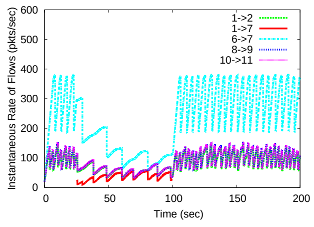


Fig. 20. WCP with delayed flow arrival

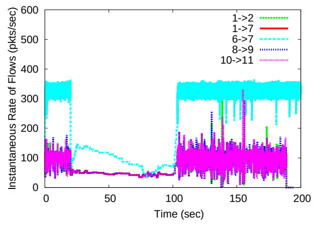


Fig. 21. WCPCap with delayed flow arrival

Performance under Network Dynamics. In the simulations above, all flows start and end at the same time. Figures 20 and 21 show the instantaneous sending rate r_f (Section III) of all the flows in the Chain-Cross topology for WCP and WCPCap respectively, when flows do not start and stop at the same time. Specifically, $f_{1\to 7}$ starts at 25s and ends at

100s, while the rest of the flows start at 0s and end at 200s. It is evident from the plot that both WCP and WCPCap are fair not only when all flows are active, but also before the arrival and after the departure of $f_{1\to7}$. Note that the rate of $f_{6\to7}$ decreases and then increases between 30s and 100s. This variation occurs to clear the queue and adaptively set the value of U at edge $4 \to 5$.

Delay and Convergence. Since WCPCap keeps the network within the achievable rate region, it is able to maintain smaller queues than WCP. Hence, WCPCap has smaller average end-to-end delay than WCP (Figure 19). The one exception is the Chain-Cross: since the throughput of flows $1 \to 7$ and $6 \to 7$ is higher in WCPCap than WCP, the total traffic over $6 \to 7$ is much higher for WCPCap (Figure 12). This results in a higher delay for these two flows.

WCPCap converges quickly as can be seen in Figure 21. It converges to $\theta < 1$ of the final rate within within $\frac{log(1-\theta)}{log(1-\rho)}$ iterations of the algorithm (see the proof of Lemma 2); which implies that for $\rho = 0.3$, it will converge to 90% of the final rate in 7 iterations of the algorithm, which, for all our topologies, takes less than 10s. Also, as is evident from Figure 20, WCP's convergence is slower as the rate per flow is additively increased by a constant α per RTT. Finally, the evolution of flow rates exhibit a sawtooth pattern which is expected from an AIMD algorithm.

Performance with Larger Interference Range. If the interference range is larger than the transmission range, then all nodes sharing a wireless channel (and thus the capacity) may not be able to communicate control information (like congestion status) amongst themselves. In the absence of this control information exchange, WCP's design principle that all flows which contribute to congestion on a link should reduce rates may break which may lead to an unfair rate allocation. Thus, we need extra mechanisms (like exchanging control information exchange between two-hop neighbors or sending control information at a lower rate/better modulation scheme) to ensure successful control message exchange between nodes which share the same wireless channel. We refer interested readers to our technical report [43] for further details as well as results on the performance of WCP for different topologies when interference range is greater than the transmission range. Finally, note that since the methodology of [27] is applicable when the interference range is greater than the transmission range as this only modifies the definition of neighborhood and not any of the analysis, and since WCPCap assumes a complete knowledge of the neighborhood topology, its performance will not suffer. However, it will also need extra mechanisms to ensure the successful exchange of control messages between all nodes sharing the same wireless channel.

Performance with Auto-Rate Adaptation. Auto-rate adaptation affects two characteristics of a topology: Link rates of individual links, and the transmission range of each link. With nodes continuously adapting their link rates, the snapshot of the network at any given instance consists of links with different rate. Since WCP implements fairness at the transport layer, it is able to adapt to different physical rates at the link layer.

However, with change in the link layer rate, the transmission range of a node changes leading to change in the number of nodes that can successfully receive a packet from a node. This causes the topology of the network and, therefore, the neighborhood of each node to change. Since the resulting changes in topology occur at a time scale faster than the convergence time of WCP, WCP is unable to converge to a fair rate. We refer interested readers to our technical report [43] for simulation results and a further discussion. Finally, note that WCPCap assumes that each edge knows the data rate at its neighboring edges, hence, assuming that this information is also passed to neighboring nodes as the rest of the control parameters, the issue is again that of time scales. Even though WCPCap converges faster than WCP, it usually does not converge fast enough to adapt to neighborhood topology changes due to auto rate adaptation in time.

D. Summary

We summarize the throughput properties of WCP and WCPCap observed via simulations in this section. (i) WCP allocates rates that depend inversely on the number of congested neighborhoods traversed by a flow and the intensity of congestion in those regions. (ii) WCPCap allocates to each flow a rate within 15% of the rate allocated to it by the max-min fair rate allocation. The main reason for the difference in throughput achieved with WCPCap and the optimal is the use of $U < 1$ with WCPCap to account for finite buffer sizes and retransmit limits, and the approximation of using only local neighborhood to estimate available capacity. WCP is less efficient that WCPCap because it is an AIMD algorithm. For the Stack topology where the rate vector achieved by WCP has fairness properties similar to the max-min rate allocation, it is within 20% of the optimal. (iii) Finally, WCPCap exhibits low delays and fast convergence.

However, while WCP is implementable (indeed, we describe results from an implementation in the next section), some challenges need to be addressed before the same can be said of WCPCap: the potentially high overhead of control information exchange, and the ability to estimate the amount of interference from external wireless networks so that the collision probabilities can be correctly computed. None of these challenges are insurmountable, and we plan to address these as part of future work.

V. Experiments

We have implemented WCP, and, in this section, report its performance on a real-world testbed. We first validate our simulations by recreating the Stack topology and showing that our experimental results are qualitatively similar to those obtained in simulation. We then demonstrate WCP's performance on a 14 node topology running five flows in a real-world setting.

Our experiments use an ICOP eBox-3854, a mini-PC running Click [37] and Linux 2.6.20. Each node is equipped with a Senao NMP-8602 wireless card running the madwifi driver [1] and an omni-directional antenna. Wireless cards are operated in 802.11b monitor (promiscuous) mode at a fixed transmission rate of 11Mbps with 18dBm transmission power.

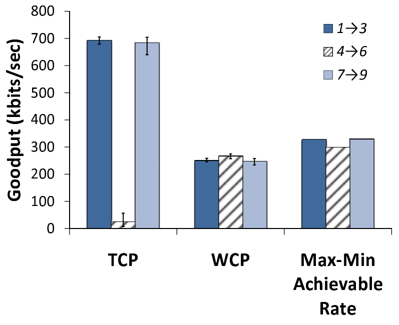


Fig. 22. Results from Stack experimental topology

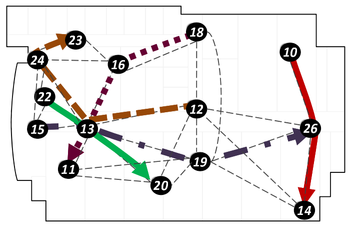


Fig. 23. Arbitrary experimental topology

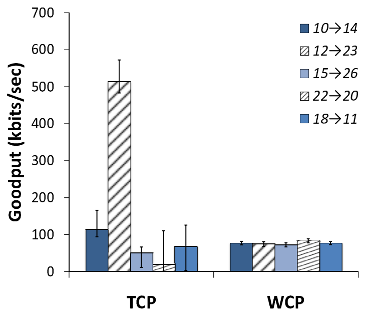


Fig. 24. Results from arbitrary topology

RTS/CTS is disabled for the experiments. We empirically determined, at the beginning of each experiment, that the packet loss rate on each link was less than 10%.

On these nodes, we run *exactly* the same code as in our simulator by wrapping it within appropriate user-level elements in Click. Furthermore, all experimental parameters are exactly the same as in the simulation (Table I), with one exception: we use receiver buffer sizes of 2048 packets so that flows are not receiver-limited. For repeatability, all experiments were performed between midnight and 8am. All our experiments ran for 500 seconds and we show results averaged over five runs.

We re-created the Stack topology by carefully placing nine nodes across three floors, and by removing the antennas on some nodes. Figure 22 shows that the experimental results are similar to the simulation results (Figure 9). Furthermore, WCP achieves goodputs within 20% of an empirically determined maximum achievable rate. (We do not use our theory for determining optimal rates, because we cannot accurately estimate the amount of interference from external wireless networks.) We determine this using CBR flows at pre-configured rates while increasing the rates as long as the goodput of the flow $f_{4\to6}$ is within 10% of the goodput of the other two flows.

Finally, to examine the performance of WCP in a real-world setting, we created an arbitrary topology of 14 nodes by placing them on one floor of our office building (Figure 23). To create a multi-hop topology, we covered antennas of nodes with aluminium foil. On this topology, we ran five flows as shown. Figure 24 shows the end-to-end goodput achieved by the flows. TCP starves $f_{15\to26}$, $f_{22\to20}$ or $f_{18\to11}$ during different runs. By contrast, WCP is able to consistently assign fair goodputs to all five flows in each run of the experiment!

VI. Conclusions and Future Work

Congestion control has vexed networking researchers for nearly three decades. Congestion control in wireless mesh networks is, if anything, harder than in wired networks. In this paper, we have taken significant steps towards understanding congestion control for mesh networks. Our main contributions include: the first implementation of fair and efficient rate control for mesh networks which yields nearly-optimal throughputs; a plausibly implementable available capacity estimation technique that gives near-optimal max-min fair rates for the topologies we study; and, insights into the impact of various factors on performance.

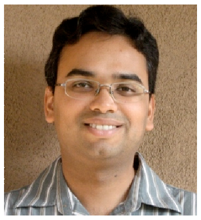
Much work remains. First, we plan to further investigate the kind of fairness achieved by WCP by rigorously defining the intuitive concept of congestion intensity. Second, we intend to investigate efficient implementations of WCPCap. Finally, we intend to explore the impact of short-lived flows and mobility on the performance of WCP and WCPCap.

References

[1] MadWifi. http://madwifi.org/.
[2] MIT Roofnet. http://pdos.csail.mit.edu/roofnet/.
[3] Qualnet. http://www.scalable-networks.com/products/.
[4] F. Abrantes and M. Ricardo. A simulation study of xcp-b performance in wireless multi-hop networks. In *Proc. of Q2SWinet*, 2007.
[5] U. Akyol, M. Andrews, P. Gupta, J. Hobby, I. Saniee, and A. Stolyar. Joint scheduling and congestion control in mobile ad hoc networks. In *Proc. of IEEE INFOCOM*, 2008.
[6] P. Bahl, A. Adya, J. Padhye, and A. Wolman. Reconsidering Wireless Systems with Multiple Radios. *ACM SIGCOMM Computer Communications Review*, 2004.
[7] A. Bakre and B. Badrinath. I-TCP: indirect TCP for mobile hosts. In *Proc. of IEEE ICDCS*, 1995.
[8] H. Balakrishnan, S. Seshan, and R. H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *Wireless Networks*, 1995.
[9] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 2000.
[10] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback based scheme for improving tcp performance in ad-hoc wireless networks. In *Proc. of IEEE ICDCS*, 1998.
[11] H. Chang, V. Misra, and D. Rubenstein. A general model and analysis of physical layer capture in 802.11 networks. In *Proceedings of IEEE INFOCOM*, 2006.
[12] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 1989.
[13] C. Cordeiro, S. Das, and D. Agrawal. Copas: dynamic contention-balancing to enhance the performance of tcp over multi-hop wireless networks. In *Proc. of IEEE ICCCN*, 2002.
[14] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proc. of ACM MobiCom*, 2003.
[15] S. M. Das, D. Koutsonikolas, Y. C. Hu, and D. Peroulis. Characterizing multi-way interference in wireless mesh networks. In *Proceedings of ACM WinTECH Workshop*, 2006.
[16] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown. Processor Sharing Flows in the Internet. In *Proc. of IWQoS*, 2005.
[17] A. Eryilmaz and R. Srikant. Fair Resource Allocation in Wireless Networks using Queue-length based Scheduling and Congestion Control. In *Proc. of IEEE INFOCOM*, 2005.
[18] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *ACM SIGCOMM Comput. Commun. Rev.*, 1996.
[19] S. Floyd. Connections with Multiple Congested Gateways in Packet Switched Networks Part 1: One-way Traffic. *ACM SIGCOMM Comput. Commun. Rev.*, 1991.
[20] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1993.

[21] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on tcp performance. In *IEEE Transactions on Mobile Computing*, 2005.
[22] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla. The impact of multihop wireless channel on TCP throughput and loss. *Proc. of IEEE INFOCOM*, 2003.
[23] G. Holland and N. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proc. of ACM MobiCom*, 1999.
[24] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proc. of ACM SenSys*, 2004.
[25] V. Jacobson. Congestion avoidance and control. In *Proc. of ACM SIGCOMM*, 1988.
[26] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu. Impact of interference on multi-hop wireless network performance. In *Proc. of ACM MobiCom*, 2003.
[27] A. Jindal and K. Psounis. The Achievable Rate Region of 802.11-Scheduled Multihop Networks. *IEEE/ACM Transactions on Networking*, 17(4):1118–1131, 2009.
[28] K. Karenos, V. Kalogeraki, and S. V. Krishnamurthy. Cluster-based congestion control for sensor networks. *ACM Trans. Sen. Netw.*, 2008.
[29] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *Proc. of ACM SIGCOMM*, 2002.
[30] D. Kim, C.-K. Toh, and Y. Choi. TCP-BuS: improving TCP performance in wireless ad hoc networks. *IEEE International Conference on Communications*, 2000.
[31] M. Kodialam and T. Nandagopal. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proc. of ACM MobiCom*, 2005.
[32] S. Kunniyur and R. Srikant. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In *Proc. of ACM SIGCOMM*, 2001.
[33] Y. Li, L. Qiu, Y. Zhang, R. Mahajan, and E. Rozner. Predictable Performance Optimization for Wireless Networks. In *Proc. of ACM SIGCOMM*, 2008.
[34] X. Lin and N. B. Shroff. Joint Rate Control and Scheduling in Multihop Wireless Networks. In *Proc. of IEEE Conference on Decision and Control*, 2004.
[35] J. Liu and S. Singh. Atcp: Tcp for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 2001.
[36] C. Lochert, B. Scheuermann, and M. Mauve. A survey on congestion control for mobile ad hoc networks: Research Articles. *Wirel. Commun. Mob. Comput.*, 2007.
[37] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. *SIGOPS Oper. Syst. Rev.*, 1999.
[38] G. Nychis, Sardesai, and S. Seshan. Analysis of XCP in a Wireless Environment. *Carnegie Mellon University*, 2006.
[39] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. A model based TCP-friendly rate control protocol. In *Proc. of NOSSDAV*, 1999.
[40] J. Paek and R. Govindan. RCRT: rate-controlled reliable transport for wireless sensor networks. In *Proc. of ACM SenSys*, 2007.
[41] B. Radunović, C. Gkantsidis, D. Gunawardena, and P. Key. Horizon: balancing TCP over multiple paths in wireless mesh network. In *Proc. of ACM MobiCom*, 2008.
[42] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. In *Proc. of ACM SIGCOMM*, 2006.
[43] S. Rangwala, A. Jindal, K.-Y. Jang, K. Psounis, and R. Govindan. Neighborhood-centric congestion control for multi-hop wireless mesh networks. Technical report 09-910 (http://www.cs.usc.edu/research/09-910.pdf), University of Southern California, 2009.
[44] A. Raniwala and T. cker Chiueh. Globally Fair Radio Resource Allocation for Wireless Mesh Networks. 2009.
[45] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: a reliable transport protocol for wireless wide-area networks. *Wireless Networks*, 2002.
[46] Y. Su and T. Gross. WXCP: Explicit Congestion Control for Wireless Multi-Hop Networks. In *Proc. of IWQoS*, 2005.
[47] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar. ATP: A Reliable Transport Protocol for Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 2005.
[48] K. Tan, F. Jiang, Q. Zhang, and X. Shen. Congestion Control in Multihop Wireless Networks. *IEEE Transactions on Vehicular Technology*, 2006.
[49] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 1992.
[50] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee. DiffQ: Differential Backlog Congestion Control for Multi-hop Wireless Networks. In *Proc. of IEEE INFOCOM*, 2009.
[51] K. Xu, M. Gerla, L. Qi, and Y. Shu. Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED. In *Proc. of ACM MobiCom*, 2003.
[52] L. Yang, W. Seah, and Q. Yin. Improving fairness among TCP flows crossing wireless ad hoc and wired networks. In *Proc. of ACM MobiHoc*, 2003.
[53] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *IEEE/ACM Trans. Netw.*, 2007.
[54] X. Yu. Improving TCP performance over mobile ad hoc networks by exploiting cross-layer information awareness. In *Proc. of ACM MobiCom*, 2004.

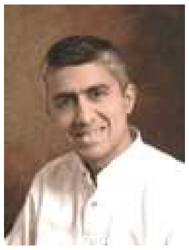
Sumit Rangwala is a Software Engineer at Cisco Systems Inc. He received his B.E. degree from Shri Govindram Seksaria Institute of Technology and Science, Indore, India, and his Masters and Doctorate degrees from University of Southern California all in Computer Science. He was a postdoctoral research associate at WINLAB, Rutgers University. His research interests include protocol design for wireless mesh networks and wireless sensor networks.



Apoorva Jindal is a Member of Technical Staff at Juniper Networks. He received his B.Tech. degree from Indian Institute of Technology at Kanpur and his MS and PhD degrees from the University of Southern California. He was a post-doctoral research fellow at the University of Michigan, Ann Arbor. His research interests include performance analysis and design of protocols for multi-hop networks.

Ki-Young Jang is a Ph.D. student at the University of Southern California, Los Angeles, CA. He received his B.S. degree in Computer Engineering from Hongik University, Seoul, Korea in 2004. His research interests are in wireless mesh and sensor networks.

Konstantinos Psounis Konstantinos Psounis is an associate professor of EE and CS at the University of Southern California. He received his first degree from NTUA, Greece, in 1997, and the M.S. and Ph.D. degrees from Stanford in 1999 and 2002 respectively.



Ramesh Govindan received his B. Tech. degree from the Indian Institute of Technology at Madaas, and his M.S. and Ph.D. degrees from the University of California at Berkeley. He is a Professor in the Computer Science Department at the University of Southern California. His research interests include scalable routing in internetworks, and wireless sensor networks.