Congestion Control in Multi-hop Wireless Networks

Kun Tan, Qian Zhang
Microsoft Research Asia
Beijing, China
{kuntan, qianz}@microsoft.com

Feng Jiang
Huazhong University of Science & Technology
Wu Han, China

Xuemin (Sherman) Shen
University of Waterloo
Ontario, Canada
xshen@bbcr.uwaterloo.ca

***Abstract*—This paper focuses on the congestion control problem in a static, multihop, wireless network. Conventional congestion control algorithm, such as TCP, suffers from the performance degradation in multihop wireless networks, due to the characteristics of the shared wireless media. Lack of precise congestion indication and coordination among nodes that compete for the share wireless channel, TCP fails to allocate resource efficiently and fairly among flows.**

In this paper, we present a novel Explicit Wireless Congestion Control Protocol (EWCCP) for stationary multihop wireless networks. By exploiting explicit coordination and multi-bit explicit feedback from routers, EWCCP gains fine-grain control and is robust to the dynamics of the wireless channel. EWCCP stabilizes at a lower but more optimal sending window regarding to TCP, and achieves low buffer occupation and low delay. With explicit coordination, EWCCP allocates resource fairly among flows that compete for the shared channel. With an analytical model, we show that EWCCP achieves proportional fairness in multi-hop wireless networks.

Extensive simulations demonstrate the effectiveness of EWCCP. It is shown that EWCCP is a viable congestion control scheme for multihop wireless networks.

***Index terms*-- Congestion control, multihop wireless network**

I. INTRODUCTION

This paper considers the problem of congestion control over stationary multihop wireless networks. In such networks, radio equipped nodes can communicate with their neighbors directly when they are within the radio transmission range. Two nodes that are far away from each other may rely on intermediate nodes to relay traffic. We are more interested in multihop wireless network based on IEEE 802.11 (WiFi) technology, since it is a *de facto* standard for wireless LAN (WLAN). Many commercial applications have been built based on multihop WiFi networks. One such example is the "community mesh networks" [22][23].

Congestion control is a critical issue to ensure the efficient and fair allocation of network resource among communication flows. In the Internet, this concern issue has been resolved by applying end-to-end congestion control algorithm, i.e. Transmission Control Protocol (TCP) [20]. However, it has been reported that TCP does not perform well in a multi-hop wireless environment [14], which usually results in inefficient and unfair bandwidth allocation among different flows.

One fundamental difference between multihop wireless networks and the wired networks is that the wireless channel is a spatially shared resource. Wireless nodes within the interference range compete for the same wireless channel, and only one transmission is allowed at the same time. Conventional TCP, using packet losses as implicit congestion feedbacks, adversely interacts with this characteristic of multihop wireless network. Firstly, TCP greedily increases the sending rate until packet loss occurs. This behavior keeps high level buffer occupation on wireless nodes and therefore increases the contention in the multihop wireless network, which consequently causes "wastage" in the broadcast medium of wireless [15]. Secondly, TCP observes congestion only by the packet loss on its own path. However, in a multihop wireless network, two TCP flows, even without any common node, may compete for the same wireless channel. With different locations, these competing flows may have largely different perceptions on the congestion (i.e. packet loss/delay). Without any coordination mechanism, some unlucky flows may always experience more packet losses (or delay) and reduce sending rate more frequently, while others get significant large share of bandwidth.

Previous works have proposed to resolve the above two issues separately, by adding explicit feedback control, i.e. RED/ECN [17], into multihop wireless networks [12][15]. However, RED/ECN scheme provides only one-bit information about the congestion state. We know that with one-bit feedback, TCP is prone to be unstable with large feedback delay [2]. As shown [12], the feedback delay of ECN in a multihop wireless network is generally large, i.e. from 100ms to 1s, in order to avoid too rugged estimations or reduce overhead. Large oscillation in sending rate may cause corresponding oscillation in network queues. This instability may cause many potential problems in multihop wireless networks. For example, it may cause under-utilization of wireless channel. As the network queues jump from empty to near full, the state of wireless channels also jump from idle to heavy contentions. Either case reduces the throughput. Moreover, coordination is required among flows that may interfere with each other to ensure the fairness[1]. The key here is to make these flows be aware of congestion state of each other. Reference [15] suggests implicitly coordinating congestion observation using

This work is done when Feng Jiang was visiting MSRA as an intern at Wireless and Networking group.

[1] We abuse the term of *interfere* here and say that two flows interfere with each other if any two links belong to them are interfere with each other.

passive monitoring of neighbors' transmission. However, accurately monitoring channel utilization is difficult in practical systems. Not only conventional hardware usually discloses the critical information from upper layer, but noisy environment of wireless networks, e.g. wireless loss and collisions, also brings challenges for precise channel measurement.

In this paper, we argue that efficient and fair bandwidth allocation can be achieved by applying *explicit multi-bit feedback* and *explicit signaling* in multihop wireless networks. By *explicit multi-bit feedback*, the congestion control protocol is more robust to feedback delay. Multi-bit feedbacks give fine-granularity control for senders and are able to stabilize the sending rate at an value that can efficiently utilize the wireless channel resource, while achieves low buffer occupation and low queuing delay. By *explicit signaling*, wireless links that mutually interfere *explicitly* exchange congestion information, i.e. average queue size information, with management packets. We show later that, with proper chosen signaling interval, explicit signaling the queuing information provides a simple yet precise view of wireless congestion, but with acceptable overhead.

This paper presents the design of a novel Explicit Wireless Congestion Control Protocol (EWCCP) for multihop wireless networks. The main contributions of EWCCP are:

1. EWCCP provides a systematic design of a congestion control protocol that achieves both efficient and fair allocation of bandwidth among flows by using i) explicit multi-bit congestion feedback from routers, which is computed based on the coordinated congestion information; and ii) using explicit signaling protocol to coordinate flows that contend for the shared wireless channel.
2. We derive an analytical model of EWCCP, by extending F. Kelly's model [4] to multihop wireless network. Our analysis demonstrates that EWCCP achieves a proportional fair allocation among flows.
3. We conduct extensive packet-level simulations on IEEE 802.11 DCF MAC and show that EWCCP performs well with commercial available IEEE 802.11 hardware. We believe EWCCP is a viable congestion control scheme for multihop wireless networks.

The rest of the paper is organized as follows. We briefly present the background in next section. Section III presents the protocol details. Analytical model of EWCCP is presented in Section IV. We conduct extensive simulations to evaluate our scheme in Section V. Related works are discussed in Section VI. We discuss some related issues and further extension of this work in Section VII. Section VIII concludes our work.

II. BACKGROUND

In this paper, we consider a static, multihop, wireless network based on IEEE 802.11 technology, which uses a CSMA/CA MAC. Each packet may be preceded by an exchange of RTS/CTS control messages to reserve the channel time for sending the data packet and an ACK. Upon overhearing the handshake, the nodes within the interference range of both the sender and receiver defer their transmissions. Therefore, two wireless links will contend with each other if an end-point of one link is within the interference range of end-points of the other link. For example, in Figure 1, link 5~6[2] is interfering with link 1~2, 3~4, 7~8, and 9~10, etc. We term a wireless link's *interference set* as the set of links that interfere with that link. For example, in Figure 1, link 5~6's interference set may contain all links shown in the figure.

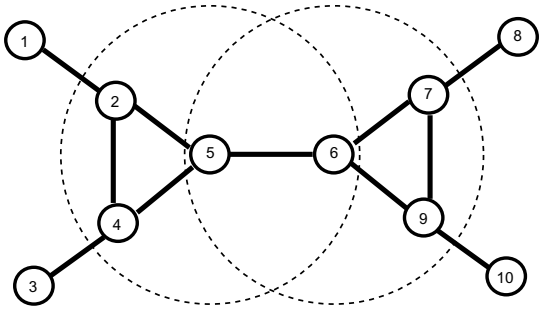

Figure 1. Illustration of wireless interference of a link. The dash circles present the interference range.

Links included in one interference set share the same channel resource. Transmissions on one link contribute the congestion over all links in its interference set. Therefore, congestion control protocols for multihop wireless networks must give feedbacks based on the congestion state in the entire interference set, rather than individual links. The congestion in the interference set can be measured by the aggregated queue size over all links in the entire interference set. We define this aggregated queue as *a neighborhood queue* of the link.

We note that in this work, we simply assume that the interference range equals to the transmission range. Therefore, the interference set of a link includes all links that contain the immediate neighbors of the sender and receiver of that link. Although we make the above assumption for sake of description, it does not prevent our scheme to be extended to the case where the interference range is larger than the transmission range (see later in Section VII).

III. EXPLICIT WIRELESS CONTROL PROTOCOL

In this section, we present the detailed design of EWCCP for multihop wireless networks. EWCCP is a joint design of the end-systems as well as the intermediate routers[3]. EWCCP routers are responsible to give congestion feedbacks by coordinating among links that mutually interfere. The congestion feedbacks are provided based on the size of the neighborhood

[2] Throughout this paper, we use #~# to denote a un-directional link; while we use #-# to denote a directional link, where the former one is the sender and the latter one is the receiver.

[3] Note that in multi-hop wireless network, there is no strict separation for end systems and core routers. We use these terms just for sake of description.

queue defined in previous section. EWCCP mimics *additive increase and multiplicative decrease* (AIMD) control scheme. That is, when congestion is detected, different flows get a portion of negative feedback proportionally to their sending rate. On the other hand, if the network is not congested, every flow will get a same amount of positive feedback. We show later in Section IV, this AIMD control scheme converges to a proportional fairness allocation.

A. Framework

EWCCP sender maintains a congestion window (or simply *window*), which controls the maximum number of packets that are allowed to be sent before receiving an acknowledgement[4]. Every data packet carries a special congestion header, which can be annotated by the intermediate routers. Figure 2 shows the format of a congestion header. There are two parts. One contains end-to-end congestion control information; the other contains the control information for one link. The light grayed field is filled by end systems and the dark grayed fields are updated by routers. EWCCP senders and receivers only care about the end-to-end congestion header. Link congestion header is used between routers. We will return to this part later in Section III.B.2.

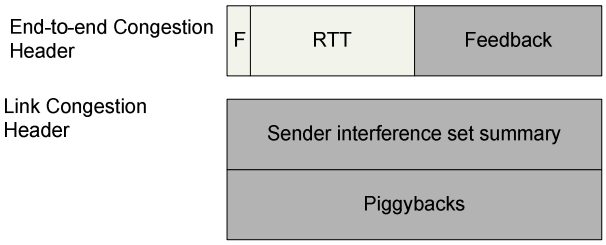

Figure 2. Congestion header.

The intermediate routers calculate the congestion feedback based on the size of neighborhood queue on each wireless link and add the feedback in the congestion header of the packet. When the packet arrives at the receiver, the feedback field in the congestion header holds the sum of all feedbacks given by all wireless links along the path. This aggregated congestion feedback is echoed back to EWCCP sender with an acknowledgement from the receiver. Then, the sender can adjust the sending window based on the routers' feedbacks. An *acknowledgement* conveys both positive feedback and negative feedback. However, since EWCCP adopts AIMD, the positive feedback is fixed, which can be easily calculated at end systems. Thus, only negative feedback requires explicit calculation on intermediate routers.

Intermediate EWCCP routers periodically exchange their local queue information with their neighbors. To completely collect the neighborhood queue information, one node may need to know the local queue information of nodes that are multiple hops away. For example, in Figure 1, if node 5 wants to calculate congestion feedback, it may need to know the queues on node 7 and 8, since link 7-8 and 8-7 interfere with link 5-6. One simple way to do so is to propagate information by re-broadcasting it for several times. Such multiple re-broadcasting increases not only overhead, but also delays. Fortunately, in EWCCP, we develop a technique called *separated control*, which allows EWCCP routers to correctly calculate congestion feedback without complete knowledge of *neighborhood queue*. With this, local queue information only needs to be broadcast within one hop neighbors. We will discuss details in Section III.B.2.

B. EWCCP Router Behavior

An EWCCP router collects the neighborhood queue information and calculates congestion feedbacks. The router makes the control decision once with a control interval d, called a control *round*. As aforementioned, routers broadcast their local queue information periodically, say, with an interval d_s. From the control theory, the control interval should not be less than signaling interval, i.e. one must see the results of a control before making another control decision. Therefore, we set $d >= d_s$[5].

1) Congestion detection

Congestion is detected on one wireless link if the size of neighborhood queue is greater than a predefined threshold in last control interval. In other words, congestion is detected if (1) satisfied.

$$\hat{Q}_{i,j} = \sum_{l \in IS(i-j)} Q_l > \gamma, \qquad (1)$$

where $\hat{Q}_{i,j}$ stands for the neighborhood queue on wireless link *i-j*; Q_l is the average queue size (in bytes) on link l in last *round*; IS(.) is the *interference set* of the given link and γ is the desired queuing level. The rule of thumb for setting γ is to let it equal to the pipe size of the wireless link. For IEEE 802.11, it is a stop-wait protocol as it adopts a DATA/ACK sequence. The pipe size over one hop is therefore at most one packet, irrelevant to the link speed [26][28]. Also it is desirable to keep only one link active within an interference set. Therefore, keeping one packet backlogged in the neighborhood queue would efficiently utilize the channel with minimal interference, so we set γ = 1MSS (Maximal Segment Size).

2) Negative feedback calculation

EWCCP is a window-based protocol. The event of receiving an acknowledgement is regarded as a positive feedback, and the "feedback" field in the congestion header conveys the

[4] EWCCP chooses to use window-based control scheme rather than a rate-based control. The reason is that wireless channel is actually a very dynamic medium. Window-based control is more stable because of the beauty of self-clocking. Note that EWCCP also adopts a pacing technique that evenly spread transmission to avoid burstiness [25].

[5] Note that the control interval should also be larger than *rtt* of any flow. Here we assume that the control interval is much larger than *rtt*. Our simulations will verify that EWCCP usually keeps *rtt* at very small level.

negative feedback. The *window* is updated based on both types of feedbacks, as we will show details in Section III.C.1.

When a control interval passed, the negative feedback on a wireless link is calculated if congestion is detected according to (1) in last *round*. Assuming the neighborhood queue size is $\hat{Q}_{i,j}$, we choose the total negative feedback on the interference set is proportionally to $\hat{Q}_{i,j}$. It is reasonable because higher $\hat{Q}_{i,j}$ means heavier congestion, and therefore we may decrease sending rate more quickly. We denote $\varphi_{i,j} = \beta\hat{Q}_{i,j}$ is the total negative feedback, and the next step is to allocate it to each flow that passes the interference set. Note that in EWCCP, each data packet carries a congestion header, and routers provide congestion feedback by updating the corresponding field in each congestion header. Therefore, we need to calculate feedback for each packet, which we will develop step by step below.

Firstly, we derive the total negative feedback on a wireless link. As EWCCP applies *multiplicative decrease* control, the negative feedback for each flow should be proportional to its throughput. Consider the link *(i-j)*. Let's denote $x_{i,j}$ the aggregated flow rate over that link. Then, the total amount of negative feedback for link *(i-j)* can be calculated using (2)

$$\phi_{i,j} = \varphi_{i,j} \cdot \frac{x_{i,j}}{\sum_{k-l \in \mathrm{IS(i-j)}} x_{k,l}}, \tag{2}$$

where IS(.) is the *interference set* of the given link.

Secondly, we divide the total negative feedback on the link, $\phi_{i,j}$, into each flow that transverses the link. The total decrease in the aggregated sending rate on that wireless link is ($\phi_{i,j}/d$). Consider flow *f* that traverses link *(i-j)*. We denote w_f the *window* of *f* and rtt_f the average round trip time of *f*. Then, the sending rate of *f* can be approximately presented as $r_f = w_f / rtt_f$. The desired decrease in the rate of flow *f* in the next interval is

$$\Delta r_f = \frac{\Delta w_f}{rtt_f} = \frac{\phi_{i,j}}{d} \cdot \frac{r_f}{x_{i,j}} = \frac{\phi_{i,j} \cdot w_f}{d \cdot x_{i,j} \cdot rtt_f}. \tag{3}$$

Lastly, we divide the feedback into each packet. The total number of packets from flow *f* in one interval *d* is approximately

$$N_f = w_f \cdot d / rtt_f / s_f, \tag{4}$$

where s_f is the packet size of flow *f*, and *d* is the control interval.

Then, the per-packet negative feedback on *window* n_f (in bytes) is shown in (5).

$$n_f = \frac{\Delta w_f}{N_f} = \frac{\phi_{i,j}}{d^2 \cdot x_{i,j}} \cdot rtt_f \cdot s_f = \beta \frac{\hat{Q}_i}{d^2 \cdot \sum_{j \in IS(i)} x_j} \cdot rtt_f \cdot s_f. \tag{5}$$

Note that when there is congestion and queues are built-up, $\sum_{k-l \in \mathrm{IS(i-j)}} x_{k,l}$ can be approximated by ψC, where *C* is the link speed of the wireless channel and ψ is a constant that counts for the MAC layer over-head. Therefore, we can simplify (5) as (6).

$$n_f = \beta \frac{\hat{Q}_{i,j}}{\psi \cdot C \cdot d^2} \cdot rtt_f \cdot s_f, \tag{6}$$

where *d* is the control interval, rtt_f is the round trip time carried in the packet congestion header.

Note that using (6) to compute the congestion feedback, all factors except $\hat{Q}_{i,j}$ are constant to a specific flow. Further, $\hat{Q}_{i,j}$ is a linear sum of all queues in the interference set. Recalling the example in Figure 1, it does not require that node 5 to know the whole neighborhood queue information. Node 5 can compute the negative feedback based on the partial knowledge of the neighborhood queue (e.g. the queue on link 1~2, 3~4, 2~3, etc.). When the packet is received by node 6, it can further compute another part of negative feedback (i.e., containing the queue information of link 7~8, 9~10, 7~9, etc). The sum of these two partial results is the complete negative feedback got from link 5-6.

We call this technique the *separated control*. The *separated control* allows nodes to perform feedback calculation with only partially knowledge of the neighborhood queue, and thereby reduces the propagation scope of the queuing information, which greatly reduces the signaling overhead.

Link congestion header, in Figure 2, is used to carry the information for *separated control*. More specifically, the link congestion header contains a hashed summary vector for IDs of links in the interference set that are known to the link sender (Note there is also a "piggyback" field, which will be detailed in Section III.D.). The summary vector can be generated using Bloom filters [27]. When the link receiver receives a data packet, it checks all links it knows in the interference set against the summary in the packet. If the link is contained in the summary, the receiver just skips it, as the link sender has already counted this link in its feedback. All other links then are considered and the link receiver feedbacks are calculated according to (6). Finally, the feedbacks are added back to the end-to-end congestion header to complete the feedback of the link.

3) Start-up probe

When a connection is started, EWCCP initializes the *window* value to be one MSS, and only one packet is allowed to be sent out. The first data packet also serves as a *probe packet* with a mark in the *flag* field in the congestion header. Probe packets are treated specially by routers. The feedback field of a probe packet is filled by routers for a proper initial *window*. And when an acknowledgement for a probe packet is received, the sender directly sets current window to the value suggested by the routers, instead of slowly probing through linear increasing. For any flow, only one probe packet is allowed to be sent at any time.

EWCCP router uses a very simple yet effect heuristic to set proper starting window. It is still based on the observation that the pipe size of an 802.11 link is at most one, irrelevant to the link speed. Therefore, if the router does not detect congestion, it will increase the feedback of a probe packet by one. However, if local congestion is detected, the router should turn off the *flag* field, and give congestion feedback as usual.

C. End-system behaviors

1) EWCCP sender

As aforementioned, an EWCCP sender maintains a *window* and a smoothed *rtt*. The sending rate of the sender can be estimated as (*window/rtt*). With a timer, the sender evenly spreads the window worth of data within *rtt*.

When a connection is started, EWCCP initializes the *window* to be one MSS, and sends the first packet. When an acknowledgement for this packet is received, the sender may directly set current window to the value suggested by the router (see Section III.B.3).

After initial startup, the EWCCP sender updates *window* according to the acknowledgement received. An we discussion before, the event of receiving an acknowledgement conveys a fixed positive feedback; while the "feedback" field in the congestion header contains the negative feedback from routers. Therefore, the *window* should be updated according to

$$w = w + p - n, \tag{7}$$

where

$$p = \alpha \cdot rtt_f \cdot s_f^2 / (w \cdot d) \tag{8}$$

and n is the feedback carried in the packet congestion header.

Note that the increase part calculated in (8) implements the *additive increase*. However, (8) is different from the one used in TCP, in which the additive increase is scaled with *rtt*, and therefore, flows with longer *rtt* are treated with bias. EWCCP makes the increase scale with the control interval of the routers, and thus removes the *rtt* unfairness. α is the parameter that controls the increase step. If $\alpha = 1$, it means that the flow increases one MSS for each control interval.

2) EWCCP receiver

An EWCCP receiver is rather simple. It only receives data packets and copies the end-to-end congestion header from data packets to acknowledgements.

D. Signaling protocol

EWCCP uses explicit signaling to coordinate with neighbor nodes. It periodic broadcasts the average queue information. The broadcasting is limited only within the one-hop neighbors.

The broadcast packet contains average queue size in last round of all links on the node. It includes not only the links that go out, but also includes those getting into the node. Note that the queue information of a link is usually available only at the link sender, and we need to make it also available at the link receiver. The trick we play here is to use "piggyback" field in the link congestion header, as shown in Figure 2. When a data packet is transmitted, the queue information of the link is also delivered to the link receiver. Then, the receiver can propagate it out with broadcasting.

Each node maintains a neighbor queue table. One example is shown in Figure 3. A link is uniquely identified by ids of the sender node and receiver node. Each entry has following three types: a) "*O*" (own), presenting a queue on a downstream link; b) "*P*" (piggyback), presenting a queue that is piggybacked from an upstream link; c) "*B*" (broadcast), presenting the other information learnt from receiving broadcast packets. The sequence number is used to judge the freshness of the entry. A node can only increase the sequence number of the entries that marks "*O*".

There is a tradeoff in the interval between two successive broadcasting. Frequent broadcasting would give more timely information, but also consumes more bandwidth. We believe signaling interval at magnitude of hundreds milliseconds is an acceptable overhead. We evaluate this with a simple calculation. Let's assume one node may have average 10 neighbors in one-hop and the signaling interval is 100ms. Each entry in the queue table may cost 6 bytes (i.e. 2 bytes link id, 2 bytes queue size, 1 byte sequence number and 1 byte flag). The total bandwidth for signaling is (10+1)*(2*8*10*6)*10 =105.6Kbps, which is around 1% of 11Mbps channel of 802.11b without counting the MAC overhead. Note that a multiplier 2 is applied in above calculation because wireless links may be bi-directional, and therefore both incoming and outgoing queue are broadcasted.

The signaling protocol is summarized as follows.

1) When a node sends a data packet along a downstream link, it fills the average queue size of that link in the piggyback field in link congestion header.
2) When a signaling interval is passed, a broadcast packet is generated and sent. The broadcast packet contains all entries in the table which marks "*O*" and "*P*".
3) When receiving a broadcast or a data packet with link congestion header, a node updates the neighbor queue table with the information contained. If an entry already exists, the one with larger sequence number wins.

Src ID	Dst ID	Type	Value(kB)	Seq #
5	6	O	0.1	6433
2	5	P	0.4	2201
2	1	B	1	1000

Figure 3. An example of neighbor queue table.

IV. ANALYTICAL EWCCP MODEL

With the detailed design of EWCCP, it is important for us to understand EWCCP in an analytical way.

In next subsection, we will first brief the network model. After that, in subsection B, we give the formulation of the congestion control based on Kelly's framework [4], and extend it to wireless networks, where the wireless interference between links is considered. Finally, we show that the EWCCP control law designed in (6) and (7) is the solution of the problem that we have formulized.

A. Network model

We consider a wireless network with *N* nodes. Let *L* denote the set of node pairs *(i,j)* such that transmission from node *i* to node *j* is allowed. Such a node pair is also called a link. Due to the shared nature of the wireless media, the capacity for a specific link *(i,j)*, c_{ij}, depends not only on the transmission power, modulation, but also on the interference with other links. We simply assume that all nodes use the same power and modulation method for any transmission, and also the interference only happens between one hop neighbors of a node. We denote $\vec{c} = \{c_{ij}, (i,j) \in L\}$ a feasible rate allocation of links in wireless network, and assume *R*, the convex hull of $\vec{c}$, is closed and bounded.

B. Congestion control problem

We denote *S* the senders of a flow and *D* the destinations. Further, we assume each user has only one flow, so that we use *s* to denote both a user and a flow in the network. Let *A* denote the routing matrix, i.e., $A_{i-j,s} = 1$, if flow *s* traverses link *i-j*.

Following [4], the congestion control problem can be modeled as a non-linear programming problem *P*:

$$\max \sum_s U_s(x_s) \tag{9}$$

subject to

$$\sum_s A_{i-j,s} x_s \le c_{i,j}, \tag{10}$$

and

$$[c_{i,j}] \in R. \tag{11}$$

$U_s(.)$ is the utility function of user *s*, and x_s is the rate of user *s*. However, unlike the wired network, the capacity on a wireless link is not fixed but changes depending on the rate on the links that can interfere with it. If two links are within interference range, they can not transmit together. Note that 802.11 DCF MAC is a random access protocol. Without synchronization, DCF can not well schedule the transmission on each link. Therefore, we only consider the lower bound of $c_{i,j}$ as

$$c_{i,j} \approx C - \sum_{(l-k) \in \mathrm{IS}(i-j)/(i-j)} c_{l,k} \Rightarrow \sum_{(l-k) \in \mathrm{IS}(i-j)} c_{l,k} \approx C, \tag{12}$$

where *C* is the link speed of the wireless channel.

We define *G* as the interference matrix, i.e. $G_{(i-j),(l-k)} = 1$, if link *(i-j)* and *(l-k)* interference with each other. Let $H = GA$. Using (12), the problem *P* can be rewritten as

$$\max \sum_s U_s(x_s)$$

$$\text{subject to } \sum_s H_{(i-j),s} x_s \le C. \tag{13}$$

We use the Lagrange relaxation to solve the problem defined in (13). Consider the Lagrangian in (14),

$$L(x,\lambda) = \sum_s w_s U_s(x_s) + \sum_{(i-j) \in L} \lambda_{i,j} (H_{(i-j),s} x_s - C) \cdot \tag{14}$$

The Lagrange relaxation of problem *P* can be defined as problem *Q*:

$$\max \sum_s V_s(x_s)$$

$$\text{where } V_s(x_s) = w_s U(x_s) - \sum_{(i,j) \in P_s} \int_0^{\sum_{(k-l) \in IS(i-j)_s} x_{k,l}} \lambda_{i,j}(y) dy, \tag{15}$$

and $\lambda_{i,j}(y)$ is nonnegative, continues, convex and increasing function of rates on links that cross the interference set of link *(i-j)*. P_s is the path that flow *s* transverses. $\lambda_{i,j}(y)$ can also be interpreted as the penalty function which is the unit price for transmission over the link *(i-j)*. Equation (15) also states that each flow can maximize its own $V_s(x_s)$, so that the system wise optimal is also obtained [3].

C. EWCCP Solution

Recall the control law represented by (7) and (8) in Section III.B, which adjusts *window* based on per packet feedback (both positive and negative). Translating (7) and (8) into fluid model, we have

$$\dot{x}_s = \alpha - \beta \cdot x_s \sum_{(i-j) \in P_s} \frac{\hat{Q}_{i-j}}{d \cdot C}, \tag{16}$$

where $\hat{Q}_{i,j}$ is the neighborhood queue of link *(i-j)*, *d* is the control interval, and P_s is the path that flow *s* transverses.

For a specific wireless link *(i-j)*, the average local queue size in the last control interval is approximately

$$Q_{i,j} \approx (x_{i,j} - c_{i,j})^+ \cdot d, \tag{17}$$

where $(.)^+$ is defined as max (0, .).

Substitute (17) into (16), we rewrite the control law as

$$\dot{x}_s = \alpha - \beta \cdot x_s \sum_{(i-j) \in P_s} \frac{\sum_{(k-l) \in IS(i-j)} (x_{k,l} - c_{k,l})^+}{C}. \tag{18}$$

Using $\sum_{(k-l) \in IS(i-j)} x_{k,l} \approx C$, and (12), we have

$$\dot{x}_s = \alpha - \beta \cdot \sum_{(i-j)\in P_s} x_s \frac{\left(\sum_{(k-l)\in IS(i-j)} x_{k,l} - C\right)^+}{\sum_{(k-l)\in IS(i-j)} x_{k,l}}. \quad (19)$$

Let $\lambda_{i,j} = \frac{(\sum_{(k-l)\in IS(i-j)} x_{k,l} - C)^+}{\sum_{(k-l)\in IS(i-j)} x_{k,l}}$, (20)

(19) can again be rewritten as

$$\dot{x}_s = \alpha - \beta \cdot x_s \cdot \sum_{(i-j)\in s} \lambda_{i,j} \cdot \quad (21)$$

Note that the congestion-control scheme (21) converges to the unique solution to problem defined in (15), where $\lambda_{i,j}(y)$ is taken the form in (20), and $U(x_s) = \log(x_s)$. The proof simply follows the lines in [3]. We know that *the proportional fairness is represented by the utility function log(.)*, and therefore, EWCCP control law achieves *proportional fairness* in multi-hop wireless networks.

V. Performance Evaluation

We have implemented EWCCP in NS2 simulator [32] and conducted extensive packet-level simulations. We demonstrated that EWCCP achieved the efficient and fair resource allocation in a multi-hop wireless environment. In the following tests, we used 802.11a DCF with RTS/CTS turned on. The channel speed was set to 54Mbps. In comparison with TCP, we did not consider any *error model* in wireless channel. We used TCP SACK and DSR [18] with robust link failure detection[6]. The parameter setting for EWCCP was $\alpha = \beta = 1$. Actually, we found with simulations, the performance of EWCCP was not sensitive with this parameter setting. We set both the signaling interval and the control interval of EWCCP to 100ms and all packet size was set to 1000 bytes.

A. Single chain topology

We started our simulation from simple chain topology as shown in Figure 4. One flow was running end-to-end from node 1 to node 8, passing 7 hops. The instantaneous throughputs of EWCCP as well as TCP are shown in Figure 5, and the *window* evolutions of both TCP and EWCCP are drawn in Figure 6. The throughput, as well as the *window*, of TCP has much larger variance comparing to that of EWCCP. In Table 1, the overall throughput, congestion window, *rtt* and link layer packet losses of the two schemes are summarized. In this simple scenario, although TCP had a much larger window than EWCCP, the total throughput of TCP was even slightly less than that of EWCCP. The reason can be explained by examining the last two columns of each protocol. We see that TCP has much larger RTT value comparing to EWCCP. This means that most of packets in one window were buffered somewhere in the network, and therefore greatly increased the queuing delay. Pushing too much packets into the network drives the wireless network into saturation, which further increases the link layer losses due to heavy contention. Note that contention causes not only packet losses, but also backoff in MAC that wastes channel time. In contrary, with multi-bit explicit feedback, EWCCP kept the queue in neighborhood small and stabilized the sending window at the value that efficiently utilized the channel resource, but maintained very small queuing delay.

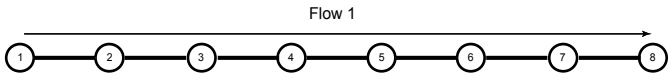

Figure 4. The chain of 7-hops.

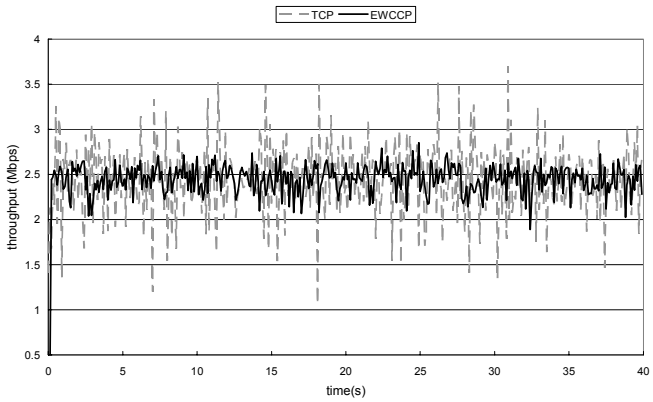

Figure 5. Instantaneous throughputs of EWCCP and TCP over a chain topology.

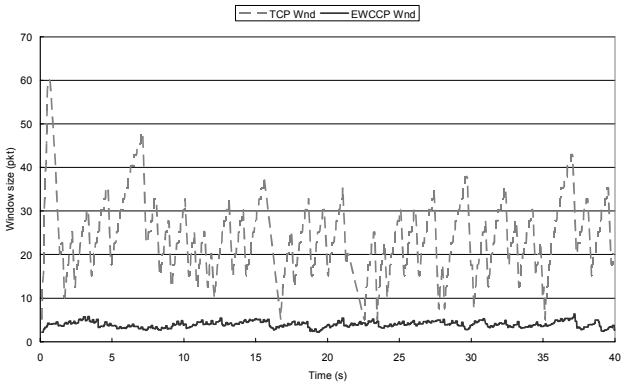

Figure 6. The window evolution of TCP and EWCCP over a chain topology.

[6] Our implementation declares a link failure only after severe packet loss (i.e. consecutive 3 packets are lost) is detected on that link. This is unlike current implementation in NS2 2.27, which declares a route failure on the first loss of a packet. It is reasonable to have such robust link failure detection in a stationary multihop wireless network, as link failure is rather uncommon and single packet loss may be caused by random interference.

	Ave. throughput	Ave. cwnd	Ave. RTT	Pkt loss #
EWCCP	2.35Mbps	3.5pkts	0.01s	1pkt
TCP	2.32Mbps	24.5pkts	0.12s	30pkts

Table 1. Throughput, window and rtt for EWCCP and TCP over a chain topology.

B. *Exposed terminal topology*

We tested in this section a simple yet typical that TCP demonstrated great unfairness because of the lack of coordination in competing flows on different nodes, as shown in Figure 7. We also evaluated EWCCP in this topology. Figure 8 and Figure 9 shows the instantaneous throughput of both EWCCP as well as TCP. As we expected, TCP 2, at a better location to compete the wireless channel eventually starved TCP 1. However, EWCCP controled sending window based on the multi-bit feedback computed from the aggregated queue over the interference set. Therefore, even though EWCCP 2 could easily obtain the channel, it gracefully retreated when queue is building at node 1, and therefore yield channel time for EWCCP 1. As a consequence, these two flows shared the channel fairly. Figure 10 shows the overall throughput of each flow and the aggregated throughput of TCP and EWCCP. Note that with multi-bit feedbacks, EWCCP still provided rather stable control in this heavy contention situation, where previously proposed RED/ECN scheme [12] demonstrates great oscillations. As shown in [12], although the distributed RED/ECN helps to improve fairness in average throughput, the instantaneous throughputs of two flows alternatively oscillated from zero to the maximum that occupies the whole channel resource.

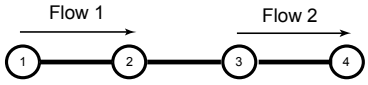

Figure 7. The exposed terminal topology.

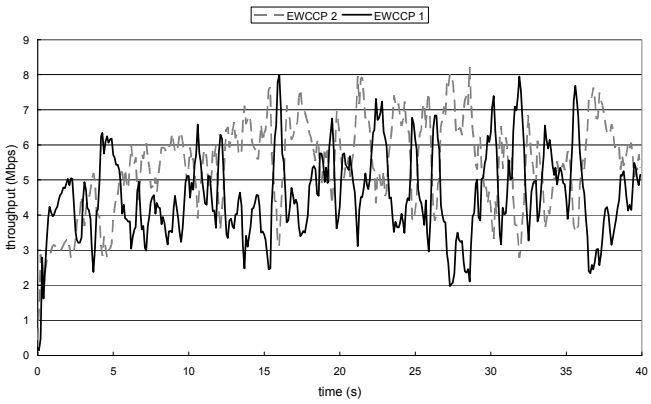

Figure 8. Instantaneous throughput of EWCCP over exposed terminal topology.

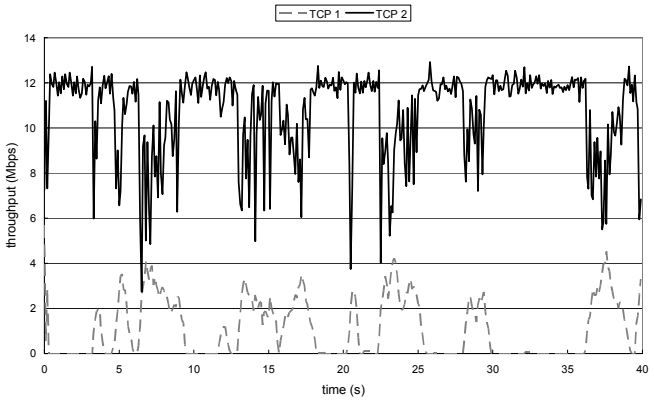

Figure 9. Instantaneous throughput of TCP over exposed terminal topology.

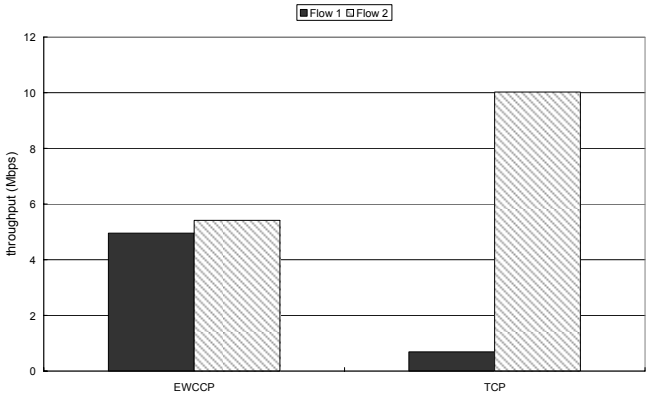

Figure 10. Overall throughput of EWCCP and TCP in exposed terminal scenario.

C. *Bar topology*

In above scenarios, we tested EWCP over symmetric topology. Now we tested EWCP with a topology that two flows experienced different congestion when competing for the same channel resource, as shown in Figure 11. The simulation results of both TCP and EWCCP are shown in Figure 12. In this figure, the aggregated throughput of the two flows is also plotted. We can see that TCP has higher aggregated throughput comparing to EWCCP. However, this reflexes the fundamental tradeoff between the efficiency and fairness. Since EWCCP 2 consumed twice channel resource to deliver packets, it only got around half throughput of that of EWCCP 1, according to the proportional fairness achieved by EWCCP. In this experiment, the throughput of EWCCP 1 was 2.87Mbps while EWCCP 2 was around 5.9Mbps. The aggregated throughput was 8.79Mbps, which should be 3/4 of the throughput if all channel resource was allocated only to EWCCP 1, which was the case of TCP. Without coordination, TCP allocated all channel resource to the flow with better condition, e.g. shorter hops, and the longer hops flow was almost starved. TCP 1 had very high throughput of 10.87Mbps; while TCP 2 only had merely 660Kbps.

D. Complex scenario

In this section, we examined a more complex topology where a long flow traveled through multiple bottlenecks. The topology is shown in Figure 13. And the overall throughput of each flow using EWCCP and TCP is shown in Figure 14. Again, we see that EWCCP preserved pretty well fairness for all flows. Note that flow 1 received near one third throughput of other flows. It is because the proportional fairness implemented by EWCCP and the long flow 1 traversed three bottlenecks while others only passed one. In this case, TCP failed to fairly allocate resource to the long flow. Short flows occupied all channel resource and achieve high throughput, while the long flow was eventually starved.

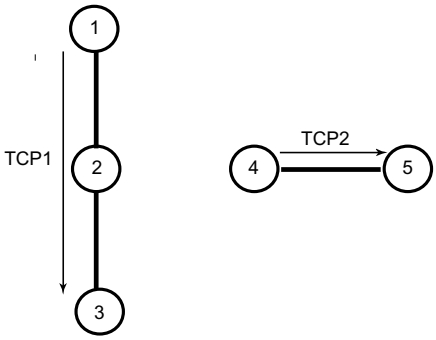

Figure 11. Bar topology. Node 2 and Node 4 can interfere with each other.

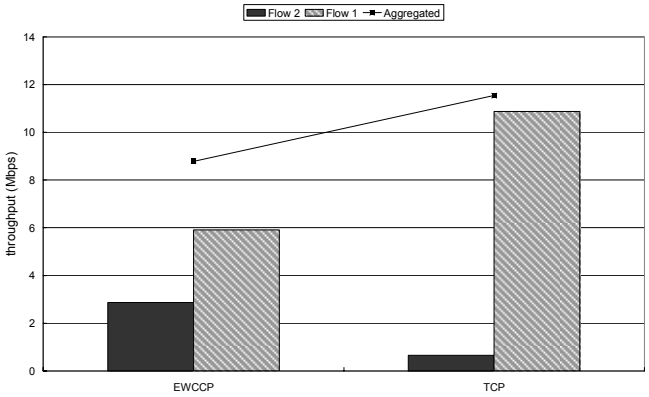

Figure 12. Overall throughput of EWCCP and TCP in bar scenario.

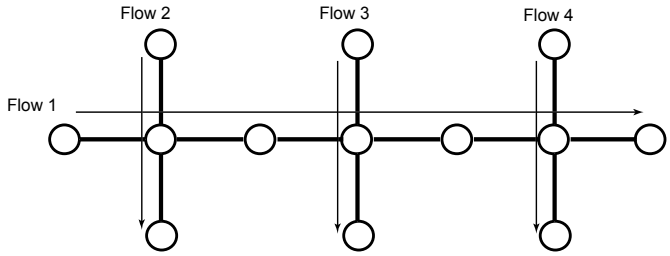

Figure 13. A parking-lot topology.

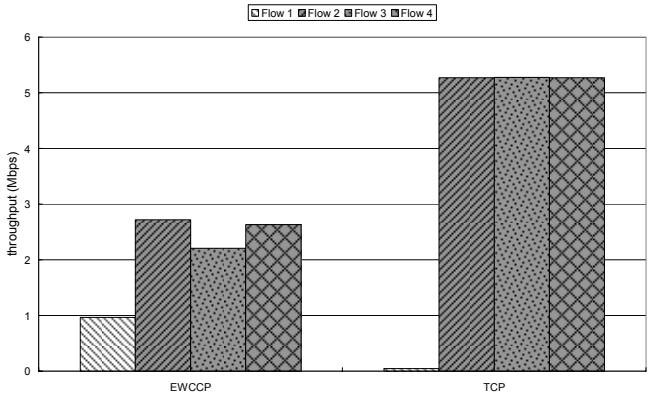

Figure 14. Overall throughput of EWCCP and TCP in parking-lot topology.

VI. Related works

The degradation of TCP's performance in a multihop wireless network has been well recognized in last decade. Many of the performance issues are related to the mobility and there are mainly addressed by cross-layer design that incorporates event notifications from network or even MAC layer [7][13][14]. However, our focus is static, multihop, wireless networks, and therefore mobility is not our focus.

Fu, *et. al.* [15], reports that the greedy behavior of TCP drives the wireless network into saturation which in turn reduces the throughput. They provide a link layer solution, link RED (LRED) and adaptive pacing, to mitigate the problem. With our experiments, the degradation due to link layer contention is around 1%. Therefore, in our setup, adaptive pacing which deliberately adds extra backoff between successive transmissions may only add additional delay in delivering packets. Moreover, LRED calculates dropping only based on its own perception. Without coordinating with other interference links, it still has the same fairness problem of TCP if different links in the same congestion region observe different congestion.

Xu, *et. al.*, [12] develops a Neighborhood-RED (NRED) to address the TCP unfairness issues in ad hoc network. The idea is to coordinate a common packet dropping rate based on a virtual distributed queue size in the neighborhood of each node. The virtual queue in [12] contains queues on the node and its 2-hop neighbors. This is different from our neighborhood queue definition. The neighborhood queue in this paper is defined over a wireless link and its interference set, which captures the interference between transmissions more precisely. As we mentioned before, reference [12] assumes that nodes can monitor the channel utilization, and uses the measured utilization as a congestion indicator. Based on this, NRED tries to maintain the channel utilization at some lower level. Although this helps improving the fairness, it reduces the aggregated throughput. Further, accurately monitoring channel utilization is still difficult in practical systems. In EWCCP, we directly use size of neighborhood queue as con-

gestion indicator. The control law properly distributes the neighborhood queue over links that mutually interfere. In this way, EWCCP improves the fairness but still keep the channel at high utilization. Moreover, NRED still relies on one-bit congestion feedback, which may make the control protocol instable. EWCCP provides a systematic way for resource allocation in multi-hop wireless networks using multi-bit explicit rate control. EWCCP also removes the RTT unfairness. EWCCP provides stable control and implements a proportional fairness among flows.

Beside the enhancements to TCP, several new protocols are also proposed for ad hoc network [10][11]. Sundaresan, *et. al.*, [10] develops and ad-hoc transport protocol (ATP) that relies on routers to give explicit rate control. The control information is the bottleneck service delay (including transmission and queuing delay). However, like TCP, ATP provides feedback only based on its own observation. Without coordination with other competing nodes, ATP suffers the same unfairness issue as TCP due to asymmetric information. Unlike ATP, which uses rate-based control, EWCCP adopts window-based scheme and enjoys self-clocking for more stable control. In EXACT [11], explicit rate calculation is used to implement max-min fairness. However, like ATP, lacking of coordination exposes EXACT to the same unfairness issue. Moreover, EXACT assumes the node has full states of all passing by flows. In contrary, EWCCP puts congestion information into packets and routers do not need to remember per flow information. This makes EWCCP more scalable for potential high user volume in a high-speed multihop wireless network.

Note that there are also a number of theoretic works on the rate-control in multihop wireless network [29][30] [31]. However, they usually assume either a slotted system with perfect scheduling [29][30], or a simple interference model [31]. These works, although gives some insights for congestion control protocol design, have a large gap from a practical scheme for multi-hop wireless networks.

VII. Discussion

EWCCP is a new congestion control protocol specifically designed for static, multihop, wireless networks. However, it does not necessarily mean to replace the widely deployed TCP stack. Actually, EWCCP can be implemented as a thin layer between IP and TCP, which provides enhanced congestion control functionality in multihop wireless networks (similar to the idea in *Ad hoc TCP* [13]). When packets are sent from TCP layer to EWCCP layer, they are also regulated by EWCCP. Packets are only sent out when they are allowed by EWCCP *window*. EWCCP can be implemented invisible to TCP. Therefore, the standard TCP stack does not need to modify, but applications can enjoy the efficiency and fairness brought by EWCCP. Another benefit for this design is that EWCCP will not adversely interact with TCPs functioning in cases where the communication is between a node in the multihop wireless network and another in the Internet. The gateway nodes that bridge the two networks can automatically remove/add the EWCCP congestion header according to the direction that the packet goes.

In previous discussion and simulation, we assume that the transmission range equals to the interference range. However, in some cases (e.g. outdoor open environment), the interference range may be much larger than the transmission range. As shown in [1][6], the distance and shape of the interference range heavily depend on the physical environment. The general ways to precisely identify the interference range are out-of-scope of this paper. We assume that the interference information is available when the multihop wireless network is planned. We note that EWCCP can still be effective when the interference range is much larger than the transmission range. The only change will be made to the signaling protocol, which instead of broadcasting within one-hop neighbors, may be allowed to re-broadcast within *k*-hop neighbors. For example, if the interference range is twice the transmission range, the local queue information may be propagated within 2 hop neighbors. We regard this as another advantage of using explicit signaling protocol than passive channel monitoring in [12], which might not be effective if the interference range is much larger than the transmission range. Note that the *separated control* can still help in reducing the signaling overhead. Evaluating the signaling overhead in this situation is subject to our future work.

VIII. Conclusion

In this paper, we have proposed a novel Explicit Wireless Congestion Control Protocol (EWCCP) for stationary multihop wireless network. EWCCP exploits explicit coordination with wireless links that mutually interfere, and provides multi-bit explicit feedbacks for fine-grained control. With explicit feedback, EWCCP sending window stabilizes at a lower but more optimal value regarding to TCP, and therefore achieves low buffer occupation and low delay. With explicit coordination among wireless links that compete for the shared channel, EWCCP allocates channel resource fairly. With our analytical analysis and packet-level simulations, it is concluded that EWCCP is a viable congestion control scheme for a stationary multi-hop wireless network.

References

[1] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network", in *Proc. of SIGCOMM*, 2004.

[2] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of tcp/aqm and a scalable control", in *Proc. of IEEE INFOCOM*, June 2002.

[3] S. Kunniyur and R. Srikant, "End-to-end congestion control schemes: utility functions, random losses and ECN marks", *ACM Tran. On Networking*, 2003.

[4] F. P. Kelly, "Charging and rate control for elastic traffic", *European Transactions on Telecommunications,* vol. 8, 1997.

[5] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", *IEEE/ACM Tran. On Networking*, Aug. 1993.

[6] D. Kotz, C. Newport, and C. Elliott, "The mistaken axioms of wireless-network research", *Dartmouth College Technical Report TR2003-467*, July 2003.

[7] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks", *IEEE Personal Communications Magazine*, Feb. 2001.

[8] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?", *IEEE Communications Magazine*, 39(6), Jun. 2001.

[9] K. Tang and M. Gerla, "Fair sharing of MAC under TCP in wireless ad hoc networks", in *Proc. IEEE MMT* 1999.

[10] K. Sundaresan, V. Anantharaman, H-Y. Hsieh, and R. Sivakumar, "ATP: a reliable transport protocol for ad-hoc networks", in *Proc. Mobihoc*, 2003.

[11] K. Chen, K. Nahrstedt, and N. Vaidya, "The utility of explicit rate-based flow control in mobile ad hoc networks", in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, March 2004.

[12] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP Fairness in Ad Hoc Wireless Networks Using Neighborhood RED", in *Proc. Mobicom*, 2003.

[13] J. Liu, "ATCP: TCP for ad hoc network", *IEEE Journal on Selected Areas in Communications, Volume 19, Issue 7,* July 2001.

[14] G. Holland and N. Vaidya, "Anaysis of TCP performance over mobile ad hoc networks", in *Proc. Mobicom*, 1999.

[15] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss", in *Proc. IEEE INFOCOM*, Mar. 2003.

[16] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks", in *Proc. SIGCOMM,* 2002.

[17] K. Ramakrishnan and S. Floyd, "Proposal to add explicit congestion notification (ecn) to ip", *RFC 2481,* Jan. 1999.

[18] D. B. Johnson, D. A. Maltz, and Y. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", *Internet Draft. <draft-ietf-manet-dsr-10.txt>*, July 2004.

[19] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, multi-hop wireless mesh networks", in *Proc. Mobicom,* 2004.

[20] W. R. Stevens, *TCP/IP Illustrated*, volume 1. Addison-Wesley, 1994.

[21] Seattle wireless. *http://www.seattlewireless.net*

[22] MIT roofnet. *http://www.pdos.lcs.mit.edu/roofnet*

[23] Microsoft Research. Self-Organizing Neighborhood Wireless Mesh Networks. *http://research.microsoft.com/mesh/*

[24] C. E. Koksal and H. Balakrishnan, "An analysis of short-term fairness in wireless media access protocols", in *Proc. ACM SIGMETRICS, Measurement and Modeling of Computer Systems*, 2000.

[25] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing", In *Proc. IEEE INFOCOM*, 2000.

[26] D. Bertsekas and R. Gallager. *Data networks,* Prentice Hall Press 2nd Edition, 1992.

[27] B. Bloom, "Space/time trade-offs in hash coding with allowable errors", *Communications of ACM*, pages 13(7):422-426, July 1970.

[28] J. Li, C. Blake, D. De Couto, H. I. Lee, and R. Morris, "Capacity of ad hoc wireless networks", in *Proc. Mobicom,* 2001.

[29] Y. Xue, B. Li, and K. Nahrstedt, "Price-based resource allocation in wireless ad hoc networks", in *Proc. International Workshop on Quality of Service (IWQoS)*, 2003.

[30] Z. Fang and B. Bensaou, "Fair bandwidth sharing algorithms based on game theory frameworks for wireless ad-hoc networks", in *Proc. IEEE INFOCOM*, 2004.

[31] Y. Yi and S. Shakkottai, "Hop-by-hop congestion control over a wireless multi-hop network", in *Proc. IEEE INFOCOM*, 2004.

[32] The network simulator ns-2. *http://www.isi.edu/nsname/ns*.