Enhancements on Router-Assisted Congestion Control for Wireless Networks

Jian Pu, *Student Member, IEEE,* and Mounir Hamdi, *Senior Member, IEEE*

***Abstract*—This paper addresses two challenges that are encountered when applying router-assisted explicit-feedback congestion control to wireless networks. The first challenge is how to distinguish between the two kinds of packet loss (non-congestion loss and congestion loss) in wireless networks and how to react to them properly. The second challenge is how to probe the unknown bandwidth capacity of wireless links which is required in calculating the router feedback. Some practical and novel enhancements on router-assisted congestion control in such environments are introduced in this paper. We have implemented these enhancements in a router-assisted congestion control protocol termed QFCP. Extensive simulation experiments using ns-2 will demonstrate that QFCP can fairly and efficiently allocate wireless bandwidth resources among competing flows in heterogeneous networks.**

***Index Terms*—Congestion control, routers, wireless links, scheduling algorithms, protocol design.**

I. Introduction

It is well-known that loss-based end-to-end congestion control such as TCP [1] does not work efficiently in lossy wireless networks [2], [3], [4]. One reason is that TCP treats packet loss as a congestion signal since it cannot distinguish between non-congestion-related loss (caused by bit errors) from congestion-related loss (caused by buffer overflow) leading to underutilization of wireless links. Another reason is that after experiencing a packet loss, TCP needs to take many RTTs to recover to the previous high throughput. This effect is aggravated in high-speed wireless networks as TCP only increases its window by one packet per RTT which is too conservative for high-speed or long-delay links.

Router-assisted congestion control (e.g., XCP [5], RCP [6], QFCP [7]) is originally introduced to overcome the problems of TCP in high bandwidth-delay product (BDP) wired networks, such as low link utilization, saw-tooth-like throughput, and large queueing delay. With the help of explicit feedback from routers to end hosts, competing flows are able to converge to the fair-share rate within a few RTTs and maintain high utilization of the bottleneck link. In particular, for lossy wireless networks, experiments show that XCP can achieve higher throughput than TCP [8], [9]. However, we note that XCP's good performance is largely due to its fast window growth instead of robustness to wireless loss, and hence there is still room for improvement.

First, XCP still cannot differentiate the two kinds of packet loss. Actually in the current implementation of XCP, it reacts to packet loss in the same way as TCP does [10]. The congestion window is halved and the throughput is slowed down unnecessarily when a bit error occurs in transmission. Although the impact of packet loss is alleviated in XCP since it can grab the unused bandwidth much faster than TCP, the flow throughput still suffers from frequent window halving in lossy environments.

Second, router-assisted congestion control requires an exact and a fixed value of the link bandwidth which is set as a parameter to the algorithm in advance. But for a wireless link, due to shared-media contention, simplex, and changing physical conditions, it is very hard to do such kind of setting. For example, in an 802.11b [11] wireless node, it even can dynamically change its MAC-layer data rate to 1, 2, 5.5, or 11 Mbps making a fixed setting of bandwidth parameter almost impossible. All of these factors cause estimation errors and performance deterioration for router-feedback control. The authors of [8] show that XCP is unable to maintain fairness and stability with improper estimation of the link capacity parameter. Thus, we need a more intelligent algorithm that can probe the variable link bandwidth in wireless networks.

Authors of XCP-b [12] also find that XCP does not work well on a shared-access multi-rate media such as 802.11 WLAN, and propose an algorithm for XCP to probe the available bandwidth. But the probing ability of their algorithm largely depends on the buffer size Q_{max}, and may not be suitable for wireless nodes with small buffers but in large BDP networks. XCP-r [13] suggests that it is better to let the receiver compute the congestion window size and send the value back to the sender through ACK packets. This modification on XCP partly solves the window mismatching problem caused by ACK loss assuming no packet loss on the forward path. EXACT [14] is another explicit rate-based flow control designed for Mobile Ad Hoc Networks (MANET). In EXACT, each router keeps track of current flows and their sending rates in a flow table. It is costly and needs much storage space and computational power if many flows coexist in the network. Thus, as its authors claimed, EXACT is not targeted for the large scale Internet but rather for MANET where the flow number is relatively small and the additional expense is acceptable.

We make the first effort to combine the two problems together and solve them in a single control framework. Although many previous studies are based on XCP or its variants, we

Manuscript received December 18, 2006; revised April 18, 2007; accepted April 20, 2007. The associate editor coordinating the review of this paper and approving it for publication was M. Guizani. This work was supported in part by the Hong Kong Research Grant Council under Grant RGC HKUST6260/04E.

J. Pu and M. Hamdi are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China (e-mail: pujian, hamdi@cse.ust.hk).

Digital Object Identifier 10.1109/TWC.2008.061085.

decide to choose QFCP [7] as the groundwork to develop a robust congestion control protocol that is suitable for wireless networks. The reason why we use QFCP instead of XCP is that XCP's feedback is packet-wise independent which makes loss recovery very difficult. That is, each packet carries unique information on window adjustment and the feedback carried on lost packets cannot be recovered by other successfully received packets. While in QFCP, each router directly uses its fair-share flow rate as the feedback and this rate is not changed during the current control interval. The feedback information in any single ACK is sufficient for the sender to compute the target window size. Further, QFCP does not store per-flow state on routers making it scalable for any number of flows. In particular, we try to find a way to probe the link capacity independent of the buffer size so that the algorithm can be applied on routers with any buffer size or bandwidth capacity. More details will be explained in Section III.

II. Router-Assisted Congestion Control

In general, a router-assisted congestion control can refer to any mechanism that involves routers in congestion control such as various kinds of Active Queue Management (AQM) schemes on routers. But here we would like to restrict the meaning of this term to protocols that use explicit multi-bit router feedback instead of implicit one-bit signal of packet loss to indicate the network congestion condition so as to differentiate from the traditional TCP/AQM approaches. We now briefly describe XCP and QFCP as examples to show some details of this kind of congestion control.

XCP [5] is a window-based congestion control protocol that uses explicit feedback from routers to adjust the congestion window size of senders. XCP introduces a new header on each packet to carry flow information such as throughput, round-trip time (RTT), delta-throughput (carrying the throughput change value allowed by upstream routers), and reverse-feedback. The Efficiency Controller (EC) on each router periodically calculates the available bandwidth F as:

$$F = \alpha \cdot (C - y) - \beta \cdot q/d, \tag{1}$$

where C is the capacity of the output link, d is the control interval, y is the aggregate input traffic rate measured in the last period d, and q is the minimum queue length observed in the last period d. α and β are two constants and are set as 0.4 and 0.2263 respectively to make the system stable. The control interval d is set to be the average RTT of all flows traversing this controller. Then the Fairness Controller (FC) on this router computes the per-packet feedback by distributing the available bandwidth F among all flows:

- If $F > 0$, allocate the positive feedback equally to all flows;
- If $F < 0$, allocate the negative feedback proportional to each flow's current throughput.

The feedback is computed for each packet and is copied to the delta-throughput field only if this feedback is less than the current value in that field. So the delta-throughput field will finally store the feedback calculated by the bottleneck router. When a packet reaches the receiver, the receiver copies the delta-throughput field into the reverse-feedback field of the corresponding acknowledgment (ACK) packet and sends the ACK to the sender. When the sender receives this ACK, it adjusts its congestion window ($cwnd$) as follows:

$$cwnd = \max(cwnd + feedback \cdot RTT, MSS), \tag{2}$$

where RTT is the round-trip time measured by the sender and MSS is the maximum segment size.

QFCP [7] is another router-assisted congestion control protocol for high-speed networks. Unlike XCP, QFCP gives per-flow feedback based on flow rate instead of per-packet feedback based on window adjustment. There are three fields in the new QFCP header of each packet: RTT, rate-request, and rate-feedback. A router maintains a fair-share rate R for each output interface. This rate R is the maximum rate allowed for flows going through this interface during the current control interval T. T is set to be a moving average of RTTs of seen packets. At the beginning of every control interval, the QFCP controller estimates the number of flows traversing this interface using the formula:

$$N(t) = \frac{y(t)}{R(t-T)}, \tag{3}$$

where y is the input traffic rate measured in the last interval T, and $R(t-T)$ is the old flow rate feedback given in the last control interval. Then the controller updates its fair-share rate R as follows:

$$R(t) = \frac{C - \beta \cdot \frac{q(t)}{T}}{N(t)}, \tag{4}$$

where C and q have the same meaning as in XCP, and β is a constant of 0.5. When a packet arrives at a router, the controller compares the value in the rate-request field with its own fair-share rate R and copies the smaller value back into that field. This rate-request field will eventually be copied into the rate-feedback field of the corresponding ACK packet and sent back to the sender by the receiver. Upon receiving an ACK, the sender reads the feedback and adjusts its congestion window as follows:

$$cwnd = \max(feedback \cdot RTT, MSS), \tag{5}$$

where $feedback$ is the router feedback on flow rate, RTT and MSS have the same meaning as in XCP. Thus, flows can send data at the highest rate allowed by all routers along the path, while routers periodically update their fair-share rates based on the number of estimated flows. Simulations show that both XCP and QFCP can achieve high utilization on large BDP links. However, QFCP can further shorten flow completion time [15] and help flows converge to the fair-share rate faster [7].

III. Wireless Enhancement

There are two problems addressed here when applying router-assisted congestion control to wireless networks. One is the unknown bandwidth capacity of a simplex contention-shared multi-rate wireless link. The other is how to deal with non-congestion-related packet loss which commonly exists in lossy wireless networks.

TABLE I
IMPACT OF THE PARAMETER VALUES ON THE FEEDBACK CONTROL SYSTEM: THE COLUMNS CORRESPOND TO DIFFERENT VALUES OF α AND THE ROWS CORRESPOND TO DIFFERENT VALUES OF β. EACH DATA RECORD IS THE TIME (IN SECONDS) NEEDED BEFORE THE SYSTEM ENTERS THE STEADY STAGE FOR EACH (α, β) PAIR.

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	2.9	2.4	2.4	2.5	0.7	2.7	2.0	2.9	1.5	2.4
0.2	2.9	2.1	1.9	1.8	0.7	1.8	1.4	1.8	1.1	1.5
0.3	3.0	2.0	1.7	1.6	0.7	1.5	1.3	1.5	1.0	1.3
0.4	3.0	1.9	1.6	1.5	0.8	1.3	1.1	1.3	0.9	1.1
0.5	2.9	1.9	1.5	1.4	0.8	1.2	1.0	1.2	0.9	1.0
0.6	2.9	1.8	1.4	1.3	0.8	1.1	1.0	1.1	0.8	0.9
0.7	2.9	1.8	1.4	1.2	0.8	1.0	0.9	1.0	0.8	0.8
0.8	2.9	1.7	1.3	1.2	0.8	1.0	0.8	0.9	0.7	0.8
0.9	2.8	1.7	1.3	1.1	0.8	0.9	0.8	0.8	0.7	0.7
1.0	2.8	1.6	1.2	1.0	0.8	0.8	0.7	0.7	0.6	0.6

A. Enhancement for Unknown Bandwidth

In order to accurately calculate the feedback, the router must know the exact bandwidth capacity in advance. For both XCP and QFCP, the output link capacity C acts as an important parameter in the control algorithm. If the router underestimates the bandwidth capacity, it will underutilize the link and waste the valuable bandwidth resource. And if the router overestimates the capacity, it will give improper feedback to senders to increase their congestion windows and may cause queue growth and even buffer overflow (congestion). But it is very hard to decide a proper value of C for a wireless link in advance. One reason is that a wireless channel is shared by competing neighbor nodes and the number of nodes sharing this channel may change all the time. Another reason is that the wireless link bandwidth is affected by many changing physical conditions, such as signal strength, propagation distance, and transmitter power. For example, an 802.11 node can change its MAC-layer data rate dynamically according to different conditions, which means the output bandwidth of this node and other neighbor nodes may also change.

Due to the inability to set the exact capacity of a wireless link, we need to design an adaptive algorithm that can find and set this capacity parameter by itself. We observe that the output traffic rate can be used to estimate the link capacity for an active network interface, and we add the following formula in QFCP for link bandwidth probing:

$$C = \begin{cases} output & \text{if } q \geq 1 \\ (1+\alpha)\cdot C & \text{else} \end{cases}, \tag{6}$$

where q is the minimum queue length in packets observed in the last control interval, $output$ is the output traffic rate, and α is a constant of 0.1. The basic idea is that:

- If the minimum queue length q is greater than or equal to one packet, which means this output interface is busy and keeps sending data in the last control interval, then the output traffic rate can be a good estimation of the current link capacity of this interface.
- If the minimum queue length is less than one packet, which means the output link is sometimes idle and underutilized during the last control interval, we can try to multiplicatively increase the link capacity estimation by a factor $(1+\alpha)$, and wait a control interval to see whether the queue is going to build up.

In order to properly deal with the burstiness nature of packet switching networks, we actually use the weighted moving average of $output$ and q in the above formula to smooth out possible oscillation caused by packet bursts:

$$output_{avg} = w \cdot output + (1-w)\cdot output_{avg}, \tag{7}$$

$$q_{avg} = w \cdot q + (1-w)\cdot q_{avg}, \tag{8}$$

where the weight w is 0.2 in our implementation. Generally this weight w should be chosen heuristically according to the burstiness of the network. A small w can smooth out a burst over more samples recently measured, but may also react slowly to any change in the network conditions. By repeating the above probing procedure for each control interval, we can finally find the proper bandwidth estimation. Note that if the link is always idle or underutilized (i.e., a non-bottleneck link), this estimation value may grow into infinity. So it is necessary to put an upper bound on the value. We can simply use the maximum MAC-layer data rate as the upper bound of C (e.g., 11 Mbps for 802.11b, or 54 Mbps for 802.11g). Also, whenever this link becomes busy again (bottleneck link), the above algorithm will adapt the bandwidth estimation to the correct value by using the output traffic rate.

There are also some implementation details we need take care of. Since the wireless link is simplex and the bandwidth is shared among uploading and downloading flows, the controller should count packets in both directions when computing the input traffic rate y. Furthermore, since our target steady state is not zero queue length but a resident queue with at least one packet, the formula to compute the rate feedback should be changed accordingly:

$$R(t) = \frac{C - \frac{\beta \cdot q(t) - MSS}{T}}{N(t)}. \tag{9}$$

Here we briefly discuss the value choosing for the parameters α in (9) and β in (4). Table 1 shows that α and β influence the speed of the system entering the steady stage. The scenario is that the initial setting of C is 1 Mbps while the real bandwidth is 11 Mbps. The queue is initially empty and the round-trip time is 100 ms. We keep monitoring the representative status variables such as $C - C_{real}$, $q(t) - MSS$, and $N(t) - N_{real}$. When all these differences are within 1% we claim that the system has entered its steady state. We find that, in general, larger α and β can help the system converge to the ideal working state more rapidly. For example, when α = 0.1 and β = 0.3, the system needs 3.0 seconds to enter the steady state; while for α = 0.9 and β = 0.9, the time is only 0.7 seconds. However, we also find that when β is greater than 2 (not shown in the table), the system becomes unstable. This is because a large β is too aggressive to drain the queue in less than one RTT, and always results in an empty queue in the next control interval. Thus, the probing algorithm (6) will grow the bandwidth estimation again and cannot maintain a correct value of C. There is a similar problem for large α, which may increase the bandwidth estimation too fast leading to packet burstiness and instantaneous large queues.

B. Enhancement for Packet Loss

For a sender in lossy wireless environments, it is important to differentiate two kinds of packet loss: for non-congestion-related loss (caused by bit errors), it should maintain the current window size; and for congestion-related loss (caused by buffer overflow), it should slow down to prevent congestion collapse. Unfortunately, currently router-assisted congestion control protocols cannot do such differentiation. For example, XCP simply inherits the standard TCP behavior when encountering a packet loss [10]. That is, upon receiving three duplicate ACKs, the congestion window $cwnd$ is halved; and on retransmission timeout, $cwnd$ is set to one. The assumption is that a packet loss may reveal a congested non-XCP router in the path, and transiting to standard TCP behavior is a conservative response. However, if we are sure that all routers along the path support router-assisted congestion control, such slow-down reaction should be unnecessary for packet loss caused by bit errors.

For TCP, the sender needs to slow down upon detecting a packet loss, because a packet loss is the congestion signal for TCP. This is due to the design rationale of TCP congestion control. A TCP flow keeps increasing its sending rate and intentionally fills up the buffer of the bottleneck router to generate packet drops. Through this approach TCP finds the available capacity of the path. But for a router-assisted approach, since congestion information has already been wrapped in the special packet header and communicated to the sender, the sender should not insist on treating a packet loss as a congestion signal any more. Instead, it should use the information in the congestion header to adjust its congestion window. For example, in QFCP, if a congestion-related loss occurs, the router feedback in the duplicate ACK will tell the sender to slow down its sending rate; but if it is a non-congestion-related loss, the rate feedback will be like the current sending rate of this flow.

We suggest that we separate the data reliability control from congestion control when receiving duplicate ACKs. When the sender receives a duplicate ACK, it implies that a data packet has successfully reached the receiver but its sequence number is greater than that expected by the receiver. Thus, for data reliability control, upon reception of 3 duplicate ACKs, the sender should retransmit the packet with the expected sequence number. While for congestion control, when a QFCP sender receives a duplicate ACK, it adjusts the congestion window to as follows:

$$cwnd = feedback \cdot RTT + num\_dupACK, \qquad (10)$$

where $feedback$ is the rate feedback from routers, RTT is the sender's estimation of round-trip time, num_dupACK is the number of duplicate ACKs received. The inherent idea is that the sender temporarily keeps the successfully-transferred but not-in-order packets in the buffer and opens the congestion window so that it can continue sending data at the router-allowed rate. The counter num_dupACK is reset to zero when a new ACK packet arrives and cumulatively acknowledges all data packets sent before the detection of the loss. Note that we do not address complicated situations such as loss of the retransmitted packet here and leave them for future study.

XCP is a little more complicated and different from QFCP. QFCP directly uses the fair-share flow rate as the feedback and this rate is not changed during the current control interval. The rate feedback information in any single ACK is sufficient for us to compute the target window size. But for XCP, we may not be able to compute the correct window size based on the feedback only when encountering a loss. Because in XCP, each ACK carries unique per-packet feedback information on window adjustment and the information carried on lost packets may not be negligible. Any packet loss will cause mismatching between the actual window size of the sender and the target window size expected by the routers. XCP-r [13] suggests computing the congestion window size at the receiver side and sending the value back to the sender through ACK packets. This modification on XCP only deals with ACK loss but packet loss on the forward path may still cause the window mismatching problem. Another possible solution is to keep the window unchanged on non-congestion loss, and halving the window on congestion loss. But first we need to distinguish the two kinds of losses in XCP. Intuitively we may say if the feedback of duplicate ACK is positive, it is non-congestion-related loss; and if the feedback is negative, it is a congestion-related loss. However, this assumption is not always true. One the one hand, since the feedback is also used for fairness control, a negative feedback may possibly only want to change the flow's rate toward the fair-share rate and may not necessarily suggest congestion. Halving the $cwnd$ or change $cwnd$ to 1 is too aggressive for this case. On the other hand, if the loss is congestion-related, window adjusting only based on feedback may not be enough since some feedback on window reduction may also be lost. Keeping $cwnd$ unchanged in this case will lead to a window larger than expected by routers and cannot help alleviate the congestion in the network. In sum, unlike QFCP, it is not so easy for XCP to differentiate the two kinds of packet losses based on current feedback information.

While for a packet loss event triggered by retransmission timeout (RTO), since no feedback information is available at this instant and the loss may be caused by severe congestion, conservatively setting the congestion window to one should be a good choice. And if this is not a congestion-related loss, the rate feedback of any subsequent ACK will recover the congestion window to the proper size in QFCP.

In addition, if a router drops packets due to buffer overflow, it should also sum up the number of dropped packets and use the virtual queue length when running the control algorithm. That is, substitute q in the algorithm with

$$virtual\_q = q + num\_drop, \qquad (11)$$

where num_drop is the number of dropped packets due to buffer overflow. Thus, if some packets are dropped by routers, the feedback computed using the virtual queue length can still precisely reflect the current congestion condition.

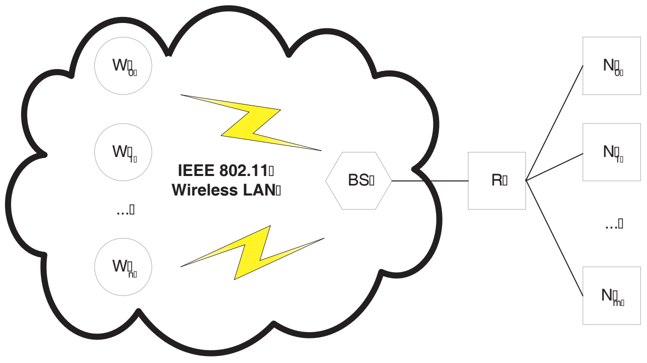


Fig. 1. Simulation network topology.

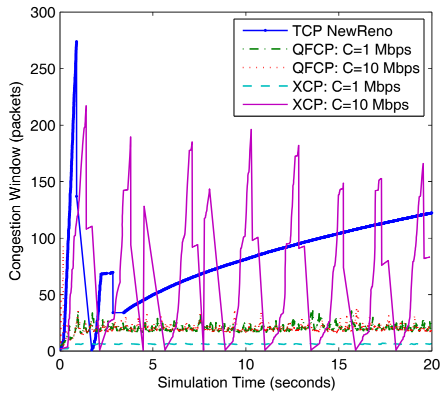
IV. Simulation Results

In this section we present simulation results using ns-2 [16]. The network topology is shown in Fig. 1. An IEEE 802.11 Wireless LAN is connected to a wired high-speed network through a base station BS. The MAC-layer data rate of the 802.11 WLAN is 11 Mbps and the basic rate is 1 Mbps. The router R is connected to BS and all wired end-systems $N_i(i = 0, 1, \ldots, m)$. All wired links are duplex links with fixed bandwidth capacity of 1 Gbps so that only the wireless link can become the bottleneck. The minimum round-trip propagation delay is 40 ms. Data packet size is 1300 KB and ACK packet size is 40 KB. The buffer size on routers for each output link is 100 packets. For TCP simulations, we choose TCP NewReno [17] and use Drop-Tail as the queue management scheme. For simulations on router-assisted congestion control protocols (XCP, XCP-b, and QFCP), all routers and end-systems are installed with proper controllers. We use the original codes of XCP embedded in ns-2 and the XCP-b codes provided by its authors. All algorithm parameters are configured using the default values as recommended by the authors. We have implemented the enhancements for wireless networks discussed above in QFCP. In the rest of this paper, we will use QFCP to denote "QFCP with wireless enhancements" for convenience.

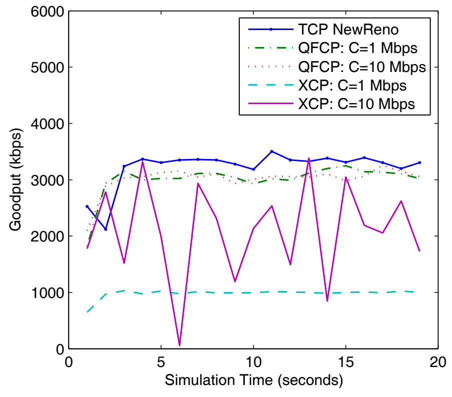
A. Single Flow Dynamics

Here we simulate the downloading of a flow from a wired node to a wireless node in the WLAN to demonstrate the dynamics of three different protocols: TCP, XCP and QFCP. For router-assisted protocols (XCP and QFCP), the bandwidth parameter C of any wireless interface is set to 1 Mbps (underestimated) and 10 Mbps (overestimated) respectively for each run, while TCP does not need such setting. We record the congestion window size, the goodput, the retransmitted bytes, and the bottleneck queue length as shown in Fig. 2.

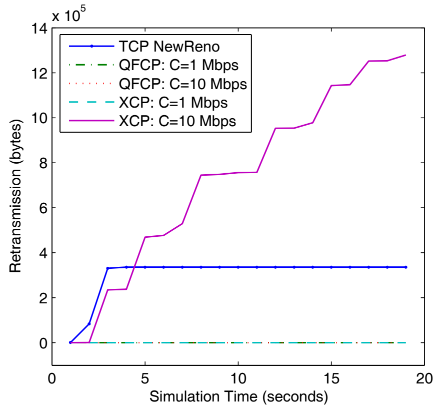
For TCP, it is able to achieve high goodput, but at the cost of massive packet drops and large persistent queues. The large amount of retransmission is due to TCP's probing approach. TCP keeps increasing the flow rate until packets are dropped by the bottleneck router (buffer overflow). Dropped packets require retransmission and this wastes the valuable bandwidth resource. In addition, TCP keeps large queues on routers, which brings extra queueing delay and increases the overall latency of the path.



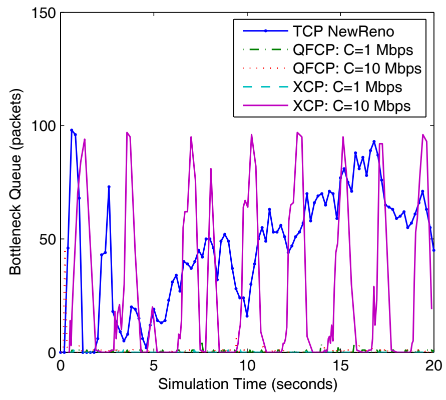
(a) Congestion Window



(b) Goodput



(c) Retransmission



(d) Bottleneck Queue Size

Fig. 2. Comparison of TCP, XCP and QFCP on wireless link with unknown bandwidth.

For XCP, it requires that the router knows the exact capacity of the output link, and its performance heavily depends on the link bandwidth setting C. When C is underestimated ($C = 1Mbps$), its target congestion window is small, its goodput is low, and the link is underutilized. When C is overestimated ($C = 10Mbps$), XCP's router feedback tends to increase the flow rate to an unachievable value. Then the queue grows toward infinity and the router buffer is overflowed periodically. This is why many packets are dropped (Fig. 2(c)) and the queue length oscillates between zero and the maximum buffer size (Fig. 2(d)).

For QFCP, the results show that the proposed bandwidth probing algorithm works quite well. QFCP always finds the correct estimation of the wireless link capacity in a short time (about 2-3 seconds) no mater what the initial value of the link bandwidth C is set to. During the steady stage, its congestion window is relatively stable and its goodput is high. The zero retransmission suggests that no packet is dropped due to buffer overflow. Moreover, QFCP's queue length is very small when the system enters the steady state. For the case when the initial value of C is 10Mbps, although the queue length grows up to 50 packets at the beginning due to the problem of bandwidth overestimation, this queue is drained up shortly after the correct bandwidth has been found.

B. Multiple Flows with Different RTTs

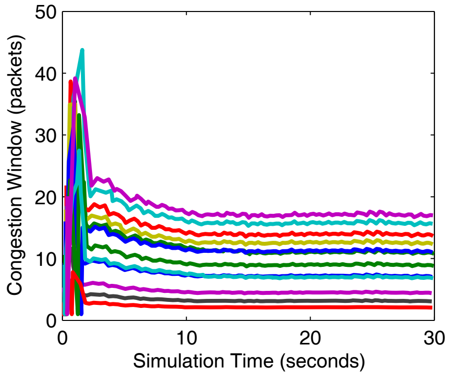
In this simulation there are twelve flows which start transferring data at time 0. Flows are in both directions, either from a node in the wired network to a wireless node in the 802.11 WLAN (downloading) or opposite (uploading). The flows' round-trip propagation delays vary from 40 ms to 370 ms. We use the Jain fairness index [18] to evaluate the fairness of bandwidth allocation among competing flows:

$$J = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2}, \tag{12}$$

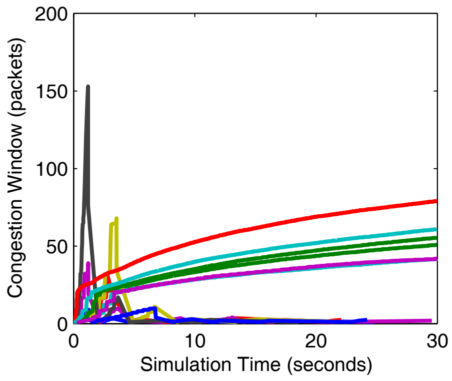
where x_i is the average throughput of flow i during a interval and n is the number of flows. We compute the Jain index at the interval of 1 second.

For QFCP, although the wrong estimation of wireless link capacity causes some throughput oscillations at the beginning, its effect is alleviated shortly after 2-3 seconds when the probing algorithm in QFCP controller finds the correct bandwidth value (Fig. 3(a)). Then the bandwidth is allocated equally among flows because the same rate feedback is sent to all flows. The senders open their congestion window proportional to their RTTs (i.e., $cwnd = feedback \cdot RTT$) and achieve good fairness (Fig. 3(c)).

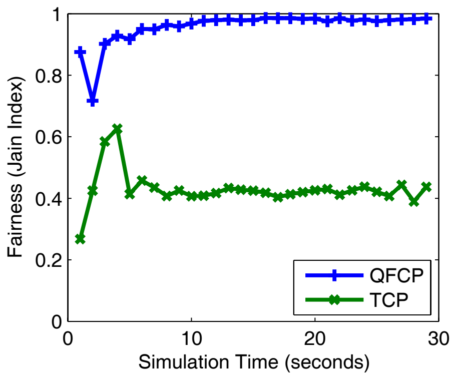
TCP also has its embedded algorithm to probe the available bandwidth, which is the additive increase multiplicative decrease (AIMD) algorithm [19]. AIMD has been proved that it can help flows with the same RTT achieve fairness on throughput. However, as shown in Fig. 3(b), TCP's performance degrades significantly in this scenario. The unfairness problem of TCP flows in 802.11 WLAN is also found in commercial networks [20]. One reason is that flows with short RTTs grow their windows faster than flows with long RTTs (e.g., the different slopes of incensement in Fig. 3(b)).



(a) Congestion window of twelve QFCP flows



(b) Congestion window of twelve TCP flows



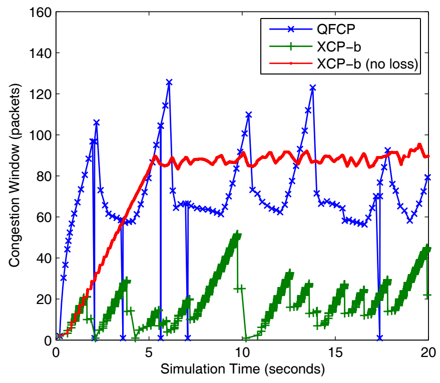
(c) Jain Fairness Index

Fig. 3. Comparison of TCP, XCP and QFCP on wireless link with unknown bandwidth.

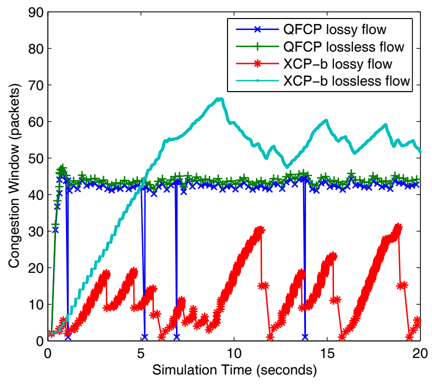
Another more important reason is that a wireless link is simplex, and downloading and uploading flows compete for this wireless channel. But all downloading flows have only one node (the base station) to contend the media access while each uploading flow has its own node to contend. The result is that downloading flows are unable to gain their fair share of the wireless bandwidth using TCP and keep sending at very low rates. Fig. 3(c) confirms that TCP's fairness index is low in this scenario. While for QFCP, since it takes control of all packets in both directions on simplex wireless links, it fairly allocates the bandwidth among all flows.

C. Random Loss

This scenario tests for the protocol's sensitivity to non-congestion-related loss. A uniform loss model is injected



(a) One lossy flow



(b) One lossy flow competing with one lossless flow

Fig. 4. Comparison of QFCP and XCP-b on lossy wireless link with unknown bandwidth.

into the wireless link so that it can randomly generate non-congestion-related loss at a fixed probability. The packet loss rate we have investigated varies from 0.0001 to 0.1 and covers most typical loss rates seen in a wireless network. Due to space restriction, here we just demonstrate typical results of a packet loss rate 0.01. The initial value of the bandwidth parameter C is set to 1 Mbps since we do not know the exact bandwidth for each wireless node and expect the probing algorithm to find it. The minimum round-trip time is 200 ms. The routers' maximum buffer size for each output link is 200 packets. We compare the performance of QFCP and XCP-b in this scenario.

XCP-b [12] is a XCP variant enhanced with an algorithm to probe the available bandwidth when the link capacity is unknown. When the output link is underutilized, XCP-b increases its available bandwidth estimation by a portion of the buffer size (scaled to the measured average RTT). However, it does not take packet loss into account. As shown in Fig. 4(a), XCP-b can find the correct bandwidth when no packet loss happens, but it cannot maintain a large congestion window when bit-error packet loss happens randomly. Its congestion window is frequently halved or set to one due to packet loss. This window shrinking significantly reduces the flow rate and prevents the probing algorithm of XCP-b to work efficiently. But this kind of slowdown is unnecessary if the packet loss is due to transmission media errors instead of buffer overflow. Furthermore, when one lossy flow competes with one lossless flow (Fig. 4(b)), XCP-b treats the lossy flow as a congested flow and unfairly allocates bandwidth between them. This simulation shows that it is important to take care of non-congestion-related loss when designing congestion control schemes for wireless environments. A bandwidth probing algorithm that works well on lossless link may fail on lossy links. This holds even for router-assisted congestion control protocols which give explicit control feedback on window adjustment.

As mentioned before, packet loss is not treated as congestion signal in QFCP. It does not halve the window upon receiving three duplicate ACKs, so it can maintain a large window even in a lossy environment. However, sometimes packet loss is not recovered by the retransmission upon three duplicate ACKs and RTO may occur (e.g., loss of retransmission packet). In this case, since no router feedback carried on an ACK is available, QFCP conservatively set the window to one packet (e.g., around 17 seconds in Fig. 4(a)). But if this is a non-congestion-related loss, and hence any subsequent ACK will recover the window to the proper size. Moreover, for the situation of one lossy flow competing with one lossless flow, QFCP still can fairly allocate the bandwidth resource between them. Note that although the loss rate is as high as 0.01, the lossy flow maintains a large window all the time except when seldom RTO occurs. But the effect of setting to one-packet window is limited since the window is recovered very soon when new ACK arrives.

QFCP is more robust to packet loss than XCP because the feedback in QFCP has more redundancy. For XCP, each ACK carries a unique feedback on congestion window adjustment and the whole ACK packets in one RTT together give the correct value of aggregate feedback. Packet loss may cause difference between the actual congestion window size and the one expected by routers, especially when the lost ACK packet carries a large value of feedback. This may happen when the current window is small while the target window is large (e.g., the situation after a timeout). But for QFCP, since the feedback is the flow rate and this value is only updated once every control period, any ACK in the current control period can give the correct window size to the sender. This kind of information redundancy help prevent senders from starvation (keep sending at low rate unnecessarily for a long time) in a lossy network.

V. Conclusion

Originally router-assisted congestion control protocols such as XCP are only designed for point-to-point full-duplex lossless wired links. Though they demonstrate good performance on such links, they experience performance deterioration in wireless networks. One reason is that the sender still cannot distinguish the two kinds of packet losses (congestion-related and non-congestion-related) and slow down its throughput unnecessarily on packet loss due to a bit error. Another reason is that the router needs to know the exact output link capacity to compute the congestion feedback, but it is hard to set this bandwidth parameter as a fixed value for a multi-access multi-rate wireless link. To solve these problems, we suggest that upon receiving three duplicate ACKs, the sender only retransmits the lost data packet without halving the window. The router feedback carried in the congestion header of duplicate ACKs will inform the sender how to adjust its congestion window. We have also designed an adaptive algorithm that can probe the unknown link capacity based on the output traffic rate and the minimum queue length measured in the last control interval. We implemented these wireless enhancements in a router-assisted congestion control protocol termed QFCP and carried some simulation experiments using ns-2. The experimental results show that enhanced QFCP can work well on lossy wireless links with unknown bandwidth. Although we mainly take XCP and QFCP as examples, we believe that the problems discussed here are common for other router-assisted congestion control schemes and the proposed wireless enhancements can also be applied to other protocols as well.

References

[1] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC2581 1999.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Trans. Networking*, vol. 5, pp. 756-769, 1997.

[3] T. V. Lakshman, U. Madhow, and B. Suter, "TCP/IP performance with random loss and bidirectional congestion," *IEEE/ACM Trans. Networking*, vol. 8, pp. 541-555, 2000.

[4] S. Biaz and N. H. Vaidya, "De-randomizing congestion losses to improve TCP performance over wired-wireless networks," *IEEE/ACM Trans. Networking*, vol. 13, pp. 596-608, 2005.

[5] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. ACM SIGCOMM '02–The Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Pittsburgh, PA, 2002, pp. 89-102.

[6] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor sharing flows in the Internet," in *Proc. Thirteenth International Workshop on Quality of Service (IWQoS)*, Passau, Germany, 2005.

[7] J. Pu and M. Hamdi, "New flow control paradigm for next generation networks," in *Proc. IEEE Sarnoff Symposium*, Princeton, NJ, 2006.

[8] G. Nychis, G. Sardesai, and S. Seshan, "Analysis of XCP in a wireless environment," Carnegie Mellon University 2006.

[9] Y. Zhang and T. R. Henderson, "An implementation and experimental study of the explicit control protocol (XCP)," in *Proc. IEEE INFOCOM 2005*, pp. 1037-1048.

[10] A. Falk, Y. Pryadkin, and D. Katabi, "Specification for the Explicit Control Protocol (XCP)," Internet draft 2006.

[11] "IEEE Std. 802.11b-1999, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band," 1999.

[12] F. Abrantes and M. Ricardo, "XCP for shared-access multi-rate media," *ACM SIGCOMM Computer Commun. Rev.*, vol. 36, pp. 27-38, 2006.

[13] D. M. Lopez-Pacheco and C. Pham, "Robust transport protocol for dynamic high-speed networks: enhancing the XCP approach," in *Proc. 13th IEEE International Conference on Networks*, Kuala Lumpur, Malaysia, 2005, pp. 404-409.

[14] 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 2004)*, Atlanta, GA, 2004.

[15] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Commun. Rev.*, vol. 36, 2006.

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

[17] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," RFC3782 (obsoletes RFC2582), 2004.

[18] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.

[19] V. Jacobson, "Congestion avoidance and control," in *Proc. ACM SIGCOMM '88*, Palo Alto, CA, pp. 314-329.

[20] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha, "Understanding TCP fairness over wireless LAN," in *Proc. IEEE INFOCOM 2003 Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, 2003, pp. 863-872.

Jian Pu (S'06) received his B.S. degree in Computer Science and Technology from the Nanjing University, Nanjing, China, in 2004. Currently, he is pursuing his doctoral studies in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology, Hong Kong, China. His research interests include the design and analysis of congestion control protocols, cross-layer optimization, bandwidth probing, and resource allocation for both wired and wireless networks.

Mounir Hamdi (S'90-M'91-SM'06) received the B.S. degree in Electrical Engineering from the University of Louisiana in 1985, and the MS and the PhD degrees in Electrical Engineering from the University of Pittsburgh in 1987 and 1991, respectively. He has been a faculty member in the Department of Computer Science at the Hong Kong University of Science and Technology since 1991, where he is now Full Professor of Computer Science, Director of the Computer Engineering Program, and Director of the Master of Science in Information Technology. He is a member of the University Senate and University Council. In 1999 to 2000 he held visiting professor positions at Stanford University, USA, and the Swiss Federal Institute of Technology, Lausanne, Switzerland. His general area of research is in high-speed wired/wireless networking in which he has published more than 220 research publications, received numerous research grants, and graduated more 20 PhD/Master students. In addition, he has frequently consulted for companies in the USA, Europe and Asia on high-performance Internet routers and switches as well as high-speed wireless LANs.

Dr. Hamdi is/was on the Editorial Board of *IEEE Transactions on Communications*, *IEEE Communication Magazine*, *Computer Networks*, *Wireless Communications and Mobile Computing*, and *Parallel Computing*. He was a guest editor of *IEEE Communications Magazine*, guest editor-in-chief of two special issues of *IEEE Journal on Selected Areas in Communications*, and a guest editor of *Optical Networks Magazine*, and has chaired more than 7 international conferences and workshops including The IEEE International High Performance Switching and Routing Conference, the IEEE GLOBECOM/ICC Optical networking workshop, the IEEE ICC High-speed Access Workshop, and the IEEE IPPS HiNets Workshop. He is/was the Chair of IEEE Communications Society Technical Committee on Transmissions, Access and Optical Systems, and Vice-Chair of the Optical Networking Technical Committee, as well as member of the ComSoc technical activities council. He is/was on the technical program committees of more than 120 international conferences and workshops. He received the best paper award at the International Conference on Information and Networking in 1998 out of 152 papers. In addition to his commitment to research and professional service, he is also a dedicated teacher. He received the best 10 lecturers award and the distinguished engineering teaching appreciation award from the Hong Kong University of Science and Technology, and various grants targeted towards the improvement of teaching methodologies, delivery and technology. He is a member of IEEE and ACM.