Applying Router-Assisted Congestion Control to Wireless Networks: Challenges and Solutions[1]

Jian Pu and Mounir Hamdi
Department of Computer Science and Engineering, HKUST, Hong Kong
{pujian, hamdi}@cse.ust.hk

***Abstract*—Router-assisted explicit-feedback congestion control protocols have recently been introduced to overcome the inefficiency problem of TCP in high bandwidth-delay product (BDP) wired networks. However, two main challenges are encountered when applying this kind of congestion control to wireless networks. One is how to distinguish the two kinds of packet loss (non-congestion-related loss and congestion-related loss) in lossy wireless networks as well as how to react to them accordingly and properly. The other is how to probe the unknown bandwidth capacity of a wireless link which is required in calculating router feedback. Through detailed analysis of these challenges, we have proposed some practical and novel enhancements on router-assisted congestion control for wireless environment. We have also implemented these enhancements in a router-assisted congestion control protocol called QFCP. Simulation results using ns-2 show that it can fairly allocate wireless bandwidth resource in heterogeneous networks.**

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. One reason is that TCP treats packet loss as congestion signal but it can not distinguish non-congestion-related loss (bit error) from congestion-related loss (buffer overflow) leading to underutilization of wireless link. Another reason is that after experiencing a packet loss TCP needs to take many RTTs to recover 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 [2], RCP [3], QFCP [4]) is originally introduced to overcome the inefficiency 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 sending rate within a few RTTs and maintain high utilization of the bottleneck link. While for lossy wireless networks, experiments [5] show that XCP can achieve higher throughput than TCP. However, we note that XCP's relatively better performance is largely due to its fast window growth instead of robustness to wireless loss, and there is still room for improvement.

First, XCP still can not differentiate the two kinds of packet loss. Actually in the current implementation of XCP the sender reacts to packet loss in the same way as TCP does [6]. The congestion window is halved and the throughput is slowed down unnecessarily when 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 networks.

Second, router-assisted congestion control requires an exact and fixed value of link bandwidth set as a parameter in the router's control algorithm in advance. But for a wireless link, due to shared-media access contention, half-duplex, and changing physical conditions, it is very hard to do such kind of setting. For example, for an 802.11b wireless node, it can even 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 error and performance deterioration for router feedback control. The authors of [5] 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 [7] also find that XCP does not work well on 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 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 [8] suggests that 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. We make the first effort to combine the two problems together and solve them in a single control framework. We choose QFCP [4] as the groundwork to develop a robust congestion control protocol that is suitable for wireless networks. In particular, we try to find a way to probe the link capacity independent of the buffer size so that

[1] This research was supported in part by the Hong Kong Research Grant Council under Grant RGC HKUST6260/04E.

the algorithm can be applied on routers with any buffer size or bandwidth capacity.

II. Router-Assisted Congestion Control

Generally router-assisted congestion control can refer to any mechanism that involves routers in congestion control such as kinds of Active Queen 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 approach. We now briefly describe XCP and QFCP as examples to show some details of this kind of congestion control.

XCP [2] 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:

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

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 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 on 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 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:

$$cwnd = \max(cwnd + feedback \cdot RTT, MSS),$$

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

QFCP [4] is another router-assisted congestion control protocol for high-speed networks. Unlike XCP, QFCP gives per-flow feedback on flow rate instead of per-packet feedback 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 as:

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

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

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

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. On receiving an ACK, the sender reads the feedback and adjusts its congestion window as:

$$cwnd = \max(feedback \cdot RTT, MSS),$$

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 the fair-share rate based on flow number estimation. Simulations show that both XCP and QFCP can achieve high utilization of large BDP links, but QFCP can further shorten flow completion time and help flows converge to fair-share rate faster [4].

III. Wireless Enhancements

There are two challenges addressed here as 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.

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 at any 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 for different physical 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 finds and sets 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:

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

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 the 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.
- 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.

As to deal with the burstiness nature of packet switching network, we actually use the weighted moving average of *output* and q in the above formula to smooth out possible oscillation caused by packet burst:

$$avg\_output = w \cdot output + (1-w)\cdot avg\_output ,$$

$$avg\_q = w \cdot q + (1-w)\cdot avg\_q ,$$

where the weight w is 0.2. Repeat the above probing procedure for each control interval and 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). And whenever this link becomes busy again, 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 rate feedback should be changed accordingly:

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

The parameter values (α, β) are chosen heuristically from experiments, and more precise and theoretical analysis for these parameters is left for future study.

B. Enhancement for Packet Loss

For a sender in lossy wireless environment, it had better differentiate two kinds of packet loss: for non-congestion-related loss (bit error), it should maintain the current window size; and for congestion-related loss (buffer overflow), it should slow down to prevent congestion collapse. Unfortunately, currently router-assisted congestion control protocols can not do such differentiation yet. For example, XCP simply inherits the standard TCP behavior when encountering packet loss [6]. That is, on receiving three duplicate ACKs, the congestion window *cwnd* is halved; and on retransmit timeout, *cwnd* is set to one. The assumption is that 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 error.

For TCP, the sender needs to slow down on detecting packet loss because 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 packets drops. Through this approach TCP finds the available capacity of the path. But for 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 treating packet loss as congestion signal now. In stead, it should use the information in the congestion header to adjust its congestion window. For example, in QFCP, if the loss is congestion-related, the rate feedback in the duplicate ACK will tell the sender to slow down; but if it is non-congestion-related loss, the rate feedback will probably be like the current sending rate of this flow.

We suggest that separate the data reliability control from congestion control when receiving duplicate ACKs. When the sender receives a duplicate ACK, it suggests 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:

$$cwnd = feedback \cdot RTT + num\_dupACK ,$$

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 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 complicate situations such as loss of the retransmission packet here and leave them for future study.

XCP is a little complicate 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 base on the feedback when encountering 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 [8] 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 firstly we need to distinguish the two kinds of loss in XCP. Intuitively we may say if the feedback is positive, it is non-congestion-related loss; and if the feedback is negative, it is congestion-related loss. However, the feedback is also used for fairness control. A negative feedback may possibly only want to change the flow's rate toward fair-share rate and may not necessarily suggest congestion. Halving the *cwnd* or change *cwnd* to 1 is too aggressive for this case. But if the loss is congestion-related, window adjusting only based on feedback may be not enough since some feedback on window reduction may be lost. In sum, unlike QFCP, it is not so easy for XCP to differentiate the two kinds of packet loss based on feedback information.

While for packet loss event triggered by retransmit timeout, since no feedback information available at this instant and the loss may be caused by severe congestion, conservatively set congestion window to one should be better. And if this is not a congestion loss, 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

$$virtual\_q = q + num\_drop .$$

Thus, if packets are dropped by routers, the feedback computed using the virtual queue length can still precisely reflect the congestion condition.

IV. Simulation Results

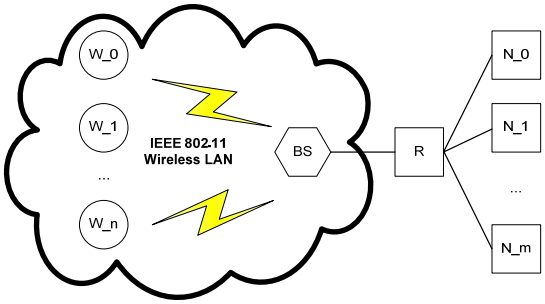


Fig. 1. Simulation network topology

In this section we present simulation results using ns-2 [9]. 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. All wired links are duplex links with fixed 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 1000 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 and use DropTail as the queue management scheme. 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 enhancement" for convenience.

A. One Flow Dynamics

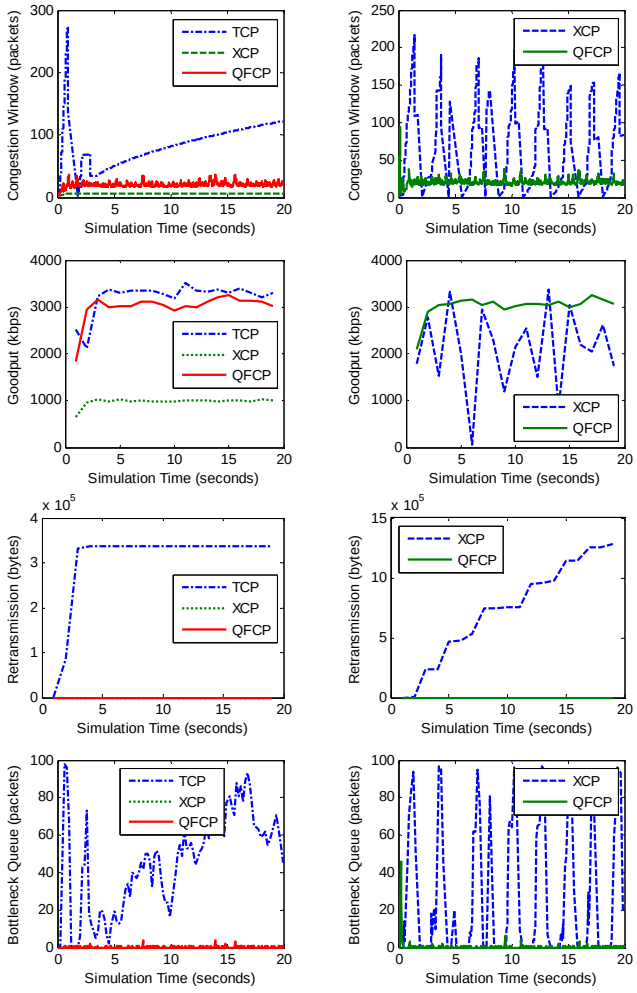
Here we simulate one downloading flow from a wired node to a wireless node in the WLAN to demonstrate the dynamics of three different protocols: TCP, XCP and QFCP. For router-assisted protocols (XCP and QFCP), the bandwidth parameter C of any wireless interface is set to 1 Mbps for Fig. 2(a) and 10 Mbps for Fig. 2(b). Since TCP does not need such setting and it behaves in the same way for both cases, TCP is omitted in Fig. 2(b). We record the congestion window size, the goodput, the retransmitted bytes, and the bottleneck queue length.

For TCP, it is able to achieve high goodput but at the cost of massive packet drops and large persistent queue. 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.

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. If C is

underestimated (Fig. 2(a)), its goodput is low and the link is underutilized. If C is overestimated (Fig. 2(b)), its feedback tends to increase the flow rate to an unachievable value, which causes the queue to grow toward infinity and the buffer to overflow. That is why many packets are dropped and the queue length oscillates between zero and the maximum buffer size.

For QFCP, the results show that the proposed probing algorithm works quite well no mater what the initial value of link bandwidth is set to. QFCP always finds the correct estimation of the wireless link capacity in a short time. Its goodput is high and stable, and its queue length is small.



(a) C = 1 Mbps (b) C = 10 Mbps

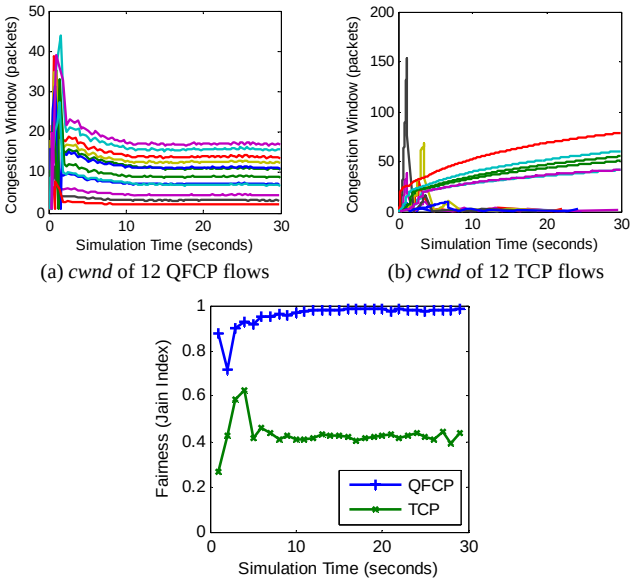
Fig. 2. Performance comparison of TCP, XCP and QFCP on wireless link with unknown bandwidth

B. *Multiple Flows with Different RTTs*

In this simulation there are twelve flows 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 [10] to evaluate the fairness of bandwidth allocation among competing flows:

$$J = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2}$$

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 every 1 second.



(a) *cwnd* of 12 QFCP flows (b) *cwnd* of 12 TCP flows

(c) Jain Fairness Index

Fig. 3. Twelve flows in both directions sharing a wireless link

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 since 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* * *RTT*) and achieve good fairness on the throughput (Fig. 3(c)).

TCP also has its embedded algorithm to probe the available bandwidth, which is the additive increase multiplicative decrease (AIMD) algorithm. 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. 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)). 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 for 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. Fairness between Lossy and Lossless Flows

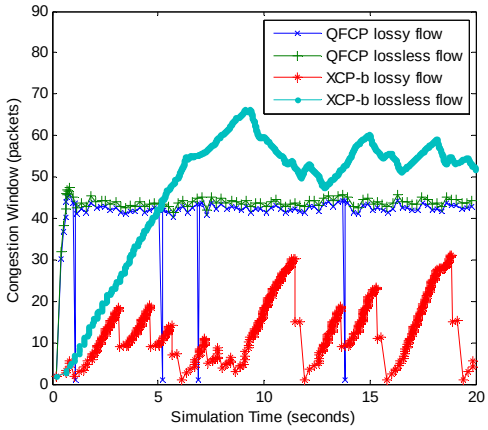


Fig. 4. One lossy flow competing with one lossless flow

This scenario tests for the protocol's sensitivity to non-congestion-related loss. A uniform loss model is injected 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 packet loss rate = 0.01. The bandwidth parameter *C* is initially set to 1 Mbps since we do not know the exact bandwidth capacity for each wireless node and expect the probing algorithm to find it. The minimum round-trip time is 200 ms. We compare the performance of QFCP and XCP-b in this scenario.

XCP-b [7] is a XCP variant enhanced with an algorithm to probe the available bandwidth when the link capacity is unknown. However, it does not take packet loss into account. As shown in Fig. 4, XCP-b 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. Furthermore, when one lossy flow competes with one lossless flow, XCP-b treats the lossy flow as a congested flow and unfairly allocates bandwidth between them.

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). But if this is a non-congestion-related loss, any subsequent ACK will recover the window to the proper size. For the situation of one lossy flow competing with one lossless flow, QFCP can fairly allocate the bandwidth resource 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.

V. Conclusion

Originally router-assisted congestion control protocols such as XCP are only designed for point-to-point full-duplex wired links. Though they show good performance on such links, they experience performance deterioration in wireless networks. One reason is that the sender still can not distinguish the two kinds of packet loss (congestion-related and non-congestion-related) and slow down its throughput unnecessarily on bit error loss. 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 on 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 tell the sender how to adjust its congestion window. And we 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 implement the wireless enhancements in QFCP and present some simulation results in ns-2. Results show that enhanced QFCP can work well on lossy wireless link with unknown bandwidth. Although we mainly take XCP and QFCP for example in explanation, we believe that the wireless enhancements described here can also be applied to other router-assisted congestion control schemes.

References

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

[2] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM*, Pittsburgh, Pennsylvania, USA, 2002, pp. 89-102.

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

[4] J. Pu and M. Hamdi, "New Flow Control Paradigm for Next Generation Networks," in *IEEE Sarnoff Symposium*, Princeton, USA, 2006.

[5] G. Nychis, G. Sardesai, and S. Seshan, "Analysis of XCP in a Wireless Environment," www.andrew.cmu.edu/user/gnychis/xcp_wireless.pdf.

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

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

[8] D. M. Lopez-Pacheco and C. Pham, "Robust transport protocol for dynamic high-speed networks: enhancing the XCP approach," in *13th IEEE International Conference on Networks*, 2005, p. 6.

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

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