Improving Quality of Service for Congestion Control in High-Speed Wired-cum-Wireless Networks

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

***Abstract*-TCP is currently the dominate congestion control protocol for the Internet. However, as the Internet evolves into a high-speed wired-cum-wireless hybrid network, performance degradation problems of TCP have appeared, such as underutilizing high-speed links, regarding wireless loss as congestion signal, and unfairness among flows with different RTTs. In order to improve the quality of service for such high-speed hybrid networks, we propose a router-assisted congestion control protocol called Quick Flow Control Protocol (QFCP). Performance evaluation using Network Simulator NS-2 shows that QFCP can significantly shorten flow completion time, fairly allocate bandwidth resource, and be robust to non-congestion-related loss.**

I. INTRODUCTION

TCP [1] has been the dominate congestion control protocol for the Internet since 1980's. However, it also demonstrates some performance degradation in nowadays high-speed wired-cum-wireless networks. First, TCP's Additive Increase Multiplicative Decrease (AIMD) algorithm is too conservative for high-speed or long-delay links. After experiencing a packet loss, TCP needs to take many Round-Trip Times (RTTs) to recover the high throughput as it only increases the sending window by one packet per RTT. Flows often need more time to finish than expected by the users and large capacity of the bandwidth is wasted. Second, TCP assumes any packet loss as congestion signal, but it can not distinguish non-congestion-related loss (transmission bit error) from congestion-related loss (router buffer overflow) leading to underutilization of wireless link. Third, TCP can not fairly allocate bandwidth resource among competing flows with different RTTs, or among uploading and downloading flows in IEEE 802.11 Wireless LAN.

As more and more high-speed (e.g., optical fibers), long-delay (e.g., satellite links, trans-ocean cables), and wireless (e.g., WLAN, CDMA) links will be employed in the Internet, this situation will continue and may even be worse. Users will complain for the poor quality of service though they have paid for the costly network equipments or services. If we want to improve the Quality of Service (QoS) for congestion control in high-speed wired-cum-wireless networks, we must design an advanced mechanism to utilize the network resource more efficiently and more fairly. There are many QoS criteria regarding to network communication such as delay jitter, drop rate, priority service and so on. Here we will focus on three aspects that the Internet users may be most interested in, i.e., flow completion time, fair bandwidth allocation, and robustness to wireless loss.

The remainder of the paper is organized as follows. Section II introduces the control architecture and algorithm of the proposed congestion control protocol. Section III evaluates the performance of the protocol using Network Simulator NS-2 and compares it with other existing protocols. Section IV concludes our work.

II. PROTOCOL DESIGN

Quick Flow Control Protocol (QFCP) [2] is a router-assisted congestion control protocol for high-speed wired-cum-wireless networks. There are some other router-assisted protocols such as Quick-Start [3], XCP [4], and RCP [5]. The common design incentive is that routers are the places where congestion happens and with explicit feedback from routers we should be able to utilize the network resource more efficiently.

Unlike XCP, QFCP gives per-flow feedback on flow rate instead of per-packet feedback on window adjustment. There are three fields in the QFCP header of each packet: RTT, rate-request, and rate-feedback. We use a similar framework of Quick-Start but we extend the rate-request-and-grant mechanism to the whole lifetime of a flow: (1) The sender sets the initial value of rate-request field in the header of each outgoing packet to be the desired sending rate of this sender; (2) When the packet reaches a router, the router compares the value in rate-request field with the router's own fair-share rate and puts the smaller one back into that field; (3) On receiving the packet, the receiver copies the rate-request field into the rate-feedback field of the corresponding ACK packet and sends it back to the sender; (4) When the sender receives the ACK packet, it reads the value in the rate-feedback field and adjusts its sending rate accordingly.

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:

$$N(t) = \frac{y(t)}{R(t-T)}, \quad (1)$$

1930-529X/07/$25.00 © 2007 IEEE

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:

$$R(t) = \frac{C - \beta \cdot \frac{q(t)}{T}}{N(t)}, \tag{2}$$

where C is the capacity of the output link, q is the minimum queue length observed in the last control period T, 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:

$$cwnd = \max(feedback \cdot RTT, MSS), \tag{3}$$

where *feedback* is the routers' feedback on flow rate, *RTT* is the round-trip time measured by the sender, and *MSS* is the maximum segment size. 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.

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:

$$C = \begin{cases} output, & if \quad q \geq 1 \\ (1+\alpha) \cdot C, & else \end{cases}, \tag{4}$$

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 as following:

- 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.

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 has 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 subsequent ACK (or dup-ACK) will tell the sender to slow down; but if it is non-congestion-related loss, the subsequent rate feedback will probably be similar to 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:

$$cwnd = feedback \cdot RTT + num\_dupACK \tag{5}$$

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 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

1930-529X/07/$25.00 © 2007 IEEE

expected by the routers. XCP-r [7] 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

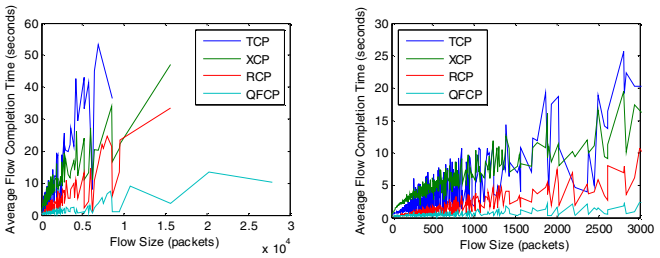
$$virtual\_q = q + num\_drop . \quad (6)$$

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

III. Performance Evaluation

A. Flow Completion Time

Research on the Internet traffic has revealed an important feature at the flow level: most of the flows are very short, while a small number of long flows account for a large portion of the traffic [8], [9]. This is also known as the heavy-tailed distribution. For fixed-size flows (e.g., FTP, HTTP), the most attractive performance criterion is the flow completion time (FCT). Here we simulate a scenario where a large number of Poison-arriving Pareto-distributed-size flows share a single bottleneck link of 150 Mbps. The total flow number is 60000. The common Round-Trip Propagation Delay (RTPD) of all flows is 100 ms. Flows arrive as a Poison process with an average rate of 625 flows per second. The packet size is 1000 bytes. The flow sizes are Pareto distributed with a mean of 30 packets and a shape parameter of 1.2. Thus, the offered traffic load on the bottleneck link can be estimated as: $8 * packet\_size * mean\_flow\_size * flow\_arrival\_rate / bandwidth = 1$. We record the size and completion time for each flow in the simulation, then average the flow completion time for flows with the same size.



(a) all flows (b) flows with size < 3000 packets

Fig. 1. Average flow completion time (AFCT) vs. flow sizes for Poison-arriving Pareto-distributed-size flows. (a) is the global picture for all flows. (b) is a close look at short flows

Note that in order to make the result more accurate, we don't wait for all flows to finish, but stop the simulation just on the arrival of the 60000th flow. The reason is that the offered traffic load may drop after the arrival of the last flow, since all active flows are going to finish but no new flow is going to join. Each simulation is conducted for each protocol: TCP-Reno, XCP [4], RCP [5], and QFCP. The scenario settings and the input data (i.e., the size and arriving time of each flow) are identical for each simulation. The results show that the Average Flow Completion Time (AFCT) in QFCP is significantly shorter than that in TCP, XCP or RCP.

For TCP, the AFCT is very oscillatory again the flow size. The reason is that although the exponential increase of congestion window in Slow Start does help some short flows finish quickly, the duration of other flows are prolonged due to packet loss. And we should point out that Slow Start is not a good way to shorten the duration of flows, because it actually does not know the proper initial sending rate but just intends to fill up the buffer of routers and cause packet losses, which prolongs the flow duration.

For XCP, it does not use Slow Start. Instead, when new flows join, XCP tries to reclaim the bandwidth from the ongoing flows and reallocate it to the new flows little by little. For short flows they may finish before reaching the fair sending rate. That is why the completion time of short flows in XCP is the longest. However, the AFCT against flow size in XCP is more stable than in TCP because XCP flows experience fewer packet losses.

For RCP and QFCP, both of them give a high initial sending rate to new flows based on the feedback from routers and help short flows finish quickly. However, the formula used in RCP to estimate the number of flows holds only when the input traffic just fills up the link capacity C, otherwise it leads to wrong estimation of the flow number. This wrong estimation of flow number makes the rate allocation in RCP under-optimum and thus prolongs the FCT in general compared with QFCP.

B. Fairness of Bandwidth Allocation

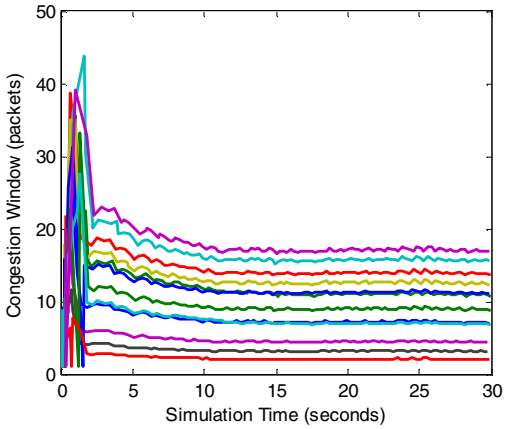
In this simulation there are twelve flows start transferring data at time 0. Flows are randomly generated in both directions, either from a node in the wired network to a wireless node in the IEEE 802.11 WLAN (downloading) or

1930-529X/07/$25.00 © 2007 IEEE

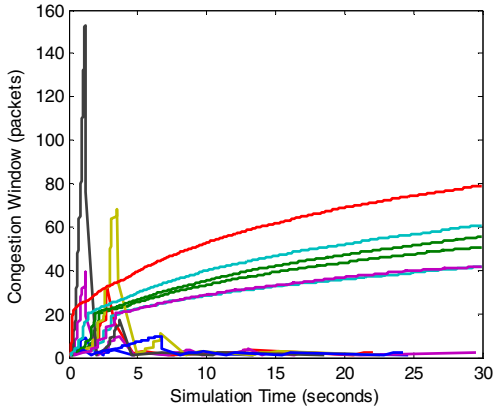
opposite (uploading). The flows' round-trip propagation delays vary from 40 ms to 370 ms. We use the Jain Fairness Index [10] to evaluate the fairness of bandwidth allocation among competing flows:

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

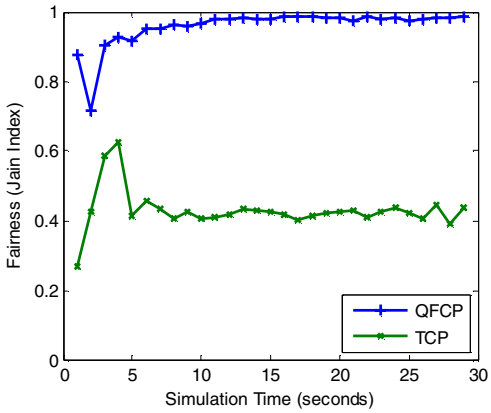
where x_i is the average throughput of flow i during a interval and n is the number of flows. We compute the Jain Fairness Index every 1 second.



(a) Congestion window of 12 QFCP flows



(b) Congestion window of 12 TCP flows



(c) Jain Fairness Index

Fig. 2. 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. 2(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. 2(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. 2(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. 2(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. 2(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. *Robustness to Wireless Loss*

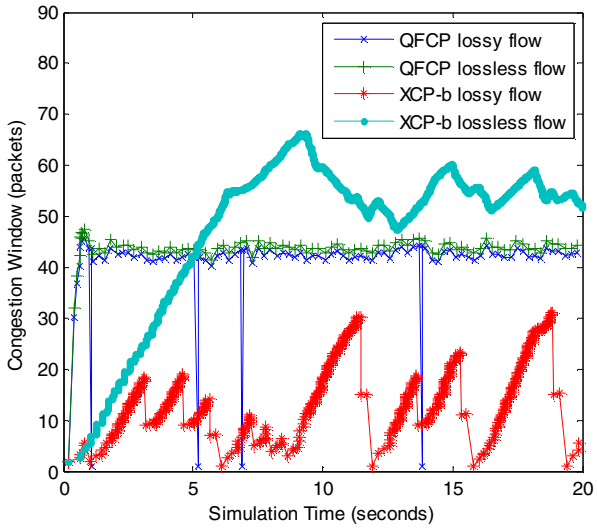


Fig. 3. One lossy flow competing with one lossless flow

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

1930-529X/07/$25.00 © 2007 IEEE

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 [11] 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. 3, 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. 3). 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.

IV. Conclusion

Quality of Service has been studied extensively for real-time multimedia flows. However, it is a little strange that the service quality of other common TCP flows (HTTP, FTP, email transfer, etc.) has seldom been studied. In this paper we point out that TCP suffers performance degradation in high-speed wired-cum-wireless networks. In order to improve the QoS we propose a router-assisted congestion control protocol named Quick Flow Control Protocol (QFCP). Through simulations using NS-2 and compared with other existing protocols, we show that QFCP can significantly shorten flow completion time, allocate bandwidth resource fairly among competing flows, and be robust to non-congestion-related loss.

References

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

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

[3] A. Jain, S. Floyd, M. Allman, and P. Sarolahti, "Quick-Start for TCP and IP," in *IETF Internet-draft, work in progress*, 2005.

[4] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *Proceedings of Proceedings of ACM SIGCOMM '02*, 2002, pp. 89-102.

[5] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor Sharing Flows in the Internet," in *Proceedings of International Workshop on Quality of Service*, 2005.

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

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

[8] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web traffic: evidence and possible causes," *IEEE/ACM Transactions on Networking,* vol. 5, pp. 835-846, 1997.

[9] K. Claffy, G. Miller, and K. Thompson, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," in *Proceedings of Proceedings of INET '98*, 1998.

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

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

1930-529X/07/$25.00 © 2007 IEEE