Can API-RCP be TCP Friendly with RED?

Yang Hong and Oliver W.W. Yang
School of Information Technology and Engineering, University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
E-mail: {yhong, yang}@site.uottawa.ca

Abstract—TCP is widely implemented for congestion control in current IP networks. As the network bandwidth increases, TCP becomes oscillatory and prone to instability, regardless of the queuing scheme. As a rate-based control scheme, XCP was proposed to obtain high link utilization in high bandwidth-delay product networks, while maintaining small queue size in the routers. XCP explicit feedbacks the congestion information in the bottleneck link to the source, and brings a flurry of research interests recently. However, XCP cannot settle to zero steady-state error due to the capacity estimation error, and may lead to arbitrarily low link utilization. Using solid control theoretical analysis and design, API-RCP has solved the potential problems of XCP successfully. Furthermore, API-RCP has simpler structure for real-time implementation. Due to its satisfactory transient network performance and its stability robustness to the dramatic change of the network traffic, API-RCP is a promising algorithm for practical implementation. Why can API-RCP solve the modeling deficiency of XCP? Can API-RCP be TCP friendly with TCP/RED (the current dominant congestion control mechanism in the Internet) in the same router? To answer these two questions, we have made control theoretical comparison of API-RCP with XCP and run OPNET simulation when the API-RCP and TCP/RED co-exist in the same router.

Keywords—Congestion Control, TCP Friendly Rate Control, TCP/RED

1. INTRODUCTION

An increasing number of subscribers are requesting more network bandwidth from Internet service providers. Multimedia applications and wireless components are becoming popular in the Internet. How to utilize the limited Internet resources and to provide a SLA (Service Level Agreement) has been a critical factor in the development of the Internet. To avoid the network collapse due to the congestion, TCP (Transmission Control Protocol) is widely implemented for congestion control in current IP (Internet Protocol) networks [1]. In TCP/IP networks, the router applies AQM (Active Queue Management) algorithm (e.g., RED (Random Early Detection) [2], PI-RED [3, 4] etc.) to implicitly inform the source the network congestion by dropping the packets. In the mean time, the source uses the AIMD (Additive Increase and Multiplicative Decrease) algorithm [5] to adjust its window size to prevent the congestion. As the network bandwidth increases, AIMD mechanism has begun to reach its limit. Gigabit-per-second file transfers, lossy wireless links, and high latency connections are all driving current TCP/AIMD congestion control outside of its natural operating regime. The resulting performance problems are of great concern for important network applications [6]. For example, when packets are lost due to wireless effects (e.g., channel fading) rather than congestion, TCP unnecessarily cuts the source sending rate by half, resulting in inefficient channel utilization and violation of QoS (Quality of Service). It has also been shown that, as a classic window-based control protocol, TCP cannot provide relatively smooth flow throughput or guarantee small jitter for streaming media traffic [7].

Rate-based control allows the sources to adjust their sending rates based on the feedback congestion information and to provide relatively smooth flow throughput for best-effort service traffic. For example, TFRC (TCP Friendly Rate Control) scheme proposed an equation in the source to calculate the source sending rate for unicast traffic based on packet loss event [7, 8]. However, most of current rate-based control algorithms for IP networks are heuristic feedback control algorithms that lack control theoretical analysis, and they cannot always guarantee the closed-loop stability in different traffic conditions [9]. XCP (eXplicit Congestion Control Protocol) [6, 10] attempts to obtain high link utilization in high bandwidth-delay product networks, while maintaining small queue size in the routers. However, it has been revealed that XCP cannot settle to zero steady-state error due to the capacity estimation error, and therefore cannot utilize wireless channels efficiently [11, 12]. Specifically, for any given parameters, the flow rates under XCP can be far from their max-min allocations in some network topologies [13]. A poor choice of parameter values could lead to arbitrarily low link utilization [13]. Using solid control theoretical design and analysis, API-RCP (Adaptive Proportional-Integral Rate Control Protocol) [9] has solved these potential problems of XCP successfully, and has also achieved performance comparable to XCP. Like XCP, API-RCP is designed to deliver the highest possible end-to-end throughput over a broad range of network infrastructure, including links with very large bandwidth-delay products. An API-RCP router is required to calculate an advertised source sending rate (carried in each data packet) based on the congestion state. All the controlled flows with different round-trip times receive the same rate in the same link, guaranteeing the fairness of bandwidth allocation. Every packet carries a field for the lowest rate along the path. An API-RCP router feedbacks the router congestion information explicitly without
recording per-flow state, which is different from other high speed TCPs that keep per-flow state in the routers and infer the router congestion state implicitly [12].

The main contributions of this paper are: (1) Making a control theoretical analysis and a comparison between API-RCP and XCP to clarify the modeling deficiency of XCP in [11]; (2) Using a heuristic method to provide the reason and mechanism of making API-RCP TCP-friendly with TCP/RED [2]; (3) Running OPNET simulations to verify that API-RCP can be TCP-friendly with TCP/RED in the same AQM router independent of the link capacity (e.g., a low of 45Mbps vs a high link capacity of 1Gbps). To the best of our knowledge, no performance evaluation and simulation of rate-based control algorithm has been conducted under a bottleneck router with a link capacity of 1Gbps or beyond. This TCP-friendly rate control mechanism and its successful Linux-based implementation in [14] have made API-RCP a promising rate control protocol for multimedia and wireless application in the next-generation Internet.

The rest of this paper is organized as follows. Section 2 make control theoretical comparison between API-RCP and XCP. Section 3 proposes how API-RCP can be TCP friendly with TCP/RED. Section 4 performs simulations to verify that API-RCP can TCP friendly co-exist with TCP/RED in the same AQM router. Finally, conclusions are given in Section 5.

2. CONTROL THEORETICAL COMPARISON of API-RCP and XCP

Fig. 1 depicts a conventional feedback control system, where \( C(s) \) and \( P(s) \) are the controller and the plant, respectively. The signals \( r, u(t), \) and \( y(t) \) are the reference input, the controller output and the plant output, respectively [15, 16].

An example of using XCP controller is shown in Fig. 2. The controller \( C(s) \) (e.g., XCP controller) compares the actual value of the plant output \( y(t) \) (e.g., the aggregated incoming traffic rate of the XCP router) with the reference input (e.g., the link capacity or the service rate of the XCP router). It then determines the deviation (e.g., spare bandwidth of the XCP router), and produces a control signal (e.g., the advertised XCP flow throughput or source sending rate) that will reduce the deviation to zero or a small value [17], as shown in Fig. 2.

2.1 Control Theoretical Analysis of XCP

In the AQM control system with XCP controller in Fig. 2, the transfer function of the system “plant” is \( P(s) = \frac{e^{-\tau t}}{s} \), where \( \tau \) is the average round trip time, \( t \) of all XCP flows. The XCP controller calculates the advertised change of source window size (or the change of source sending rate \( \Delta q_i(t) \)), and then the XCP controller output (or the control signal) becomes the advertised source sending rate \( q_i(t) = q_i(t) + \Delta q_i(t) \). The AQM control system output \( y(t) \) is the aggregated arrival rate of the XCP flows entering the router, which can be expressed as \( y(t) = \sum q_i(t - \tau_i) = \sum q_i(t - \tau_i) \). The parameters \( \tau_i \) and \( \tau_i' \) are the round trip time and the forward path delay of the ith XCP flow. The reference input of the AQM control system is the link capacity \( C_i(t) \). With this introduction, we can now revisit and clarify the XCP modeling deficiency (previously presented by [11]) using the control theoretical analysis.

Inaccuracy in AQM Control System Output

Since XCP router does not maintain per-flow information, it relies on the source sending rate \( q_i(t) \). The actual source sending rate \( q_i(t) \) may be less than the advertised value \( q_i(t) \) when the current XCP router is not a bottleneck router for the ith flow, or when there are limiting factors at the destination (e.g., low receiver-window size). This would result in a mismatch between the perceived value and the actual measured value of the aggregated incoming traffic rate [12]. The XCP controller would consider this difference as a part of the spare bandwidth (i.e., the deviation between the control system output and the reference input), thus yielding an inaccurate advertised source sending rate (i.e., the controller output or the control signal). This may drive the source sending rate away from a steady state (or a stable value), thus causing an instability in the control system.

Inaccuracy in AQM Control System Input

Wireless networks have continued to develop and its applications have significantly grown. According to the Shannon’s capacity theorem [18, 19], when SNR (Signal-Noise Ratio) changes with the wireless channel fading, the upper-limit of the link capacity \( C(t) \) would change dynamically. XCP router may mis-estimate the link capacity
Nyquist’s stability criterion [15, 17]. Therefore, using the solid performance in source sending rate and the router queue size advertised source sending rate API-RCP flows. The API-RCP controller calculates the output queue size, which is specified by a user instead of an controller parameters are obtained based on a proper phase margin. Positive phase margin can always guarantee the stability of the AQM control system, in accordance with the Nyquist’s stability criterion [15, 17]. Therefore, using the solid control system design and analysis, API-RCP solves the modeling deficiency of XCP successfully.

In summary, the difference between API-RCP and XCP lies in the accuracy of API-RCP to track the system input and the capability of the API-RCP controller to cancel the steady state error.

3. CAN API-RCP be TCP FRIENDLY with RED?

RED [2] has been widely implemented in the current TCP/AQM routers. RED attempts to clamp the instantaneous queue size (or the average queue size) around a fixed target queue size that can be specified by a user. Obviously, both RED and API-RCP use a target queue size to restrict the queueing delay and to avoid the buffer overflow. We can set up two sub-queues to store the TCP/RED packets and the API-RCP packets respectively. The router can servers the RED queue and API-RCP queue alternately so that the TCP/RED packets and the API-RCP packets can share the link capacity equally. By this approach, the API-RCP can co-exist (be TCP-friendly) with TCP/RED in the same AQM router, and fairness in bandwidth allocation between API-RCP and TCP/RED is guaranteed.

4. PERFORMANCE EVALUATION and SIMULATIONS

We would like to run simulations using OPNET Modeler [20] to verify that the API-RCP can be TCP-friendly with TCP/RED in the same AQM router. We consider two types of TCP Reno sources that use the Fast-Retransmit and Fast-Recovery mechanisms in the simulations – the long-lived sources (i.e., controlled FTP sources for best-effort service traffic), which always have IP packets to send as long as their congestion windows allow; and the short-lived sources (i.e., HTTP sources), which enter the network after a random period of think time, send a file of a random length, and then wait for another think time period. The API-RCP sources also adopt window control mechanism of TCP Reno. The destination’s advertised window size is set sufficiently large so that the throughputs of the TCP connections are not constrained at the destination.

![Figure 3: AQM Control System with API-RCP Controller](image)

![Figure 4: Network Topology for Performance Evaluation](image)
following simulations, 50 greedy FTP sources, 50 HTTP sources and 50 API-RCP sources were used between the time \(t=0\)sec and \(t=100\)sec, while 100 greedy FTP sources, 100 HTTP sources and 100 API-RCP sources were used between the time \(t=100\)sec and \(t=200\)sec. All the IP packets have the same size of 1024 bytes. The maximum window size allowed for all the TCP sources is specified to be 2000packets. The buffer size \(B\) of the AQM Router 1 (or the bottleneck router) is determined based on the different bandwidth-delay product [3], while the buffer size of the Router 2 is big enough so that no packets will be lost. The RTPDs (Round-Trip Propagation Delays) between the sources and the destinations are uniformly distributed between 80ms and 240ms with the mean of 160ms. Taking account in the queuing delay, the average round trip time can be approximated as 0.22s. The choice of RED parameters can be referred to [2, 3]. The target queue size of API-RCP is specified to be 30% of the buffer size, i.e., \(x_0=0.3B\) from our many simulation experiments (some of which can be found in [9]). The phase margin for API-RCP is specified to be \(\phi_m=45^0\) based on the control engineering practice [9, 16].

Figure 5 shows the instantaneous queue size of RED sub-queue and API-RCP sub-queue. It is obvious that the queue size in the RED sub-queue fluctuates with large amplitude. Between the time \(t=0\)s and \(t=100\)s, RED clamps sub-queue size around a target queue size (about 200packets). The new target queue size of RED sub-queue increases to about 250packets. In the mean time, the queue size of API-RCP sub-queue always remains around the target queue size (or 300packets) with little oscillation, and is quite stable upon the great traffic change at time \(t=100\)s. Because the phase margin had gone outside its specified interval, API-RCP controller self-tuned and gave a new \(\phi_m=45^0\). This has resulted in the instantaneous queue size of API-RCP still clamping back down to the target queue size of 300 packets very quickly.

Figure 6 shows the instantaneous sending rates (i.e., flow throughputs) of a controlled FTP source and an API-RCP source (both with \(RTPD=160\)ms). The source sending rate of TCP/RED exhibits significant fluctuation. API-RCP provides a relatively smooth source sending rate required by streaming media traffic [7]. Between the time \(t=0\)s and \(t=100\)s, the source sending rate of API-RCP is around 45Mbps/100=450Kbps. After time \(t=100\)s, it becomes 45Mbps/200=225Kbps, well matching our expectation and guaranteeing the fairness in bandwidth allocation for different TCP/RED and API-RCP flows.

4.1 Case of a Low Link Capacity
In this case, we would like to investigate the performance of API-RCP and TCP/RED under a bottleneck router (or AQM router 1) with a low link capacity of T3 (44,736,000bps, i.e., 5,461packets/sec). The buffer size of two sub-queues is 1000packets each. For RED, the maximum value for drop probability \(p_{\text{max}}\) is 0.1; the maximum queue threshold \(\text{max}_{\text{th}}\) is 700; the minimum queue threshold \(\text{min}_{\text{th}}\) is 150; the queue weight \(w_q\) is 0.002. For API-RCP, the target queue size is 300packets.

4.2 Case of a High Capacity
In this case, we would like to evaluate the performance of API-RCP and TCP/RED under a bottleneck router with a high link capacity of 1Gbps (i.e., 122,070packets/sec). The buffer size of two sub-queues is 20,000packets each. For RED, the maximum value for drop probability \(p_{\text{max}}\) is 0.1; the maximum
queue threshold $max_{th}$ is 14,000; the minimum queue threshold $min_{th}$ is 3,000; the queue weight $w_q$ is 0.002. For API-RCP, the target queue size is 6,000 packets.

Figure 7 shows the instantaneous queue size of RED sub-queue and API-RCP sub-queue. It can be seen that the queue size in the RED sub-queue fluctuates around 2,500 packets with noticeable amplitude. At the same time, the steady queue size of API-RCP sub-queue always remains exactly on a target queue size (or 6,000 packets) without any oscillation except some spike upon the great traffic change at time $t=100s$. It demonstrates that API-RCP is robust to the modification of the system plant (or the traffic change).

Figure 8 shows the instantaneous sending rates (i.e., flow throughput) of a controlled FTP source and an API-RCP source (both with $RTPD=160ms$). Source sending rate of TCP/RED fluctuates with dramatic amplitude. Flow throughput of API-RCP is still relatively smooth with very small oscillation. Between the time $t=0s$ and $t=100s$, the source sending rate of API-RCP is around $1Gbps/100=10Mbps$. After time $t=100s$, it becomes $1Gbps/200=5Mbps$. Smooth flow throughputs of both 10Mbps and 5Mbps can guarantee the QoS for delivering IPTV to numerous Internet subscribers.

In summary, API-RCP can guarantee a relatively smooth flow throughput and fairness in bandwidth allocation no matter what a low link capacity or a high link capacity. In the mean time, API-RCP can also be TCP-friendly with current dominant TCP/RED in the Internet.

5. CONCLUSIONS

We have exposed the difference between API-RCP and XCP using a control theoretical analysis and demonstrated the capability of API-RCP in solving the potential problems of XCP. On the other hand, we use a heuristic analysis to propose how API-RCP can be made TCP friendly with TCP/RED in the same AQM router. OPNET simulation has been conducted successfully under two different link capacities of 45Mbps and 1Gbps. Performance evaluation has verified that API-RCP and TCP/RED can share the network bandwidth friendly and fairly, on matter what a high capacity or a high capacity. API-RCP can provide a relatively smooth and high source sending rate for multimedia traffic (e.g., IPTV) and effectively exploit the link capacity of the wireless networks (e.g., WiMAX).

ACKNOWLEDGEMENT

We would like to appreciate the help from Mr. HanLiu Chen on OPNET simulations. We would also like to acknowledge the financial support from NSERC Research Discovery Grant (#RGPIN42878) and the partial support from the industrial partners, through the Agile All-Photonic Networks (AAPN) Research Network, NSERC.

REFERENCES


W.R. Stevens, TCP/IP Illustrated, Volume 1, Addison-Wesley, Boston, 1994.


