General Congestion Control for High Bandwidth-Delay Product Networks

Chengnian Long, Xudong Chai, Xinping Guan  
Dept. of Automation  
Shanghai Jiao Tong University  
Shanghai, China  
longcn@sjtu.edu.cn

Qian Zhang  
Dept. of Computer Science and Engineering  
Hong Kong University of Science and Technology  
Hong Kong, China  
qianzh@cse.ust.hk

Abstract—Existing congestion control protocols have significant limitations in achieving high throughput and reasonable fairness while maintaining fast convergence speed in high bandwidth-delay product networks. In this paper, we propose the General Congestion Control Protocol (GCCP) to address this limitation. GCCP allows for aggressive behavior in large underutilized links to achieve high throughput, but leverages only one ECN bit for network utilization feedback. Once the link is sensed to be highly utilized, the protocol dedicates to fair and rapid bandwidth allocation by requiring congestion window increment is conservative and monotone decreasing with congestion window increasing. The ns2 simulations show that GCCP achieves a pretty good tradeoff between high throughput and reasonable fairness while exhibiting fast convergence speed.

Keywords— congestion control; convergence; fairness; TCP; throughput

I. INTRODUCTION

TCP congestion control algorithm has performed remarkably well in the Internet in general, since its proposal in 1988 [1]. Unfortunately, the Additive Increase Multiplicative Decrease (AIMD) congestion control mechanism employed by TCP [2] is known to prevent TCP from effectively utilizing available bandwidth in high bandwidth-delay product (BDP) networks.

A great deal of new high-speed congestion control protocols have been proposed for this problem. These can be mainly classified into two categories: sender side and network feedback based protocols. The sender side protocols focus on modifications to the congestion response function of TCP itself for addressing its limitation in high BDP networks. They adjust the congestion window (cwnd) by detecting packet loss and/or queuing delay at the sending hosts, such as HSTCP [3], STCP [4], BIC-TCP [5], LTCP [6], FAST TCP [7], TCP-Africa [8], Compound TCP [9], and TCP-Illinois [10]. Although the sender side protocols are shown to overcome TCP’s deficiency in high BDP networks, using loss and/or delay as the only congestion signal(s) poses fundamental limitations on achieving high throughput and reasonable fairness while maintaining fast convergence speed. For the loss-based protocols, such as HSTCP and STCP, it is essential to be highly aggressive to achieve high throughput. However, this aggressive nature also causes severe RTT (Round Trip Time) unfairness [5], bad TCP friendliness [3], and slow convergence speed [11]. On the other hand, the protocols containing delay component, such as FAST TCP, TCP-Africa, and Compound TCP, achieve high throughput and reasonable fairness. Nevertheless, the performances of these algorithms highly depend on the accurate measurement of the delays. In the real networks, delay cannot be measured accurately and usually the RTT measurement is buried with noise. In addition, a recent experimental evaluation [12] shows that for both Compound TCP and TCP-Illinois link utilization would be low and network responsiveness become sluggish as BDP increases.

To address some of the limitations of sender side protocols, many researchers have proposed the network feedback based protocols, such as XCP [13], VCP [14]. XCP has routers decouple efficiency control and fairness control and then explicitly communicate the rate to end-hosts to cause the system to converge to optimal efficiency and min-max fairness. However, XCP is hard to deploy in today’s Internet as it requires a non-trivial number of bits to encode the rate, bits which are not available in the IP headers. Unlike XCP, VCP decouples efficiency and fairness control algorithms at the sender. VCP routers classify the level of congestion into different regions by computing a load factor, and encode the regions in two ECN bits (already present in the IP header) to return to the sender. The sender runs one of the two algorithms as a function of the congestion level. VCP approximates XCP’s performance in terms of high efficiency and reasonable fairness by leveraging only the two ECN bits to encode the congestion feedback, but it also has sluggish convergence performance.

To sum up, existing protocols have significant limitations in achieving high throughput and reasonable fairness while maintaining fast convergence speed in high BDP networks. To achieve this goal, this paper presents the General Congestion Control Protocol (GCCP). GCCP provides a joint design of congestion control and fairness control and then explicitly communicates the rate to end-hosts to achieve high throughput and reasonable fairness. GCCP allows for aggressive behavior in large underutilized links to achieve high throughput, but leverages only one ECN bit for network utilization feedback. Once the link is sensed to be highly utilized, the protocol dedicates to fair and rapid bandwidth allocation by requiring congestion window increment is conservative and monotone decreasing with congestion window increasing. The ns2 simulations show that GCCP achieves a pretty good tradeoff between high throughput and reasonable fairness while exhibiting fast convergence speed.
Monotone Decreasing-Additive Increase (MD-AI) rule in the high-utilization region. By aggressively increasing $cwnd$ in the low-utilization region, GCCP allows quick and high utilization of available bandwidth. Once high utilization is attained, MD-AI provides fair and rapid bandwidth allocation among competing flows. With MD-AI, the $cwnd$ increment per RTT is as conservative as that of TCP Reno and monotone decreasing with $cwnd$ increasing. Therefore, the flow with smaller $cwnd$ can increase more aggressively than that with larger $cwnd$ in the high-utilization region, and thus flows can rapidly converge to fair bandwidth allocation.

The remainder of this paper is organized as follows. In Section II, we list the requirements for a high-speed protocol, review some related protocols in general. Section III gives the design guidelines and details of the proposed scheme. Section IV presents the ns2 simulations. Section V concludes the paper.

II. RELATED WORK

In this section, we begin by listing some requirements that a high-speed protocol must meet before it can be really deployed into the Internet. Then, we briefly discuss some related work to respond to the requirements and lay the foundation for GCCP protocol.

A. Requirements for a High-speed Protocol

1) Efficiency: The protocol must improve the throughput of the connection to efficiently use the high-speed network link.

2) TCP fairness: The protocol should only make better use of free available bandwidth rather than competing for bandwidth with other standard TCP on the same path.

3) RTT fairness: Network resources should be fairly allocated to the flows running the same protocol when the competing flows have different RTTs.

4) Convergence: Flows with different bandwidth utilization level should quickly achieve their fair bandwidth allocation.

B. HSTCP Protocol

HSTCP is a typical sender side protocol, which uses the principle of AIMD in standard TCP to adjust its $cwnd$. However, both the increasing parameter and the decreasing parameter are functions of the $cwnd$ size rather than constants in TCP Reno. HSTCP updates its $cwnd W$ as follows:

- a new ACK: $W = W + \frac{a(W)}{W}$
- packet loss: $W = W * (1 - b(W))$

Herein, $a(W)$ and $b(W)$ are given by:

$$a(W) = \frac{2 * W^2 * b(W) * p}{2 - b(W)},$$  

$$b(W) = (b_{high} - 0.5) \frac{\log(W) - \log(W_{low})}{\log(W_{high}) - \log(W_{low})} + 0.5,$$

and

$$W = \frac{0.15}{p^{0.32}},$$

where $b_{high}$, $W_{low}$, and $W_{high}$ are parameters used in HSTCP. They are set to 0.1, 31, and 83000 respectively in [3]. In addition, $p$ denotes the packet loss rate. It is clearly shown that $a(W)$ becomes larger and $b(W)$ becomes smaller with $W$ increasing. In this way, HSTCP can sustain a large $cwnd$ and aggressively utilize the bandwidth in high BDP networks. However, the aggressive $cwnd$ behavior of HSTCP induces other deficiencies such as bad TCP friendliness, serious RTT unfairness, and slow convergence speed.

C. VCP Protocol

VCP provides a joint design of end-hosts and routers. VCP routers compute a load factor and use this factor to classify the level of congestion into three regions: low-load, high-load and overload and encode the level of congestion in the two ECN bits. As with ECN, the receiver sends the congestion information to the sender via ACK packets. Based on the load region reported by the network, the sender uses one of the following $cwnd$ policies: Multiplicative Increase (MI) in the low-load region, Additive Increase (AI) in the high-load region, and Multiplicative Decrease (MD) in the overload region. By using MI in the low-load region, flows can exponentially ramp up their bandwidth to achieve high efficiency. Once high utilization is attained, AIMD provides log-term fairness amongst the competing flows. In addition, VCP handles RTT heterogeneity with parameter scaling. The result is that over any time period, the $cwnd$ increase under either MI or AI is independent of the flows' RTTs, and two flows with different RTTs achieve fair rate allocation. However, the AI rule results in sluggishly converging to fairness. Because during AI all flows increase the same amount of $cwnd$ regardless of their current rate, only MD reduces the $cwnd$ difference. Therefore, the fairness can be improved once per AIMD epoch, and thus all flows slowly converge to fair bandwidth allocation.

III. THE GCCP PROTOCOL

A. Design Guidelines

1) Decouple efficiency control and fairness control: The concept that decoupling efficiency control and fairness control was first introduced in XCP. The efficiency control only aims to maximize throughput and link utilization. The fairness control, on the other hand, aims to fairly apportion bandwidth among flows sharing a common link. VCP decouples efficiency and fairness control at sender by switching $cwnd$ policy between MI and AIMD according to network congestion level feedback. To achieve high efficiency and reasonable fairness in high-speed networks, GCCP inherits the characteristic of decoupling control at sender. When network is underutilized, the efficiency control dedicates to high and quick utilization of available bandwidth by aggressively increasing $cwnd$. When network is sensed to be highly utilized, the fairness control dedicates to rapid and fair bandwidth allocation among competing flows by using the MD-AI policy as mentioned before, and thus the flow with smaller $cwnd$ can
increase more aggressively than that with larger cwnd in the fairness control.

2) Use routers’ queue length feedback as the metric reflecting network utilization level: Efficiency control or fairness control is determined according to a given network utilization level. Therefore, the metric reflecting network utilization level should be accurate deployment. Before congestion (packet loss) really happens, the sender judges network utilization level by delay measurement or network information feedback. TCP-Africa detects network utilization level by estimating delay variations [8], but the big problem is that delay cannot be measured accurately and usually the RTT measurement is buried with noise. By contrast, VCP router computes a link load factor which separates network utilization level into three regions, and encodes the regions in two ECN bits which accurately inform sender about network utilization level. In GCCP, the router uses queue length region to classify network utilization level into low utilization level and high utilization level before packet loss, and encodes the utilization level using one ECN bit to return to the sender. The sender determines the controller that is suitable for a given utilization level. Compared to the delay estimation at the sender, the ECN feedback is more accurate. Although one-bit encoding conveys less information than two-bit encoding, we show that GCCP still achieves excellent performances by cwnd algorithms.

B. Detailed Description of the Protocol

GCCP is a combination of sender and router, but mainly a sender side protocol. Each router classifies the network utilization level into two levels depending on the queue length region, and encodes the utilization level in the CE bit of each data packet’s header. In turn, the receiver sends the utilization level to the sender via ACK packets, and the sender updates cwnd accordingly. To proceed, we first introduce the following new variables:

- \( q_{\text{length}} \): the real-time queue length in a router;
- \( CE_{\text{bit}} \): the CE bit in the IP header of a data packet, its initial value is set to 0 at the sender;
- \( buffer_{\text{size}} \): the buffer size of a router;
- \( e(W) \): the cwnd \( W \) increase parameter of GCCP for efficiency control
- \( f(W) \): the cwnd \( W \) increase parameter of GCCP for fairness control

1) The GCCP Router: In GCCP, each router operates DropTail queue management mechanism, which enqueues arriving packets in router buffer for its output links and monitors \( q_{\text{length}} \). When a new packet arrives at a router buffer, \( q_{\text{length}} \) is updated. Define \( \gamma \) as the real-time queue length region threshold, which is set to 1. If \( q_{\text{length}} \) is no less than \( \gamma \), the output links exist in high utilization level. Thus, \( CE_{\text{bit}} \) of the new arriving packet is set to 1; if \( q_{\text{length}} \) is less than \( \gamma \), the output links exist in low utilization level. Thus, \( CE_{\text{bit}} \) of the packet maintains its previous value. This is because high utilization in any router will push the whole network into high utilization. Therefore, if the upriver links exist in high utilization level, the encoding bit in upriver routers should be preserved in this router. Finally, if \( q_{\text{length}} \) exceeds \( buffer_{\text{size}} \), congestion occurs and the new arriving packet is dropped.

2) The GCCP receiver: A GCCP receiver is almost identical to a TCP receiver except that the receiver copies the value of \( CE_{\text{bit}} \) from the arriving data packet to its ACK packet.

3) The GCCP sender: When receiving a new ACK packet, the GCCP sender checks \( CE_{\text{bit}} \). If \( CE_{\text{bit}} \) equals 0, the network signals low utilization. Thus, the sender applies efficiency control to aggressively increase cwnd. If \( CE_{\text{bit}} \) equals 1, the network signals high utilization. Thus, the sender applies fairness control to achieve rapid and fair bandwidth allocation among competing flows. Finally, if the sender detects packet loss through triple duplicate ACKs or overtime of the retransmission timer, it decreases cwnd using HSTCP’s decrease rule.

For the efficiency control, we decide to use the cwnd increase parameter \( a(W) \) of HSTCP:

\[
e(W) = a(W) = \left( 1 - \frac{W_{f}}{W_{c}} \right) e(W)\,.
\]

According to (1), \( e(W) \) is increasing from 0.95755 packets to 74.691 packets when cwnd is increasing from 31 packets to 83000 packets in Fig. 1. Thus, \( e(W) \) is highly aggressive to achieve high throughput. For the fairness control, GCCP performs the following MD-AI cwnd increasing rule:

\[
f(W) = k_{1} - k_{2} * e(W)\,.
\]

We then discuss the choice of parameters \( k_{1} \) and \( k_{2} \). For the fairness control requirement, \( f(W) > 0 \) and \( f(W) < e(W) \) should be satisfied. From \( f(W) > 0 \), we have \( k_{1} > k_{2} * e(W) \), and thus \( k_{1} > 74.691k_{2} \). To avoid heavy congestion and minimize packet loss rate, \( f(W) \) should be small as much as possible. Thus, we set

\[
k_{1} = 80k_{2}\,.
\]

\[\text{Figure 1. Efficiency control parameter } e(W)\text{ and fairness control parameter } f(W) \text{ in GCCP}\]
Meanwhile, we substitute (5) and (6) in the inequality $f(W) < e(W)$ to get

$$k_2 < \frac{e(W)}{80 - e(W)}.$$  

(7)

A basic high speed connection with 1500-byte packets and a 100 ms RTT should achieve a throughput no less than 100Mbps, which would require a $cwnd$ no less than 833.3333 packets, and thus $e(W)$ no less than 7.5435 packets. This result, together with (7), leads to $k_2 < 0.1041$. According to (5) and (6), small value of $k_2$ determines small value of $f(W)$, which can provide good fairness, but slow convergence speed. To rapidly converge to fairness, we set $k_2$ to 0.1 and get $k_1 = 8$ according to (6). Thus, $f(W)$ is decreasing from 7.9042 packets to 0.5309 packets when $cwnd$ is increasing from 31 packets to 83000 packets, as shown in Fig. 1.

From a practical view, $f(W)$ is only a simple modification to $e(W)$, which can satisfy the fairness control requirement and make the fairness control compatible with the efficiency control without causing added complexity when sender switches between the two control policies. The pseudo-code of GCCP is given in Fig. 2.

IV. SIMULATION STUDY

In this section, we conduct ns2 simulations to compare the performances of GCCP with HSTCP and VCP. Since VCP is a representative feedback based protocol, and GCCP inherits its characteristic: combining sender and router and decoupling efficiency control and fairness control at the sender. Meanwhile, HSTCP is the first high-speed protocol, and GCCP use HSTCP’s $cwnd$ increase rule as its efficiency control. Thus, this comparison directly reflects the benefits of GCCP.

The general setup for the following simulations is as follows: the data packet size is 1500bytes, while the ACK packet is 40 bytes. The bottleneck buffer size is set to 20% of the bandwidth-delay product.

A. One Bottleneck

We first conduct simulations with a simple dumbbell topology in Fig. 3. In this figure, $(S_i, R_i)$ represents the source/sink pair of nodes for flow $i$, and $r_i$ represents the router.

1) Convergence and efficiency: In this experiment, four high-speed flows using the same protocol compete the bottleneck link $rl_1r_2$. Two flows start randomly in $[0, 20]$ seconds, and the other two start randomly in $[200, 220]$ seconds. The bottleneck link $rl_1r_2$ has a bandwidth of 2Gbps and a simplex propagation delay of 40 ms. All the other links have a bandwidth of 4Gbps and a simplex propagation delay of 10 ms. Thus, the end-to-end RTT is 100ms. A small amount of UDP traffic is added to the bottleneck link to simulate a very lightly utilized link. The sending rate of the UDP traffic is at 5% of the bottleneck link bandwidth. The Jain’s fairness index [15] of the four flows is measured at each second. The convergence time is defined to be the time from 220 seconds until the fairness index reaches 0.98. This result gives an indication on how fast flows converge to a fair share. Fig. 4 shows the measured fairness index of different protocols from 220 seconds to 1400 seconds. It is clear that the convergence time of HSTCP, VCP, and GCCP are 223 seconds, 830 seconds, and 116 seconds respectively. Meanwhile, GCCP achieves efficiency (goodput and bottleneck link utilization) similar to that of HSTCP and higher than that of VCP, as is shown in Table I.

2) TCP friendliness: One high-speed flow, which adopts HSTCP, VCP and GCCP respectively, shares the bottleneck link $rl_1r_2$ with one TCP Sack flow. The bottleneck link bandwidth is set to 300Mbps, 500Mbps and 1000Mbps respectively, and other link bandwidth is doubled correspondingly. Both the link propagation delay and the sending rate of the UDP traffic are the same as that in section IV.A.1. Table II shows the throughput ratio of high-speed traffic to TCP traffic. Larger throughput ratio means that the high-speed traffic pillages much more bandwidth at the expense of the TCP friendliness, whereas smaller throughput ratio means better TCP friendliness. The TCP friendliness of HSTCP is clearly observed to be the worst. By contrast, VCP achieves the best TCP friendliness. Compared to VCP, GCCP slightly degrades the throughput of TCP Sack, but still achieves TCP friendliness better than that of HSTCP.

3) RTT fairness: Two flows with different RTTs share the bottleneck link $rl_1r_2$. The link $rl_1r_2$ has a bandwidth of 622Mbps and a simplex propagation delay of 5 ms. All the other links have a bandwidth of 1Gbps. The links $Sl_1r_1$ and $r_2R_1$ have a simplex propagation delay of 5 ms. Thus, the flow from $S1$ to $R1$ have a RTT of 30ms. Meanwhile, the propagation delay of the links $S2r_1$ and $r_2R_2$ are set so that

/* The GCCP router algorithm */
if (q_length < q) CE_bit maintains its previous value;
if (q <= q_length <= buffer_size) CE_bit = 1;
if (q_length > buffer_size) packet loss occurs;
/* The GCCP sender algorithm */
/* On a new ACK */
if (CE_bit == 0) // using HSTCP increasing rule
    e(W) = a(W);
    cwnd = cwnd + \frac{\alpha(W)}{cwnd};
}

Figure 2. The pseudo-code of GCCP protocol.
the flow from $S_2$ to $R_2$ have the RTTs of 60ms, 90ms, 120ms, 150ms, and 180ms. Therefore, the RTTs of the two flows have ratio 2, 3, 4, 5, and 6. Table III shows the results of different protocols. HSTCP has most serious RTT unfairness due to its aggressiveness. Since the parameter scaling for VCP is chosen to insure that both the cwnd increase under MI and the sending rate under AI are independent of the flow's RTT, VCP shows the best RTT fairness. GCCP displays RTT fairness much better than that of HSTCP but a little worse than that of VCP, that is, the goodput ratio of two flows with different RTTs is roughly proportional to the inverse of the RTT ratio.

**B. Multiple Bottlenecks**

Next, we study the performance of GCCP with a more complex topology of multiple bottleneck links.

1) **Convergence and Efficiency:** We repeat the foregoing experiment in Section IV. A. 1) with the parking-lot topology in Fig. 5. In this figure, $(S_i, R_i)$ and $r_i$ have the same meaning as that in Fig. 3, and $(X_i, Y_i)$ represents the source/sink pair of nodes for cross traffic $i$. The cross traffic is also the UDP traffic whose sending rate is set to 5% of the bottleneck bandwidth. In this experiment, each individual bottleneck link has 2Gbps bandwidth and 10ms one-way propagation delay, and other links have 4Gbps bandwidth and 5ms one-way propagation delay. Fig. 7 shows the Jain’s fairness index of different protocols from 220 seconds to 1400 seconds. It is clear that the convergence time of HSTCP, VCP, and GCCP are 301 seconds, 889 seconds, and 212 seconds respectively. Meanwhile, Table IV shows that GCCP achieves efficiency similar to that of HSTCP and higher than that of VCP. The experiment results indicate that GCCP obtains good tradeoff between convergence and efficiency.

2) **TCP friendliness:** We also repeat the foregoing experiment in Section IV. A. 2) with the parking-lot topology in Fig. 5. In this experiment, one TCP Sack flow shares the five bottleneck links with one high-speed flow. The throughput ratio of high speed traffic to TCP traffic is shown in Table V. The throughput obtained by the TCP Sack flow is similar when competing with VCP flow and GCCP flow respectively, but the throughput obtained by the TCP flow is the lowest when competing with HSTCP flow.

3) **RTT fairness:** Fig. 6 shows the parking-lot topology 2 used in the following experiment. Two flows with different RTTs share the bottleneck link $r_2r_3$. The links $S_1r_1$ and $r_6r_1$ have varying one-way propagation delay, and all the other links have 5ms one-way propagation delay. Therefore, the RTT of flow2 is fixed at 30 ms, while varying the RTT of flow1 such that the RTTs have ratio 2, 3, 4, 5, and 6. All the bottleneck links have 622Mbps bandwidth. Table VI shows the RTT fairness of GCCP in comparison with that of HSTCP and VCP. VCP still displays the best RTT fairness depending on its parameter scaling. The RTT fairness of HSTCP is observed to be the worst due to its aggressiveness. GCCP displays roughly linear RTT fairness which is a little worse than that of VCP but much better than that of HSTCP.

**V. CONCLUSION**

In this paper, we propose the GCCP protocol, which decouples efficiency control and fairness control while requiring only one bit to encode the network utilization information. The efficiency control achieves high throughput and link utilization by aggressively increasing cwnd. The fairness control achieves good fairness and fast convergence speed by developing a simple MD-AI rule that allows cwnd increment is conservative and decreasing with cwnd increasing. The ns2 simulation results show that GCCP achieves efficient and fair bandwidth allocation while maintaining fast convergence speed, and thus satisfies all the requirements for an ideal high-speed protocol.

As a future research topic, we plan to develop an analytical model of GCCP to analyze its efficiency, fairness and convergence properties.
TABLE IV.  EFFICIENCY

<table>
<thead>
<tr>
<th>Protocol</th>
<th>HSTCP</th>
<th>VCP</th>
<th>GCCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>goodput (Mbps)</td>
<td>1842.2</td>
<td>1710.7</td>
<td>1831.6</td>
</tr>
<tr>
<td>utilization (%)</td>
<td>97.1780</td>
<td>90.6027</td>
<td>96.6506</td>
</tr>
</tbody>
</table>

TABLE V.  TCP FRIENDLINESS

<table>
<thead>
<tr>
<th>Bottleneck Link Bandwidth</th>
<th>Throughput Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HSTCP/TCP</td>
</tr>
<tr>
<td>300 (Mbps)</td>
<td>13.6825</td>
</tr>
<tr>
<td>500 (Mbps)</td>
<td>15.1128</td>
</tr>
<tr>
<td>1000 (Mbps)</td>
<td>17.7368</td>
</tr>
</tbody>
</table>

TABLE VI.  RTT FAIRNESS

<table>
<thead>
<tr>
<th>RTT Ratio</th>
<th>HSTCP</th>
<th>VCP</th>
<th>GCCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:2</td>
<td>12.9383</td>
<td>1.1066</td>
<td>2.7720</td>
</tr>
<tr>
<td>1:3</td>
<td>16.4494</td>
<td>1.3126</td>
<td>3.5027</td>
</tr>
<tr>
<td>1:4</td>
<td>104.7983</td>
<td>1.4641</td>
<td>7.8950</td>
</tr>
<tr>
<td>1:5</td>
<td>190.6711</td>
<td>1.5899</td>
<td>10.6048</td>
</tr>
<tr>
<td>1:6</td>
<td>407.0801</td>
<td>1.7007</td>
<td>9.7167</td>
</tr>
</tbody>
</table>

REFERENCES