2010 24th IEEE International Conference on Advanced Information Networking and Applications

# Performance Analysis of VCP Controller on DCCP Traffic

Hanliu Chen, Oliver W. W. Yang and Yang Hong
CCNR Lab, School of Information Technology and Engineering, University of Ottawa
Email: {yang, yhong}@site.uottawa.ca

***Abstract*—TCP is an important communication protocol as more and more multimedia technologies are deployed in the Internet nowadays. However, existing TCP control mechanism has some shortcomings in supporting real-time streaming traffic. Although UDP can be used as an alternative to transmit real-time application data, it lacks congestion control. Introducing a new TCP-friendly protocol that supports the multimedia applications as well as congestion control has been the aim for many researchers for a long time. The transport layer protocol DCCP was designed to reach this goal recently. Another challenge in high bandwidth-delay product networks is to achieve a fair bandwidth allocation with efficiency, while minimizing packet loss and bottleneck queue. As a simple and low-complexity protocol, VCP can meet this challenge by leveraging the two ECN bits in network congestion feedback. In this paper, we study and implement the VCP controller in a bottleneck network with DCCP traffic. We use VCP to analyze the traffic load into different congestion states in order to effect congestion avoidance and control in the presence of huge DCCP-CCID2 traffic. We perform mathematical analysis and run OPNET simulation to validate our DCCP/VCP controller. We compare the performance of DCCP/VCP with TCP/RED, DCCP/RED and TCP/VCP in a bottleneck network. Our simulation results show that VCP controller can provide DCCP flows with a smooth throughput, a small queue size and a low packet drop rate.**

***Keywords-****Congestion Control, DCCP, TCP, TFRC, RED, VCP.*

## 1. Introduction

Significantly growing multimedia application promotes the rapid development of the Internet, but it also increases the occurrences of congestion in the Internet. Thus, it is necessary to introduce an appropriate congestion control protocol to allocate limited network resources more effectively. TCP (Transmission Control Protocol) has been widely adopted for congestion control due to its simple implementation architecture. However, it introduces an intolerable arbitrary delay for real-time streaming media [1]. For real-time transmission, UDP (User Datagram Protocol) was proposed to mitigate delay jitter effectively without congestion control and packet retransmission. However, this would cause too many packets to be dropped and lost, thus deteriorating the quality of multimedia traffic. To decouple loss recovery from congestion control, DCCP (Datagram Congestion Control Protocol) [2] provides congestion control without packet retransmission for streaming media. It allows a framework to deploy multiple congestion control algorithms. Only TCP-like algorithms and TFRC (TCP-Friendly Rate Control) [3] can be chosen by users currently and these choices are referred by their CCIDs (Congestion Control Identifiers).

Since there is no data retransmission over DCCP, an efficient network operation with fair bandwidth allocation and minimized packet loss becomes an essential requirement. In addition, the MD (Multiplicative Decrease) scheme adopted from TCP may cause large fluctuations in DCCP source throughput. Whenever there are a number of loss events, the congestion window size is halved each time until there is no loss reported by the router. Thus, the source throughput can be suddenly decreased a lot. On the other hand, the window size is increased based on an AI (Additive Increase) scheme. As the most famous AQM (Active Queue Management) algorithm, RED (Random Early Detection) [4] cooperates with a combined AI-MD scheme for the congestion control mechanism in the Internet. RED implicitly informs the network congestion state to the traffic sources by dropping the packets in the bottleneck router. However, its early congestion notification relies on the unnecessary loss of packets [5]. It also wastes network resources, increases timeouts, introduces too much fluctuation on source sending rate and may result in an unhealthy multimedia communication environment.

To address the problems of RED/AQM, an explicit congestion control has been proposed to explicitly feed the congestion state in the bottleneck router back to the sources. Two typical examples are XCP (Explicit Congestion Control Protocol) [6] and API-RCP (Adaptive Proportional Integral Rate Control Protocol) [7, 8]. However, explicit congestion control requires multiple ECN (Explicit Congestion Notification) bits to encode the congestion control information exchanged between routers and end-hosts. Unfortunately, there are only two ECN bits allowed in the IP header currently. Adding more bits into the IP header involves a non-trivial and time-consuming standardization process.

VCP (Variable-structure congestion Control Protocol) [9] is proposed to achieve a high link utilization, a low buffer occupancy, and a negligible packet loss rate by leveraging the only two ECN bits in the IP header. It has been verified

1550-445X/10 $26.00 © 2010 IEEE
DOI 10.1109/AINA.2010.88

[9] that it can achieve comparable good performance to XCP with simple operation. As an XCP-type protocol, VCP is different from the traditional AQM algorithms.

In this paper, we would like to implement the VCP algorithm to control a network with a large number of DCCP-CCID2 sources. DCCP-CCID2 (DCCP Congestion Control Identifier 2) [10] has a TCP-like congestion control, and is suitable for senders who can tolerate abrupt changes in congestion window[1], e.g., Internet telephony. This is particularly useful for senders who would like to take advantage of the available bandwidth in an environment with rapidly changing conditions [10]. Using OPNET simulation, we demonstrate that the RED problem can be solved. The contributions of this paper are: (1) implementation of the DCCP-CCID2 and VCP using an OPNET simulator; (2) mathematical analysis of the equilibrium properties for DCCP/VCP networks, and (3) performance evaluation of the TCP and DCCP AQM algorithms in a bottleneck network.

The rest of the paper is organized as follows. In Section 2, we describe more details about the network operation of DCCP-CCID2 traffic and the VCP controller in our bottleneck network. Section 3 provides the mathematical analysis that focuses on the equilibrium properties for our DCCP/VCP networks. The network configuration, the simulation results and discussions on TCP and DCCP combined with RED and VCP are presented in Section 4. Finally, Section 5 gives conclusion of the paper.

The following notations pertain to the remainder of this paper. paper.

| | |
|---|---|
| $C$ | link capacity in bits/second. |
| $cwnd$ | congestion window. |
| $N$ | number of DCCP flows. |
| $pipe$ | number of outstanding packets in the network. |
| $q$ | queue size in packets. |
| $r$ | flow throughput or source sending rate in bits per second. |
| $R$ | round-trip time in seconds. |
| $R_p$ | RTPD, round-trip propagation delay in seconds. |
| $R_q$ | queuing delay. |
| $s$ | packet size in bits. |
| $ssthresh$ | slow start threshold. |
| $T$ | time delay in the destination node to send out an ACK. |
| $T_{avg}$ | average of $T$. |
| $w_i(t)$ | flow $i$'s congestion window size at time $t$. |
| $\alpha$ | AI parameter set at 1.0. |
| $\gamma$ | link target utilization set at 0.98. |
| $\rho(t)$ | traffic load factor in the router at time t. |
| $\zeta$ | acknowledgement ratio in DCCP-CCID2. |

[1] The abrupt changes are typical of TCP's AIMD (Additive Increase Multiplicative Decrease) congestion control.

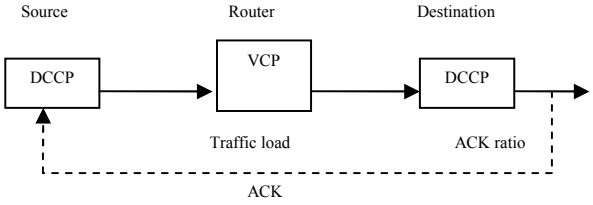


**Figure 1: DCCP/VCP Network Operation**

## 2. Network Model and Operation of DCCP/VCP

Figure 1 depicts the functional blocks in our DCCP/VCP operation. Based on the rules of the slow start phase and the congestion avoidance phase, a DCCP source node sends its data packets through a VCP router to their DCCP destination node. The VCP controller appends control information to the data packets as they pass through the router. The link between each functional block has a capacity of $C$ bits per second, and each packet emerging from the VCP router has a size of $s$ bits. The destination node does not acknowledge each received packet. Instead, it only sends one ACK back to the source node per $\zeta$ received data packets. We called $\zeta$ the acknowledgement ratio, and define it to be the number of data packets received between two successive acknowledgements. The acknowledgement ratio ensures that there is no ACK congestion. Once there is a loss event reported by the destination, there is no retransmission. Note that $\zeta$ cannot be smaller than 2 and cannot exceed $cwnd$ in our operations to insure that there must be an ACK for each $cwnd$ packets.

We shall use the DCCP-CCID2 protocol for each source-destination pair. The three biggest differences between DCCP-CCID2 and TCP are: 1) CCID2 uses acknowledgement ratio $\zeta$ to execute congestion notification on DCCP traffic in the reverse direction. 2) DCCP-CCID2 is an unreliable protocol. It never retransmits data upon a loss or a timeout. 3) As a datagram protocol, the parameters such as $cwnd$ in DCCP-CCID2 are specified in units of packets, not in bytes as in TCP.

| Slow start phase ($cwnd < ssthresh$) | Congestion avoidance phase ($cwnd \geq ssthresh$) |
|---|---|
| ■ the $cwnd$ is increased by one for every received packet<br>■ $pipe$ is decreased also by one for that received packet, it means that the sending rate is doubled per received packet | ■ the $cwnd$ is increased by one when there is no loss in the last $cwnd$ packets<br>■ the sending rate is increased linearly |

**Figure 2: Different Phase in DCCP Source**

The DCCP-CCID2 source sends packets in two phases: the slow start phase and the congestion avoidance phase. A parameter called "$pipe$" is used to control the variable $cwnd$. It is an estimate by the sender on the number of data packets

outstanding in the network. In CCID2, the sender may send a data packet when *pipe* < *cwnd*, but must not send a data packet when *pipe* ≥ *cwnd*. When the *cwnd* is smaller than the slow start threshold (*ssthresh*), the sender operates in the slow start phase. When the *cwnd* is greater or equal to the *ssthresh*, the sender is set as congestion avoidance phase. Although there is no retransmission of data packet, the *cwnd* is halved in case of a loss event. Figure 2 summaries the differences between the above two phases in CCID2.

When a router receives data packets sent by a DCCP source, its VCP controller uses a traffic load factor (instead of packet loss) to control the *cwnd* in the source. It adds an MI (Multiplicative Increase) scheme into its AIMD algorithm. With MI, VCP does not care about fairness when the traffic load is low. Flows can exponentially ramp up their bandwidth to improve the network utilization. When high utilization is attained, the traffic load factor triggers the sender to perform the AIMD operation in order to provide the flows with long-term fairness. VCP uses the traffic load factor to report the traffic load. More mathematical details can be found in the Appendix.

VCP algorithm also uses a parameter called *target link utilization* $\gamma$, which is a constraint to prevent the output traffic rate from reaching the link capacity $C$. The traffic load factor reports that the network is congested when the link utilization reaches the value of $\gamma$. Normally, it is set at 0.98 to avoid buffer overflow and excessive packet loss by preventing aggregated incoming data rate from becoming larger than the link capacity. First, at the moment when the traffic load factor $\rho$ reaches 1, the link utilization is just 98%, which is a little bit lower than 100%. It ensures that there is no buffer overflow. Second, when the router reports the load factor to the source, it has to first encode the load factor into the two ECN bits before sending the packet out of router to the destination. The destination has to take the ECN bits out and encapsulate it into the acknowledgement. Then, it sends the acknowledgement back to the source. Finally, the source executes the MD algorithm. All of these procedures need time, so the target link utilization ensures no heavy traffic congestion and limits the packet loss rate.

For simplicity, we shall use DCCP to mean DCCP-CCID2 for the remainder of the paper.

### 3. Mathematical Analysis on DCCP/VCP

In this section, we perform some mathematical analysis in order to provide more insight into our proposed network operation shown in Figure 1. Our analysis focuses on the fairness and stability properties of a DCCP/VCP network at equilibrium. In networks with a high BDP (Bandwidth-Delay Product), a DCCP source performs the MD algorithm on each loss event. That is, *cwnd* is halved once when a loss event happens. The *cwnd* may suddenly decrease a lot when more loss events happen. This results in more fluctuations in throughput and slow convergence under the high utilization. VCP attempts to mitigate the problem caused by AIMD scheme.

To simplify the analysis, we shall use a continuous-time fluid model to present the DCCP/VCP network. We consider a single bottleneck router with an infinite buffer size shared by $N$ DCCP flows. The sources are indexed by $i$=1,…,$N$. We assume all flows have the same round trip time.

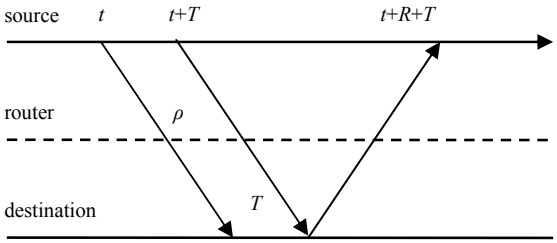


**Figure 3: The DCCP/VCP Model**

Since the congestion information is carried back to the DCCP source by an ACK per every $\zeta$ received packets, we can make the DCCP nodes cooperate with the VCP controller to change the DCCP source *cwnd* via the acknowledgement ratio $\zeta$. Fig. 3 depicts this interaction. $T$ is the time period from the time instant when the first bit of a packet is sent out until the instant when the first bit of the ACK is returned. Obviously, $T$ is a random variable and a function of the ACK ratio $\zeta$.

To conduct our mathematical analysis, we have to find out $T_{avg}$, the average of $T$ as follows. Because of the queuing delay and the service delay of a packet, the packets arrive at the destination node per $s/C$ second. Let $T_j$ be the $T$ for the $j$th packet. $T_1=(\zeta\text{-}1)\cdot s/C$, $T_2=(\zeta\text{-}2)\cdot s/C$, …, $T_{(\zeta\text{-}1)}=1\cdot s/C$, $T_\zeta=0\cdot s/C$. Thus, the average time that an ACK is delayed before sending from the DCCP destination can be obtained as

$$T_{avg} = \frac{\frac{\zeta}{2}\cdot(\zeta-1)\cdot\frac{s}{C}}{\zeta} = \frac{(\zeta-1)\cdot s}{2C}. \quad (1)$$

In addition to the round trip propagation delay $R_p$ and the queuing delay $R_q$, the average amount of time $R'$ that *cwnd* is changed by the DCCP source is

$$R' = R_p + R_q + \frac{s}{C} + T_{avg} = R_p + R_q + \frac{(\zeta+1)\cdot s}{2C}. \quad (2)$$

Let $F(x) = h\cdot\frac{1-x}{x}$, where $h$ is the stability coefficient within the range of $0<h\leq 0.5$. Combining Equation (2) with Equations (A1), (A2) and (A3) in the Appendix and solving, we obtain

$$\dot{w}_i(t) = \frac{1}{R'}\left[F(\rho(t)) \cdot w_i(t) + \alpha\right] = \frac{2C \cdot \left[F(\rho(t)) \cdot w_i(t) + \alpha\right]}{2C \cdot (R_p + R_q) + (\zeta + 1) \cdot s} \quad (3)$$

The traffic load factor $\rho$ can also be obtained as shown in the Appendix. By summing Equation (3) over all $N$ DCCP flows in the router and obtaining the sum of all *cwnd*s, we have

$$\dot{w}(t) = \frac{1}{R'}\left[F(\rho(t)) \cdot \sum_{i=1}^{N} w_i(t) + N \cdot \alpha\right] = \frac{2C \cdot \left[F(\rho(t)) \cdot w(t) + N \cdot \alpha\right]}{2C \cdot (R_p + R_q) + (\zeta + 1) \cdot s}, \quad (4)$$

where $w(t) = \sum_{i=1}^{N} w_i(t)$. Based on Figure 3 and the definition in the Appendix, the load factor $\rho$(t) received by the source at the moment $t+R+T$ can be computed based on the incoming sender's rate $r$ at $t$ as

$$\rho(t) = \frac{r}{\gamma \cdot C} = \frac{\sum_{i=1}^{N} w_i(t) / R'}{\gamma \cdot C} = \frac{w(t) / R'}{\gamma \cdot C} = \frac{2 \cdot w(t)}{\gamma \cdot \left[2C \cdot (R_p + R_q) + (\zeta + 1) \cdot s\right]}. \quad (5)$$

Now, substituting Equations (2), (3), (5) into Equation (4), we obtain

$$\dot{w}(t) = \frac{2C}{2C \cdot (R_p + R_q) + (\zeta + 1) \cdot s} \cdot \left[h \cdot w(t) \cdot \left(\frac{2 \cdot \gamma \cdot C \cdot (R_p + R_q) + (\zeta + 1) \cdot s \cdot \gamma}{2 \cdot w(t)} - 1\right) + N \cdot \alpha\right] \quad (6)$$

From the proof of [9], we know that in a single bottleneck VCP router traversed by a set of DCCP flows with the same round trip time delay $R$, the delayed differential Equation (6) is globally asymptotically stable and has a unique steady state equilibrium $w^*$

$$w^* = \gamma \cdot \frac{2C \cdot (R_p + R_q) + (\zeta + 1) \cdot s}{2} + N \cdot \frac{\alpha}{h}. \quad (7)$$

Let Equation (3) be zero at steady state. Then the *cwnd* of each flow at steady state is simply Equation (7) divided by $N$, i.e.,

$$w_i^* = \frac{2C \cdot \gamma \cdot (R_p + R_q) + (\zeta + 1) \cdot s \cdot \gamma}{2N} + \frac{\alpha}{h}, \; i \in [1, N] \quad (8)$$

By symmetry, the steady source sending rate is the same for all flows and is equal to

$$r_i^* = \frac{w_i^*}{R} = \frac{C \cdot \gamma}{N} + \frac{(\zeta - 1) \cdot s \cdot \gamma}{2N \cdot R} + \frac{\alpha}{h \cdot R}. \quad (9)$$

where $R = R_p + R_q + s/C$. Equation (9) shows that the equilibrium sending rate depends only on the ACK ratio, the packet size, the link capacity, the target link utilization, round trip time and the number of DCCP flows. We also can see that if the number of flows and round trip time increase, the steady flow rate decreases. If the link capacity, ACK ratio, packet size, or target link utilization increase, the equilibrium rate increases. During the MI and AI phases, there is almost no packet loss, the ACK ratio of the DCCP-CCID2 decreases to 2 and stays at this value until the first loss event happens.

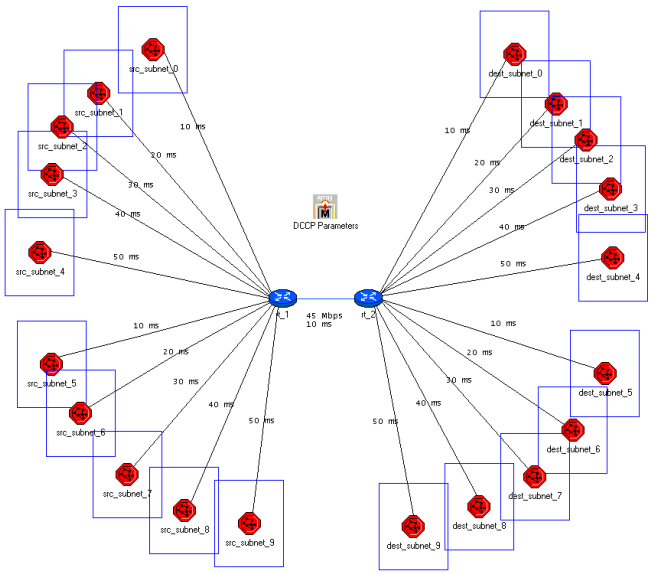


**Figure 4: Network Configuration**

## 4. Performance Evaluation

We have conducted OPNET [11] simulation to evaluate the performance of our congestion controller. Figure 4 depicts the network simulation topology. There are 10 source subnets and 10 corresponding destination subnets labeled from 0 to 9 respectively. Each source (destination) subnet has 20 DCCP senders (receivers). In total there are $N$=200 DCCP flows across the bottleneck router. The bottleneck link capacity is T3 (i.e. $C$=45Mbps). The RTPD (Round Trip Propagation Delay) between sources and destinations is uniformly distributed between 60ms and 220ms. The average queuing delay $R_q$ is estimated to be 20ms approximately. The packet size is $s$=1024 bytes. There are two types of sources: (1) long-lived DCCP source (e.g., video conference), which always tries to send as many data packets as possible, and (2) short-lived source (e.g., telephony or http), which enters the network after a random think time which has a distribution of

$$P\{T > t\} = p_h\left[1 + (\frac{t}{\pi_h})^{\eta_h}\right]^{-1} + p_l\left[1 + (\frac{t}{\pi_l})^{\eta_l}\right]^{-1}.$$

Then it sends a file which has a length distribution of $P\{F > f\} = \left[1 + (\frac{f}{\pi})^{\varphi}\right]^{-1}$,

before waiting for another think period [12]. A short-lived TCP source works as the background noise to interfere the whole network. This mechanism can evaluate the performance under the diversity of real world internet environment [12].

We conduct our simulation on a Pentium IV CPU and 512M memory computer with Windows XP operating system. A typical simulation run takes 120 seconds to achieve the steady state and the full use of the bottleneck utilization.

We make performance comparison among TCP/RED, DCCP/RED, TCP/VCP, and DCCP/VCP under the same network configuration. For the scenarios TCP/RED and DCCP/RED, the maximum packet drop probability is set at 0.1, the minimum buffer queue threshold is set at 150 packets, and the maximum buffer queue threshold is set at 700 packets. The queue filter gain is set to 0.002. For the parameters of VCP, we use the values that are provided in Section 3 and Appendix. That is: $k_q$=0.5, $t_p$=200 ms, $t_q$=10 ms, $\alpha$=1.0, $\sigma_{AI}$=10, $\sigma_{MI}$=2.5, $\beta$=0.875, $\gamma$=0.98, $h$=0.25 and $\mu$=0.0625 [9]. For the ACK ratio $\zeta$ of DCCP, the default value $\zeta=2$ is adopted [2]. Using Equation (9), we can obtain the steady source sending rate of DCCP/VCP source with RTPD=60 ms as $r^*$=220.8 Kbps. The parameters we use for the think time and file length distributions are $P_h$=0.4953916, $P_l$=0.5046084, $\pi_h$=1.0, $\pi_l$=0.0245032, $\eta_h$=1.243437, and $\eta_l$=3.252665. We select $\varphi$=1.15066 as its positive constant with a median value $\pi$=2190.

### 4.1 Throughput

Figure 5 shows the flow throughputs of the source node from the subnet with an RTPD of 60 ms under TCP/RED, DCCP/RED, TCP/VCP and DCCP/VCP, respectively. Source throughput is defined to be data transmission rate (in kilobits per second) from a DCCP source node. The flow throughputs of TCP/RED (Figure 5a) and DCCP/RED (Figure 5b) are fluctuating with a very large amplitude. The fluctuation amplitude of TCP/VCP (Figure 5c) and DCCP/VCP (Figure 5d) are much less except at the beginning of the simulation. This is because all the sources send packets aggressively in the beginning of simulation, which causes many packets to be lost in the bottleneck router buffer. In the TCP and DCCP sources, *cwnd* halves per loss packet so that the throughput decreases immediately to a very low level. After a short period when there is no loss event, the VCP controller starts to affect the *cwnd*. Its MI algorithm increases the *cwnd* to a relatively high value. Then the AI and MD algorithms of the VCP take over and operate alternatively on the control of the source *cwnd*. The disturbance from the short-lived sources has resulted in the small fluctuations in Figure 5c and Figure 5d. One can see that DCCP/VCP source with RTPD=60 *ms* reaches a steady rate at around 220 K*bps* (Figure 5d), which matches well with the theoretical calculation using Equation (9).

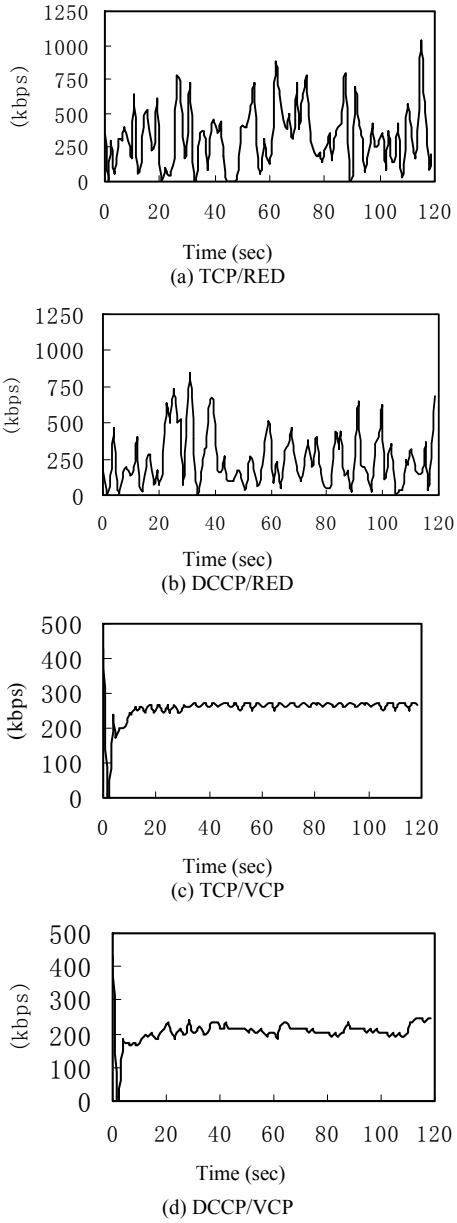


**Figure 5: Flow Throughputs under Different Control Strategies**

### 4.2 Queue Size

Figure 6 shows the average buffer queue size of the bottleneck router under the four strategies. Buffer queue size is defined to be the number of packets in the buffer (as seen by a departure) that are waiting to be serviced by the router. As we can see from Figure 6a, the queue size of RED router with $N$=200 TCP sources is unstable. The average queue size is around 180 packets. The RED router queue size with 200 DCCP sources is shown in Figure 6b. It can be seen that the DCCP/RED queue size fluctuates around its average value of 125 packets, which is lower than that of TCP/RED

queue. This is due to the *pipe* limitation of *cwnd* and the acknowledgement ratio mechanism in DCCP-CCID2. The VCP controller gives the 200 TCP flows a shorter buffer queue size, whose average value is around 105 packets (Figure 6c). Figure 6d shows the buffer queue size of VCP router with 200 DCCP sources is further reduced. The queue length is oscillating and the average value is the smallest (around 80 packets). The larger fluctuations in Figure 6c and Figure 6d are caused by the interaction of the traffic load factor $\rho$ and the short-lived sources. As a whole, VCP controller can maintain a small average queue size.

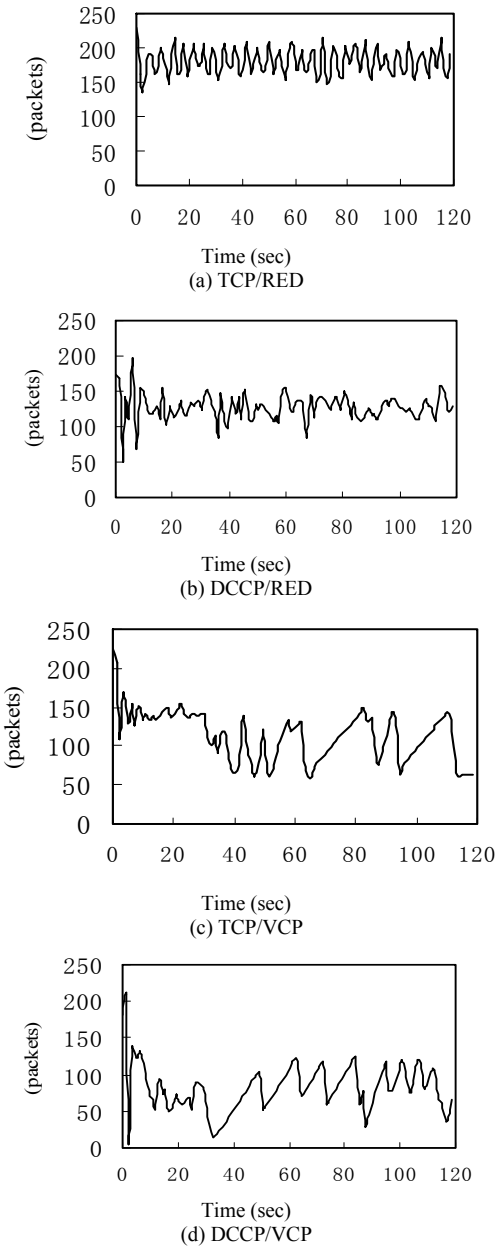


**Figure 6: Queue Size under Different Control Strategies**

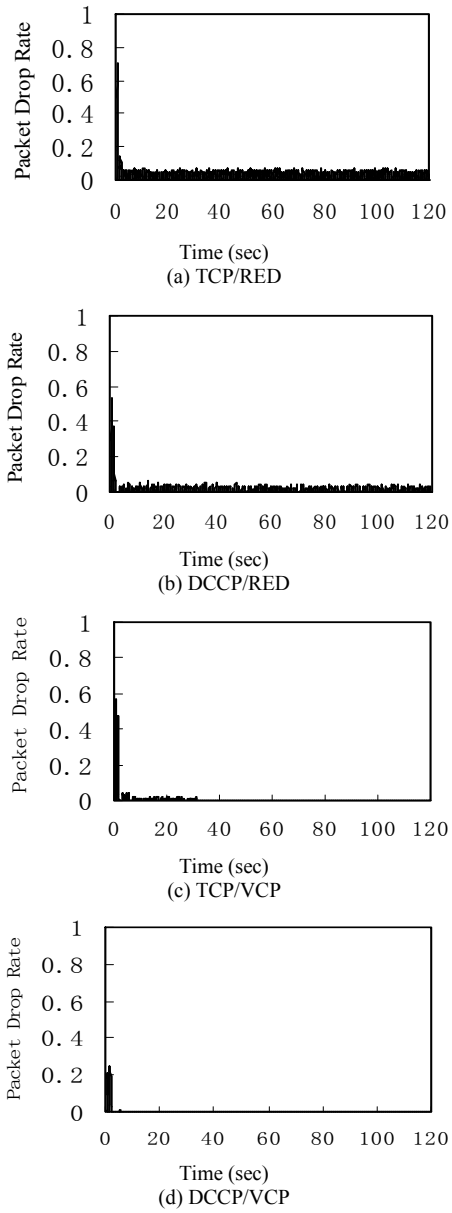


**Figure 7: Packet Drop Rate under Different Control Strategies**

### 4.3 Packet Drop Rate

Figures 7a to7d show the packet drop rate of the bottleneck router in our network model. Packet drop rate is defined to be the average ratio between the number of packets dropped by the router and the number of packets received by the router measured in consequence 10 ms time intervals. It can be seen from Figure 7a that the RED router with 200 TCP sources keeps dropping packets to maintain the traffic un-congested. About 7% of the packets are dropped during the simulated time period. The highest packet drop rate happens at the beginning ($t$=0 s) when all the TCP sources start to send packets to the router, after the senders realize the link

is super congested, the router has already dropped nearly 70% of the TCP packets that cannot be processed. Figure 7b shows the packet drop rate of RED router with 200 DCCP sources. As the first scenario, the RED router always discards 50% of the DCCP packets sent into the router in the beginning of simulation. The average packet drop rate is around 3% which is a little bit lower than that of TCP/RED. DCCP has no data retransmission when a drop event happens and CCID2's TCP-like congestion control algorithm cooperate with the ACK ratio mechanism to limit the number of dropped packets.

From Figures 7c and 7d, it can be seen that the packet drop rate of VCP router with 200 TCP or DCCP sources is only at a similarly high level (around 55%) with that of DCCP/RED in the beginning of simulation. Then the router reports the senders to reduce their packet sending rate. The packet drop rate starts to decrease. At the moment of the $30^{th}$ second (Figure 7c) and $10^{th}$ second (Figure 7d), the packet drop rate falls and stays at 0. This value keeps from the moment till the end of simulation, which proves that under the VCP controller, there is almost no packet dropped by the bottleneck router.

From the simulation results in a bottleneck network, the flow throughput under the control strategy of DCCP/VCP is much smoother than that of RED. The buffer overflow problem in RED is well solved by DCCP/VCP. Thus, there is almost no packet dropped by the VCP router.

### 4.4 Co-existence of VCP and TCP

One can observe that VCP uses a traffic load factor to control the source sending rate, while RED employs packet loss to regulate the source sending rate of TCP traffic. To achieve the congestion control strategy, VCP obtains the traffic load factor based on the link utilization, while RED calculates the packet drop probability based on the average queue size. Due to the different congestion control mechanism, VCP controlled traffic and TCP/RED controlled traffic can not share the same queue in the bottleneck router when both of them co-exist in the same network. Instead, one can set up two sub-queues as proposed in [6, 13]: one for the VCP controlled traffic and the other for the TCP/RED controlled traffic.

## 5. Conclusion

We have implemented the Datagram Congestion Control Protocol in a large-scale bottleneck network using OPNET Modeler. We have also proposed to use VCP controller to control the congested DCCP traffic. We have derived mathematically and verified experimentally the steady flow throughput of DCCP/VCP. By comparing the simulation results under the four different control strategies (TCP/RED, DCCP/RED, TCP/VCP and DCCP/VCP), we have demonstrated that VCP controller gives DCCP flows a smooth throughput. When compared with RED, VCP reduces the average queue size significantly, thus making the bottleneck router to exhibit almost no packet drop and a high link utilization. These features make VCP a suitable candidate to support a healthy, resilient real-time DCCP communication over a bottleneck network.


## Acknowledgment

This work has been partly supported by an NSERC Accelerated Discovery Grant and an Auto21 Grant of Canada.


## References


[1] B. Mukherjee and T. Brecht, "Time-lined TCP for the TCP-friendly delivery of streaming media," *Proceedings of 8th Annual International Conference on Network Protocols (IEEE ICNP)*, Osaka, Japan, November 2000, pp.165-176.

[2] E. Kohler, M. Handley and S. Floyd, "Designing DCCP: Congestion Control without Reliability," *Proceedings of ACM SIGCOMM,* Pisa, Italy, September 2006, pp. 27-38.

[3] M. Handley, S. Floyd, J. Padhye and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," *IETF RFC5348,* September 2008.

[4] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking,* Vol.1, No.4, August 1993, pp.397-413.

[5] W.C. Feng, D.D. Kandlur, D. Saha and K.G. Shin, "The BLUE active queue management algorithms," *IEEE/ACM Transactions on Networking,* Vol.10, No.4, August 2002, pp. 10-23.

[6] D. Katabi, M. Handley and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," *Proceedings of ACM SIGCOMM,* Pittsburgh, USA, August 2002, pp. 89-102.

[7] Y. Hong and O. W. W. Yang, "Design of Adaptive PI Rate Controller for Best-Effort Traffic in the Internet Based on Phase Margin," *IEEE Transactions on Parallel and Distributed Systems*, 18(4), April 2007, pp. 550-561.

[8] Y. Hong and O. W. W. Yang, "An API-RCP Design Using Pole Placement Technique," to appear in *Proceedings of IEEE International Conference on Communications* (*IEEE ICC*), Cape Town, South Africa, May 2010.

[9] Y. Xia, L. Subramanian, I. Stoica and S. Kalyanaraman, "One More Bit is Enough," *IEEE/ACM Transactions on Networking,* 16(6), December 2008, pp.1281-1294.

[10] S. Floyd and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control," *IETF RFC4341*, March 2006.

[11] OPNET Technologies Inc., OPNET Modeler Manuals, OPNET Version 10.5, 2005.

[12] F.J. Ott, T.V. Lakshman and L. Wong, "SRED: stabilized RED," *Proceedings of IEEE INFOCOM,* New York, USA, March 1999, pp. 1346-1355.

[13] Y. Hong and O.W.W. Yang, "Can API-RCP be TCP Friendly with RED?" *Proceedings of 42nd Annual Conference on Information Sciences and Systems* (*IEEE CISS*), Princeton

University, Princeton, NJ, U.S.A., March 2008, pp. 1137-1142.

## Appendix: The VCP Controller [9]

VCP uses the load factor to report the traffic load. In the router, the load factor $\rho$ is calculated by Equation (A1),

$$\rho = \frac{\lambda_l + k_q \cdot q_l}{\gamma \cdot C \cdot t_p}. \quad \text{(A1)}$$

where $\lambda_l$ is the incoming traffic size during each $t_p$ period. It is measured by a packet counter and is reinitialized to 0 when a time period of $t_p$ (200ms) passes. The parameter $k_q$ represents how fast to drain the router steady queue and is set at 0.5. The persistent queue size $q_l$ is measured by Eq. (A2) per $t_q$ (10ms) period. The exponential moving average weight $a$ is set at 0.875. The link target utilization $\gamma$ is set at 0.98.

$$q_l = a \cdot q_l + (1-a) \cdot q(t). \quad \text{(A2)}$$

When the load factor $\rho \in [0, 0.8)$, the traffic load is in a low load region. The router requests the source to set in MI algorithm. When $\rho \in [0.8, 1)$, it is in a high load region. The source is set to run the AI algorithm. When $\rho \in [1, \infty)$, it is in an overload region. The source now performs the MD algorithm. In MI, the flow throughput increases very fast to achieve a bottleneck link utilization at 80%. The *cwnd* is changed according to Eq. (A3), where $\mu$ is the MI parameter 0.0625. The parameter $\sigma_{MI}$ is the MI scaling limiter to bound the burst traffic due to the MI scaling. It is set at 2.5. After the link utilization reaches 80%, the load factor triggers the source to execute the AI and MD algorithms. The *cwnd* changes in AI based on Equation (A4), in which $\alpha$=1, The AI scaling limiter $\sigma_{AI}$ is set at 10 to bound the burst traffic introduced by AI scaling. When using the AI algorithm, the traffic load is becoming higher, and getting close to the maximum threshold 100%. If the traffic load is over 100%, heavy congestion would set in and cause buffer overflow and excessive packet loss. To avoid this situation, once the traffic load reaches 100%, the router requests the source to go to MD algorithm described by Equation (A5), in which $\beta$ is 0.875. MD forces the window size to rapidly decrease to a relatively low value. To avoid the repetitious decreasing of *cwnd*, after doing Equation (A5), the *cwnd* calculation follows AI algorithm for one *R*.

MI:

$$cwnd = cwnd + (1+\mu)^{\min(\frac{R}{t_p}, \sigma_{MI})} - 1. \quad \text{(A3)}$$

AI:

$$cwnd = cwnd + \frac{\alpha \cdot \min\left((\frac{R}{t_p})^2, \sigma_{AI}\right)}{cwnd}. \quad \text{(A4)}$$

MD:

$$cwnd = \max(1, \beta \cdot cwnd). \quad \text{(A5)}$$