# Distributed ECN-Based Congestion Control

Xiaolong Li Homayoun Yousefi'zadeh
Department of EECS
University of California, Irvine
[xiaolonl,hyousefi]@uci.edu

***Abstract*—Following the design philosophy of XCP, VCP is a router-assisted congestion protocol that intends to balance the efficiency and the fairness control in high Bandwidth-Delay Product networks. While both VCP and XCP achieve comparable performance, VCP represents a more practical alternative of deployment as it only requires the use of two ECN bits in the IP header. However, the use of two ECN bits only allows for establishing three levels of congestion notification signaling. Our previous work reveals that VCP suffers from relatively low speed of convergence and exhibits a biased fairness behavior in moderate bandwidth high delay networks due to utilizing an insufficient amount of congestion feedback. In this paper, we propose a distributed ECN-based congestion control protocol to which we refer as Double-Packet Congestion Control Protocol (DPCP). DPCP is capable of relaying a more precise congestion feedback compared to earlier proposed Variable-structure Congestion-control Protocol (VCP) yet preserving the utilization of the two ECN bits. By distributing (extracting) congestion related information into (from) a series of packets, DPCP is able to circumvent the limitations of VCP related to the use of three congestion levels encoded into two ECN bits. We implement DPCP in Linux and demonstrate its performance improvements compared to VCP through experimental studies.**

## I. Introduction

As pointed out by [1], conventional TCP and end-to-end TCP-based Active Queue Management (AQM) schemes such as those proposed by [2], [3], [4], [5], [6] suffer from inefficiency and unfairness in high Bandwidth-Delay Product (BDP) networks. While a wide variety of techniques [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] have been developed to address the problem, all such techniques retain an integrated controller design and thus often fail to achieve both efficiency and fairness. Alternatively, recently proposed eXplicit Congestion-control Protocol (XCP) [17] and Variable-structure Congestion-control Protocol (VCP) [18] take an approach in which the fairness of congestion control is decoupled from its efficiency. Consequently, the protocols can achieve high utilization, low persistent queue length, insignificant packet loss rate, and sound fairness depending on the heterogeneity characteristics of a network.

Essentially, both XCP and VCP are router-assisted protocols. Routers are required to extract and forward congestion related information to end nodes along a transmitting path. While XCP requires routers to explicitly relay the next transmitting rate to transmitting side, VCP only marks one of the three levels of congestion sampled by routers to the transmitting side of a flow. Thus, XCP typically requires the use of a larger number of bits in the IP header of each packet to relay congestion information introducing significant deployment obstacles. In contrast, VCP only uses the two existing ECN bits in the IP header to encapsulate three congestion levels. Given that VCP demands the use of no extra bits in the IP header, it represents a more practical alternative of deployment than XCP.

That said, the operation of VCP is subject to the following shortcomings. First, VCP can only deliver limited feedback to end hosts since two bits can at most represent four levels of congestion. Second, in order to avoid sudden bursts, VCP has to control the growth of transmission rates by setting artificial bounds. The latter, yields slow convergence speed and high transition times. Third, due to the use of fixed parameters for fairness control, VCP exhibits poor fairness characteristics in high delay networks. It is important to note that any feedback-based congestion control protocol must address the tradeoff between the amount of the feedback information and the practicality. It is this tradeoff that provides the motivation for this paper.

In this paper, we propose a new congestion control protocol by extending the design of VCP. Our proposed protocol, Double-Packet Congestion control Protocol (DPCP) can achieve faster convergence and significantly better fairness characteristic than VCP in high BDP networks. In DPCP, finer-grained congestion levels are provided to the transmitting side without requiring the use of extra bits in IP packet header, i.e. DPCP still uses the two ECN bits in the IP header. A congestion level is carried by a chain of two packets and each packet provides two bits out of four bits of information associated with a congestion level. Utilizing a simple and practical design, routers are made responsible for computing and distributing congestion signaling into two packets. At an end node, a congestion level can be retrieved by concatenating a group of two ECN bits together from a set of packets.

Most notably, this paper makes several key contributions. First, we propose a novel approach to overcome the limitations of VCP in high BDP networks by providing more accurate feedback to the sender. While the approach defines a larger number of congestion levels than VCP, it avoids demanding extra bits in packet headers by utilizing a chain of packets to carry congestion levels. Second, we propose a simple and low-overhead yet efficient scheme for distributing and extracting congestion related information into and/or from a chain of packets. Finally, we implement DPCP in both ns-2 [19] and Linux. Through experimental studies, we demonstrate that DPCP is able to make a significant performance improvement compared to VCP in terms of convergence speed and fairness.

This work was sponsored by the grants from Boeing Integrated Defense Systems and UC Discovery Industry-University Cooperative Research Program.

The rest of the paper is organized as follows. In Section II, we review the high level design methodology of VCP along with its limitations. In Section III, we present the fundamentals of the DPCP. Experimental studies are presented in Section IV. Finally, we present several conclusions and discuss future work in Section V.

## II. Background

In this section, we first review the fundamentals of VCP. Then, we show the limitations of VCP in high BDP networks.

### A. Fundamentals of VCP

Fundamentally, VCP inherits the sliding window characteristic of TCP while applying a quite different window management mechanism. In VCP, the $cwnd$ is regulated by different congestion control policies defined according to the level of congestion in the network. Three congestion levels are defined as low-load, high-load, and overload that allow for encoding the level of congestion into two ECN bits in the IP packet header. VCP-capable routers frequently compute the Load Factor (LF) and map it to one of the congestion levels mentioned above. When a data packet arrives, each router examines the congestion level of its upstream link carried in the ECN bits of a packet and updates ECN bits only if its downstream link is more congested. Finally, a receiver receives the congestion level associated with the most congested link. Then, the receiver signals the sender of its session with the congestion information via acknowledgement (ACK) packets. Accordingly, a VCP sender reacts with three congestion control 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. The MI operation is utilized to achieve a better efficiency than that of TCP tied to slow start phenomenon, while the AI and MD operations are used to support the fairness characteristic of TCP.

### B. The Limitations of VCP

We open this section by noting that VCP executes a MIAIMD policy to achieve the so-called max-min fairness [20] characteristic. However, VCP tends to allocate more bandwidth to flows that traverse fewer bottleneck links due to the fact that it implements MIAIMD through the use of a quantized representation of LF instead of its exact value [18]. Furthermore, VCP enforces the MD policy only once with a fixed parameter of 0.875 if an overload is detected. After holding a decreased value of $cwnd$ for a Round Trip Time (RTT), VCP applies the AI policy. However, such a decrease is insufficient when the real value of the LF is more than 115% calculated as the result of a parameter setting of $115 \times 0.875 > 100$. After that, the LF remains in the overload region and a subsequent AI makes the network even more congested. As evidenced by Section IV, VCP exhibits a fairness bias in multi-bottleneck environments, especially in networks with large delays typical of wireless and satellite networks.

Furthermore, VCP applies artificial bounds to its MI and AI parameters in order to avoid sudden bursts. As a result, in moderate bandwidth high delay networks, VCP's bandwidth consumption speed can be very low. As the result of applying MI policy, VCP can even be slower than TCP affected by the slow start phenomenon. Our previous work [21] shows that VCP has higher transition times than XCP and TCP when the average RTT is around 400ms or higher.

While it has been shown that increasing the number of bits used for encoding LF can improve fairness and convergence speed of VCP in such environments [18], such increase will introduce significant deployment obstacles. Therefore, the important question that we raise is whether one can use a larger number of bits to quantize LF without requiring the use of more than two ECN bits in the IP packet header. The answer to this question is the subject of investigation in this paper.

## III. DPCP: Double-Packet Congestion Control Protocol

As an end-to-end congestion control protocol, DPCP is focused on overcoming the limitations of VCP by utilizing more bits in encoding the LF. Rather than demanding more bits in a single packet header, DPCP distributes the bits necessary for encoding the LF into a chain of packets. Each packet uses the two ECN bits in the IP header to carry partial feedback information associated with its chain. By concatenating the set of ECN bits in a packet chain, DPCP allows for signaling end nodes with a more accurate feedback than VCP without demanding more bits in the header of an individual packet.

Although the concept of using more bits for encoding LF is not new [22], [23], DPCP minimizes the overhead and preserves the transparency of deploying VCP and TCP. To that end, DPCP attempts at transparently segmenting and reassembling the header bits used to encode LF without changing the format of the packet. This segmentation and reassembly introduces unique challenges of this work related to out of order arrival of packets in a chain, partial loss of packets in a chain, and backward compatibility to VCP. Specifically, the key concepts of the DPCP can be subdivided into the following:

- *Segmentation and Reassembly (SR) of LF:* As DPCP distributes the LF among a chain of packets, LF needs to be segmented and reassembled at routers and end nodes. To keep backward compatibility, the utilized SR scheme allows for easy exception handling and downgradability to the original VCP.
- *Packet ordering management:* DPCP relies on the feedback distributed in a chain of packets. To retrieve the correct LF, the relative ordering of packets has to be assessed and managed. DPCP provides a simple and efficient mechanism that allows for easily identifying the packet ordering of a chain. Most importantly, there is no need to buffer packets for maintaining the ordering of packets belonging to a chain internally at the routers.
- *Exception handling:* During transmission, exceptions such as packet loss and Out of Order (OO) packet delivery may occur. By detecting the appropriate ordering

TABLE I
DPCP LF DEFINITIONS

| Low Load | | | High Load | | | Over Load | | | |
|---|---|---|---|---|---|---|---|---|---|
| MSP | LSP | LF | MSP | LSP | LF | MSP | LSP | LF | MD Factor |
| 01 | 01 | $< 20\%$ | 10 | 01 | $< 85\%$ | 11 | 01 | $< 105\%$ | 0.875 |
| 01 | 10 | $< 40\%$ | 10 | 10 | $< 95\%$ | 11 | 10 | $< 140\%$ | 0.6 |
| 01 | 11 | $< 80\%$ | 10 | 11 | $< 100\%$ | 11 | 11 | $< 200\%$ | 0.43 |

of packets at the end nodes, DPCP reacts appropriately to exceptions in order to avoid failure. We discuss the impacts of exceptions later in this paper.

### A. DPCP Overview

DPCP employs $4$ bits to encode the LF allowing for defining $16$ congestion levels. In DPCP, the four bits necessary to encode an LF are distributed between two packets transmitted consecutively. We refer the packet carrying the first part of LF as Most Significant Packet (MSP) and the other packet as Least Significant Packet (LSP). The MSP is sent out first. For example, given an LF of $1011$, the MSP carries $10$ in its ECN bits and the LSP carries $11$ in its ECN bits. To keep backward compatibility with TCP, we exclude the combinations containing $00$. Thus, DPCP is left with $9$ combinations that can be used for encoding LF. In contrast to VCP, DPCP defines three congestion zones with three congestion levels in each zone. The boundaries for MIAIMD operations remain the same as those in VCP. Consequently, in low-load and high-load zones, DPCP grows the $cwnd$ using both multiplicative and additive factors as the original MIAIMD model of [20] does. While the original MIAIMD model uses one value of LF per region, each LF represents a range of values in DPCP.

Thus when growing $cwnd$, DPCP conservatively computes increments using the upper bound of an LF. In overload zone, DPCP cuts the $cwnd$ with three factors to guarantee a safe descent to the high-load zone. Table I shows the definitions of LF and MD factors. Notably, the congestion levels are defined for seamless transition between DPCP and VCP.

### B. Packet Ordering Management

DPCP's design introduces an integrated scheme for appropriately managing packet ordering. First, we note that there is no room in the packet header for ordering information. That mentioned, it is important to note that packet ordering information can be captured by a binary value pointing to either MSP or LSP. Exploring the TCP header of a packet, we note that there are two 32-bit numbers, a sequence number ($seq$), and an acknowledgement number ($ack$)[1]. During communications, both numbers can only grow at end nodes and the relative ordering of $seq$ and $ack$ barely changes. Furthermore, modern implementations of TCP make initial $seq$ and $ack$ sufficiently apart from each other. Under typical network operation scenarios, there is a slim chance to change the relative ordering of $seq$ and $ack$ as both number grow. It is this observation that forms the foundation of DPCP.

Specifically, upon the establishment of a TCP connection, the first data packet is treated as the MSP of the first chain of packets. To simplify the operation, DPCP utilizes the relative ordering of $seq$ and $ack$ as an indication of MSP ($seq > ack$) or LSP ($seq < ack$). Once the relative ordering is determined, it will never be changed during transmission. There is a binary flag $MSP$ maintained at end nodes which flips over upon the receipt of each packet. The sender is responsible for signaling the routers MSP or LSP by switching the $seq$ and the $ack$. The operation at the sender is described by the pseudo code below.

**Algorithm 1** Packet Ordering Manager

```
if $MSP$ is TRUE then
    if $seq > ack$ then
        Do nothing
    else if $seq < ack$ then
        Switch $seq$ and $ack$
    else
        $ack - 1$
    end if
else
    if $seq < ack$ then
        Do nothing
    else if $seq > ack$ then
        Switch $seq$ and $ack$
    else
        $seq - 1$
    end if
end if
$MSP \leftarrow \sim MSP$
```

At the receiving side, the same logic is followed for the ACK packet processing. Note that routers do not care for ACK packets and the maintenance of the ordering of $seq$ and $ack$ for ACK packets is only for retrieving the LF at the sender. By simply using the relative ordering of information between $seq$ and $ack$, DPCP is able to distribute LF between two packets without needing to buffer those packets. Furthermore, this mechanism significantly simplifies the routers operation. Since the original ordering of transmission might not reflect at the arrival sequence of packets due to loss or other factors, a router might not know if a packet is the MSP or LSP for a particular chain. Relying on the observation described above, a router simply compares the $seq$ and $ack$, and directly encodes the MSP/LSP of the current LF into the ECN bits of the current packet. The operation described above is neat and simple without introducing any significant overhead. Moreover, the mechanism eliminates the need to keep a mapping between packets and an LF. In the case of facing a tie, i.e., when the $seq$ value is the same as the $ack$ value, the end node simply subtracts $1$ from either the $ack$ or the $seq$ number whichever that is supposed to be smaller. For example, for an MSP, the sender will subtract $1$ from the $ack$ to make $seq > ack$. If the difference between $seq$ and $ack$ is equal to $1$ at the receiving side, the TCP checksum needs to be checked. In

[1]As defined in TCP standard [24], every octet of data sent over a TCP connection has a sequence number. The $seq$ denotes the sequence number of the first data octet in a segment, while the $ack$ contains the value of the next sequence number the sender of the segment is expecting to receive. The initial value of both numbers for a connection are randomly determined when the connection is established. After completing a three-way handshake, both sides of a connection have the initial sequence number of others.

the latter case, an incorrect checksum indicates a tie. Before further processing of the packet, DPCP recovers the original value by adding 1 to $seq$ and/or $ack$ number whichever is smaller. While resolving the tie breaker introduces computing overhead, such overhead is nearly negligible considering the fact that facing a tie situation is extremely unlikely and that the processing only happens at end nodes.

In next subsection, the details of encoding and decoding are presented.

## C. Encoding & Decoding

The encoding happens at both routers and end nodes. For correct encoding, the router needs to keep track of a flag for each flow. That is the only state that needs to be kept at the router. Specifically, the operations associated with encoding and decoding are presented as below. Given a router, assume $MSP1$ and $LSP1$ are associated with the router's downstream link and $MSP2$ and $LSP2$ exist in the header of incoming packets that represent the LF of the router's most congested upstream link.

**Algorithm 2** Encoding

```
if $seq > ack$ then
    if $MSP1 > MSP2$ then
        Mark ECN bits with $MSP1$
        $flag \leftarrow MSP\_LOW$
    else if $MSP1 < MSP2$ then
        $flag \leftarrow MSP\_HIGH$
    else
        $flag \leftarrow MSP\_HIGH$
    end if
else
    if $flag = MSP\_LOW$ then
        Mark ECN bits with $LSP1$
    else if $flag = MSP\_EQ$ then
        if $LSP1 > LSP2$ then
            Mark ECN bits with $LSP1$
        end if
    end if
end if
```

The *complete* decoding operation happens only at end hosts, where *complete* means that intermediate routers can accomplish encoding without knowing the complete value of an LF. Initially, at the sender, the LF is set as $LOW\_LOAD$, i.e. $0101$. Upon the arrival of the first Acknowledge (ACK) packet, the sender immediately starts regulating $cwnd$ without waiting for the LSP. This will cut the response time from two RTTs to one RTT. Once the sender gets a complete LF, DPCP does a finer adjustment based on the new value. Specifically, upon arrival of an MSP, the sender simply replaces the MSP part of the saved LF with the newly arrived MSP. Meanwhile, the sender starts adjusting $cwnd$ conservatively under the assumption that the LF is the highest one in the congestion zone defined by the MSP. For example, if an MSP indicates $HIGH\_LOAD$, then DPCP assumes the complete LF is 1011. In the subsequent operations, whenever an MSP is updated the previous LSP is ignored and will be replaced with 11. Normally, the appearance of MSP and LSP should follow an interleaved pattern. The case for a consecutive MSP and LSP pair will be discussed in the next subsection.

## D. Exception Handling

DPCP relies on the feedback carried by two in-order packets. Thus, DPCP must be able to handle OO transmission and packet loss events. Specifically, DPCP provides mechanisms to respond to the following exceptions:

- Packet Loss & OO transmission: In this case, end nodes will receive consecutive MSPs/LSPs. Rather than attempting to recover the appropriate order, DPCP uses the higher value in MSPs to construct the LF if receiving consecutive MSPs. Otherwise, it ignores arriving LSPs, and uses a pairing of saved MSP and 11 to construct the LF. Generally speaking, receiving consecutive MSPs/LSPs is an indication of congestion. Thus, after receiving three consecutive MSPs/LSPs, DPCP downgrades to the original VCP by simply skipping the packet ordering operation, i.e. no change to the packet ordering is made and then all nodes treat packets as MSP. This behavior allows DPCP to seamlessly convert its behavior to that of the original VCP. After several RTTs, DPCP can resume its normal multi-packet operation. Note that only the sender is involved with the switching of operation between DPCP and the original VCP, while routers and the receiving end are not even aware of it. If OO transmission continues happening, it implies a lossy link, then DPCP will not try to resume a multi-packet operation from the operation of the original VCP. In such situation, a more complicated scheme can be implemented by keeping track of received $seq$s. Due to the limitation of space, we omit a detail discussion of the latter scenario.
- Multipath: While it is possible that packets follow different paths during transmission, it is unlikely that packets are assigned to different paths in an interleaved way. Thus from the perspective of end nodes, the arrival pattern of packets appears to be according to an $OO$ transmission pattern when transmission switches paths. Therefore, DPCP will not be ill-behaved in this case.

## IV. Performance Evaluation

In this section, simulation studies and experimental studies of DPCP are presented. We implement DPCP in both ns-2 simulator and Linux kernel. Performance of DPCP and VCP are compared in terms of efficiency and fairness. Since DPCP is proposed to address the limitations of VCP, our target environment is characterized by moderate bandwidth ($2-10Mbps$) high delay ($200-1000ms$) links.

## A. Simulation Studies

In this subsection, we compare DPCP and VCP in a multi-bottleneck scenario with a typical parking-lot topology consisting of four links. All of the links have a $250ms$ one-way delay and $4Mbps$ bandwidth except the middle link #2 that has

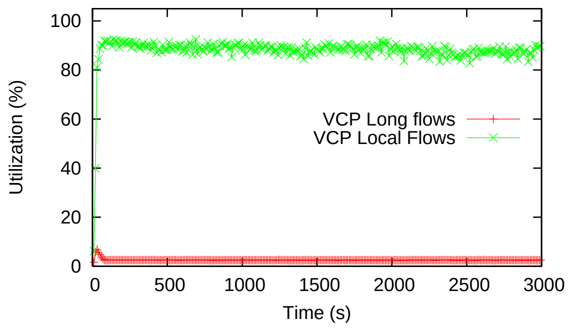

Fig. 1. VCP fails to achieve fairness over bottleneck link #0.

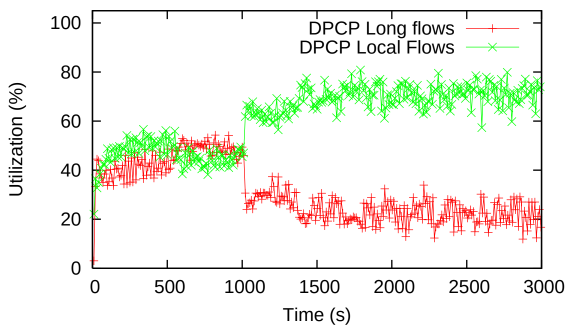

Fig. 2. Bandwidth utilization of DPCP at bottleneck link #0.

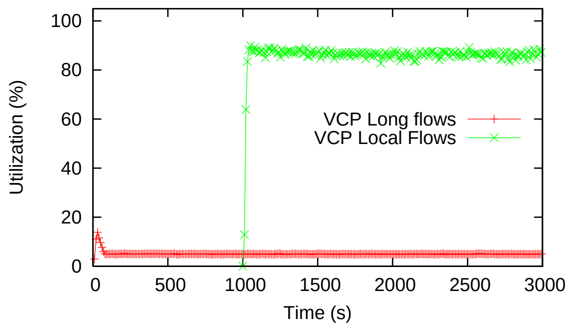

Fig. 3. VCP fails to achieve fairness over bottleneck link #2.

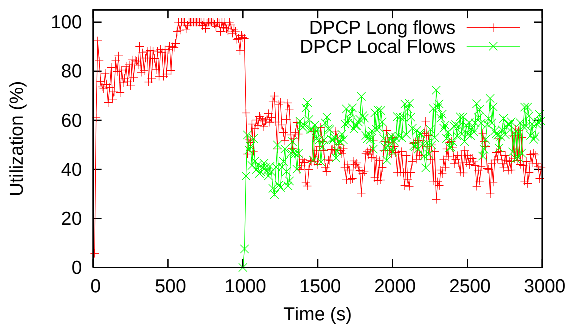

Fig. 4. Bandwidth utilization of DPCP at bottleneck link #2.

a $2Mbps$ bandwidth. There are two types of aggregate FTP flows traversing over the topology. The first type is referred to as a Long Flow and represents the combined traffic of 30 FTP flows traversing all of the links in the forward direction. The second type is referred as to a Local Flow. There are four Local Flows each of which representing 10 FTP flows traversing each individual link in the forward direction. Except those flows that traverse link #2 and start after $1000$ seconds, all other Local Flows start at the beginning of the experiments.

Ideally, during the first 1000 seconds, both Long and Local Flows are to equally split the bandwidth of a shared link. Starting from $1000$-th second when an extra Local Flow starts at link #2, the utilization of Long Flows at Link #0 should drop to 25% while the utilization of Local Flows should go up to 75%. Fig. 1 shows the split of link bandwidth among Local and Long Flows in the case of VCP. In the figure, VCP exhibits a biased fairness characteristic splitting the bandwidth of link #0 with a ratio of $15$ to $1$. In contrast, DPCP demonstrates a significantly better fairness characteristic as shown in Fig 2.

At link #2, we expect to see a near 100% bandwidth utilization for Long Flows during the first 1000 second and a split of 50% in the last 2000 seconds between Long and Local Flow when the Local Flow joins. As illustrated by Fig. 4, DPCP shows good fairness and responsiveness. To the contrary and as shown by Fig. 3, the bandwidth split ratio does not change even when Local Flows are turned on. The latter observation proves that VCP fails to achieve fairness in high BDP multiple bottleneck topologies serving flows with heterogeneous RTTs.

## B. Experimental Studies

In this subsection, we describe our implementation of DPCP in Linux kernel. The implementation approach follows that of VCP as described in [21]. In this section, we do present our experimental study conducted over a real testbed comparing the performance of VCP and DPCP. Due the limitation of space, we only present the results associated with a single bottleneck scenario. We use a dumbbell topology, the settings used for experiments match those of [21]. While not shown in here, the performance of DPCP in multi-bottleneck scenarios follows the pattern shown in our simulation studies.

Fig. 5 compares the bottleneck bandwidth utilization of VCP and DPCP. In contrast to VCP, DPCP converges rapidly and introduces a transition time of less than $4$s compared to $20$s observed in the case of VCP. In addition, DPCP achieves higher bandwidth utilization compared to VCP.

## C. The Effect of Number of Exceptions

As the packet ordering scheme can well handle OO delivery, we measure the effect of packet loss only. Note that packet loss causes OO delivery as well. We install Nistnet network emulator [25] to enforce packet loss. Fig. 6 shows the effects of loss on the Average FTP Completion Time (AFCT) as Packet Loss Rate (PLR) varies. While the performance of both protocols degrades as the number of PLR increases, DPCP consistently outperforms VCP since it achieves faster convergence and higher bandwidth utilization than VCP. Notably, the performance of DPCP is not significantly affected when PLR is less than 30% although DPCP is sensitive to

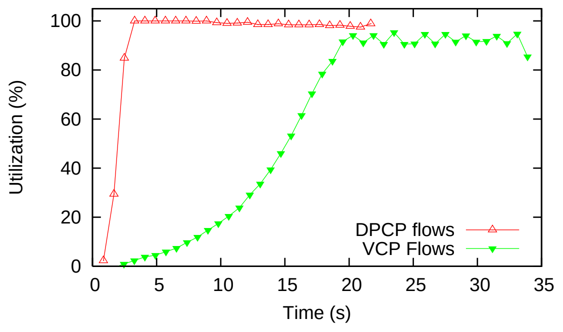

Fig. 5. A Performance Comparison of VCP and DPCP.

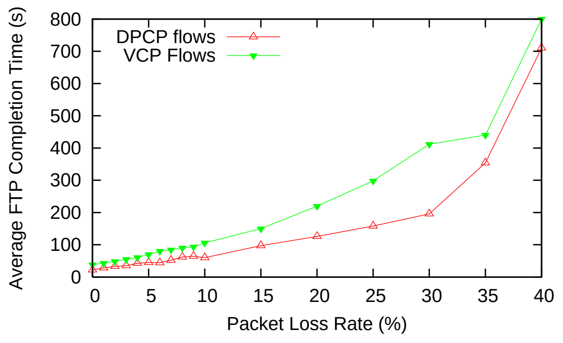

Fig. 6. The effect of packet loss on the performance of VCP and DPCP.

packet loss. Since DPCP will downgrade to VCP when the link is identified as a lossy link, DPCP shows a significant performance degradation after the value of PLR exceeds 35%. The performance gap between DPCP and VCP shrinks as PLR gets larger.

## V. Conclusion

In this paper, we proposed Double-Packet Congestion control Protocol (DPCP) as an extension of Variable-structure Congestion-control Protocol (VCP). We demonstrated how DPCP overcomes the limitations of VCP by providing a more accurate feedback to sender. Rather than demanding extra bits in the header of an individual packet, DPCP used two consecutive packets to carry congestion related information in a distributed manner. Furthermore, we proposed a packet ordering management scheme to enable a simple, low-complexity mechanism of segmentation and reassembly. We also demonstrated how the scheme took advantage of the sequence number and the acknowledgement number as well as the checksum in TCP header to encode ordering information. We implemented DPCP in both ns-2 and Linux kernel. Through both simulation and experimental studies, we demonstrated that DPCP could overcome the limitations of VCP achieving a significant performance improvement in terms of fairness and efficiency in high BDP networks. We are currently working on a multi-packet version of the DPCP that can utilize a chain of up to four packets to carry the exact value of LF.

## References

[1] M. Goutelle, Y. Gu, and E. He, “A Survey of Transport Protocols other than Standard TCP,” in *Data Transport Research Group*, 2004, work in progress, April 2004. https://forge.gridforum.org/forum/forum.php?forum id=410.

[2] V. Jacobson, “Congestion Avoidance and Control,” in *ACM SIGCOMM ’88*, Stanford, CA, Aug. 1988, pp. 314–329.

[3] S. Kunniyur and R. Srikant, “Analysis and design of an adaptive virtual queue (avq) algorithm for active queue management,” in *Proc. of the 2001 ACM SIGCOMM Conference*, 2001.

[4] K. Ramakrishnan, S. Floyd, and D. Black, “The addition of explicit congestion notification (ECN) to IP,” in *IETF RFC 3168*, 2001.

[5] S. Floyd and V. Jacobson, “Random Early Detection gateways for Congestion Avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, no. 9, pp. 397–413, Aug 1993.

[6] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, “REM: Active Queue Management,” *IEEE Network*, vol. 15, no. 3, pp. 48 – 53, May/June 2001.

[7] L. Xu, K. Harfoush, and I. Rhee, “Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks,” in *Proc. of the Infocom 04*, 2004.

[8] I. Rhee and L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” in *Proc. of the PFLDNet’05*, Feb. 2005.

[9] S. Floyd, “HighSpeed TCP for Large Congestion Windows,” Aug. 2002.

[10] D. Leith and R. Shorten, “H-TCP: TCP for High-speed and Long-distance Networks,” in *Proc. of the PFLDNet’04*, Feb. 2004.

[11] T. Kelly, “Scalable TCP: Improving Performance in HighSpeed Wide Area Networks,” Feb. 2003, available at http://wwwlce.eng.cam.ac.uk/ctk21/scalable/.

[12] C. Jin, D. Wei, and S. Low, “FAST TCP: Motivation, Architecture, Algorithms, Performance,” in *Proc. of the Infocom 04*, 2004.

[13] S. Bhandarkar, S. Jain, and A. Reddy, “Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control,” in *Proc. of the PFLDNet’05*, Feb. 2005.

[14] B. Wydrowski and M. Zukerman, “MaxNet: A Congestion Control Architecture for Maxmin Fairness,” *IEEE Comm. Letters*, vol. 6, no. 11, pp. 512 – 514, Nov. 2002.

[15] S. Floyd, M. Allman, A. Jain, and P. Sarolahti, “Quick-Start for TCP and IP,” in *IETF RFC 4782*, Jan. 2007.

[16] K. Xu, Y. Tian, N. Ansari, and S. Member, “Tcp-jersey for wireless ip communications,” *IEEE J. Select. Areas Commun*, vol. 22, pp. 747–756, 2004.

[17] D. Katabi, M. Handley, and C. Rohrs, “Congestion Control for High Bandwidth-Delay Product Networks,” in *Proc. ACM SIGCOMM, 2002*, Aug. 2002.

[18] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, “One More Bit Is Enough,” in *Proc. ACM SIGCOMM, 2005*, Aug. 2005.

[19] -, “UCB/LBNL/VINT Network Simulator - ns (version 2),” available at www.mash.cs.berkeley.edu/ns/.

[20] D. Bertsekas and R. Gallager, *Data networks (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

[21] X. Li and H. Yousefi’zadeh, “An Implementation and Experimental Study of the Vraiable-Structure Congestion Control Protocol (VCP),” in *Proc. of the IEEE MILCOM, 2007*, Oct. 2007.

[22] I. A. Qazi and T. Znati, “On the design of load factor based congestion control protocols for next-generation networks,” in *Proc. of the IEEE INFOCOM 2008*, Apr. 2008.

[23] I. A. Qazi, L. L. H. Andrew, and T. Znati, “Two bigs are enough,” in *Proc. of the ACM SIGCOMM 2008*, Aug. 2008.

[24] J. Postel, “Transmission Control Protocol,” *RFC 793*, Sept. 1981.

[25] M. Carson and D. Santay, “NIST Net-A Linux-based Network Emulation Tool,” *Computer Communication Review*, June 2003.