MPCP: Multi Packet Congestion-control Protocol

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

ABSTRACT

In the recent years, end-to-end feedback-based variants of TCP as well as VCP have emerged as practical alternatives of congestion control by requiring the use of only one or two ECN bits in the IP header. However, all such schemes suffer from a relatively low speed of convergence and exhibit 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 novel distributed ECN-based congestion control protocol to which we refer as Multi Packet Congestion Control Protocol (MPCP). In contrast to other alternatives, MPCP is able to relay a more precise congestion feedback yet preserve the utilization of the two ECN bits. MPCP distributes (extracts) congestion related information into (from) a series of n packets, thus allowing for a $2n$-bit quantization of congestion measures with each packet carrying two of $2n$ bits in its ECN bits. We describe the design, implementation, and performance evaluation of MPCP through both simulations and experimental studies.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Congestion control, TCP, VCP, ECN, Fairness, Efficiency

1. INTRODUCTION

It has been demonstrated in [7] that conventional TCP and end-to-end TCP-based Active Queue Management (AQM) schemes such as [8, 15, 6, 1] suffer from inefficiency and unfairness in high Bandwidth-Delay Product (BDP) networks. In order to address such problems, numerous techniques such as those in [19, 11, 9, 3, 17, 5] have been proposed. However, these techniques often fail to achieve both efficiency and fairness due to their integrated controller design approach. By decoupling the fairness of congestion control from its efficiency, recently proposed eXplicit Congestion-control Protocol (XCP) [10] and Variable-structure Congestion-control Protocol (VCP) [18] can achieve high utilization, low persistent queue length, insignificant packet loss rate, and sound fairness depending on the heterogeneity characteristics of a network.

While both XCP and VCP require routers to extract and forward congestion related information to end nodes along a transmitting path, their operations are quite distinctive. In XCP, routers are required to explicitly relay the next transmitting rate to the transmitting side. As a result, XCP typically requires the use of a larger number of bits in the IP header of each packet to relay congestion information thus introducing significant deployment obstacles. In contrast, VCP only marks one of the three levels of congestion sampled by routers to the transmitting side of a flow, which allows for utilizing the two existing ECN bits in the IP header for indicating three congestion levels. Hence, VCP represents a more practical alternative of deployment than XCP.

By the nature of its design, VCP can only deliver limited feedback to end hosts due to the fact that two bits can at most represent four levels of congestion. While the design of VCP offers more practicality, it falls short in the following aspects. First, VCP has to control the growth of transmission rates by setting artificial bounds to prevent potential bursts yielding slow convergence speed and high transition times. Second, VCP can only use fixed parameters for fairness control producing poor fairness characteristics in high delay networks. In essence, any feedback-based congestion control protocol must address the tradeoff between the amount of feedback information and practicality. It is this tradeoff that provides the motivation of the work reported in this paper.

Following the design philosophy of decoupling the fairness and efficiency of congestion control, this paper proposes a distributed ECN-based congestion control protocol to which we refer as Multi Packet Congestion control Protocol (MPCP). Compared to VCP, MPCP can converge faster and exhibit a significantly better fairness characteristic in high BDP networks. In MPCP, the transmitting side can be fed with finer-grained congestion levels without requiring the use of any extra bits in the IP header beyond the two ECN bits. Specifically, a congestion level is carried by a chain of n packets and each packet provides two bits out of $2n$ bits of information associated with a congestion level. While routers compute and distribute congestion signaling into n packets, end nodes retrieve a congestion level by con-

*This work was sponsored by grants from the Boeing Company and UC Discovery Industry-University Cooperative Research Program. A preliminary version of this work appears in the Proc. of ICC2009 [13].

catenating a group of $2n$ ECN bits together from a set of packets. A noteworthy comment is that while the recent work of UNO [16] may seem to share a similar idea with MPCP, it differs from MPCP in several aspects. First, although UNO passively utilizes existing bits in the IP ID field of a packet header, it introduces deployment issues. For example, UNO will not work in certain encrypted networks where only 6 ToS and 2 ECN packet header bits can pass through encryption boundaries. In contrast, MPCP only requires the use of two ECN bits in each packet. Second, UNO senders need to collect at least 8 specific packets translating to an average of $8 \ln 8 = 24$ consecutively transmitted packets in order to derive the maximum congestion level before regulating *cwnd*, while MPCP senders perform regulations on a per-ACK basis. Over lossy wireless links, consecutive loss of packets associated with the maximum load factor could yield an oscillatory behavior in the case of UNO.

Most notably, this paper makes several key contributions. First, a novel approach is proposed to overcome the limitations of end-to-end feedback based congestion control protocols including VCP in high BDP networks. While the approach allows for relaying a fine-grain quantized congestion information necessary to provide accurate feedback to the sender, it only demands the use of two ECN bits in each packet header by utilizing a chain of packets to carry congestion information. Second, a simple and low-overhead yet efficient scheme is designed for distributing and extracting congestion related information into and from a chain of packets. Finally, we implement MPCP in both NS2 and Linux kernel. Through experimental studies, we demonstrate that MPCP is able to make significant performance improvements compared to VCP in terms of convergence speed and fairness.

The rest of the paper is organized as follows. In Section 2, we review the high level design methodology of VCP along with its limitations. In Section 3, we present the fundamentals of our protocol design by focusing on the double packet version of our protocol to which we refer as Double Packet Congestion-control Protocol (DPCP). Experimental studies are presented in Section 4. In Section 5, we present the extension of DPCP to a multi packet protocol version. Finally, several conclusions are presented in Section 6.

2. BACKGROUND

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

2.1 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 can be encoded as the level of congestion into two ECN bits of the IP packet header. VCP capable routers frequently compute the so-called 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.

2.2 Limitations of VCP

We open this section by noting that VCP executes a MIAIMD policy to achieve the so-called max-min fairness [2] characteristic. As the result of implementing MIAIMD through the use of a quantized representation of LF instead of its exact value, VCP tends to allocate more bandwidth to flows that traverse fewer bottleneck links [18]. Additionally, 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 exceeds 115% (resulted by a parameter setting of $115\% \times 0.875 > 100\%$). After that, the LF remains in the over-load region and a subsequent AI makes the network even more congested. As evidenced by Section 4, 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, VCP's bandwidth consumption speed can be very low in moderate bandwidth high delay networks.

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 in this paper 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.

3. DPCP: DOUBLE PACKET CONGESTION CONTROL PROTOCOL

We start our discussion by describing the design of DPCP, the two packet, i.e. $n = 2$, version of MPCP. DPCP focuses 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 two 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.

Although the concept of using more bits for encoding LF is not new [14, 16], 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 re-

Table 1: LF and MD Specifications of DPCP

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

assembly 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 divided into the following categories: *i*) *Segmentation and Reassembly (SAR) 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 SAR scheme allows for easy exception handling and downgradability to the original VCP. *ii*) *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 (PO) 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. *iii*)*Exception handling:* During transmission, exceptions such as packet loss and Out of Order (OO) packet delivery may occur. By detecting the appropriate ordering 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.

3.1 DPCP Overview

DPCP utilizes 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 the Most Significant Packet (MSP) and the other packet as the 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 in both MSP and LSP. 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 [2] 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 1 shows the specifications of LF and MD factors for DPCP.

3.2 Packet Ordering Management

DPCP's design introduces an integrated scheme for appropriately managing PO. First, we note that there is no room in the packet header for ordering information. That mentioned, it is important to note that PO 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 numbers 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 packet chain. 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 ←∼ MSP
```

At the receiving side, the same logic is followed for the ACK packet processing. Note that routers do not utilize the information in the 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 the 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 arriving 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

[1]As defined in TCP standard, 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 values 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.

packet. The simple approach described above does not introduce any significant overhead. Moreover, the mechanism eliminates the need for keeping 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 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.

3.3 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 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 ← MSP_LOW
  else if MSP1 < MSP2 then
    flag ← MSP_HIGH
  else
    flag ← 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 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 steps, the previous LSP is ignored and replaced with 11 whenever an MSP is updated.

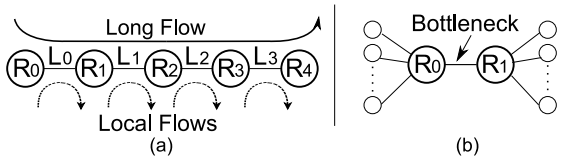

Figure 1: An illustration of (a) parking lot and (b) dumbbell topologies used in our experiments.

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.

3.4 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: i)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 PO operation, i.e., no change to the PO 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 switch back to its multi packet operation mode from the VCP operation mode. In such situation, a more complicated scheme can be implemented by keeping track of received *seq* numbers. Due to the limitation of space, we omit a detail discussion of the latter scenario. ii)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.

4. PERFORMANCE EVALUATION

In this section, simulation studies and experimental studies of DPCP are presented. We implement DPCP in both NS2 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 topologies including moderate bandwidth ($2 - 10Mbps$) high delay ($200 - 1000ms$) links.

4.1 Simulation Studies

In this subsection, we compare the performance of DPCP and VCP over a four bottleneck parking lot topology as il-

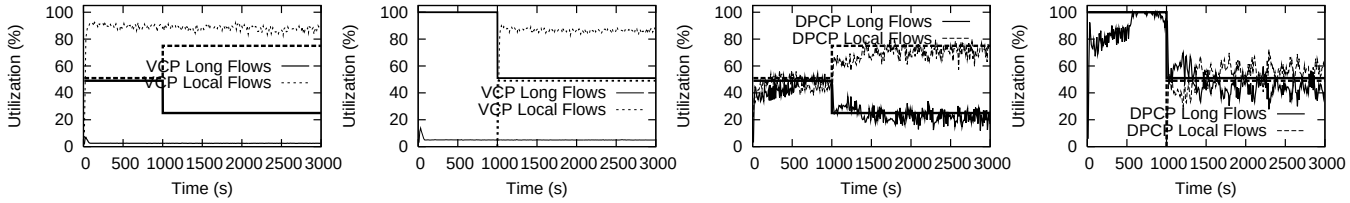

Figure 2: A performance illustration for (a) VCP over link L_0, (b) VCP over link L_2, (c) DPCP over link L_0, and (d) DPCP over link L_2.

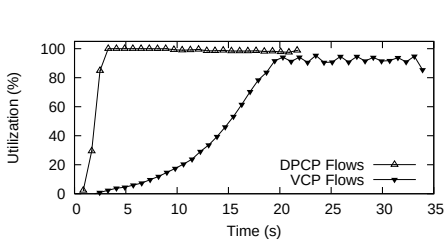

Figure 3: A Performance Comparison of VCP and DPCP.

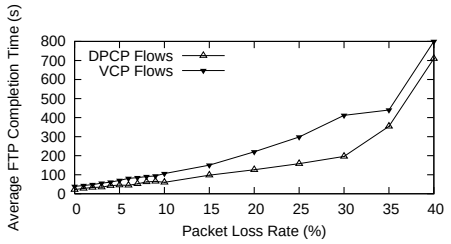

Figure 4: The effect of packet loss on the performance of VCP and DPCP.

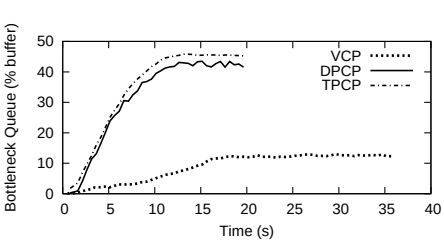

Figure 5: Queue dynamics of VCP, DPCP, and MPCP.

lustrated by Figure 1(a). All of the links have a $250ms$ one-way delay and $4Mbps$ bandwidth except L_2 that has a bandwidth of $2Mbps$. There are two types of aggregate FTP flows traversing 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 left-to-right direction. The second type is referred as to as 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 L_2 and start after 1000 seconds, all other Local Flows start at the beginning of the experiments.

Ideally speaking, both Long and Local Flows are to equally split the bandwidth of a shared link during the first 1000 seconds. Starting from the 1000-th second when an extra Local Flow starts utilizing L_2, the utilization of Long Flows at L_0 is expected to decrease to 25% while the utilization of Local Flows is expected to increase to 75%. Figure 2(a) 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 L_0 with a ratio of 15 to 1. In contrast, DPCP demonstrates a significantly better fairness characteristic as shown in Fig 2(c).

We also expect to see a nearly 100% bandwidth utilization of L_2 for Long Flows during the first 1000 seconds and a split of 50% in the last 2000 seconds between the Long and Local Flow after the Local Flow starts. As illustrated by Figure 2(d), DPCP shows good fairness and responsiveness. To the contrary and as shown by Figure 2(b) for VCP, 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.

While not shown here due to shortage of space, we also evaluate the performance of DPCP in various scenarios similar to those presented in [18]. In all experiments, DPCP achieves slightly better bandwidth utilizations in steady status and significant better convergence speeds than VCP.

4.2 Experimental Studies

In this section, we present our experimental study conducted over a wireless emulation Linux testbed comparing the performance of VCP and DPCP. Due the limitation of space, we cannot describe the details of our Linux kernel implementation. We refer the interested reader to our work of [12] for the details of our implementation. Further, we can only present the results associated with a single bottleneck scenario. We use a dumbbell topology as illustrated by Figure 1(b). Figure 3 compares the bottleneck bandwidth utilization of VCP and DPCP. In contrast to VCP, DPCP converges rapidly and introduces a transition time of less than $4s$ compared to $20s$ observed in the case of VCP. In addition, DPCP achieves a higher bandwidth utilization compared to VCP. While not shown here, the performance of DPCP in multi-bottleneck scenarios follows the same patterns as those shown in our simulation studies.

4.3 The Effect of the Number of Exceptions

As our proposed PO scheme can well handle OO delivery, we only measure the effect of packet loss only. Note that packet loss causes OO delivery as well. We install Nistnet network emulator [4] to enforce packet loss. Figure 4 shows the effects of loss on the average FTP completion time as a function of packet loss rate. 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 for packet loss rates less than 30% although DPCP is sensitive to packet loss. Since DPCP will downgrade to VCP when the link is identified as a lossy link, DPCP shows a significant performance degradation when packet loss rates exceed 35%. The performance gap between DPCP and VCP shrinks as packet loss rates grow larger.

5. MPCP BEYOND DPCP

The design of MPCP can be easily expanded to cover packet chains of more than two packets. Beyond DPCP, we implement and evaluate a triple packet version of MPCP. We refer to the triple packet version of MPCP as Triple Packet Congestion-control Protocol (TPCP). Simply put, the PO management of TPCP can be done by using the checksum approach. In the case of TPCP, a binary indicator of PO, i.e., the relative order of *seq* and *ack* is not sufficient for determining the ordering of a chain of packets. Thus, by directly subtracting the order number from the *ack* or *seq*, MPCP is able to retrieve the ordering number from the difference between the calculated TCP checksum and the one in the packet. This approach allows for using an arbitrary number of packets to encapsulate congestion information.

We evaluate the performance of MPCP utilizing the testbed of Section 4. Figure 5 compares the bottleneck queue dynamics of VCP, DPCP, and TPCP. While VCP yields the longest transition time due to conservatively growing the transmitting rate, VCP achieves the lowest persistent bottleneck queue length. To the contrary, both DPCP and TPCP consume three times as much buffer space of the bottleneck link as VCP does. In exchange, both DPCP and MPCP converge faster and achieve a better fairness than VCP. The oscillatory pattern of DPCP demonstrates that while the amount of feedback to sender has been increased nearly by a factor of three, the reaction to congestion remains at a relatively coarse level. In the case of TPCP, a smoother pattern is observed while the queue length is slightly higher than DPCP. The latter is due to the fact that the sender can obtain an even finer-grained feedback. While not shown in the paper due to shortage of space, we have observed that DPCP and TPCP achieve comparable bandwidth utilization and convergence speed in all our experiments[2]. Considering the performance gains and the overhead associated with the TPCP, we believe DPCP represents a better alternative of MPCP design and implementation. Notably, the PO management of MPCP may introduce compatibility issues in the context of IPSec, NAT, and IDS or DPI. While for IPsec an alternative PO scheme could differentiate between MSP and LSP relying on the LSB bit of the IP ID field of the IP header, in-order delivery of ECN bits crossing encryption boundaries is not trivial. For NAT, IDS, and DPI, MPCP must be aware of their existence and integrate necessary functionality in its router module.

6. CONCLUSION

In this paper, we discussed the design, implementation, and performance evaluation of Multi Packet Congestion control Protocol (MPCP). We demonstrated how MPCP used multiple consecutive packets to carry congestion information in a distributed manner without demanding the use of more than two ECN bits in the header of individual packets. Moreover, a low-complexity packet ordering management scheme was proposed to deal with segmentation and reassembly. We discussed the double and triple packet versions of MPCP and also demonstrated how our design could be expanded to an arbitrary number of packets by manipulating the sequence number, the acknowledgement number, and the checksum in TCP header. We implemented both double packet and triple packet versions of MPCP in both NS2 and the Linux kernel. Through both simulations and experimental studies, we demonstrated that MPCP could overcome the limitations of end-to-end feedback-based congestion control protocols thereby achieving a significant fairness and efficiency improvements, specially in high BDP networks. Moreover, we compared the performance of the double packet version of MPCP with its triple packet version.

[2]While this approach requires all nodes along the transmission path to compute the TCP checksum, the overhead can be easily alleviated relying on TCP Checksum Offload (TCO) feature supported in hardware by many of today's Network Interface Cards. In terms of our protocol design, the required slight change can be easily inserted to the TCO code.

7. REFERENCES

[1] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: Active Queue Management. *IEEE Network*, 15(3):48 – 53, May/June 2001.

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

[3] 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.

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

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

[6] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(9):397–413, Aug 1993.

[7] 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.

[8] V. Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM '88*, Stanford, CA, Aug. 1988.

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

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

[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] 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.

[13] X. Li and H. Yousefi'zadeh. Distributed ECN-Based Congestion Control. In *Proc. of the IEEE ICC*, June 2009.

[14] 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.

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

[16] N. Vasic, S. Kuntimaddi, and D. Kostic. One Bit Is Enough: a Framework for Deploying Explicit Feedback Congestion Control Protocols. In *Proc. of the First COMSNETS*, Jan. 2009.

[17] B. Wydrowski and M. Zukerman. MaxNet: A Congestion Control Architecture for Maxmin Fairness. *IEEE Comm. Letters*, 6(11):512 – 514, Nov. 2002.

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

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