On Tuning TCP for Superior Performance on High Speed Path Scenarios

Kazumi Kumazoe, Hirofumi Ishizaki, Takeshi Ikenaga, Dirceu Cavendish, Masato Tsuru, Yuji Oie
Department of Computer Science and Electronics
Kyushu Institute of Technology
Fukuoka, Japan 810-0004
Email: {zaki,ike}@ecs.kyutech.ac.jp, {kuma,cavendish,tsuru,oie}@ndrc.kyutech.ac.jp

***Abstract*—Transmission control protocol performance varies considerably, depending on network and path conditions. In this paper, we discuss path conditions that affect TCP performance, from round trip delays to path capacity and buffering. We characterize throughput performance of popular TCP congestion avoidance mechanism as well as recently proposed TCP variants via open source based network experiments. We show that superior TCP performance may be achieved via careful selection of congestion avoidance mechanism, as well as parameter tuning.**

***Keywords*—high speed networks; TCP congestion avoidance; Packet retransmissions; Path capacity and buffering;**

I. INTRODUCTION

Transmission control protocol (TCP) is the dominant transport protocol of the Internet, providing reliable data transmission for the large majority of applications. User experience depends heavily on TCP performance. In the last decade, many TCP variants have been proposed, mainly motivated by performance reasons. As TCP performance depends on network characteristics, and the Internet keeps evolving, TCP variants are likely to continue being proposed. Most of the proposals deal with congestion window size adjustment mechanism, the so called congestion avoidance phase of TCP.

In prior works, we have introduced a delay based TCP window flow control mechanism that uses path capacity and storage estimation [6], [7]. The idea is to estimate bottleneck capacity and path storage space, and regulate the congestion window size using a control theoretical approach. Two versions of this mechanism were proposed: one using a proportional controlling equation [6], and another using a proportional plus derivative controller [7].

In this work, we study TCP performance of most popular TCP variants - Reno [3], Cubic (Linux) [11], Compound TCP (Windows) [12] - as well as our most recently proposed TCP variants: Capacity and Congestion Probing (CCP) [6], and Capacity Congestion Plus Derivative (CCPD) [7], under various path conditions. Our contributions are as follows. We show that most used TCP variants of today perform differently over various network scenarios. In addition, for our TCP variants, we tune their performance according to network scenarios for superior performance. Our results show that there is no single TCP variant that is able to best perform under all network scenarios. For our protocols, we investigate best protocol parameters to deliver superior performance. For other protocols, our results can be seen as a call for protocol tuning. The material is organized as follows. Related work discussion is provided on Section II. Section III introduces the TCP variants addressed in this paper, their features and differences. Section IV addresses their performance evaluation. Section VI addresses directions we are pursuing as follow up to this work.

II. RELATED WORK

Research studies of TCP performance on various network environments abound. Many of these studies, have focused on mobile wireless networks [5], [9], [13], as loss based congestion avoidance has the issue of not being able to differentiate between random packet loss and buffer overflow packet loss [4]. [5] studies throughput performance of TCP variants for various Packet Error Rates (PERs) on a mobile network via simulations. [9] also studies TCP variants performance under various PERs, but it also investigates the impact of routing protocols on TCP performance. Wireless network scenarios typically involve a low speed bottleneck link capacity, which limits the size of the congestion window to small values, masking the buffer overflow problem on routers.

On wired high speed networks, [8] has conducted a study of the impact of buffer size, packet error rate, and network delay on throughput performance of NewReno, BIC, Cubic, Highspeed, and Compound TCP variants under large bandwidth delay product and high capacity bottlenecks, via simulations. Although our work has similarities with theirs, we evaluate unique aspects of TCP such as throughput recovery upon cross traffic via open source experiments rather than simulations.

III. TRANSMISSION CONTROL PROTOCOL FRAMEWORK

TCP protocols fall into two categories, delay and loss based. Advanced loss based TCP protocols use packet loss as primary congestion indication signal, performing window regulation as $cwnd_k = f(cwnd_{k-1})$, being ack reception paced. Most f functions follow an Additive Increase Multiplicative Decrease strategy, with various increase and decrease parameters. TCP NewReno and Cubic are examples of AIMD strategies. In contrast, delay based TCP protocols use queue delay information as the congestion indication signal, increasing/decreasing the window if the delay is small/large, respectively. CCP and CCPD are examples of delay based protocols.

 ISBN: 978-1-61208-204-2

Most TCP variants follow a framework composed of few phases: slow start, congestion avoidance, fast retransmit, and fast recovery.

- **Slow Start(SS) :** This is the initial phase of a TCP session, where no information about the session path is assumed. In this phase, for each acknowledgement received, two more packets are allowed into the network. Hence, congestion window $cwnd$ is roughly doubled at each round trip time. Notice that the $cwnd$ size can only increase in this phase. In this paper, all TCP variants make use of the same slow start except Cubic [11].
- **Congestion Avoidance(CA) :** This phase is entered when the TCP sender detects a packet loss, or the $cwnd$ size reaches a target upper size called $ssthresh$ (slow start threshold). The sender understands that the $cwnd$ size needs to be controlled to avoid path congestion. Each TCP variant has a different method of $cwnd$ size adjustment.
- **Fast Retransmit and fast recovery(FR) :** The purpose of this phase is to freeze all $cwnd$ size adjustments in order to take care of retransmissions of lost packets.

Figure 1 illustrates various phases of a TCP session. A comprehensive tutorial of the evolution of TCP features can be found in [2].

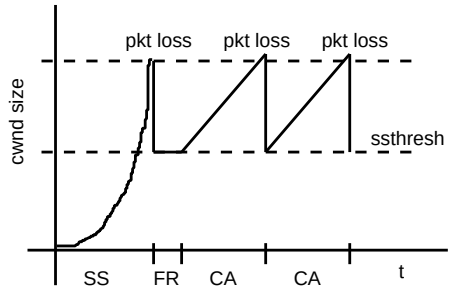

Fig. 1: TCP Congestion Window Dynamics

A. Reno TCP

Reno is a loss based TCP, and may be considered the oldest implementation of TCP to achieve widespread usage. Its congestion avoidance scheme relies on increasing the $cwnd$ by $1/cwnd$ increments, and cutting it in half on packet loss detection, as per equation 1.

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k} \\ PktLoss: \quad cwnd_{k+1} &= \frac{cwnd_k}{2} \end{aligned} \tag{1}$$

Notice that for large $cwnd$ values, the increment becomes small. So, for large bandwidth delay product paths, Reno $cwnd$ ramps up very slowly. A new version of Reno, TCP NewReno introduces an optimization of the Fast Recovery mechanism, but its congestion avoidance scheme remains the same.

B. Cubic TCP

TCP Cubic is a loss based TCP that has achieved widespread usage due to being the default TCP of the Linux operating system. Its congestion window adjustment scheme is:

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= C(t-K)^3 + Wmax \\ K &= (Wmax\frac{\beta}{C})^{1/3} \\ PktLoss: \quad cwnd_{k+1} &= \beta cwnd_k \\ Wmax &= cwnd_k \end{aligned} \tag{2}$$

where C is a scaling factor, $Wmax$ is the $cwnd$ value at time of packet loss detection, and t is the elapsed time since the last packet loss detection ($cwnd$ reduction). Although the equations look complicated, the rational is simple. Cubic remembers the $cwnd$ value at time of packet loss detection - $Wmax$, when a sharp $cwnd$ reduction is enacted, tuned by parameter β. After that, $cwnd$ is increased according to a cubic function, whose speed of increase is dictated by two factors: i) how long it has been since the previous packet loss detection, the longer the faster ramp up; ii) how large the $cwnd$ size was at time of packet loss detection, the smaller the faster ramp up. The shape of Cubic $cwnd$ dynamics is typically distinctive, clearly showing its cubic nature. Notice that upon random loss, Cubic strives to return $cwnd$ to the value it had prior to loss detection quickly, for small $cwnd$ sizes.

C. Compound TCP

Compound TCP is the TCP of choice for most Windows machines. It implements a hybrid loss/delay based congestion avoidance scheme, by adding a delay congestion window $dwnd$ to the congestion window of NewReno [12]. Compound TCP $cwnd$ adjustment is as per Equation 3:

$$\begin{aligned} AckRec: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k + dwnd_k} \\ PktLoss: \quad cwnd_{k+1} &= cwnd_k + \frac{1}{cwnd_k} \end{aligned} \tag{3}$$

where the delay component is computed as:

$$\begin{aligned} AckRec: dwnd_{k+1} &= dwnd_k + \alpha dwnd_k^K - 1, \text{if } diff < \gamma \\ & \quad dwnd_k - \eta diff, \quad \text{if } diff \geq \gamma \\ PktLoss: dwnd_{k+1} &= dwnd_k(1-\beta) - \frac{cwnd_k}{2} \end{aligned} \tag{4}$$

where α, β, η and K parameters are chosen as a tradeoff between responsiveness, smoothness, and scalability. $diff$ is defined as the difference between an expected throughput and the actual throughput, as $diff = cwnd/minRtt - cwnd/srtt$, $minRtt$ is the minimum rtt experienced by the TCP session, and $srtt$ is a smooth round trip delay computation.

D. Capacity and Congestion Probing TCP

TCP CCP is our first attempt to design a delay based congestion avoidance scheme based on solid control theoretical approach. The $cwnd$ size is adjusted according to a proportional controller control law. The $cwnd$ adjustment scheme is called at every acknowledgement reception, and may result in either window increase and decrease. In addition, packet loss does not trigger any special $cwnd$ adjustment. CCP $cwnd$ adjustment scheme is as per Equation 5:

 ISBN: 978-1-61208-204-2

$$cwnd_k = \frac{[Kp(B - x_k) - in\_flight\_segs_k]}{2} \quad 0 \leq Kp \qquad (5)$$

where Kp is a proportional gain, B is an estimated storage capacity of the TCP session path, or virtual buffer size, x_k is the level of occupancy of the virtual buffer, or estimated packet backlog, and in_flight_segs is the number of segments in flight (unacknowledged). Typically, CCP $cwnd$ dynamics exhibit a dampened oscillation towards a given $cwnd$ size, upon cross traffic activity. Notice that $cwnd_k$ does not depend on previous $cwnd$ sizes, as with the other TCP variants.

E. Capacity and Congestion Plus Derivative TCP

TCP CCPD is our second attempt to design a delay based congestion avoidance scheme based on solid control theoretical approach, being a variant of CCP. The scheme $cwnd$ adjustment follows the same strategy of CCP. The difference is that it uses a proportional plus derivative controller as its control equation, as per Equation 6:

$$cwnd_k = Kp[B - x_k - in\_flight\_segs_k] + \frac{Kd}{t_k - t_{k-1}}[x_{k-1} + in\_flight\_segs_{k-1} + -x_k - in\_flight\_segs_k] \qquad (6)$$

where Kp is a proportional gain, Kd is a derivative gain, t_k and t_{k-1} are two consecutive ack reception epochs, and the other parameters are defined as per CCP congestion avoidance scheme. Typically, CCPD $cwnd$ dynamics present similar dampened oscillatory behavior as CCP, with a much faster period, due to its reaction to the derivative or variation of the number of packets backlogged.

IV. TCP Variants Performance Characterization

It is well known that TCP throughput performance is affected by the round trip time of the TCP session. This is a direct consequence of the congestion window mechanism of TCP, where only up to a $cwnd$ worth of bytes can be delivered without acknowledgements. Hence, for a fixed $cwnd$ size, from the sending of the first packet until the first acknowledgement arrives, a TCP session throughput is capped at $cwnd/rtt$.

As mentioned earlier, for all TCP variants, the size of the congestion window is computed by a specific algorithm at time of packet acknowledgement reception by the TCP source. In this section, we characterize TCP performance regarding data throughput in various network scenarios. For CCP, and CCPD protocols, we shall use $CCP(K_p)$ notation for CCP using proportional parameter K_p, whereas $CCPD(K_p, K_d)$ for CCPD using proportional and derivative parameters, K_p and K_d, respectively.

We evaluate the throughput performance of TCP variants in the presence of controlled cross traffic. Fig. 2 depicts the network scenario used for evaluating TCP protocols against interfering UDP and TCP types of cross traffic. One TCP session shares a 1Gbps access link with UDP cross traffic of 200Mbps intensity to a dumb-bell topology emulator highspeed network, depicted in Fig. 2 a). The PacketStorm 4XG IP Network Emulator [10] is used to vary the end-to-end round trip time of the TCP sessions. Two Alaxala switches [1] were used, AX-3630-24T2X and AX-2430-48T-B. As endpoints, Dell PowerEdge2950 Xeon 1.6GHz machines were used, running Linux 2.6.26.

Fig. 2 b) describes the timeline of the TCP and UDP sessions, with the TCP session lasting for 150 secs, and the UDP traffic starting 50 seconds after the TCP session start, and finishing 50 secs prior to the end of the TCP session. Figure 2 c) describes the timeline of a TCP and TCP two session scenario, where two TCP sessions compete for bottleneck link bandwidth.

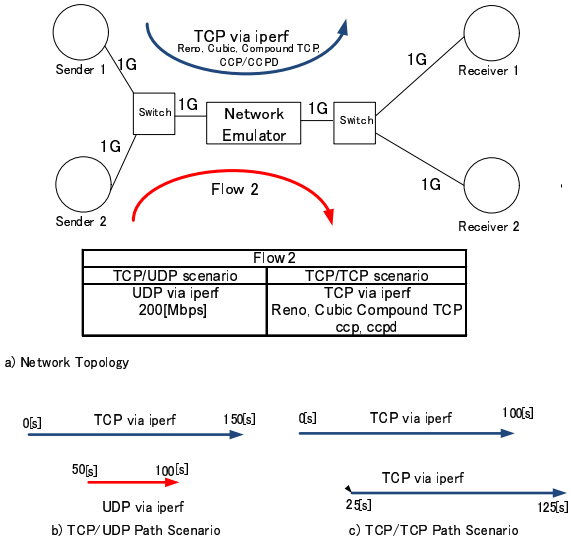

Fig. 2: TCP Coexisting Evaluation Scenarios

A. Coexisting TCP/UDP sessions

This experiment set is designed to study the performance of TCP variants on a single session, when facing cross traffic. Performance measurers of interest are throughput and throughput recovery, defined as the ratio between the throughput achieved after cross traffic exists the session path, divided by the amount of throughput achieved before the session experience any cross traffic.

1) Short round trip time paths: Figure 3 reports throughput performance of a TCP session subjected to UDP cross traffic on short rtts, similar to local or countrywide session. Overall, Compound TCP, Reno, and Cubic are best performers across all TCP variants, followed close by CCPD(4,4000), although the later is not able to reclaim as much throughput after UDP traffic goes away than the former protocols.

2) Large round trip time paths: Figure 4 reports TCP throughput performance over a session with large rtt, similar to transoceanic data transfers. Looking at the time prior to the UDP traffic injection, the outperformers are Compound TCP and Cubic, followed by CCPD(2,1000) and CCP(4). In the presence of UDP traffic, the best performers are Cubic, and most of CCPD protocols.

Figure 5 reports the throughput recovery ratio, which shows how much the TCP session is able to ramp up back after cross traffic is finished. The best performers for large rtt sessions

 ISBN: 978-1-61208-204-2

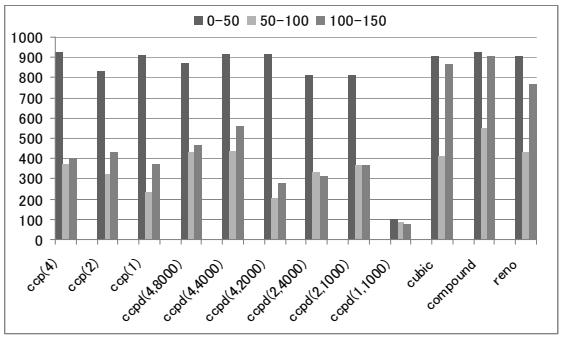

Fig. 3: TCP/UDP throughput - short rtt(20msecs)

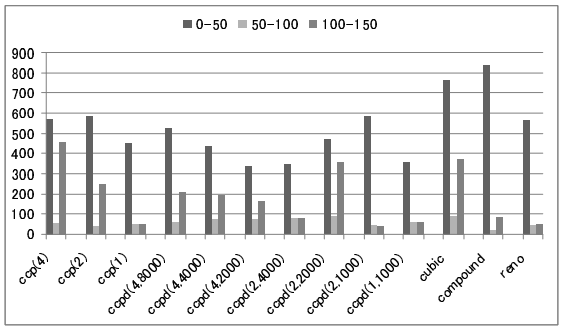

Fig. 4: TCP/UDP throughput - large rtt(200msecs)

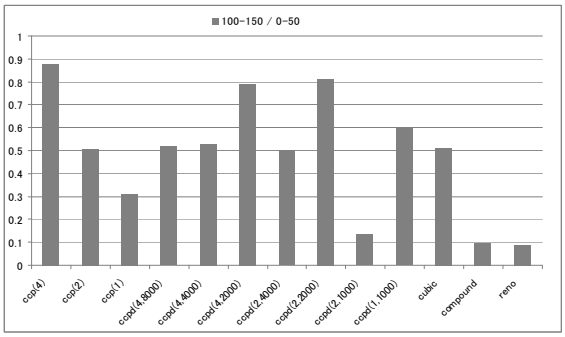

Fig. 5: TCP/UDP throughput recovery - large rtt(200msecs)

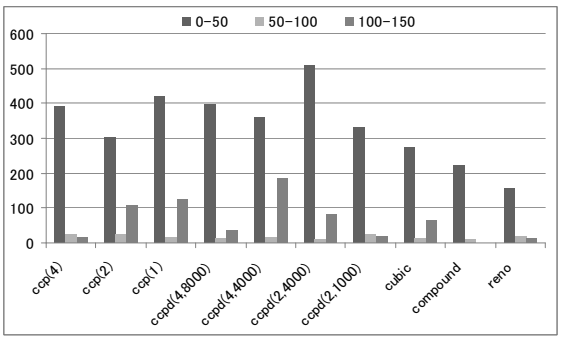

Fig. 6: TCP/UDP throughput - very large rtt(600msecs)

are CCP(4), CCPD(2,2000) and CCPD(4,2000). The worst performers are Reno and Compound TCP.

3) Very large round trip time paths: Figure 6 reports TCP throughput performance over a very large rtt, typically incurred in satellite paths. In this case, traditional TCP variants have the worst performance across all TCP variants investigated. Best performers are CCPD(2,4000), CCPD(4,4000) and CCP(1). For very large rtt paths, CCPD(2,4000) and CCPD(4,4000) seem to be the top TCP variants performers.

B. Coexisting TCP/TCP sessions

In this subsection, we investigate the throughput performance of two TCP flows sharing a single bottleneck. Two cases can be distinguished: homogeneous case, where the two TCP sessions belong to the same TCP variant; heterogeneous case, where the two TCP sessions belong to different TCP variants.

1) Small round trip time paths: Figure 7 reports throughput performance of two TCP sessions, staggered in time, over a short rtt path. For the initial period, with only a single session, all TCP variants perform similarly. During the period of the two sessions sharing a bottleneck, CCP with large alpha parameter delivers best performance. During "recovering period", where the first session leaves the system, Reno and Compound TCP present best throughput ramp up performance, followed by CCPD(4,4000) and Cubic as second best.

Figures 8 and 9 report throughput performance of CCPD(4,2000) competing with Reno, Cubic, and Compound TCP variants over a short rtt path. Figure 8 reports performance when the first flow is CCPD(4,2000), whereas Figure 9 reports the reverse scenario, when the second flow is CCPD(4,2000). In general, Flow 1 and flow 2 path bandwidth resource is shared unevenly. Comparing the two cases, throughput ramp up of flow 2 after flow 1 departs is best achieved by CCPD(4,2000), except when Compound TCP is used by flow 2, in this short rtt scenario.

Figures 10 and 11 report throughput performance of CCPD(4,4000) competing with Reno, Cubic, and Compound TCP variants over a short rtt path. In general, flow 1 and flow 2 path bandwidth resource is shared very unevenly. Comparing Figure 10 and 11, the following observations can be made. i) Throughput performance of all TCP variants are similar when there is no cross traffic; ii) CCPD(4,4000) is able to ramp up throughput to higher levels once cross traffic vanishes, except against Compound TCP ramp up performance, which again is better for this short rtt scenario. When compared with CCPD(4,2000) performance, it is clear that CCPD(4,4000) retains more throughput under TCP cross traffic for short rtt scenario.

2) Large round trip time paths: Figure 12 reports throughput performance of two TCP sessions, staggered in time, over a long rtt path. For the initial period, with only a single session, Compound TCP and Cubic deliver best throughput performance. During the period of the two sessions sharing a bottleneck, Cubic, Compound TCP and Reno present the best aggregate performance. Notice, however, that this is because the first flow retains most of its throughput prior to the sharing of bandwidth with the second flow. During "recovering period", where the first session leaves the system, Cubic and CCPD(4,2000) deliver best throughput.

 ISBN: 978-1-61208-204-2

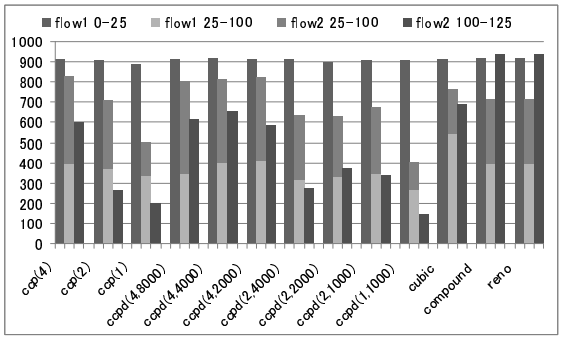

Fig. 7: TCP/TCP throughput - small rtt(20msecs)

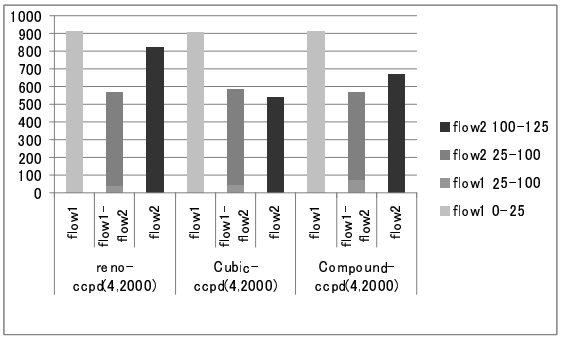

Fig. 9: TCP/CCPD(4,2000) throughput - small rtt(20msecs)

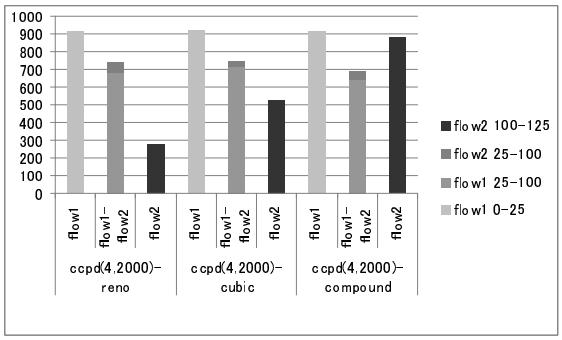

Fig. 8: CCPD(4,2000)/TCP throughput - small rtt(20msecs)

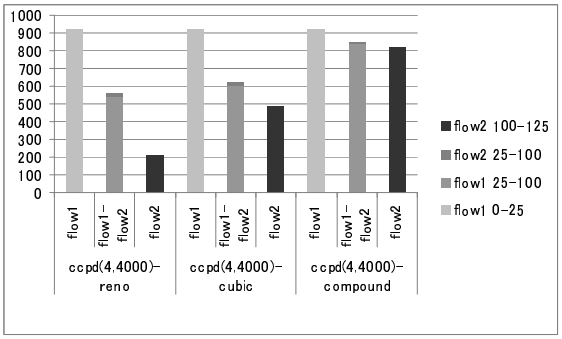

Fig. 10: CCPD(4,4000)/TCP throughput - small rtt(20msecs)

Figures 13 and 14 report throughput performance of CCPD(4,2000) competing with Reno, Cubic, and Compound TCP variants over a large rtt path. Figure 13 shows that Reno, Cubic, and Compound TCP variants deliver poor flow 2 throughput ramp up performance when both flows share path resources. In contrast, Figure 14 shows a much better flow 2 ramp up performance of CCPD(4,2000) for large rtt scenario. Moreover, CCPD(4,2000) is able to further ramp up flow 2 throughput to a highest level among all TCP variants.

Figures 15 and 16 report throughput performance of CCPD(4,4000) competing with Reno, Cubic, and Compound TCP variants over a large rtt path. TCP flow 2 throughput is low, as compared with CCPD(4,4000) flow 1, when both flows share path resources. Fig. 15 shows that Cubic TCP recovers flow 2 throughput the most, whereas Reno flow 2 has negligible throughput. Fig. 16 shows that CCPD(4,4000) flow 2 recovers the most throughput after cross traffic ends.

V. Discussions

Round trip time is used in the calculation of Retransmission Time Out (RTO). In addition, round trip time estimate may be used as an indication of path congestion in various TCP variants, such as TCP Vegas, and TCP CCP and CCPD. In these schemes, a TCP session minimum rtt, rtt_{min}, is computed, and current rtt measurement deviation from this minimum is taken as an indication of path congestion. Some TCP schemes also use an estimate of maximum rtt seen, or rtt_{max}. Care must be taken by these schemes so as to ensure robustness to path condition changes.

Firstly, an early TCP session may compute a rtt^e_{min}, whereas a late TCP session may compute a rtt^l_{min} over a same path, such that $rtt^e_{min} < rtt^l_{min}$. In this case, the early session may perceive less congestion than the late one, even though they share the same path, with the same cross traffic. Hence, an already established session may be biased to higher performance than a newly entrant one. This problem can be mitigated by having the TCP session to "release" some bandwidth once in a while. Judicious $cwnd$ variation hence is encouraged for that purpose. Another issue arises when a TCP path route changes, due to link/router failures in the network, causing path measures to become invalid. Path capacity estimates (TCP CCP and CCPD) need to be updated upon route changes.

VI. Future work

In this paper, we have characterized TCP performance over a high speed wired network scenario via open source experiments for the most widely used TCP variants, i.e., Cubic, Reno, and Compound TCP, as well as our proprietary variants, CCP and CCPD. We have shown the need to tune TCP variant parameters to network scenarios. In addition, we have selected appropriate CCP and CCPD parameters for short and long rtt paths. We are currently investigating fairness issues vis a vis TCP variants path condition estimators, such as rtt estimation for new and already established TCP sessions. We are also investigating how to make estimators more robust to sudden change of path conditions, such as re-routing.

Acknowledgment

Work supported in part by JSPS Grant-in-Aid for Scientific Research KAKENHI B (23300028).

 ISBN: 978-1-61208-204-2

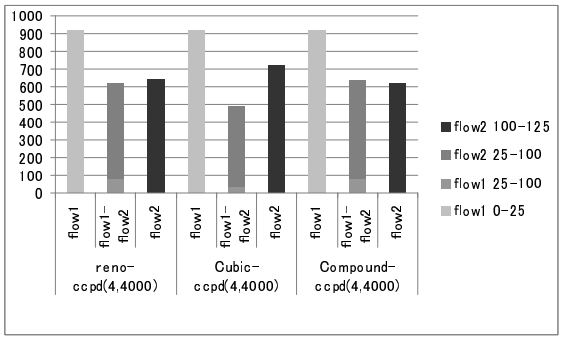

Fig. 11: TCP/CCPD(4,4000) throughput - small rtt(20msecs)

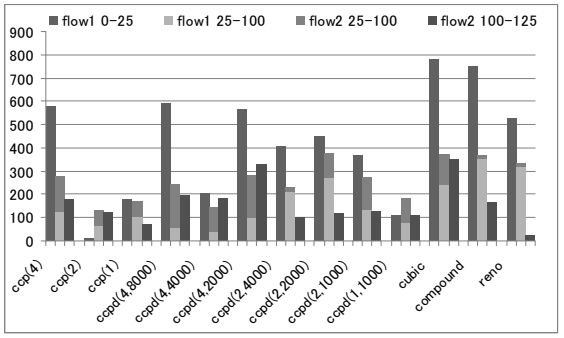

Fig. 12: TCP/TCP throughput - large rtt(200msecs)

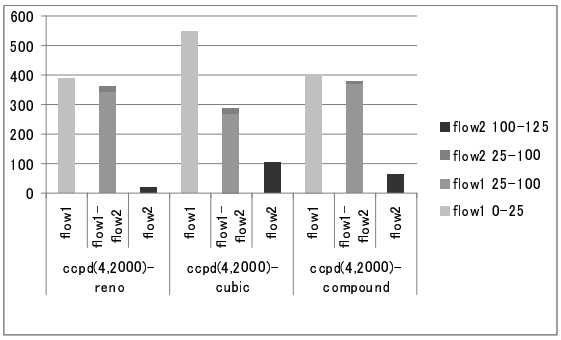

Fig. 13: CCPD(4,2000)/TCP throughput - large rtt(200msecs)

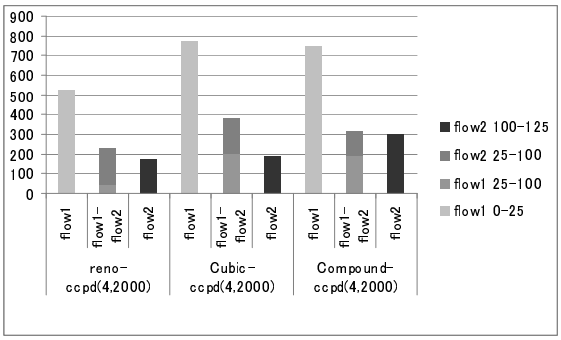

Fig. 14: TCP/CCPD(4,2000) throughput - large rtt(200msecs)

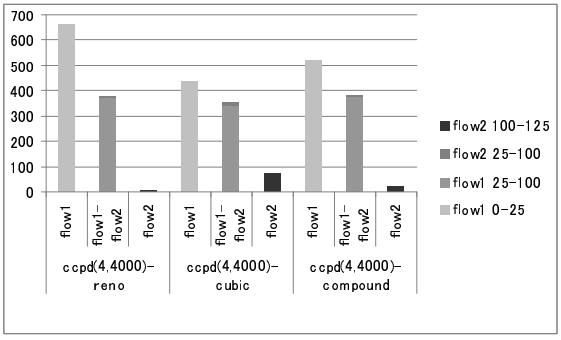

Fig. 15: CCPD(4,4000)/TCP throughput - large rtt(200msecs)

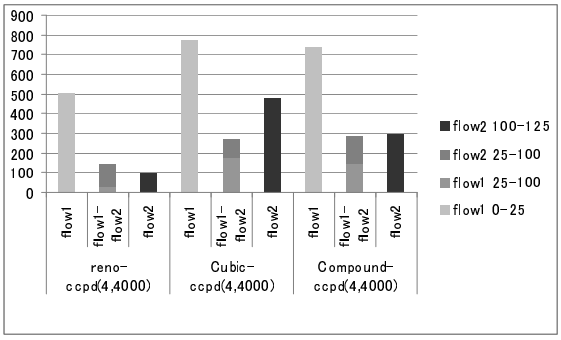

Fig. 16: TCP/CCPD(4,4000) throughput - large rtt(200msecs)

REFERENCES

[1] Alaxala Networks Corporation, “High Performance Layer 3 Switches” *http://www.alaxala.com/en/products/index.html*, accessed Apr. 16, 2012.

[2] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock, “Host-to-Host Congestion Control for TCP, ” *IEEE Communications Surveys & Tutorials*, Vol. 12, No. 3, pp. 304-342, Third Quarter 2010.

[3] M. Allman, V. Paxson, and W. Stevens, “TCP Congestion Control,” *IETF RFC 2581*, April 1999.

[4] M. Alnuem, J. Mellor, and R. Fretwell, “New Algorithm to Control TCP Behavior over Lossy Links, ” *IEEE Int. Conference on Advanced Computer Control*, pp. 236-240, Jan 2009.

[5] A. Ahmed, S.M.H. Zaidi, and N. Ahmed, “Performance evaluation of Transmission Control Protocol in mobile ad hoc networks, ” *IEEE Int. Networking and Communication Conference*, pp. 13-18, June 2004.

[6] D. Cavendish, K. Kumazoe, M. Tsuru, Y. Oie, and M. Gerla, “Capacity and Congestion Probing: TCP Congestion Avoidance via Path Capacity and Storage Estimation,” *Second Int. Conference on Evolving Internet*, September 2010.

[7] D. Cavendish, Hiraku Kuwahara, K. Kumazoe, M. Tsuru, and Y. Oie, “TCP Congestion Avoidance using Proportional plus Derivative Control,” *Third Int. Conference on Evolving Internet*, June 2011.

[8] J. Chicco, D. Collange, and A. Blanc, “Simulation Study of TCP Variants,” *IEEE Int. Symposium on Computers and Communication*, pp. 50-55, June 2010.

[9] S. Henna,“A Throughput Analysis of TCP Variants in Mobile Wireless Networks,” *Third Int. Conference on Next Generation Mobile Applications, Services and Technologies - NGMAST*, pp.279-284, Sept. 2009.

[10] PacketStorm Communications, Inc., “PacketStorm 4XG Network Emulator” *http://www.packetstorm.com/psc/psc.nsf/site/4XG-software*, accessed April 16, 2012.

[11] I. Rhee, L. Xu, and S. Ha, “CUBIC for Fast Long-Distance Networks,” *Internet Draft*, draft-rhee-tcpm-ctcp-02, August 20088.

[12] M. Sridharan, K. Tan, D. Bansal, and D. Thaler, “Compound TCP: A New Congestion Control for High-Speed and Long Distance Networks,” *Internet Draft*, draft-sridharan-tcpm-ctcp-02, November 20088.

[13] S. Waghmare, A. Parab, P. Nikose, and S.J. Bhosale, “Comparative analysis of different TCP variants in a wireless environment,” *IEEE 3rd Int. Conference on Electronics Computer Technology*, Vol.4, p.158-162, April 2011.

 ISBN: 978-1-61208-204-2