SQUARE: FF on CUBIC TCP over Links with Differenct RTT

Tomoki Kozu, Yuria Akiyama, Saneyasu Yamaguchi
Electrical Engineering and Electronics
Kogakuin University Graduate School
Tokyo, Japan
{cm13012, cm13001}@ns.kogakuin.ac.jp, sane@cc.kogakuin.ac.jp

***Abstract*—CUBIC TPC is a congestion control algorithm for TCP. It is the current default TCP algorithm in Linux. Because many Internet servers, such as web servers, are running on Linux operating system, keeping throughput obtained with this TCP algorithm enough is quite important. Then, many performance studies have been published. However, most of these studies were based on network simulators. Thus, evaluations using an actual TCP implementation and actual network elements are important in addition to these existing studies. In this paper, we focus on RTT (round trip time) fairness of CUBIC TCP, which is performance fairness among CUBIC TCP connections with different network delay times. Firstly, we present RTT fairness evaluation using actual TCP implementations and actual network elements and show that the RTT fairness is not enough. Secondly, we discuss the cause of the unfairness based on CUBIC TCP behavior. Thirdly, we propose a method for improving RTT fairness of CUBIC TCP, which is called *SQUARE: FF* (Feedback for Fairness based on square root of RTT). It has a mechanism with which an insufficient congestion window size is automatically increased. Finally, we present evaluation results and demonstrate that the proposed method provides better fairness than original CUBIC TCP implementation.**

Keywords— Internet, TCPIP, Transport protocols

I. INTRODUCTION

TCP (Transmission Control Protocol) is one of the core protocols of the Internet. Its implementations have congestion control algorithms and their behavior has strong impact on obtained throughput. Classical TCP implementations and some current implementations have TCP Reno [1]. It has been widely used, but it is pointed out that enough throughput cannot be obtained over current long fat networks with this algorithm [2]. For this issue, several new congestion control algorithms were proposed, such as TCP Vegas [3], BIC TCP [4], CUBIC TCP [5], and Compound TCP [6]. CUBIC TCP is considered one of the most suitable algorithms for current networks. Then, it is the default TCP algorithm of current Linux operating system. Because most Internet traffic is sent from servers, such as web server processes, to client PCs and most of these server processes are running on Linux operating system, CUBIC TCP is one of the most important TCP congestion control algorithms. Thus, we think improving RTT fairness of this TCP implementation contributes many users of the Internet.

On TCP congestion control algorithms, their performance has been rigorously discussed using network simulators, such as NS2 [7]. However, performance obtained with actual TCP implementations and actual network elements have not been studied enough. Especially, RTT fairness, which is performance fairness between connections with different RTTs, has not been discussed.

In this paper, we focus on RTT fairness of CUBIC TCP. We evaluate "fairness"' with Jain's Fairness Index [8], and define our objective to improve this value. We evaluate RTT fairness of CUBIC TCP connections and demonstrate that the fairness is not sufficient. Then, we discuss the cause of unfairness, and we propose a method for improving RTT fairness.

II. RELATED WORK

A. CUBIC TCP

CUBIC TCP [5] is an algorithm based on BIC TCP [4], so it has high scalability similar to BIC TCP. Moreover, it has better TCP fairness and RTT fairness than BIC TCP. TCP fairness is performance fairness among TCP algorithms. RTT fairness is performance fairness among connections with different RTTs, as described above. CUBIC TCP uses the following cubic function, while BIC TCP uses binary search.

$$cwnd = C(t-K)^3 + W_{max} \qquad (1)$$

$$K = \sqrt[3]{\frac{W_{max}\beta}{C}} \qquad (2)$$

cwnd is congestion window size, *t* is time from the last packet loss, W_{max} is congestion window size at the last packet loss, *C* and β are parameters to tune increasing speed in usual state and dropping ratio at packet losses, respectively. The larger *C* results in the faster increase. In most cases, *C* is 0.4 and β is 0.2. Because this function uses time from a packet loss and does not depend on ACK receiving, it is expected to provide good RTT fairness and avoid too aggressive increasing over a short RTT network. As shown in formula (1), *K* is time to the inflection point and *cwnd* achieves W_{max} at time *K*. Therefore, you can

understand *K* as time to recovery.

B. *Fairness on TCP*

The following works are on RTT fairness of TCP. For achieving RTT fairness, Floyd *et al.* proposed Constant-Rate window increase algorithms [10] [11], with which each connection increases its window size by roughly $a*r^2$ packets each roundtrip time, for some fixed constant *a*, and for *r* the calculated average roundtrip time. Henderson *et al.* profoundly investigated RTT fairness and showed Constant-Rate policy [10] [11] could improve fairness dramatically [12].

Marfia *et al.* suggested that TCP's RTT unfairness was caused by its ACK-based mechanism and proposed a new TCP algorithm named TCP-Libra [13]. It increases congestion window size of large RTT connections more quickly. Ogura *et al.* proposed a new TCP algorithm, which is named "HRF (Hybrid RTT Fair)-TCP" for improving RTT fairness [14]. This work also presented analytical models. They evaluated the proposed method with both a simulator and their implementation.

The works in [10], [11], and [12] were based on additive increase policies. Therefore, these works are useful for classical TCP algorithms, such as TCP Reno, but are not directly effective for modern fast TCP algorithms and modern operating systems. On the other hand, our work focuses on one of the most important modern fast TCP algorithms and discusses with a practical TCP implementation. The methods in work [13] and [14] are pioneer ones, but their purposes are quite different from that of our work. They aimed to propose novel TCP algorithms while we aim to improve a currently widely working TCP algorithm.

In work [15],[16] and [17], methods for improving RTT fairness of CUBIC TCP were proposed. These methods were evaluated with actual TCP implementations and actual network elements. However, the former was based on a tuning function. Thus, achieved fairness depended on heuristic optimization. The latter is straightforward one and it cannot present enough fairness with large RTT difference.

III. RTT FAIRNESS ON CUBIC TCP

A. *RTT Fairness Evaluation*

In this subsection, we give RTT fairness evaluation between CUBIC TCP connections. We constructed the experimental network shown in Fig. 1 and measured CUBIC TCP performance.

The executed experiments are as follows. We established two

Table 1. Computer specifications

CPU	Intel Celeron CPU G530, 2.40GHz
Memory	2 [GB]
OS	(PC1, PC2,) Linux 2.6.32.27 (PC3) Linux 2.6.35.6 (Network Emulator) FreeBSD 8.2
NIC	Intel PRO/1000 GT Desktop Adapter

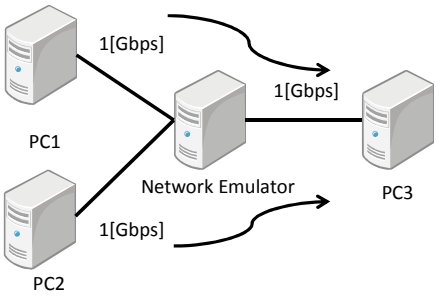

Fig. 1 Experiental Network

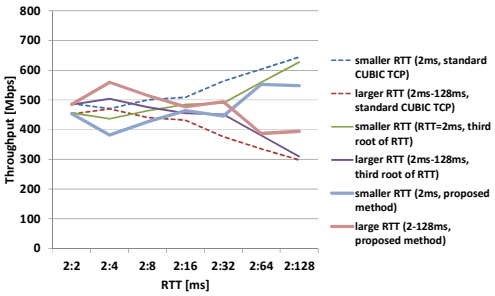

Fig. 2 Throughput

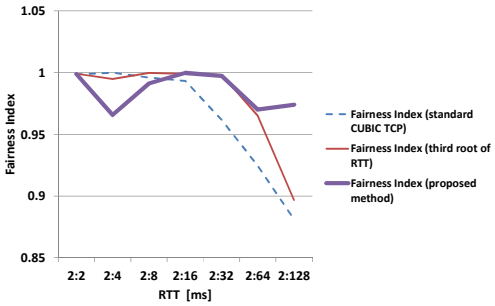

Fig. 3 Fairness Index

CUBIC TCP connections. One was a connection between PC1 and PC3. The other was a connection between PC2 and PC3. These connections were established by netperf [1], which was a network throughput benchmark software. Packets were sent from PC1 and PC2 to PC3. The network delay times of these connections were set independently. Both of the connections shared network elements between the emulator and PC3. The network emulator was a network bridge which was able to emulate network delay. This emulator was constructed by FreeBSD Dummynet0. All elements in this experimental network supported 1Gbps Ethernet. Specifications of the computers in this network are shown in Table 1. *C* and β of CUBIC TCP were set 0.4 and 0.2, respectively. These are the standard settings [5] and the default settings of the current Linux operating system. Advertise windows size was set 16MB.

The experimental results are shown in Fig. 2 and Fig. 3. In

these figures, the network delay time of one connection was always 2ms. The delay time of the other connection ranged from 2ms to 128ms. The horizontal axes of these graphs represent RTTs of the two connections. The vertical axis in Fig. 2 shows the obtained throughput and the vertical axis of Fig. 3 depicts Jain's Fairness Index [8]. Fig. 2 indicates that the throughput obtained by the connection with the smaller RTT is higher than that with larger RTT. These graphs indicate also that performance difference grows and Fairness Index declines as RTT difference grows. In the case of RTT 2ms and RTT 128ms, one connection obtained more than twice performance. From these, we can conclude that RTT fairness of CUBIC TCP is not enough.

B. Cause of Unfairness

In this subsection, we explain why different *K*s result in unfair performance.

In case of a usual network, which is constructed of routers using drop tail, all connections encounter congestion and a packet loss at the same time. This is called "global synchronization". Fig. 4 illustrates transitions of congestion window size of two connections with different *K*s. We call the connections with larger and smaller *K* "connection L" and "connection S", respectively. K_L and K_S in the figure are "*K* of connection L" and "*K* of connection S", respectively. Congestion and a packet loss occurred at *timeA*. Each connection sets its *K* according to its W_{max}, as described in section II.A. Therefore, connection L and connection S set larger and smaller *K*, respectively.

The next congestion occurred at *timeB*. It was the time when sum of the both congestion window sizes reached the sum at *timeA*. *TimeB* was between K_L and K_S. As a result, connection L reached the next congestion before *timeA* + K_L and its congestion window size at *timeB* is smaller than W_{max} of L at *timeA*. In other words, congestion window size of connection L was decreased. On the contrary, connection S encountered the next congestion after *timeA* + K_S. This indicates that congestion window size of connection S was increased. From the both connections' behavior, we can say that congestion window size of a larger RTT connection will be decreased as congestion avoidance phases will be repeated. That is to say, a connection with larger RTT requires a larger congestion window size but the connection cannot keep larger congestion window size because larger congestion window size results in larger *K* and decrease of congestion window size.

C. Feedback to K with third root of RTT

The second most straightforward method for improving RTT fairness is giving feedback to *K* with third root of *RTT* as the following formula.

$$K = \sqrt[3]{\frac{W_{max}\beta}{C\,RTT}} \qquad (3)$$

As described in section III.A, an obtained throughput is limited to *WindowSize/RTT* by window size. Thus, fairness is

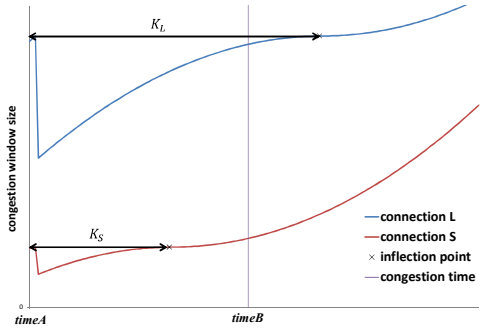

Fig. 4 Cause of unfairness

achieved when the all connections have the same *WindowSize/RTT*. In this case, window size and W_{max} are proportional to *RTT*. In this subsection, we assume this situation as a target situation, and discuss a method which tries to move congestion window size and W_{max} close to this situation. If W_{max} is exactly proportional to *RTT*, performance fairness is achieved and *K* is the same for all connections with this method. In theory, the next congestion will occur after *K* and the all connections encounter packet losses at their flexion point, then all connections keep their W_{max} with the same *K*. However, this balance is not stable. As described in section III.I.A, RTT fairness is not achieved with constant *K*. For this issue, this method adjusts window size and W_{max} close to the target situation. With formula (3), a larger RTT connection with insufficient W_{max} gains smaller *K* than other connections. Therefore, this connection can recover its congestion window size faster. As described in section III-B, faster recovery results in increase of congestion window size. Similarly, a connection with excess window size gains larger *K* then its window size is decreased. That is to say, a connection with smaller W_{max}, *K*, and performance than the target situation increases their values and performance, and a connection with larger values decreases their values and performance. As a result, this mechanism leads a situation into a fair one.

Performance evaluation with this method is shown in Fig. 2 and Fig. 3. The figures show that RTT fairness is improved. However, the complete fairness is not obtained, especially with large RTT difference such as RTT 2ms and 128ms. From these, we can conclude that more advantage for larger RTT connection is required in order to achieve more fairness with large RTT difference.

IV. Proposal

In this section, we propose a method for improving RTT fairness using square root of *RTT*. We call this method "SQUARE: FF (Feedback for Fairness)". From the discussion in the previous section, we can conclude that larger feedback to large RTT connection than the third root of RTT is required. We propose to adjust *K* with the following formula.

$$\mathrm{K} = \sqrt[3]{\frac{W_{max}\beta}{C}}\frac{1}{\sqrt[2]{RTT}} \qquad (4)$$

Giving feedback based on square root of *RTT*, this method presents more advantage to larger RTT connections.

V. Evaluation

In this section, performance evaluation of the proposed method is presented. We have implemented the method in Linux operating system and evaluated our proposal with the network in Fig. 1. The experimental setup is the same as that in section III.A.

The obtained performances and fairness are shown in Fig. 2 and Fig. 3. These figures indicate that the proposed method achieved much better fairness with large RTT difference than standard CUBIC TCP and the straightforward method with third root of *RTT*. In the cases of small RTTs, such as 2ms vs 4ms and 2ms vs 8ms, a little overshooting is detected. That is, the larger RTT connection obtained higher throughput. Comparing fairness improvement with large RTT difference, we think this little fairness decline is not severe.

VI. Conclusion

In this paper, we presented RTT fairness evaluation of CUBIC TCP and demonstrated its insufficient fairness. Then, we have proposed a method for improving RTT fairness of CUBIC TCP by giving feedback for fairness to *K* according to square root of RTT. Our evaluation has demonstrated that the proposed method has been able to improve RTT fairness of CUBIC TCP significantly.

We plan to evaluate fairness between the proposed method and other TCP congestion control algorithms.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers 24300034, 25280022, 26730040.

References

[1] W. Stevens, “TCP Congestion Control,” IETF RFC 2581, 1999.

[2] Dina Katabi, Mark Handley, and Charlie Rohrs, “Congestion Control for High Bandwidth-Delay Product Networks,” SIGCOMM '02. Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 89-102, 2002.

[3] Jeonghoon Mo, Richard J. La, Venkat Anantharam, and Jean Walrand, “Analysis and Comparison of TCP Reno and Vegas,” INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, March 1999.

[4] L. Xu, K. Harfoush and I. Rhee, “Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks,” Proc. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, March 2004.

[5] Injong Rhee, and Lisong Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant,” Proc. Workshop on Protocols for Fast Long Distance Networks, 2005.

[6] Kun Tan, Jingmin Song, Qian Zhang, and Murari Sridharan, “A Compound TCP Approach for High-speed and Long Distance Networks,” Proceedings 25th Conference on Computer Communications (InfoCom 2006), April 2006.

[7] Network Simulator-ns-2, http://www.isi.edu/nsnam/ns/

[8] D. Chiu and R. Jain, “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks,” Computer Networks and ISDN Systems, Volume 17, Issue 1, pp. 1-14, 1989.

[9] SUZUKAWA Ryuji, and ADACHI Naotoshi, “Enhancing Fairness of Bandwidth Sharing for CUBIC-TCP,” IEICE Technical Report NS2008-108, 2008 (in Japanese).

[10] Sally Floyd, “Connections with multiple congested gateways in packet-switched networks part 1: one-way traffic,” SIGCOMM Comput. Commun. Rev., Vol. 21, No. 5, pp. 30-47, Oct. 1991.

[11] Floyd and V. Jacobson, “On Traffic Phase Effects in Packet-Switched Gateways,” Computer Communication Review (ACM), Vol. 21, No. 2, 1991.

[12] Thomas R. Henderson, Emile Sahouria, Steven Mccanne, and Y H. Katz, “On Improving the Fairness of TCP Congestion Avoidance,” Global Telecommunications Conference, 1998. GLOBECOM 1998. The Bridge to Global Integration. IEEE, vol.1, pp. 539-544, 1998.

[13] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla M. Y. Sanadidi, and M. Roccetti, “Balancing Video on Demand Flows over Links with Heterogeneous Delays,” Proceedings of the 3rd international conference on Mobile multimedia communications, pp. 1-6, 2007.

[14] Kazumine OGURA, Yohei NEMOTO, Zhou SU, and Jiro KATTO, “A New TCP Congestion Control Supporting RTT-Fairness,” IEICE TRANSACTIONS on Information and Systems Vol.E95-D No.2 pp.523-531, 2012.

[15] Tomoki Kozu, Yuria Akiyama and Saneyasu Yamaguchi, “Improving RTT Fairness on CUBIC TCP,” The First International Symposium on Computing and Networking, 2013.

[16] Tomoki Kozu, Yuria Akiyama, Saneyasu Yamaguchi, “Improving RTT Fairness on CUBIC TCP,” International Journal of Networking and Computing, Vol 4, No 2(2014)

[17] Tomoki KOZU, Yuria AKIYAMA, Saneyasu YAMAGUCHI, “Improving of RTT fairness on CUBIC TCP with Adjusting Congestion Window Recovery Time,” IEICE Technical Report, vol. 113, no. 472, NS2013-193, pp. 97-102, 2014 (in Japanese).

[18] netperf homepage, http://www.netperf.org/netperf/