Modified Controlling Queue Delay for TCP Fairness Improvement

Masato Hanai
Electrical Engineering and Electronics, Kogakuin University Graduate School
Tokyo, Japan
cm16036@ns.kogakuin.ac.jp

Saneyasu Yamaguchi
Electrical Engineering and Electronics, Kogakuin University Graduate School
Tokyo, Japan
sane@cc.kogakuin.ac.jp

Aki Kobayashi
Electrical Engineering and Electronics, Kogakuin University Graduate School
Tokyo, Japan
aki@cc.kogakuin.ac.jp

***Abstract*—For overcoming performance decline caused by current long fat networks, many high performance TCPs have been proposed, such as Compound TCP and CUBIC TCP. These proposals have raised a new issue, which is fairness among these modern TCPs. For this issue, several fairness evaluations and some proposals for improving fairness were published. In our past work, we focused on Controlling Queue Delay (*CoDel*) and proposed a method for improving performance fairness of modern TCPs by modifying CoDel. However, this existing method assumed that the most bandwidth-consuming connection was given for network elements. In this paper, we discuss a method for improving TCP fairness without such an assumption. First, we present fairness evaluation among modern TCPs with and without CoDel, and then demonstrate that fairness among these TCPs is very poor. Second, we propose a method for improving performance fairness. The method monitors network packets and guesses the most bandwidth-consuming connection from the monitored packets. Then, it drops packets in the most bandwidth-consuming connections more aggressively in the network element. Third, we evaluate our proposed method and show that our method can improve performance fairness.**

Keywords—CoDel; TCP fairness; congestion control algorithm

I. Introduction

TCP Reno [1] is the classical standard TCP congestion avoidance algorithm. It is still widely used. However, it is pointed out that enough throughput is not obtained using TCP Reno over current long fat networks [1][2]. Thus, many TCP congestion avoidance algorithms, such as BIC TCP [3], CUBIC TCP [4], and Compound TCP [5] were proposed in order to remedy this issue. In this paper, we call these algorithms modern TCPs. These proposals raised a new issue of fairness among modern TCP algorithms [2][6], which is called TCP Friendly. Accordingly, some fairness evaluations were executed [7][8], and these demonstrated that performance fairness among these TCP algorithms were severe.

For this issue, we proposed a method for improving TCP fairness[9] based on CoDel, and demonstrated that the method could improve TCP fairness. However, this method assumed that the most bandwidth-consuming connection was given for network elements.

In this paper, we focus on CoDel and discuss a method for improving TCP fairness without such an assumption. We evaluate performance fairness among modern TCPs with and without CoDel, and then demonstrate that the fairness is severely low. For the issue, we propose a method for improving performance fairness by modifying CoDel. The proposed method monitors network packets in network elements and guesses the most bandwidth-consuming connection from the packets. It drops packets in the most bandwidth-consuming connections more aggressively in the network element.

II. Related Work

A. TCP Congestion Avoidance Algorithms

In order to avoid network congestion, TCP implementations manage congestion window size and controls output speed. There are many TCP congestion control algorithms. These are classified into three general groups, loss-based methods, delay-based methods, and hybrid methods.

Loss-based methods manage congestion window size based on packet losses. In usual cases, congestion window size is increased every ACK packet receiving. When a packet loss occurs, congestion window size is decreased significantly. TCP Reno [1], BIC TCP [3], and CUBIC TCP [4] are loss-based methods.

Delay based methods manage congestion window size based on RTT. These methods decrease congestion windows size according to RTT increase, which implies increase of network routers' load and queue length. While loss-based methods decrease its congestion window size after congestion, delay-based methods decrease it before congestion. Thus, obtained throughput is expected to be stable. However, these methods have a performance issue. In case of sharing network link with a loss-based TCP, performance obtained by a delay-based method is much less than that of a loss-based method, because a delay-based method decreases its congestion window size

before congestion and a loss-based method does not decrease it until a packet loss [2]. TCP Vegas [10] is a delay-based method.

Hybrid methods are methods adopting both loss-based policy and delay-based policy. Compound TCP [5] is a hybrid method.

B. CUBIC TCP and Compound TCP

CUBIC TCP [4] is an algorithm based on BIC TCP [3], 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.

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

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

cwnd is congestion window size, t is time from the last packet loss, W_{max} is congestion window size at the last 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.

Compound TCP determines its send window size (*swnd*) using both loss-based *cwnd* and delay-based *dwnd*. The algorithm for *cwnd* is composed of the slow start phase and the congestion avoidance phase as well as TCP Reno.

C. TCP Fairness

The following works are on improving TCP fairness. Itsumi et al. proposed a method for improving TCP fairness by dropping packets[8]. The method monitors queue length in a router, and a packet is dropped when queue length grows sharply. Hasegawa et al. proposed a method for dynamic optimizing of RED parameters in backbone routers [7]. The method optimized the parameters according to extent of congestion. These works were based on simulation[11]. Thus, a discussion based on actual TCP implementations and actual network elements is required in addition to these works.

D. RED

RED[12] is a queue managing method with which a packet is dropped according to probability based on average queue length in a network element. In the case of using Tail Drop, the all packet in the all connections are dropped in a short period when queue length reaches buffer length. On the contrary, RED executes distributed packet losses. It is expected that performance fairness is improved with RED.

E. Active Packet Dropping

Active Packet Dropping [13] are methods for improving TCP fairness. There are two types of methods, the static method and the dynamic method. Both of these methods are based on RED. These methods preferentially drop packets in bandwidth-consuming connections.

The static method assumes that the most bandwidth-consuming connection is given for a router. The router drops

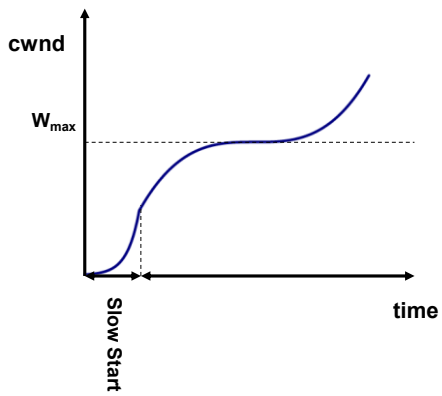

Fig. 1 Transition of congestion window size of CUBIC TCP

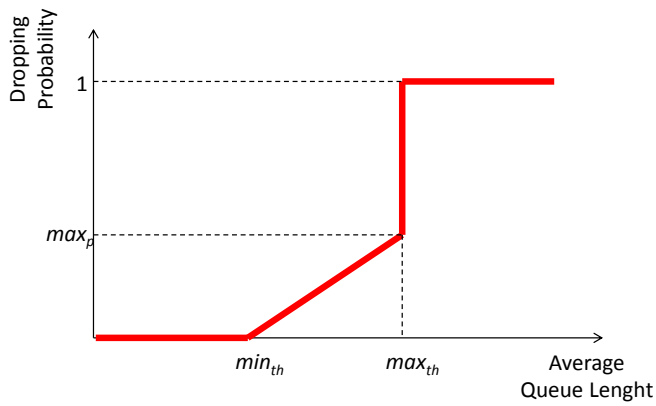

Fig. 2 Relation between average queue length and packet dropping probability on RED

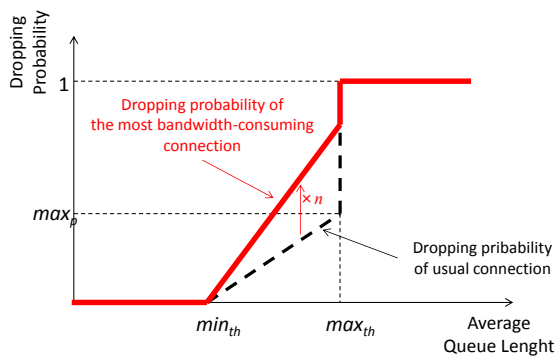

Fig. 3. Active Packet Dropping

packets in the connection preferentially using the probability in Fig. 3. The probability is n times greater than the standard RED. The dynamic method does not assume that the most bandwidth-consuming connection is given. The router monitors traffic and estimates the most consuming connection. Then, the router drops packets in the estimated connection using probability in Fig. 3.

The evaluation with an actual TCP implementation and an actual network elements in [13] demonstrates that both of the methods can improve TCP fairness. Especially, the static

method with the suitable setting can significantly improve the fairness. However, it is also demonstrated that the total performance is declined with these methods, because these methods actively drops packet.

F. Bufferbloat

Current network routes have huge size of buffers and these buffers are always full of packets. As a result, every packet has to experience long queuing delay (the time spent waiting to be processed or transmitted). At the congestion control in TCP, buffer size is desired to exceed BDP (bandwidth-delay product), where bandwidth is the bottleneck link and delay is the RTT (round-trip time) between sender and destination. Therefore, this may decline performance. This issue is called *Bufferbloat* [14]. For this issue, CoDel was proposed[15]. It will be described in the next subsection.

G. CoDel

Nichols et al. proposed a new packet queue scheduling algorithm, called CoDel[15]. This drops packets when queue delay time, time length between time at enqueue and dequeue of a packet, exceeds the threshold time, which is called *target*. The threshold is 5 ms with the default setup.

Unlike RED, CoDel has only one parameter, i.e. target, and dropping is determined only with queueing delay time. Because of this ease of use, this is expected to become popular. We think that CoDel can improve TCP fairness, like RED, by controlling a queue, and discussion on improving fairness with CoDel is important.

H. Static Fairnss Improving Method

A method for improving performance fairness of modern TCPs by modifying CoDel was proposed [9]. CoDel always drops packet when queue delay time exceeds the `target`. We proposed to introduce dropping provability p for the connections except for the most bandwidth-consuming connection. For the most bandwidth-consuming connection, a packet that exceeds the target was always dropped. For the other connections, an exceeding packet was dropped with probability p.

III. TCP FAIRNESS EVALUATION

In this section, we evaluated the performance fairness between TCP algorithms with TailDrop and with CoDel.

We have performed iperf [16] using CUBIC TCP and Compound TCP over the network shown in Fig. 4. PC CoDel and PC Delay are network switches. PC CoDel manages the queue using CoDel if it is enabled. PC Delay emulates network delay using *Netem*. The specifications of PC Linux 1, PC Linux 2, and PC Windows are shown in TABLE I. Those of PC CoDel and PC Delay are shown in TABLE II. All of these elements support 1 Gigabit Ethernet. 10 connections are established for each TCP algorithm, i.e. total 20 connections, simultaneously.

The experimental results are depicted in Fig. 5 and Fig. 6. The horizontal axes show the two way network delay time emulated in PC Delay. The vertical axis in FIg. 5 shows the average throughput of 10 connections of each TCP algorithm. This figure demonstrates that fairness without CoDel is severe. Linux with CUBIC TCP remarkably outperforms Windows with Compound TCP. Especially, the fairness with 64 ms of emulated

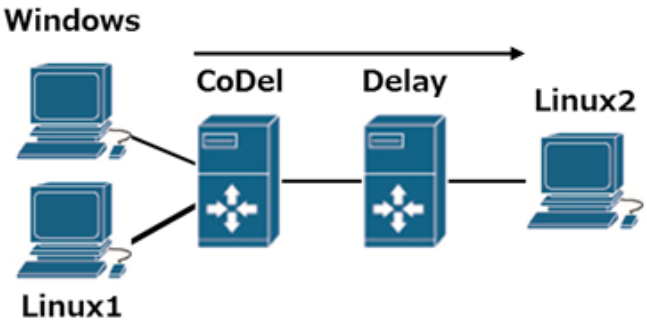

Fig. 4. Experimental Network

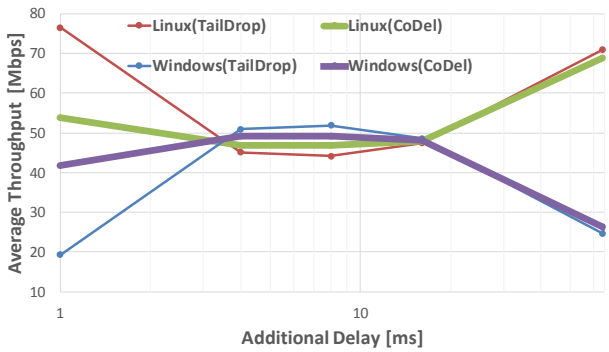

Fig. 5. Experimental results(Average Throughput)

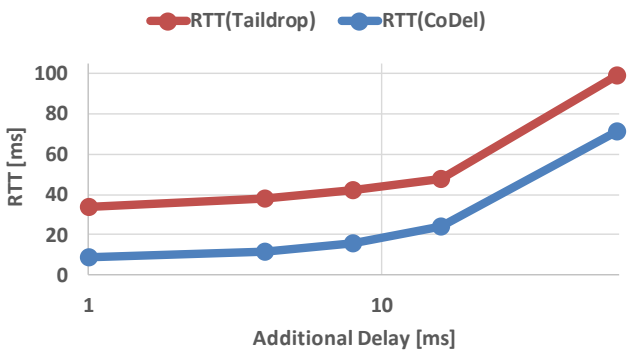

Fig. 6. Experimental results (RTT)

TABLE I. SPECIFICATION OF COMPUTERS (1)

CPU	AMD Turion, 2.20[GHz]
Memory	4[GB]
OS	(Linux1) Linux 4.2.3 (Linux2) Linux 3.3.4 (Windows) Windows7 Enterprise

TABLE II. SPECIFICATION OF COMPUTERS (2)

CPU	Intel CelronG540, 2.4[GHz]
Memory	2[GB]
OS	(Codel, Delay) Linux 3.17.4

network delay is poor. We can see also that utilizing CoDel improves TCP fairness in some cases. However, the fairness with large network delay, such as 64 ms, is still severe. Fig. 6 shows CoDel effectively reduces communication RTT without decline of throughput performance.

IV. PPOPOSED METHOD

A. Improving CoDel

In this section, we propose a method for improving TCP fairness without an assumption that the most bandwidth-consuming connection is given for network elements.

The most bandwidth-consuming connection is estimated as follows, see Fig. 7. The most consuming connection is defined as the connection that sends the most packets to the network element. The network element records the connection information of a packet every *rec_int* packets receiving, which is recording interval. The element stores the latest *hist_len* records, which is history length. Then, the most frequently stored connection in the log is extracted every *stat_int* packets receiving, which is statistical interval. The determined connection is assumed as the most bandwidth-consuming connection.

Packets in the estimated most consuming connection are dropped more actively according to the static method in II.H than those in the other connections. That is, a packet in the most bandwidth-consuming connection that exceeds the target is always dropped. An exceeding packet in the other connections is dropped with dropping probability p. p is a tuning parameter. Fig. 8 illustrates the behavior of the proposed method.

B. Implementation

We have implemented the proposed method by modifying the implementation of CoDel in Linux. Devices are identified using MAC addresses. Connection information is composed of MAC addresses of the sender and the receiver. Packet dropping with probability p should be executed using random number, but our implementation periodically drops a packet once per $1/p$ packets using a loop counter for simplyfying.

V. EVALUATION

In this section, we evaluate our method. We measured performance using iperf on the network in Fig. 4 with 1, 4, 8, 16, and 64 ms emulated network delay. The relation between network delay time and fairness is shown in Fig. 9. The horizontal axis is added network delay. The vertical axis is Jain's Fairness Index[17]. Fig. 10 depicts the total throughput of all the methods. Fig. 11 shows the average throughput of each operating system. Fig. 12 shows the relation between the dropping probability and Fairness Index. In these figures, "*Static*" and "*Dynamic*" mean the existing static method and the proposed dynamic method, respectively. We used the dropping probability $p = 1/3$ for Fig. 9 to Fig. 11 because the work [9] showed that the probability provided the best fairness.

Fig. 9, 10, and 11 indicate that the proposed method can improve TCP fairness remarkably. The static method can improve the fairness with 64 ms delay, but cannot improve with other delay times. This indicates that *static* the method

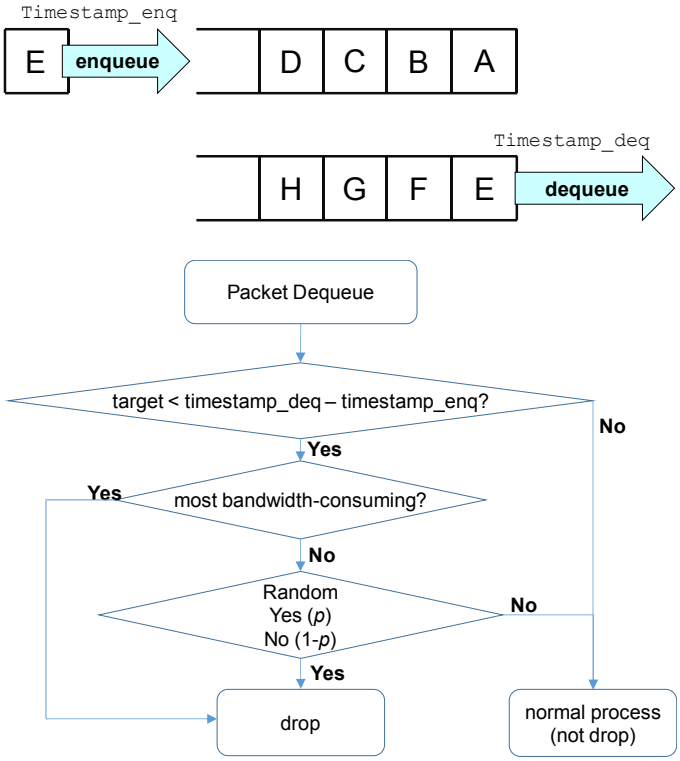

Fig. 7. proposed method (decision of drop)

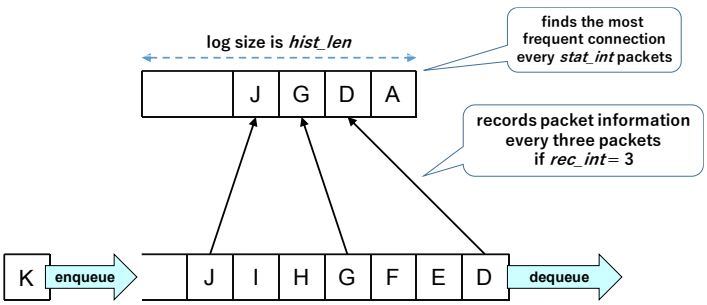

Fig. 8. proposed method (decision of the most bandwidth-consuming)

can control fairness, but cannot always provide efficient fairness without suitable tuning. On the other hand, the proposed dynamic method can provide good fairness independent of delay time. From Fig. 12, we can see that the best fairness is achieved with $p = 1/3$. In addition, we can see also that almost the best fairness is obtained with small p. Thus, we can say that the proposed method can provide almost the best fairness using small p without detailed tuning.

VI. CONCLUSION

In this paper, we have proposed a method for improving TCP fairness based on CoDel without an assumption that the most bandwidth-consuming connection is given. Our evaluation has demonstrated that the proposed method have been able to improve TCP fairness among modern TCP algorithms without large performance decline.

For future work, we plan to discuss a method for improving fairness without introducing a new parameter.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Numbers 24300034, 25280022, 26730040, 15H02696.

This work was supported by CREST, JST.

REFERENCES

[1] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in Proceedings of ACM SIG-COMM 2002, Aug. 2002.

[2] Jeonghoon Mo, Richard J. La, VenkatAnantharam, and Jean Walrand, "Analysis and comparison of TCP Reno and Vegas", in Proceedings of IEEE INFOCOM'99, March 1999.

[3] L. Xu, K. Harfoush and I. Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks," Proc. IEEE Info COM 2004, March 2004

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

[5] Kun Tan, Jingmin Song, Qian Zhang, and Murari Sridharan, "A Compound TCP Approach for High-speed and Long Distance Networks," Proc. of IEEE Info COM 2005, July 2005.

[6] Ryo Oura, Saneyasu Yamaguchi, "Fairness Comparisons Among Modern TCP Implementations," The 6th International Workshop on Telecommunication Networking, Applications and Systems (TeNAS 2012), Mar. 2012.

[7] Hasegawa G, Itaya Natsuki , Murata Masayuki, "The dynamic threthold control algorithm of RED for thousands of TCP flows," IEICE Technical Report , NS2001-11, 2001 (in Japanese)

[8] ITSUMI Hayato, YAMAMOTO Miki, "Improving Fairness between CUBIC and Compound TCP," IEICE Technical Report, vol. 110, no. 372, NS2010-160, pp. 103-108, 2010 (in Japanese)

[9] Masato Hanai, Saneyasu Yamaguchi, and Aki Kobayashi, "Improving TCP Fairness Between Modern TCP Algorithms Based on Controlling Delay," Workshop of 2016 IEEE 17th International Conference on High Performance Switching and Routing, Jun. 2016.

[10] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", IEEE Journal on Selected Areas in Communication, Vol.13, No.8, pp.1465-1480, October 1995.

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

[12] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, pp. 397.413, Aug. 1993.

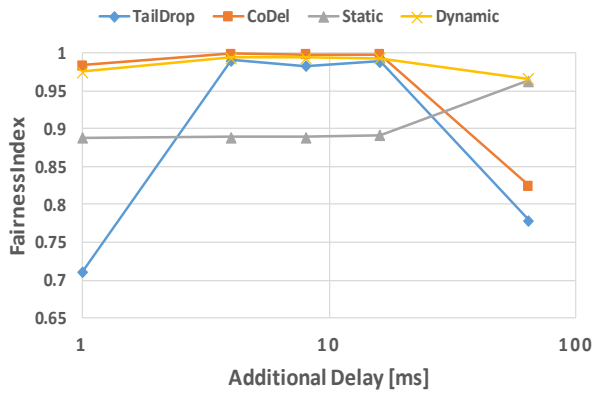

Fig. 9. Experimental results (Fairness Index)

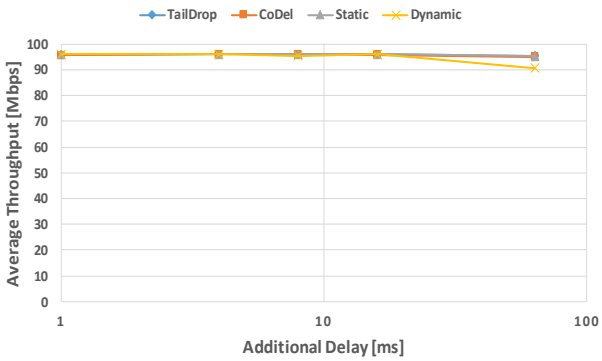

Fig. 10. Experimental results (Total throughput)

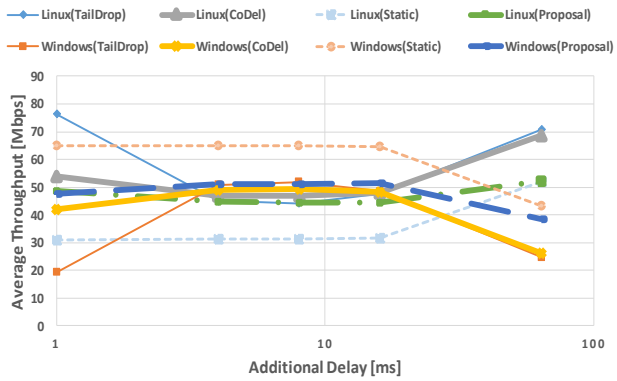

Fig. 11. Experimental results (Throughput of each OS)

[13] Yuria Akiyama, Tomoki Kozu, Saneyasu Yamaguchi, "Active packet dropping for improving performance fairness among modern TCPs," The 16th Asia-Pacific Network Operations and Management Symposium (APNOMS 2014), 2014

[14] J. Gettys, "Bufferbloat: Dark Buffers in the Internet," in IEEE Internet Computing, vol. 15, no. 3, pp. 96-96, May-June 2011.

[15] Kathleen Nichols and Van Jacobson. 2012. Controlling queue delay.Commun. ACM 55, 7 (July 2012), 42-50.

[16] iperf homepage, https://iperf.fr/

[17] R. Jain, D. Chiu and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared systems", DEC-TR-301, Tech. Rep., 1984

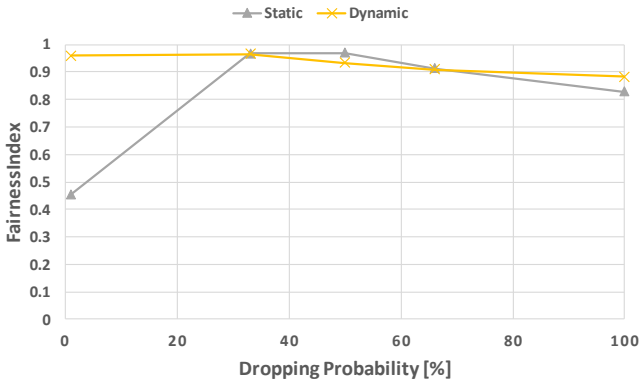

Fig. 12. Experimental results (Dropping probability and Fairnes Index)