# Performance Improvement of TCP Communication based on Cooperative Congestion Control in Android Terminals

Ayumi Shimada
Ochanomizu University
Tokyo, Japan
ayumi@ogl.is.ocha.ac.jp

Saneyasu Yamaguchi
Kogakuin University
Tokyo, Japan
sane@cc.kogakuin.ac.jp

Oguchi Masato
Ochanomizu University
Tokyo, Japan
oguchi@is.ocha.ac.jp

## ABSTRACT

Network congestion is a prevalent problem on the Internet. Modern loss-based TCP algorithms aim to increase throughput by using aggressive congestion window (CWND) control strategies. Due to such strategies, a large number of packets may be backlogged and eventually dropped at the entry point to a wireless access network. Our previous work proposed CUBIC-based CWND control middleware for Android terminals to solve the backlog issue. It utilizes Round Trip Time (RTT) as an indication of the TCP ACK backlog condition at the WLAN Access Point (AP), and controls the upper and lower bounds of its CWND size to suppress excessive transmissions of its own TCP DATA packets. However, the middleware causes the severe fairness dwindling due to the timing difference of the middleware of each node when to start the control. In this study, we propose a new mechanism which takes the fairness of the transmission into account. Our new method starts control with taking cooperativeness into account. Our evaluation demonstrated that our new method gained more cooperativeness and fairness over the previous one by comprehending the situation of all other competing devices in the network.

## CCS CONCEPTS

• **Networks → Transport protocols**; **Wireless access points, base stations and infrastructure**; **Traffic engineering algorithms**; **Network performance evaluation**;

## KEYWORDS

TCP, Congestion control, RTT, Android, Throughput, Fairness

## 1 INTRODUCTION

As the technology of smart devices (e.g. smartphones and tablets) advances and the bandwidths of networks become greater, loss-based TCPs have improved. Particularly, modern Loss-based TCPs such as BIC[17] and CUBIC[6] use aggressive CWND control strategies in order to gain better throughput over other competing TCP sessions. Due to such strategies, a large number of packets may be backlogged and eventually dropped at the entry point to a wireless access network, since a wireless link usually offers much narrower bandwidth than wired backhaul and backbone networks. This pitfall applies not only to downstream TCP sessions but also to upstream TCP sessions when the terminal is connected via a WLAN, which disregards packet size in CSMA/CA[11] based scheduling. Moreover, mobile traffic from handheld devices connecting WLAN is growing at an unprecedented rate. The ACK packet backlog problem is further exacerbated by device proliferation.

In our previous study [14] [8] [7], CUBIC-based CWND control middleware for Android terminals has been developed to solve the issue with upstream TCP sessions. It utilizes Round Trip Time (RTT) as an indication of TCP ACK backlog conditions at the WLAN Access Point (AP) and controls the upper and lower bounds of its CWND size to suppress excessive transmissions of its own TCP DATA packets. The middleware improves the aggregate throughput by avoiding ACK packet accumulation at AP. In our previous work, we evaluated the middleware using Nexus S phones running Android OS version 4.1 (Jelly Beans) as client terminals.

In our study of [15], we evaluated the method on a system composed of Nexus 7 tablets where OS version is 6.0.0 (Marshmallow) and Nexus 5 smartphones where OS version is 5.1.1(Lollipop) in addition to Nexus S. The evaluation demonstrated that the aggregate throughput improved on all kinds of devices by the method to avoid ACK packet accumulation at AP.

In our study, we introduce a new mechanism which takes the fairness of the transmission into account. In the previous method, the middleware in each device starts controlling the CWND size independently, when it detects the substantial increase of the RTT value of its own. The RTT values of all terminals might sometimes coincidentally increase substantially and the middleware of all terminals start controlling CWND size almost at the same time. The previous method is effective if this is the case. However, we found that it is highly possible that not all clients start their own CWND controlling by the previous mechanism because each device starts controlling CWND size only by detecting the increase of the RTT value of its own. In case of some terminals control their CWND and solve the backlog issue, the other terminals do not control and gains remarkably higher throughput. It means that the clients in the same network cannot share the bandwidth equally. The point is the existence of controlled and uncontrolled devices in the same network. The previous middleware improved aggregate throughput since only a part of devices occupy the bandwidth and get high throughput, while fairness is significantly decreased because other clients can use only small amount of the bandwidth. In general, the phenomenon of fairness declining occurs when only throughput is tried to be improved. In this paper, we investigate the cause

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*WOODSTOCK'97, July 1997, El Paso, Texas USA*
© 2016 Copyright held by the owner/author(s).
ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4


**Figure 1: Behavior of CDUBIC-TCP**

of the fairness dwindling and propose a new mechanism which takes the fairness of the transmission into account. Specifically, we show the severe faieness to be caused by how the middleware decides to start the control. Our new proposed mechanism starts controlling CWND size of all terminals connected to the same AP when it detects significant RTT increase of at least one client. Thus, by taking cooperativeness into account for the control start condition, we attempted to improve the fairness between the clients. By comprehending the situation of all other competing devices in the network, our method gained more cooperativeness and fairness over previous one. We then show the effectiveness of the method on heterogeneous terminals, Nexus 5 and Nexus 7.

## 2 BACKGROUND

### 2.1 Android OS

This study focuses on the implementation of our proposed mechanism as middleware for the Android platform. The Android OS is a platform for mobile terminals whose development is led by Google and is distributed as a package that includes an Operating System, middleware and a set of applications. The source code of the Android is available via the Android Open Source Project by the Open Handset Alliance [9]. Thus we can apply our proposed method to any other Android terminal with minor modification.

Please note that Android is built based on the Linux kernel, which provides basic capabilities such as multistack networking, multitasking, virtual-memory management and virtual machines. Therefore, this work can be applied to any other Linux-based terminal or system, although the performance evaluation has only been conducted with Android terminals.

Since the default TCP version in Linux is CUBIC, Android adopted CUBIC as the transport protocol. CUBIC, like many other transport protocols, controls the rate of data packet transmissions based on CWND, the maximum number of packets that can be transmitted without receiving an ACK packet from the data packet receiver. Setting an appropriate CWND is the key to achieving high throughput, which is the primary difference between versions of TCP.

$$W_{cubic} = C(t - \sqrt[3]{\frac{W_{max}}{C}})^3 + W_{max} \quad (1)$$

Figure 1, which is captured in our experimental system, shows the behavior of CUBIC. CWND size is calculated with Expression 1. CWND is increased gradually upon receiving an ACK and halved every time a packet loss is experienced. Since the CWND size is reduced upon a packet loss, it is called a loss-based TCP. CUBIC also has a unique feature that changes its CWND with the passage

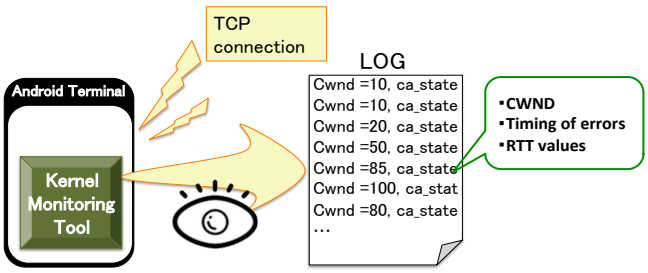



**Figure 2: Summary of Kernel monitoring tool**

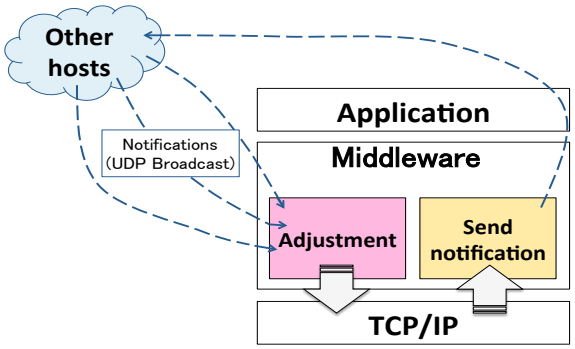



**Figure 3: Summary of Middleware**

of time, which is not seen in other loss-based TCP. Further, a delay-based TCP [12] reduces the CWND size based on the RTTs (e.g. TCP Vegas, TCP Westwood).

### 2.2 Kernel Monitoring Tool

The Kernel Monitoring Tool is a system tool that can observe the behavior of parameters inside the Linux Kernel code, which is generally never observed. Our previous work successfully embedded it in the Android platform in order to analyze the connection status of a mobile host in real time [14]. As shown in Figure 2, it allows users to monitor parameters in the kernel processing at the mobile host, which include CWND, RTT, and timing of errors.

### 2.3 CWND-controlling Middleware

This subsection describes the CWND-controlling middleware (Figure 3) that was implemented in our previous work [8][7]. This middleware was developed based on the Kernel Monitoring Tool so that it can access the current values of the parameters in the kernel memory space. It controls CWND size when the Android terminal is connected to the server via a WLAN and congestion is detected. Congestion occurs when other terminals that share the same AP begins communication and the RTT values substantially increase as a result. CWND control is done by setting both the upper and lower limit of CWND size based on the length of backlog in AP and the number of communication terminals using the *proc* interface.

A flowchart of the middleware is given in Figure 4. The process flow of the proposed system is as follows.

(1) Start controlling the CWND size if a substantial increase of the RTT value is detected during TCP communication

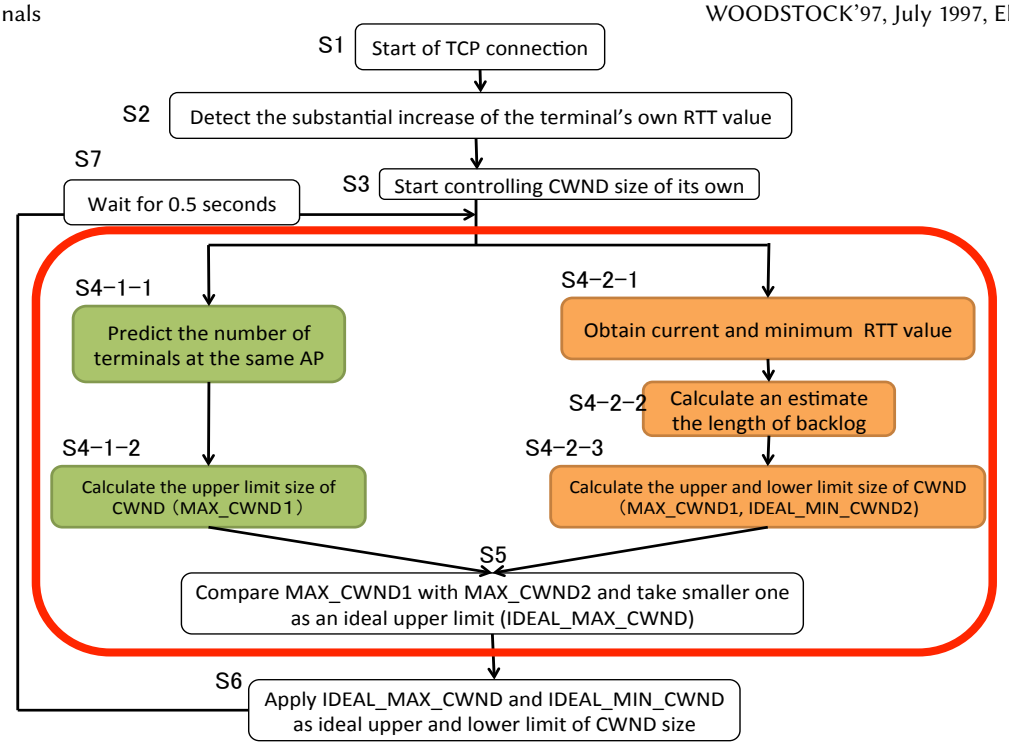



**Figure 4: Flowchart of the previously implemented middleware**

(2) Calculate the ideal limit of the upper and lower CWND size.
   (a) The number of terminals using proc interface is counted by broadcasting User Datagram Protocol (UDP) packets to each terminal connected to the same AP.
   (b) First candidate of the upper limit size (MAX CWND 1) is set based on the number of terminals according to Expression 2.
   (c) Obtain current RTT value from Kernel Monitoring Tool.
   (d) Calculate an estimate of the length of the backlog in the AP based on the acquired RTT values.
   (e) Second candidate of the upper limit size (MAX CWND 2) is set according to the RTT value in accordance with TABLE 1.

(3) Compare MAX CWND 1 with MAX CWND 2 and select the smaller one as an ideal upper limit size for CWND (IDEAL MAX CWND).

(4) Apply IDEAL MAX CWND as an ideal upper limit of CWND size.

(5) Repeat the process from (2) to (4) and adjust every 0.5 seconds.

**Table 1: Approach of CWND adjustment**

| RTT | 0- | 51- | 106- | 161- | 331- |
|---|---|---|---|---|---|
| max_cwnd2 | 555 | 100 | 10 | 8 | 1 |

$$Ideal\_cwnd = \frac{Bandwidth[Mbps] \times RTT[sec]}{Segmentsize(1.5Kbyte) \times NumberOfTerminals} \quad (2)$$

Using this method, the control system can limit the quantity of traffic outbreak. We did not modify the congestion control algorithm itself in the middleware so that the basic TCP function can maintain interoperability. Nevertheless, the communication has been optimized by setting the levels of the upper and lower limits for the CWND, and the congestion control is adjusted based on the communication situation in the AP surroundings.

Our previous work used Android Nexus S smartphones as client terminals. Since Android is still being substantially improved and upgraded either in terms of features or firmware, we introduced the newer model and OS version of Android devices such as Nexus 5 smartphones and Nexus 7 tablets (See Table2). In this paper, we propose a new method taking account of fairness that is effective on heterogeneous terminals.

## 3 BASIC TEST

In the case of a wired network, the TCP initially assumes that a packet drop is an indication of network congestion, since the primary reason for a packet to be dropped is queuing overflow at one of the routers along the path to the other communication peer. However, wireless communications introduce other causes for packet drops such as fading, collisions and interference, which confuse the TCP CWND control algorithm and lead to suboptimal performance. The effects of wireless communications on TCP performance and

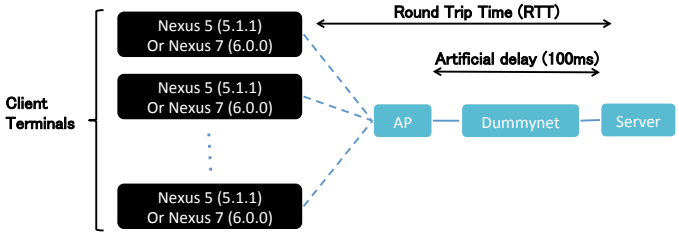



**Figure 5: Experimental topology**

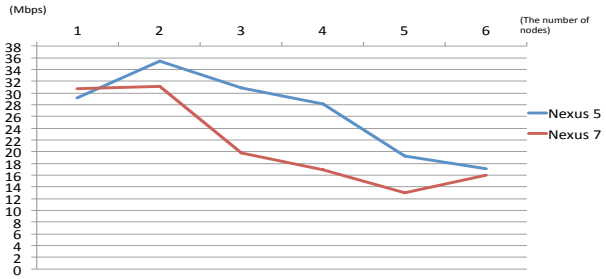



**Figure 6: Relation between number of terminals and total throughput**

the techniques to address these have been extensively studied [16] [2] [5] [10] in the literature.

Our previous work revealed that total throughput rapidly decreases when the number of terminals around the same AP increases. This is because a buffer overflow may occur in the AP and the AP starts dropping packet.

In this section, we created a basic test to prove this phenomenon. We evaluated device performance in a WLAN network with an experimental system, as shown in Figure 5. The client terminals were connected to an AP over an 802.11g network, and the AP was connected to the server host through a wired route. As client terminals, we used Nexus 5 smartphones and Nexus 7 tablets. The number of terminals ranged from 1 to 6. To emulate network delay and packet loss, a network emulator, Dummynet[1], was inserted between the AP and the server host; 100 ms delay was set by assuming a high-delay environment. The wired parts were connected with higher rate because of the Gigabit Ethernet, whereas the bandwidth was only about 20Mbps in the wireless environment. As a network benchmark tool, Iperf for Android [4] was installed on all the terminals. The specification of each device used in the experiment is shown in Table 2.

**Table 2: Specications of devices**

| | | | |
|---|---|---|---|
| Android | Model number | Nexus 5 | Nexus 7 |
| | Firmware version | 5.1.1 | 6.0.0 |
| | Kernel version | 3.4.0-geaa8415-dirty | 3.4.0-g272b14b-dirty |
| | Build number | LMY49H | MRA58K |
| server | OS | Ubuntu 14.04 (64bit) / Linux 3.13.0 | |
| | CPU | Intel(R) Core(TM)2 Quad CPU Q8400 | |
| | Main Memory | 8.1GiB | |
| AP | Model | MZK-MF300N(Planex) | |
| | Communication system | IEEE 802.11g | |

Figure 6 represents the relation between the number of terminals and total throughput. The red and blue lines show the results of the Nexus 5 smartphones and Nexus 7 tablets, respectively. As depicted by both lines, when the number of terminals approaches 3 and the AP become overloaded, the throughput starts to degrade. The radio transmission sections between the AP and the terminals were bottleneck and severely restricted the overall throughput. To solve this problem proposed middleware has been developed.

## 4 PERFORMANCE EVALUATION OF PREVIOUS IMPLEMENTATION

### 4.1 Overview of Experiments

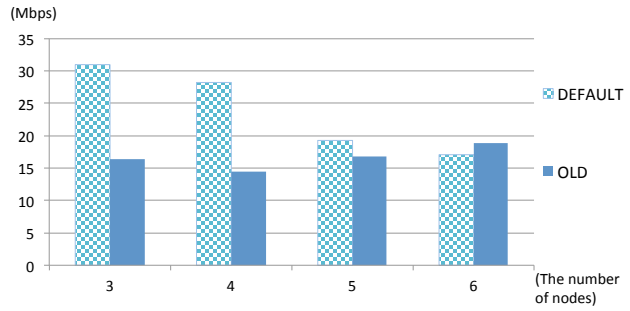
To evaluate the middleware that was developed in the previous study, we installed it both on smartphones (Nexus 5) and tablets (Nexus 7). Using the same experimental topology as Figure5, we observed the behavior of the terminals and evaluated the effect of the previously implemented middleware. The number of terminals ranged from 3 to 6.

### 4.2 Experimental Results

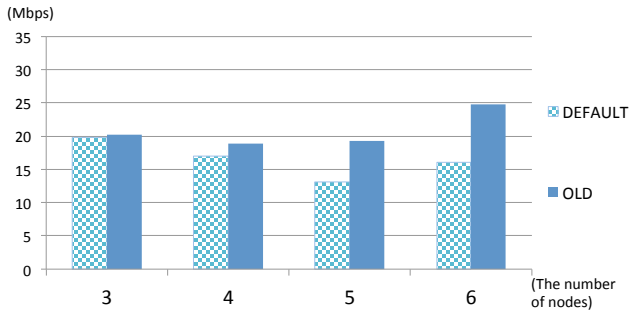
Figure 7 shows the relation between the number of terminals and total throughput observed on Nexus 5 smartphones and Nexus 7 tablets, respectively. The light blue bars (DEFAULT) show the results without the middleware and the bright blue bars (OLD) show the results when running previously proposed middleware. As depicted in Figure 7, when 3 to 5 Nexus 5 smartphones were connected to the same AP and communicating at the same time, the middleware turned out to be counterproductive. Although the middleware made the total throughput a little bit higher when six terminals were communicating, it is still much lower than the value of 3 and 4 terminals communicating without middleware. It means that the previous middleware did not solve the throughput declining phenomenon that occurs when the number of terminals around the same AP increases.

In contrast, the middleware enhanced the throughput of the Nexus 7 tablets communicating on any number of the devices.

Next, we evaluated the communication performance using the Fairness Index [3] to confirm that the available bandwidth is fairly assigned to each terminal. The Fairness Index indicates the equitableness, and a calculated value of 1 corresponds to the highest fairness. Figure 8 shows the relation between the number of terminals and the fairness observed on Nexus 5 smartphones and Nexus 7 tablets, respectively. The light pink bars (DEFAULT) show the results without the middleware and the bright red bars (OLD) show the results when running previously implemented middleware. As shown in Figure 8, the fairness was improved by means of the middleware when 4 and 5 Nexus 5 phones were communicating. However, when 5 terminals are communicating with the mechanism, the fairness is not high enough compared with the results of

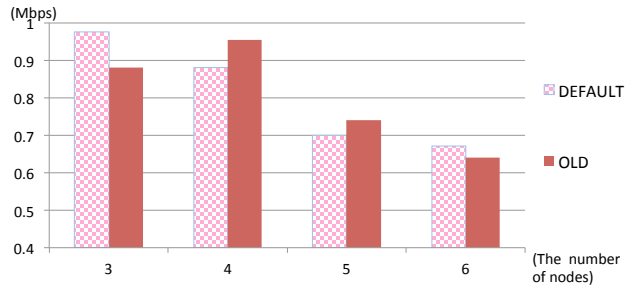



Nexus 5 smartphones

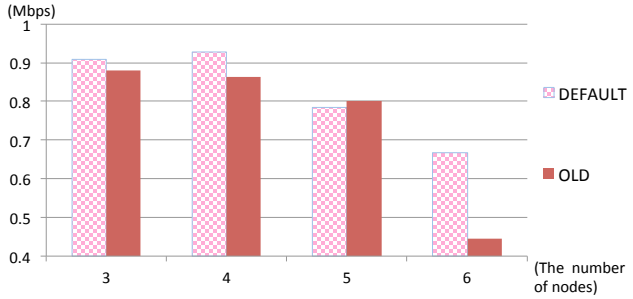



Nexus 7 tablets

**Figure 7: Relation between the number of terminals and aggregate throughput**




Nexus 5 smartphones




Nexus 7 tablets

**Figure 8: Relation between the number of terminals and fairness**

3 and 4 terminals. Similar results could observed on Nexus 7 as well. Moreover, what is common on both results of Nexus 5 and Nexus 7 is that the fairness is quite low when 6 terminals are communicating, especially when the previously implemented middleware was running. In other words, the middleware has an adverse effect not only for throughput but also for the fairness.

$$FairnessIndex: f_i = \frac{(\sum_{i=1}^{k} x_i)^2}{k \sum_{i=1}^{k} x_i^2} (1 \leq i \leq k) \quad (3)$$

# 5 PROPOSED MECHANISM

## 5.1 Problem of the previous implementation

As described in Subsection 2.3, the mechanism starts controlling CWND size when the RTT value of its own terminal substantially increased. As shown in Figure 9, only a part of terminal devices (node 1 - 5) are strictly controlled, whereas the other device (node 6) is not controlled and occupy the bandwidth. Thus, one problem (hereinafter called PROBLEM 1) is that there is a possibility for the existence of controlled and uncontrolled devices in the same network because the middleware assesses the timing to start controlling the CWND size only when detecting an increase in its own RTT value. As a result, we revealed that this problem causes the depression of the fairness. Furthermore, as you can see in Figure 10, the CWND size which is controlled by the middleware is substantially small. It means the bandwidth remained to be used because of the strictness of the control. This could be the second problem (hereinafter called PROBLEM 2) that causes low total throughput.

## 5.2 Detail of the Improvement of the mechanism

To solve the problems found in the previous middleware, we improved the method to take account of the fairness. First of all, to prevent the PROBLEM 1, we changed the condition when the system starts controlling CWND size. Figure 13 shows the system flowchart after the improvement. The modified parts are STEP 2 (S2) and STEP 3 (S3). In the previous method, a terminal, in which the middleware is installed, starts controlling itself, when it detects the substantial increase of the RTT value of its own. By contrast, the proposed mechanism starts controlling CWND size of all terminals connected to the same AP when it detects at least one substantial RTT increase in the terminals around the AP. By giving Cooperativeness to the control start condition, we attempted to improve the fairness between the clients. We call this method Proposed mechanism 1. To collect the RTT values of all terminals in the same network, the terminals broadcast RTT information of their own by UDP.

On top of that, we improved the middleware to solve PROBLEM 2. As shown in Table 1 in Subsection 2.3, the middleware controlles the CWND size based on the RTT diff value. Since this table parameter might be too strict, we made those parameters bigger to share the bandwidth with all clients efficiency. The updated parameters are shown in table 3. We call this modifing Proposed mechanism 2.

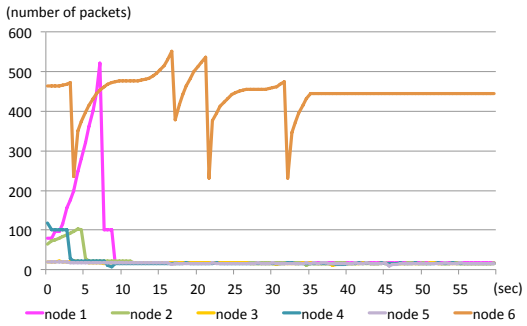



**Figure 9: Behavior of the CWND controlled by previous mechanism (Nexus 7 tablets)**

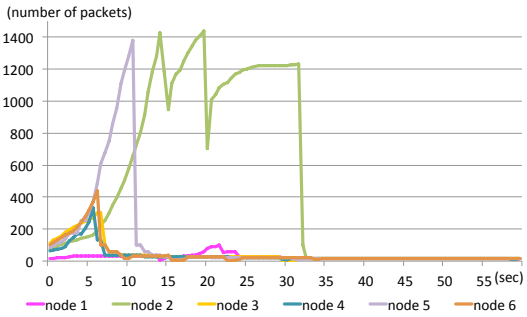



**Figure 10: Behavior of the CWND controlled by previous mechanism (Nexus 5 smartphones)**

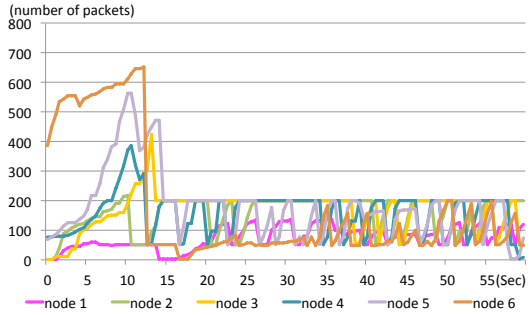



**Figure 11: Behavior of the CWND controlled by proposed mechanism 1 (Nexus 5 smartphones)**

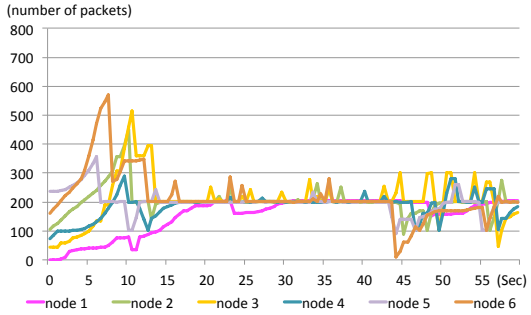



**Figure 12: Behavior of the CWND controlled by proposed mechanism 2 (Nexus 5 smartphones)**

Using the proposed mechanism 1 and 2, we again performed tests under the same environmental topology as Figure 5.

**Table 3: Approach of CWND adjustment**

| ratio_rtt | before modifying | | after modifying | |
|---|---|---|---|---|
| | max | min | max | min |
| 0- | 555 | 200 | 1000 | 200 |
| 51- | 200 | 50 | 700 | 150 |
| 106- | 100 | 10 | 400 | 100 |
| 161- | 10 | 2 | 200 | 10 |
| 331- | 1 | 1 | 50 | 1 |

## 5.3 Performance Evaluation of the proposed mechanism

*5.3.1 Behavior of the proposed mechanism.* Figure 11 and Figure 12 show the transition of CWND size controlled by the proposed method 1 and 2, respectively. As described in the Figures, the bandwidth is shared fairly among the clients because the control start timing is synchronized. Note there is a bit of a discrepancy between each of the control start timings because the RTT value notifications are broadcasted by UDP. In addition, the CWND size of Figure 12 is bigger (Some lines reached 300) than the one of Figure 11 by means of proposed method 2. Although the results of Nexus 7 is skipped, such effectiveness was shown also on the teblets.

*5.3.2 Correlation between throughput and fairness.* As we mentioned before, the important feature of our approach is to improving the fairness without reducing the aggregate throughput. To evaluate it, we present the aggregate throughput and the fairness values of Nexus 5 and Nexus 7 as a scatter plot in Figure 14 and 15, respectively. The vertical axis shows the fairness, and the horizontal axis shows the aggregate throughput. The red, blue, green and orange marks represent the results when 3, 4, 5 and 6 terminals are communicating at the same time, respectively. We performed tests in the same environment as Figure 5 without any middleware (hereinafter called DEFAULT), with the previously implemented system (hereinafter called OLD), with the proposed mechanism 1 (hereinafter called NEW1) and with the proposed mechanism 2 (hereinafter called NEW2).

As shown in Figure 14, NEW1 and NEW2 reach higher throughput and fairness than DEFAULT and OLD. Conversely, the result of OLD values quite low throughput with any number of terminals. For example, the spot of 4 terminals communicating obtained using the previously implemented middleware reaches quite high fairness because the CWND of all terminals was controlled in the almost same value. However, the total throughput is quite low because the controlled CWND size given to each device was too small and

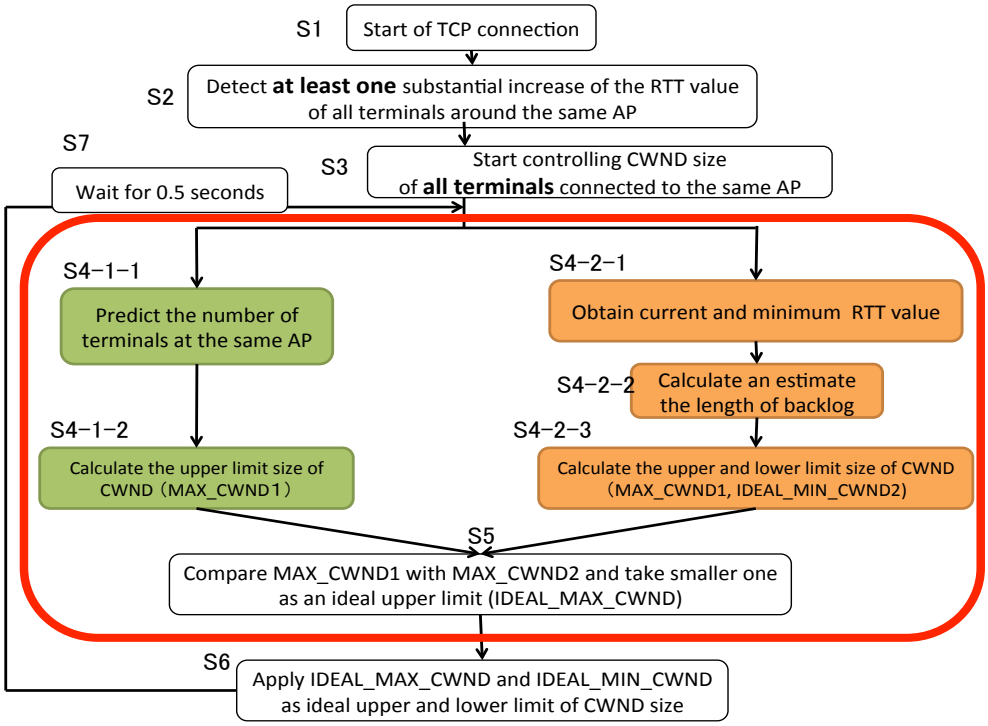



**Figure 13: Flowchart of the proposed middleware**

the bandwidth remained to be used as a result. Such problem has been solved by the change of parameters mentioned in Subsection 5.2. The another example is that the spots of DEFAULT of 5 and 6 terminals appear on bottom left in the Figure 14. In other words, both fairness and throughput degrade when many (5 or 6) terminals are communicating at the same time. The previously implemented middleware could not improve those values, whereas our middleware (NEW1 and 2) successfully has made both values significantly high as you can see top right in Figure 14.

As shown Figure 15, NEW1 and NEW2 reach higher throughput and fairness than DEFAULT and OLD also on Nexus 7 tablets. In addition, degrading both of fairness and throughput when many (5 or 6) terminals are communicating in default mode is the problem on Nexus 7 as well. Although old middleware improved the throughput of six terminals communicating, the fairness is terribly low because of the existence of controlled and uncontrolled devices in the same network . Some terminals, the previous middlware of which was not running, continued their default transmission and occupied too much bandwidth. NEW1 solved this issue by changing the control start condition as mentioned in Subsection 5.2. Whereas, since the fairness of NEW2 when six terminals were communicating became worse than NEW1, we need to consider the parameters mentioned in Subsection 2.3 further.

## 6 CONCLUSIONS

This study has focused on the ACK packet backlog problem with upstream TCP sessions and has proposed CUBIC-based CWND-controlling middleware taking both throughput and fairness into account. It controls the upper and the lower bounds of its CWND size to suppress excessive transmissions of its own TCP DATA packets. We installed the middleware on Nexus5 and Nexus7 heterogeneous terminals. By making the control start condition takes cooperativeness into account, we attempted to improve the fairness between the clients.

As a result, the proposed middleware yielded substantially high fairness and throughput. By comprehending the situation of all other competing devices in the network, our method gained more cooperativeness and fairness over previous one.

We are set to make more tests in an environment with a mix of Nexus 5 and Nexus 7 devices.

## ACKNOWLEDGMENTS

This work was partly supported by JST CREST Grant Number JPMJCR1503, Japan.

## REFERENCES

[1] Massimo Mario Augello. 1996. The dummynet project. (Feb. 1996). Retrieved February 14, 2017 from http://info.iet.unipi.it/~luigi/dummynet
[2] Claudio Casetti, Mario Gerla, Saverio Mascolo, M. Y. Sanadidi, and Ren Wang. 2002. TCP Westwood: End-to-end Congestion Control for Wired/Wireless Networks. *Wirel. Netw.* 8, 5 (Sept. 2002), 467–479. https://doi.org/10.1023/A:1016590112381
[3] Dah-Ming Chiu and Raj Jain. 1989. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems* 17, 1 (1989), 1–14.
[4] Technion CIO. [n. d.]. Iperf For Android Project in Distributed Systems. (Sept. [n. d.]). Retrieved September 10, 2011 from http://www.cs.technion.ac.il/~sakogan/DSL/2011/projects/iperf/index.html
[5] Luigi A. Grieco and Saverio Mascolo. 2004. Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control. *SIGCOMM*

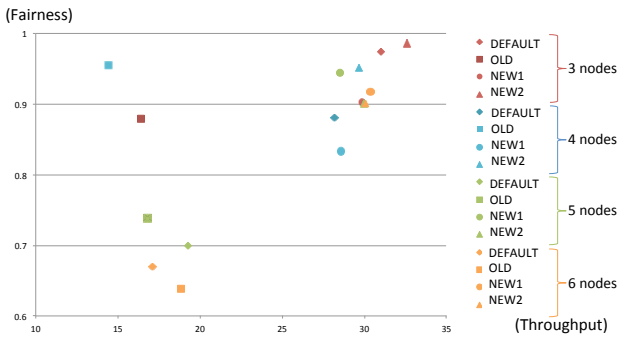



**Figure 14: Scatter plot of the total throughput and fairness of Nexus 5 achieved by the default transmission, the previous implemented middleware, the proposed mechanism 1 and 2. The vertical axis shows the fairness, and the horizontal axis shows the aggregate throughput.**

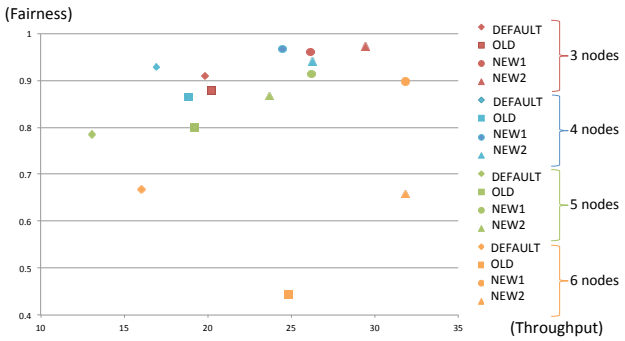



**Figure 15: Scatter plot of the total throughput and fairness of Nexus 7 achieved by the default transmission, the previous implemented middleware, the proposed mechanism 1 and 2. The vertical axis shows the fairness, and the horizontal axis shows the aggregate throughput.**

*Comput. Commun. Rev.* 34, 2 (April 2004), 25–38. https://doi.org/10.1145/997150.997155

[6] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review* 42 (2008), 64–74.

[7] Ai Hayakawa, Saneyasu Yamaguchi, and Masato Oguchi. 2015. Reducing the TCP ACK Packet Backlog at the WLAN Access Point. In *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication (IMCOM '15)*. ACM, New York, NY, USA, Article 37, 8 pages. https://doi.org/10.1145/2701126.2701164

[8] Hiromi Hirai, Saneyasu Yamaguchi, and Masato Oguchi. 2013. A Proposal on Cooperative Transmission Control Middleware on a Smartphone in a WLAN Environment. In *Proc. IEEE WiMob2013*. IEEE, 701–717. http://ieeexplore.ieee.org/document/6673432/

[9] MarkMonitor Inc. 1997. Android open source project. (Aug. 1997). Retrieved August 9, 2017 from http://source.android.com

[10] Shao Liu, Tamer Başar, and R. Srikant. 2006. TCP-Illinois: A Loss and Delay-based Congestion Control Algorithm for High-speed Networks. In *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools (valuetools '06)*. ACM, New York, NY, USA, Article 55. https://doi.org/10.1145/1190095.1190166

[11] Yanchang Liu, Shaohai Hu, Yang Xiao, Xiaoli Liu, Guangzhi Qu, , and Kiseon Kim. 2010. CSMA/CA-based MAC protocol in Cognitive Radio network. In *IET 3rd International Conference on Wireless, Mobile and Multimedia Networks*. IET. https://doi.org/10.1049/cp.2010.0643

[12] Saverio Mascolo, Claudio Casetti, Mario Gerla, M. Y. Sanadidi, and Ren Wang. 2001. TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)*. ACM, New York, NY, USA, 287–297. https://doi.org/10.1145/381677.381704

[13] Makiko Matsumoto and Masato Oguchi. 2012. Multiple access in MAC layer based on surrounding conditions of wireless stations. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean*. IEEE, 133–140.

[14] Kaori Miki, Saneyasu Yamaguchi, and Masato Oguchi. 2011. Kernel Monitor of Transport Layer Developed for Android Working on Mobile Phone Terminals. In *Proc. the Tenth International Conference on Networks*. ICN, 297–302. https://doi.org/10.1109/WiMOB.2013.6673432

[15] Ayumi Shimada, Masato Oguchi, Saneyasu Yamaguchi, Heidi Kaartinen, Marjo Heikkilä, and Joni Jämsä. 2016. Performance Improvement in WLAN and LTE Based on Backlog Control Middleware. In *Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services (MOBIQUITOUS 2016)*. ACM, New York, NY, USA, 201–206. https://doi.org/10.1145/3004010.3004029

[16] Prasun Sinha, Thyagarajan Nandagopal, Narayanan Venkitaraman, Raghupathy Sivakumar, and Vaduvur Bharghavan. 2002. WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks* 8, 2/3 (2002), 301–316.

[17] L. Xu, K. Harfoush, and I. Rhee. 2003. *Binary Increase Congestion Control for Fast, Long Distance Networks*. Proceedings of Tech. Report. NC State University.