Performance Improvement in WLAN and LTE Based on Backlog Control Middleware

Ayumi Shimada, Masato Oguchi
Department of Information Sciences
Ochanomizu University
Tokyo, Japan
ayumi@ogl.is.ocha.ac.jp
oguchi@is.ocha.ac.jp

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

Heidi Kaartinen, Marjo Heikkilä, Joni Jämsä
Centria Research and Development
Centria University of Applied Sciences
Ylivieska, Finland
heidi.kaartinen@centria.fi
marjo.heikkila@centria.fi
joni.jamsa@centria.fi

ABSTRACT

Smartphones have highly functional operating systems similar to PCs. Their communication throughput depends on behavior of Transmission Control Protocol (TCP). Modern loss-based TCP algorithms take aggressive congestion window (CWND) control strategies in order to gain better throughput, but such strategies may cause a large number of packets to be backlogged and eventually dropped at the entry point to the wireless access network. This problem applies not only to the downstream TCP sessions but also to the upstream TCP sessions when the terminal is connected via a wireless network, such as Wireless Local Area Network (WLAN) and Long Term Evaluation (LTE) network, which disregards the size of packets in its scheduling. This paper focuses on the ACK packet backlog problem with the upstream TCP sessions, and proposes a CUBIC based CWND control mechanism as part of the middleware for the Android terminals. It utilizes the Round Trip Time (RTT) as an indication for the TCP ACK backlog condition at the WLAN AP and LTE base station, and controls the upper and lower bounds of its CWND size to suppress excessive transmissions of own TCP DATA packets. Our experimental study with up to seven Android terminals shows that the proposed mechanism can improve both aggregate throughput and fairness of the WLAN. In addition, our evaluation on LTE network demonstrates that the method suitably controls congestion and communication delay also on LTE network.

MOBIQUITOUS '16 Adjunct Proceedings, November 28-December 01, 2016, Hiroshima, Japan
© 2016 ACM. ISBN 978-1-4503-4759-4/16/11...$15.00
DOI: http://dx.doi.org/10.1145/3004010.3004029

CCS Concepts

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

Keywords

TCP, congestion control, CWND, RTT, Android

1. INTRODUCTION

As performance of terminals is improved and a bandwidth of a network becomes higher, modern loss-based TCPs such as BIC [1] and CUBIC [2] take aggressive CWND control strategies in order to gain better throughput over other competing TCP sessions. Although such strategies are suitable for wired connections, they may cause a large number of packets to be backlogged and eventually dropped at the entry point to the wireless access network. This is because a wireless link can usually offer much narrower bandwidth than its wired backhaul and backbone networks. This problem applies not only to the downstream TCP sessions but also to the upstream TCP sessions when the terminal is connected via a WLAN, which disregards the size of packets in its CSMA/CA [3] based scheduling. This problem depends on performance of terminals and type of networks. Therefore, this paper focuses on the ACK packet backlog problem with the upstream TCP sessions in several environments, and proposes a CUBIC based CWND control mechanism as part of the middleware that can be implemented in the Android terminals. It utilizes the RTT as an indication for the TCP ACK backlog condition at the WLAN AP, and controls the upper and lower bounds of its CWND size to suppress excessive transmissions of own TCP DATA packets. An experimental study with up to seven Android terminals shows that the proposed mechanism can improve aggregate throughput of the WLAN, and that it is highly effective particularly for cases where very long RTTs are observed.

2. RELATED WORK

In the case of a wired network, TCP has originally assumed that a packet drop is an indication of network congestion, since the primary reason for a packet to be dropped is the 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 to lead to suboptimal performance. Such effects of wireless communications on the TCP performance as well as techniques to combat those have been extensively studied [4] [5] [6] [7] in the literature.

We also have studied intensively about this problem in various environments. Our previous work [8] [9] is one of them, in which each WLAN terminal attempts to estimate the number of neighboring terminals that operate on the same channel by monitoring broadcast activities, and adjusts its CWND size accordingly to gain its fair share. Details of this work are described in section 3.3. This paper presents the results of the behavior of Android terminals in LTE networks.

3. BACKGROUND

3.1 Android OS

This study focuses on the implementation of our proposed mechanism as a middleware of the Android platform. Android is a platform for mobile terminals whose development is led by Google, and is distributed as a package that includes an Operating System and basic applications. The source code of Android is available via the Android Open Source Project by the Open Handset Alliance [10]. Thus we can apply our proposed method to any other Android terminals 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 terminals or systems although the performance evaluation has been conducted only with Android terminals.

Since the default TCP version in Linux is CUBIC, Android adopts CUBIC as the transport protocol. CUBIC, like any 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 various versions of TCP.

In CUBIC, CWND is increased gradually per receiving an ACK and halved every time a packet loss is experienced as shown in Figure.1. This figure is captured in our experimental system. As the CWND size is reduced upon a packet loss, it is called a loss-based TCP. Other CWND control mechanisms used in TCP Vegas and TCP Westwood are based on the observed RTTs, and are called a delay-based TCP [11]. Cubic also has a unique feature that changes its CWND with passing time, which is not seen in other loss-based TCP.

3.2 Kernel Monitoring Tool

We have developed a method to observe a behavior of parameters inside the Linux Kernel code. Our previous work successfully embedded a system tool called Kernel Monitoring Tool in the Android platform in order to analyze the connection status of a mobile host [12]. 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. They are defined in the TCP implementation of the Linux Kernel code, and applications in the user space can generally never observe or even recognize them. By means of Kernel Monitoring Tool, our middleware can access the current values of these parameters in the kernel memory space

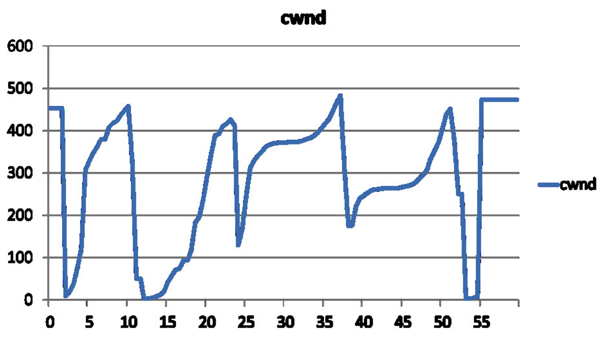

Figure 1: Behavior of CUBIC TCP

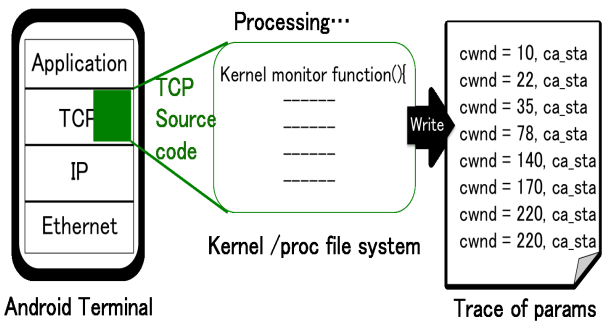

Figure 2: Kernel Monitoring Tool

3.3 CWND-Controlling Middleware

This subsection describes the CWND-controlling middleware that has been implemented in our previous work [8] [9]. This middleware has been developed based on Kernel Monitoring Tool(Section 3.2). It is responsible for controlling middleware to determine CWND size when the Android terminal is connected to the server via a WLAN or LTE network and congestion is detected. Congestion happens when another terminal that shares the same AP begins communication and the RTT values suddenly increase as a result. CWND control is done by setting levels of upper limit for the CWND based on the length of backlog in AP and the number of communication terminals using the *proc* interface.

First, an estimate of the length of backlog in AP is obtained from RTT value which is monitored by Kernel Monitoring Tool. Based on the acquired value, RTT Ratio (ratio_rtt) is calculated with Expression.1, which indicates increase and decrease of RTT. A value of minimum RTT (min_rtt) is updated by overwriting with the smallest RTT during the communication. After calculating the RTT ratio value, upper limit size called MAX_CWND1 is set according to RTT ratio in accordance with Table.1. The bigger RTT ratio becomes, the smaller MAX_CWND1 becomes.

$$ratio\_rtt = \frac{current\_rtt}{min\_rtt} \quad (1)$$

Table 1: Approach of CWND ajustment

ratio_rtt	1	2	3	4	5	6	7	8	9	1
max_cwnd1	330	150	100	50	10	10	10	10	10	1

Next, broadcasting User Datagram Protocol (UDP) packets to each terminals connected to the same AP, the number of terminals using *proc* interface is counted. The notification of UDP is broadcasted every 0.3 seconds because the kernel parameters frequently change. By contrast, the adjustment by middleware is executed every 10 seconds because the number of mobile hosts changes less often, and this lower frequency is sufficient to collect information from all hosts, considering the interval and the arrival rate of the notifications. After obtaining the number of terminals, upper limit size called MAX_CWND2 is set based on the number of terminals according to Expression.2. The more the number of terminals increase, the smaller the MAX_CWND2 becomes.

$$MAX\_CWND2 = \frac{Bandwidth[Mbps] \times RTT[sec]}{Segmentsize(1.5Kbyte) \times NumberofTerminals} \quad (2)$$

Finally, this system compares MAX_CWND1 with MAX_CWND2 and applies smaller one as ideal upper limit size of CWND. Using this method, the control system can limit the quantity of traffic outbreak and share the bandwidth of an AP.

In this middleware, we do not modify the congestion control algorithm itself of the basic TCP, which functions similar to the default case and should be good for the interoperability. Nevertheless, the communication is optimized by setting the levels of upper limit for the CWND, and the congestion control is adjusted based on the communication situation of AP surroundings. In this paper, we prove our method works well in various environments. Particularly, it is possible to adapt our method to different network environments, such as WLAN and LTE-networks.

4. TESTING AT WLAN NETWORK

We evaluated basic performance in WLAN network with an experimental system as presented in Figure.3. 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 7 tablets. The number of terminals ranged from 1 to 7. To emulate network delay and packet loss, a network emulator, Dummynet [13], was inserted between the AP and the server host; 256 ms delay was set by assuming a high-delay environment. The wired parts were connected with higher rate because of Gigabit Ethernet, whereas the bandwidth was only about 20Mbps in the wireless environment. Thus, the radio transmission sections between the AP and the terminals were a bottleneck, which is a typical case when mobile terminals access to a remote server through a wireless network. Especially, when the number of the terminals connected simultaneously increases, a buffer overflow may occur in the AP and the AP starts dropping packet. As a network benchmark tool, Iperf for Android [14] was installed on all the terminals.

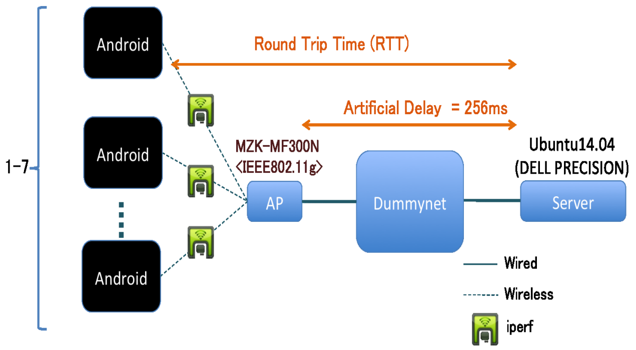

Figure 3: Experimental topology 1

Figure.4 shows the behavior of CWND and Figure.5 shows RTT value when 1, 3, 5 and 7 terminals communicate. Dotted lines show the CWND of the communication without middleware and solid lines show the CWND of the communication when middleware was used. When the terminals communicated without middleware, the RTT value, which is the end-to-end delay time, significantly increased in the case of 5 and 7 terminals. By contrast, the middleware suppresses the size of CWND. Therefore, when the terminals communicated with middleware, the RTT increase was suppressed.

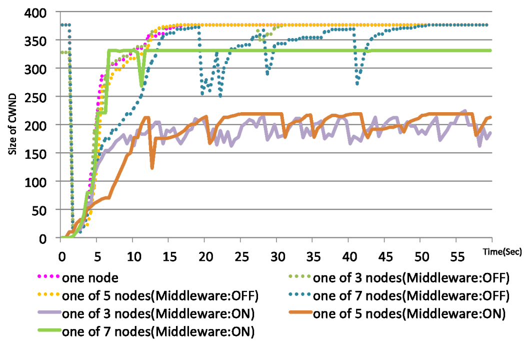

Figure 4: Time transition of the CWND

Figure.6 shows the relation between number of terminals and total throughput. The blue and red lines show the throughput without and with the middleware, respectively. As depicted by blue line, when the number of terminals approached 5 and the AP became overloaded, the throughput without middleware started to degrade. In contrast, the red line shows that the throughput reaches a stable level, which demonstratesan advantageous effect of the middleware. The performance with 7 terminals improved by approximately 1.25 times.

The middleware was also tested on Nexus S phones. Figure.7 shows the relation between number of terminals and total throughput in an environment same as the Nexus 7 experiment. The blue and red lines show the throughput without and with the middleware, respectively. The performance with 6 terminals improved by approximately 1.74 times.

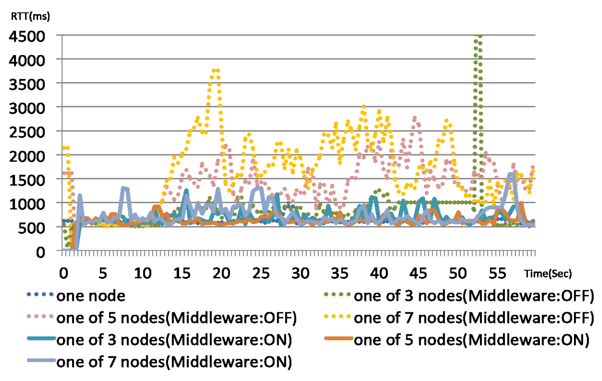

Figure 5: Time transition of the RTT

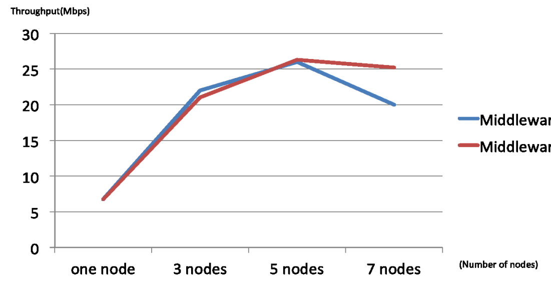

Figure 6: Effect of the middleware on Nexus 7 terminals

To further improve experimental coverage, we conducted another test, running the middleware on Nexus 5 phones. The results showed that it is effective in a heavily crowded network. We evaluated basic performance of a Nexus 5 phone in a WLAN network with an experimental system as presented in Figure.8. The difference from the experimental setup shown in Figure.3 is the load. In this setup, we replaced Dummynet, and added Nexus 7 tablets running Iperf to generate the network traffic. In this heavily loaded network, we used two Nexus 5 phones as client terminals.

Figure.9 shows the total throughput in an environment shown in Figure.8. The left bar shows the total throughput without the middleware and the right bar shows the total throughput with the middleware. The performance improved by approximately 1.43 times.

5. TESTING AT LTE-NETWORKS

The testing environment at Centria creates a possibility to test the effects of data traffic and congestion in a heterogeneous network (HetNet). In their previous studies, Centria has been working to evaluate and improve the performance of the mobile network. Lately, Centria has concentrated on evaluating the performance of active antenna system (AAS) [15]. As a result, it is seen that AAS can provide a flexible beamforming for changing situations in the network. According to Centria's tests, the AAS improves the throughputs within the system and gives flexibility of adding capacity to locations where it is needed. However, AAS needs a lot of configuration, which is necessary in each new situation. Therefore, a significant amount of development work

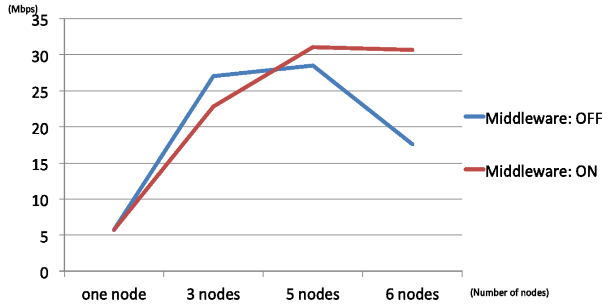

Figure 7: Effect of the middleware on Nexus S terminals

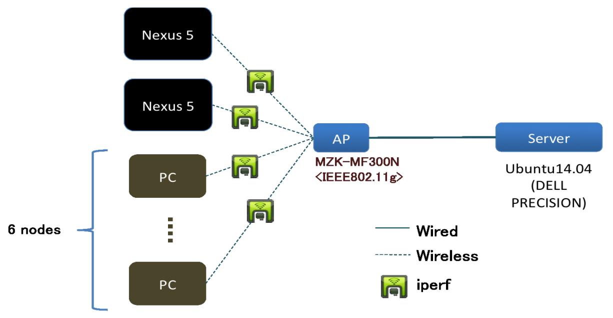

Figure 8: Experimental topology 2

is still to be done before the AAS is fully exploitable and some lighter solutions are needed.

The Centria test environment includes a number of different wireless network systems, which enables us to perform a variety of tests(see Figure.10). Centria test based on the WLAN setup (Figure.8).The LTE network environment consisted of 2.1 GHz passive antennas. For User Equipment we used two Nexus 5 smartphones and 6 laptops with LTE dongle. We chose Nexus 5, since it supports LTE FDD network on used band 1. One Nexus specific benefit is its clean Android operating system without vendor specific add-ons. This makes possible to recompile Kernel with modified functionality. Iperf and the middleware were installed both on them.

Figure.11 shows RTT values during the experiment. Dotted lines show the RTT value of the communication without middleware and solid lines show the RTT value of the communication when middleware was used. When the terminals communicated with middleware, the RTT increase was reduced to about a half. This shows that the middleware was effective in this LTE network. In addition, it is possible that the RTT of wired part affects the general performance. Therefore, the middleware can be expected to improve total throughput.

6. CONCLUSIONS

This study has focused on the ACK packet backlog problem with the upstream TCP sessions and has proposed a CUBIC based CWND control mechanism that utilizes the RTT as an indication for the TCP ACK backlog condition at the WLAN AP. It controls the upper and the lower bounds of its CWND size to suppress excessive transmissions of own TCP

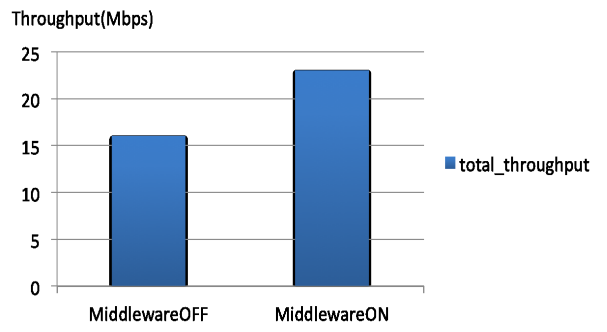

Figure 9: Effect of the middleware on Nexus 5 terminals

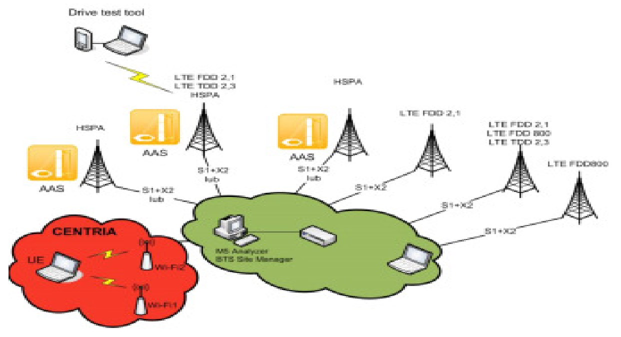

Figure 10: Drive test setting of AAS on testing environment of Centria

DATA packets. Moreover, we created environment similar to the WLAN test setup (Section 4) in the Centria's LTE network test (Section 5) and evaluated the transmitting performance of the Android terminal. The experimental study with up to seven (7) Nexus 7 terminals and up to six (6) Nexus S terminals in WLAN network shows that the congestion control middleware can improve the total throughput of the WLAN. In particular, it improved the TCP throughput in case of many terminals communicating through the same AP. Furthermore, in a heavily crowded network, the middleware can improve the total throughput of two (2) Nexus 5 terminals.

Continuing the analysis, we evaluated the performance in LTE network. Then, we observed that the middleware worked in LTE network and improved the RTT values of two Nexus 5 terminals in a heavily congested network.

Centria, in collaboration with Ochanomizu University, will continue testing smartphones in different networks. For our future work, we will evaluate total throughput in LTE network to see the effect.

7. ACKNOWLEDGMENTS

The authors would like to acknowledge all the people contributing to the tests. Juhana Jauhiainen, Tero Kippola, Marjut Koskela and Juha Erkkilä have worked on the tests with the middleware on Centrias test network. Centria would also like to thank the Finnish Funding Agency for Technology and Innovation (Tekes) for funding the project Cybersecurity in the Wireless Industrial use case - CyberWI, on which these tests were executed.

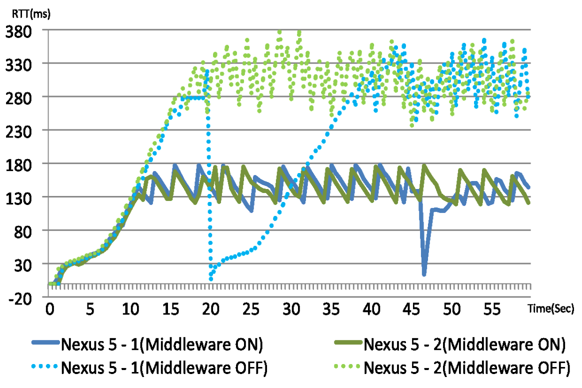

Figure 11: Time transition of the RTT

8. REFERENCES

[1] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control for fast, long distance networks. In *Proceedings of Tech. Report*. Computer Science Department, NC State University, 2003.

[2] S. Ha, I. Rhee, and L. Xu. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review- Resarch and developments in the Linux kernel*, 42:64–74, July 2008.

[3] *A CSMA/CA MAC protocol of Cognitive Networks - emfield*.

[4] P. Sinha, T. Nandagopal, N. Venkitaraman, R.y Sivakumarand, and V. Bharghavan. Wtcp:a reliable transport protocol for wireless wide-area networks. *Wireless Networks - Selected Papers from Mobicom'99 archive*, 8:301–316, March - May 2002.

[5] C. Casetti, M. Geria, S. Mascolo, M. Y. Sanadidi, and R. Wang. Tcp westwood: end-to-end congestion control for wired/wireless networks. *Wireless Networks archive*, 8:25–38, September 2002.

[6] L. A. Grieco and S. Mascolo. Performance evaluation and comparison of westwood+, new reno, and vegas tcp congestion control. *ACM SIGCOMM Computer Communications Review*, 34:417–440, April 2004.

[7] S. Liu, T. Başar, and R. Srikant. Tcp-illinois: a loss and delay-based congestion control algorithm for high-speed networks. *Innovative Performance Evaluation Methodologies and Tools: Selected Papers from ValueTools 2006*, 65:467–479, June 2008.

[8] H. Hirai, S. Yamaguchi, and M. Oguchi. A proposal on cooperative transmission control middleware on a smartphone in a wlan environment. In *proc*, pages 710–717. the 9th IEEE International Conference on Wireless and Mobile Computing Networking and Communications(WiMob2013), October 2013.

[9] A. Hayakawa, S. Yamaguchi, and M. Oguchi. Reducing the tcp ack packet backlog at the wlan access point. In *proc.* the 9th ACM International Conference on Ubiquitous Information Management and Communication (IMCOM2015), January 2015.

[10] *Android open source project website*. http://source.android.com.

[11] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In *proc.* ACM

SIGMOBILE 7/01 Rome, 2001.

[12] K. Miki, S. Yamaguchi, and M. Oguchi. Kernel monitor of transport layer developed for android working on mobile phone terminals. *Proceedings of The Tenth International Conference on Networks (ICN)*, pages 297 –301.

[13] *The dummynet project website.* http://info.iet.unipi.it/˜luigi/dummynet/.

[14] *Iperf For Android Project in Distributed Systems.* http://www.cs.technion.ac.il/˜sakogan/DSL/2011/projects/iperf.

[15] M. Heikkilä, T Kippola, J. Jämsä, A. Nykänen, M. Matinmikko, and J. Keskimaula. Active antenna system for cognitive network enhancement. In *5th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pages 19–24, November 2014.