Performance Evaluation of Transmission-Control Middleware in WLAN and LTE networks

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, Joni Jämsä
Centria Research and Development
Centria University of Applied Sciences
Ylivieska, Finland
heidi.kaartinen@centria.fi
joni.jamsa@centria.fi

***Abstract*—Since mobile terminals such as smartphones are basic information tools for users, their communication performance is always significant. Modern loss-based Transmission Control Protocols (TCP) 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 Local Area Network (WLAN), 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 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 three Android terminals shows that the proposed mechanism can improve both aggregate throughput and fairness of the WLAN, and that it is highly effective particularly for cases where very long RTTs are observed.**

Keywords—WLAN; LTE; TCP; CWND; RTT; congestion control; Android

I. 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 three Android terminals shows that the proposed mechanism can improve both aggregate throughput and fairness of the WLAN, and that it is highly effective particularly for cases where very long RTTs are observed.

II. 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 II C. This paper also presents the results of the behavior of Android terminals in in Long Term Evolution (LTE)-networks.

III. BACKGROUND

A. Android Operating System

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 (OS) 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 Fig. 1. 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.

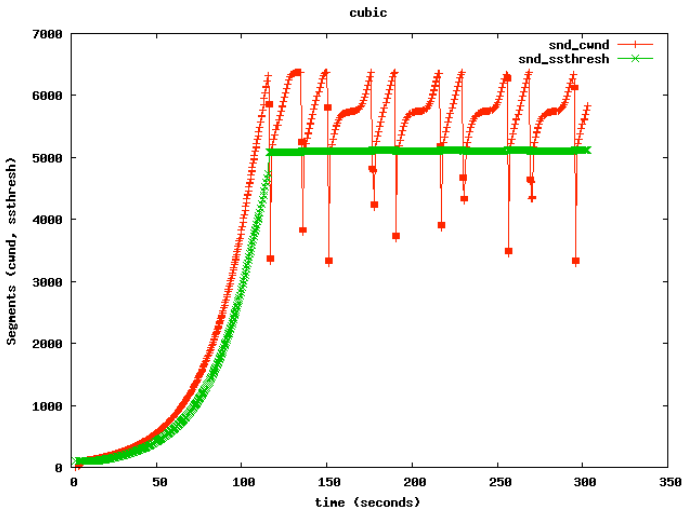

Fig. 1. Behavior of CUBIC TCP

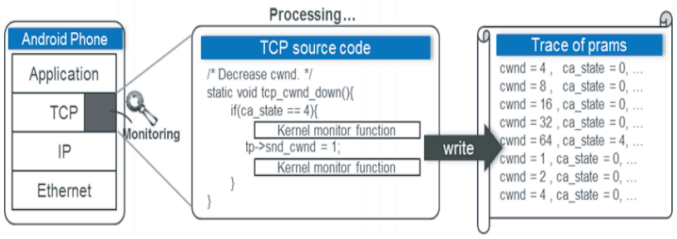

Fig. 2. Kernel Monitoring Tool

B. Kernel monitoring tool

We have developed a method to observe a behavior of parameters inside the Linux Kernel code. Our previous work [12] successfully embedded a system tool called Kernel Monitoring Tool in the Android platform in order to analyze the connection status of a mobile host. As shown in Fig. 2, it allows users to monitor parameters in the kernel processing at the mobile host, which include CWND, RTT, and socket buffer queue. 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.

C. CWND-controlling middleware

This subsection describes the CWND-controlling middleware that has been implemented in our previous work [24]. The middleware controls CWND if the terminal originates TCP traffic and is connected to the server via a WLAN, in order to address congestion among the AP and other Android terminals.

The middleware can be divided into two parts, as shown in Fig. 3. One part adjusts the congestion control, using the process interface to prevent segments from overflowing and filling the bandwidth. The notification is broadcast by User Datagram Protocol (UDP) every 0.3 seconds from the other part because the kernel parameters frequently change. The adjustment 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 notification interval and the arrival rate of the notifications.

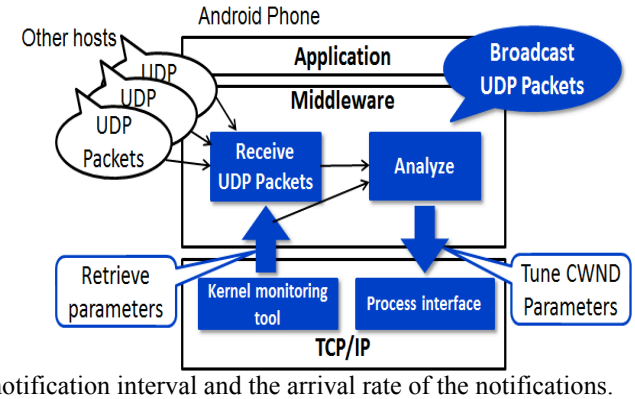

Fig. 3. Summary of the system

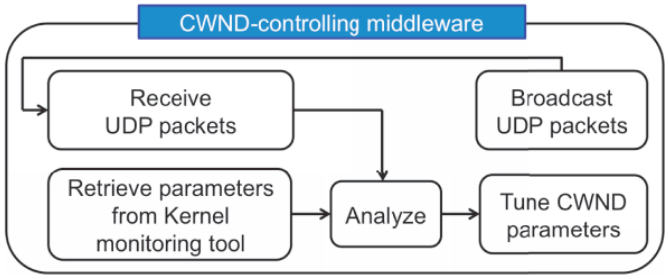

Fig. 4. Composition of the middleware after modification

Fig. 4 shows the composition of the middleware. The values of RTT and minimum RTT (min_rtt) are acquired by observing data from the kernel monitoring tool constantly. A value of min-rtt is updated by overwriting with the smallest RTT during the communication. Based on the acquired values, RTT Ratio (ratio_rtt) is obtained with Equation 1, which indicates increase and decrease of RTT. The behavior of the middleware can be customized based on this value.

$$ratio\_rtt = \frac{RTT}{\min\ rtt} \tag{1}$$

(1)

Simultaneously, traffic condition is predicted by receiving packets and knowing the communication situation of other terminals. This system sets levels of upper and lower limit for the CWND based on the RTT ratio and the number of communication terminals using the proc interface, which is controllable from an outside process of kernel and can be tuned up for optimization. Using this method, the control system can limit the quantity of traffic outbreak and share a bandwidth of a terminal, after the initial communication is enabled by setting a CWND level at an appropriate value.

The communication situation is checked approximately every 0.5 seconds in the current implementation. CWND level is modified when the middleware detects another terminal that shares the same AP begins communication and the RTT values suddenly increase as a result. With 0.5 seconds frequency, the kernel monitoring tool can grasp the timely situation moderately and optimize it. If the frequency is too high, it may become an overhead.

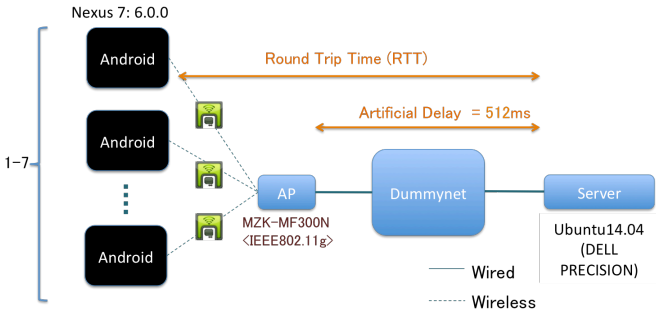

Fig. 5. Environmental topology

$$Bandwidth\_DelayProduct\,(BDP) = Bandwidth[Mbps] \times RTT[sec] \tag{2}$$

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 and lower 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.

$$IdealThroughput = \frac{CWND \times 1.5\,[Kbyte] \times 8}{RTT[sec]} \tag{3}$$

D. Equations for proposed control mechanism

In each delay environment, we decide the levels of upper and lower limit for the CWND referring to Equations 2, 3 and 4 [13]. In these equations, the Bandwidth Delay Product (BDP) is filled with data transferred by connections that are assigned the divided bandwidth. The limit values are decided based on these equations.

$$Idealcwnd = \frac{BDP}{Segment\ size\ (1.5\ Kbyte) \times number\ of\ terminals} \tag{4}$$

IV. Testing at WLAN Network

We evaluated the basic performance in WLAN network with the experimental system in Figure 5. The client terminals were connected to an AP over 802.11g, and the AP was connected to the server host through the wired route. The client terminals were Nexus 7. The number of terminals ranged from 1 to 7. To emulate network delay and packet loss, a network emulator, Dummynet [14], was inserted between the AP and the server host; 256 ms delay was set by assuming a high-delay environment. In this environment, the wired parts are connected with higher rate because of Gigabit Ethernet, whereas a bandwidth is only about 20Mbps in the wireless parts. Thus, the radio transmission sections between the AP and the terminals were a bottleneck, which was 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 length of the packet queue for transmission increase. As a network benchmark tool, Iperf for Android [15] was installed on all the terminals.

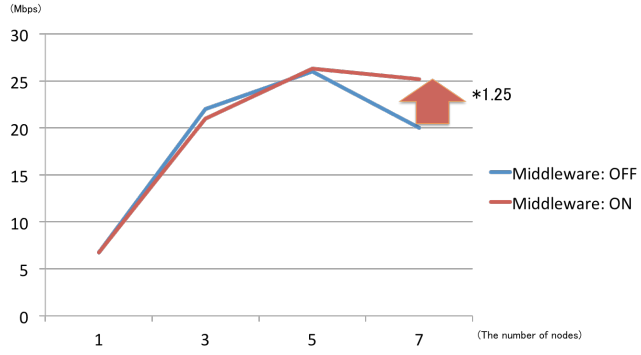

Fig. 6. Effect of the middleware

Fig. 6 shows the relationship between the number of terminals and the total throughput. The blue and red lines show the throughput without and with the middleware, respectively. The blue line drops when the AP is overloaded, i.e. the number of terminals exceeds five (5). In contrast, the red line does not decrease with an advantageous effect of the middleware. The performance with seven (7) terminals is improved by approximately 1.25 times.

V. Testing at LTE networks

The testing environment at Centria enables a possibility to test the effects of data traffic and congestion in a heterogeneous network (HetNet). In its previous studies, Centria has been working to evaluate and improve the performance of the mobile network. Recently, Centria has concentrated on evaluating the performance of active antenna system (AAS) [16]. As a result it is seen that AAS provides a flexible beamforming for changing situations in the network. According to Centrias tests the AAS improves the throughputs within the system and flexibly adds capacity to locations where it is needed. However, AAS configuration is a challenge, as tasks are performed in changing environment. Some lighter solutions are required while AAS is developed to be fully exploitable on tests and demonstrations.

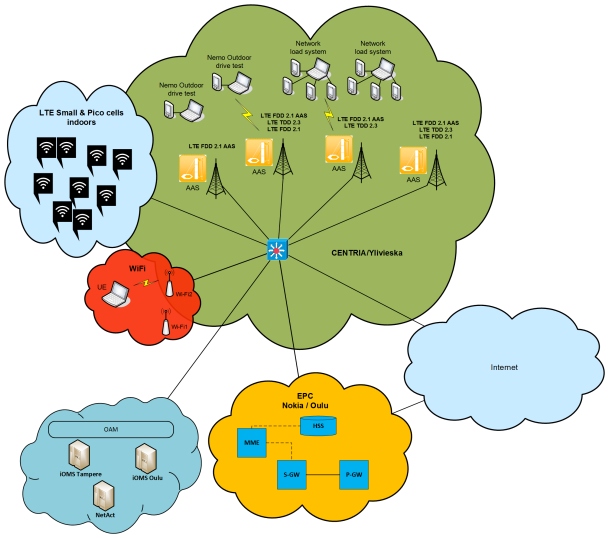

Fig. 7. Drive test setting of AAS on the testing environment of Centria.

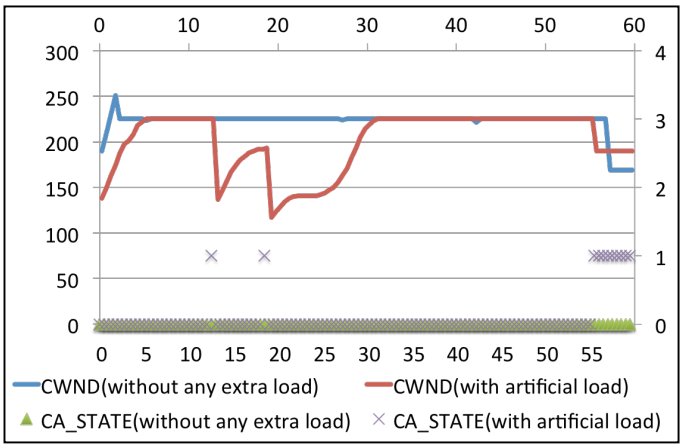

Fig. 8. Transmission of CWND and CA_STATE

As there are different kinds of LTE bands within the Centria test environment (see Fig. 7), many types of demonstrations can be performed with them. The setup of WLAN (described in Section IV) was implemented on the Centria's test on LTE network environment. 2.1 GHz passive antenna was used for testing with two Nexus 5 smartphones. Nexus 5 was selected to be the terminal of the test because it supports the used LTE bands. Both Nexus-terminals were using Android OS. Iperf and the transmission control middleware were installed in them. One of the phones was used as a server and the other one was used as a client. As there were only two terminals at use in this test, the LTE network was congested by uploading packets to the network with a load generator. On the first test round the throughput was tested without any extra load on the network, where 71.5 Mbytes were transferred within the test sequence of 60 seconds. The average transfer speed was 9.99 Mbits/sec. In the second round the test was executed the same way, but artificial load was added to the network with uplink of a load generator with a capacity of 10.1 Mbits/sec. During the second test round the amount of transferred data was 40.2 Mbytes and the speed was 5.63 Mbits/sec.

As the tests were executed without the middleware on, the amount of transferred data was 71.9 Mbytes and the speed 10 Mbits/sec and with generated load 41.8 Mbytes and 5.80 Mbits/sec. This indicates that the results of the tests in LTE network differ from the results on the WLAN. In the cases with the brief testing period, it was observed that the total size of data transferred without the middleware was greater than that with it. This is because the middleware was developed for cases when the number of terminals connected concurrently to the same AP is large. In other words, we can utilize the middleware effectively with heavily crowded network. Hence, we need to evaluate with a larger number of terminals to observe the effect of middleware clearly. However, network congestion can be observed in the network with uplink of load generator (Fig 8). Also, a time transition of CA_STATE that indicates the state of TCP acquired by the kernel monitoring tool is shown in Fig 8. In these states:

- "0" means "Open": normal state

- "1" means "Disorder": replacement of packets
- "2" means "Congestion Window Reduced (CWR)": congestion notice
- "3" means "Recovery": Fast Retransmission
- "4" means "Loss": Timeout and loss of Packets

The communication with artificial load has some errors and congestion happens. It is considered that the middleware is likely to achieve higher performance if a heavier network burden is observed.

VI. 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 DATA packets. Moreover, we applied the environment of WLAN (Section IV) to the Centria's LTE network test (Section V) and evaluated the transmitting performance of the Android terminal.

The experimental study with up to seven (7) Nexus 7 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 by a factor of 1.25 when many terminals are communicate through the same AP.

Continuing the analysis, we evaluated the performance in LTE network. Then, we revealed that the experimental study with the two terminals in LTE network did not properly demonstrate the effect of the middleware, because it is believed that the middleware is effective in the case of heavy network burden such as when artificial delay is larger or when there is a larger number of terminals communicate in the same network, while such conditions were not present during aforementioned experiment.

Solutions of cognitive infocommunications set a new kind of demand for the networks and their capacity, as the amount of communication between human and infocommunication devices within the networks will increase. [17] In case of congestion in the network, the reliability of the data transfer must be improved. New solutions are to be developed and implemented to meet this rapidly growing demand. Such solutions as we have presented on this paper, will utilize the maximum capacity available within the network and speed up the data traffic.

Centria will continue in collaboration with Ochanomizu University testing smartphones in different networks. For our future work, we will test the middleware with a large number of smartphones and a server-PC to see the effect.

Acknowledgment

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 Cyber-security in the Wireless Industrial use case – CyberWI, on which these tests were executed.

References

[1] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast, Long Distance Networks," 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 - Research and developments in the Linux kernel, vol.42, pp.64-74, July 2008.

[3] "A CSMA/CA MAC protocol of Cognitive Networks – emfield".

[4] P. Sinha, T. Nandagopal, N. Venkitaraman, R.y Sivakumar, and V. Bharghavan, "WTCP:a reliable transport protocol for wireless wide-area networks," Wireless Networks - Selected Papers from Mobicom'99 archive, vol. 8, issue 2/3, pp. 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, vol. 8, issue 5, pp. 467-479. 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, vol. 34, no. 2, pp. 25-38, 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, vol. 65, issues 6–7, pp. 417-440, 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. the 9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob2013), pp.710-717, 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), 5-4, Bali, Indonesia, January 2015.

[10] Android open source project, 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, Italy, 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), pp. 297-302. 2011.

[13] W. R. Stevens, TCP/IP illustrated, Vol.1 Protocol, Pearson Education, 2000.

[14] The dummynet project, http://info.iet.unipi.it/ luigi/dummynet

[15] Iperf For Android Project in Distributed Systems, http://www.cs.technion.ac.il/sakogan/DSL/2011/projects/iperf/index.html

[16] M. Heikkilä, T Kippola, J. Jämsä, A. Nykänen, M. Matinmikko, and J. Keskimaula, "Active Antenna System for Cognitive Network Enhancement," 5th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), pp. 19-24, Vietri sul Mare, Italy, November 2014.

[17] P. Baranyi, A. Csapo, and P. Varlaki, "An Overview of Research Trends in CogInfoCom," in IEEE International Conference on Intelligent Engineering Systems, pp. 181-186, Tihany, Hungary, July 2014.