Empirical Evaluation and Analysis of Application-Layer Delay Reduction Methods over Wireless Access Networks

Yoshiaki Nishikawa, Takashi Oshiba, Dai Kanetomo, and Kazuaki Nakajima

NEC Corporation, 1753 Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa, 211-8666 Japan
{y-nishikawa,nakajima}@ah.jp.nec.com, oshiba@cp.jp.nec.com,
d-kanetomo@ce.jp.nec.com

Abstract. In this paper, we discuss the experimental results of application-layer delay reduction methods, which can reduce the delay during TCP applications. Due to the popularization of smart phones, Wireless Local Area Network (WLAN) access such as WLAN hot spot or WLAN tethering is increasing. This increase leads to a WLAN channel being shared by multiple data flows, resulting in the degradation of the communication performance. Specifically, the delay of the real time applications that communicate over TCP can be made longer. To reduce the delay, we developed three methods: reducing the amount of data to send, triggering TCP's fast retransmission earlier, and returning TCP's acknowledgment immediately. In our experiments, we measured the delay during the real time TCP applications connected with each other over WLAN and commercial fixed network and analyzed the effectiveness of our methods in the case of WLAN performance being degraded by the file transfer flow. Results indicate that our methods can reduce the 95th percentile of the delay by as much as 84%.

Keywords: TCP, Delay Reduction, Application-Layer Method.

1 Introduction

Due to the popularization of smart phones, Wireless Local Area Network (WLAN) access is increasing. Internet access over WLAN is offered in various places such as homes, offices, and hotels. Mobile devices such as smart phones and tablets on which only wireless LAN is available have become popular. WLAN is thus being increasingly accessed by mobile devices in addition to PCs. It has also became common to access the Internet through mobile PCs, tablets, or gaming devices by using a mobile router or a WLAN tethering function on a smart phone. A mobile router is connected to the Internet via mobile wireless communication technology such as 3G and LTE and can function as a WLAN access point. The WLAN tethering involves using a smart phone as a mobile router. People are establishing their own WLAN on the basis of their demands and are connecting their mobile devices to the Internet.

M. Matera and G. Rossi (Eds.): MobiWIS 2013 Workshops, CCIS 183, pp. 17–24, 2013.

There are several problems degrading the performance of WLAN and real time applications. Competing and interfering problems occur because WLANs that do not cooperate with each other are established side by side. In the competing problem, the performance is degraded by the impact from other WLANs sharing the same communication channel. In contrast, in the interfering problem, the performance is degraded by the impact from other WLANs overlapping in the spectrum of the communication channel. There is also the problem of the growing traffic. More traffic is offloaded from mobile wireless systems to WLAN. These problems significantly affect the real time applications that communicate frequently over TCP because of its retransmission and reordering.

In this paper, we propose delay reduction methods on the real time applications and perform an experiment to evaluate these methods in the case of the performance of WLAN being degraded by the file transfer flow. We propose three methods: to reduce the number of data to send, to trigger the TCP fast retransmission earlier, and to return an ACK immediately. We try to reduce the delay to less than 100ms, which is required as the delay of a real time application [8]. Results show that our methods can reduce the 95th percentile of the delay by as much as 84% and meet the above requirement.

2 Related Work

In this section, we describe work related to this paper. Some researchers have sought ways to manage competing and interfering problems. Choi et al. proposed a channel selecting algorithm that scans channels used by other access points and selects its own channel automatically [3]. Lee et al. [2] proposed an approach to optimize access point placement and channel assignment in WLANs. Though these approaches to manage WLAN directly can improve WLAN performance, the cases in which we cannot manage WLAN ourselves need to be considered. Brosh et al. presented experimental results for the delay of a real time TCP application [4]. According to their paper, the method that splits data and uses multiple TCP connections can reduce the delay. In this paper, we evaluate our simple application-layer methods when the performance of WLAN is degraded by the file transfer flow.

3 Application

In this section, we explain the real time application we use. We use the Android device as a platform on which the application and our methods run. This is because Android devices are widely used to access to the Internet. For the real time application, we select an application that sends and receives a small size data that indicates user's operation on a touch panel. A remote desktop application is an example of this. We are interested in the End-to-End one way delay, which is the time that it takes data to travel across the network from source to destination. This is because the responsiveness of the real time application depends on how short the End-to-End delay is. For example, in a remote desktop application, a local device

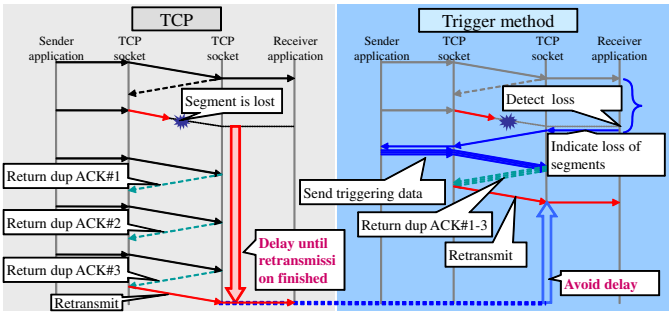

Fig. 1. Left: sequence of TCP fast retransmission. Right: sequence when Trigger method is used.

sends data indicating user's operation to a remote device and receives the result image from a remote device. The End-to-End delay of operation data becomes shorter, so users can obtain the result image earlier. Thus, the End-to-End delay of operation data must be reduced. In ITU-T Recommendation Y.1541 [8], 100ms is required as the End-to-End one way delay of a real time application. Hereinafter, we refer to the delay as the End-to-End one way delay.

We suppose that two of TCP's controls make the delay long. One is the reordering control to reconstruct the sending order on the receiving side. TCP performs the retransmission when it detects a loss of a segment. Throughout this paper, we refer to the transmission unit of TCP as a segment and to the transmission unit of application as data. By the reordering control, the segment sent after the dropped segment is waited for at the buffer of the receiving side until the lost segment has been transmitted safely. Then, the delays of data corresponding to the waited segments are extended. More data are delayed by the reordering control in the real time application than in the non-real time application that sends data less frequently. Second is the flow control to avoid the buffer overflow at the receiving side. TCP limits the number of segments that TCP can send without receiving ACK. After reaching the limit, TCP delays sending till receiving ACK at the sending side. At the receiving side, TCP also delays returning ACK until the timer have reached a certain timeout value or the receiver has received multiple segments. While TCP delays returning ACK, the number of segments that have been sent already by the sending side is more likely to reach the limit in the real time application than in the non-real time application.

4 Methods

In this section, we propose three delay reduction methods on the real time applications. In the first method, the application can reduce the number of data to send and its consumption of bandwidth. We call this data reducing method the Reduction-P method, where P is the factor representing reduction percentage.

Sending segments frequently can increase the long queuing time in the network and the delay by causing congestion and reordering. This method reduces the times and the total size of sending data to expand the interval between sending events so that the number of segments in the network and the number of congested segments can be reduced.

In the second method, the application can trigger retransmission faster by sending several datasets and reduce its delay. We call this data sending method the Trigger method. Figure 1 illustrates the sequence of TCP fast retransmission on the left and the sequence using Trigger method on the right. TCP retransmits the dropped segment when multiple (commonly two or three) dup ACKs have been received. The dup ACK is generated at the receiving side and sent in reply to the sending side when the segment is received in the wrong order. In Trigger method, the receiver infers the loss of segments from the interval of receiving data that is too long. Then the receiver sends the data indicating the loss of segments. Immediately after receiving this data, the sender sends the certain number of the triggering data that are very few. When these triggering data are received at the receiver side, dup ACKs corresponding to the triggering data is sent in reply to the sender. Then at the sending side, the dropped segment is retransmitted since a certain number of dup ACK has been received.

In the third method, the application can sending TCP ACK in reply immediately by returning data even though Delayed ACK is enabled. We call this method the Return method. TCP delays sending ACK in reply as we mentioned before. TCP has the piggyback ACK function in which ACK can be sent in reply when there is a segment to be sent at the receiver side. In the Return method, the receiver sends data in replay immediately after receiving data to send ACK in replay immediately.

5 Experimental Setup

In this section, we explain the experimental environment and settings. Figure 2 shows the experiment network we used. The network consists of three home networks illustrated in the left of the figure and one server's network illustrated in the bottom right. These four networks are connected with each other over the commercial fixed network via FTTH. The sender application runs on the Android device in the sender's network. Two receiver applications run on the Android device in both two receiversf networks. Two receiversf networks have different Received Signal Strength Indication (RSSI). The higher one is -40 dBm, and the lower is -58 dBm. Android devices are connected to wireless access points over IEEE802.11n. In the server's network, the relay server is connected to the wired router. The sender application sends data to the relay server, and the relay server relays the data to one of the receiver applications. Our methods are controlled between the sender application-to-relay server and relay server-to-receiver application. To simulate the degradation of the WLAN performance, Windows PC is used, which is the same model to the relay server connected to the access point over IEEE802.11n in the sender's network and can make

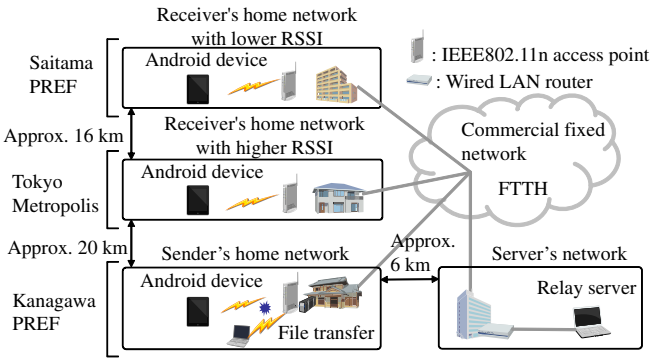

Fig. 2. The experiment network

Table 1. Device specification

	Android device	Relay server
Product name	SGH-N023[6]	VY14AC-W[7]
OS	Android OS 2.3	Windows XP SP3
CPU	S5PC110 1GHz	core2duo 1.4GHz
RAM	0.51 GB	2.96 GB
NIC	Wireless (IEEE802.11n)	Wired (1000BASE-T)

bulk file transfer flow from network attached storage in the same WLAN. The specifications of the Android device and relay server are shown in Table 1. We use AtermWR8750N as an WLAN access point and wired router. Its available bandwidth is about 184 Mbps as an wireless access point and about 840 Mbps as a wired router [5].

In our experiment, we use the data generated sequentially by sliding a finger on the touch panel as the data communicated between applications. These data are sent at 60 times by each 30ms. Each piece of data is 50 bytes and is sent immediately from application to TCP. Like most delay sensitive applications, Nagle's algorithm [1] is disabled in this paper. We refer to the stream as the data generated by one sliding event and the delay of a stream as the delay calculated by the average of the delay of data of which one stream consists. In the following part of this paper, we describe the delay of the stream, not the delay of each piece of data. The delay of the real time application needs to be less than 100ms[8]. Our goal is to fulfill this requirement. Statistics typically used to evaluate the delay are mean, median, or mode averages. To evaluate the delay sensitive real time application, these statistics are not considered appropriate. This is because the long delay that does occur not very often spoils the real time application. Thus, we use the 95th percentile, which is the most commonly used percentile value.

We measured to ensure that the file transfer flow degrades the performance of WLAN. Figure 3 shows the distribution of the frequencies of the delay of all 300 streams measured in the two cases: with or without a file transfer flow.

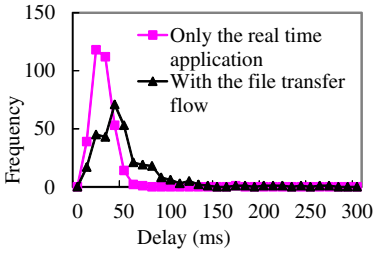

Fig. 3. Distribution of the frequencies of the delay

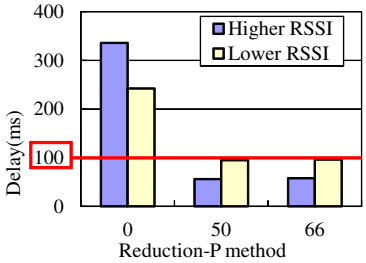

Fig. 4. Delay using TCP and Reduction-P method

This distribution is drawn with the delay of the stream on the abscissa axis and the frequencies of the delay on the ordinate. We can see that the delay in the case with the file transfer flow is longer than that in the case of only the real time application flow. In the case of only the real time application flow, the maximum delay is 162ms and the 95th percentile of the delay is 42ms. In the case with the file transfer flow, the maximum delay is 2451ms, which is not displayed, and the 95th percentile of the delay is 354ms. Thus, the performance of WLAN is clearly degraded due to the file transfer flow.

In our experiment, each of our methods is set as follows. In the Reduction-P method, P is set to 50 or 66. Intervals are expanded from 30ms to 60ms or 90ms, respectively. In the Trigger method, we set the time used by the receiver to infer the loss of segments to the time made by adding the interval decided by the Reduction-P method and 20ms. For the Return method, we tested with and without the Return method.

6 Experimental Results

In this section, we show experimental results of each method compared with results of TCP. Then, we show results of combinations of our methods. In the following section, we refer to the 95th delay as the 95th percentile of the delay, and figures are drawn with the name of methods on the abscissa axis and the 95th delay on the ordinate. Each 95th delay in this section is calculated by delays of 300 streams.

Figure 4 shows the 95th delay when using TCP and the Reduction-P method with higher RSSI and lower RSSI. In the case with higher RSSI, the 95th delay by using TCP is 336ms, and the delays by using Reduction-50 and Reduction-33 are 56ms and 58ms, respectively. In the case with lower RSSI, the 95th delay by using TCP is 242ms, and the delays by reducing the number of data to 30 and 20 are 95ms and 98ms, respectively. By reducing the number of data to 30, the delay is reduced by 83%. This is despite there being just a marginal difference between 30 and 20 as the number of data. Figure 5 shows the 95th delay when using TCP and the Trigger method with higher RSSI and lower RSSI. In the case with higher RSSI, the 95th delay by using the Trigger method is 96ms and is reduced by 73% compared with the delay by using TCP. In the case with lower

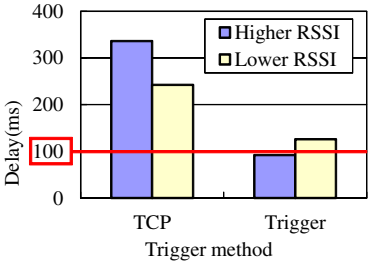

Fig. 5. Delay when using TCP and Trigger method

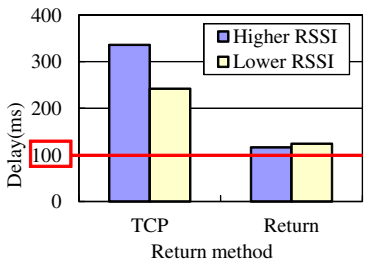

Fig. 6. Delay when using TCP and Return method

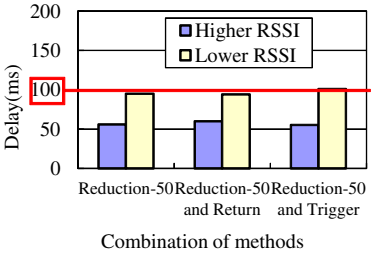

Fig. 7. Delay when using only Reduction-50, by Reduction-50 with Trigger and Reduction-50 with Return

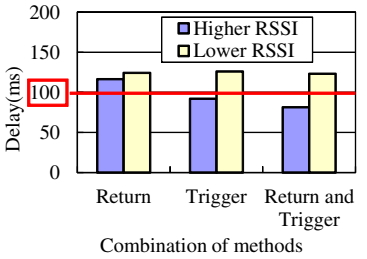

Fig. 8. Delay when using only Trigger, by only Return and Trigger with Return

RSSI, the 95th delay by using the Trigger method is 126ms and is reduced by 48%. Figure 6 shows the 95th delay when using TCP and the Return method with higher RSSI and lower RSSI. In the case with higher RSSI, the 95th delay by using the Return method is 116ms and is reduced by 65% compared with the delay using TCP. In the case with lower RSSI, the 95th delay by using the Return method is 124ms and is reduced by 49%.

Figure 7 shows the 95th delay with higher RSSI and lower RSSI by using only Reduction-50, Reduction-50 with Trigger, and Reduction-50 with Return. In the case with higher RSSI, the 95th delays by using Reduction-50 with Trigger and with Return are 55ms and 60ms, respectively, and are reduced by as much as 84% compared with the delay using TCP. In the case with lower RSSI, the 95th delays by using Reduction-50 with Trigger and with Return are 101ms and 94ms, respectively, and are reduced by as much as 61%. Reduction-50 method does not need to be combined with other methods. In the case with higher RSSI, Reduction-50 and Return decrease their gains in combination. In the case with lower RSSI, Reduction-50 and Trigger decrease their gains too.

Figure 8 shows the 95th delay with higher RSSI and lower RSSI when using only the Trigger method, only the Return method, and the Trigger method with the Return method. In the case with higher RSSI, the 95th delay by using Trigger and Return at the same time is 81ms and is reduced by 76%. In the case with

lower RSSI, the 95th delay by using Trigger and Return at the same time is 123ms and is reduced by 49%. In the case with higher RSSI, Trigger and Return increase their gains in combination. However, in the case with lower RSSI, the gain is marginal. We also have concern about overhead of our method. Indeed the consumption of bandwidth by using Trigger and Return at the same time grows about three times as much as only user operation data. However, the overhead is negligible, because overhead bandwidth up to 100kbps is sufficiently narrower than the available bandwidth up to 180Mbps[5].

7 Conclusion

We have ensured that the file transfer flow degrades the performance of WLAN and shown that our methods running on applications can reduce the 95th percentile of the delay as much as 84% to less than 100ms when the performance of WLAN is degraded by the file transfer flow. Though the method reducing the amount of data is always the most effective, the method trying to trigger TCP fast retransmission is more effective in the case with higher RSSI and less effective in the case with lower RSSI than the method returning ACK immediately. This is because the delay of data is stable in the case with higher RSSI though it is moving strenuously in the case with lower RSSI. The method that tries to trigger the TCP fast retransmission earlier can reduce the delay due to losses of segments, which does not happen very often in the case with higher RSSI. However, this method is not very effective due to the fixed timer when the delay is changing strenuously. Therefore, the method that tries to return ACK immediately is more effective in the case with lower RSSI.

References

1. Nagle, J.: Congestion Control in IP/TCP Internetworks. RFC 896 (January 1984)
2. Lee, Y., Kim, K., Choi, Y.: Optimization of AP Placement and Channel Assignment in Wireless LANs. In: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks, LCN 2002, November 6-8 (2002)
3. Choi, J., Lee, K., Lee, S.R., Ihm, J.(J.): Channel Selection for IEEE 802.11 Based Wireless LANs Using 2.4GHz Band. IEICE Electronics Express (ELEX) 8(16), 1275–1280 (2011, 2012)
4. Brosh, E., Baset, S.A., Rubenstein, D., Schulzrinne, H.: The Delay-Friendliness of TCP. In: SIGMETRICS (June 2008)
5. NEC Corporation. Product Information of AtermWR8750N, `http://121ware.com/product/atermstation/product/warpstar/wr8750n-hp/`
6. SAMSUNG. Product Information of GALAXY Tab SC-01C, `http://www.samsung.com/jp/support/model/SGH-N023CWNDCM`
7. NEC Corporation. Product Information of VersaPro, `http://121ware.com/e-manual/m/nx/vp/base/vy14acw.html`
8. ITU-T Recommendation Y.1541: Network performance objectives for IP-based services (2011)