Journal of Mobile Multimedia, Vol. 2, No.1 (2006) 081-096
© Rinton Press

IMPROVING PERFORMANCE OF MULTIMEDIA DATA DOWNSTREAM WITH PDA IN AN INFRASTRUCTURE NETWORK

HYE-SUN HUR,   YOUN-SIK HONG

University of Incheon, Korea

{mshush, yshong@incheon.ac.kr}

Received August 27, 2005
Revised January 1, 2006

Assuming that PDAs are used mainly to download data from a stationary server such as a desktop PC in an infrastructure network based on a wireless LAN, an overall performance heavily depends on the performance of such a download with a PDA. Unfortunately, the time taken to receive data from a PC to a PDA is longer than the time taken to send it by 53%. Thus, we measured and analyzed all possible factors that could cause the receiving time of a PDA to be delayed with a test-bed system. There are two crucial factors: *TCP window size* and *file access time* of a PDA that can affect the receiving time. The window size of a PDA during the downstream is reduced dramatically to 686 bytes from 32,581 bytes. In addition, because a PDA embeds a flash memory in it, writing data into its flash memory takes 2 times longer than reading data from it. To alleviate these, we propose three distinct remedies: First, to keep the window size at a sender be constant, both the size of a socket send buffer for a desktop PC and the size of a socket receive buffer for a PDA should be increased. Second, to shorten the file access time, the size of an application buffer implemented in an application layer should be set to 8,192 bytes. Finally, an inter-packet delay of a PDA and a desktop PC at an application layer should be adjusted asymmetrically to lower its traffic bottleneck between the heterogeneous terminals.

Key words: multimedia data downstream, PDA, wireless LAN, window size, inter-packet delay, file access time
Communicated by: D Taniar

1 Introduction

A wireless LAN (WLAN) service is a form of telecommunication that uses electromagnetic waves to communicate among mobile terminals within a 50~100 meter radius of their AP (Access Point), where it plays a role of a base station. It is cost effective to use WLAN instead of CDMA/GSM when one wants to utilize his/her multimedia based service within a fixed area.

WLANs can be broadly classified into two types, infrastructure network and mobile ad hoc networks (MANETs), based on the underlying architecture. Infrastructure networks contain special nodes, i.e., APs, which are connected via existing networks. APs are special in the sense that they can interact with wireless nodes as well as with the existing wired network. In this paper, to limit our research scope, we confine our test-bed system to an infrastructure network.

The transmission of multimedia data over wireless networks using mobile devices is becoming a research topic of growing interest. With the emergence of small wireless handheld devices such as PDAs (Personal Digital Assistants) and smart phones, it is expected that interactive multimedia data will be a major source of traffic to these handheld devices [15][18]. Moreover, small handheld devices are utilized broadly in the various applications of multimedia. Accelerometer-based gesture training and recognition are performed inside a small handheld device to support multimedia application control with personal and public devices [19]. ActiveCampus is the architecture of a campus-wide ubiquitous computing network for PDAs using Windows CE. It simultaneously supports extensibility and tight integration in a context-aware infrastructure for distributed, context-aware mobile and wearable systems [20]. CubeView gives data visualization to desktops and mobile devices. A user on the mobile device uses a specific user interface on the user's device to navigate on the screen between OLAP(On-Line Analytical Processing) data and perform OLAP analysis based on data stored locally on the device in highly aggregated and summarized format [21].

Most Internet users spend a lot of time for downloading data from a web server, instead of uploading. We may think that this situation occurs not only in a wired network but also over a wireless network. For example, within the hot spot that is an area for utilizing a WLAN service, a PDA user can download multimedia contents such as soap dramas, movies or music videos and then play them with his/her PDA [1].

Different types of terminals such as desktop PCs as fixed hosts and PDAs as mobile hosts can be connected to an infrastructure network. Assuming that a PDA is currently used for downloading data from its stationary server such as a desktop PC, a desktop PC and a PDA acts as a fast sender and a slow receiver, respectively, due to a substantial difference in their computational capabilities. Actually, a PDA has a lower performance, less memories and poor user interfaces compared to a desktop PC. Most of the works deal with measurements and analysis of their performance with emphasis on laptop PCs [2][4]. In that case, a desktop PC and a laptop PC could be considered as a fast sender and a fast receiver, respectively, or vice versa.

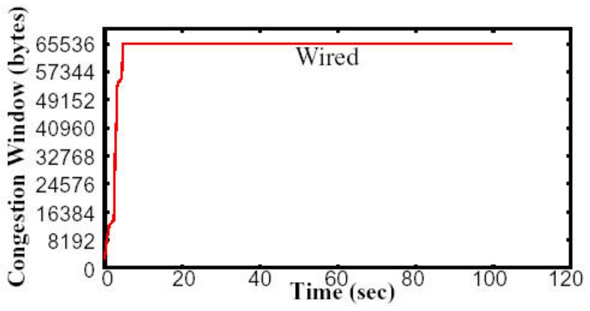
Because the most part of data transmission in an infrastructure network happens during a downstream which is the process of receiving data from a desktop PC to a PDA, an overall performance heavily depends on the performance of the downstream with PDA. Particularly, the time taken to receive data to a PDA from its stationary server is longer than the time to send it by 53% [3] as will be explained in Section 3.2. Thus, we will analyze the factors that cause the receiving time to take longer and propose a method to improve it.

Data transmission protocol adopted in this paper is TCP (Transmission Control Protocol) [8]. TCP calls the congestion window to determine how many packets can be sent at one time. The larger the congestion window size becomes, the higher the throughput becomes [14][16]. TCP uses the additive increase multiplicative decrease (AIMD) algorithm as the window update algorithm. In other words, the congestion window size increases gradually [17].

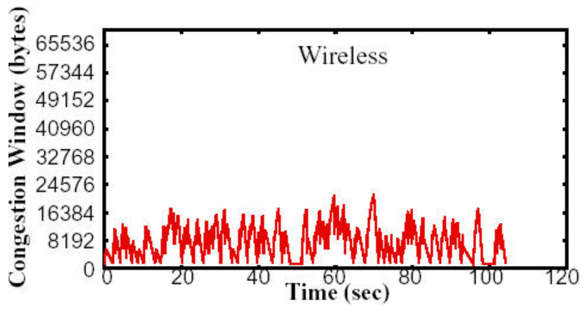
Typically, the congestion window size in a wired network remains constant after short delay as shown in Figure 1(a). However, it over a WLAN oscillates too rapidly as shown in Figure 1(b). If the congestion window size increases too rapidly, it can add to network traffic before the network has completely recovered from congestion. If congestion is experienced again, the congestion window size will shrink rapidly. This alternating increase and decrease in congestion window size causes the performance of data transmission over a WLAN to reduce remarkably [2].

The TCP parameters, the window size at the receiver and the ACK delay, are chosen for our analysis. As shown in Figure 2, we consider a manipulation of the send buffer and the receive buffer at the transport layer as well as an application buffer at the application layer to enhance the performance of a PDA by tuning TCP [12] [13].

According to operating systems, the socket buffer size is different [12]. For each socket, there is a default value for the buffer size, which can be changed by the program using a system library call just before opening the socket. There is also a kernel enforced maximum buffer size. The buffer size can be adjusted for both the send end and the receive end of the socket [14]. It varies with the operating systems. FreeBSD gives 16,384 bytes of default TCP socket buffer size, whereas Windows 2000 and XP give 8,192 bytes of it. But we can't find the figure of it about Windows CE to this time.



(a)



(b)

Figure 1 the congestion window (a) in a wired network and (b) over a WLAN

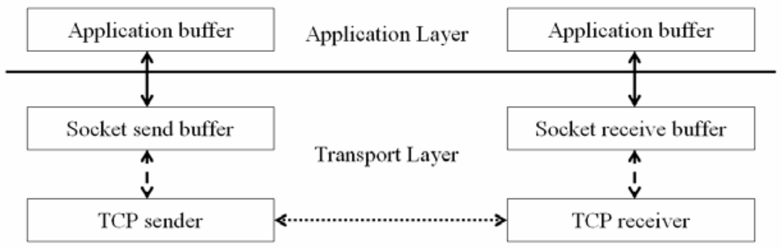


Figure 2 TCP buffers

Table 1 TCP socket buffer size (unit: bytes)

Operating system	default	maximum
FreeBSD	16,384	256K
Linux 2.4/2.6	32,768	64K
Windows 2000/XP[12]	8,192	1G

To achieve maximal throughput it is critical to use optimal TCP send and receive socket buffer sizes for the link you are using. If the buffers are too small, the TCP congestion window will never fully open up. If the buffers are too large, the sender can overrun the receiver, and the TCP window will shut down [14].

We will propose a method for improving the performance of multimedia data downstream at application layer through an analysis based on the following metrics: file access time, window size, and inter-packet delay. The first metric represents a hardware feature of a PDA, whereas the rest of them represent the software features at both a transport layer and an application layer of a given protocol stack. Detailed explanation about these will be given in Section 3.3.

This paper consists of the following; we discuss the related works in Chapter 2. In Chapter 3, we briefly explain file access time, window size and inter-packet delay that are the performance metrics of multimedia data transmission in a WLAN environment. The experimental results are shown in Chapter 4. Finally, we conclude our works in Chapter 5.

2 Related Works

It is important to improve the TCP performance in wireless networks without any modifications of TCP syntax. The snoop protocol of *Balakrishan* et al. [2] modified network-layer software at a base station to improved TCP performance in wireless networks. It used a split mode that a base station connected a fixed host and a mobile host. In split mode, a base station has to reveal data before it reaches the destination, thus violating the end-to-end semantics of the original TCP. But we propose the method to improve the TCP performance in application-layer. It does not modify the base station and uses an end-to-end mode. It modifies several option values in the application of a fixed host and a mobile host.

Balakrishnan et al. [2] used a laptop PC as a server with a WLAN NIC (Network Interface Card) to measure TCP performance in a wireless network. However, when *Pilosof* [5] performed data transmission between mobile hosts, he just simulated it using NS (Network Simulator) software instead of real devices. However, we build a test bed system consisting of a PDA as a mobile host and a desktop PC as a fixed host for a real application and measured actual data. In addition, *Pilosof* used the devices with the same or similar performance in his experiments, while setting an inter-packet delay between each packet to 1-2 ms [5] as a nearly fixed value. In this paper, since there is a significant difference in their performance between a PDA and a desktop PC, we will vary inter-packet delay from none to 10 ms to take it into account.

Users of high-speed network had felt the limitations of Ethernet's small frame sizes and was the consumers of Jumbo Frames - extended Ethernet frames that range in size from the standard 1,500 bytes up to 9,000 bytes [9]. In addition the packet size becomes bigger until the maximum segment size (MSS, 1,460 bytes), the TCP performance becomes better. Therefore we propose data transmission using various packet sizes bigger than MSS in consideration of high-speed networks within a near future.

Much of the works did their experiments based on UNIX or Linux Operating Systems [2] [4], whereas we have done all of our experiments based on Windows 2000/CE.

As mentioned above Figure 1, TCP has socket buffers in a transport layer and application buffers in an application layer. In Table 1, the default TCP socket buffer means both a socket send buffer and a socket receive buffer.

3 Performance Metrics for Evaluating Multimedia Data Transmission in WLAN

3.1 A Test-bed Infrastructure Network Based on WLAN

We have designed and implemented VMS (Voice Messenger Systems) as shown in Figure 3 to measure the performance of multimedia data transmission. VMS is an infrastructure network that integrates a wired LAN, based on Ethernet with a WLAN based on the IEEE 802.11b standard. The WLAN contains only one BSS (Basic Service Set) for simplicity.

VMS is a kind of file transfer system, which consists of a VMS server and VMS clients. A desktop PC and a PDA represents FH (Fixed Host) and MH (Mobile Host), respectively. One of the desktop PCs is used as the VMS server. The rest of them and all PDAs are used as VMS clients. PDA with a PCMCIA NIC can be connected to a wired LAN through a BS (Base Station). In our test-bed network, the BS is simply an AP as shown in Figure 3.

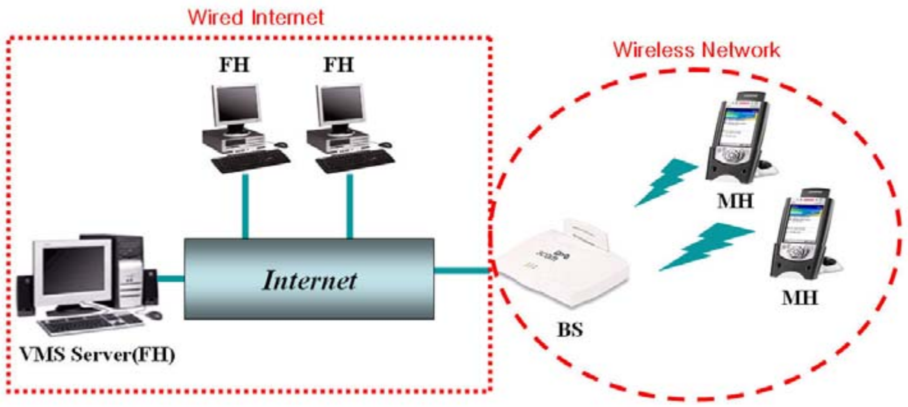


Figure 3 A test-bed infrastructure network based on WLAN

Let us briefly explain how a VMS works: Each VMS client records one's voice and sends it to the VMS server after converting it into a wave file format. The server receives this voice message sent from its client and stores it in its hard disk. It transfers this message to the authenticated client that requests it.

The hardware specification of the hosts used in the VMS is listed in Table 2.

Table 2 the hardware specification of the hosts used in the VMS

Host type		CPU	RAM	NIC
MH	PDA	Strong Arm(206MHz)	64MB	PCMCIA (11Mbps)
FH	PC	Pentium 4(2.4GHz)	512MB	PCI (100Mbps)

Before we discuss about performance metrics, we should define the terminology to be used: PDA upstream is a process of transferring data from a PDA as a MH VMS client to the server. During this process, the PDA client reads and sends its internal data and then the server receives and writes them into its file system. On the contrary, we call a process of receiving data for a PDA client from the server PDA downstream. It moves in an opposite direction to the upstream.

PDA sending time is the time that has elapsed to complete data transmission during the upstream. PDA receiving time is the time that has elapsed to complete data transmission during the downstream. Notice that when measuring the performance of data transmission, we located MH VMS clients, i.e. PDAs, within a 5-meter radius of an AP to maintain both good signal quality and strength, while keeping away from multi-path and fading effects.

3.2 The Preliminary Factors Considered for Performance Evaluation

3.2.1 Round Trip Time

PING program running on Linux systems gives us RTT (Round Trip Time) which allows to predict the transmission delay and PER (Packet Error Rate) of a given network. We specify the destination address of PING program as an IP address of a PDA client. The 14 different sizes of a packet transmitted were applied as shown in Figure 4. For each packet size the average of RTTs for 200 repetitions has been calculated.

With the same metrics as a wireless network, we did the same experiments for a wired network as summarized in Figure 5. The experimental results show that RTT is proportional to the size of packet to be sent both a wireless and a wired network. The shortest mean RTTs of a wired and a wireless network is 0.22 ms and 4ms, respectively. Besides, the longest one is 1.61ms and 31ms, respectively, for a wired and a wireless network. The mean RTT of a wireless network is at least 18 times longer than that of a wired network.

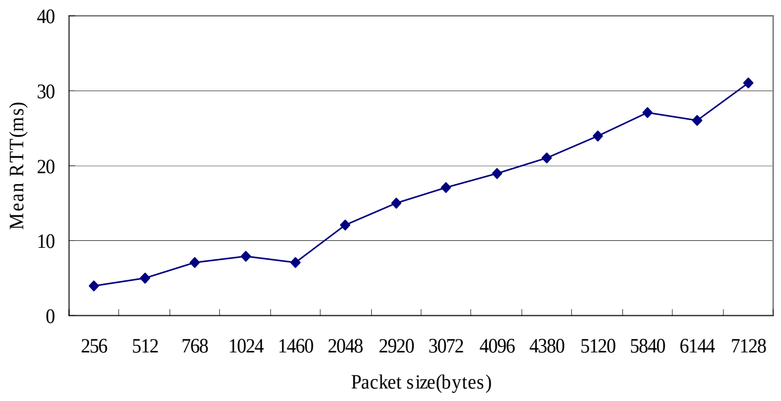


Figure 4 the mean RTT over a wireless network

In another words, in a wired network a network delay does not affect an overall performance even when the size of a packet becomes larger. However, in a wireless network the occurrence of packet retransmission may happen frequently because the signal strength gets weak due to the multi-path and fading effects. Thus, the bigger the size of a packet to transfer data in a wireless network, the more delays in the network there will be.

However, the occurrence of packet loss is hardly generated in a wired and a wireless network either. We come to a conclusion that the size of a packet has almost nothing to do with the PER in a wireless network. *Nguyen* et. al.[4] presented the similar results as ours that the PER is less than 0.001 within 5m radius of AP.

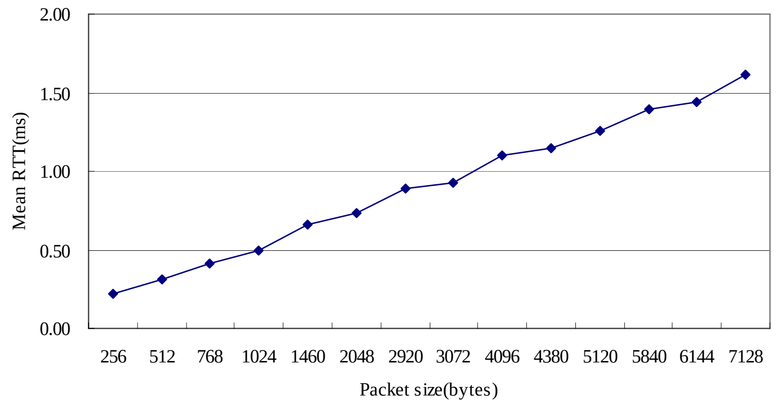


Figure 5 the mean RTT in a wired network

3.2.2 Packet size

To find the packet size at the application layer that gives the best performance of transmission time, we varied it from 512 bytes to 7,128 bytes at application layer, not at transport layer. Figure 6 shows the PDA sending time and the PDA receiving time measured while varying its packet size, when the file size transmitted was about 469 Kbytes. A transmission error occurred when the packet size was equal to or larger than 8,192 bytes because the default size of the TCP send/receive buffer in Windows Operating Systems is set to that value. Thus, in this paper the largest packet size at application layer was restricted to 7,128 bytes.

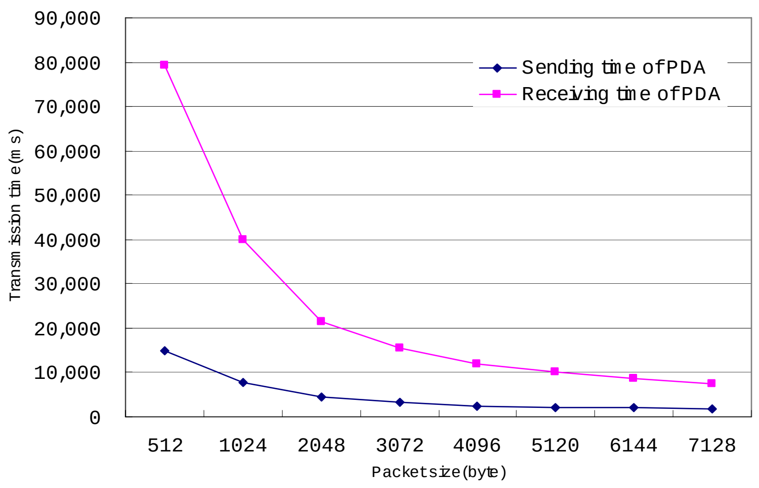


Figure 6 the transmission time measured while varying packet sizes when the file size transmitted is 469Kbytes

Figure 6 shows that as the packet size becomes larger, the corresponding transmission time becomes faster. That is, as a file to be transmitted is divided by a larger packet size, both the total number of data packets segmented and the number of ACK packets get smaller. As a result, with a larger packet size its transmission time becomes faster, even though such a large packet should be needed more RTT.

3.2.3 The sending time versus the receiving time

As you see in Figure 4, the PDA receiving time is always slower than the PDA sending time. For more thorough analysis, the ratios of the receiving time to the sending time for PDA client with the file size are listed in Table 3 when the packet size is set to 4,096 bytes. Recording one's voice for 30 to 600 seconds generated the sample wave files. The file size varies from 235 Kbytes to 4,688 Kbytes. For the purpose of comparison, the same ratios for the desktop PC client are also listed in Table 3.

Table 3 A comparison of the sending time with the receiving time when the packet size is set to 4,096 bytes

Recording time (seconds)	File size (KB)	The ratio of receiving time to sending time	
		PC client	PDA client
30	235	1.0	1.53
60	469	1.0	1.63
120	938	1.0	1.66
300	2,344	1.0	1.70
600	4,688	0.99	1.70

For the desktop PC client, the receiving time is almost the same as the sending time independent of the file size transmitted. However, for the PDA client, the receiving time is longer than the sending time by 53% to 70%. Notice that the ratio increases as the file size becomes larger.

3.3 Performance Metrics of PDA Downstream

As we see in Figure 6, the time that elapsed during the PDA downstream is longer than during the PDA upstream. Especially, as the data to be transmitted become larger, the receiving time is much longer than the sending time (Table 3). Thus, our main concern is that all possible factors that cause the receiving time to be delayed during PDA downstream should be listed and analyzed. Then we will propose a method to speed it up.

Figure 7 shows a trace of the PDA upstream that sends the data from a PDA to the VMS server to examine the behaviour of TCP over a wireless LAN by using Analyzer as a tool for analyses of the Network [10]. Once a PDA client receives an ACK flag from the server, it sends two consecutive packets to the server. In that case, the congestion window size should be set to 2. Notice that the congestion window size remains constant during the upstream process. The number on the left of the figure represents an elapsed time in seconds. In addition, the number in parentheses indicates a time difference between two adjacent packets in the sequence.

Figure 8 shows a trace of the PDA downstream that receives data for a PDA from the server. The second packet updates its window size. As we indicated in Figure 1(b), the alternating increase and decrease in the congestion window size happens. The reason why is that a desktop PC and a PDA acts as a fast sender and a slow receiver, respectively, due to a substantial difference in their computational capabilities. This is clear from Figure 8 that the server sends three consecutive packets to the PDA client within 0.000094 seconds, whereas a PDA client sends the ACK packet after 0.01 seconds.

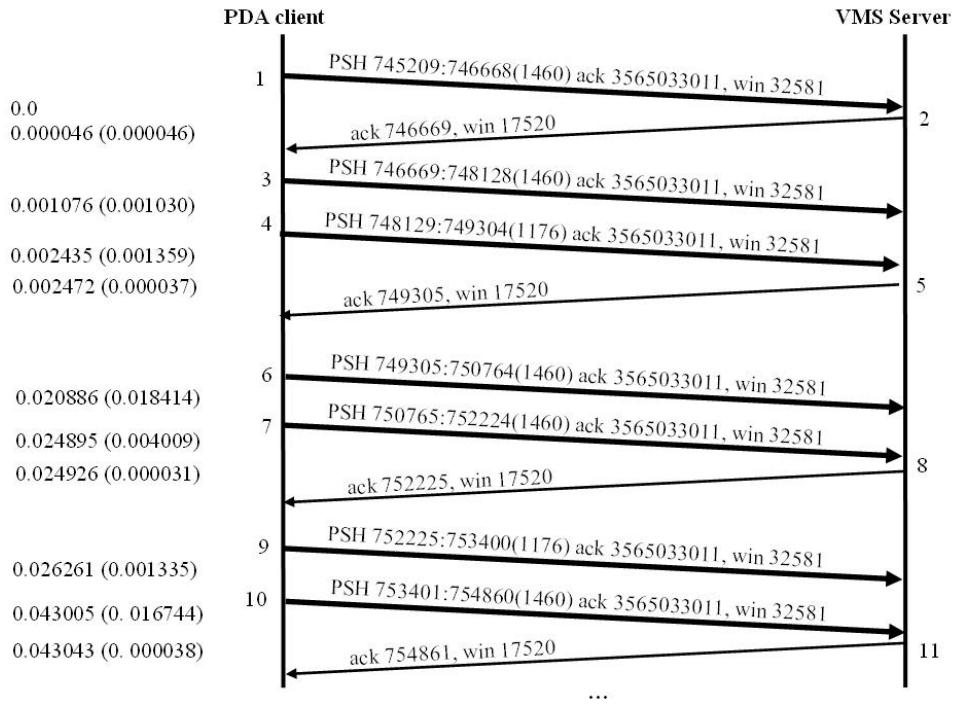

PDA client
VMS Server
1
PSH 745209:746668(1460) ack 3565033011, win 32581
0.0
2
0.000046 (0.000046)
ack 746669, win 17520
3
PSH 746669:748128(1460) ack 3565033011, win 32581
0.001076 (0.001030)
4
PSH 748129:749304(1176) ack 3565033011, win 32581
0.002435 (0.001359)
0.002472 (0.000037)
5
ack 749305, win 17520
6
PSH 749305:750764(1460) ack 3565033011, win 32581
0.020886 (0.018414)
7
PSH 750765:752224(1460) ack 3565033011, win 32581
0.024895 (0.004009)
0.024926 (0.000031)
8
ack 752225, win 17520
9
PSH 752225:753400(1176) ack 3565033011, win 32581
0.026261 (0.001335)
10
PSH 753401:754860(1460) ack 3565033011, win 32581
0.043005 (0. 016744)
0.043043 (0. 000038)
11
ack 754861, win 17520
...

Figure 7 the trace during the PDA upstream using the Analyzer (when the packet size is set to 4,096 bytes)

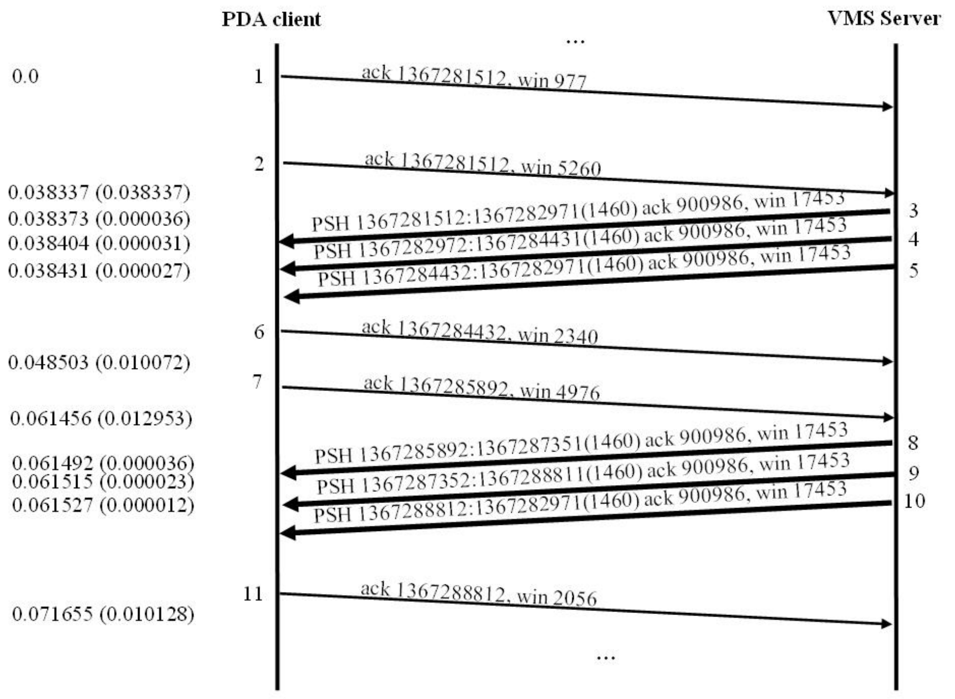


Figure 8 the trace during the PDA downstream using Analyzer (packet size: 4,096 byte)

Thus, the performance metrics that we should consider are the following: TCP window size, file access time of PDA, and inter-packet delay between packets.

3.3.1 Window size

Window size is the value that a receiver advertises to a sender on receiving a request to begin data transmission from the sender. It is a 16-bit field, limiting the window to 65,535 bytes [6]. The receiving process can usually control the size of the window offered by the receiver. This can affect the TCP performance.

Equation (1) shows the relation between the window size and the rate of packet transmission [9].

$$X(t) = \frac{W(t)}{RTT} \tag{1}$$

In Equation (1), X(t) is the rate of packet transmission at time t and W(t) is the window size at time t. RTT is an abbreviation of round trip time. It represents the status of transmission delay in a given network. Its unit is second.

X(t) is inversely proportional to RTT, whereas it is proportional to W(t). That is, as the window size is bigger, the rate of packet transmission increases. The Analyzer [10] has been used to trace the variation of TCP window size during the upstream and the downstream.

3.3.2 File access time of a PDA

As shown in Table 2, there are distinct differences between PDAs and desktop PCs in the throughput of data processing due to the substantial gap of the H/W and the firmware processing times [4].

PDA uses memory chips as storage devices instead of the hard disc. With an iPAQ 3660 model chosen as a PDA client in the VMS, it consists of 16 MB flash memory and 64 MB SDRAM [7]. A program buffer allocated by the VMS is to be stored into SDRAM. On the other hand, a data file is to be stored into the flash memory. When a file is read from the PDA, it reads from the flash memory and stores into the buffer that resides in the SDRAM. On the contrary, when a file is written in the PDA, it writes to the SDRAM temporarily and stores into the flash memory.

Comparing the file access time during the data transmission between a desktop PC and a PDA, we analyzed it to see how it can affect the overall performance.

3.3.3 Inter-packet delay

In the application layer, an inter-packet delay is the time interval between when a previous packet was sent and the time that the current packet was first sent. In general, the transmission time becomes faster with a shorter inter-packet delay. However, if there is a distinct difference in processing capability between sender and receiver, inter-packet delay at normal should be adjusted to give enough time intervals to complete its internal processing for low-end devices. Therefore, it is necessary to consider it more carefully.

4 The Experimental Results

We have performed our experiments to measure the above metrics and that are to be used to find a way to improve the receiving time during the downstream. In these experiments, a packet size of 4,096 bytes at the application layer was used.

4.1 The Variation in the Window Size

During the upstream, the window size of a PDA and a desktop PC was 32,581 bytes and 17,520 bytes, respectively. Both of them were kept constant. During the downstream, the window size of a desktop PC was 17,453 bytes which is nearly the same as before, whereas the window size of a PDA was reduced remarkably to 686 bytes from 32,581 bytes (Table 4).

Table 4 the least value of TCP window size when the packet size is set 4,096 bytes (unit: bytes)

PC window size		PDA window size	
Upstream	Downstream	Upstream	Downstream
17,520	17,453	32,581	686

Due to the substantial differences in performance between them, during the downstream a desktop PC and a PDA represents the characteristics of fast sender and slow receiver, respectively [6]. Data transmission time and ACK response time measured in both the upstream and the downstream are shown in Figure 9. The sum of the number of data packets and the number of ACK packets is 101.

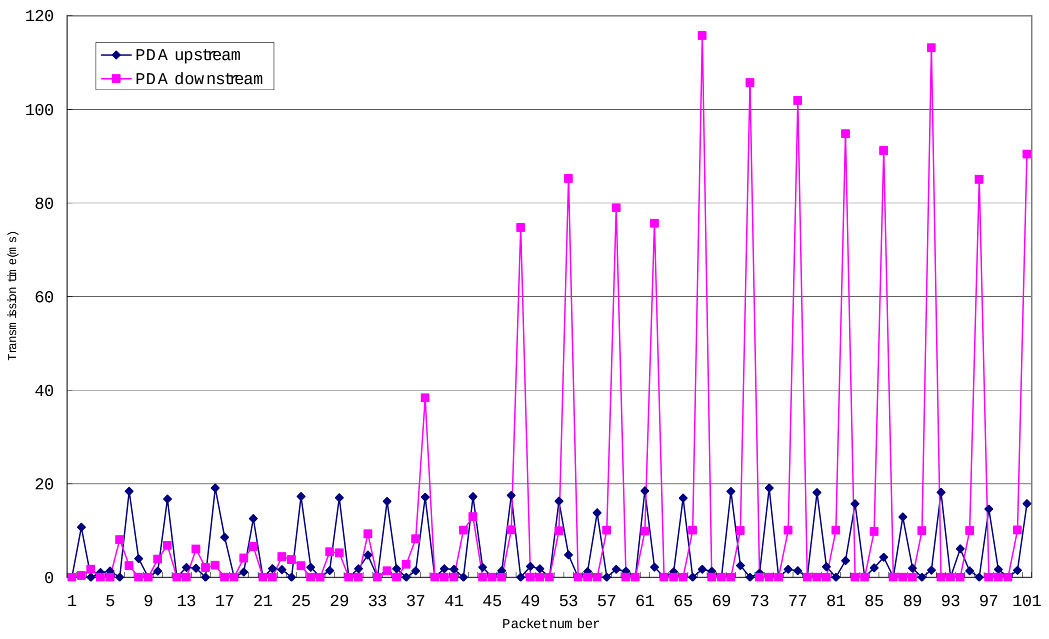


Figure 9 Data transmission time and ACK response time

When the packet size is 4,096 bytes, the number of packets at the transport layer will be 3 that consist of two of 1,460 bytes and one of 1,176 bytes. In the PDA upstream, every time the first segment is to be sent, its transmission time will be delayed. However, because the time to send it will not exceed 20 ms, the window sizes of both a PDA and a desktop PC will be kept constant.

In the case of the PDA downstream, the first time it will be more stable than the upstream. But, after the transmission of the 38th packet the response time of ACK packet will be delayed and then the window update packet should be sent. It is the ACK packet that informs the updated window size at receiver to sender. Thus, the transmission time will be increased dramatically by up to 115ms. This phenomenon will occur frequently until the receiving process is completed. This is the reason why the receiving time is delayed.

The problem has been occurred because the time that has elapsed to receive data from the socket send buffer of the desktop PC (*fast sender*) and then process it in the socket receive buffer of the PDA (*slow receiver*) was delayed. Thus, to improve the performance during the downstream, it is necessary to increase both the size of the send buffer of the desktop PC and the size of the receive buffer of the PDA.

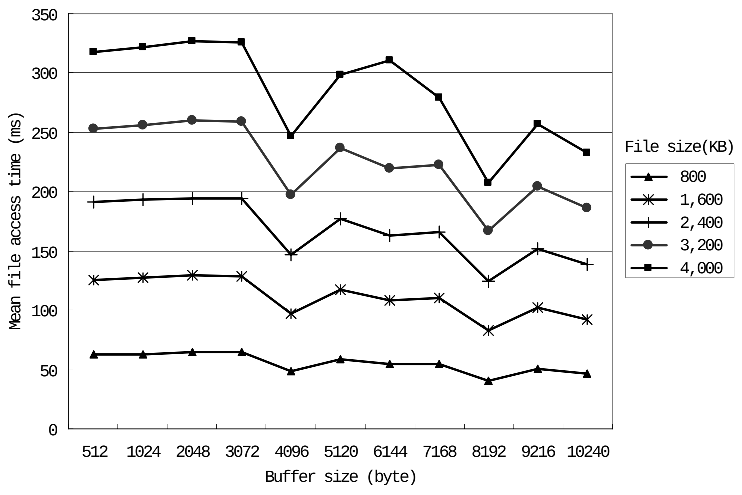


Figure 10 the mean file access time with the pre-specified buffer size during the PDA upstream

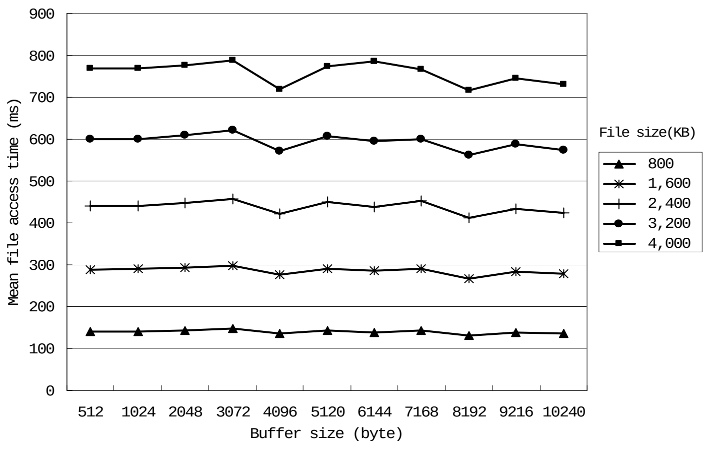


Figure 11 the mean file access time with the pre-specified buffer size during the PDA downstream

4.2 File Access Time

By varying the size of an application buffer from 512 bytes to 10,240 bytes, we measured the file access times of both the PDA upstream and the PDA downstream. Five sample files with different sizes were used. We measured it over 100 times with each specified buffer size and took the mean of them. The mean file access time during the upstream and the mean file access time during the downstream are shown in Figure 10 and Figure 11, respectively. The reason why the PDA downstream takes 2 times longer than the PDA upstream is that the time taken to write data into the flash memory takes 2 times longer than the time taken to read data from it. When the buffer size is set to 8,192 bytes, the comparison of the file access times of the PDA upstream and the downstream with different file sizes is summarized as shown in Table 5. As the file size increases, the file access time during the PDA downstream becomes longer than that during the upstream.

Table 5 the comparison of the file access times of the PDA upstream and the downstream with various file sizes

File size(KB)	Upstream(ms) (A)	Downstream(ms) (B)	Ratio (B/A)
400	21	64	3.11
1,200	62	198	3.19
2,000	104	337	3.26
3,200	167	563	3.37
4,000	207	718	3.46

As the size of an application buffer both at the sender and at the receiver increases, the mean file access time becomes shorter. Particularly, independent of the type of the process, the file access time is the fastest when the size of the application buffer is set to 8,192 bytes. It is faster than by 10% when the buffer size is set to 3,072 bytes which is the slowest.

4.3 Adjustments of Inter-Packet Delay

An inter-packet delay can be adjusted by using the sleep function, which is one of Visual C++ 6.0's libraries [11]. Its unit is millisecond (ms). We set it to 1, 10, and 100 ms by using that function and no delay (i.e., no call to it). Transmission times during the upstream and the downstream were measured with the predetermined delay and summarized in Table 6. Notice that the same delay was applied to both. In this experiment, the packet size was set to 4,096 bytes and a sample file of 469 KB was used.

Table 6 Measurements of the transmission time with its inter-packet delay (unit: ms)

Inter-packet delay	Transmission time (avg.) during upstream	Transmission time (avg.) during downstream
None	13,656	4,622
1	8,163	12,559
10	2,249	12,629
100	13,004	21,460

Assuming that the devices with the same performance are used, it is generally true that their transmission time becomes faster with a shorter inter-packet delay. However, because we are using devices (a PDA and a desktop PC) with great differences in their performance, a shorter delay does not guarantee higher performance.

Table 7 shows that during the upstream the transmission time in average is the fastest when it was set to 10 ms. The reason why is that the socket receive buffer (its size, 8,192 bytes) of the desktop PC is less than the socket send buffer (16,384 bytes) of the PDA by a ratio of 2:1. Thus it is necessary for the desktop PC to give enough time to maintain the balance of the processing time due to its smaller socket buffer size. In other words, if we increase its size to the same size as that of the PDA, an overall throughput will be enhanced. During the downstream the best performance was achieved when applying no inter-packet delay.

Table 7 the comparison results of with and without applying the proposed method (unit: ms)

	Without applying the method				With applying the method			
Process type	Upstream		Downstream		Upstream		Downstream	
Direction of transmission	Send at PDA	Receive at PC	Send at PC	Receive at PDA	Send at PDA	Receive at PC	Send at PC	Receive at PDA
Inter-packet delay	1	10	1	10	10	10	None	None
Transmission time	2,502		11,993		2,249		4,622	

From this analysis, we will get a better performance by applying distinct inter-packet delay to each process. Without applying the proposed method, an inter-packet delay of 1 ms and 10 ms was applied, thus affecting when to send to the PDA (or PC) and when to receive to the PDA (or PC) as is shown in Table 6, respectively. In the proposed method taken here, the inter-packet delay of 10ms was applied to both when sending to the PDA and when receiving to the PC during upstream. But, no delay was applied to both of them during the downstream. During the downstream the receiving time with applying the method is decreased by 61% compared to that without applying it. Notice that the transimission time during the upstream shows little change between with and without applying it.

5 Conclusions

We measured a performance of multimedia data transmission between a PDA as a mobile host and a desktop PC as a fixed host in an infrastructure network. To evaluate such a performance more precisely, a test-bed system called VMS that adopted TCP as a transmission protocol was built.

In the case of a WLAN environment that provides multimedia-based services, a PDA as a mobile terminal is used mainly to receive a large size data from its stationary server. However, for the PDA client of the VMS, the receiving time is longer than the sending time by 53% to 70%. Notice that the ratio increases as the size of the file to be transmitted becomes larger. For a desktop PC client, the receiving time is almost the same as the sending time independent of its file size.

There are crucial factors: file access time of a PDA and TCP window size that can affect the receiving time of a PDA. Since the time to write data into a flash memory of a PDA takes 2 times longer than the time to read data from it, the PDA downstream takes 2 times longer than the PDA upstream. To shorten it, the size of an application buffer that is implemented by an application program should be enlarged depending on the memory structure of a PDA. It will be shorter by 10% by doubling the size of the buffer. In our experiment, the buffer size of 8,192 bytes is the fastest.

During the upstream, both the window sizes of a desktop PC and a PDA are kept constant. However, during the downstream a desktop PC and a PDA represent the characteristics of a fast sender and a slow receiver, respectively. That is, the response time of an ACK packet is to be delayed and then the window update packet should be sent. Therefore, an overall transmission time will be increased dramatically. To alleviate such a delayed response time of ACK packet, both the size of the socket send buffer of a desktop PC and the size of the socket receive buffer of a PDA should be enlarged as possible. With the larger size of the buffer, the window size at the receiver will remains stable due to a faster processing.

An inter-packet delay should be adjusted to give enough time intervals to complete its internal processing for low-end terminals like as PDAs. Thus, we got a better performance by applying distinct inter-packet delays to each process. During the downstream the receiving time with applying the proposed method is decreased by 61% compared to that without applying it.

Acknowledgement

This work was supported by University of Incheon Research Fund in the Republic of Korea.

References

1. Nespot web site, http://www.nespot.com
2. Balakrishnan, H., Seshan, S., Amir, E., Katz, H.: Improving TCP/IP Performance over Wireless Networks, ACM MOBICOM (1995)
3. Hur, H-S., Hong, Y-S.: Performance Analysis of Multimedia Data Transmission with PDA over an Infrastructure Network, ICCSA (2004)
4. Nguyen, G. T., Katz, R. H., Noble, B., Satyanarayanan, M.: A Trace-Based Approach for Modeling Wireless Channel Behavior, In Proceedings of the Winter Simulation Conference pp. 597-604 (1996)
5. Pilosof, S., Ramjee, R., Raz, D., Shavitt, Y., Sinha, P.: Understanding TCP fairness over Wireless LAN, IEEE INFOCOM (2003)
6. Stevens, W. R.: TCP/IP Illustrated – Volume 1: The Protocols, Addison-Wesley (1994)
7. Muench, S.: The Windows CE Technology Tutorial: Windows Powered Solutions for the Developer, Addison Wesley (2000)
8. Postel, J.: Transmission Control Protocol, RFC 793 (1981)
9. Hassan, M., Jain., R.: High Performance TCP/IP Networking: Concepts, Issues, and Solutions, Pearson Prentice Hall (2004)
10. Analyzer web site, http://analyzer.polito.it
11. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/sleep.asp
12. Karadia, D.: Understanding Tuning TCP, Sun Microsystems, Inc., http://www.sun.com/blueprints (2004)
13. Iannello, I., Pescapè, A., Ventre, G., and Vollero, L.: Experimental Analysis of Heterogeneous Wireless Networks, WWIC (2004)
14. Tierney, B. L.: TCP tuning guide for distributed application on wide area networks, ;login: The magazine of USENIX & SAGE, Vol. 26, No. 1, pp. 33-39 (2001)
15. Botta, A., Emma, D., Guadagno, S., and Pescapè, A.: Performance Evaluation of Heterogeneous Network Scenarios, Technical report (2004)
16. Kung, H. T., Tan, K-S., and Hsiao, P-H.: TCP with Sender-Based Delay Control, ISCC'02 (2002)
17. Matsushita, Y., Matsuda, T., and Yamamoto, M.: TCP Congestion Control with ACK-Pacing for Vertical Handover, IEEE WCNC 2005, pp. 1497-1502, (2005)

18. Zheng, B. and Atiquzzaman, M.: A Novel Scheme for Streaming Multimedia to Personal Wireless Handheld Devices, IEEE Transcations on Consumer Electronics, Vol. 49, No. 1, (2003)
19. Mäntyjärvi, J., Kallio, S., Korpipää, P., Kela, J. and Plomp, J.: Gesture Interaction for Small Handheld Devices to Support Multimedia Applications, Journal of Mobile Multimedia, Vol. 1, No. 2, pp. 92-111, (2005)
20. Endres, C., Butz, A., and MacWilliams, A.: A Survey of Software Infrastructure and Frameworks for Ubiquitous Computing, Mobile Information Systems, Vol. 1, No. 1, pp. 41-80, (2005)
21. Maniatis, A., Vassiliadis, P., Skiadopoulos, S., Vassiliou, Y., Mavrogonatos, G., and Michalarias, I.: A Presentation Model & Non-Traditional Visuaization for OLAP, International Journal of Data Warehousing and Mining, Vol. 1, No. 1, pp. 1-36, (2005)