A Multipath Approach for Improving Performance of Remote Desktop Transmission

Cao Lethanhman, Hiromi Isokawa, and Takatoshi Kato
Systems Development Laboratory, Hitachi Ltd.
Ohzenji 1099, Asao-ku, Kawasaki-shi, Kanagawa 215-0013 Japan
{lethanhman.cao.eq, hiromi.isokawa.yt, takatoshi.kato.bb}@hitachi.com

Abstract—Remote desktop systems enable business users to access critical data located in the office from outside it with a mobile thin client or any other portable ubiquitous devices without having to worry about data leakage. However, high latency and low bandwidth in wireless-network environments may degrade the performance of remote desktop clients, such as causing slow responses to keyboard/mouse input and low speed in playing animated presentations. We therefore propose a method of transmitting data to improve the way business applications are perceived in wireless thin clients. It utilizes multiple wireless-network paths to achieve short latency in transmitting input signals and receiving responses as well as increases the throughput of large desktop images. We evaluated the performance of the proposed method using a network simulator and a real network environment. The experimental results revealed that the round trip time of a small data packet was shorter and the throughput of large amounts of data was higher than those with single network paths or existing multipath transmission algorithms.

Keywords - remote desktop; multipath transmission; wireless network; thin client

I. INTRODUCTION

All data and applications are centralized in a server, which provides a remote desktop service in remote desktop systems, such as the Citrix XenDesktop [1], Microsoft Remote desktop service [2], X-Window System [5] and AT&T VNC [7]. The remote desktop client sends input signals from the keyboard/mouse to the server and receives differences in desktop images from the server. The remote desktop client requires neither a large amount of CPU power nor large amounts of data storage, and is, therefore, usually a thin client, i.e., a resource-poor computer whose components are limited to network interfaces and input/output devices. Because the thin client does not have any confidential data stored on its local hard disk, business users can use it outside the office without having to worry about losing important data. As information leakage has been an increasingly serious problem for many enterprises, more thin clients and remote desktop systems are currently rapidly emerging.

Thin clients depend heavily on the performance of networks and remote desktop protocols have been designed for reliable networks. However, a business user who works outside the office would use a mobile thin client in a high latency, high packet loss ratio and low bandwidth wireless network environment. In these cases, there are often gaps between the requirements of remote desktop protocols and the capabilities of wireless-network services. The result is that wireless thin clients perform badly and may not reach a sufficiently acceptable level of service to be used for fast typing or presenting an animated presentation.

We improved the quality of data transmission performed by a wireless thin client using a method of multipath data transmission, where the thin client simultaneously deploys two or more wireless-network access attempts to communicate with the server. Unlike existing multipath transmission algorithms that can only be used for reducing transmission latency [13] (duplicate-and-forward algorithm) or aggregating network bandwidths [14, 15] (divide-and-forward algorithm), our proposed algorithm, called the data-aware multipath transmission (DAMT) algorithm, dynamically changes from duplicate-and-forward to divide-and-forward according to the size of the forwarded data. The DAMT algorithm can accomplish short latency in transmitting input signals and their responses and simultaneously increase the throughput of large desktop images.

We evaluated the DAMT algorithm in many network conditions created by a network simulator. The results revealed that when the thin client leveraged two network connections using the DAMT algorithm, the round trip time of a small data packet is smaller and the throughput of data is higher in comparison with single network paths or existing multipath transmission algorithms.

We also evaluated the DAMT algorithm in a real network environment. We found that a thin client equipped with one Worldwide Interoperability for Microwave Access (WiMAX) and one High Speed Downlink Packet Access (HSPA) data card could respond to key input faster while a video file was being played with a lower dropped-frame rate when it transmitted multipath data using the DAMT algorithm, compared with when only a WiMAX or an HSPA data card was used.

The remainder of this paper is organized as follows. Section 2 explains limitations with wireless thin clients. Section 3 discusses some related research. We then introduce the DAMT algorithm in Section 4 and discuss its implementation in Section 5. Sections 6 and 7 present the results from evaluating the DAMT algorithm. Section 8 concludes the paper and mentions future work.

II. LIMITATIONS WITH WIRELESS THIN CLIENTS

This section first discusses the characteristics of data that are sent and received by a thin client and it then points out the reason for the degradation in performance of thin clients in wireless network environments.

A. Characteristics of traffic in thin clients

We first investigated the characteristics of data sent and received by thin clients in remote desktop systems. Figure 1 shows the size of data that was received by a thin client when it used Microsoft's Remote Desktop Protocol (RDP) to download desktop information from a remote desktop server. As we can see, the data includes two types of data blocks, small and large. The small blocks occur when a user is typing and the large ones occur when he/she is playing a presentation. That reason for this is that when the user presses a key on the keyboard, the change in the desktop image is much smaller than when he/she plays a presentation on the remote desktop server. The desktop server only sends differences in desktop images to thin clients, instead of the entire image. Thus, the data received by the thin client in the remote desktop system can be in the form of both small and large data blocks.

We next examined the data that a thin client sends to the remote desktop server while the user is operating the thin client. As shown in Figure 2, the data only compounds small blocks, regardless of the kinds of applications the user is using. This is because the thin client only sends the input signals from the keyboard/mouse to the server.

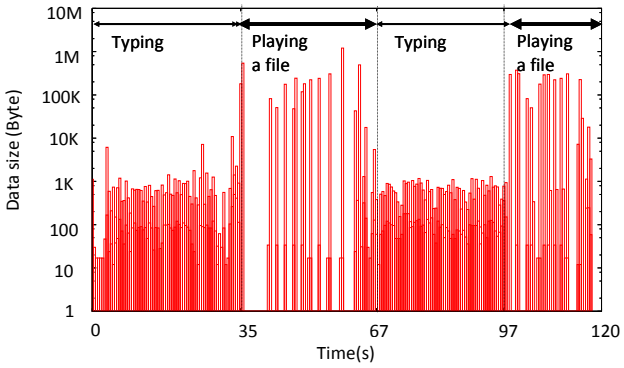

Figure 1: Data received by thin client when using RDP

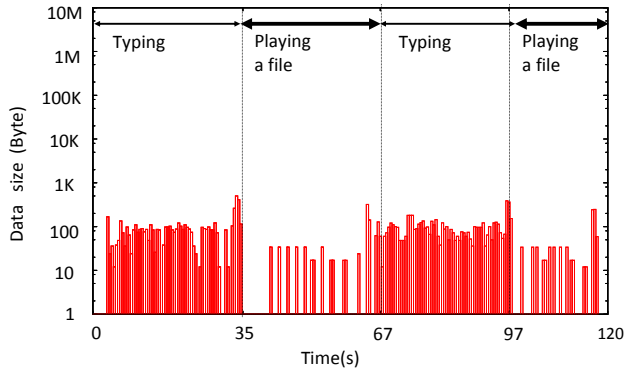

Figure 2: Data sent by thin client when using RDP

From the above-mentioned results, we can conclude that the data that are sent and received by thin clients generally include two different-sized data blocks.

1. Small data blocks (several KB): These kinds of data often appear with keyboard/mouse input. The thin client sends small-capacity keyboard/mouse input signals to the server. The server then sends small differences in desktop images (that occur when the user is typing) to the thin client as the responses to the input.
2. Large data blocks (several MB): These kinds of data appear when desktop images change dramatically, such as when animated presentations are played, desktops are scrolled, or new documents are opened.

B. Limitations with wireless thin clients

Thin clients have two main limitations with above-mentioned two characteristics of transmission data, in wireless-network environments with long latency and narrow bandwidth:

- Long latency in the transmission of small data blocks. This causes extended responses to keyboard/mouse input. The user will feel difficult typing a document because the stress because the key pressed does not appear on the screen immediately.
- Low throughput of large data blocks. This causes long delays in the time it takes to download desktop images and therefore decreases the speed at which animated presentations are played. The user of thin client will find difficult browsing web pages with animated pictures.

We will next discuss some related research on improving how applications are perceived in thin clients.

III. RELATED RESEARCH

A great deal of research related to improving the QoS of remote desktop systems has been conducted thus far. Some of this work [8, 9] has introduced the concept of localization within the context of the thin client computing model and its ability to be applied to wireless and mobile environments. Their work focused on decreasing the number of short yet frequent exchanges of communication between the thin client and the server to solve the problem of high-latency networks. However, it did not deal with the problem of narrow-bandwidth networks that occurs when the thin client transmits large data blocks. Other researchers [10, 11] have proposed QoS-aware mechanisms in remote desktop protocols. Their solution was able to compensate for temporary decreases in network performance or disruptions in transmission between a client and a server. Optimizing the use of resources by making remote desktop protocols QoS-aware, introduces efficient control over differentiated application-level traffic. Their work was similar to Distortion-based Packet Marking [12], an on-the-fly relocation of network resources in a shared link for multiple data flows with different QoS requirements. However, both the mechanisms worked on a single network path and

therefore could not exceed the physical limitations of the network.

Various multipath approaches have thus far been proposed to achieve higher rates of data transmission than those in a single network. Some studies [12] have improved the robustness of data transmission by sending the same data on multiple network paths. This approach is effective in networks with high packet-loss ratios. Others have increased the throughput of data transmission by sending different parts of the data over different network path simultaneously [14, 15]. However, as we will explain in the next section, existing multipath algorithms are not suitable for data transmission in remote desktop systems that have the characteristics described in Section 2.

We proposed a multipath transmission algorithm for remote desktop systems and evaluated the performance using network simulator in [6]. This paper describes the algorithm in more details and provides some experiment results in the real network environment.

IV. PROPOSED METHOD OF DATA TRANSMISSION

We introduce a multipath approach to improve the data transmission of remote desktop systems. This section first introduces a TCP proxy that transmits multipath data in a remote desktop system. We next discuss our evaluations of the existing multipath transmission algorithm and then introduce our proposed algorithm.

A. Overview

We inserted a TCP proxy between the thin client and the server on both sides. The proxy programs were called multipath communication control proxies (MCCPs). The MCCPs received the data from remote desktop software and forwarded them over multiple wireless-network connections.

Figure 3 outlines an example of the placement of MCCPs in remote desktop systems. The MCCPs do not require any changes in the OS components of the server, thin client, or the remote desktop protocol. The remote desktop server passes the data to the MCCP in the server (server-side MCCP) and then the server-side MCCP sends the data over Network A and Network B to the MCCP thin client (client-side MCCP). The Client-side MCCP then passes the data it received to the remote desktop client. Thus, the MCCPs simultaneously deploy two networks (networks A and B) to send and receive data exchanged between the thin client and server.

The server-side MCCP can be installed outside the remote desktop server. As seen in Figure 4, it can work in a gateway. The gateway receives data from the server and forwards the data to the thin client over two networks using a multipath data-transmission protocol. Here, no software needs to be installed in the server while the thin client can perform multipath data transmission.

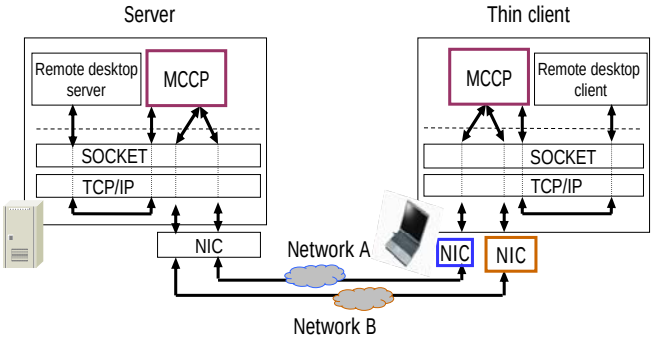

Figure 3: Placement of MCCPs

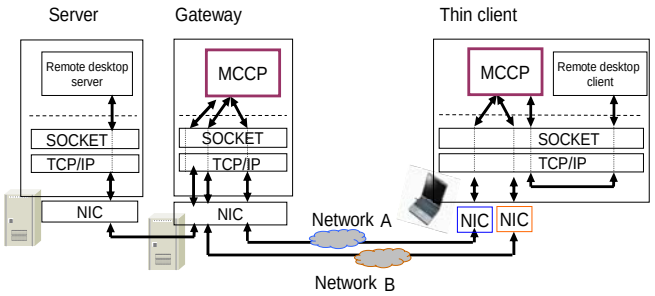

Figure 4: MCCPs with gateway

The client-side MCCPs can also be installed outside the thin client. For example, the client-side MCCP can work in a gateway that redirects the data flow from the thin client to the server. Here, the gateway will transmit the multipath data through the server instead of the thin client. It needs two network interfaces that are connected to two different networks.

B. Existing algorithms for multipath data transmission

We next discuss the algorithms for the MCCPs to send and receive data over multiple network connections. The existing algorithms in the domain of multipath communication can be divided in two types. The first decreases transmission delay and increases the robustness of transmission [13]. Figure 5 outlines the flow of data when the MCCPs deploy this algorithm. When the server-side MCCP receives data from a remote desktop's server program, it duplicates the received data and then forwards them to both networks, A and B.

The client-side MCCP forwards data that arrive sooner to the remote desktop client and disposes of the later. We called this the duplicate-and-forward algorithm. It could transmit a small data block through a network with lower latency but could not improve the throughput especially for a large data block.

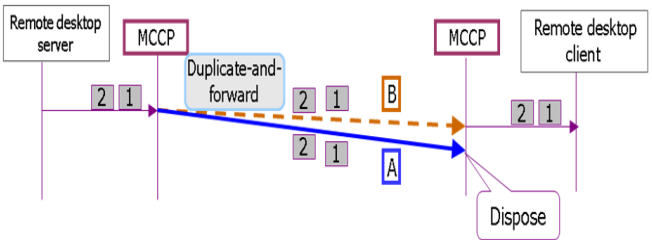

Figure 5: Duplicate-and-forward algorithm

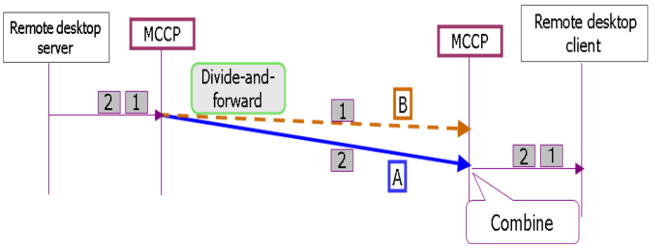

Figure 6: Divide-and-forward algorithm

The second algorithm aggregates the bandwidths of multiple network connections to increase the throughput of data transmission [14, 15]. Figure 6 shows the flow of data when the MCCPs deploy the algorithm. When the server-side MCCP receives data from the remote desktop server program, it divides the received data and then forwards different parts of the data to networks A and B. The client-side MCCP combines the data received and then forwards them to the remote desktop client. We called this the divide-and-forward algorithm. It could improve throughput especially for a large data block but did not decrease the transmission latency of a small data block.

As existing multipath transmission algorithms can only optimize transmission delay or aggregate bandwidths, they cannot be used to solve the above-mentioned two limitations with wireless thin clients.

C. *Data-aware multipath transmission algorithm*

We propose an algorithm called the data–aware multipath transmission (DAMT) algorithm that improves the efficiency of thin clients in wireless environments. The algorithm combines the two existing transmission algorithms: duplicate-and-forward and divide-and-forward. The DAMT algorithm can transmit small data blocks with short latency and large data blocks with high throughput. First, when a data block arrives, an MCCP runs the duplicate-and-forward algorithm. While transmitting the data, the MCCP also monitors the access network of the server. If the access network becomes congested, the MCCP determines that the block that is being transmitted is large. The MCCP then changes the algorithm to divide-and-forward. Thus, the DAMT algorithm dynamically changes from duplicate-and-forward to divide-and-forward according to the size of the data being transmitted. The details on the MCCP algorithm are described in what follows.

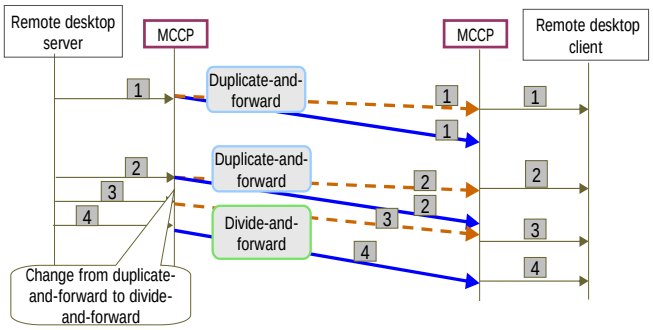

Figure 7: DAMT algorithm

Figure 7 shows the flow of data when the DAMT algorithm is used. First, the server-side MCCP receives data 1 from the remote desktop's server program, duplicates them and forwards them to both networks, A and B. Next, when a large block of data that includes data 2, 3, and 4 arrives, the server-side MCCP also duplicates data 2 and forwards them to networks A and B. Suppose that when data 2 are being transmitted, the server-side MCCP finds that the buffers for the TCP connections are full. The server-side MCCP then switches from duplicate-and-forward to divide-and-forward and performs the latter on data 3 and 4

Sender-side MCCP:

a) Wait until data that need to be transmitted arrive. Go to step *b)* when the data arrive.

b) Perform duplicate-and-forward tranmission while monitoring the buffer for TCP connections (e.g., by checking the return value of the send() function). If the buffer is full, go to step *c)*. Go to step *a)* if data transmission has been completed.

c) Perfom divide-and-forward tranmission on the remaining part of the data block and then go to step *a)*.

Receiver-side MCCP:

a) Wait until data have arrived. Go to step b) when the data arrive.

b) Receive the data. Combine the divided data. Delete the duplicated data. Go to step a) when no more data arrive after a certain interval.

Thus, the DAMT algorithm uses duplicate-and-forward transmission for small-capacity data and divide-and-forward transmission for large-capacity data. This means that interactive data, which are always small capacity, are transmitted with short latency while large desktop images caused by animated presentations are transmitted at high throughput.

V. IMPLEMENTATION ISSUES

This section discusses some issues concerning the implementation of MCCPs. Our prototype was implemented in the Windows XP environment using Winsock for data transmission. We will first explain the sequence for the remote desktop client and remote desktop server to

establish/close the TCP connections for transmitting desktop information. We then explain the functions of the main components in MCCPs.

A. *Establishing/closing sequence for TCP sessions*

The flow chart in Figure 8 outlines the sequence for establishing TCP sessions before the remote desktop client can exchange data with the remote desktop server. Before the TCP sessions are started, the server-side MCCP and client-side MCCP must wait in certain ports for the connection requests. First, the remote desktop client establishes a TCP session with the client-side MCCP (Step 1). If the client-side is installed at the thin client, the remote desktop client connects to a local port where the MCCP is listening. Second, the client-side MCCP establishes two TCP sessions with the server-side MCCP. In Step 2, the client-side MCCP uses network interface NIC A to connect to port X on the server and in Step 3, it uses network interface NIC B to connect to port Y. Thus, the client-side MCCP uses different network interfaces for each TCP session. In Windows, the metric parameter in the routing table of the two network interfaces A and B must have the same value, otherwise only an interface with a smaller metric will be deployed for both TCP sessions because the two sessions will have the same destination.

In Step 4, the server-side MCCP establishes a session (TCP session 4) with the remote desktop server after establishing a session with the client-side MCCP. The server-side MCCP needs to know the port where the remote desktop server is listening. This will be the connection to a local port if the server-side MCCP is located on the same machine as the remote desktop server. After TCP session 4 has been established, the MCCPs start forwarding the data exchanged by the remote desktop client and server using the DAMT algorithm.

The closing sequence for TCP sessions is as follows. When the user finishes work on the thin client and signs out, the remote desktop server will close session 4 in Step 5. The send-side MCCP then closes sessions 2 and 3 right after session 4 has closed. The receive-side MCCP will also close session 1 if one of two sessions, 2 or 3, has closed. The closing sequence will occur in the reserve direction if the user closes session 1 before signing out, e.g., by closing the remote desktop client.

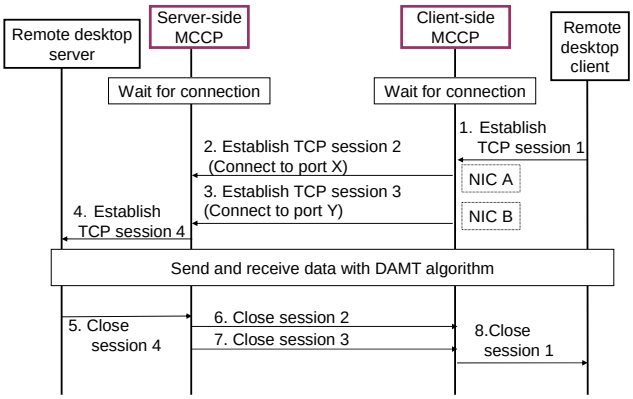

Figure 8: Sequence for establishing TCP session

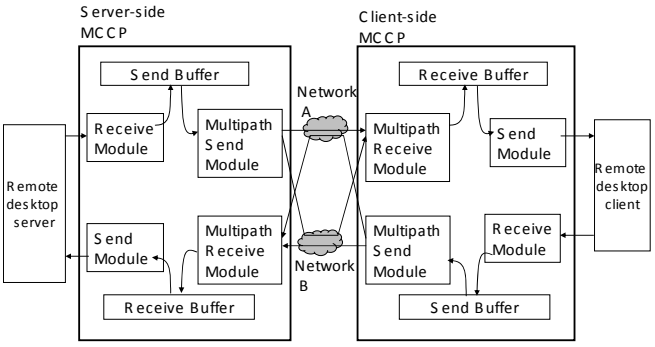

Figure 9: Structure of MCCPs

B. *Structure of MCCP*

We will next explain the components of MCCP and give details on the data processing flow of the components.

The MCCPs on the client-side and server-side have exactly the same functions. These MCCPs have five main components.

1. Data buffers

The MCCP has two buffers for storing data before it forwards them to the networks (Send Buffer) or to the server/client (Receive Buffer).

2. Receive Module

The Receive Module of the server-side MCCP receives data from the remote desktop server and saves the data in the Send Buffer. The Receive Module of the client-side MCCP receives data from the remote desktop client. If the Send Buffer is full, it stops reading data from the socket of the TCP session and waits until some space clears.

3. Send Module

The Send Module of the server-side MCCP sends the data in the Receive Buffer to the remote desktop server. The Send Module of the client-side MCCP sends the data in the Receive Buffer to the remote desktop client.

4. Multipath Send Module

The Multipath Send Module runs the DAMT algorithm. It sends the data in the Receive Buffer to the remote desktop server or remote desktop client over multipath network paths. It will not act if there are no data in the Receive Buffer.

Figure 9: Program data unit (PDU)

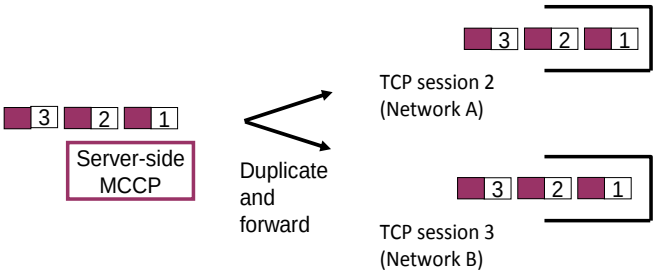

Figure 10: PDU allocation in duplicate-and-forward mode

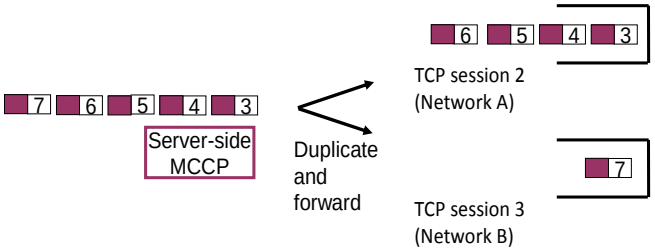

Figure 11: PDU allocation in duplicate-and-forward mode

The Multipath Send Module divides the data in the Send Buffer into small data blocks called Program Data Units (PDUs). Figure 9 outlines the structure of a PDU. Each PDU includes a sequence number (*Seq*), the length of the data (*Len*), and the data (*Payload*). The maximum size of the Payload is 500 KB. The Payload may be smaller than 500 KB if the amount of data remaining in the Send Buffer is less than 500 KB. Here, the module creates one PDU from all the data remaining in the buffer.

The Send Buffer Module continuously monitors the buffer of the two TCP sessions, 2 and 3, using select() commands. When the sessions are available for transmitting more data, the module chooses a PDU in the Send Buffer and sends it over TCP sessions 2 and 3. As previously mentioned, the Send Buffer Module first sends the PDU in duplicate-and-forward mode while monitoring the returned value of the send() function. It will change to the divide-and-forward mode if the send() function return value is smaller than the size of the PDU that the module requested to be sent, because this is the sign indicating that the TCP session's buffer is full. We will now explain how the Send Buffer Module chooses a PDU in the Send Buffer to send to the network.

In the duplicate-and-forward mode, the Send Buffer Module selects the PDU with the following sequence number: N = R + 1, where R is the largest number of PDUs that have already been sent at that time. The module sends the PDU twice, once over network session 2 and the second over session 3 (Figure 10).

However, in the divide-and-forward mode for session 2, the next PDU to be sent to the network is:
N = R + K + 1, where R is the largest *Seq* of PDUs sent at that time; K is the number of PDUs that is sent by session 3 when session 2 sends one PDU. The value of K reflects the transmission speed of networks A and B. Similarly, the PDU that session 3 will send is N = R+K'+1, where K' is the number of PDUs that is sent by session 3 when session 2 sends one PDU. Thus, the module sends data with larger *Seq* in the sessions that have low transmission speed. By doing so, the receiver can receive the data in the right order, saving time in buffering and in sorting disordered data. Figure 11 shows an example where network A is four times faster than network B. Transmitting one PDU on network B (with session 3) takes the same amount of time as transmitting four PDUs on network A. Here, the Send Buffer Module sends PDUs 3, 4, 5, and 6 in session 2 and PDU 7 in session 3. The PDUs will then arrive at the receiver in the right order.

5. Multipath Receive Module

The Multipath Receive Module receives PDUs from networks A and network B. It stores received data into the Receive Buffer. If the Receive Buffer is full, the module stops reading data from the socket of sessions 2 and 3; it will wait until the buffer becomes available for more data.

The Multipath Receive Module monitors TCP sessions 2 and 3 using the command select() of Winsock. Whenever a PDU arrives, the module first checks if the PDU is a duplicate of any PDU that has previously arrived. If the PDU is a duplicate, then the PDU will be disposed of. Otherwise, the PDU will be stored in the Receive Buffer (Figure 12). Note that the processing flow in the Multipath Receive Module does not need to change when the Multipath Send Module changes from duplicate-and-forward to divide-and-forward.

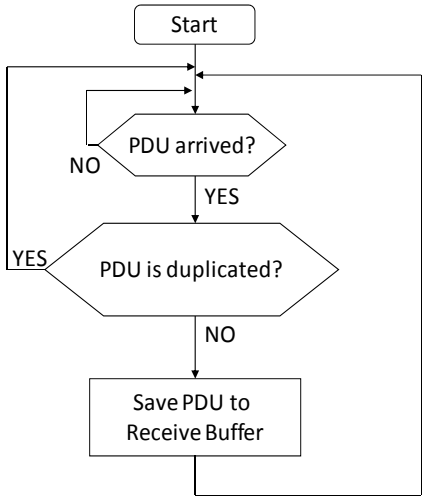

Figure 12: Algorithm for Multipath Receiver Module

VI. EVALUATION WITH NETWORK SIMULATOR

This section discusses our evaluation of the proposed algorithm in a laboratory network environment. We implemented MCCPs and installed programs on a thin client as well as a remote desktop server, as shown in Fig. 13. The simulator was a FreeBSD [3] machine with a Dummynet program [4] installed. There were two networks (A and B) in the simulator. The thin client communicated with the server through the two networks. The delay and bandwidth of the networks varied in each experiment.

A. Evaluation of tranmission delay

We first checked the round trip time (RTT) of a probe packet traversing between the thin client and the server when the DAMT algorithm was applied. We also investigated a case where the thin client only used network A or B to compare them. The propagation delay of networks A and B varied in a range from 100 ms to 300 ms. The bandwidth of both networks was set to 100 Mbps; this value was sufficiently large that the bandwidth did not cause additional delay in transmitting the probe packets. The settings for networks A and B are listed in Table I. The propagation delays of the two networks were set to the values shown in Figure 14.

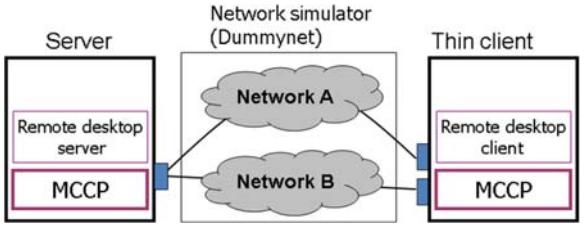

Figure 13: Network topology

TABLE I. SETTINGS FOR NETWORK METRICS

	Propagation delay (one way)	Bandwidth
Network A	Varies within range from 50 to 150 ms	100 Mbps
Network B	Varies within range from 50 to 150ms	100 Mbps

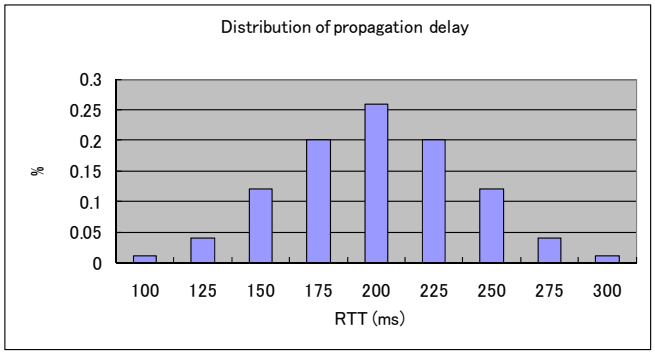

Figure 14: Distribution of propagation delay (round-trip)

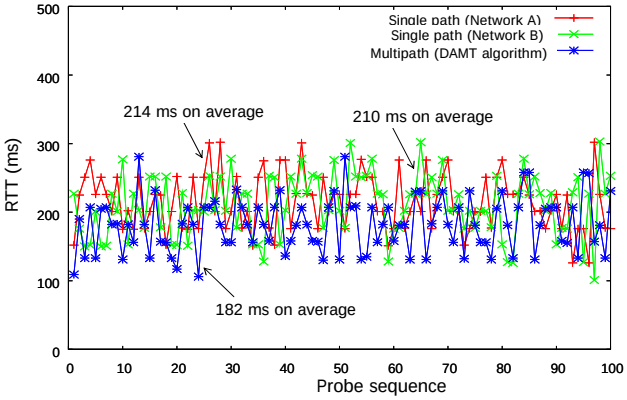

Figure 15: RTT-measurement results

The probe packet was 30 bytes and was sent from the thin client to the server and then immediately sent back from the server to the client.

Figure 15 shows the RTT values of the probe packet when it traversed network A, network B, and when it was forwarded by MCCPs with the DAMT algorithms. The RTT of the probe packets on average was 182 ms when using the multipath control algorithm, 214 ms when using network A and 210 when using network B. This means that DAMT could successfully reduce delay in the transmission of small packets.

B. Evaluation of data-transmission throughput

We evaluated the throughput of transmitted data in an experiment. The propagation delay for networks A and B was kept constant at 2 ms. The bandwidths of both networks were set to various values such as 500 Kbps, 1 Mbps, 1.5 Mbps and 2 Mbps (See Table II). For each value of bandwidth, we compared the throughput for bulk data transmitted between the client and the server when a single path (network A or B) was used and when multipath algorithms were used.

TABLE II. SETTINGS FOR NETWORK METRICS

	Propagation delay (one way)	Bandwidth
Network A	2 ms	500 Kbps, 1 Mbps, 1.5 Mbps, 2 Mbps
Network B	2 ms	500 Kbps, 1 Mbps, 1.5 Mbps, 2 Mbps

The results obtained from measuring throughputs are in the bar chart in Figure 16. The DAMT algorithm always outperformed the single paths (A and B) by up to 1.9 times. We can also see the throughput for multipath transmission when existing algorithms were used. We can see that DAMT achieved the same performance as the divide-and-forward algorithm, which is proposed for an aggregate bandwidth for multiple network connections.

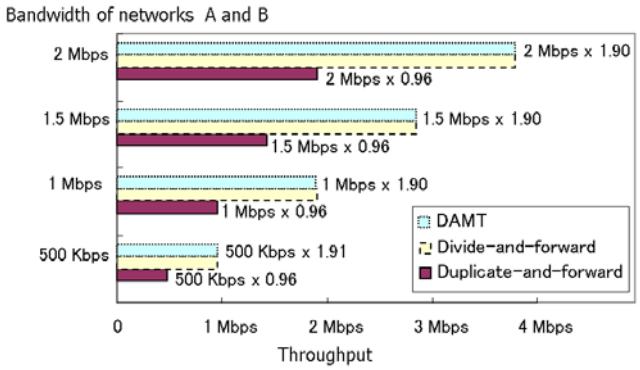

Figure 16: Throughput-measurement results

C. *Performance of thin client when using PowerPoint*

We next evaluated the performance of the thin client when Microsoft's PowerPoint program, which is a popular business application, was used. The settings for networks A and B are listed in Table III. The settings were determined according to the measured characteristics of two popular commercial mobile networks in Japan in May 2008. The remote desktop protocol used was RDP [12].

Figure 17 shows the PowerPoint file used in the experiment. The playing time of the file is about 4(s), the amount of data that was transmitted in the network when the file was being played was a total of 1626 KB.

Figure 18 is a bar chart of the average delay in response to keyboard input. This was calculated as the time from when the packet (including the input signal) left the thin client to when the first response packet returned back to it. We can see that the DAMT algorithm responded as quickly as the duplicate-and-forward algorithm.

TABLE III. SETTINGS FOR NETWORK METRICS

	Propagation time (one way)	Bandwidth
Network A	52 ms	380 Kbps
Network B	150 ms	600 Kbps

Figure 17: PowerPoint slides

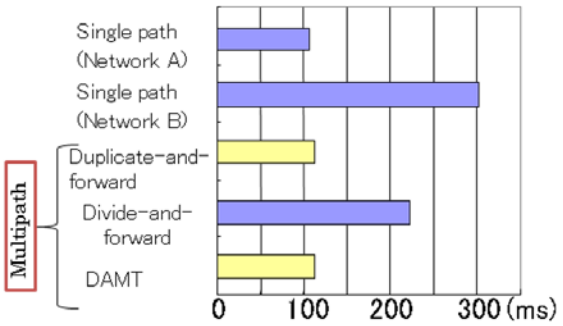

Figure 18: Response time to keyboard input

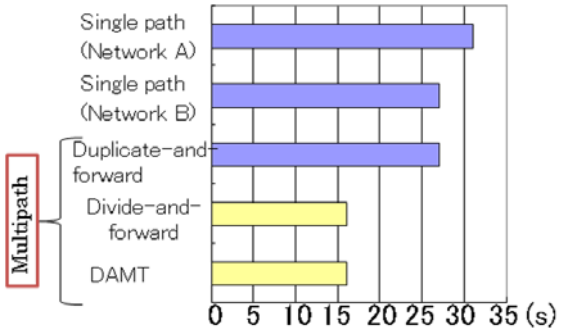

Figure 19: Time to play presentation file

Figure 19 is a bar chart that shows the time it takes to play a presentation file. The speed at which the file is played depends on the throughput of the desktop images transmitted from the server to the client. We can see that the DAMT algorithm takes the same time as the divide-and-forward algorithm. And the time is far shorter than the playing time when only network A or B is deployed.

The DAMT algorithm thus shortens the response to keyboard input to that of the duplicate-and-forward algorithm, and the speed at which the presentation file is played is as high as that of the divide-and-forward algorithm.

VII. EVALUATION IN REAL NETWORK ENVIRONMENT

We next evaluated the performance of the proposed method DAMT in a real network environment. The network used in the experiment is outlined in Figure 20. A thin client equipped with two high-speed wireless network access cards was used. The two cards had different data-transmission methods and different access networks. The first used WiMAX and the other used HSPA. The thin client was located in an office on the top floor of a six-storey office building. The thin client accessed an RDP server located in a data center through a gateway located in the same local network as the server. A server-side MCCP was installed in this gateway. The data center was connected to a WAN by a high-speed network (20 Mbps).

We first evaluated the time it took for a TCP packet to traverse from the thin client to the server and back again. This time indicated how fast a letter appeared on the screen of the thin client when the user pressed a key.

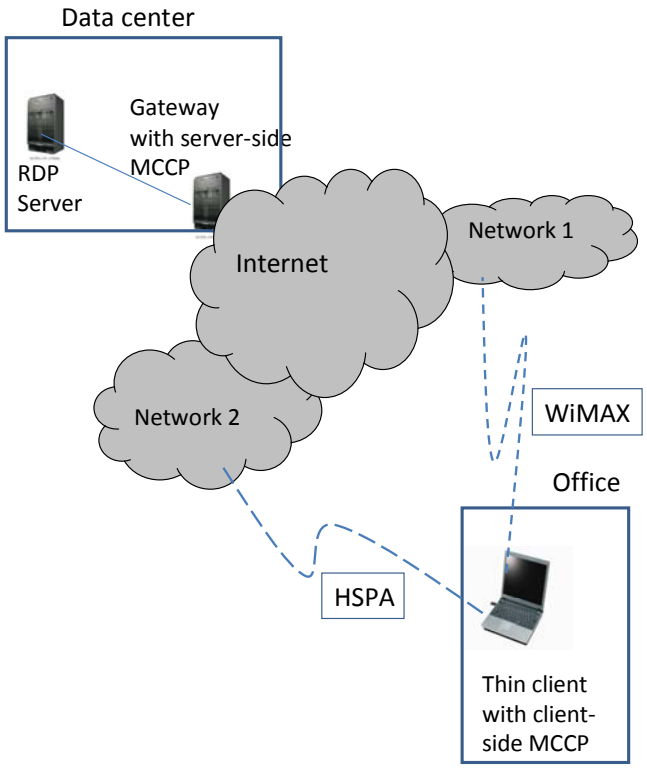

Figure 20: Experimental network

We also evaluated the amount of RDP traffic transmitted from the RDP server to the thin client when the user watched a 15-s video. This amount of data reflected the quality of the video shown on the screen of the thin client. If the amount of data was small then the video had many too lost frames. We checked the amount of data when WiMAX access and HSPA access were used singly and when DAMT was used.

Figure 21 has bar charts of the average value (in 50 experiments) of the round trip time of a TCP data packet when it used network 1 (through WiMAX access), and network 2 (through HSPA access). The figure also indicates the round trip time when the multipath algorithm was deployed. We can see that when the thin client used HSPA access, the response time was about 159 ms, and the WiMAX access was 201 ms. However, the response time when the proposed algorithm DAMT was deployed was only 139 ms. The results attained when using WiMAX access and HSPA access may change if we carry out the experiments in other places. However, as the DAMT algorithm will choose the shortest access for all data packets, we can rest assured that DAMT will introduce a shortened response time in any environment.

We next checked the amount of data that the thin client received during the time it took to play a 15-second video file. In a LAN environment, the thin client played the video with almost no difference to playing it directly. Here, the thin client received about 20 MB of data for the video file. Table 4 lists the amount of data when WiMAX, HSPA, and the proposed multipath data transmission algorithm were used.

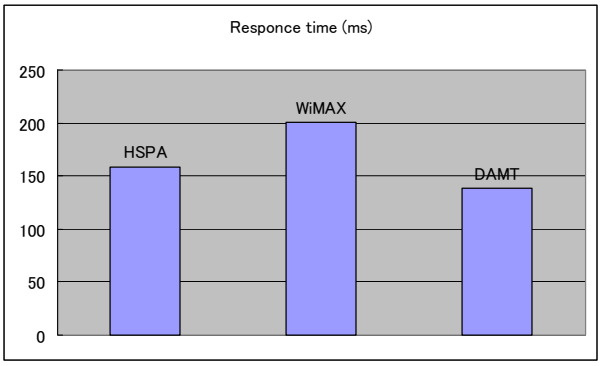

Figure 21: Average value of response time

As seen in Table IV, when using HSPA access, the thin client could receive only 21.5% of the data in comparison with the LAN environment. In WiMAX access, the ratio was 23.8%. However, the DAMT algorithm performed better when the ratio was 43.8%. The reason for the higher throughput of DAMT is that, the thin client can use the bandwidth of both wireless networks to transmit the desktop information (Figure 22). From Figure 22, we can see that DAMT deployed both network interfaces to receive data from the HSPA and WiMAX networks.

We could test and confirm the superior performance of the proposed algorithm through a real-network environment. Similar to the results we obtained in the laboratory, the DAMT algorithm could introduce short response times for small packets such as the RDP packets that occur when users are typing. As they also increased the bandwidth of the network path between the thin client and RDP server, video files could play more smoothly than when using a single wireless network.

TABLE IV. AMOUNT OF DATA RECEIVED BY THIN CLIENT

	Received data (bytes)	Compared with LAN
Network 1 (WiMAX)	4,554,858	21.5%
Network 2 (HSPA)	5,040,438	23.8%
Multipath (DAMT)	9,046,273	42.8%
LAN	21,134,323	100%

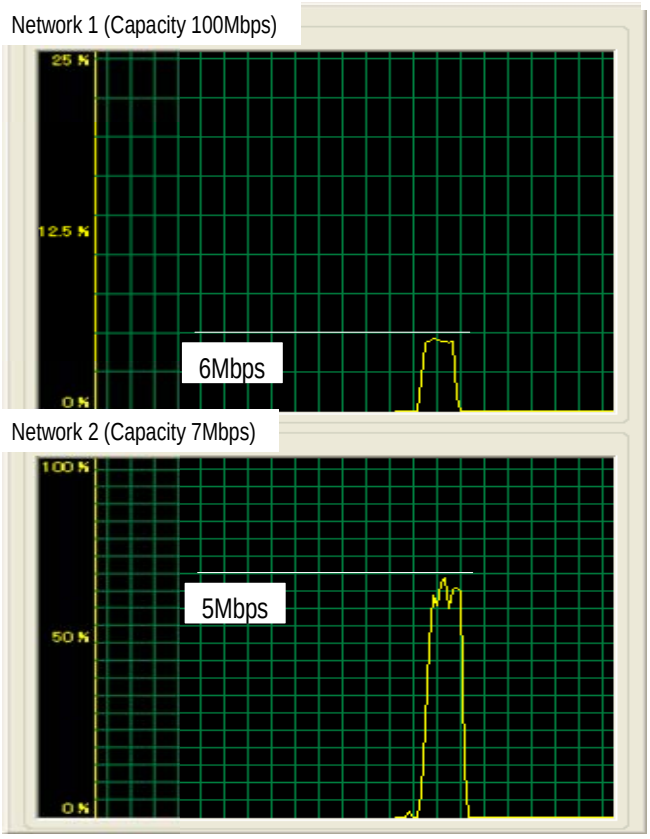

Figure 22: Data transmission on network interfaces of thin client

Also note that the thin client in the experiment was not located in the best place to achieve the best performance with WiMAX and HSPA access. The aims of the experiment were not to evaluate the performance of these wireless access networks, but just to confirm that DAMT could work well in a real-network environment. We also found that DAMT could outperform single network access by using the multipath approach.

VIII. Conclusion and Future Work

We proposed an algorithm called DAMT that leveraged multiple wireless network connections to attain short latency in the transmission of input signals and its responses as well as to increase the throughput of large desktop images.

Due to the large range of experiments in both the simulator and in a real network, we found that the thin client could quickly respond to keyboard input with the DAMT algorithm and could quickly play animated presentations.

However, some future work still needs to be done in this respect. First, we intend to carry out more experiments in real networks to find what additional benefits DAMT can provide. We intend to assess the performance of thin clients in places other than offices, such as in trains or cafeterias. Implementing MCCPs that can work over three or more wireless networks is also within the scope of future work.

Indication of Trademark Registration

XenDesktop(TM) is trademarks of Citrix Systems, Inc.
Remote Desktop(TM) is a trademark of Microsoft Corporation.
X Window System(TM) is a trademark of The Open Group.
VNC is a trademark of AT&T Laboratories Cambridge.
FreeBSD is a trademark of the FreeBSD Foundation.
PowerPoint is a registered trademark of Microsoft Corp. in the U.S. and other countries.
Windows is a registered trademark of Microsoft Corp. in the U.S. and other countries.

References

[1] Citrix Systems, http://www.citrix.com 05.05.2009

[2] Remote Desktop Protocol, http://msdn.microsoft.com/en-us/library/aa383015.aspx 05.05.2009

[3] FreeBSD, http://www.freebsd.org 05.05.2009

[4] Luigi Rizzo, Dummynet, http://info.iet.unipi.it/~luigi/ip_dummynet/ 05.05.2009

Article in a journal:

[5] R. W. Scheifler and J. Gettys, “The X Window Systems”, ACM Transactions on Graphics, 5(2), 1986.

Article in a conference proceedings:

[6] ---------,“Multipath Data Tranmission for Wireless Thin Clients”, Proc. UBICOMM 2009.

[7] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, “Virtual Network Computing”, Proc. IEEE Internet Computing, 1999.

[8] S. Ramamurthy and A. Helal, “Optimising Thin Client for Wireless Computing via Localization of Keyboard Activity”, Proc. IEEE Conference on Performance, Computing, and Communications, pp. 249-252, 2001.

[9] C. Aksoy and S. Helal, “Optimising Thin Clients for Wireless Active-media Applications”, Proc. Third IEEE Workshop on Mobile Computing Systems and Applications, pp. 151-160, 2000.

[10] M. Lubonski, V. Gay, and A. Simmonds, “An Adaptation Architecture to Improve User-Perceived QoS of Multimedia Service for Enterprise Remote Desktop Protocols”, Proc. Next Gerenation Internet Networks (NGI 2005), 2005.

[11] M. Lubonski, V. Gay, and A. Simmonds, “A Conceptual Architecture for Adaptation in Remote Desktop Systems Driven by the User Perception of Multimedia”, Proc. Asia-Pacific Conference on Communications, pp. 891-895, 2005.

[12] J. C. De Martin and D. Quaglia, “Distortion-based Packet Marking for MPEG Video Transmission over DiffServ Networks”, Proc. IEEE International Conference on Multimedia and Expo, 2001.

[13] J. Chen, K. Xu and M. Gerla, “Multipath TCP in Lossy Wireless Environment,”Proc. IFIP Third Annual Mediterranean Ad Hoc Networking Workshop, 2004.

[14] H. Hsieh and R. Sivakumar, “A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts”, Proc. ACM MobiCom, 2002.

[15] K. Park, Y. Choi, D. Kim, and D. Park, “MTCP: A Transmission Control Protocol for Multi-Provider Environment”,Proc. IEEE CCNC, 2006.