Design and Implementation of A Network Application Architecture for Thin Clients

Chia-Chen Kuo, Ping Ting, Ming-Syan Chen*, and Jeng-Chun Chen+

Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, ROC

Philips Research East Asia+
Philips Corporation
Taipei, Taiwan, ROC

Abstract

This paper explores the issues and the techniques of enabling multimedia applications for the thin client computing. Our study on the network applications over thin clients is devoted to the universal plug-in architecture which supports multimedia applications over thin clients. With the proposed architecture, the universal plug-in technique is devised to shift jobs from thin clients to a supporting server, thus reducing the computing overhead required by the thin clients. In addition, this plug-in architecture is implemented and empirically evaluated. It is shown by our experiments that the proposed architecture not only significantly enhances the multimedia capability of thin clients but also reduces the memory consumed by the clients for various applications, showing the very advantage of the universal plug-in architecture for thin clients of limited resources.

1. Introduction

Recent technology advances have brought revolutionary impacts to our life. In particular, real-time transmission of multimedia data over the Internet is attracting an increasing amount of attention. Nowadays, more and more devices, varying from a game station to a cellular phone, are equipped with Internet-access functions. Since most of these devices are not as powerful as desktop PCs, they are often called "thin client" [5][6]. In the thin-client/server computing model discussed in this paper, an application is split into two parts: a front-end client that presents information to the user, and a back-end server that stores, retrieves, and manipulates the data, and generally handles the bulk of the computing tasks for the clients [10]. With this model, applications are deployed, managed, supported and executed completely in the server. It requires a multi-user operating system and a remote presentation service protocol for distributing the presentation of an application interface to a client device [2][3].

In this paper, we explore the approach of adding multimedia extensions to such a computing system to allow the system of real-time multimedia processing with the constrained network bandwidth and computing power of the server and the thin client devices. Under the proposed architecture, the universal plug-in devised shifts jobs from thin clients to a supporting server, thus reducing the computing overhead required by the clients.

We have implemented a prototype of universal plug-in architecture that contains the required plug-in modules both on the thin client and plug-in server. Without loss of generality, we take Netscape Navigator as the Web browser in the thin clients [7]. We also adopt widely deployed protocols when prototyping the proposed architecture [8]. With these considerations in mind, the architecture can be extended to a variety of thin clients smoothly. Based on the prototype implemented, several experiments have been conducted. Extensive performance studies demonstrate that such a universal plug-in approach is feasible and effective. More importantly, we adopt the run-length encoding [9] in the implemented prototype system, reaching the compression rate about 20:1. This feature reduces the amount of data transmission between the thin client and the supporting server significantly, thus making the proposed architecture feasible on the thin client which is sustained with limited Internet-access bandwidth. We show that some of the most popular Web document formats, such as Microsoft Word and Acrobat PDF, can be effectively transcoded and displayed on a thin client with limited performance impact to the server. It is shown by our experiments that the proposed architecture not only significantly enhances the multimedia capability of thin clients but also reduces the memory consumed by the clients for various applications, showing the very advantage of the universal plug-in architecture for thin

Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC'02)
0730-3157/02 $17.00 © 2002 IEEE

clients of limited resources. The limited performance impact to the server and the efficiency in the generic format processing make the universal plug-in architecture devised in this paper a very important addition to the thin-client/server computing model.

This paper is organized as follows. A system architecture that resolves the issues and constraints for the multimedia extensions to thin clients is described in Section 2. With the proposed architecture, the design and implementation of universal plug-in is presented in Section 3. The performance evaluation of the universal plug-in architecture will be addressed in Section 4. This paper concludes with Section 5.

2. Preliminary

To enable the multimedia support under thin-client/server computing model, we need to facilitate the multimedia streaming between the server and clients. The streaming capability allows the transmission of audio/video of the applications between the server and remote thin clients. In this section, we will explore the solutions with such a computing model based on the constraints of current RDP (Remote Desktop Protocol) [4] developed in the Windows-based Terminal Server [2]. We adapt the H.323 [1] videoconference application with the thin-client computing model in order to demonstrate the capability of supporting multimedia applications with a terminal server and multiple thin clients.

2.1. Remote Concept Device

The current commercial solution for remote presentation protocols only supports the capability of remote display. In order to add Audio/Video streaming capability, one solution is to include the multimedia streaming handling protocol into the remote presentation service protocol directly. To add the extra multimedia streaming handling protocol into RDP, modifying the Windows-based Terminal Server and developing new client applications will be necessary.

There are some factors to be considered when the multimedia extensions are being implemented, including (1) Network bandwidth that may vary from telephone dial-up link to an Ethernet LAN connection; (2) Server computing power which is required to provide multimedia extensions to multiple thin client devices; (3) Client computing power and capability that is to handle minimum processing for audio and video codecs; (4) Display overlap of the video screen since the video output needs to be merged with the screen updates of client applications.

To cope with the constraints mentioned above, we come up with a mechanism for the thin-client/server computing model, i.e., the remote device concept. A thin client can be viewed as a remote "display + UI" device of the terminal server. The applications that are executed on the server can also use the remote devices as their local devices, including the display, the keyboard, and the mouse. The addition of the multimedia streaming handling protocol can coexist with the RDP to enable the extended functionality for real-time multimedia applications. For example, to support the videoconference application, it requires remote video capture devices, remote audio capture devices, remote video output devices, and remote audio output devices.

Considering the audio device for example, we implement the remote audio device control agent as a full-duplex audio streaming driver in the terminal server. This audio control agent is responsible for receiving the input audio stream from the service agent and transmitting the output audio stream to the service agent. This audio service agent is responsible for transmitting the recorded voice to the audio control agent and outputting the received voice stream from the control agent into the sound output device. The audio streaming process in audio device agents is composed of three steps: packetization/de-packetization, buffering, and compression/de-compression.

The remote audio device control agent is implemented as a transparent two-way audio streaming driver. Hence, we can easily install and use it as a general sound card driver. The remote audio device service/control agent provides most Windows Waveform functions, e.g., device open, play, stop, pause, record, and device close, and so on. To meet the need of an H.323 videoconference application, we support the following speech compression schemes, i.e., G.711 and G.723. The choice of the codecs depends on the network bandwidth, the capability of the client device, and the requirement of applications.

2.2. Universal Plug-in

Our approach taken in the proposed architecture is to shift the heavy jobs from the thin client to a supporting server which is installed with all the auxiliary software modules needed. For instance, a client sends HTTP request to the Web server and the Web server sends back the content of the requested Web page. If there are some contents unrecognizable by the browser, the universal plug-in module is invoked and passed the URL of unrecognizable files. The URL is then forwarded to a plug-in server by the plug-in module on the client side. The actual data content of the unrecognizable file is *either* passed through the client and then forwarded to the

plug-in server *or* directly sent to a plug-in server from the Web server. In the latter case, the plug-in server actively creates an HTTP request to the Web server with the URL received from the plug-in module on the thin client. Once receiving the content of a file, the plug-in server invokes WebBrowser Control to process the file. The WebBrowser Control in turn utilizes a variety of Active Document Components to cope with the particular file type. All processed data is transcoded into a universal format which is recognizable by the client and then sent back to the plug-in module on the thin client. The data flow is illustrated in Figure 1.

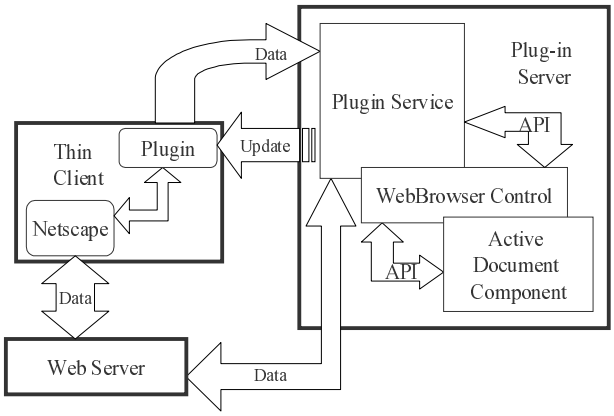

Figure 1: Data flow in the universal plug-in

The data at the plug-in server side can be categorized into two types: static data and video stream. The static data refers to the data with document formats such as Acrobat PDF, Microsoft Word, and PostScript, as so on. All these documents are rendered on the screen with a static frame content which may change only in response to the user's input. On the other hand, the video stream refers to the video data type like MPEG-1, H.263, or QuickTime. These video files are played with continuous frames rendered swiftly on the screen at a certain frame rate which is usually no less then 15 frames per second. The transcoding technology is required in processing video streams.

Figure 2 illustrates the flow chart when we process the static data which is Acrobat PDF in this case. Suppose there is no software module able to decode PDF installed on this thin client. When the Web browser on thin client intends to view a PDF file on the Web, the universal plug-in sends a request to a plug-in server to ask for help. The plug-in server downloads and displays this PDF file on its own screen in response to the thin client's request. Then the plug-in server captures the screen data and sends the data to the thin client. The universal plug-in module on the thin client is able to decode the screen data from the plug-in server, and the screen data is thus decoded and rendered on the thin client's screen. This plug-in architecture will be implemented and empirically studied later.

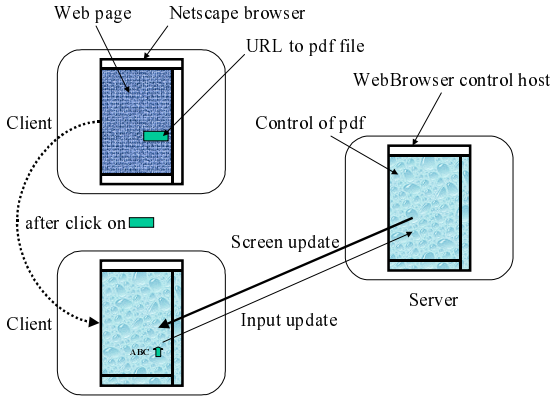

Figure 2: Scenario of static data screen update

3. Design of Universal Plug-in

This section describes the design and the implementation of a universal plug-in architecture which is devised to provide an alternative solution as opposed to the one of having many plug-ins installed on a thin client.

3.1. Thin-client/Server Computing Model

The universal plug-in is an innovative architecture devised in this paper to reduce the level of expertise required for a software engineer to develop a new auxiliary software module on all platforms. This plug-in work complements the thin-client/server computing model in that some clients are usually equipped with too few resources to accept many multimedia applications. In addition, we also aim at providing a better user experience by minimizing the work of upgrading users' thin client devices.

Figure 3 shows the network connections between the Web server, thin client devices and the plug-in server. The Web server is an HTTP server hosting HTML Web pages and related files. The thin client is connected to the Internet and equipped with a Web browser. The universal plug-in software module is installed in the thin client system to cooperate with the Web browser. The plug-in module is configured to send requests to the plug-in server. The approach taken in this architecture is to shift the heavy jobs from the thin client to a supporting server installed with all the auxiliary software modules needed. The processed result at the server end is transcoded to a

pre-defined universal format and then sent back to the thin client.

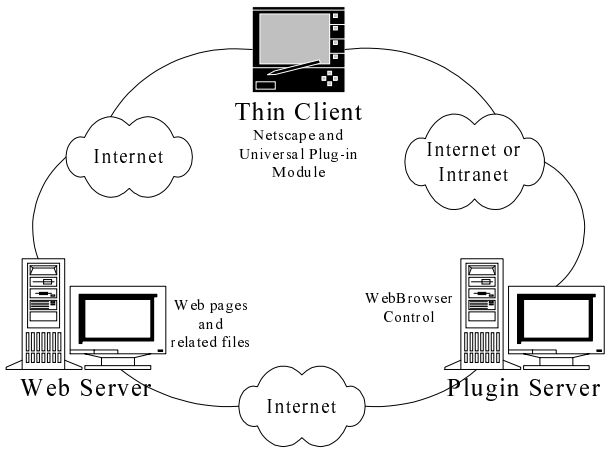

Figure 3: Thin-client/server computing model

3.2. System Module of Universal Plug-in

The system module of universal plug-in is composed of a plug-in server and compression/decompression codecs. To develop the plug-in server, we employ multithread programming techniques in our implementation. Also, we create two additional threads to control the socket to the client for handling input and output. Figure 4 illustrates the concept of our plug-in module. We employ the WebBrowser control of Windows in the prototype system. All jobs related to rendering Web pages are pushed into the WebBrowser Control. When these jobs are completed, the WebBrowser Control notifies their owners by firing Windows events conveyed through the COM interface. Therefore, we implement an event sink which is capable of receiving Windows events in plug-in server program.

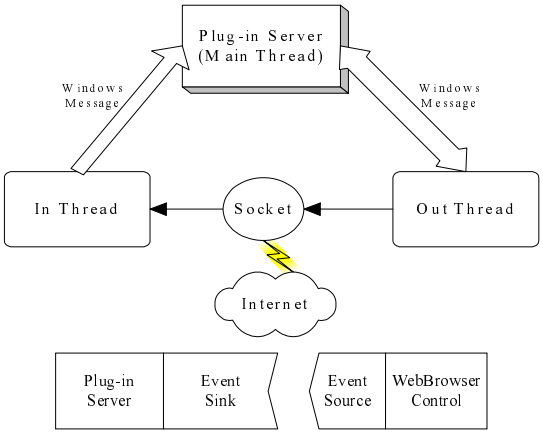

Figure 4: System module of universal plug-in

The data bandwidth from server to client is mainly consumed by the captured screen shot. The amount of pixel data for a screen with high resolution is huge. Thus, the screen update mechanism would not be feasible without a high degree of compression. Since the pixel block of a screen often consists of a variety of sub-blocks with the same color, encoding these blocks with a lossless encoding algorithm is able to dramatically reduce the data amount. The run-length encoding algorithm [9] is implemented in our system to compress the screen update data from the server to a client. This feature makes the proposed architecture be feasible on the thin client which is often sustained with limited Internet-access bandwidth.

4. Performance Study

The Web browser, i.e., Netscape Navigator, running on thin client is originally capable of processing file types of Acrobat PDF and Microsoft Word. For experimental purposes, these features are intentionally disabled. The universal plug-in module is installed on a thin client and is programmed to process the PDF and Word files. When the Web browser is employed to view PDF or Word files, the universal plug-in module is loaded into Navigator's process memory space. The plug-in module is configured to send requests to the plug-in server. It is noted that we choose Word and PDF files for demonstration in order to compare with traditional local plug-ins which are commonly in cooperation with Web browsers. The universal plug-in architecture can be extensively applied to multimedia applications.

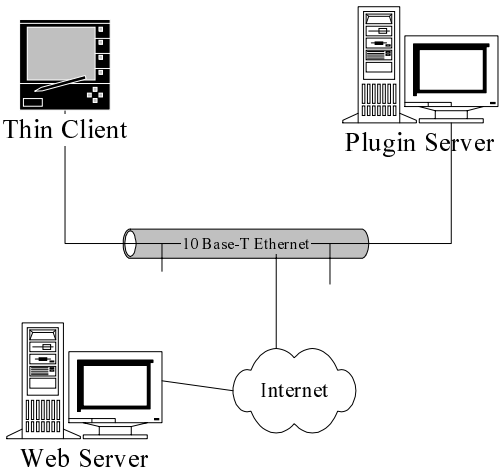

Figure 5: Network connections in experiments

Figure 5 shows the network connections between a thin client, a Web server, and a plug-in server. In this scenario, the thin client is able to access the Web server directly

via the Internet. The Web page data go through the plug-in server only when an unrecognized file type is encountered; otherwise, all data are transferred directly from the Web server to the thin client.

In our experiments, the thin client and the plug-in server are connected by a 10-based LAN. Both the thin client and the plug-in server have their direct links to the Web server through the Internet. In addition, the compression and decompression codecs are employed in the system. All performance indices are measured in two cases in which the compression modules are disabled or enabled. As validated by our experimental results, the universal plug-in architecture significantly reduces the memory consumed by the client with the trade-off of slightly increased response time. This fully demonstrates the very advantage of the universal plug-in module for thin clients of limited resources. The thin client and the plug-in server are connected by a 10 base-T Ethernet which is in turn connected to the Web server. The plug-in server is equipped with Pentium III 700 MHz CPU and 512 MB RAM. The client is equipped with Pentium 90 MHz CPU and 8 MB RAM. The resource in the thin client is intentionally limited.

4.1. Experiments to Assess Plug-in Server Load

We measure the peak value of memory consumed by the plug-in server process when there is only one client connected. The peak values of the amount of memory consumed, when the server program is (1) in the idle state, (2) in the state of processing a Microsoft Word file, and (3) in the state of processing an Acrobat PDF file, are depicted in Figure 6. The amounts of memory needed by processing Word and PDF files are almost the same, and exceed the one needed in the idle state only by 2MB, showing the limited impact to the server since the server is in general equipped with adequate resources and is not as price sensitive as a client which is usually produced in a large quantity. The requirement of the hardware equipment of a plug-in server can be profiled by the observed value of the memory usage incurred by the client connections. Also, it can be observed that the compression module adds little memory overhead to the plug-in server system.

4.2. Experiments to Assess Thin Client Load

The memory consumed by Netscape Navigator on the thin client is also measured. The results are shown in Figure 7. It should be noted that there is a remarkable drop in the memory consumed by the browser when the universal plug-in mechanism is used in the processing of Word and PDF files. This fully demonstrates the very advantage of the universal plug-in module in the thin client which is equipped with very limited resource and is, as mentioned above, very price sensitive. Even in the case with compression module enabled, only a little overhead of memory adds to the client system.

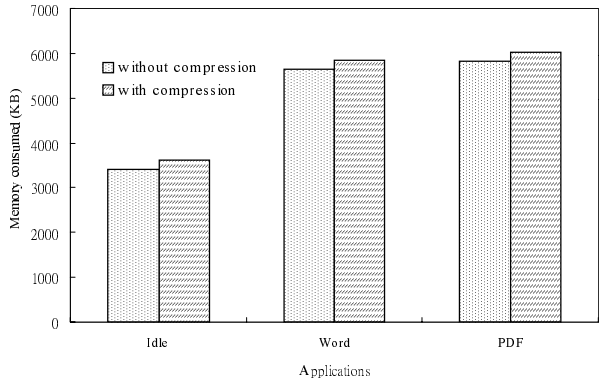

Figure 6: Memory consumed by server program

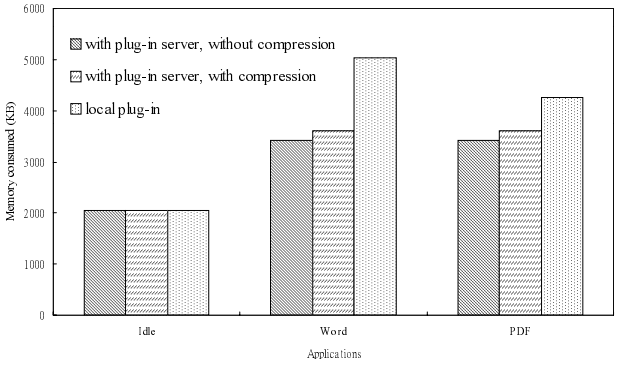

Figure 7: Memory consumed by the thin client

4.3. Experiments to Assess Response Time

We have implemented interactive functionality in this universal plug-in system. The time required for various operations is measured and shown in Figure 8. The measurement is made by the use of specific Windows function calls. It should be noted that the time of initial loading page in the case of using the plug-in server is shorter than that of the local plug-in. For other operations, the round trip delay between the plug-in server and the thin client is acceptable. With the compression module enabled, the response time can be further reduced.

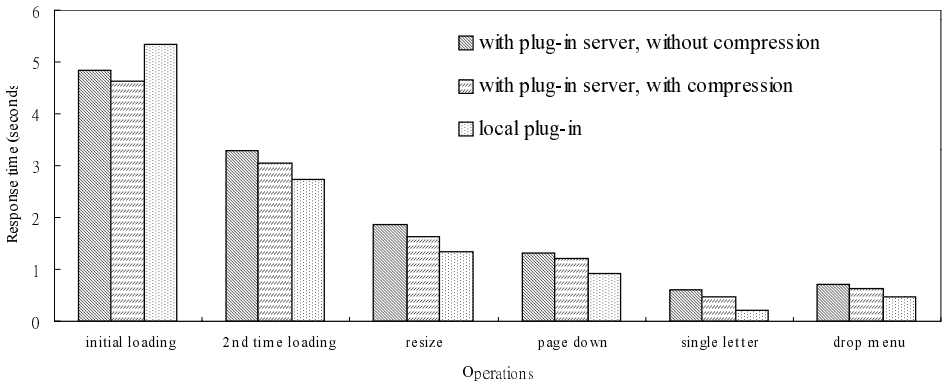

Figure 8: Response time of various operations

5. Conclusion

The universal plug-in technique was devised in this paper to shift jobs from thin clients to a supporting server, thus reducing the computing overhead required by the clients. This approach not only significantly enhances the multimedia capability of thin clients but also releases end users from the suffering of upgrading software and hardware. It is shown by our experiments that the universal plug-in architecture proposed leads to a reduction on the memory consumed by the clients for various applications, showing the very advantage of the universal plug-in architecture for thin clients of limited resources. The system architecture proposed renders the advantages of reduced total cost as well as enhanced ease-of-use with little degradation on the overall performance.

6. References

[1] ITU-T Recommendation,"H.323: Packet-Based Multimedia Communications Systems". 1997.

[2] White Paper, "Microsoft Windows NT Hydra: The Benefits of Windows with the Low Cost of a Terminal". Microsoft Corporation, 1998.

[3] White Paper, "Thin-Client/Server Computing". Citrix System, Inc, 1998.

[4] White Paper, "Remote Desktop Protocol (RDP) Features and Performance". Microsoft Corporation, 2000.

[5] M. Jern. "Thin vs. Fat Visualization Client". Proceedings of Computer Graphics International, 1998.

[6] J. Nieh, S. J. Yang, and N. Novik. "A Comparison of Thin-Client Computing Architectures". Technical Report CUCS-022-00, Department of Computer Science, Columbia University, 2000.

[7] M. A. Cusumano and D. B. Yoffie. "What Netscape Learned from Cross-Platform Software Development". Communications of the ACM, 42(10):72-78, 1999.

[8] S. W. Tak, J. M. Son, and T. K. Kim. "Experience with TCP/IP Networking Protocol S/W Over Embedded OS for Network Appliance". Proceedings of International Workshops on Parallel Processing, pages 556-561, 1999.

[9] S. W. Golomb. "Run-length Encoding". IEEE Transactions on Information Theory, 12(4):399-401, 1966.

[10] T. Richardson, Q. StaRord-Fraser, K. R. Wood, and A. Hopper. "Virtual Network Computing". IEEE Internet Computing, 2(1):33-38, 1998.