DISTRIBUTED CLIENT/SERVER ARCHITECURE WITH DYNAMIC MIDDLE TIER

CHOONG KHONG NENG

DISTRIBUTED CLIENT/SERVER ARCHITECURE WITH DYNAMIC MIDDLE TIER

By

CHOONG KHONG NENG

Thesis Submitted in Fulfilment of the Requirements for the Degree of Master of Science in Faculty of Engineering Universiti Putra Malaysia

December 1999

ACKNOWLEDGEMENTS

I would like to express my earnest gratitude to my supervisor, Prof. Malay Raj Mukerjee, for his guidance and support throughout my research. He provided me a lot of freedom to explore and investigate the research topics and ideas that I found interested. He was always available for providing useful direction and advice when I needed. I'm really impressed with the mastery of different aspects with which he maintained.

Thanks to Assoc. Prof. Dr. Borhanuddin Mohd. Ali for believing in me enough to help me grow throughout my research. I count myself lucky to have admitted to the department and be one of his students. Thanks for his dedications and efforts in making everything possible.

Another important person who I enjoyed working with the most is my terrific mentor, Assoc. Prof. Dr. Dzaharudin Mansor. Not only have he offered invaluable technical guidance, advice and inspiration, but he has also helped me develop as a professional engineer. I would like to thank him for the invaluable experiences and in-depth knowledge that he imparted to me. I really benefited a lot from him. Thanks for convincing me and making everything possible towards the implementation this project.

Thanks also to Chee Bon Kok, my helpful senior, for his encouragement and support. Thanks also go to Low Wai Yan and Chen Shong Der for their assistance throughout my research. I feel happy to have friends like them.

Thanks to my parents, brother and sister for their love and support in the dark days. Most of all, I thank my beloved girl friend, Yee Yoke Chek for her constant support through the years of challenges in my life, as well as for her patience, trust and understanding.

Special thanks also go to all the colleagues or fellow friends such as Ali Mohd. Abdelrahman, Tan Kee Leong and Yem Poh Cheang for their helps and co-operations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	-	Application Programmer Interface
CMC	-	CHEK Management Console
CORBA	-	Common Object Request Broker Architecture
CPF	-	CHEK Proxy Framework
DCOM	-	Distributed Component Object Model
DNS	-	Domain Name Service
DOMS	-	Distributed Object Monitoring System
DPE	-	Distributed Processing Environment
FTP	-	File Transfer Protocol
HTML	-	Hyper Text Markup Language
HTTP	-	Hyper Text Transfer Protocol
IIOP	-	Internet Inter-ORB Protocol
JVM	-	Java Virtual Machine
MOM	-	Message-Oriented Middleware
OMG	-	Object Management Group
ORB	-	Object Request Broker
RMI	-	Remote Method Invocation
SPE	-	Software Performance Engineering
TP	-	Transaction Processing
WAN	-	Wide Area Network
WWW	-	World Wide Web

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirements for the degree of Master of Science.

DISTRIBUTED CLIENT/SERVER ARCHITECTURE WITH DYNAMIC MIDDLE TIER

By

CHOONG KHONG NENG

December 1999

Chairman : Associate Professor Borhanuddin Mohd. Ali, Ph.D.

Faculty : Engineering

Widespread use of computer networks and the demanding needs of current network applications and technology impose a challenge to use the bandwidths in an efficient manner so as to solve the network congestion and server overloading problems. Some past and on-going solutions such as server replications and caching have been proposed to overcome these deficiencies. However, these solutions have not been implemented in an economical and configuration-transparent manner. Moreover, the problems of caching and disseminating real-time multimedia data in a bandwidth-conservative manner have not been addressed.

In this thesis, a CHEK Proxy Framework (CPF) has been developed using a proxy solution to address these problems. By caching, proxy has become a traditional solution in reducing user-perceived latency and network resource requirements in the network. CPF helps to create a middle-tier application platform proxy transparently and dynamically in the client sub-network to execute the sharable section of any server application codes. This is as the application proxy.

Besides caching static web contents, this local application proxy helps to deliver real-time multimedia data on behalf of the remote server with lower bandwidth and better performance. CPF helps to minimize WAN connections while maximizing LAN interactions by multiplexing and de-multiplexing client requests through to the server via the proxy. As a result, the central server is made more reliable and scalable.

The monitoring and management of the CHEK distributed objects is also made easier through the use of the CHEK Management Console (CMC). CMC displays the inter-relationships between the distributed objects and their status information on a GUI-based control panel for ease of management.

With its dynamic and transparent features, software versioning and maintenance problems are readily overcome. CPF has been shown to be useful in most client/server applications, particularly those of broadcasting and collaborative nature such as video broadcastings and chat systems. CPF solves the network congestion and server overloading problems with the presence of a middle-tier proxy application platform which is allocated in the client sub-network with no manual configurations.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia untuk memenuhi keperluan ijazah Master Sains.

SENIBINA PELAYAN/PELANGGAN TERTABUR DENGAN PERINGKAT PERTENGAHAN YANG DINAMIK

Oleh

CHOONG KHONG NENG

Disember 1999

Pengerusi : Profesor Madya Borhanuddin Mohd. Ali, Ph.D.

Fakulti : Kejuruteraan

Penggunaan rangkaian komputer yang meluas serta keperluan kepada aplikasi dan teknologi rangkaian masa kini meletakkan satu cabaran untuk mengguna lebarjalur dengan berkesan bagi menyelesaikan masalah kesesakan rangkaian dan lebihbeban pelayan. Beberapa rumusan seperti *caching* dan penggandaan pelayan telah dicadangkan untuk mengatasi kekurangan ini. Walau bagaimanapun, penyelesaian ini tidak dilaksanakan dengan ekonomikal dan telus-konfigurasi. Lebih lebih lagi, masalah *caching* dan penyampaian data multimedia masa nyata dengan kaedah jimat-lebarjalur, tidak diatasi dengan jelas.

Dalam Tesis ini, satu kaedah *proxy*, diberi nama, CHEK Proxy Framework (CPF) telah dibangunkan untuk mengatasi masalah ini. *Proxy* merupakan satu penyelesaian bagi merendahkan penangguhan dari aspek pengguna dan keperluan sumber rangkaian melalui kaedah *caching*. CPF dapat membantu merekakan satu aplikasi platform *proxy* peringkat tengah secara dinamik and telus, di rangkaian pelanggan untuk menjalankan kod pelayan yang boleh dikongsi dan dijadikan sebagai kod proxy. Selain daripada *caching* kandungan web statik, aplikasi proxy tempatan ini dapat menghantar data multimedia masa nyata bagi pihak pelayan jauh dengan penggunaan lebarjalur yang rendah dan hasil yang lebih baik. CPF mengurangkan sambungan WAN dan meningkatkan interaksi LAN dengan memultipleks dan menyahmultipleks permintaan pelanggan menerusi *proxy* ke pelayan. Justeru itu, pelayan pusat akan dijadikan lebih bolehpercaya dan bolehtingkat.

Pengawasan dan pengurusan objek tertabur CHEK dimudahkan dengan penggunaan CHEK Management Console (CMC), satu panel kawalan berasaskan GUI yang menunjukkan perhubungan antara objek tertabur dan kedudukan informasi mereka.

Dengan sifat CPF yang dinamik and telus, masalah versi perisian dan penyelenggaraan dapat diatasi. Penggunaan CPF didapati berguna dan membawa manfaat kepada kebanyakan aplikasi pelanggan/pelayan, terutamanya aplikasi yang bersifatkan penyebaran am dan interaktif seperti sistem penyebaran video dan sistem perbualan. CPF didapati berkesan untuk mengatasi masalah kesesakan rangkaian dan lebihan beban pelayan dengan adanya aplikasi platform proxy peringkat tengah yang direka di rangkaian pelanggan tanpa konfigurasi manual.

CHAPTER I

INTRODUCTION

The Challenges to Efficient Use of Network Resources

All networks have a limited data-carrying capacity. When the load is light, the average time for transmitting a packet from the client to server is short. When many users are vying for connections and communicating, the average delay increases. This delay causes the transmission of data to be slower as longer time is needed to send the same amount of data under the congested conditions. In extreme circumstances, an application can fail completely under a heavy network load. Sessions may encounter time-outs and disconnection, and applications or operating systems may actually crash, requiring a system restart.

The network is identified as one of several possible bottlenecks and the common causes of congestion in today's networks. Some of the reasons are:

- High demand from bandwidth-intensive applications, such as video and audio broadcasting which involves transfer of large data in a continuous manner.
- The number of users accessing the Internet increases at an exponential rate. This includes the indirect users within the private sub-networks.

- The increased power of new PCs and servers. Today's personal computer interface (PCI) systems are fast enough to move files at 30 to 90 Mbps, easily overloading the actual 8- to 9-Mbps throughput capacity of a shared Ethernet network channel (Cisco, 1999). The speed and bandwidth of these desktop machines, the size of popular Internet files, and the size of attachments sent via e-mail continue to increase at an accelerating pace.

The main reason behind network congestion is the long delay incurred during the data transmission between the distance server and the clients. Each client needs to make a WAN connection in order to link to the server. Thus, the number of WAN connections grows at direct proportion to the number of clients. Some of these connections are actually directing to the same server, as in the case of web browsing and video broadcasting applications. Ignoring these similarities resulting wastage of network bandwidth and causing unnecessary network congestion simply because the network is flooded with similar packets.

The key to good network design is to shorten the distance between the server and the client, i.e. to place the server in relation to clients. Ideally, the server should be placed on the same network as the clients. This minimizes the load on the network backbone which carries WAN traffic between networks, while maximizing the LAN traffic within the local network.

Server overloading is another common problem in the network environment nowadays. In a single server network environment, the server is subjected to high utilization and is likely to be overloaded. When the server reaches its overloading level, client requests may take longer time to serve. Eventually this may lead to a complete failure, as the server is no longer reliable and capable in handling too many requests.

One of the approaches taken to solve this problem is to replicate and distribute the server into different geographical areas. This indeed solves the single server bottleneck deficiency and improves the overall network performance as the clients are redirected to the nearest server. However, this does not guarantee that the nearest server is situated very closely to the clients unless a large number of servers are distributed evenly across all networks. Moreover, implementing server replications imposes extra costs and maintenance problems such as data consistency and software upgrading.

A more promising solution is to set up a client-side server called the proxy in the client network. This proxy is responsible to offload the central server by caching the most frequently accessed resources for follow up client requests. In this project, a dynamic proxy solution called the CHEK Proxy Framework (CPF) is implemented to solve both network congestion and server overloading problems. This proxy provides an application harness or platform which is able to execute a mirror of the server application in the client sub-network according to the types of applications.

Proxy Server

Proxy refers to a server sitting between the central server (service provider) and client. It forms an interface which the clients must conform to in order to communicate with the remote server. In a LAN environment, a proxy serves as the gateway for all client requests. It intercepts, observes and controls how traffic flows in and out of the network. The most obvious benefits are the capability in caching most frequently accessed data and masquerading client IP addresses as if there is only one active machine to the Internet. Clients are no longer making WAN connections unnecessarily as resources are located in the LAN.

Although proxy can improve throughput performance, the support for interactive systems such as the video broadcasting system and chat system is still absent. The current available proxies cache only static web data such as those required by the web browser. No proxies are capable of supporting interactive multimedia data in a sharable manner as needed by the multimedia or collaborative clients in a LAN. As a result, establishing separate virtual WAN connections to the central server are still required for these types of applications.

Proxies work at different levels from the application point of view. Standard proxies (Netscape, 1996; Wingate; WinProxy, 1997) provide a network level abstraction in forming an underlying network connection from clients to servers. This connection is not established if manual pre-configurations in both clients and proxies are not present.

Network-level proxy routes only network packets which use the protocols such as HTTP, FTP, Telnet, POP3, SOCKS, News, Mapped Ports for TCP and UDP, and some proprietary applications protocols as in Real Audio etc. It does not support the execution of any customized applications.

In the proxy's settings, each type of service must be given a port to listen to and the client must communicate only through this pre-configured port in order to get routed to the correct server host. In short, network level proxy is not application-sensitive, it functions only as a generic network packet router for certain pre-allocated service as depicted in Figure 1.

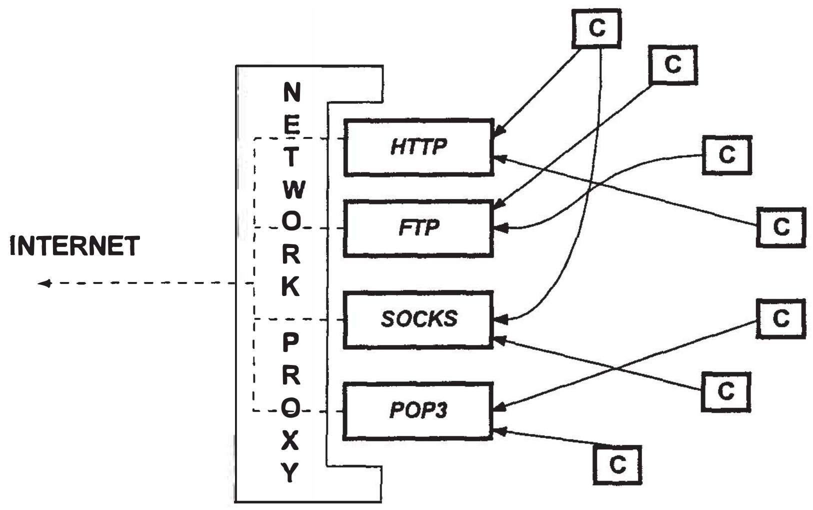

Figure 1: Network Level Proxy with Pre-defined Port Configurations

In addition to the network-level proxy, an application level proxy, which could be either static or dynamic, is introduced. In the static proxy environment, all the proxies are pre-allocated and loaded with static web-contents at fixed network points. These proxies serve only as the contents repositories and are not equipped with any functionality in routing network packets. The purpose of these proxies is to improve the overall network performance by increasing the availability of the data which are needed by the clients.

An example of static proxies is an existing technology called FreeFlow (Akamai Technology, 1998) from Akamai Technology Inc, which pre-deliver its customer web-contents to all the proxies around the world. Client requests are then fulfilled from the nearest proxy for answers rather than from some distant or overloaded server. Figure 2 shows the Front End (FE) processor which schedules and monitors a pool of static proxies. It is responsible for accepting and redirecting clients requests to these proxies.

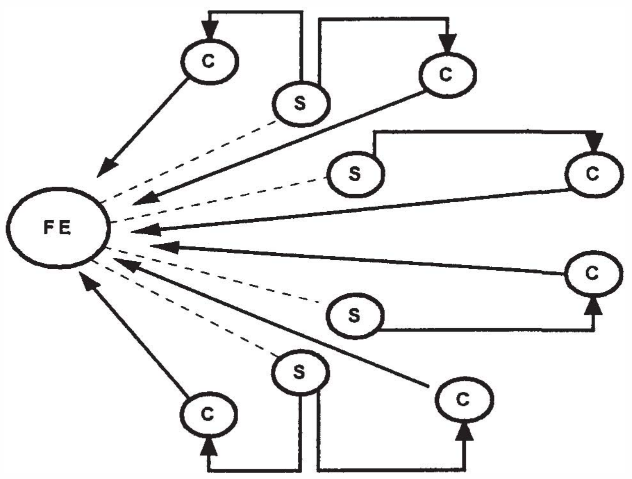

Figure 2: Application Level Proxy with Static Location Configurations

The static proxy solution does not come without any costs and demerits. Firstly, the set up and maintenance fees are high as hundreds of proxy machines are connected with high speed networks around the world. The maintenance and trouble-shooting processes become difficult as software upgrading and inspection have to be done on the site. Moreover, the nearest proxy could be actually quite far due to uneven distribution. At the moment, Akamai has only one proxy server installed in Indonesia to cater the needs of all the clients in the Asia region. As a result, the performance is not fully guaranteed for all clients.

To reduce the cost and software maintenance problems of the static proxy, a dynamic proxy solution is presented. This proxy is created dynamically based on the client requests without any user interventions. This provides better guarantees in terms of performance as the service provider (proxy) is created with the principle of locality in mind.

Dynamic Proxy Server

A dynamic proxy does not distribute only specific web-contents, but also offers an application harness or platform, which supports the execution of the server-side applications at the nearest machine to the client requests. With the capability of binding any application object on the proxies, the main server is allowed to delegate its tasks to a place nearer to the requests. The capability of creating any application in the client sub-network enables the remote server to gain the performance and proximity advantages of the sub-network such as performing local IP multicasting, which could not be done dynamically and transparently in the past. The proxy serves as a local service provider which is responsible for handling all local client requests with a faster response and manual-free configuration.

The dynamic proxy is analogous to the supervisor in a company who looks after a group of workers for a particular job. As the manager is always busy, workers are expected to report only to their supervisor for any questions and problems. The supervisor then communicates with the respective manager for further discussions. All these scenarios adapt the capability of the hierarchical structure which breaks a big problem into tiny and manageable sub-problems.

The dynamic proxy scheme can be implemented in 3 different ways:

Fixed Machines. Similar to static proxy arrangement, a central server is able to create the dynamic proxy with the appropriate functionality in the pre-allocated machine according to the types of requests. The clients are then routed to the selected proxy machine for further requests. However, the hardware and software cost, and the uneven distribution problems still exist.

Any Client Machine. Creating the proxy at the client-end seems to be feasible solution because the services are brought to the nearest location to the requests. However, this logical closeness does not induce a physical closeness advantage to other clients because of the existing physical network connections. Furthermore, the client has no obligation to share its machine resources to other clients in the Internet.

Sub-network Client Machines. Implementing the proxy within a sub-network provides better promise in delivering data as all sub-network clients are both logically and physically connected. A powerful machine is first nominated as the CHEK proxy, then it is made responsible by the central server for handling subsequent client requests within the sub-network domain. The CHEK proxy helps in multiplexing and de-multiplexing all sub-network client requests to the central server in an bandwidth-economical manner.

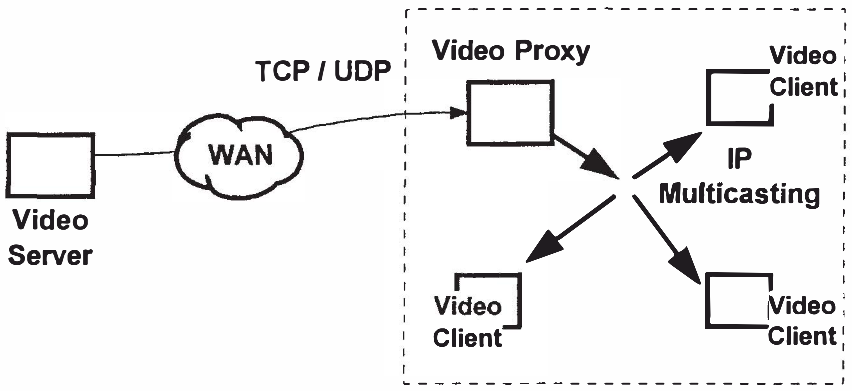

Figure 3: Application Level Proxy with Dynamic Location Configurations

The usefulness of the dynamic CPF is best demonstrated with the video broadcasting system as shown in Figure 3. A remote video server is made possible of multicasting the video data to all sub-network video clients via the dynamic video proxy. Besides reducing the number of WAN connections, CPF removes the static network configurations as needed in the video proxy and video clients.

In short, the dynamic CPF improves the overall network performance in a cost-effective and efficient manner. With the "on-the-fly" network traffic characterization capability, duplicate network packets are eliminated in the LAN. Furthermore, it simplifies the maintenance procedures as the latest proxy software is always getting delivered automatically upon the client requests, just the way applet is downloaded to the client browser.

Objectives

In this project, a generic CHEK Proxy Framework (CPF) is developed to support the middle-tier application objects (application proxy) in a dynamic and relocatable manner. This framework allows the application proxy to be remote-created in the client sub-networks transparently. It hides away the tedious and error-prone manual configurations needed in setting up a proxy. It is implemented using Java and ObjectSpace Voyager ORB (ObjectSpace, 1997).

A three-tier application architecture is employed in the development because the server, proxy and client map naturally into a three-tier object relationship with each tier performing a set of unique functionalities. The three-tier architecture is used as an effective distributed client/server design to provide increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user (Software Technology Review, 1997), when compared to the two tier arrangement (Software Technology Review, 1998).

The key contributions of the CPF are:

- Conserving network bandwidth by creating a proxy in the client sub-network acting as a local service provider.
- Increasing the server reliability and scalability by delegating part of the server tasks to the client-side proxy.
- Characterizing the network traffic by creating application virtual group which is controlled by a generic proxy harness.
- Reducing network congestion by eliminating similar traffics within the same sub-network as a consequence of characterizing the network traffic.
- Providing a cost effective network improvement because the machine in the sub-network is used to serve all local clients transparently.
- Monitoring the network from a single point through a GUI-based management console where all proxies and clients can be controlled.

Organization

This thesis is divided into seven chapters. Chapter I highlights the problem areas and provides brief explanations on existing technologies. The literature review is presented in Chapter II where the relevant technologies and methods of solving are covered. Chapter III describes the system architecture and an applicable environment of the system. Chapter IV talks about the implementation of CPF and followed by the management aspects of the CPF in Chapter V. The system evaluation and result discussions are provided in Chapter VI. Chapter VII concludes the thesis and highlights some future works.