IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-4, Issue-2 E-ISSN: 2347-2693

File Service Architecture for Heterogeneous Clients

Sumaya Sanober[1*] and Amtul Waheed[2]

[1*,2]*Department of Computer Science and Information,* College of Arts and Science,
Prince Sattam Bin Abdul Aziz University, Saudi Arabia

www.ijcseonline.org

Received: 11/Jan/2016 Revised: 18/Jan/2016 Accepted: 03/Feb/2016 Published: 29/Feb/ 2016

Abstract- Internet usage has gone far beyond expectation and continues to grow. Services on the Internet are also more innovative than ever before. Many more different system architectures are currently used to obtain the same pool of services. Now, the trend is to make uniform architecture to provide services. SOA is a service that aims to integrate and be interoperable with various implementation languages. SOA is a concept of distributed computing, the best example of which is file sharing. Remote sharing remains an issue. Although many services are available to provide remote file sharing, they have problems in accessing file with ease, user friendliness, a generic communication protocol, a secured method to communicate during transferring and traversing the secured firewalls. This paper focuses on how easily the file can be shared with the generic communication protocol and a secured method by bypassing the issues of NAT. To provide such service, we do not have any option other than XML, which enables the system to service the end users, who can choose to download or upload files using a browser or an application client. Thus, we have built a middleware for file sharing. The middleware and the application client are equipped with the NAT facilities to traverse the firewalls. Our proposed middleware can service the users of both systems (Browser and Application client). The performance of the proposed system for remote sharing suggests using XML as an interface to implement the SOA-based applications.

Keywords— File Service, Distributed Systems, XML, SOA, WEB Service

I. INTRODUCTION

Research on the Internet has provided countless opportunities to develop innovative applications and methods of communication [1]. The Internet has a critical function in human life because its users can easily obtain information and conduct interactive communication [2]. Web browsers are required to retrieve text and image information and facilitate social interaction online. Web browsers use the hypertext transfer protocol (HTTP) to retrieve information from websites [3]. HTTP is the most commonly used Internet protocol in the world [4], and information providers use it to enable interactive communication. Among these operations, Web browsers can transfer files as attachments through email services. Email service providers use Web browsers to enable instant messaging (IM) and audio and video communications [5]. However, Web browsers have certain limitations in transferring files. Service- oriented architecture (SOA) is becoming a more popular technology to facilitate Web services because it allows interoperability and application integration [6]. Interoperability provides a platform for application developers to access the SOA through standard languages and protocols. SOA is Flexible, Replaceable, and Scalable Fault – tolerant and can also create new services from an existing infrastructure, which enables the interface of current technology with heterogeneous applications and technologies. The Web service enables software system designers to support interoperable communicating entities to interact over a network. Figure 1 shows communication is achieved through XML frameworks such as the WSDL, SOAP, and UDDI.

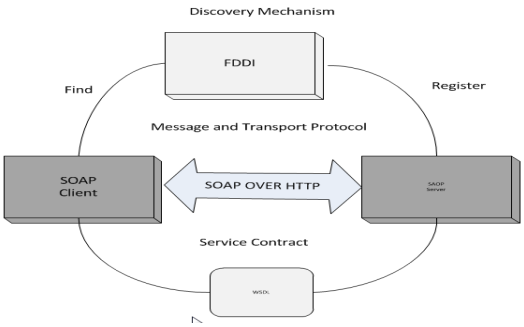

Figure 1: Communication of web services.

A. *Service-Oriented Application Architecture Description*

SOA (service-oriented architecture) is a component model and links different functional units (called services) of the application through well-defined interfaces and contracts between these services. Interface is defined by a neutral manner and it should be independent of implementation services, hardware platforms, operating systems, and programming languages. This allows the service to be built

Corresponding Author: Sumaya Sanober

Department of CSI, College of Arts and Science, Prince Sattam Bin Abdul Aziz University, Saudi Arabia

in a variety of such systems to interact in a uniform and general way .The service is the basis of the SOA; thereby, they can be applied directly and effectively depending on system and interaction of software agents.

B. Web Services Description Language (WSDL)

1. The Web Services Description Language (WSDL) forms the basis for the original Web Services specification.
2. A service provider describes its service using WSDL. This definition is published to a repository of services. The repository could use Universal Description, Discovery, and Integration (UDDI). Other forms of directories could also be used.
3. A service consumer issues one or more queries to the repository to locate a service and determine how to communicate with that service.
4. Part of the WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what the requests and responses are for the service provider.
5. The service consumer uses the WSDL to send a request to the service provider.
6. The service provider provides the expected response to the service consumer.

C. Universal Description, Discovery, and Integration (UDDI)

The UDDI registry was intended to eventually serve as a means of "discovering" Web Services described using WSDL. The idea is that the UDDI registry can be searched in various ways to obtain contact information and the Web Services available for various organizations. How much "discovery" was ever used is open to discussion. Nevertheless, even without the discovery portion, the UDDI registry is a way to keep up-to-date on the Web Services your organization currently uses. It can be used at design time and with governance [7].

D. Simple Access Control Protocol (SOAP)

SOAP essentially exchanges the information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, HTTP is the familiar connection we all use for the Internet. In fact, it is the pervasiveness of HTTP connections that will help drive the adoption of Web Services [8].

II. LITERATURE REVIEW

Several tools can transfer files from a source to a destination via the Internet. Files can be transferred as email attachments through multipurpose internet mail extensions (MIME) [9]. Applications such as IM are also heavily used and typically bundled with file transfer capabilities [10]. However, file transfers among remote users require logging into the IM system [11, 12]. The Microsoft Windows operating system allows users to share files in the network with the permission of other system users [13]. The file transfer protocol (FTP) is dedicated to file sharing [14]. In FTP communication, files must first be sent to the FTP server before other users can access a copy of the file through a software application called an FTP client. The peer-to-peer (P2P) system allows users to share (send and receive) files over the Internet [15]. However, P2P systems work outside of the browser, and P2P applications must be downloaded before the files can be transferred. The innovative storage abstraction layer method, which is a file service for mobile devices [18], is used to access cloud services in the heterogeneous environment for mobile devices. Table 1 and table 2 represents method uses Google Docs, FTP services, Amazon S3 services, and others. XML-based services are extended to the ubiquitous environment, where these services are provided to mobile users and new types of devices that can use the mobile networks [19].

Services	Storage	Description
Google Doc	Drop box	Enables user to store information and help to sync their files and folders with their devices.
File Transfer Protocol	Syncplicity	Uses a storage cloud and has the facility to sync files.
Amazon s3	iCloud	Enables users to save their data such as emails, Calendars, and address books.

Table 1: Cloud Services in Heterogeneous Environment

System	Description
Coda(constant Data Availability)	System that works even when Internet access is disconnected or has Low bandwidth.
LBF(Pound Force)	System can provide service under low-bandwidth conditions and Can be installed on already running systems.
GridFTP	System that is secure and reliable in which data transmission method For High-bandwidth networks

Table 2: Network Based File Storage

A. *Problem Statement*

.NAT (Network Address Translation) traversal encountered certain problems like connectivity, Security, Scalability and improved support for quality of service (Qos). NAT makes application compatibility more difficult since NAT tampers with IP header fields which cause issues with File Transfer Protocol (FTP), when communicating with a global server through NAT devices. The most common method to allow incoming and outgoing traffic is to configure routers to allow global ports and global Internet protocols to communicate with NAT clients. However, this solution may give intruders an opportunity to disturb the system

B. *Proposed Solution*

Inspired by the characteristics of data the design of middleware with the service-oriented architecture was employed in this research, compatible with various types of data and the agreement has been divided. Consequently, this paper presents the basic framework of SOA- based applications, Service providers (producers) use of various types of environmental sensing technology. Data process sing platform is responsible for data processing, data filtering, and data integrity. It provides XML scheme for data unification and metadata consistency and standardization of heterogeneous data processing. In our study, we built the middleware to enable the file transfer among heterogeneous architectures. File transfer is conducted between 2 browsers within application clients and between browser and application clients, but the communication among these entities is achieved through only the XML protocol.

We obtained information about the existing file transfer methods and concluded that they should have certain features to provide better services. We propose a solution that provides easy access to files, a user-friendly GUI, a generic protocol to communicate, a secure transmission method, and a method to prevent network address translation (NAT) issues. Our system offers improvements in the following aspects:

a) **Accessibility:** Easy access to files. Currently, the users have only two options to download and to upload remote files: through a web browser and through an application client.

b) **Coupling:** User friendliness. The existing solutions provide the user with only one choice. The users must register to the service to transfer files either with browsers or applications but not with both options together. In our system, the users can download or install the application client to obtain the file transfer service, or they can use the browser to transfer files.

c) **Availability:** Generic communication protocol. SOA is currently the only solution to achieve service availability. SOA is easily accessible through any program, and the user must not worry about the system architecture and implementation details. Thus, our project uses Web services as the middleware to provide a file transfer service.

d) **Reliability:** Secure file transfer. In our system, using an encrypted communication channel between file transferring and the receiving client is not necessary except when the clients are not securely communicating with each other. The proposed architecture uses the email service as the secure mode of communication. Our system generates a random number for each uploaded file, and the random number is sent to the intended recipient of the file.

III. File Transfer Architecture

"client-server" describes the relationship between two computer programs where the client makes request to the server. This relationship is seen in such tasks as sending emails and accessing the web. Many business applications currently use the client-server model as well as the main Internet application protocols (HTTP, SMTP, telnet etc.)Servers would include web servers, FTP servers, application servers, mail servers, file servers, and most web services.

Client-server architecture enables the responsibilities of a system to be distributed amongst several individual computers. These individual computers are accessed only through a network so it is easier to maintain and repair or even relocate a server without affecting the clients. Another benefit exists in that data is stored on servers which generally have greater security than clients. Servers can also control user access and resources. One example Â of an innovator in this field is Ipswich – a software company which has been offering FTP software since 1991.As the field of file transfer continues to grow, so will the need for safe servers and happy clients who together form a healthy, networked relationship.

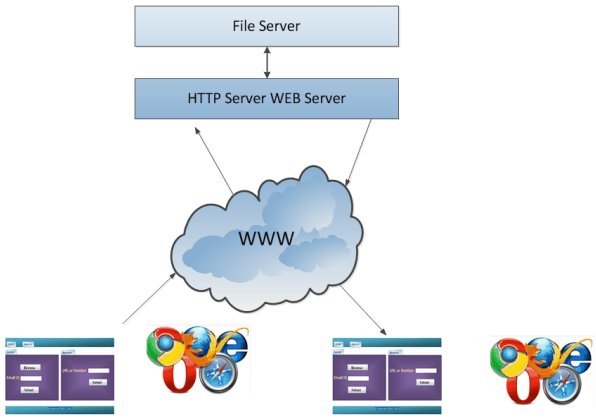

Figure 2: file-transferring architecture

Figure 2 illustrates the file transferring architecture which consists of the Apache Web service to serve an HTTP-based browser, application, or non-browser clients. The file server stores the files. Browser clients can communicate only with the Web server to upload and download files, whereas the

Web server can communicate with the file server to retrieve and upload the files. The Apache Web service aims to serve the file transfer requirements of application clients. Figure 1 presents the file transfer architecture where the browsers or clients are used to receive and send files. This architecture supports a large number of clients. In the proposed system, the transfer browser or application clients send the file to the server. Communication between the Web service and the file server is conducted through socket programming. The browser clients use HTML and JavaScript with PHP to demonstrate the file transfer among browsers and application clients.

IV.METHODOLOGY

A. Client-Server Model

The Web-based file transfer is a client-server model divided into three levels of functionality: Web interface, Web server, and file server figure 2. The client side is considered the initial level of the Web interface, which accepts requests for processing. The server-side architecture is a two-tiered system that consists of Web and file servers. When a client initiates a request, the Web server transfers the data between the client and the file server. The Web server consists of three major modules for file chunking, reconstruction, and transfer.

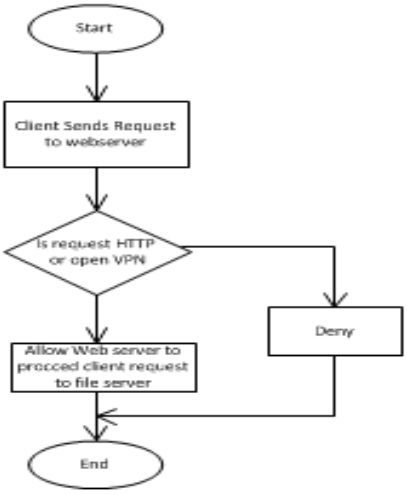

Figure 3: File Transfer Flow Chart for the client server communication.

B. Procedure of browser and web-server communication

When the sender browser wants to share a file, the user must upload the file and provide a valid email address. The HTTP server receives the file and sends it to the file server. Then, the file server generates a link and a random number for the current file. At the receiving side, the recipient browser sends the link or random number to the server. If the link or random number is valid, the server starts sending the file to the recipient browser.

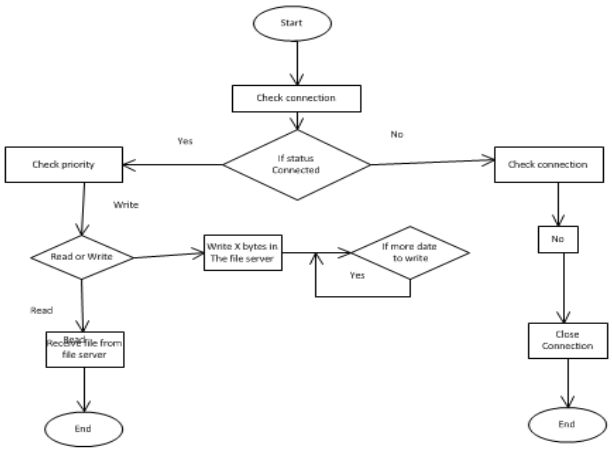

Figure 4: Flow of information between the browser and the WEB service. Initially, the web browser establishes a session with the web server to upload or download a file.

C. Procedure of web-server and file server communication

Figure 4 illustrates the communication at the server side (HTTP and file servers). According to the request received from the clients or browsers (sender or receiver), the HTTP performs tasks such as receiving or sending files. If the HTTP server receives a request from a client to upload a file, it establishes a relay session with the file server to store the requested file and accordingly generates a URL link or a random number for the file to be stored. The HTTP server also acknowledges the sender client through the link or the random number.

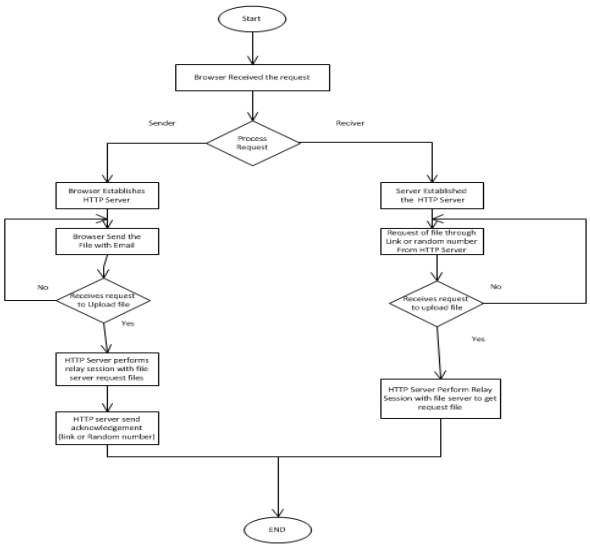

Figure 5:.Flow of information between the browser and the WEB service according the request received by the server to transfer the files.

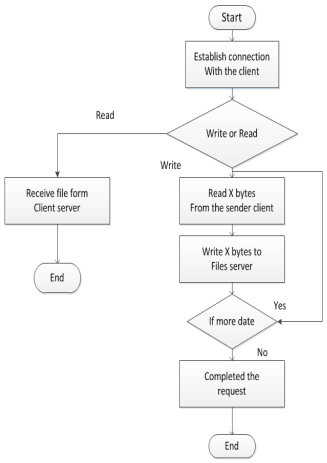

Figure 6: File transfer flow chart between the server and the file server.

Figure 6 shows the communication between the http and file servers. Initially, the HTTP server waits for connections from the clients. If the connection is POST, the server reads the files from the client (i.e., from the sender browser) until it writes the file to the file server. After the file is written to the file server, the latter generates a link or random number for the written file. The communication between the HTTP server and the file server is described as a relay session.

V.IMPLEMENTATION

The Apache server is a secured server that provides HTTP services for HTTP standards [16]. The HTTP server that hosts the file transfer method is exposed to one or more clients such as browsers or relay clients. These relay clients (browsers) are under the control and services provided by the HTTP server, which implements the previously described protocol for HTTP requests by other clients. The JavaScript part of the hosted HTTP server is also configured with the URLs of one or more file servers, so that it can be used to store and retrieve files. Instead of the clients making a direct HTTP request to the server for communication, the Web browser builds an HTTP request pipeline, encloses it in a normal HTTP request, and sends the enclosed request to the HTTP server. The clients support the embedded messages, which are enclosed requests with a content type of message or HTTP. The clients also support embedded pipelines with a content type of application or HTTP. With the limitations of the browser in uploading files, one possible method to share files faster and more safely is to upload the files in separate byte chunks. Then, the file server method reconstructs the file content in the correct order. Figure 5 presents a screen shot of the prototype. To examine the feasibility of the model, the prototype system was implemented using PHP, HTML, JavaScript, and the Apache Web Service server.

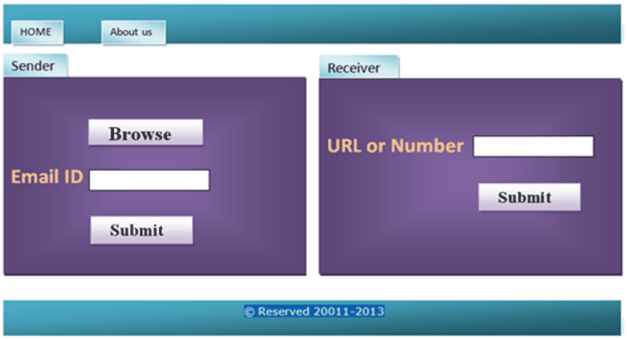

Figure 7: Browser Client, which was built using PHP, HTML, and Java Script to transfer files among the clients through a server.

Application architecture: it consist of three layers

1) The presentation layer: user interface consisting of HTML and JavaServer Page (JSP) forms;

2) The business layer: processing and interpretation of XML document;

3) The data layer: provides access to specified data source

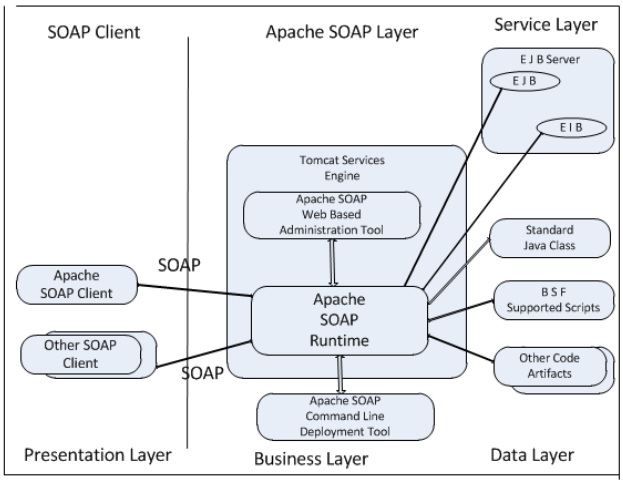

Figure 8: *Application architecture*

For applications, we used the Apache SOAP Web service. The Apache SOAP Project is a Java-based implementation for SOAP version 1.1. SOAP is a communication protocol that provides data transportation for Web services [17]. Each method is associated with a Soap Document Method Attribute that defines the SOAP actions and connects the Apache SOAP service to obtain or set information in the HTTP header of SOAP requests or responses. The client also uses the XML encoding style to format parameters in the body of the SOAP request. For NAT, we have a proxy class on the client side and a global proxy method on the server side, which interact with each other when the client is running. These methods do not flood the network by polling with regular intervals because the application clients only

serve the users when receiving or transferring files. After the client comes in contact with the server, it obtains the list of clients who are currently using the server.

Flow of communication from user (client) to file server:

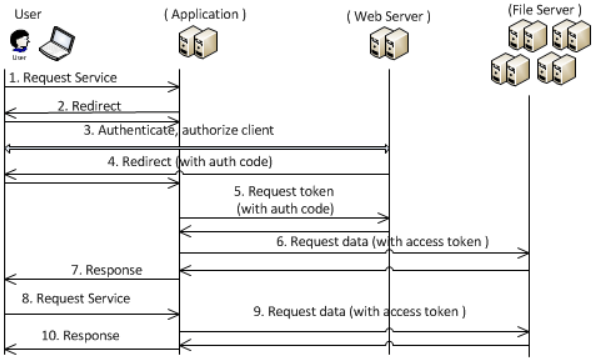

Figure 9: Communications between client and web server

Parsing the XML message for communication between client and web server:

The client will send a connection request to the server.

- Connection Information:

< connection – info >

< connection – url > …… < / connection – url >

< / connection – info >

The server will validate the client's authentication details.

- Authentication Information:

< authentication – info >

< auth – name > …. . < / auth – name >

< auth – code > …… < / auth – code >

< auth – role > ……. < / auth – role >

< /authentication – info >

The client can send a message (command) to the server.

- Request message with a specific command:

< request >

< command > …… < / command >

[command parameters]

< / request >

The server will process the request and return a response to the client.

- Response message to:

< response >

< command > …. < / command >

[response details]

< / response >

ROUTING MESSAGES TO THE CORRECT FILE SERVER:

After parsing the XML message received from a client application, the XML gateway requires a method to determine the correct file server to which to send the messages [21].

The following three methods were considered:

1) Using a unique address location mechanism: Many request commands require (as an input parameter) some sort of unique address or field that indicates where the data can be located. This address may be a directory location, a database table and/or field name, a file name, a network node and file name, and many other possible location-type variables.

2) Using a specific XML element: option two considers including an additional element (the protocol element) to the XML model that will identify the server application that is a child element of the request element. This means that all commands within the request element will be directed to the identified file server i.e.

< request >

< protocol > LDAP

< command > read < / command >

< command > modify < / command >

< / protocol >

< / request >

This approach requires that the client application specify, in the creation of the XML document, the server for which server the requests are intended.

3) Trying the first available (connected) server application specified in the connection URL:

Option three attempts a connection to each server application specified in the connection URL. The request is sent to the first server application that is available and the result returned to the client.

VI. RESULTS

Communication between the Web and the file servers is achieved using C socket programming. The backbone of our architecture is the file server, which hosts the files of the users and is a POSIX standard server. The file server implements the cache manager, which caches and manages the most frequently used commands by the clients in our environment. The server hosts a dedicated directory for every client. Thus, an active client works on the space provided by the server. The client-side cache manager always logs its data onto the log file. The log file contains information such as date, time, and all attributes of the files, and it also maintains records of the uploaded, removed, and downloaded files. On the client side, when the user works in the space to retrieve information about the files or data, the cache manager retrieves the information from the log instead of contacting the server. Table 1 presents the

algorithm to retrieve data from the server in the client workspace.

1. Connect to the server
2. If (clientlog.feild == serverlog.feild)
3. The, display the attibutes of file directories
4. Else;
5. UpdateLogFileData (Client ID, Client, Path);
6. End

Algorithm 1. Retrieval of Attributes File

Table 3: Retrieval of File attributes. Data are retrieved from the server only when there is a mismatch in their log file IDs at the server and the client.

The log file data are shared with the cache of the respective clients from the server, so that information can be made available locally instead of remotely to the clients who request to browse the workspace. The log files on either side are shared by the server after the transaction is conducted on the server. The client side cache is only updated after the transaction. The local cache can reduce the overhead of data retrieval from the remote server. Fig. 6 indicates that with a cache, the overhead is much lower than that without a cache.

The "no cache" condition requires contacting and retrieving data from the server, whereas with a cache, the client must not retrieve the workspace attributes; thus, the overhead is drastically reduced. However, the time taken to perform a task such as removing a file from the server remains identical for both conditions. Downloading and uploading files require approximately the same amount of time with or without a cache because these tasks require interaction with the remote server. The listing of file attributes does not require interaction with the server, which reduces the overhead.

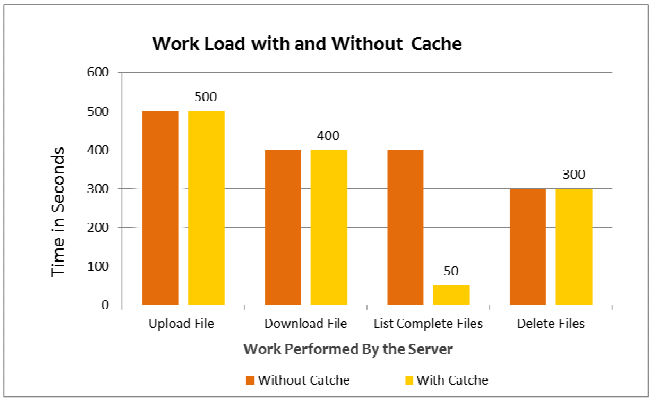

Figure 10: Performance of the server with and without the cache manager. The Y axis shows the time in seconds, and the X axis shows the work performed on the server.

VII. Conclusion

File sharing services have increasingly attracted attention over the years. In this paper, we presented XML-based middleware file service architecture. Our proposed solution focuses on various aspects such as ease of access, user friendliness, secure transmission, and NAT traversal. For ease of access to the remote file, we provided options to download using a Web browser or an application client. To obtain a secure method of transmission, we used the services of email providers to receive files. The proposed solution can traverse any NAT network. The results of our study show that by using XML, we reduced the overhead of messages per second across the network. By applying a cache, we minimized the overhead involved in browsing remote files. We envision that the use of this solution can be extended to heterogeneous mobile devices.

References

[1] Lee, A.; Girgensohn, A.; Zhang, J., "Browsers to support awareness and social interaction," *Computer Graphics and Applications, IEEE* , vol.24, no.5, pp.66,75, Sept.-Oct. 2004 doi: 10.1109/MCG.2004.24

[2] Heath, T., "How Will We Interact with the Web of Data?," Internet Computing, IEEE , vol.12, no.5, pp.88,91, Sept.-Oct.2008 doi: 10.1109/MIC.2008.101

[3] Hanson, V.L.; Brezin, J.P.; Crayne, S.; Keates, S.; Kjeldsen, R.; Richards, J.T.; Swart, C.; Trewin, S., "Improving Web accessibility through an enhanced open-source browser," IBM Systems Journal , vol.44, no.3, pp.573,588, 2005 doi: 10.1147/sj.443.0573

[4] Support Protocols, In: Thomas Porter, Jan Kanclirz, Andy Zmolek, Antonio Rosela, Michael Cross, Larry Chaffin, Brian Baskin and Choon Shim, Editor(s), Practical VoIP Security, Syngress, Burlington, 2006, Pages 205-237

[5] Rennie, J.; Zorpette, G., "The social era of the web starts now," Spectrum, IEEE , vol.48, no.6, pp.30,33, June 2011.

[6] David Chen, Guy Doumeingts, François Vernadat, Architectures for enterprise integration and interoperability: Past, present and future, Computers in Industry, Volume 59, Issue 7, September 2008, Pages 647-659, ISSN 0166-3615, 10.1016/j.compind.2007.12.016.

[7] Matjaz B. Juric, Ana Sasa, Bostjan Brumen, Ivan Rozman, WSDL and UDDI extensions for version support in web services, Journal of Systems and Software, Volume 82, Issue 8, August 2009, Pages 1326-1343, ISSN 0164-1212, 10.1016/j.jss.2009.03.001.

[8] Anura Gurugé, 8 - Web Services, Corporate Portals Empowered with XML and Web Services, Digital Press, Burlington, 2002, Pages 245-272, ISBN 9781555582807, 10.1016/B978-155558280-7/50010-3.

[9] P. Resnick Internet message format Request For Comments (RFC), 5322 (2008)

[10] FILE TRANSFER PROTOCOL (FTP), RFC 959.

[11] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. 2001. Freenet: a distributed anonymous information storage and retrieval system. In International workshop on Designing privacy enhancing technol0ogies: design issues in anonymity and unobservability, Hannes

Federrath (Ed.). Springer-Verlag New York, Inc., New York, NY, USA, 46-66.
[12] Napster. http://www.napster.com
[13] Gnutella. http://www.gnutella.co.uk.
[14] Huajian Mao, Nong Xiao, Weisong Shi, Yutong Lu, Wukong: A cloud-oriented file service for mobile Internet devices, Journal of Parallel and Distributed Computing, Volume 72, Issue 2, February 2012, Pages 171-184, ISSN 0743-7315, 10.1016/j.jpdc.2011.10.017.

[15] Ivar Jørstad, Do Thanh van, Personalised ubiquitous file access with XML Web Services, Computer Networks, Volume 51, Issue 16, 14 November 2007, Pages 4655-4668, ISSN 1389-1286, 10.1016/j.comnet.2007.06.009
[16] Dropbox, http://www.dropbox.com
[17] Google Docs, http://docs.google.com
[18] iClouds, https://www.icloud.com/
[19] Coda file http://www.coda.cs.cmu.edu
[20] Addison-Wesley Professional "Understanding web services "1- edition (May 23, 2002)
[21] Suvendi Chinnappen-Rimer and Gerhard P. Hancke, Senior Member "An XML Model for Use across Heterogeneous Client–Server Applications", IEEE Transactions on Instrumentation and Measurement, VOL. 57, NO. 10, October 2008
[22] Girish M. Tere et.al / International Journal on Computer Science and Engineering (IJCSE) "JAX-WS Web Service for Transferring Image" ISSN : 0975-3397 Vol. 5,Page No- 196 03 Mar 2013

AUTHORS PROFILE

Mrs. Sumaya Sanober pursed Bachelor of Science from Osmania University, Hyderabad India in 2006 and Master of Computer Application from Osmania University in year 2009. She is currently working as lecturer in Computer Science and Information Department in College of Arts and Science, Prince Sattam Bin Abdul Aziz University, Saudi Arabia since 2012. Her research interest includes Web Services, Wireless Network, Cloud Computing and Testing Tools.

Mrs. Amtul Waheed pursed Bachelor of Computer Application from Osmania University, Hyderabad, India in 2003, and Master of Computer Application from Osmania University in year 2006 and Master of Technology from Jawaharlal Nehru Technological University. She is currently working as lecturer in Computer Science and Information Department in College of Arts and Science, Prince Sattam Bin Abdul Aziz University, Saudi Arabia since 2011. Her research interest includes Wireless Sensor Network, Web Services and Cloud computing.

