isrjournals

International Journal of Advanced Research in Computer Science Engineering and Information Technology

Volume: 6 Issue: 1 Jul,2016,ISSN_NO: 2321-3337

Creating Web Server in Android Mobile and Easy Serving of Information to Clients

First G.Koteswara Rao, Second R.Jeya

Abstract— **Android is software platform and operating system for mobile devices. Being an open-source, it is based on the Linux kernel. It was developed by Google and later the Open Handset Alliance (OHA). It allows writing managed code in the Java language. Due to Android here is the possibility to write applications in other languages and compiling it to ARM native code. In this paper a mobile-based web server was introduced for serving static HTML/JavaScript pages to the client systems (the systems could be PCs or Mobiles) for access by these client systems. A static website will be designed to serve from the mobile (web server). The Mobile Web server is based on Android OS. The static website saved inside the SD Card of the Android Mobile would be the source for the web server, thereby increasing the capability of the Mobile to become a web server for even PCs in the Network created by Android Mobile itself.**

Key words: **Webserver, Wireless Fidelity, Tethering Hotspot, Android Operating System, Multi-threading, File Sharing, Peers, Mobile Hosting**

I. INTRODUCTION

The exchange of web content from server to client and the connectivity of the clients imposing challenges with increasing usage of mobile communications. Mobile webserver deals with lot of problems with normal servers which always need LAN connections and stationary which is not portable and face many challenges. This approach affords a number of critical benefits including a portable server and hosting files even without Wi-Fi-connectivity. Mobile Hosting serves one simple but fundamental purpose; they allow end users to upload the files or static web content to mobiles and to be shared among interested users without any connectivity. No need to maintain a dedicated server at a place and connect to it with internet connectivity. Most such services simply return a URL that can be shared to others who download the content or navigate through the webpages. . The development of web server involves creating threads for running server and clients. The exchange of web content from server to client and the connectivity of the clients is managed by sockets created using Socket programming concepts and multi-threading.

As online file hosting systems become increasingly popular, however, server bandwidth costs have become prohibitively expensive, as files are hosted in either content distribution networks or dedicated large data centers. Rapid share [2], one of the most well-known one-click hosting systems, deployed a total of 1,500 terabytes of online storage in its data centers in Asia alone. Skyrocketing bandwidth costs from server-based architectures have made it necessary for all online file hosting systems to impose certain restrictions so that they can afford to remain free of charge to users. These restrictions include download bandwidth limits per day, file size limitations, as well as a maximum time period that files may remain available online.

Though it may seem intuitive to take advantage of peer bandwidth contributions to mitigate server bandwidth costs, the architectural and protocol design of such a peer assisted online file hosting system should not be taken lightly. It is nontrivial to design and fine-tune a new system that utilizes peer bandwidth contributions in a complementary and transparent fashion, without sacrificing the ease of use, reliability, and performance of one-click hosting services. The architectural design should be able to scale to a large number of users, and to withstand the test of real world usage over a long period of time.

In this paper, we present Mobile Webserver, a file hosting system that is centrally

controlled from a smart mobile itself. Mobile Webserver is designed to mitigate the connectivity issues along with maintenance problems that occur with stationary servers, while supporting the ease of use and performance comparable to best server-based solutions.

II. Problem Statement

In this section, we describe the problem and motivate the need for information to design the mobile webserver. Designing a server which serves static web server that runs on an android mobile and from this web server, the files uploaded by the mobile user into his android mobile, are made available to all the PCs or Laptops within the network in which the mobile is connected.

A. General Webserver

A Web server is a computer that delivers (serves up) Web pages. Any computer can be turned into a Web server by installing server software and connecting the machine to the Internet. The communication that occurs between a Web server and client browser is made possible through several protocols. When you are browsing the Web the protocol you use most often is called the Hypertext Transfer Protocol (HTTP). HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

1. User's browser initiates a connection to the Web server that is storing the files by converting the domain name into an IP address and then locating the server that is storing the information for that IP address.
2. The browser then requests the data from the Web server and the server delivers the data back to your browser using HTTP.
3. The browser converts the file's data into what you see displayed in your browser.

III. Mobile Web Server

The development of mobile web server involves creating threads for running server and clients. The exchange of web content from server to client and the connectivity of the clients is managed by sockets created using Socket programming concepts and multi-threading.

A. Problems with existing systems

The existing system has web servers that are confined to only wired devices like Laptops and standard systems. A Wireless LAN Adapter is required in all devices that need to access the webserver. Legacy systems like PCs cannot access the files for the web server as there is no provision for Wi-Fi in these systems. When there is no internet connectivity, system fails to contribute the very purpose of communication.

B. How Mobile server works

A Wireless LAN Adapter is required in all devices that need to access the webserver. Legacy systems like PCs cannot access the files for the web server as there is no provision for Wi-Fi in these systems. Even Legacy systems can access the web server. It is based on two –tier architecture can be extended further for enterprise applications.

C. Processing

Processing of the mobile webservers requests is done in a sequential manner and is subdivided into few stages of execution. They are Discovering Static content followed by Dynamic client content Processing, then Server and Server Handler stages.

1) Discovering Static content: Designing the static html pages that are to be served by the Android Mobile to other Android Mobiles or even PCs in network. Static content is referred to as the HTML webpages that are to be hosted from the mobile. It will be stored in a desired location in mobile. Adding

authentication schemes at client side in this stage improves the client server architecture. Website is edited and is stored in the SD card of the mobile and storage requirement of the mobile can be enhanced by using larger memory slotted SDcards.

2) Dynamic client content Processing: This stage is to display the dynamic client content in the browsers of client mobiles or client PCs. Whenever the server is started, it provides a IP address with port number associated with that application. Now, client enters in the URL field those details and the static content will be loaded on the browsers of clients.

3) *Server:* This stage is required for multi-threading the webpage's to various clients connected to this module by synchronizing the data for each client and by starting the socket connection between the server and a given client.

4) *Server Handler:* Server Handler mediates between the Server and Clients by calling the connection from the server to accept the requests from the clients. Multi-client processing is achieved in this stage.

D. General Activity Life Cycle:

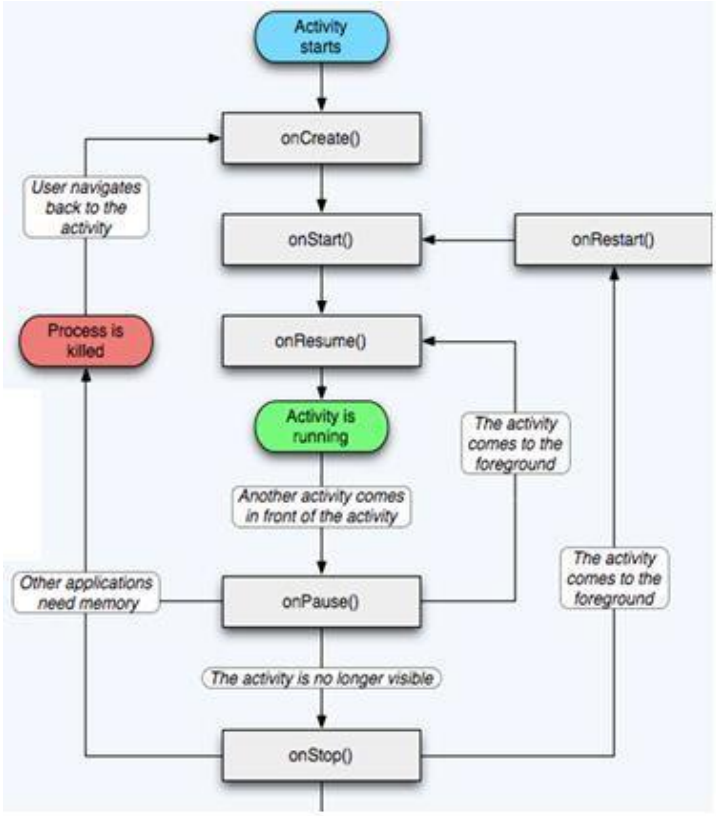


Fig. 1 Activity Life Cycle (android)

E. Sequence of events:

Step 1: Start Server in listener mode.

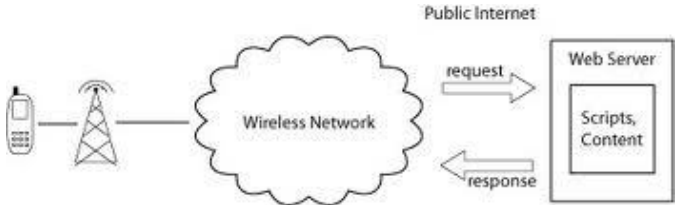
Step 2: Retrieve IP address of the network and provide port for running the application on client side using sockets.

Step 3: Call Server stage whenever a new connection request is made to the server.

Step 4: Server will create a socket connection with specified port and then adds clients to the client queue.

Step 5: Server then calls the Server handler to handle the requests from the client and waits for another connection.

Step 6: If all the requests are acknowledged and new connections are not allowed, then Stop Server.



IV. File Sharing

Peer-to-peer file-sharing networks enable users to "publish" or "share" files – any file from music to video to spreadsheets. P2P networks provide a ready-made sharing infrastructure that is difficult to block and even harder to track, providing cover for espionage and criminal activity. They encourage users to leave their computers on and connected to the internet at all times, running software that heavily uses their network, disk, and processor. Recent legal battles being won by the content industry (RIAA/MPAA) seem to have done little to really reduce file sharing, but have rather pushed users onto new clients and networks that are even harder track. Current P2P networks are designed "publish" or to "share" data. The user configures the client software to share items in a particular folder, and directs the client to move particular files and deposit them in that folder. In normal operation, a P2P client simply writes files to disk as it downloads them and reads files from disk as it uploads them. There are several routes for confidential data to get on to the network: 1) a user accidentally shares folders containing the information; 2) a user stores music and other data in the same folder that is shared; 3) a piece of software the user has chosen to download and execute surreptitiously shares it (malware); 4) client software bugs result in unintentional sharing of file directories.

A. How Does Sensitive Information Become Exposed?

It is often not necessary for a worm or virus to expose personal or sensitive documents because many users will expose these documents unknowingly. Multiple reasons exist why users might expose personal or sensitive information:

• Misplaced Files – If a file is dropped accidentally into the wrong folder. Users may simply forget about the letter they wrote to the bank, or the documents from work they brought home one night. Similarly, teenagers using P2P may not know what Dad keeps on his Desktop.

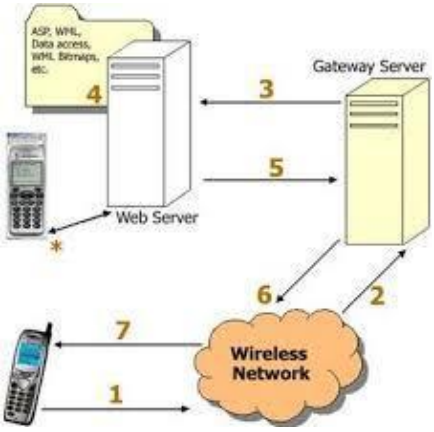
• Confusing Interface Design – Users may be unaware of what folders are being shared or even that they are sharing files. For example, in a user study, Good and Krekelberg found that the KaZaA interface design contributed to user confusion over what files were being shared.

• Incentives to Share a Large Number of Files – Certain programs reward users for making files available or uploading more files. Some users may believe they can gain an advantage by sharing their entire hard drives.

• General Laziness on the Part of the User – If a user has a folder such as "My Documents" with many media folders inside, they may share My Documents rather than selecting each media folder individually to share, thus exposing all the other types of documents and folders contained within.

• Wizards designed to determine media folders – Some sharing clients come with wizards that scan an individual's computer and recommend folders containing media to share. If there is an MP3 or image file in a folder with important documents, that entire folder could be exposed by such a wizard.

• Poor Organization Habits –



Certain people may not take the time to organize their computers. MP3s, videos, letters, papers, passwords, and family pictures may all be kept in the same folder. Many of these reasons point to the interface design and features of P2P clients that facilitate inadvertent sharing. In many ways, the security risk of P2P clients is similar to Trojan horses, malware, and phishing scams: security breaches that depend on human intervention, abetted by a carelessness or lack of proper security education among users. The remedies are also similar: P2P network monitoring, user education, proper controls on corporate information, site blocking, periodic tests. We believe that the vast majority of information leaks are the result of such accidentally shared data rather than the result of malicious outsiders extracting data from an organization. However, there are many other trends that are driving more security concerns.

B. Growing Usage and Network Heterogeneity Means More Leaks

The amount of data being shared will likely continue to increase as P2P network usage grows. Assuming that current usage patterns persist, more and more confidential information will find its way on to these networks. Despite the significant positive network effects associated with using a particular P2P client (the larger the network, the more diverse the content, the greater the reliability, and the greater the speed), P2P networks are far more heterogeneous and faster moving than operating systems. With many networks and clients, users are not likely to grasp the security issues and P2P developers will likely not focus on security.

C. Set and Forget Increases Losses

The P2P usage model is very different than the standard internet model, and potentially reduces security. P2P Traffic load, like most internet traffic, is cyclical. P2Ptraffic peaks at 10:00 p.m. EST. significantly; studies have shown that peak traffic is only double the traffic at the slowest time of day. In comparison, peak web traffic levels tend to be five times greater than minimum web traffic levels. If we use web traffic as a proxy for a user's presence at the keyboard, this indicates that P2P programs are being left running unattended. Indeed, research indicates that P2P clients tend to be "set and forget" applications that run in the background and while the user is not at the computer [14]. This suggests that the user is not carefully tracking the activities of the P2P client,

increasing the opportunity for abuse. Further, even benign file sharing programs consume a good amount of processor time and network bandwidth, conditioning the P2P user to tolerate sluggish performance that, for others, might be a first sign that a system has been compromised. For typical internet applications, the vast majority of consumer traffic is downloads (web pages and email, complemented by the occasional outgoing message). Indeed, most ISPs have traditionally made the download pipe larger than the upload pipe (ADSL, cable modems). P2P generates significant uploading volume. Previously, significant outbound data on a consumer computer would be a signal to IT staff that a computer is compromised. Now it is par for the course, capable of cloaking a real compromise.

D. No Borders Result in Global Losses

Geography is largely irrelevant in P2P networks, meaning no particular country or region is safer than another. A computer logging on in Bombay or Brussels becomes part of the same network as a computer in Pittsburgh. Some studies, based on networking logs, have shown that hosts that are physically close to each other are only slightly more likely to be connected and sharing files than hosts that are physically distant. In addition, the network retains these characteristics if we dig into a particular region. On the other hand, Fess ant et al.argue that the opposite is true on the eDonkey network. They claim that "for 60% of the files, 80% of the replicas are located in the main country." This supports some level of geographical clustering. Some of this may be explained by the popularity of content: "peers requesting a given video file may in a large proportion of cases download it from peers in their own country, thus achieving low latency and network usage compared to downloading it from a randomly chosen peer." In any case, files certainly migrate globally and threats can come from any corner of the globe.

E. Digital Wind Spreads Files

Second generation P2P networks typically create file indexes using the names of files and metadata associated with them (the MS Word user who created it, for instance, or the company the product is registered to). As a result, malicious users are often searching opportunistically for files with key words or phrases, such as "credit card".

F. Malware

While the overwhelming majority of traffic on P2P networks is entertainment content (games, movies, music, etc.), also lurking on P2P networks are files that pose severe security risks. Viruses and worms that exist in email and other programs also have variants that exist in peer-to-peer networks. A particularly severe virus known as An tinny, appeared on the Japanese-based Winny network that led to the disclosure of a large amount of private data including, U.S. military base security codes, and documents belonging to a police investigator involving a major investigation and 1,500 individuals. An tinny propagates by making copies of itself, disguised to match filenames in a user's share folder or common names on the network. When another user downloads this file and opens it, they are greeted by a common error screen, indicating the file is unreadable. (Some variants even launch Windows Media Player.) In reality, the virus has been launched and opened a backdoor, often making every file on the user's computer available on the network. It may also take screenshots, note the user name, organization, and email address stored by Windows, and make changes to the registry. The file hides its true nature (an "exe" file extension) by cleverly inserting a large number of spaces after the name and the extension of the file it is masquerading as. If a user does not notice the ellipsis indicating a filename longer than can be displayed, they will likely launch the file [45].

V. How we Dealt

Since Port numbers are dynamically changed we can provide authentication to the peers and incoming connections. Ip addresses of the incoming connections are monitored and

is displayed by the server itself. Connections can be dynamically managed. Files are to be maintained carefully that only those which are to be accessed are made available to clients or so called peers.

VI. Conclusions

The android web server is a light weight mobile web server that could be used to serve static web pages from an android mobile to any wireless or wired device like Laptop, Tablet or PC or sometimes even other mobiles which need not be android mobile phones. The static content like uploaded PDF files or PPT files or even Video files in the web server could be accessed by using this web server in any client machine. The access to the web server could be made in the same way as any other PC-based web servers like Apache Tomcat or IIS i.e., via a web browser in the client. The client need not have any heavy software for accessing the content from the static web server, except for a web browser in his device. This feature particularly facilitates a mobile user to share his content to any of clients or colleagues for a promotion of his websites or for transferring easily any content like PPTs, PDFs or Videos to them without much difficulty. The only requirement is that the server could be started with any existing Wi-Fi network available in the premises. This can even make effective use of the Hot Spot technology that is available with android devices, i.e., we can share files or webpages even without Wi-Fi connectivity. Since it is implemented in Android, it can be widely used through Tethering Hot Spot technology also.

VII. Future Enhancements

As per the existing mobile computing standards, the possibility of extending a static web server into a dynamic web server is considered to be a future enhancement for this project. Because with this enhancement, the possibility of writing code using server-side scripting languages is very much possible and this is the most cherished dream of a web developer on a mobile platform. As Android OS is based on Linux Kernel, with little addition of Java APIs for supporting web application with a complete web kit gives opportunity to enhance the functionality of the static web server into a full-fledged dynamic web server. Then the clients can upload any data to the mobile phone acting as webserver and SQLite database functionalities for storing the web data makes it a truly complete web server.

References

1. Beginning Android 4 Application Development by Wei-Meng Lee.
2. Beginning Android by Mark Murphy.
3. Professional Android 2 Application Development by Reto Meier(Wrox)
4. Introducing Android Development with Ice Cream Sandwich by Shane Conder, Lauren Darcey.
5. Sams Teach Yourself Java in 24 Hours (Covering Java 7 and Android) By Rogers Cadenhead
6. Programming Android By ZigurdMednieks, Laird Dornin, G. Blake Meike, Masumi Nakamura
7. Amazing Android Apps For Dummies by Daniel A.Begun
8. http://developer.android.com/guide/index.html
9. http://www.codeproject.com/Articles/102065/Android-A-beginner-s-guide
10. http://mobile.dzone.com/articles/fundamentals-android-tutorial
11. http://mobile.tutsplus.com/tutorials/android/java-tutorial/
12. https://developers.google.com/places/documentation/
13. Callaghan, M.J.; Harkin, J.; El-Gueddari, M.; McGinnity, T.M.; Maguire, L.P., "Client-Server Architecture for Collaborative Remote Experimentation," Information Technology and Applications, 2005. ICITA 2005. Third International Conference on , vol.2, no., pp.125,129, 4-7 July 2005

14. McCann, J.A.; Jawaheer, G.; Linxue Sun, "Patia: adaptive distributed Webserver (A position paper)," Autonomous Decentralized Systems, 2003. ISADS 2003. The Sixth International Symposium on , vol., no., pp.303,310, 11-11 April 2003

15. Mo Guan; Minghai Gu, "Design and implementation of an embedded web server based on ARM," Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on , vol., no., pp.612,615, 16-18 July 2010

16. McCann, J.A.; Jawaheer, G.; Linxue Sun, "Patia: adaptive distributed Webserver (A position paper)," Autonomous Decentralized Systems, 2003. ISADS 2003. The Sixth International Symposium on , vol., no., pp.303,310, 11-11 April 2003

17. Johnson, M.E.; McGuire, D.; Willey, N.D., "The Evolution of the Peer-to-Peer File Sharing Industry and the Security Risks for Users," Hawaii International Conference on System Sciences, Proceedings of the 41st Annual , vol., no., pp.383,383, 7-10 Jan. 2008.