**Client-Server Communication**

The topic of client-server communication is an immense one, with substantial detail and subtopics. It consists of a model which forms an application that distributes its tasks and workloads across servers and clients alike. The relationship between a client, the receiver of a service, or the one who requests service, and a server, the provider of that resource or service, is such that a single server shares its services with multiple clients, but a client shares none of its resource and can only request cooperation from the server. There are numerous examples of applications which can exist on a client, but require communication to a server in order to function. The most famous is, of course, the internet, or World Wide Web, although a network printer is another notable example. Web services are systems that are identified by their URL, or location on the internet. The server is the entity which houses this URL and web service, and the client is the entity which approaches that URL and requests its contents and services.

Clients and servers are able to communicate with each other by sending streams of bytes over network connections. The server, or program, runs on a specific computer, generally a dedicated computer which has no other function but to house that program and service, and listens for clients to request a connection with it. This communication requires a protocol to be followed in order for either party to understand the data being sent to it. There are different protocols which must be followed depending on the type of service being requested or provided. These different protocols, namely the the Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and the Internet Message Access Protocol (IMAP), will be examined in more depth later. The connection, once established, is a pattern of message requests and responses between the parties. The client requests something from the server, and the server responds. This is an example of inter-process communication, wherein operating systems allow processes to share data, because it “involves data from both the client and server whereby each of them performs different functions” (Oluwatosin, p. 67). Another entity within this model of server and client is a database. Most of the time, databases are used in a 3-tier architecture wherein the client requests a response from the server, who, in turn, requests a response from the database. The database returns a response to the server, who, in turn, forwards that response to the client. In recent years, Cloud computing has gained an exponential amount of popularity. This type of computing exists as a way for many clients to receive shared processing resources and data on demand.

As described earlier, there are many different protocols that are enacted when a web service is provided or requested from either a server or client. Possibly the most prominent, or at the very least, the most recognized, protocol is the HyperText Transfer Protocol. It has been “the foundation for data communication for the World Wide Web … since 1990” (“HTTP Tutorial”, n.d.). This protocol is used in order to send data such as text, images, database results, etc., from client to server over the internet. There are two main ports which are used for receiving this data on a client-sider computer which are 80, for unencrypted data streams, and 443, for encrypted data. When this protocol is used, a client-side computer uses a web client to send a request to a specific server defined by a URL. The web server receives this request, and initiates a server-side script, which may access a database of information which is returned to the original requester. This information which is returned by the server-side computer is generally viewed as websites on a web browser. The information is all the data which makes up the various components of that website, including the text, images, or variable data.

The Simple Mail Transfer Protocol (SMTP) is the protocol which plays a primary role in the transferring of e-mail, however, this protocol is not generally the protocol which is used for client to server transferring. Instead, it is used for communication between the electronic mail servers themselves as well as other transfer agents. Client-side applications generally use Post Office Protocol 3.0 (POP3) or Internet Message Access Protocol (IMAP) for communication with mail servers. POP3 was an earlier protocol which IMAP has made POP3 more obsolete, although most web providers provide support for both. These different mail protocols all work over different ports, SMTP functioned over port 25, IMAP over 143, and POP3 over 110. When using IMAP, a client may read an e-mail directly off of the server rather than having to download and store it on their personal computer, unlike the antiquated POP which downloaded the data to the client-side computer and removed it from the server. This older system meant that once an e-mail had been downloaded, it could no longer be retrieved by any other client-side computer. With IMAP, a client may view their e-mail from multiple different sources without damaging the contents of that email. While IMAP is a newer protocol which is rife with advantages, many systems continue to use POP3 due to it’s simplicity and success rate.

The File Transfer Protocol (FTP) is a service which is used for the transferring of data from a server to the client’s computer. Rather than HTTP, which temporarily houses data on a client’s computer for perusal through a web browser, FTP is used to download content indefinitely. This way users are able to “transfer files (download) to their computer or to transfer (upload) their own files to the server (computer)” (“FTP and SFTP Protocols”, n.d.). Generally, an FTP session uses two streams of data across the connection, one for sending requests or commands between the server and the client, and another to send the specified data (i.e. a file). These FTP sessions may be either active or passive. In an active mode session, a client-side computer sends a request from a random port of its own to the server’s port 21 with a PORT command. For all FTP sessions, the server’s input port is always 21, but the client may use any number of ports. After this first contact is made with the server in an active session, the server connects back to the client-side computer through its own port 20. The connection between the client (random port) and server (port 21) is the stream for command or request data. The connection between the server (port 20) and the client (random port) is used to transfer files. Passive mode sessions are very similar. They begin, again, with the client-side computer connecting to the server’s port 21, except that this time they send a PASV command, rather than a PORT command. The server replies to the client with an indication of which random port it has opened for the transfer. The client then connects from another random port of it’s own to the port indicated by the server.

When discussing the communication between servers and clients, it is illogical to exclude the topic of security, for there is a clear opening for threats to both the server and the client when open-ended sessions are initiated. Clients are susceptible to viruses, worms, and Trojan horses. A virus is malicious software which infects other programs by infecting them. These files can be simple data files, but can also be as important as data within the boot sector of the hard drive. Trojan horses are programs which are allowed into a client-side computer by sending false information about its purpose to that computer. Unlike viruses, however, they do not generally infect other files, but are used to steal data or resources from the host computer for purposes such as botnet (automated spamming), illegal proxies, electronic money theft, or password or personal information theft. A worm replicates itself in order to spread itself eventually to other computers, but unlike a virus it does not attach itself to a program, it simply repeats itself. This generally results in high bandwidth consumption, rather than data corruption as with viruses.

Servers must be wary of illegal eavesdropping on data streams, denial of service (DoS) attacks, or data packet corruption. In eavesdropping, an individual or organization, attempts to capture packets transferred across connections between servers and clients. These data packets could contain any of the information that might be needed for a session between a client and server. For example, in a connection between an individual and their bank, some of the packets will contain username, password, account number, and other personal information. The entity doing the eavesdropping who collects these packets now has access to all that information. Denial of Service attacks are slightly less malicious than eavesdropping or data corruption, but still cause sufficient distress that it is necessary to protect against them. In DoS attacks, attackers attempt to overload the system so that servers are no longer able to provide their services to clients that need them. Packet corruption, or more correctly, packet modification, happens when an attacker has collected data from a packet in a connection, and altered it so that the information received by the server, or client, is incorrect. This can do considerable harm, for example, when online purchases are made, in which case the amount of money is essential and a modification of that amount could result in either a client spending more than they wish or a provider receiving less than they wish.

Most protocols for client-server communication have many protections in place to prevent malicious data from being sent or received, however a full understand how these safeguards work or are implemented would require a much more in-depth awareness of the entire client-server architecture than is possible within such a brief synopsis. It is important to know that these security risks exist, nonetheless, because even with countless security measures set up, there is always room for error.

The internet, and consequently communication between client and server, is an ever-changing matter, as is computer programming. The protocols and architecture involved in that communication as well as the security risks and safeguards in place are many varied, ever-changing, and eternal. Understanding even a portion of how these different threads weave themselves, or rather are weaved by the providers and requesters of web services, helps not only to more efficiently use web services, but more importantly for programming, help to gain insight into how web services should be written.

Works Cited

"Active V.s. Passive FTP Simplified - Understanding FTP Ports." *Active V.s. Passive FTP Simplified - Understanding FTP Ports*. N.p., n.d. Web. 19 Aug. 2016.

Chase, Jeff. "Sockets and Client/Server Communication." (n.d.): n. pag. Duke University. Web. 19 Aug. 2016.

"Client–server Model." *Wikipedia*. Wikimedia Foundation, n.d. Web. 19 Aug. 2016.

Dustdar, Schahram, and Wolfgang Schreiner. "A Survey on Web Services Composition." *IJWGS International Journal of Web and Grid Services* 1.1 (2005): 1. Web.

"Explanation of the FTP and SFTP Protocols." *Explanation of the FTP and SFTP Protocols*. N.p., n.d. Web. 19 Aug. 2016.

"HTTP Tutorial." *Www.tutorialspoint.com*. N.p., n.d. Web. 19 Aug. 2016.

"Internet Message Access Protocol." *Wikipedia*. Wikimedia Foundation, n.d. Web. 19 Aug. 2016.

"Inter-process Communication." *Wikipedia*. Wikimedia Foundation, n.d. Web. 19 Aug. 2016.

Oluwatosin, Haroon Shakirat. *Client Server Model*. Place of Publication Not Identified: Book On Demand, 2013. *IOSR Journal of Computer Engineering*. Web.