Implementation of Distributed Applications using Java

Obaid Yousuf and Jennifer McManis *

Abstract

Java's portability, architectural neutrality, and support of network connectivity make it ideal for implementation of applications on networks. Java applets in particular are designed to interact with Web browsers to provide a variety of services over the network. Unfortunately, Java applet security constraints restrict the user's ability to connect with and transfer data to and from remote sites and thus limit the user's ability to utilize distributed applications. In this paper we explore ways to get around this limitation. Specifically, we propose to use an intermediate server to connect the user of the applet to remote sites that they wish to access.

1 Introduction

The Java language emerged in 1995 and has been attracting considerable interest since that time. Java programs compile to an architecturally neutral byte code format. This combined with its portability and support of network connectivity makes Java ideal for network-based applications. A wide variety of references discussing all features of the Java language are already available [1], [2], [3], [4], [5].

Java programs may be run either as applications which are executed by a Java interpreter or applets which are executed within an applet viewer or web browser. Generally, applications are run when the code is well understood and trusted by the user. Java applets, on the other hand, are designed specifically to be embedded inside web pages, providing easy access to users. Because applets are meant to be run by users who have little or no idea of their functionality, stringent security measures are in place to ensure that the applet code run by the user is not able to damage the user's system. As with all security measures, they cannot absolutely guarantee the user's safety [6], [7], [8], but they go a long way towards doing so. These security measures restrict the ability of the applet to access user data and other system resources, and its ability to connect to remote sites. Unfortunately, this also limits the applet's usefulness in distributed application where it becomes necessary to interact with multiple remote locations.

This paper explores a way in which applets can be used to develop a distributed application while still obeying the security constraints placed on them. The proposed scheme uses the server from which the client initially downloads the applet as an intermediate link between the client and remote locations. Section 2 gives an overview of Java security features. Particular attention is paid to those features which will restrict a Java applet's ability to interact with remote sites. Section 3 discusses our general approach to implementing distributed applications using the server as an intermediate link between the client and remote sites. In section 4 our implementation of a small test application is discussed. The functionality of this application is limited, but it demonstrates that data can be moved between the client and remote data sites through the use of an intermediate server. Naturally, this does not implement a completely distributed application since all interactions must be run through the intermediate server. In section 5 possibilities for easing the load on this intermediate link are discussed.

2 Java Security Features

Java security features exist at three levels. First, Java has eliminated many language features that lead to potential security breaches. For instance pointers have been eliminated. Secondly, the compiled Java bytecode is checked to ensure that it does not contain forged pointers, use illegal data conversion, violate access restrictions, etc. This is accomplished by the class loader. The third Java security feature is a set of security restrictions which are specific to Java applets and are enforced by the Java enabled browser. It is these security restrictions

*Department of Computer Science and Engineering,Auburn University

that have the most significant impact on ones ability to implement distributed applications in Java. The most significant of these restrictions are as follows.

- Applets do not have access to local file systems.
- Applets are allowed to read and write only on the original host.
- Applets cannot invoke a program on the local system.
- Applets cannot access or load classes in any package other than the standard eight of the Java API.
- An applet cannot make a socket connection to any machine other than the one from which it was downloaded.
- Applets cannot create or manipulate any thread on the local system.

The restriction of access to the local file system and the inability to open socket connections to remote locations are particularly restrictive in terms of implementing a distributed application since the applet is limited to communication with a single machine, and not the multiple machines necessary for a truly distributed application.

3 Distributed Applications Using Java

Java's architectural neutrality makes it ideally suited to network applications. In particular, the Java applet is designed to interact with web browsers to provide straightforward client/server connections, freeing clients from the burden of having to install the software themselves and providing security measures to ensure that the software will not be able to damage the system it is running on. If the client only needs to interact with processes on the machine from which it downloaded the original applet, the security features are only minimally restrictive.

The next logical step is to think about what would be required to support remote distributed applications where the resources the client may access (and potentially modify) are located on a number of remote machines. An example of this would be a distributed database where the client wishes to access information which may be stored on any of a number of remote machines. Ideally, if there were no security restrictions, supporting such an application would requires that connections be made between the client and multiple remote sites as in figure 1. Unfortunately, Java applet security constraints prevent connections being implemented between a downloaded applet and remote sites.

In order to get around this problem, it is necessary to use the server from which the applet was originally downloaded as an intermediate link between the client and the remotes sites the client might wish to access. Data transfer between client and server and between server and remote locations may be accomplished using Java applications running on the server. See figure 2.

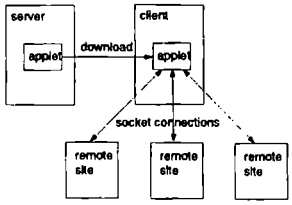

Figure 1: Ideal (No Security) Implementation of Distributed Application

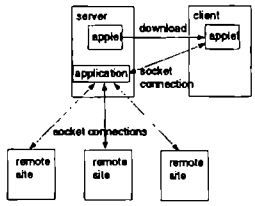

Figure 2: Implementing a Virtual Connection Through an Intermediate Server

In this solution, the client downloads an applet from the intermediate server for the purposes of data transfer. The only direct connection the client makes is to this intermediate server. Through this connection, the client may request the server retrieve data from a remote site, access this data from a local repository at the server site, and request that the server transfer modified data back to the remote locations. The client stores all retreived data in the browser buffer, and thus the applet never needs to access the local file system.

There are basically two host server processes that must be run. The first provides a connection through which data may be retrieved from a remote site upon client request. The second provides a connection through which data may be retrieved from the client to be forwarded to the appropriate remote site. These processes are implemented as Java applications as opposed to applets and thus are not subject to the security constraints placed on applets. Thus the server can make socket connections to multiple remote sites. It is reasonable to use applications, since as the service provider, the server is confident that its software is reliable. Figures 3 and 4 illustrate the procedures followed when moving data between the client and the remote sites.

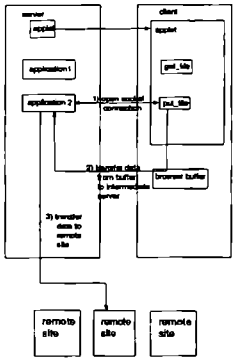

Figure 3: Sending Data to Remote Sites

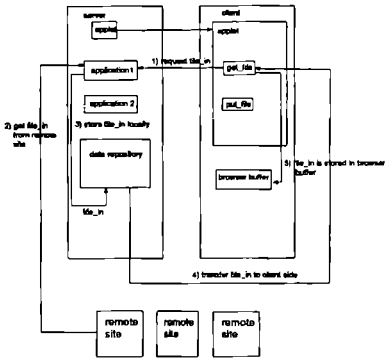

Figure 4: Retrieving Data from Remote Sites

4 Implementation

A small *proof of concept* has been implemented in order to test the approach described in section 3. The implementation tests the ability of the client to transfer data to and from remote sites through an intermediate server. The application used is a 'phone book' which is contained in three files at remote locations which the client wishes to access. When the client loads the applet, the applet establishes a virtual connection between the client and remote data sites. The files are automatically retrieved from the remote location for the client's use. This simplifies the functionality of the remote data sites since they no longer have to determine what are the relevant pieces of data to send. The client may create and send a file containing selected names, phone numbers, and addresses to the of the remote sites. The code for this implementation is too large for this paper, but may be found in [9].

5 Conclusion

There are two issues associated with distributed computing – security and functionality. Evidently, both of these cannot be maximized at the same time. Taking into consideration the relatively insecure nature of the Internet, more emphasis has been placed on security issues in Java. Applet security in Java limits its flexibility. In this paper we have presented a solution to overcome the limitations of Java applets.

The idea behind this implementation is to establish a virtual link between a remote site and the applet running on the client side through the local host. This indirect connection needs to be established because applets cannot directly connect to any machine other than the one from which the applet was originally downloaded. With the virtually link, the client connects with the original server and then relies on this server to make connections with the remotes sites. Thus no direct links need to be established between client and remote site and Java security constraints are not violated.

Naturally, this approach leads to a potential bottleneck at the server. Future work would include looking at possible solutions to this problem. One likely solution is to have the initial server act as a router guiding the client to multiple sites from which they could download the applet. The server would then be responsible for determining the load on possible alternate server sites and attempting to balance that load. The tradeoff of added complexity of implementation versus possibly better distribution of workload remains to be explored.

References

[1] J. Gosling and H. McGilton. The java language environment.

[2] B. Boone. *Java Essentials for C and C++ Programmers.* Addison-Wesley, 1996.

[3] D. Flannagan. *Java in a Nutshell.* O'Reilly and Associates, inc., 1996.

[4] M. Danconta. *Java for C/C++ Programmers.* Wiley Computer Publishing, 1996.

[5] D. Friedel and A. Potts. *Java Programming Language Handbook.* Coriolis Group, 1996.

[6] D. Dean, E. Felton, and D. Wallach. Java security: From hotjava to netscape and beyond. *IEEE Symposium on Security and Privacy*, 1996.

[7] D. Hopwood. Java security bug. *RISKS Forum*, March 1996.

[8] M. Mueller. Regarding java security. *RISKS forum*, November 1995.

[9] O. Yousuf. A distributed client server interactive implementation using java applets. Master's thesis, Auburn University, 1996.