812 Project Proposal - "Socket Shifting"

Grant Birchmeier Dehua Hang Jason Stredwick Sameer Arora

1 Project Summary

For our project, we propose to implement a procedure we call "socket shifting". A client application will communicate with a server application in traditional fashion, but we want to implement functionality such that at any time the server process may decide to shift its role to a different server application (likely the same application running on a different server, or possibly a different instance of a kernel application on the same server). This shift will be completely invisible to the client application, in that the client programmer will not have to write special code to renegotiate a connection with each server application.

The core of this implementation is the "shifting" of sockets between services. We intend to alter the Linux kernel at the TCP layer to manage the socket transfer on both the client and the server. This will be implemented by altering the TCP system calls and adding some kernel-level functionality that will deal strictly with socket transfer messages. This low-level implementation will be responsible for setting up and modifying the connected sockets so the connection does not appear altered to the application level client. Generally, client applications will appear as normal networking applications, but server applications will require system calls to the networking layer in order to initiate the socket shift.

At present, we see three main steps of this project:

1. Designing and implementing the protocol that the kernel-level implementation will use to communicate between socket-shift-enabled computers to negotiate a socket shift
2. Determining and implementing the system calls that need to be implemented in the kernel-level implementation for server applications to call to initiate a socket shift
3. Implementation of the client/server program that will be used to test and demonstrate the project.

There are alternate solutions to this problem, but we chose this method because it is interesting and will provide experience in kernel programming. There is no reason this functionality could not be implemented in middleware or a user library and API (and some may argue that it should be).

2 Related Work

The primary inspiration for this project was the Cyclone project, detailed in [**?**] and [**?**]. Cyclone is a cluster-based web server system that uses "socket cloning" to shift the page-serving duty from the web server node that initially received it to a different node. This is useful in an instance where the second node has the document cached in memory but the first does not. This technique is useful because there is no "middleman"

from the second node to the client, though communications *from* the client still need to go through the first node, therefore eliminating a source of "bottleneck" in the cluster.

The socket cloning technique used in Cyclone requires the communication to be relayed through the initial server to the cloned socket in the second server. The cloned socket then sends its packets directly to the client. The client is completely ignorant of this process though a special socket cloning layer manages from whom it will receive packets. Socket shifting is different because it will wholly move the connection from the first server to the second one. It will lose the overhead of the "middleman" server, but it will require more work in the layer that shifts the sockets. We hope to use the socket cloning code of Cyclone as a starting point of our project.

Another somewhat related technique is that of network address translation or IP Masquerading [**?**] [**?**]. Essentially, the client submits a request to a service, and regardless of how the service sends the data back to the client, the client has performed the proper transaction. The two techniques are similar as two whole techniques rather than there being specific similarities among the processes that comprise them. IP Masquerading is similar in that it maps socket connections in the opposite direction. Instead of mapping the incoming packets to a specific address and port, it maps the address and port of the incoming packets to where the request was originally sent. We want to borrow the basic idea behind the implementation details of IP Masquerading, in that the whole service is invisible to the client. Some go-between entity handles the implementation details.

References

[1] Yiu-Fai Sit, Cho-Li Wang, Francis Lau, "Cyclone: A High-Performance Cluster-Based Web Server with Socket Cloning," submitted to Cluster Computing: The Journal of Networks, Software Tools and Application, Special Issue on Cluster Computing in the Internet, (June 6, 2002)
http://www.csis.hku.hk/~clwang/projects/cyclone.htm

[2] Yiu-Fai Sit, Cho-Li Wang, Francis Lau, "Socket Cloning for Cluster-Based Web Server," IEEE Fourth International Conference on Cluster Computing (CLUSTER 2002), September 2002.
http://www.csis.hku.hk/~clwang/papers/cluster2002-FNL-socket-cloning.pdf

[3] David A. Ranch, "Linux IP Masquerade HOWTO," January 2003
http://www.ecst.csuchico.edu/~dranch/LINUX/ipmasq/m-html/ipmasq-HOWTO-m.html

[4] Linux IP Masquerade Resource
http://www.e-infomax.com/ipmasq/