# 812 Project Proposal Addendum- "Socket Shifting"

Grant Birchmeier Dehua Hang Jason Stredwick Sameer Arora

## 1 Project Addendum

In order to get more perspective on our proposed project we sought references to material that dealt with similar concepts. These concepts vary from implementation details that would simplify our workload to those that would help us further refine what we want to accomplish and suggest avenues to take toward that end. The following are a series of these resource that each have various degrees of relatedness to our project. These resources are Mockets, MigS, migrating sockets for internet quality of service, and MIGSOCK.

The first reference, Mocket [**?**], is an application-level TCP-socket encapsulation providing an API and infrastructure for applications to open network connections and move between hosts without the applications (like mobile agents) themselves having to interrupt, disconnect and reconnect any connections. This project was implemented in Java as a user level service. The Mocket infrastructure provides a wrapper around the traditional TCP sockets and streams, along with two other processes which together relieve an application from managing the connection during migration.

The Mocket team decided that two sockets were necessary for their implementation. One socket is used for end-point communication and one is used for Mocket control messages. The Mocket control messages suspend, resume and renegotiate the connections between end-point applications. This implementation had unacceptable overhead, therefore, they created a standalone process that would focus the Mocket control system for all applications on a system into a single point. Thus all applications that seek to use Mockets must go through this standalone process, and communication with remote machine can only occur if they too have the Mocket process. While this approach is similar to ours, as it relieves client programmer to renegotiate a connection with after moving, it's different as it is implemented at user level while we aim to modify the kernel. Also, we do not plan to exchange our control messages out-of-band, instead we will modify TCP which will generate and intercept our control messages.

Yau and Lam [**?**] have implemented a version of Migrating Sockets as an implementation framework for user-level protocols that must provide quality of service guarantees. They implemented this as a library with backward compatibility with Berkeley Sockets, thus socket migration would be transparent to the programmer. The internal socket migration is handled using a client/server architecture. All user applications are considered clients and all connections between these clients go through a connection management server. This is different than our proposed project which explicitly avoids the use of a server as a middle man for socket migration. Other differences involve the fact that migration only occurs between a client and the server. Their server will push a socket connection to a client when a connection is one to one, and pull the socket back to the server when a connection is a many to one or many to many. They have a similar idea involving the passing of socket state information as their migration mechanism. While some concepts are shared between our projects, this project is very unrelated implementation-wise.

MigS [**?**] is a migrating sockets implementation that aims to provide a kernel-level application-independent socket migration facility to use in building scalable or fault-tolerant systems. Its creators have implemented a session layer End-point Control Protocol to handle operations on communication end-points, and an interface between the session layer and transport layer. The existing interface of the 2.2.16 Linux kernel to the transport layer was deemed adequate. Many aspects of this project are similar to our proposed project but the specific implementation and demonstration will be different.

MIGSOCK [**?**] is a project that closely parallels our own proposed project. It was a master's thesis done to tackle the issue of kernel support for socket migration used in process migration. The support was in the form of a kernel module that completely reimplements TCP in Linux. Using a kernel module made migration transparent to a socket programmer. It did this by using special messages in the kernel to halt and transfer and continue a socket. The user of the socket would never see these messages because they were trapped by the kernel and were never allowed up into user space.

Fortunately and unfortunately, the first step of MIGSOCKs was a socket hand-off scenario rather than full-blown process migration. This project is almost identical to our proposed project. One similarity is with respect to the kernel's involvement in socket migration. MIGSOCK put this modification into a module and reimplemented TCP. We were only going to modify the Linux source directly by adding in our functionality rather than reimplement TCP. Another similarity is how the socket migration will be demonstrated. The main difference is the service we were going to provide. We both plan on having one program communicate with another who will then pass its connection to a duplicate program on another computer without renegotiation. The last similarity is that we also plan to use kernel level messages to accomplish our task. We have asked for access to the source, but we have not heard back from them at this time.

Each of these references has given us some benefit toward accomplishing our goals, and further avenues of investigation. We will be keeping them in mind as we progress.

# References


[1] Timothy S. Mitrovich, Kenneth M. Ford, and Niranjan Suri, "Transparent Redirection of Network Sockets"
http://nomads.coginst.uwf.edu/mockets.pdf

[2] David K.Y. Yau and Simon S. Lam, "Migrating Sockets for Networking with Quality of Service Guarantees,"
http://www.nmsl.cs.ucsb.edu/~ksarac/icnp/1997/papers/1997-8.pdf

[3] Michael Haungs, Raju Pandey, Earl Barr, and J. Fritz Barnes "Migrating Sockets: Bridging the OS Primitive/Internet Application Gap,"
http://pdclab.cs.ucdavis.edu/~pandey/Teaching/289C/Papers/migs.pdf

[4] MIGSOCK
http://www-2.cs.cmu.edu/~softagents/migsock.html