Building GeoPlex Gate Proxies on GUNet

Yuriy Shapirshteyn* Vera Zaychik† William C. Regli‡

Geometric and Intelligent Computing Laboratory
Department of Mathematics and Computer Science
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104
http://gicl.mcs.drexel.edu

August 24, 1999

Overview

In the GeoPlex architecture, Peer-based and Peerlet-based clients and services attach to the main network Cloud via Gate nodes that sit on the edges of the Cloud. The Cloud is a hardware and software infrastructure built on a high speed network, providing authentication, usage recording and encryption. When designing GeoPlex, it was assumed that all of the communication will be done through the Cloud and that all peers were connected directly to a Gate—a model common with Internet Service Providers (ISPs).

The current GeoPlex is intended for use as general IP middleware. To create facilities for security and data tracking in this environment, we are using GeoPlex Gate Proxies to recreate this situation by proxying all services through gate nodes (e.g., sj-gate.gunet.net). If the user has Internet access other than through GeoPlex there might be a closer route from the client to the server that does not flow through GeoPlex. In this case communications may be done outside of the cloud through this route, as shown in Figure 1.

As a part of the cloud implementation a set of internal services (set up as Proxies) may be needed in order to run things efficiently or get access to the internals of the cloud. We present a set of demo proxies that we wrote in order to understand the difficulties and provide an easier learning experience for others.

Implementation Details

We have built two proxies to support demonstration services that we were working on: ASRAUR Redirection Proxy and Echo Proxy. A diagram showing these proxies is given in Figure 2. The ASRAUR Proxy is shown in Figure 2 (a). It is a TCP based proxy which listens for connections and retrieves data related to the request. The request is “1” for active users and “2” for active services. It accesses GeoPlex Monitoring Management System web server located on the Core in this case core.gunet.net on port 2221 for users and 2223 for services. It then uses URL and URLConnection classes to retrieve the data and sends it back to the requester.

The Echo proxy, as shown in Figure 2 (b), reads in blocks 2 Kilobytes in size which consist of 0’s. The last block that is read starts with a 1. Once that block is read, reading is terminated and an acknowledgement in a form of a string

*Email: uyshapir@mcs.drexel.edu; URL: http://www.mcs.drexel.edu/~uyshapir.
†Email: zaychik@drexel.edu.
‡Email: regli@drexel.edu; URL: http://www.mcs.drexel.edu/~regli.
“ACK” is sent back. The reading is done using InputStream/OutputStream that are provided by the Java TCP socket class.

Echo proxy was written in order to do bandwidth estimation between the gate and the Peer.

**Integrating with GeoPlex**

To make these proxies GeoPlex-enabled, couple of steps should be taken, such as opening up a firewall to allow access to the proxy. First a user should be created who will own this proxy. Once this is done, a new service corresponding to the proxy should be registered and announced on the port and machine that proxy is running on. Note that all logins should be done using “geoplex” account.

1. Login to sj-gate.gunet.net
2. Login to reg.gunet.net machine
3. Run register service and answer all of the questions. Write down everything entered (you will need it later).
4. Login to core.gunet.net
5. Run manageAur -e serviceHandle serviceID userID serviceIP netmask homeProxy port protocol proxyType equivalence accessPolicy encryptionFlags. Example:

   
   ```
   manageAur -e EchoProxy 9978 1012 135.197.23.131 255.255.255.255 111.111.111.111 7171 6 TCP 0 1 0
   ```

   In the preceding example you have to specify all the required information about the new proxy: EchoProxy is a proxy's serviceHandle or shortcut; 9978 is serviceID that you select for your service; 1012 is the your user number, where you are service owner (which can be found by running the account manager from the peer console); 135.197.23.131 is the ip of sj-gate.gunet.net or ip of any other gate where you install the proxy; 255.255.255.255 is the netmask; 111.111.111.111 is the homeProxy; 7171 is the port your proxy is running on; 6 is the GeoPlex protocol identifier (6 for TCP, 17 for UDP); TCP is proxy type; 0 is the GeoPlex...
equivalence identifier; 1 for the GeoPlex accessPolicy (1 - full public, 0 - cloud public, -1 - private); 0 sets the GeoPlex encryptionFlags (0 means do not encrypt, 1 means encrypt).

After this is done a proxy can be compiled and started.

**Future Ideas**

1. **UDP Proxy:** UDP is a type of communication protocol where there is no guarantee that the data packets will arrive and if the will that they will arrive in a correct order. This is useful when user is dealing with video and/or audio. At this point there is no working UDP proxy on the gunet.net or any other cloud, and therefore there is no possible way of sharing video or audio.

2. **RMI Proxy:** RMI (Remote Method Invocation) is a system for accessing objects located across the network as if the were on the same machine using client-server architecture. This system works by hiding all of the low level communication from the user. RMI manages ports and sockets in run time which creates a problem due to the callbacks initiated by the client. In this case client should be viewed as a service and has to be registered and announced with the cloud at run time. Due to the complications there is no RMI proxy in place.

3. **Dynamic Redirection Proxy:** A Dynamic Redirection Proxy needed if one wants to explicitly route the traffic through the cloud. This is done by specifying the final destination and the proxy will open a channel to it. At this point in time data can be routed through the cloud to the desired destination. Once this proxy is in place, it
will be trivial to enforce all of the supported traffic to go through the cloud by adding a simple IP to the SDK. Keeping this in mind it will be possible for the client software to access the server software without actually knowing the final IP and port number beforehand.