GUNet Applications: Active User and Active Service Registry Browsers

Vera Zaychik* Yuriy Shapirshteyn † William C. Regli‡

Geometric and Intelligent Computing Laboratory
Department of Mathematics and Computer Science
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104

August 19, 1999

Application Overview

This document describes the browsers for the Active Service Registry (ASR) and Active User Registry (AUR) which were created in absence of the corresponding GeoPlex API. To understand this application, we first need to provide some background information.

Background

The GeoPlex architecture includes entities called Peers which are machines running geoplex software that are outside the cloud. Peers can be servers running services and clients. Whenever a client connects to the cloud (by means of authentication or anonymously) through the gate, she/he/it becomes an Active User. Each gate keeps track of its active users and passes this information along to the core. Core has a dynamic record of each active user on the cloud which is stored in the AUR. In the same fashion, cloud keeps track of all the active services in the ASR. For a service to be active, it needs to be announced by its administrator on a particular machine. Currently there are no API that would provide this information back to peers.

Implementation Details

The ASR/AUR Browser consists of a proxy running on a gate and peer client software. A client is a peerlet. It is assumed that the user is authenticated to the cloud through the Peer Console prior to starting the GetASRPeerlet/GetAURPeerlet. Both peerlets work in a similar fashion so only one will be described here.

When the user starts the GetASRPeerlet with two command line arguments for the proxy ip and port, the connection is established to the running proxy inside the cloud. The proxy is itself a service. The client indicates the request for ASR information by ’1’ (’2’ for AUR). Proxy then issues a command to the GMMS system running on the core which can accept HTTP request. Since ’1’ and ’2’ are the only 2 commands ASRAUR proxy can accept,
security of the cloud is not compromised. The proxy then redirects GMMS response to the client peerlet. The peerlet parses the obtained information and displays it in the table. A user has an option of refreshing the data (it is not done automatically) and requesting more detailed view.

Future Ideas

1. A Start button is provided for the GetASRPeerlet so that in future client software for the services could be started from within the peerlet. Currently no mechanism is available for the cloud to pass the client application for a particular service to the peer. Although the service can provide a download URL, that information is actually not available to anybody but the cloud administrator and the administrator for that service which renders the existence of such information virtually useless.

2. Selecting a User from the AUR Browser brings up information about the user. Presently, all user information is private.

3. Selecting a Service from the ASR Browser brings up the service client and attaches to the service. I.e. run the service from the browser. This could also be tied in with tools to monitor and assess available network bandwidth—determining which services are capable of being run under the current network conditions.