Load Balancing ArcIMS with an Apache/Tomcat Reverse Proxy

by

Victor R. Conocchioli, M.S.,

State of Minnesota

Department of Transportation

GIS Administrator

Load Balancing ArcIMS with an Apache/Tomcat Reverse Proxy

Victor Conocchioli

Abstract

Load balancing hardware with ArcIMS delivers high performance map services. This paper discusses the hardware and software requirements to integrate ArcIMS with Load Balancing hardware, Apache, Tomcat, and Firewall access to ArcSDE. The system architecture provides good throughput as well as security for public ArcIMS Web sites. Apache Reverse Proxy can serve Tomcat and ArcIMS content on the same or other networked server while translating URLs to control access to network infrastructure. RedHat Linux was chosen for the operating environment. ArcIMS deployment and operating procedures were developed for integrating MS Windows and Linux system users. The Minnesota Department of Transportation created an example of this architecture to publish a large volume of spatial base map data.

Table of Contents

1. Executive Summary

The Minnesota Department of Transportation (Mn/DOT) provides access to a large spatial data set through interactive Web page programs. These data represent a comprehensive set of digital maps that support transportation planning and design. ESRI's ArcIMS system enables the interactive retrieval of map data from a Web browser and through the tools in ArcGIS clients. This paper discusses an infrastructure built around ArcIMS to deliver mapping services to the public Internet as well as internal customers.

We chose components to build our infrastructure that would maintain security as well as provide ease of access to the transportation data. The strategy to install components that are designed to form connections with ArcIMS gives our infrastructure flexibility to meet changing enterprise needs. Components installed from the Cisco load balancer, Apache Reverse Proxy, Tomcat, Red Hat Linux and in-house developer support software interact with ArcIMS to complete a system that accommodates a variety of transportation Web applications.

The typical ArcIMS installation serves a small number of Web developers when built in a centralized configuration. However, Mn/DOT supports a large enterprise user and Web developer base. Our infrastructure reconfigures ArcIMS into a multi-user Web content development system. We added user authentication connections to Microsoft Active Directory through the Vintela Authentication Services. Single sign-on authentication accommodates the growth of ArcIMS development. Active Directory authentication allows user groups based on Web content topic to develop ArcIMS map services in a centrally managed environment.

A centralized ArcIMS installation standardizes development processes and system recovery. Web site developers and system support teams establish naming conventions which also reflect in server storage structures. System imaging and backup occur at one location to minimize overhead in recovery or image restoration tasks. User authorization separates ArcIMS services into groups that develop in their respective content areas. Our ArcIMS infrastructure will readily adapt as Web delivery and deployment processes advance with new Web serving technologies.

2. Introduction

The Minnesota Department of Transportation provides interactive Web applications to view geographic data for public Internet users. ArcIMS delivers the interactive applications through an Apache/Tomcat infrastructure running on the Red Hat Linux operating system. A large volume of data requests by customers necessitates high availability and redundancy in designing the system. This paper discusses the components of the ArcIMS system and how they work together to achieve a secure, reliable and high performance infrastructure.

3. Infrastructure Design

ESRI supports several methods for configuring ArcIMS on Web servers. More common designs use Tomcat Java server in combination with the Apache http (hyper text transport protocol) Web server. The Apache serves static Web content from Tomcat. ESRI officially supports the loadable module (mod_jk) connector for use with ArcIMS. An alternative to the mod_jk connector is the Reverse Proxy method.[1] We chose the Apache Reverse Proxy method because of ease in maintaining the configuration with Tomcat and implementing firewall changes when required. (Illustration 1 shows the infrastructure components and networking in our system.)

The Mn/DOT ArcIMS computer system infrastructure requires a high performance and high availability map delivery system to our Internet customers. We designed the system around a Cisco Content Service Switch (load balancer) to serve identical ArcIMS servers. Each of these servers connect to a back-end ArcSDE/Oracle data base server located within our internal network. The access to the spatial data is read-only and the back-end server operates with a Storage Area Network for redundancy and reliability. Mn/DOT built these servers with ArcIMS to design a base map template that several offices use to launch their Web sites.

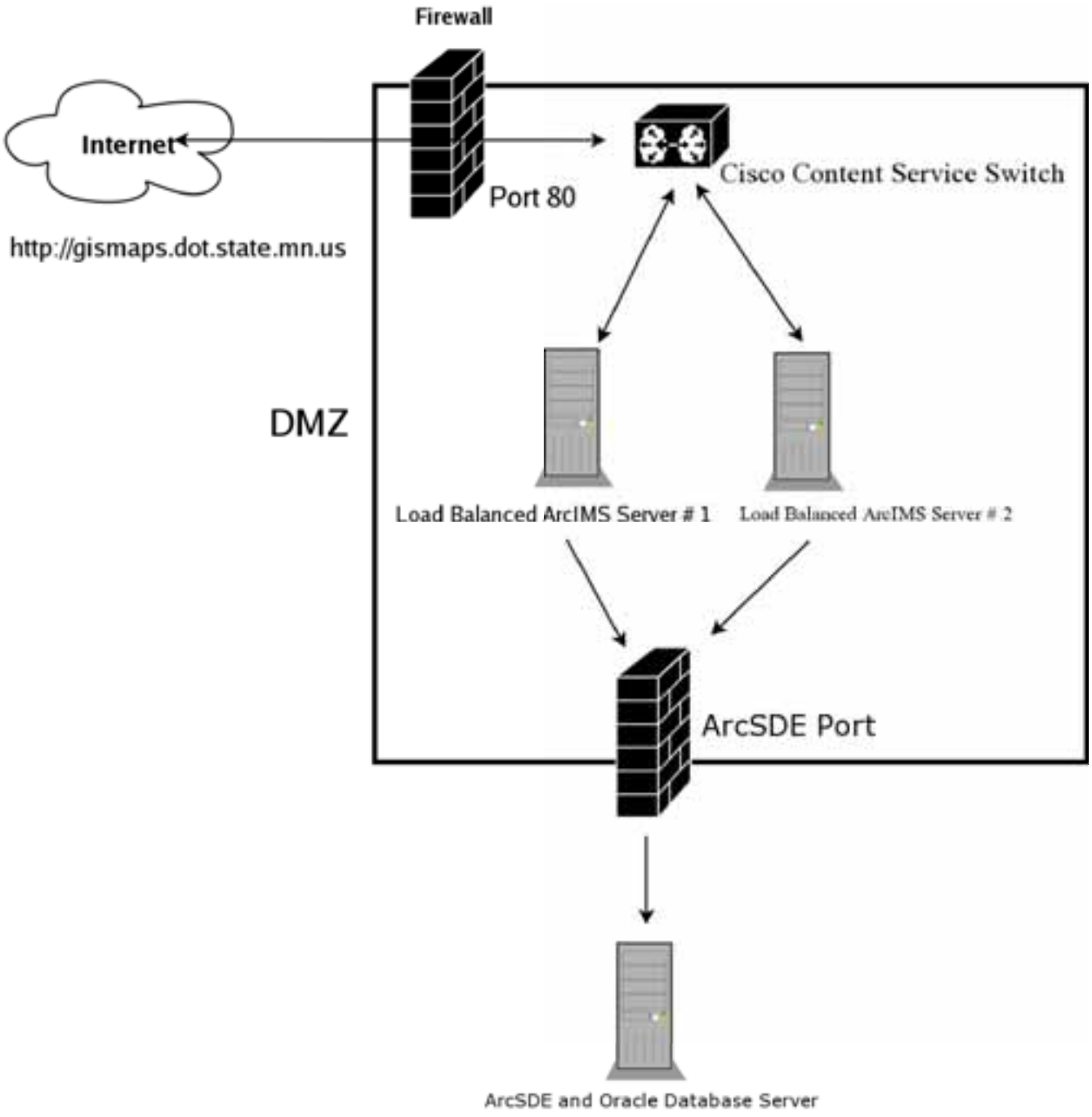

Illustration 1: ArcIMS Infrastructure

3.1 Load Balancing

A set of load balanced ArcIMS servers forms an ideal service administration environment. One server can be taken down for maintenance or reconfiguration while the ArcIMS services remain available. The load balancer sends a keep-alive signal to each Apache server to determine if either one is up and running. A load balancer can be configured to assign sessions based on round-robin or on numbers of concurrent sessions. In our configuration, we send client browsers connections to the server with fewer sessions.

The load balancer infrastructure requires that all ArcIMS components stay within the same computer. The reason for this constraint is that ArcIMS processes requests from and delivers output to the Apache static Web content locations. These locations are known as website and output in the default installation. When a browser initiates the connection through the load balancer, the browser must remain with the computer where it started the ArcIMS request. The load balancer must be configured to use a "sticky session" for about 30 minutes before a timeout. If we designed the infrastructure by separating Apache, Tomcat and ArcIMS on different computers, we would increase the complexity in the load balancer farm.

3.2 Apache/Tomcat Reverse Proxy

ArcIMS requres a front-end Java application server to deliver spatial content to the Web. We use the Tomcat open source Web server. The Tomcat server runs as a non-privileged account in Linux. Unix standards enforce a rule that non-privileged accounts must use TCP/IP port numbers above 1000. Tomcat serves its content on port 8080 by default. Most Web users do not prefer or may not be allowed to access Web servers that serve Web content outside of the standard port 80. There are three techniques to allow Tomcat to serve content through port 80 in the Linux environment. They are listed in order of preferred implementation.

1) Configure Apache with Reverse Proxy directives

   We found this to be the easiest method to create the Tomcat front-end while serving the ArcIMS content from the website and output directories.[1] There may be one caveat in this method, see section 4.2 below.

2) Configure Apache with the mod_jk connector

. This will require significant overhead with maintaining files on both the Apache and Tomcat configurations. The mod_jk connector is no longer supported by the Tomcat developers.

3) Reconfigure the Linux kernel at the networking level to redirect port 8080 to 80.

   This is difficult and prevents Apache from running on its normal port 80.

The Reverse Proxy method works well for ArcIMS and allows flexibility for creating secure Web services. The Proxy rewrites URLs seen by the client Web browser thus hiding access to a Web service or other back-end computer. Apache directives for an ArcIMS Reverse Proxy are given as examples at the end of this document. We set up our infrastructure to deliver content to the World Wide Web. This exposes a computer to many potential security concerns. The Apache Reverse Proxy provides a good first step in implementing adequate security on the Web. We added an additional module to the Apache Reverse Proxy to complete a secure infrastructure. The Apache Mod_security option will be discussed later in this document.

4. Software Configuration

The greatest challenge in setting up our infrastructure was coordinating the operation of all infrastructure components. Each infrastructure piece has particular settings that needed to be researched and tested throughout the whole ArcIMS installation. We started with configuring ArcIMS within the Red Hat Linux operating system including critical networking and system management pieces.

4.1 ArcIMS and Red Hat Linux

The ArcIMS installation CD for Red Hat Linux follows a simplified wizard to create an initial set up. We build our Red Hat servers without any graphical displays so installation occurs remotely through a Linux workstation or a Windows machine using Hummingbird Exceed. ESRI recommends that the ArcIMS account execute in the C-Shell. We use the tcsh, which is a Cornell University variant of the C-Shell. This shell allows for command line completion, retrieval and editing. Red Hat provides this package in the default system.

An advantage of running ArcIMS in the Red Hat Linux environment allows the strict separation of the operating system and the application administration functions. ESRI provides shell scripts to automatically start ArcIMS from a system reboot. However, these shell programs do not conform to what is known as Red Hat System V init script standards. We have adapted the ESRI start up shell scripts into a Red Hat init script template. The system administrator simply types: service aimsd start or service aimsd stop to recycle ArcIMS. We also use the Unix sudo mechanism of Linux to give ArcIMS application support administrators the rights to recycle the ArcIMS system. An example of our System V script can be found at http://gismaps.dot.state.mn.us/ArcIMS/supplement/aimsd.sh or at the end of this document.

When the infrastructure team completes all the system planning, configuration and tweaking, a backup/disaster recovery procedure known as imaging must be performed on the system. Imaging provides two benefits, one to create a quick recovery of the entire system and second, to use as a baseline in creating identical systems. Our ArcIMS infrastructure consists of a development, a test/staging and two production Dell 2850 computers. Imaging ensures configuration consistency throughout each machine. All of the ArcIMS configuration files use the name of localhost for networking to keep the set up of ArcIMS generic. This technique allows a fully configured ArcIMS installation to be cloned into another computer. The most common programs for cloning Linux are Ghost and the open source program called Partimage.

4.2 Apache/Tomcat communication with ArcIMS

The Apache Reverse Proxy creates an easily managed and good performing front-end to a Tomcat Java Web server. Unlike the mod_jk connector, the Reverse Proxy requires only one set of configuration directives to maintain in central Apache httpd.conf file or the conf.d directory. The proxied Tomcat server can be on the same machine or on another networked computer. (Illustration 2 shows the logical connections between all major software components in the infrastructure.)

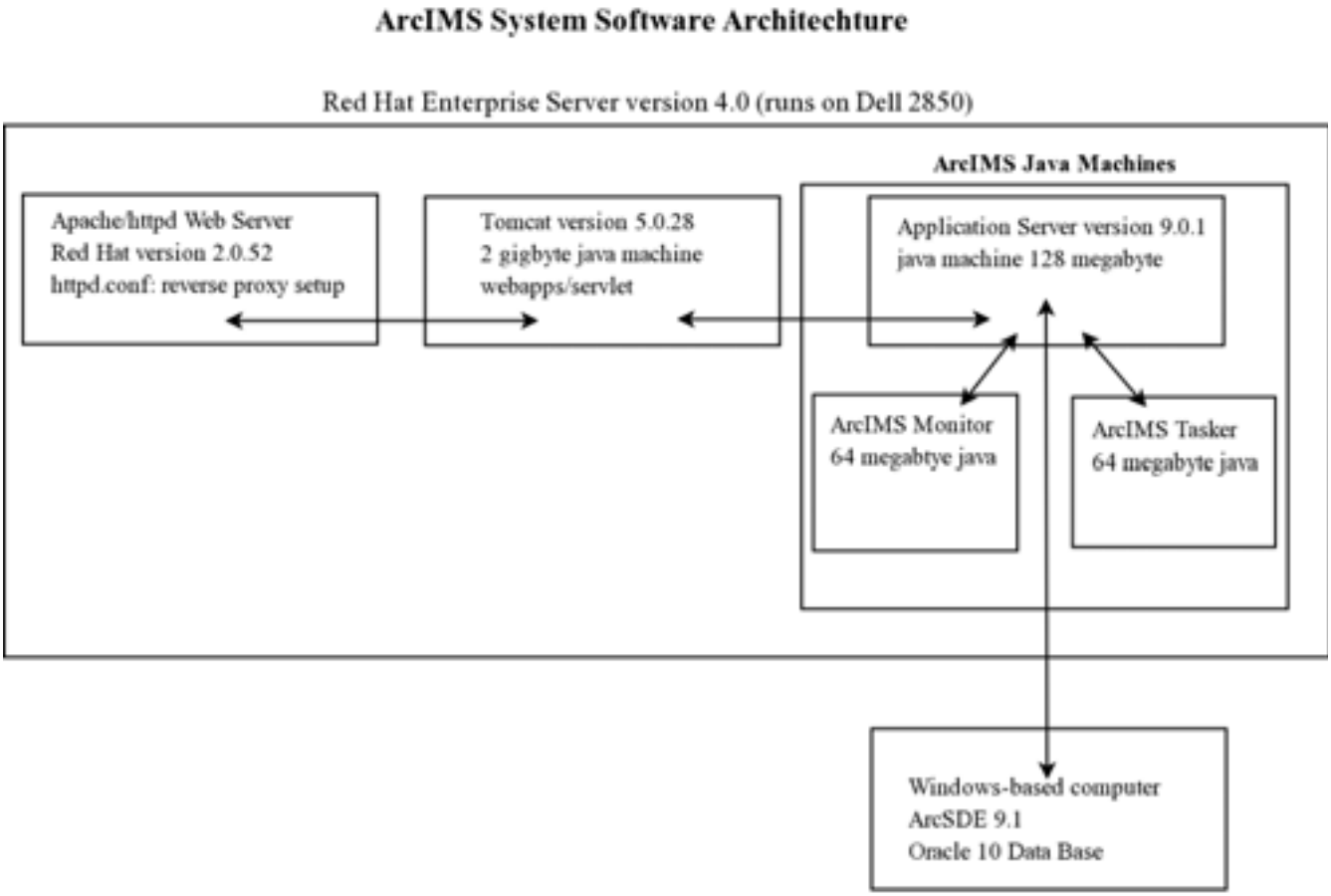

Illustration 2: Software Components

The Proxy rewrite engine can create very sophisticated and complex rules using regular expressions. We implemented the Reverse Proxy in a very simplified configuration. A small section of our Apache configuration file included in the Appendix gives the necessary directives to establish the Proxy between Apache, Tomcat and ArcIMS.

4.3 Apache Mod_security

The Mod_security project complements Apache with a mechanism to scan incoming http GET or POST requests. Most concerning are POST requests, which put data from a client browser onto the ArcIMS server. A POST request may contain scripting programs that attempt to compromise server security. Each incoming URL from a client Web browser proceeds through a set of rules to check for any malicious attempt to break ArcIMS. Basic mod_security installation begins with a minimal set of rules. The minimal set is adequate for most ArcIMS implementations. However, we found that when ArcGIS (workstation or server) acts as a client to a ArcIMS, one of the rules requires modification to allow ArcGIS to POST URL data to the Apache server. See the Appendix of this document for the modified rule. Mod_security can be obtained from http://www.modsecurity.org

5. Web Content Deployment

A reliable and well maintained ArcIMS Web service depends on good organization of the contributing Web content. The website directories, AXL files, static Web content and developer access should be standardized by system administration components of file system structure, development interfaces and user access procedures. This section gives examples of ArcIMS file structures and discusses strategies for developer access to ArcIMS.

5.1 Application Administration Process

The Linux operating system design allows for a clear separation between system level and application administration processes. ArcIMS programs, Web content, and pre-deployments can operate in a non-privileged environment. This prevents any unintentional effects on system performance, infrastructure stability and Web service design/testing. We separate the production system from the developer environments. (Illustration 3 shows the relationships between our systems.)

Our developer desktops typically run in the Microsoft Windows system software. We load two easily installed programs for access to the ArcIMS computers. Hummingbird Exceed is a licensed X-terminal emulator that displays Linux graphically based programs such as ArcIMS Administrator or Developer on the Windows desktop. SSH client software allows authenticated logins to the ArcIMS servers and file transfer from the developer workstation. Linux desktops do not need either of these programs since the software comes with nearly all installations.

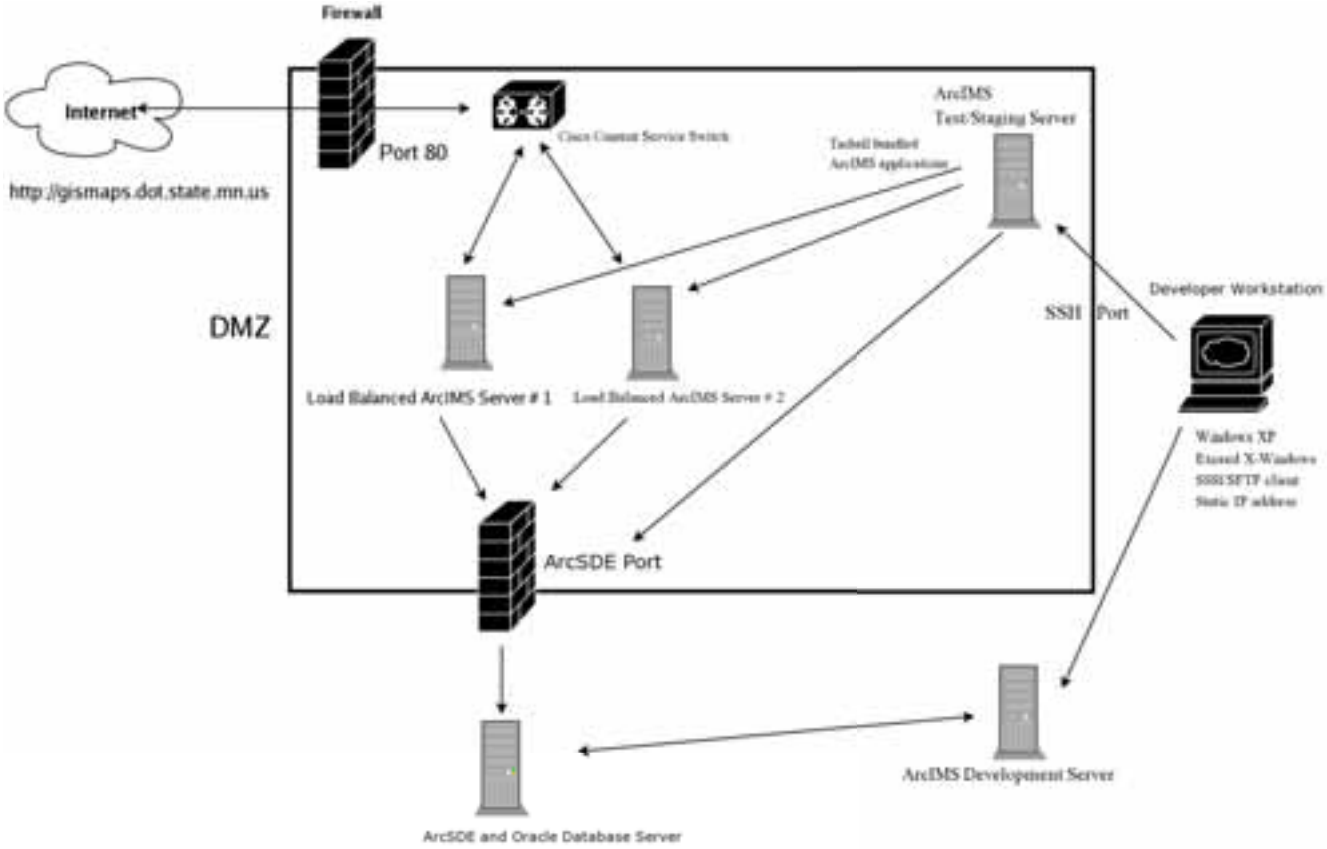

Illustration 3: ArcIMS Application Development Support

5.2 Authentication

The development and test ArcIMS servers require developers and application administrators to log in to a user account. This differs from the ArcIMS service authentication handled within the ArcIMS system. Each development group manages an area of Web application content on the servers. We grant ownership and group access through Lightweight Directory Access Protocol (LDAP) so that development groups can protect their own AXL files and directory structures. Our organization uses the Microsoft Active Directory for network user accounts and general workstation logins. We purchased a product by Quest Software called Vintela Authentication Services (VAS) to connect Red Hat Linux authentication into our Active Directory. The application development groups that we create in the Active Directory map to permissions in the Red Hat Linux file system to support ArcIMS. (Illustration 4 displays the set up of Microsoft Active Directory Users and Computers with our ArcIMS developer groups.)

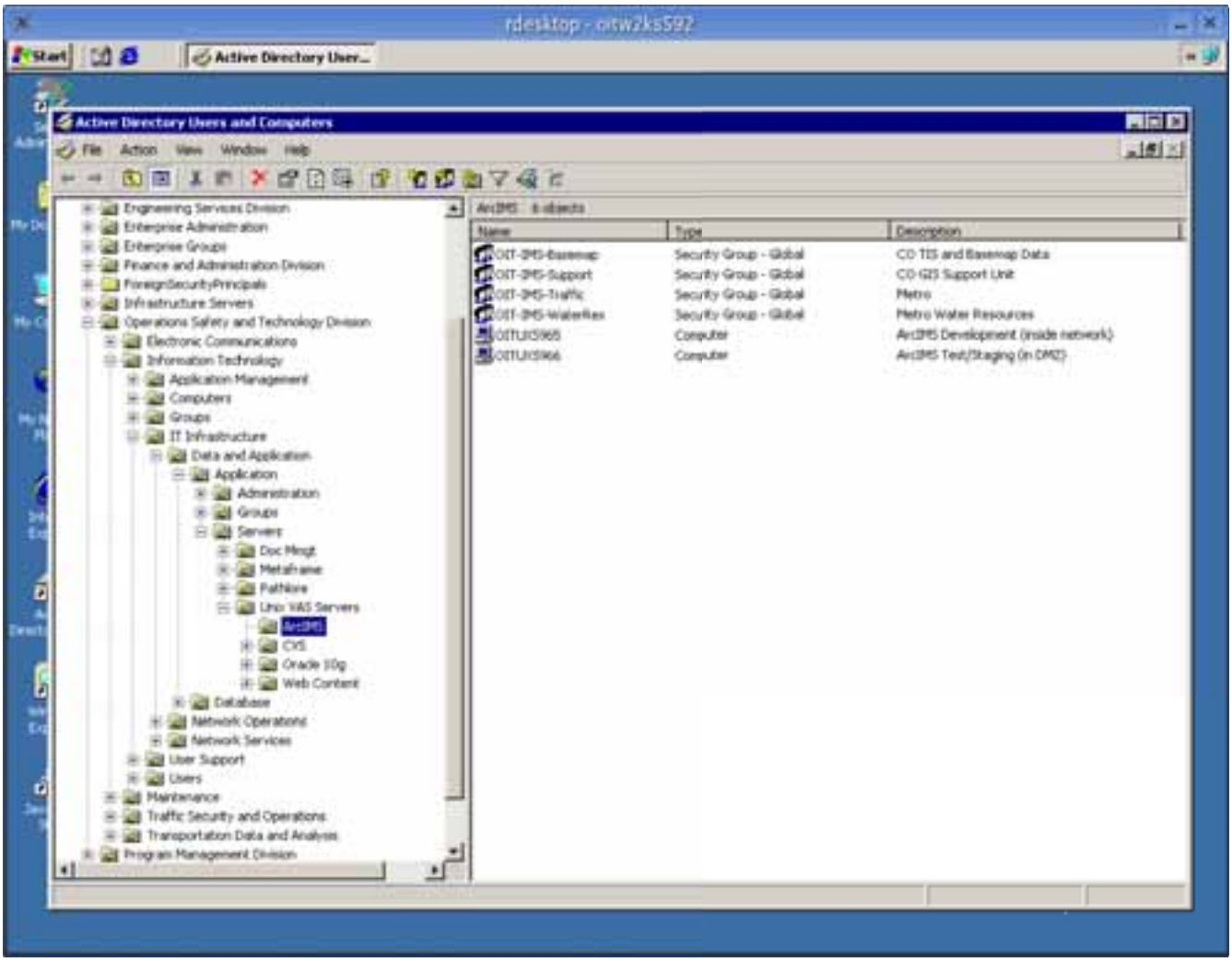

Illustration 4: ArcIMS Active Directory Groups

5.3 File System Organization

The default ArcIMS installation creates a directory for AXL files and for each Web site. A multiuser configuration requires that each AXL file retain owner and group information for each developer. The website directory structure begins with ownership by a local arcims account created to run the ArcIMS software. Subdirectories below the website directory contain ownership or file permissions for each developer. In the Linux file system configuration, the top level website directories have the “sticky bit” or “set GID” permission set. This permission setting will be inherited for all files and subdirectories created below the website directory.

Our developer and test/staging systems contain nearly identical ArcIMS file system structures. As ArcIMS Web sites and services are developed, they will be copied through workstation software to the test/staging system. Meta data is kept in a common directory in one location on each ArcIMS computer so that all map services display consistent information. The developers review and test the Web sites before submitting a request to the system administrators to deploy the ArcIMS production Web site.

5.4 System Administration Components

The developers and application administrators do not directly access our production ArcIMS servers. When the developer submits a production deployment request to system administration, support software creates a Unix tape archive or “tarball” bundle of the ArcIMS system in the developer's home directory. We developed the support software as an aid for the creation of the tarball.

5.4.1 Deployment Support Utilities

The ArcIMS Linux servers have no displays and only serve users through command line logins. Our developers prefer to work with graphical programs to perform tasks. An open source project called Xdialog by Thierry Godefroy provides a method to create graphical dialogs through Unix shell programming. We used the Xdialog package to give developers a graphical interface to create a tarball of their ArcIMS Web sites that system administrators deploy to the production servers. (Illustration 5 shows the beginning of a series of interactive dialogs to produce the deployment.)

The archive produced by Xdialog contains all the files, directories and paths within the Linux file system in a single disk file. Our system administration group runs an automated deployment program to copy the archive to the production systems. We developed the deployment program to simplify the process. The archive is unpacked on the two load balanced production systems into the correct file path locations. System administrators run the final step to create a map service through the ESRI aimsadmin uitlity on each load balanced server.

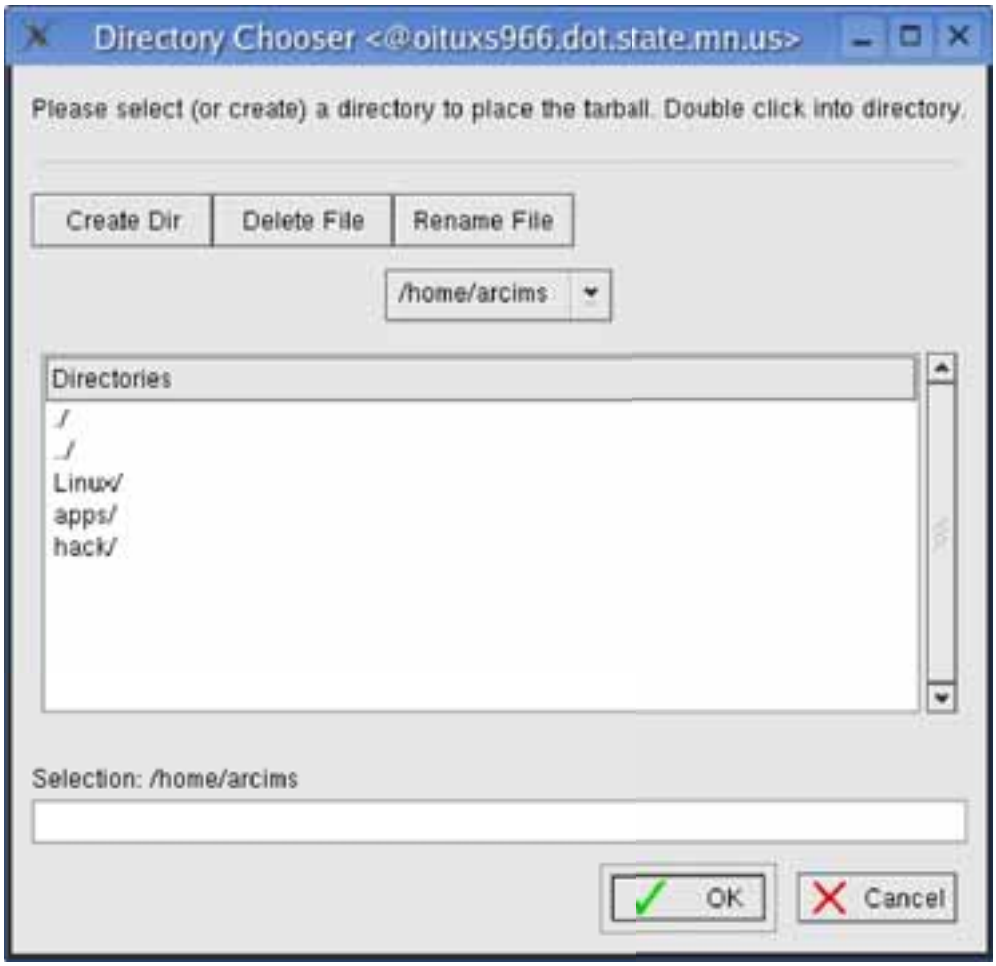

Illustration 5: Xdialog Interactive Menu

6. Conclusions and Future Developments

The release of ArcIMS for the Linux platform integrates well into a development and deployment infrastructure that delivers reliable, high performance map services. Red Hat Linux has good support processes to complement the installation with Web servers, security, developer interaction and comprehensive administration. Our baseline system can be imaged and restored with minimal effort. These features incorporate a design that can be scaled and grow to meet changing enterprise needs.

Our ArcIMS infrastructure has the following components that give it good flexibility and manageability:

(1) Reverse Proxy – communicates well with Tomcat and ArcIMS, very stable, can be reconfigured easily

(2) Load Balancing – provides high performance map services and ease of server maintenance, the server farm can be scaled to accommodate needs

(3) System Imaging – creates clones to keep configurations constant between systems, reduces troubleshooting efforts

(4) Developer Interfaces – implements familiar graphical interaction method to create deployments

(5) Security – allows ArcIMS to serve the World Wide Web with low risk to enterprise networking and computing

As software releases become more sophisticated our system can grow with the technology. At some future point, ArcIMS will communicate with Jboss and Apache 2.2. These two Web servers will have features that improve on the current schemes. We also will move into 64-bit computing to break through the 2 gigabyte barrier for the Java virtual machines. This will permit large scale mapping and placing more applications into the map server. Improvements in networks with gigabit speeds will create opportunities to access much larger data bases. In conclusion, the future with these infrastructures will deliver map services to a larger user base promote growth in the map consuming community.

Appendix

1. Apache Reverse Proxy configuration file directives

```
Redirect permanent /esriconn http://<Apache/Tomcat/ArcIMS computer name>/esriconn


<Location /esriconn>
      ProxyPass http://<Apache/Tomcat/ArcIMS computer name>:8080/servlet
      ProxyPassReverse http://<Apache/Tomcat/ArcIMS computer name>:8080/servlet
</Location>

Redirect permanent /servlet http://<Apache/Tomcat/ArcIMS computer name>/servlet

<Location /servlet>
      ProxyPass http://<Apache/Tomcat/ArcIMS computer name>:8080/servlet
      ProxyPassReverse http://<Apache/Tomcat/ArcIMS computer name>:8080/servlet
</Location>

Redirect permanent /esriadmin http://<Apache/Tomcat/ArcIMS computer name>/esriadmin

<Location /esriadmin>
      ProxyPass http://<Apache/Tomcat/ArcIMS computer name>:8080/esriadmin
      ProxyPassReverse http://<Apache/Tomcat/ArcIMS computer name>:8080/esriadmin
</Location>

</VirtualHost>
```

2. Example mod_security rule for ArcGIS access

```
   # Only accept request encodings we know how to handle
   # we exclude GET requests from this because some (automated)
   # clients supply "text/html" as Content-Type
   SecFilterSelective REQUEST_METHOD "!^(GET|HEAD)$" chain
   SecFilterSelective REMOTE_IDENT "^ArcGIS/*$" skipnext
   SecFilterSelective HTTP_Content-Type "!(^application/x-www-form-urlencoded$|^multipart/form-
data;)" skipnext
   SecFilterSelective HTTP_Content-Type "!(^application/x-www-form-urlencoded$|^multipart/form-
data$|^text/text$|^text/plain;)"

   # Do not accept GET or HEAD requests with bodies
   SecFilterSelective REQUEST_METHOD "^(GET|HEAD)$" chain
   SecFilterSelective HTTP_Content-Length "!^$"

   # Require Content-Length to be provided with
   # every POST request
   SecFilterSelective REQUEST_METHOD "^POST$" chain
   SecFilterSelective HTTP_Content-Length "^$"

   # Don't accept transfer encodings we know we don't handle
   SecFilterSelective HTTP_Transfer-Encoding "!^$"
```

3. Red Hat System V Script for ArcIMS

```
#!/bin/sh
#
#       /etc/rc.d/init.d
#
# Automatic start of the ArcIMS web server running Linux system
#
# chkconfig: 345 84 15
# description: Activates the ArcIMS web server which
# processname: java
#
#
# Source function library.
. /etc/rc.d/init.d/functions

debugfile=/home/arcims/arcims.log
AIMSHOME=/ea/adm/arcgis/arcims
J2SDK_HOME=/usr/java/current
lockfile=/var/lock/subsys/arcims

chkuptime() {

  upt=`uptime | grep min`
  if [ -n "$upt" ] ; then
    upmin=`echo $upt | tr -s ' ' | cut -d' ' -f3`
    if [ $upmin -le 2 ] ; then
      sleep 35
    fi
  fi
}

#echo "############################################################" >> $debugfile
#echo "############################################################" >> $debugfile
#echo "Initializing debug file" >> $debugfile
#echo "`date`" >> $debugfile
#echo "Current working directory is: `pwd`" >> $debugfile
#echo "Script called with parameter $1" >> $debugfile
#echo "arcims in `pwd` called with parm $1" >> $debugfile

RETVAL=0

# See how we were called.
case "$1" in
  start)
        # arcims 5 starting up
         if [ ! -f $lockfile ] ; then
#           echo "Starting arcims from runlevel with parm $1" >> $debugfile
            chkuptime
            touch $lockfile
            su - arcims -c "cd $AIMSHOME/Xenv; tcsh aims_bootup"
            echo -n "Starting ArcIMS service daemons: "
            echo_success
            echo
```

```
          echo_success
          echo
          pgrep -U arcims 'aims*' >> $lockfile
          pgrep -U arcims 'java*' >> $lockfile
        else
          echo "Lock file: $lockfile exists, assume arcims already running, will not run startup script"
#         echo "Lock file: $lockfile exists, assume arcims already running, will not run startup script"  >> $debugfile
        fi
       ;;
  stop)
       if [ -f $lockfile ] ; then
          pgrep -U arcims 'aims*' >         /tmp/pidnow
          pgrep -U arcims 'java*' >>        /tmp/pidnow
          aipid=`cat /tmp/pidnow` ; /bin/rm /tmp/pidnow
          if [ "$aipid" == "`cat $lockfile`" ] ; then
            echo "Stopping ArcIMS service daemons: "
#             echo "Shutting down ArcIMS through $AIMSHOME/Xenv/aims_shutdown" >> $debugfile
            su - arcims -c "cd $AIMSHOME/Xenv;  tcsh aims_shutdown"
#             echo "After $AIMSHOME/Xenv/aims_shutdown" >> $debugfile
            echo -n "Stopped ArcIMS service daemons: "
            echo_success
            echo
          else
            echo "No matching arcims pid=$tcpid for saved=`cat $lockfile`"
          fi
          pgrep -U arcims 'aims*' >         /tmp/pidnow
          pgrep -U arcims 'java*' >>        /tmp/pidnow
          aipid=`cat /tmp/pidnow` ; /bin/rm /tmp/pidnow
          if [ "$aipid" == "`cat $lockfile`" ] ; then
            echo "ArcIMS still running, aims_shutdown failed, killing manually"
            for each in $aipid ; do
              kill -9 $each
            done
          fi
          rm $lockfile
       else
          echo "Lock file: $lockfile does not exist, assume arcims not running, cannot run shutdown script"
#           echo "Lock file: $lockfile does not exist, assume arcims not running, cannot run shutdown script"  >>
$debugfile
       fi
       ;;
  status)
       aipid=`pgrep -U arcims java | tr '\n' '\040'`
       if [ -n "$aipid" ] ; then
           echo "ArcIMS (pid $aipid) is running..."
       else
           echo "ArcIMS is stopped"
       fi
       RETVAL=$?
       ;;
  *)
       echo "Usage: aimsd {start|stop|status}"
       echo "Usage: aims {start|stop|status}"
       exit 1
esac

exit  $RETVAL
```

References

1. ArcGIS® Enterprise Security: Delivering Secure Solutions, An ESRI® White Paper, July 2005

Victor R. Conocchioli
State of Minnesota
Department of Transportation, GIS Administrator
MS 240
395 John Ireland Boulevard
Saint Paul, Minnesota 55155
phone: 651-296-6094
fax: 651-297-1473
email: vic.conocchioli@dot.state.mn.us