

Using The ASP GUI Topology Tool

Ted Faber
USC/Information Science Institute
faber@isi.edu

February 18, 2003

1 Introduction

The ASP GUI topology tool (*GUITopology*) is a simple, graphical interface that allows experimenters to deploy collections of connected ASP nodes on multiple machines. These collections can be deployed either in the ABONE[2] or on local nodes over which the user has control. Specifically, there are provisions made in GUITopology to deploy ASP on ABONE nodes and to start and stop them there. When used for local deployment, GUITopology is useful for creating the configuration files.

This document will describe briefly the configuration for ASP[3] and for running ASP in the ABONE. Then it will describe using the tool to accomplish these these tasks. The discussion of ASP and the ABONE will necessarily be brief; detailed configuration and use of these environments is available in the cited references.

2 Configuring ASP

ASP is an Execution Environment (EE) for active networking applications. As such, its primary role is to dynamically load and provide services for Active Applications (AAs), the programs that provide services to users and the network.[5]

ASP is remarkably flexible in the ways in which AAs can start. An AA may be requested directly by a user by passing an AA specification (AASpec) to a running ASP via a known TCP port, called the API port; ASP can be configured to run a specific AA automatically when a packet appears addressed to a given VNet (see below) or UDP port; finally other UDP or VNet ports can be designated as *wildcard ports*, and packets arriving on them must include the AASpec of the application they are intended for. If an AA requested in a packet received on a wildcard port is not already running, ASP starts an instance of it.

Any time an AA is started by ASP, it is potentially loaded across the network. ASP requests the Java class files that make up the application from another running ASP node that has them. Which nodes to ask is part of the AASpec, but by convention the same TCP or RDP[4] port is used by all ASP nodes in a given topology.

The other primary service provided by ASP is a virtual networking topology. This topology is overlaid on UDP, and defines its own set of virtual interfaces, addresses and transport ports, all visible only in ASP.[3] Configuring this virtual topology is the primary function of GUITopology.

ASP is a fairly complicated system to configure, but the primary configuration files of interest to a user are:

- asp.conf

This file controls several global options of ASP, including:

- File Server Port

The TCP port that this instance of ASP will use to load AAs from another instance of ASP.

- API Port

The TCP port on which ASP will listen for API requests from users.

- Wildcard Ports

The VNet and UDP ports on which ASP will look for an AASpec in the incoming packet and start a new AA based on it. For example, this is the VNet port on which an incoming RIP packet will start a RIP AA if none is already running. Although the UDP and VNet ports are distinct they are included in this bullet.

- Dedicated Ports

The VNet and UDP ports bound to a specific AA. These are not used very often.

- Bootstrapped AAs

These are AAs that are started when ASP starts. This is related to the AASpec.conf file.

- AASpec.conf

Each bootstrapped AA or AA started on a dedicated channel is specified by a single identifier, like “rip,” but a full AASpec is required for ASP to load the AA. This file maps from those identifiers to AASpec for loading.

- capabilities

This file contains the special capabilities that ASP grants to an AA. For example the RIP AA needs to access the route tables.

- TopologyTable.txt

The virtual topology in which this ASP node operates. The interfaces and routes for this node are required, and generally this will contain the complete description of the topology.

In many cases there are simple defaults for these files, but especially in the case of `asp.conf`, there are many places for conflicts. For example, if any of the TCP or UDP ports specified in `asp.conf` cannot be opened, ASP will exit. `TopologyTable.txt` also specifies several UDP ports that need to be conflict free.

The `TopologyTable.txt` contains 3 kinds of information ¹:

- Interface Descriptions

These describe the basic connectivity of the network. Each interface description describes one side of a point-to-point link between two ASP nodes. The VNet address of each side is given as well as information about which UDP addresses and ports the virtual link is carried over. If ANEP[1] is in use, the ANEP type ID is given.

- Route Descriptions

These define how a node routes packets to nodes that are not directly connected. They represent static routes. A simple topology may not need them, and routing can also be done using the RIP AA.

- Coordinate Descriptions

These are ignored by ASP, and are hints for laying out the topology in a viewer or editor.

Configuring ASP requires generating consistent copies of all these files, but mostly generating a `TopologyTable.txt`. The next section describes how to use `GUITopology` to generate a topology table; we will return to the other files when we discuss deploying ASP in the ABone.

3 Using GUITopology To Create TopologyTable.txt

When a user invokes `GUITopology`² they are presented with a blank drawing canvas on which they will draw the topology they want. There are some options at the bottom and a menu above that we will get back to.

To insert a node, the user clicks on the white area where they want the node positioned. A dialog box pops up asking for a name for the node, and that name is used. If no name is entered (but the enter key is typed), `GUITopology` assigns a name of the form `node x` where x is an integer. If the user decides not to add a node, clicking cancel on the dialog cancels the operation. The node appears on the white area as a red circle with the name in blue near it. A node can be moved by placing the mouse pointer on it, holding down a button, and dragging it. A node can be deleted by double clicking on it, and choosing “Delete Node” from the menu.

¹The syntax of each type is omitted - after all the whole purpose of `GUITopology` is to hide it. The syntax is available in [3]

²on most systems this is accomplished by typing `java tools.topology.GUITopology` at the shell prompt.

Once two or more nodes have been created, a link can be added between them. The user clicks on the source node, which makes it active and colors it yellow, and then clicks on the destination, which forms a link. The link is drawn as a blue line, labeled in red at each end with the VNet address of the interfaces.

Clicking on an existing link brings up a dialog box that allows a user to edit the interface names, i.e., the VNet addresses, or to delete the link altogether.

The specifics of the topology file can be modified using the parameters at the bottom of the window. Specifically:

- **Bind to any**
If this check box is selected, the TopologyTable created will specify that ASP binds to the `INADDR_ANY` address on the machine. De-selecting this will request that hosts bind to the interface with the same IP address as the name of the node. What happens if that DNS address maps to multiple IP addresses is undefined. In most cases this box can be safely checked.
- **Use anetd**
If this check box is selected, the TopologyTable created will specify that ASP nodes should not open UDP ports to listen for VNet traffic, but should expect incoming traffic from anetd. Choosing this box also creates a TopologyTable.txt that sends all outgoing traffic to the same UDP port, presumably anetd's. When this box is checked, the UDP port is set to the default Anetd port. If this topology is to be deployed on the ABONE, this box should be checked.
- **ANEP ID**
This specifies the ANEP identifier to put on outgoing packets and to look for on incoming packets. It should be set in the ABONE. Any string that is not a number results in no ANEP identifier being assigned.
- **UDP port**
This specifies the lowest UDP port to be used for carrying VNet traffic, or the only port to be used if "Use anetd" is checked. Outside the ABONE, each interface maps to a distinct UDP port, and GUITopology assigns them in ascending order starting from the value in this field. For example if a node has 3 VNet interfaces, and this field is 8585, the three interfaces will use UDP ports 8585 8586 and 8587. If "Use anetd" is set, they will all use the same UDP port as a destination.

To save or load topology files, the file menu is used. The choices are fairly conventional, but are:

- **New**
Clear this topology file and start with a blank slate.
- **Open**

Open a new topology file. A dialog box will appear that will help the user navigate the file system to find the proper file. The file to open should be of the format of an ASP TopologyTable.txt file. Any file written by GUITopology will work. Starting GUITopology with a filename on the command line opens that file.

- Save
Save the topology into its known filename. If the current state has never been saved or read from a file, this option will be unavailable. This choice saves to the most recent Open or Save As choice.
- Save As
Save the topology to a given file. A dialog similar to the one used for “Open” is presented.
- Exit
Close GUITopology. If there are unsaved changes, the user is prompted for how to handle them. If a code server has been started, it is stopped (starting a codeserver is discussed below).

When a topology is saved, the parts that are useful to ASP are saved in the filename given by the user. The parts that are useful only to GUITopology are saved to a second file. If the filename the user gave ends in `.txt`, the GUITopology data is saved in a file with the `.txt` changed to `.gui`, otherwise `.gui` is simply appended to the user filename. If this second file is not available, some settings will be lost.

The first 2 entries under the “Tools” menu manipulate routes in the topology. If there are static routes in the TopologyTable.txt, GUITopology keeps them with their nodes. In other words the routes are unchanged unless a node is deleted, in which case the routes originating on that node are deleted. To forcibly remove all routes from the topology, use the “Clear Routes” choice from the “Tools” menu. Choosing “Generate Routes” deletes any existing routes, then creates static routes that provide shortest-hop-count routes between all nodes.

The functionality described above is sufficient to create a TopologyTable.txt that can be distributed to several local nodes (or shared via NFS) and allow for some ASP experimentation. The other configuration files can be edited by hand or the defaults from the ASP distribution used. GUITopology does not support distribution of these configuration files in arbitrary environments because the possible mechanisms vary widely. However, in the ABONE, where mechanisms are more uniform, more services are provided.

There is also a provision for creating local copies of the other ASP configuration files that is discussed below.

4 Working In The ABONE

Deploying a topology of ASP servers in the ABONE to run experiments has several phases, and GUITopology can simplify many of them. First, the user has to have appropriate credentials granted

from the ABONE and have the ABONE software, particularly the `sc` command, installed in their path. That part GUITopology cannot help with.

Once the user has appropriate credentials and software, starting a topology and experimenting with it consists of:

- Establishing a code server to serve any AAs needed by the topology.
- Distributing the ASP configuration files to the ABONE nodes.
- Starting ASP on the ABONE nodes.

Once the experiment is done, the codeserver and ABONE nodes need to be stopped.

Because there are several interrelated files that need to be consistent, GUITopology has a set of general ABONE parameters that can be configured. For most cases the defaults are exactly right, but we describe the parameters here to make things somewhat clearer. Choosing “ABONE Info” from the “Tools” menu gives the user the opportunity to edit the following fields:

- File server port
The TCP and RDP port used for the file server. Any code server and the ASP EEs started in the ABONE need to agree on this. It is initialized to 5000 plus the last byte of the IP address of the host on which GUITopology is run, to avoid conflicts with other hosts.
- API port
The API port for ABONE nodes. Like the fileserver port, this is initialized to 4000 plus the last byte of the host IP address.
- Code server
The DNS name of the code server - that is, the machine that will be serving AA code to the topology. It does not have to be part of the VNet topology. If the code server is to be started from GUITopology, this value must be the current host.
- Code base
This is the home directory of the code server.
- EE server
The node that serves EE code to the topology. Do not modify this unless you are an experienced ABONE user.
- EE base
The directory on the EE server containing the EE code to be served. Do not modify this unless you are an experienced ABONE user.

- EE class
The class to load from the EE server to run ASP. Do not modify this unless you are an experienced ABONE user.
- ABONE user:
The ABONE user under which the topology is running. The default (`anpub`) is usually correct.
- Topology comment
The ABONE comment field that shows up on various `sc` commands, like `QUERY`.

Again, the defaults are usually correct, and should a user need to restore them, there is a button to do so on the “ABONE Info” dialog.

The parameters above appear in many of the ASP configuration files discussed earlier, and some of those files may need to be customized. To modify those files, a user can choose “Modify ASP Config” from the “Tools” menu, choosing the file you wish to modify. This pops up a simple text editing interface that allows simple changes to be made to the selected file.

The specific syntax of these files is discussed in the ASP documentation[3]. The major modification for GUITopology is that before the files are sent out to the ABONE, a simple variable expansion is done. The following variables are recognized and expanded in the files:

- `${FILE_SERVER_PORT}`
- `${API_PORT}`
- `${CODE_SERVER}`

The default files use these variables to customize the configuration files to the “ABONE Info” parameters. The default configuration files contain examples. Should a user modify a configuration and need to restore the default behavior, the configuration editing choices offer that option.

Selecting “Save Local ASP Configuration” will save copies of `asp.conf`, `AAspec.conf`, and `capabilities` to the current directory, either for debugging, or for use in a local topology.

Once the parameters are set and the configuration files customized, the actual ABONE deployment can begin. First a code server should be started. The “Start Codeserver” menu selection will start such a server. It can be stopped via the “Stop Codeserver” menu selection, and will be stopped when GUITopology exits.³

Once the code server is started, the configuration files are distributed using the “Configure ABONE Topology” choice, which uses `sc` to distribute copies of the configuration files to the ABONE nodes. This step can take place before or after the code server is started.

³Some JVMs do not properly implement the process control methods, and on those machines the code server must be stopped by a standard system command, e.g, `kill`.

Once the code server is started and the configuration has been passed around, the topology can be started. “Start ABONE Topology” starts the EEs running and “Kill ABONE Topology” stops the nodes.⁴ Both choices use `sc` and the ABONE info parameters to start or stop the EE on the given nodes.

GUITopology also allows users to retrieve output files from nodes running under its control in the ABONE. To retrieve output, click twice on a node, and select the output to retrieve from the “Retrieve ABone File” submenu. Standard choices are the ASP log, the standard output, which is usually the ASP and `anetd` logs, or the standard output, which is often empty. To retrieve another file, choose “Other File” from the same submenu and give the name in the dialog box. Files are rooted in the `anetd` directory.

The requested file will appear in a window with scrollbars that allow the user to navigate the file. There is also a button that allows the user to save the file locally for more intensive processing. The user picks the save file name from a standard file chooser.

5 Conclusion

This document has covered the basic operation of the GUITopology tool. In closing, it is wise to remember that this is a simple tool designed to make getting started in the ABONE as easy as possible. However, more esoteric configurations in the ABONE may be more easily done by hand. Even if files need to be tweaked by hand, it is often worthwhile to start from a GUITopology generated topology and make modifications.

References

- [1] D. Scott Alexander, Bob Braden, Carl A. Gunter, Alden W. Jackson, Angelos D. Keromytis, Gary J. Minden, and David Wetherall. Active network encapsulation protocol (ANEP). ftp://www.cis.upenn.edu/pub/switchware/public_html/ANEP/index.html, 1999.
- [2] S. Berson, B. Braden, and L. Ricciulli. Introduction to the abone. <http://www.isi.edu/abone/DOCUMENTS/ABarch/>, 2000.
- [3] Bob Braden, Alberto Cerpa, Ted Faber, Bob Lindell, Graham Phillips, and Jeff Kann. *Introduction to the ASP Execution Environment*, 2000. <http://www.isi.edu/active-signal/ARP>.
- [4] Ted Faber. *An Implementation of the Reliable Data Protocol for Active Networking*, 2000. <http://www.isi.edu/active-signal/ACC>.
- [5] Active Network Working Group. Architectural framework for active networks. <http://www.cc.gatech.edu/projects/canes/papers/arch-1-0.ps.gz>, July 1999.

⁴Kill currently depends on `perl` being in the user’s path as well as `sc`.