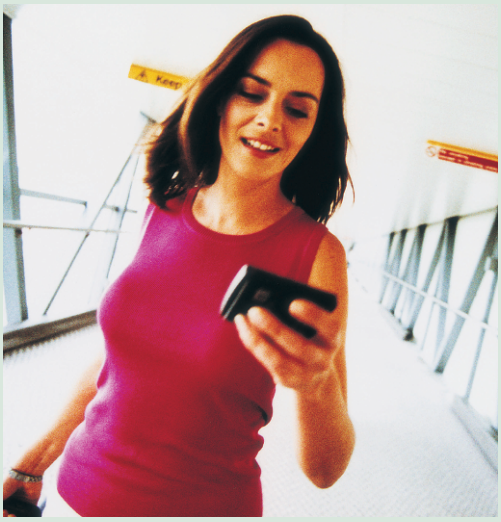
The Sysadmin's Daily Grind: Active Port Forwarder

Bridging and Tunneling



Even if a machine resides behind a firewall or a NAT gateway, it is possible to respond to requests for public services – thanks to the Active Port Forwarder. BY CHARLY KÜHNAST

It would seem that most couples get to know each other at their workplace. I don't intend to speculate on the long-term stability of a relationship of this kind, but I do know that it is normally a good idea to separate your private life and your work. In the case of my Web server, private means that I set up the server at home.

As I do not have a leased line at home, my server's IP changes each time it dials up the Internet. One typical workaround for this issue, is to use a DNS service like Dyndns [1] which will map a server's name to its current IP address. This will not work for me, as my machine lives behind a NAT gateway, the DSL router. The router takess the public IP supplied by my provider for its own use, leaving my network with private addresses in the the 192.168.*X*.*X* range.

The router prevents machines on the Internet from accessing my local system – quite a useful security feature in normal circumstances, but a bit unfortunate in this case. Modern Linux firewalls will allow you to use DNAT (Destination NAT) to forward an external port to an internal machine, but many devices simply do not provide this kind of facility.

Back-to-Front

As NAT routers are typically quite permissive towards outgoing traffic (the exception being routers with packet filter rules), I have no trouble establishing connections to the Internet. All I need is the right kind of software to allow me to build a tunnel back to my private network (see Figure 1). And that is exactly what Active Port Forwarder [2] (Version 0.5.3) does.

Compiling the sources will create the client and server components. In addition to these, I need a "bridgehead" on the Internet, a server that will provide

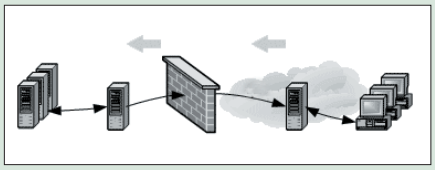


Figure 1: The AF server provides a bridgehead on the Internet.

me with a Shell. A friend was so kind as to provide that (not a colleague, as previously explained) in the form of access to his leased root server. Now I need to compile a port forwarder on the bridgehead. I only need the server component of the compiled binaries. The server is launched with the following default settings by typing *./afserver*:

- The server will listen on port 50127.
- The client component, which I will be creating next, can attach to port 50126.
- The server will handle a maximum of five simultaneous TCP connections.
- Logging is disabled.

If you do not like these particular settings, you can simply create a configuration file (the package includes an example) and enter the following:

```
./afserver -f configfile
```

The client is as simple to set up. I compiled the package on my home Web server, and ran the client component:

```
./afclient -n servername -p 80
```

Replace *servername* with the FQDN or IP address of the bridgehead. Finished! Surfing to port 50127 on the server now takes me my home Web server. The connection between these two machines is SSL encrypted, by the way. So each make will create both a server and a client certificate. ■

INFO

[1] Dyndns: *http://www.dyndns.org*

[2] Active Port Forwarder: *http://www.gray-world.net/pr_af.shtml*

THE AUTHOR

Charly Kühnast is a Unix System Manager at the data- center in Moers, near Germany's famous River Rhine. His tasks include ensuring firewall security and availability and taking care of the DMZ (demilitarized zone).