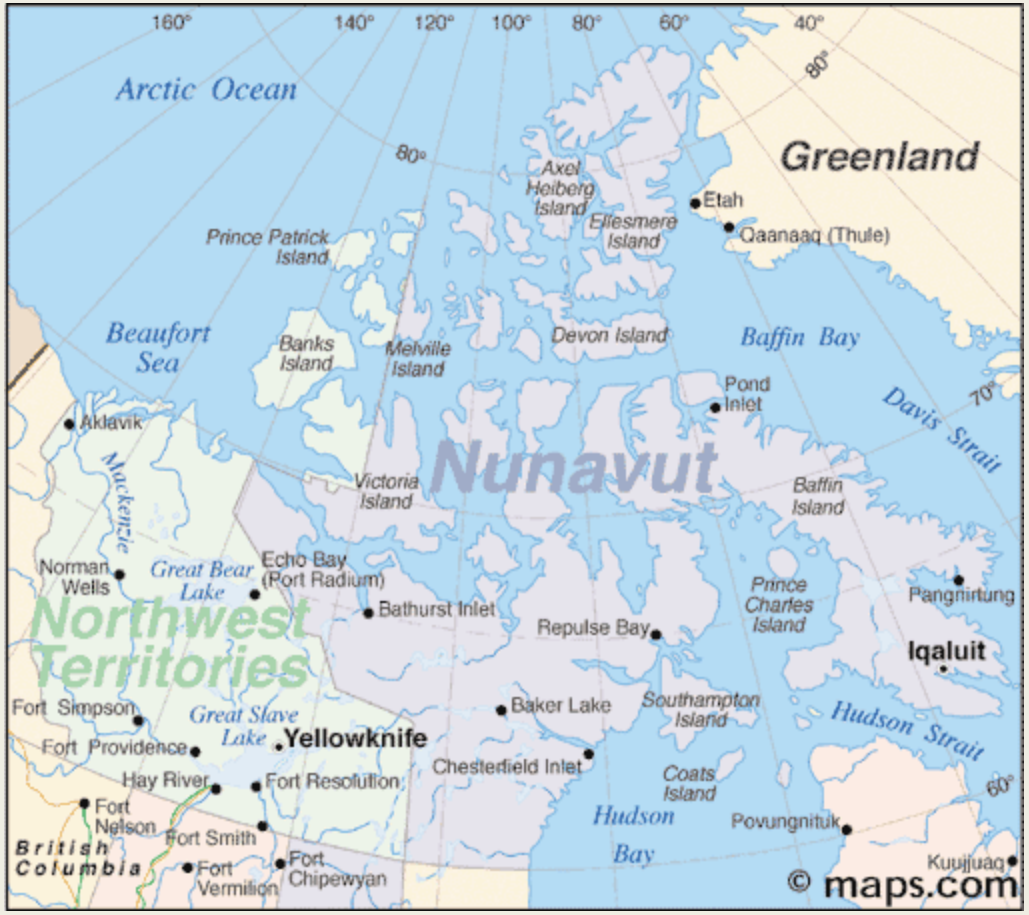
Tales from the North

System Administration of a Geographically Disperse Network

Dwayne Hart
dwayne.hart@gmail.com

Background

Over five years ago I joined a Northern Internet Service Provider (ISP) in Yellowknife, NWT, Canada providing high-speed internet, email, website hosting, and other specialized network services to approximately 50 communities located within the Northwest Territories and Nunavut. The satellite headend was located in Ottawa.


160°
140°
120°
100°
80°
60°
40°
80°
Arctic Ocean
80°
Greenland
Axel Heiberg Island
Etah
Ellesmere Island
Qaanaaq (Thule)
Prince Patrick Island
Beaufort Sea
Banks Island
Melville Island
Devon Island
Baffin Bay
Pond Inlet
70°
Davis Strait
Aklavik
Mackenzie
Nunavut
Victoria Island
Baffin Island
Norman Wells
Great Bear Lake
Echo Bay (Port Radium)
Bathurst Inlet
Prince Charles Island
Pangnirtung
Repulse Bay
Iqaluit
Northwest Territories
Fort Simpson
Great Slave Lake
Yellowknife
Baker Lake
Southampton Island
Hudson Strait
Fort Providence
Chesterfield Inlet
Hay River
Fort Resolution
Coats Island
60°
Fort Nelson
Hudson Bay
Povungnituk
Fort Smith
British Columbia
Fort Vermilion
Fort Chipewyan
Kuujjuaq
© maps.com

Typical Site Configuration

- Satellite equipment
- Local Area Network equipment (router and switch)
- Wireless network equipment
- Two or more physical servers

Focusing on Remote Sites

Originally each site ran a full install of FreeBSD 5.2.1 on the commodity hardware. As the server hardware failed. The entire unit was replaced with reserve systems when that option was depleted new systems were purchased.

Unfortunately, the components comprising these newer systems were not supported by FreeBSD 5.2.1. Causing a major headache. How do we run the same software stack on a newer version of FreeBSD?

The solution was to create two thin jail instances per site and bundle certain services together. For example, sendmail and DNS were grouped together while LDAP, DHCP and our network usage software stack were bundled together.

Troubleshooting

Not all platforms were running the same OS version. If a system or jail encountered an issue how do we try to diagnose the issue? We were lacking debugging/analyzing applications.

To get around this I started to think about how do we take all our remote heterogeneous systems and bring them inline?

The solution I came up with was to implement a Diskless FreeBSD server on both of the community servers. Each system had DHCP, TFTP and NFS service configured.

The NFS service offered up a share that was created containing a custom built FreeBSD tarball.

Then in turn. One of the servers became the application server and hosted the community jail instances. While the other machine was configured to boot off its network card and act as a Diskless client.

At which point, I carved up the disk with a new layout and UFS2 file systems. Then used the tarball to install the custom FreeBSD OS. After which the client is rebooted. The build process is reversed and the jails are migrated to the newer OS platform.

TFTP

Even though a base version is installed by default on all FreeBSD systems, This instance was replaced with tftp-hpa from ports. To make use of the bug fixes and the eventual use of gpxeboot, memdisk from the SYSLINUX project.

/etc/inetd.conf

tftp dgram tcp wait root /usr/local/bin/tftp tftpd -p -s -B 1024 --ipv4 /tftpboot

/etc/rc.conf

Network daemon (miscellaneous)
inetd_enable="YES"
inetd_flags="-wW -C 60 -a 192.168.1.2"

DHCP

Made use of a stock build of isc-dhcpd (server) from ports

/usr/local/etc/dhcpd.conf

```
subnet 192.168.1.0 netmask 255.255.255.0 {
  #use-host-decl-names on;
  option subnet-mask 255.255.255.0;
  option routers 192.168.1.254;
  option broadcast-address 192.168.1.255;

  next-server 192.168.1.2;

  ### Host:volterra
  host 192.168.1.20 {
    hardware ethernet 00:1e:4f:f0:69:e7;
    fixed-address 192.168.1.20;
    filename "pxeboot-6.2-i386";
    option root-path "192.168.1.2:/diskless/fbsd62-i386";
  }
```

NFS

Basic configuration of the NFS service was put in place.

/etc/exports

/diskless -alldirs -ro -maproot=root 192.168.1.0/255.255.255.0

/etc/rc.conf

```
### Network daemon (NFS): All need rpcbind_enable="YES" ###
rpcbind_enable="YES"
nfs_server_enable="YES"
nfs_server_flags="-t -u -h 192.168.1.2 -n 15"
mountd_enable="YES"
mountd_flags="-r"
```

Debugging One Step Further

Download the latest version of the SYSLINUX project from kernel.org. Then copy the following binaries to the root of your tftp directory.

chain.c32, gpxelinux.0, ldlinux.c32, libcom32.c32, libutil.c32, memdisk, menu.c32, reboot.c32, vesamenu.c32

Create a directory called pxelinux.cfg and a file called default containing the following content...

```
ui menu.c32
menu title Utilities

label memtest
    menu label Memtest
    linux memdisk
    initrd images/memtest86.iso
    append iso raw

label reboot
    menu label Reboot
    kernel reboot.c32
```

Create a directory called /tftpboot/images and store whatever ISO you wish to use in there.

You will then change the string 'filename "pxeboot-6.2-i386";' to 'filename "gpxelinux.0";' in /usr/local/etc/dhcpd.conf.

Perform a sanity check of your dhcpd.conf file to ensure that you did not busted. If correct, restart the service reboot your client machine and test the machine's memory using the latest memtest ISO.

Core Infrastructure

Seeing that the jail implementation worked out in the remote communities we made use of the same framework so that in the event of a server failure we could launch the hosted jails on another platform.

Once the remotes were updated to a standard version of FreeBSD. The same techniques were used to rebuild the core servers located at the satellite headend.

Interesting Notes

- By deploying a different custom FreeBSD tarball. We can either setup a new NFS share and in the event we needed to rebuild the server with either OS instance. Or replace the previous share with the newer material.
- By slicing up the hard drive to contain three slices. The first two can be used to contain the system's OS while the third can be used to store our jail instances. We can then use the command "boot0cfg -s 2 ad0" to instruct the boot manager to boot from the second slice.
- The content contained in the custom FreeBSD tarball was turned into an ISO. This could then be used as a building block.
- A build/patch server was created in order to keep in-line with security notifications from FreeBSD.

Future Work

With the knowledge and experience I had garnered from this position administering FreeBSD systems. I have been able to quickly deploy FreeBSD systems in my current position as a systems administrator for the Department of Mathematics & Statistics at MUN.

I have setup one server based on FreeBSD 9.1-RELEASE amd64 where I have installed FreeBSD 8.3-RELEASE amd64 in an NFS share and have booted three client machines off this system. Two as compute nodes using iSCSI targets for data store and to use one of the targets to implement a memory backed swap space. The remaining node I setup a local ZFS mirror to store backup data.

The same FreeBSD server has a second disk with ZFS installed and offers up NFS shares for various research groups and an iSCSI target for a dedicated RedHat compute node to store certain data sets.

Questions?