And Then There Was BOOTP

BOOTP, DHCP and IBM Power LPAR Builds

By David Bank/Copyright (c) 2019-2024 David Bank

While working in a mixed IBM Power9 and Intel x86_64 environment, I designed a solution for supporting simultaneous builds on both architectures. This required not just DHCP, but also **BOOTP**.

DHCP server configuration to support VMware *vs* PowerVM

My design goal was to create an environment in which I could could deliver SLES for SAP v15 hosts on both VMware and PowerPC platforms. For various reasons outside of the scope of this paper, the environment had a dedicated “build network” (described below); a separate VLAN and IP address space in which new LPARs and VMs were initially loaded. Also, and again for various reasons, I wasn’t deploying using images. The SUSE Manager (SUMA) host provided network-based installations, using the build network; and I was using the “Traditional”, instead of SALT-based, management.

If you live in the **World of Intel** (and never visit the fabled **PowerVM-land**), the usual approach starts with DHCP and PXE-booting, then loading just enough executable code to launch a Linux kernel with the parameters needed to kick off an installer. Sounds reasonably simple, right? Well, it is. SuSE even has some documentation about doing it with ARM and x86.

BUT….what if you’re on PowerPC under PowerVM (*aka* **PowerVM-land**)?

It turns out that PowerVM doesn't do DHCP - it only uses the older **BOOTP**.

Simultaneously supporting both architectures means supporting both protocols. I’ll get into the innards of the necessary server configuration in a moment; what I want to focus on now is why the difference is important.

Most network administrators don't like extraneous traffic, especially broadcasts, bouncing across their wires. In some situations, a protocol like DHCP is part of the production environment, used to assign IPs to Vagrant boxes on an as-needed basis, or for on-demand application containers *via* Docker, or similar functions. But plenty of other places, including the deployment environment I inhabited at the time, don't have those things.

So, if UDP/68 broadcasts wandering around the network aren't kosher, it is necessary to contain them. They were not in my environment; hence, the “build network” I mentioned above. This was a dedicated VLAN in the 192.168.0.0/24 address space. The SUSE Manager (SUMA) responsible for the builds had a network in both this network and the production network, but didn’t route between them.

The SUMA would normally want to control the DHCP install, but the configuration wouldn’t meet my needs. Similarly, the normal YaST-based configuration tool didn’t provide the level of control I needed. So I turned off the DHCP server functions inherent in the SUMA, and wrote my own configuration files.

```
# /etc/sycsconfig/dhcpd
#################################################################
# Define how the OS generally manages the DHCP daemon
#
# IMPORTANT! It does NOT define the configuration of dhcpd !!
#
# The daemon itself is configured via /etc/dhcpd.conf
#################################################################
# NOTES:
#     0) The main use of this file is to specify that dhcpd
#            only listens on bond1 (the "Build" network)
#       1) Support for IPv6 is disabled
#################################################################
# Change Log (Reverse Chronology)
# Who When______ What_____________________________________________
# dxb 2019-04-18 Initial creation
#################################################################
# Interface(s) where DHCPv4 server binds
# Specified as a space-separated list
# Special keyword "ANY" tells dhcpd to autodetect
#     active interfaces
# Examples: DHCPD_INTERFACE="eth0 eth1"
DHCPD_INTERFACE="bond1"

# Interface(s) where DHCPv6 server binds
# Uses same syntax as DHCPD_INTERFACE
DHCPD6_INTERFACE=""

# Should OS restart DHCPv4 server if interface to which it is
#     bound goes down and then comes back up?
# Valid Values:
#    no = do not restart dhcpd
#    yes = force a dhcp server restart
#    auto (DEFAULT) = restart for virtual interfaces
#            (bond,bridge,vlan) when all interfaces in
#            DHCPD_INTERFACE are up
DHCPD_IFUP_RESTART=""

# Should OS restart DHCPv4 server if interface to which it is
#     bound goes down and then comes back up?
# Uses same syntax as DHCPD_IFUP_RESTART
DHCPD6_IFUP_RESTART="no"

# Should DHCPv4 server run in chroot jail (/var/lib/dhcp)?
# Valid Values:
#    no = do NOT run dhcpd in a Jail
#    yes = (DEFAULT) Jail the dhcp server
# If set to "yes", then each time dhcpd is (re)started,
#     /etc/dhcpd.conf is copied to /var/lib/dhcp/etc/
# Also, other files that are important for hostname-to-IP
#     address resolution (/etc/{gai.conf,nsswitch.conf,
#     resolv.conf,host.conf,hosts,localtime},
#     /lib/lib{resolv.so.*,libnss_*.so.*, libpthread.so.0,
#     libdl.so.2}) are also copied to the chroot jail
# The PID file will be /var/lib/dhcp/var/run/dhcpd.pid
DHCPD_RUN_CHROOTED="yes"

# Should DHCPv6 server run in a chroot jail (/var/lib/dhcp6)?
# Uses same syntax as DHCPD_RUN_CHROOTED
# The file /etc/dhcpd6.conf is copied to the Jail; the PID
#     file will be in /var/lib/dhcp6/var/run/dhcpd.pid
DHCPD6_RUN_CHROOTED="yes"
```



```
# Additional configuration files/directories for DHCPv4
# Specified as a space-separated string of files and/or
#     directories (omit trailing /)
# Default: "/etc/dhcpd.d"
# NOTE: If the daemon is chroot-ed, then configurations
#     added by this directive are also copied to the Jail
DHCPD_CONF_INCLUDE_FILES="/etc/dhcpd.d"

# Additional configuration files/directories for DHCPv6
# Uses same syntax as DHCPD_CONF_INCLUDE_FILES
# Default: "/etc/dhcpd6.d"
DHCPD6_CONF_INCLUDE_FILES="/etc/dhcpd6.d"

# User ID under which DHCPv4 server runs
# Specified as a string that must be a valid ID
# Default: "root"
DHCPD_RUN_AS="dhcpd"

# User ID under which DHCPv6 server runs
# Uses same syntax/default as DHCPD_RUN_AS
DHCPD6_RUN_AS="dhcpd"

# Additional arguments for the DHCPv4 server command-line
# Specified as a string of valid command-line
#     options for dhcpd
# Default: Empty string
# Example: "-p 1234" for a non-standard port
DHCPD_OTHER_ARGS=""

# Additional arguments for the DHCPv6 server command-line
# Uses same syntax/default as DHCPD_OTHER_ARGS
DHCPD6_OTHER_ARGS=""

# End of /etc/sycsconfig/dhcpd
###############################
```

And also:

```
# /etc/dhcpd.conf
#####################################################################
# Main Configuration for DHCP daemon
#####################################################################
# IMPORTANT! This file interacts with /etc/sysconfig/dhcpd
#
# Also, if the DHCP server runs in a chroot Jail, then do NOT
#     edit the copy in /var/lib/dhcp – it will get overwritten
#####################################################################
# Change Log (Reverse Chronology)
# Who When_____ What_____________________________________________
# dxb 2019-04-18 Initial creation
#####################################################################
# Global Options #
##################
# These are definitions common to all supported networks
# DHCP Reply packets will report this host as the DNS server;
#     may also be a list
option domain-name-servers 192.168.1.10;

# Time, in seconds, for which an IP address is normally
#     leased; a client that complies with the standard
#     will request a re-lease when time expires
# Default: 600 (10 minutes)
default-lease-time 1800;
```



```
# The maximum time, in seconds, for which any lease is
#     valid; must be at least "default-lease-time"
# Default: 7200 (2 hours)
max-lease-time 7200;

# Enable/disable DDNS globally
ddns-updates off;
ddns-update-style none;

# Since the "build network" is (or is designed to be)
#     isolated, the "official" DHCP server should have
#     the "authoritative" directive
authoritative;

# Send DHCP log messages to syslog LOCAL7 (you may need to
#     configure rsyslog as well; choose any Facility you want)
log-facility local7;
#################################################################
# Define DHCP Option 93 ("client-system-arch") to indicate if
#     the client is 32- or 64-bit, and also EFI or BIOS; this
#     allows the client to pass information to the server
# See --> https://ipxe.org/cfg/platform and
#     http://www.ipamworldwide.com/ipam/isc-dhcpv4-options.html
option client-system-arch code 93 = unsigned integer 16;
#################################################################
# Define a client class that works based on the client self-ID
# This will help distinguish VMware-based hosts
class "pxeclients" {
    match if substring(option vendor-class-identifier, 0, 9) = "PXEClient";
       # Optionally, this can also be set using
       #     client-system-arch; for example, if all x86
       #     clients are PXE boot via DHCP
       # match if option client-system-arch = 00:07;
}
#################################################################
# The way the rest of this config works is that the first six
#     lines apply to all requests
# Then the first "pool" sets items specific to VMware clients
# The second "pool" is applied (by virtue of exclusion)
#     to PowerPC LPARs
subnet 192.168.1.0 netmask 255.255.255.0 {
       allow bootp;
       server-name "fqdn-of.dhcp.server";
       next-server 192.168.1.10;
       option routers 192.168.1.1;
       option broadcast-address 192.168.1.255;
       pool {
              allow members of "pxeclients";
              range 192.168.1.128 192.168.1.223;
              option domain-name-servers 192.168.1.10;
              filename "grub/x86_64-efi/grub.efi";
       }
    pool {
              deny members of "pxeclients";
              option domain-name-servers 192.168.1.10;
              range dynamic-bootp 192.168.1.32 192.168.1.63;
              range 192.168.1.64 192.168.1.127;
              filename "boot/grub2/powerpc-ieee1275/core.elf";
              # NOTE: This next option is not RFC-compliant!
              #     It "expires" BOOTP leases (which technically
              #     violates the RFC, but there will be problems
              #     in PowerVM-land without this)
              dynamic-bootp-lease-length 3600;
       }
}
# End of /etc/dhcpd.conf
########################
```


PowerVM, its use of BOOTP, and how that impacts building SLES for SAP

For building PowerPC LPARs, my system design relies on the **lpar_netboot** command, invoked from the Hardware Management Console (HMC). I'm getting a little ahead of myself, but there is a tangent relevant to the network component of my design; the command takes a lot of parameters... no, I mean a LOT:

```
lpar_netboot -v -f -D -m ADEB272D5102 -i -t ent -T off -s auto -d
auto -S 192.168.1.10 -G 192.168.1.1 -C 192.168.1.32 -K 255.255.255.0
"<hostname in HMC>" "<profile name in HMC>" "<managed system name>"
```

The parameter **-C 192.168.1.32** tells the LPAR to utilize **BOOTP** but sets the IP address; sharp-eyed readers will note that it falls in the scope of

```
range dynamic-bootp 192.168.1.32 192.168.1.63;
```

Once the LPAR boots, it sends out a **BOOTP** request (even though an IP was supplied); if the **BOOTP** reply (from the DHCP server) does not match the value of the **-C** parameter, then the boot sequence can fail (sometimes, but not always, with code `BA01B015`). Even after reaching the GRUB menu, the LPAR continues to rely on its **BOOTP**-associated IP address to retrieve, using TFTP, a Linux kernel. Once the kernel is loaded, the SLES installer is retrieved, using HTTP. Finally, the installer will use DHCP to configure an IP address (and the DHCP server should respond with the same address contained in the **BOOTP** reply).

The solution I've presented was designed for a deployment rate of one (1) LPAR *per* 60-minute period, which suited the needs of the environment. If the reader needs a faster rate, then by tweaking the configuration, the required time *per* LPAR is easily reduced to 15 minutes. With a little more effort, it can probably get closer to 5 minutes; part of the delay is an artifact of the **lpar_netboot** command, but by coordinating the commands with selected IP addresses, it would be possible to launch multiple builds at the same time. Care must be taken to avoid overloading the HMC CLI (I have not yet experimented with the REST interface).

This contrasts directly with the World of Intel. Since the DHCP server configuration uses a different address block for x86, and DHCP supports both lease expiration and booting without a pre-assigned IP, any deployment limitations occur more due to the network address space and capacity of the SUMA.

If you find any of these ideas applicable to your environment, you're welcome to use them.