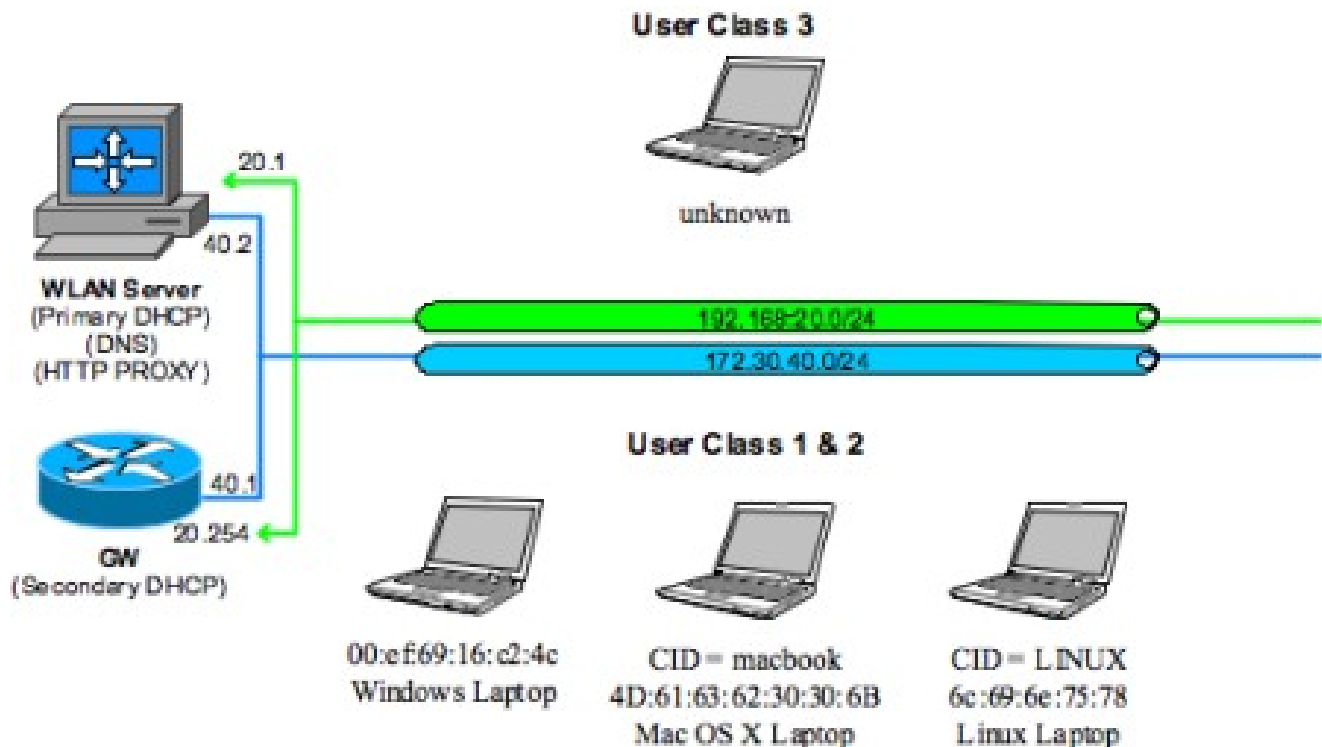


## Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

A second dhcpd.conf example may seem a little excessive. But this second example shows both the flexibility of the ISC DHCP solution and the ease of which network access control can be implemented using DHCP. The operational goal here was to implement the same degree of "network services" segmentation achieved in the first example using different SSID segments by using a combination of a single SSID, DHCP and application layer service control. On the DHCP side, this example utilizes the same hybrid DHCP lease model where some of the subnets are "open" and the others are closed. The example diagram below depicts four DHCP client users, which will make up a population of our three different user access classes.



Network service access, in this model, is enforced by utilizing two different IP subnets on the same "physical" segment, with IP forwarding between the two segments disabled using ACL's on the router and server. This provides the capability to handle the traffic for each of the service classes differently. How we do this is the subject for the future articles in this series. In order to support the two different IP subnets, we first need to configure secondary interfaces on the IOS routers and \*NIX WLAN server.

On IOS routers this is accomplished using the <ip address x.x.x.x. x.x.x.x secondary> interface command:

```
interface Ethernet0/0
ip address 172.30.40.1 255.255.255.0
ip address 192.168.20.254 255.255.255.0 secondary
ip access-group ap-wlan-in in
```

A relic from the days before 802.1q, the IP secondary command still comes in handy, especially when the need to readdress a network segments comes up. Secondary interfaces on \*NIX host's provide the ability to virtualize network services, such as Web or FTP servers. They are added simply using the ifconfig command:

```
root@trinity #ifconfig eth0 172.30.40.2 netmask 255.255.255.0 up
```

## Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

```
root@trinity #ifconfig eth0:1 192.168.20.1 netmask 255.255.255.0 up
root@trinity #
```

Secondary interfaces can be configured on both 802.1Q and non-Q \*NIX server interfaces. However, iptables filtering can only be applied to primary interfaces, so when implementing filtering rules the primary, not secondary interface need to be defined. Now let's move onto the DHCP configuration. The global declarations for this configuration are, for the most part, the same as the previous DHCP configuration:

```
ddns-updates off;
ddns-update-style none;
ping-check true;
one-lease-per-client true;
update-static-leases true;
deny declines;
deny duplicates;
deny bootp;
one-lease-per-client true;
log-facility local5;
option dhcp-server-identifier 172.30.40.1;
authoritative;
```

In the previous dhcpd.conf example a separate subnet declaration was configured for each IP segment. Here, both segments and three IP address pools are configured within the shared-network declaration, which is required because we are utilizing two different IP networks. What is also required is a way to distinguish which hosts are assigned to what pool. Otherwise the server just allocates leases out of the available pools. To properly direct the users a closed lease policy is used with one of the IP subnets and an open lease policy is used with the other -- a feat that is easily accomplished by using the <deny unknown clients> in the first subnets pool and group declarations:

```
shared-network SECWLAN {
# Global Options
option netbios-node-type 2;

# First Subnet
subnet 172.30.40.0 netmask 255.255.255.0 {

pool {
deny unknown clients;
range 172.30.40.20 172.30.40.25;
option routers 172.30.40.1;
option domain-name-servers 172.30.40.2;
option domain-name "secure.outlan.net";
default-lease-time 28800;
max-lease-time 28800;
option dhcp-message "Welcome to Outlan Secure! Access is restricted to
HTTP/HTTPS only.";

# Known Host Definitions for the first pool:
host guest1 {
hardware ethernet 00:ef:69:16:c2:4c;
}
}
```

## Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

```
} <-- End Of pool Statement

group {
deny unknown clients;
option routers 172.30.40.1;
option domain-name-servers 172.30.40.2;
option dhcp-message "Welcome to Outlan Admin, full network access is
available with proper credentials";

# Known Host Definitions for the second pool:

host book {
    fixed-address 172.30.40.78;
    option dhcp-client-identifier "\000happy";
}

host book {
    fixed-address 172.30.40.79;
    option dhcp-client-identifier "linux";
}
} <- End Of Group Statement
} <- End Of 1st Subnet Statement
```

Now, with the IP allocations of both pool and group definitions within the first subnet open only to known clients, all of the unknown client's, i.e., "HTTP Only users," lease requests fall to the second subnet declaration.

```
# Second Subnet
subnet 192.168.20.0 netmask 255.255.255.0 {
range 192.168.20.10 192.168.20.30;
option routers 192.168.20.1;
option domain-name "cell.outlan.net";
option domain-name-servers 192.168.20.254;
option dhcp-message "You are an unknown user, your access will be
limited. Please contact Administration @ ext 4090";
default-lease-time 1800;
max-lease-time 1800;
min-lease-time 300;
}
```

Granted this is not the most advanced network access control solution around, but it's quite effective with the added benefit of reducing the wireless RF/SSID complexity and the potential for expanded network range and improved performance. We will examine other potential benefits to this approach when we start to implement Web caching. So, now as we have come to the end of our long journey through the dhcpd.conf jungle, its time to look at controlling the wily dhcp daemon.

### Starting and Stopping DHCPD

The ISC DHCP can be launched from the command prompt (assuming you are using the default configuration by simply typing):

```
fido$ > dhcpd
```

## Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

To launch the service using our chroot cell configuration the command line is a little busier than the default, which requires the user and group and chroot cell to be defined along with the configuration file:

```
fido$ > /usr/local/dhcpd/sbin/dhcp-3.0.1rc11 -user dhcpd -group dhcpd
-chroot /usr/local/dhcpd -cf /usr/local/dhcpd/etc/dhcpd.conf
```

While launching from the command line is fine, it's far better to have the system launch the service as part of the boot process. Now, if DHCP was one of those "run and forget" types of applications we could just add the command line to the /etc/rc3.d/S99local file and it would be loaded at the end of the bootstrap process. However, with ISC DHCP it's not that easy; the process needs to be restarted each time changes are made to the dhcpd.conf file. And it's more than likely we will need to make changes to the dhcpd.confM file with regular frequency since we are running a hybrid lease policy. So to make our lives easier we need to build an rc script to drop into /etc/rc3.d that will let us start and stop the DHCP service with ease.

```
#!/bin/sh

PID="/usr/local/dhcpd/var/run/dhcpd.pid"
BIN="/usr/local/dhcpd/sbin/dhcp-3.0.1rc11"
DHOME="/usr/local/dhcpd"
DHCPD="$BIN -user dhcpd -group dhcpd -chroot $DHOME -cf
$DHOME/etc/dhcpd.conf"

# Source function library.
. /etc/init.d/functions

# Get config.
if [ -f /etc/sysconfig/network ]; then
    . /etc/sysconfig/network
else
    echo $"Networking not configured - exiting"
    exit 1
fi

# Check that networking is up.
if [ "$NETWORKING" = "no" ]; then
    exit 0
fi

prog=DHCP

RETVAL=0

start() {
    echo -n $"Starting $prog: "
    if [ -f $DHOME/etc/dhcpd.conf ]; then
        $DHCPD > /dev/null 2>&1
    else
        echo "No DHCPD config"
        exit 1
    fi
}
```

## Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

```
}  
  
stop() {  
  
echo -n $"Stopping $prog: "  
kill -9 `cat /usr/local/dhcpd/var/run/dhcpd.pid`  
}  
  
case "$1" in  
    start)  
        start  
        ;;  
    stop)  
        stop  
        ;;  
    *)  
        echo $"Usage: $0 {start|stop}"  
        exit 1  
esac  
  
echo -n $"OK"  
echo " "
```

Operation of the script is simple. To start the DHCP service from the command line, use the following:

```
fido $ >./S96dhcpd start  
Starting DHCP: OK  
fido $ >
```

To stop the DHCP service from the command line, use the following:

```
fido $ >./S96dhcpd stop  
Stopping DHCP: OK  
fido $ >
```

The script is based on the standard Red Hat rc scripts. It should be named S96dhcpd and installed in /etc/rc3.d. Once installed, it will be executed as part of the systems bootstrap process.

Now, with the standard control stuff out of the way, there is only one other problem we need to address. As DHCP is a critical path network service, we need to make sure that it stays up all of the time. The network is not much use if you can get an IP address. For this problem, we need to set up a CRON job that will run a check script. Now the S96dhcpd rc script starts and stops the service; it does not check to see if it's running; it assumes it's not. After all, it's a bootstrap script. Here is a DHCPd control script that lets you stop and start the service, but also checks to see if it's running and if so, does nothing, and if not starts the service -- perfect for running CRON checks:

```
#!/bin/sh  
  
touch /usr/local/dhcpd/var/tmp/pid.test  
touch /usr/local/dhcpd/var/run/dhcpd.pid  
  
PIDTEST="/usr/local/dhcpd/var/tmp/pid.test"  
PID="/usr/local/dhcpd/var/run/dhcpd.pid"
```

# Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

```
BIN="/usr/local/dhcpd/sbin/dhcp-3.0.1rc11"
```

```
DHOME="/usr/local/dhcpd"
```

```
# Restart DHCPD
```

```
if [ "$1" = "-k" ]
```

```
then
```

```
kill -9 `cat /usr/local/dhcpd/var/run/dhcpd.pid`;exit
```

```
fi
```

```
# Check to see if DHCPD is running, if not start it.
```

```
DHCP="$BIN -user dhcpd -group dhcpd -chroot $DHOME -cf
```

```
$DHOME/etc/dhcpd.conf"
```

```
ps -aux | grep `cat $PID` | awk '{print $2}' | sed '$d' > $PIDTEST
```

```
if [ `cat $PID` = `cat $PIDTEST` ]
```

```
then
```

```
echo "were running"
```

```
else
```

```
$DHCP
```

```
fi
```

Install this script in the same directory that dhcpd service daemon is, /usr/local/dhcpd/sbin, named dhcp-ctl.sh. To check if the DHCP service is running (or start the service), just run the script:

```
fido $ > ./dhcp-ctl.sh
```

To stop the DHCP service run the script with the "-k" flag:

```
fido $ > ./dhcp-ctl.sh -k
```

For best results the check script should be run every five minutes using CRON. Here is how you set it up. First open the crontab file for the user root:

```
fido $ > crontab -u root -e
```

This will run your default text editor (more than likely it will VI) and in most cases the crontab file will be blank, now add:

```
0,5,10,15,20,25,30,45,50,55 * * * * /usr/local/dhcpd/sbin/
```

```
dhcp-ctl.sh
```

```
:wq
```

```
crontab: installing new crontab
```

```
fido $ >
```

This configures CRON to run the /usr/local/dhcpd/sbin/dhcp-ctl.sh every five minutes. Once you save the file and exit the editor, CRON installs the crontab file and starts to process the job.

## Monitoring DHCPD

## Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

ISC DHCP stores information on its operation in two different files. Information on IP address bindings is stored in /usr/local/dhcpd/var/state/dhcpd/leases. Here is a sample:

```
# All times in this file are in UTC (GMT), not your local timezone. This is
# not a bug, so please don't ask about it. There is no portable way to
# store leases in the local timezone, so please don't request this as a
# feature. If this is inconvenient or confusing to you, we sincerely
# apologize. Seriously, though - don't ask.
# The format of this file is documented in the dhcpd.leases(5) manual page.
# This lease file was written by isc-dhcp-V3.0.1rc11

lease 172.30.40.25 {
    starts 0 2006/06/18 15:08:21;
    ends 0 2006/06/18 15:09:57;
    tstp 0 2006/06/18 15:09:57;
    binding state free;
    hardware ethernet 00:14:51:7e:3a:cd;
    uid "\000trinity";
}

lease 172.30.40.20 {
    starts 0 2006/07/09 03:40:48;
    ends 0 2006/07/09 04:29:50;
    tstp 0 2006/07/09 04:29:50;
    binding state free;
    hardware ethernet 00:16:cb:bb:0d:b8;
    uid "\001\000\026\313\273\015\270";
}

lease 172.30.40.24 {
    starts 1 2006/08/28 22:15:09;
    ends 1 2006/08/28 22:17:09;
    tstp 1 2006/08/28 22:17:09;
    binding state free;
    hardware ethernet 00:17:f2:40:97:6e;
    uid "\001\000\027\362@\227n";
    client-hostname "condor";
}
```

The ISC service cannot start without this file and if the file becomes corrupted or is deleted, all of the previously assigned lease information is lost. If you want to find out what host got what address, the dhcpd.leases file is the first place to check.

To see information on the operational status and DHCP event processing you need to check the log file. Unless specified, DHCPd will send log data to the systems standard log file /var/messages. To set up an individual log file for DHCPd, start off by defining a SYSLOG facility in the dhcpd.conf file.

```
log-facility local5;
```

Once the facility has been defined, the SYSLOG daemon needs to be configured to send any messages sent to the log facility (in this case local5) we have defined in dhcpd.conf. The SYSLOGd configuration file is located in /etc. For our purposes we want SYSLOGd to send log messages to /usr/local/dhcpd/var/log/dhcp.log. Before we edit /etc/syslog.conf we need to create the log file (and the directories where it's stored):

## Building A WLAN Proxy Server - DHCP Services Part 2

Michael J. Martin

```
fido $ > mkdir /usr/local/dhcpd/var/log/
fido $ > touch /usr/local/dhcpd/var/log/dhcp.log
fido $ > chown dhcpd /usr/local/dhcpd/var/log/dhcp.log
fido $ > chgrp dhcpd /usr/local/dhcpd/var/log/dhcp.log
```

With the directories built we can edit /etc/syslog.conf:

```
fido $ > vi syslog.conf
```

```
# logfile for dhcpd
local5.*                /usr/local/dhcpd/var/log/dhcp.log
```

```
fido $ >
```

Once /etc/syslog.conf has been edited, we need to restart the syslogd service using the rc file S12syslog bootstrap script:

```
fido $> ./S12syslog restart
Shutting down kernel logger:      [ OK ]
Shutting down system logger:     [ OK ]
Starting system logger:          [ OK ]
Starting kernel logger:          [ OK ]
fido $ >
```

Once the logfile has been configured we can monitor the servers operation and DHCP event processing using the tail command with the "-f" operator flag:

```
fido $ > tail -f /usr/local/dhcpd/var/log/dhcp.log
Apr 17 10:54:51 localhost dhcpd: DHCPREQUEST for 172.30.80.205 from
00:14:51:7e:3a:cd (trinity) via eth0.80
Apr 17 10:54:51 localhost dhcpd: DHCPACK on 172.30.80.205 to
00:14:51:7e:3a:cd (trinity) via eth0.80
Apr 17 10:57:46 localhost dhcpd: DHCPREQUEST for 172.30.80.205 from
00:14:51:7e:3a:cd (trinity) via eth0.80
Apr 17 10:57:46 localhost dhcpd: DHCPACK on 172.30.80.205 to
00:14:51:7e:3a:cd (trinity) via eth0.80
```