DHCP: Unexplored Capacities

EuroBSDCon 2004 - Karlsruhe

Diablotin - Tous droits réservés

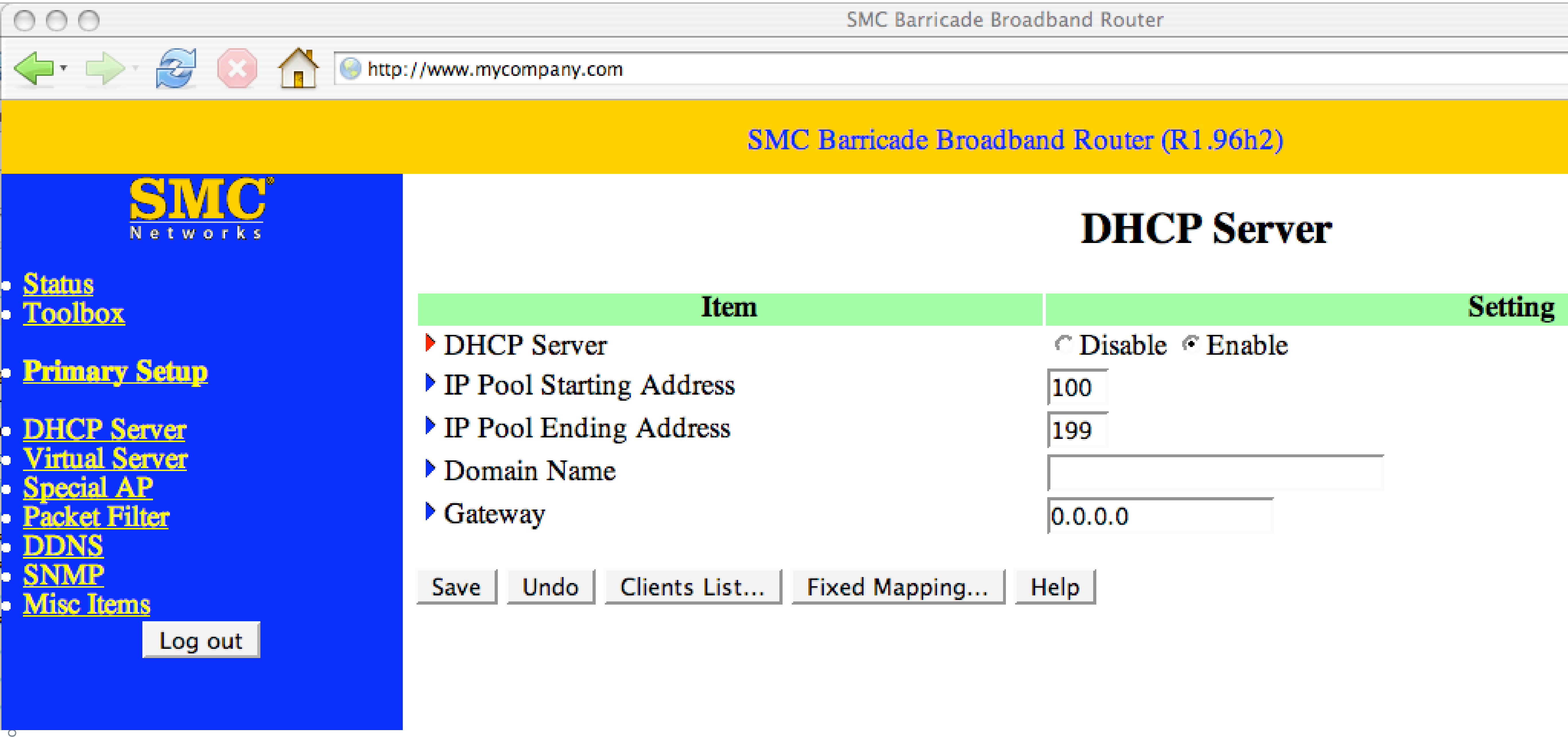
No current SOHO router comes without DHCP.

Within a few years, DHCP became one of the most popular network protocols.

An insidious side effect of the sudden widespreading of an embedded version of DHCP is that many administrators never realized that this service offers many resources, most of which remain unused. DHCP is also prone to evolve and be extended.

There would be a lot to say about client-side customization, communication between DHCP servers and dedicated directories like DNS, or even specific uses by some operating systems.

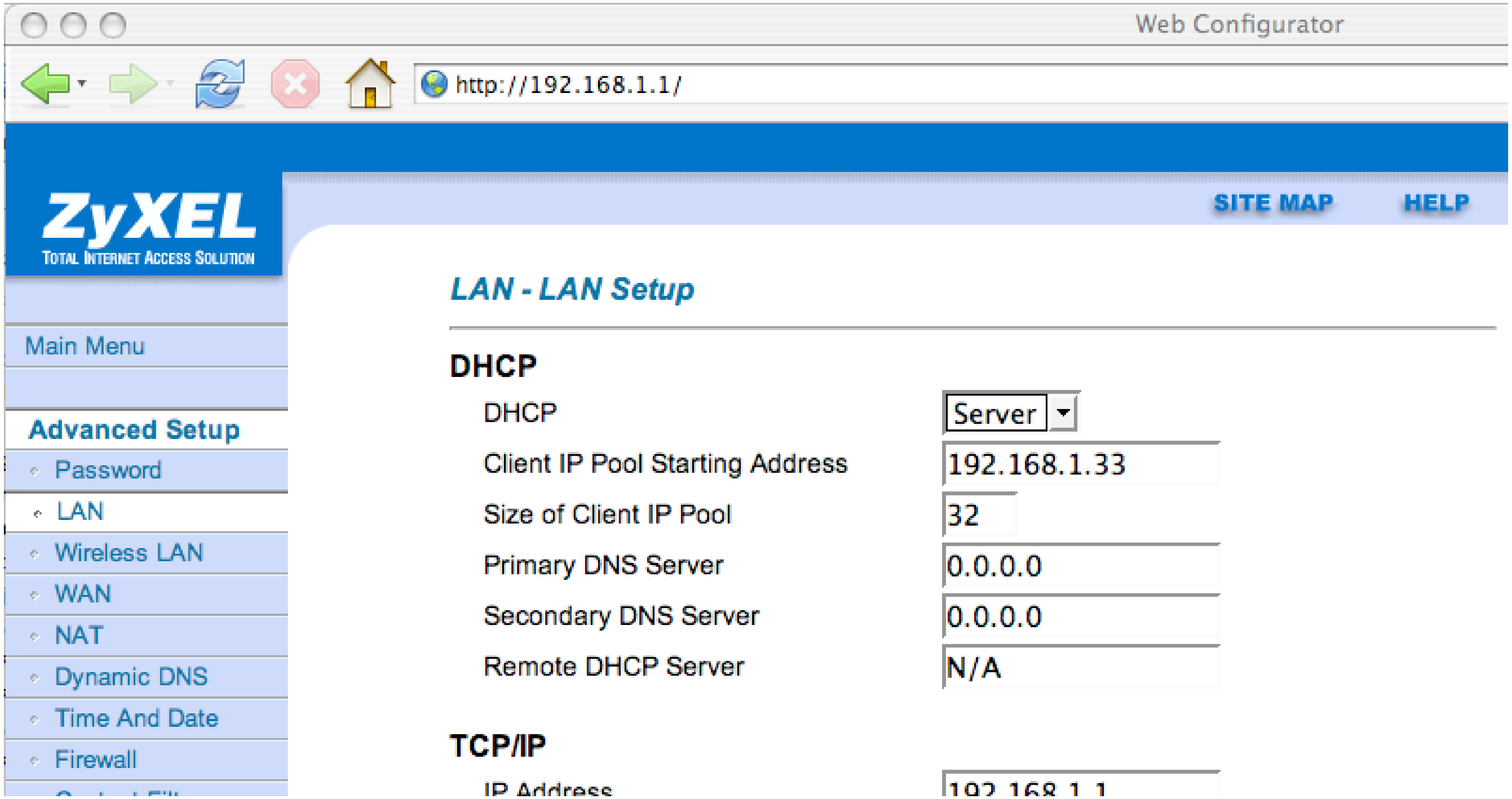
Maybe for you a DHCP server is something like this...



DHCP Server

Item	Setting
DHCP Server	○ Disable ● Enable
IP Pool Starting Address	100
IP Pool Ending Address	199
Domain Name	
Gateway	0.0.0.0

or...



LAN - LAN Setup

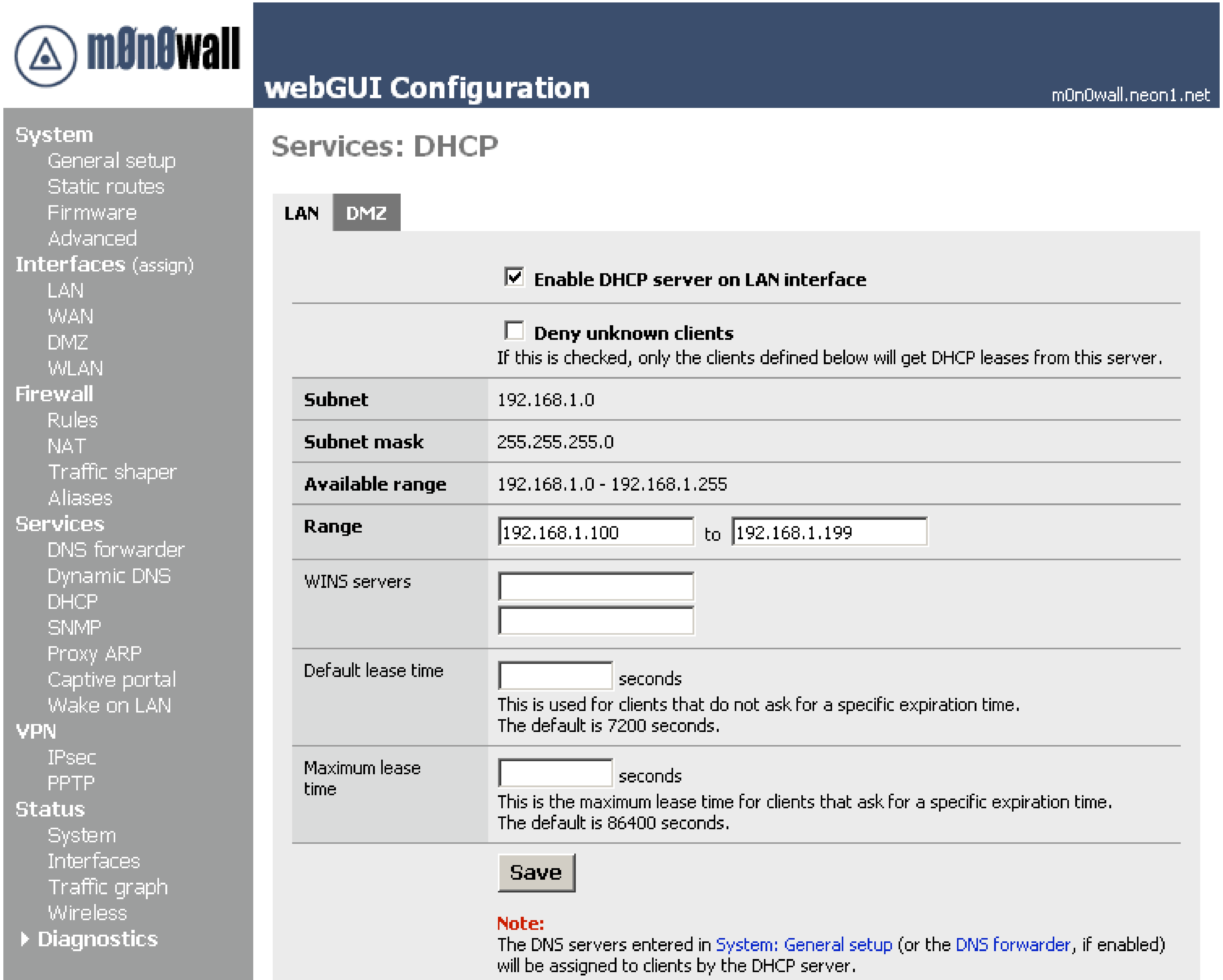
DHCP

DHCP	Server
Client IP Pool Starting Address	192.168.1.33
Size of Client IP Pool	32
Primary DNS Server	0.0.0.0
Secondary DNS Server	0.0.0.0
Remote DHCP Server	N/A

TCP/IP

IP Address	192.168.1.1

or, in the best case…



Services: DHCP

☑ **Enable DHCP server on LAN interface**

□ **Deny unknown clients**
If this is checked, only the clients defined below will get DHCP leases from this server.

Subnet	192.168.1.0
Subnet mask	255.255.255.0
Available range	192.168.1.0 - 192.168.1.255
Range	192.168.1.100 to 192.168.1.199
WINS servers	
Default lease time	seconds This is used for clients that do not ask for a specific expiration time. The default is 7200 seconds.
Maximum lease time	seconds This is the maximum lease time for clients that ask for a specific expiration time. The default is 86400 seconds.

Note:
The DNS servers entered in System: General setup (or the DNS forwarder, if enabled) will be assigned to clients by the DHCP server.

But DHCP is much more!

With a DHCP server you can…

- … Broadcast many administrative and also generic informations
- … Send predefined and also custom informations
- … Talk with DNS servers
- … Benefit from High Availability (HA)
- … Conditionally execute actions
- … Use script language expressions
- … Remote monitor and control
- …

Prejudices

- DHCP is a directory that broadcasts a finite number of pieces of information
  - One can add to DHCP data that were not planned for in the beginning or even in revisions, in a very official manner, what is more
- DHCP offers simple mechanisms which require only little configuration and attention
  - A network can rapidly find itself in a delicate situation if DHCP is managed in an anarchic manner, or if it is simply improperly configured

Prejudices (2)

- DHCP mostly serves to allocate IP addresses to hosts, as well as a few other closely related data
  - DHCP can very well deliver tens of important data to a host WITHOUT offering it any IP address (DHCP is used as a directory -of course a very specialized one- but a directory nevertheless, able to offer a very important number of pieces of information
- DHCP will be cast into the shadows by the deployment of IPv6
  - Several companies and organizations are working on DHCPv6

Prejudices (3)

- DHCP only works on local networks
    - That is both true and false. Based on UDP and making massive use of broadcasting, the DHCP protocol cannot pass through routers. Nevertheless, since the very beginning (as soon as the time of its ancestor BOOTP), a relay system known as DHCP relay which has to be installed in each subnetwork makes communication possible between centralized servers and clients spread over several subnetworks, even beyond routers
- DHCP's characteristics hardly change
    - More than 30 RFCs since the beginning in late 1993

Prejudices (4)

- Only clients can come and ask for their information to be put up to date by a server
  - Should become false any time soon when a new "message" is introduced into DHCP: FORCERENEW (RFC 3203 - DHCP Reconfigure Extension)
- DHCP doesn't allow to send information bigger than 255 bytes
  - About to become obsolete as RFC 3396 (Nov 2002) spreads out. Encoding Long Options in the Dynamic Host Configuration Protocol explains how a datum can be split into several pieces, to be later reassembled by the addressee (more or less as fragmentation does in the IP context)

Prejudices (5)

- DHCP implements reliable methods in case an IP address it wants to allocate has already been 'stolen' by a host on the current network subsection
  - Not really! The DHCP server uses a mere ping to ascertain the presence of a host which already uses the IP address it wants to send to his client. What is more, it awaits an answer within one second. If the spoofer host does not answer pings -that would be a critic-prone configuration of its firewall- or answers too slowly, the DHCP server can wrongly assume that the address is really free and assign it to its client
  - ping-check true|false and ping-timeout delay_in_seconds directives can be used to somewhat alter this behavior (but that doesn't make the DHCP server's check any more 'reliable')

Options (RFC 2132)

Of course you know a DHCP server delivers:

- Client IP address
- Subnet mask
- Broadcast address
- Routers (used as the default gateway by the client)
- Domain name
- DNS servers

Options (RFC 2132)

BUT do you know that it can deliver as well:

- Time Server
- Log Server
- Cookie Server
- LPR Server
- Impress Server
- Resource Location Server
- IP Forwarding Enable/Disable
- Merit Dump File

- Swap Server
- Root Path
- Extensions Path
- Non-Local Source Routing Enable/Disable
- Policy Filter
- Maximum Datagram Reassembly Size
- Default IP Time-to-live
- Path MTU Aging Timeout
- Path MTU Plateau Table
- Interface MTU
- All Subnets are Local
- Perform Mask Discovery

- Mask Supplier, Perform Router Discovery,
- Router Solicitation Address, Static Route
- Trailer Encapsulation, ARP Cache Timeout
- Ethernet Encapsulation, TCP Default TTL
- TCP Keepalive Interval, TCP Keepalive Garbage
- Network Information Servers (NIS), NIS Domain
- Network Time Protocol Servers
- Vendor Specific Information
- NetBIOS over TCP/IP (NBT) Name Server
- NBT Datagram Distribution Server, NBT Node Type, NBT Scope
- X Window Font Server, X Window Display Manager
- Network Information Service+ (NIS+), NIS+ Domain

- Mobile IP Home Agent
- Simple Mail Transport Protocol (SMTP) Server
- Post Office Protocol (POP3) Server
- Network News Transport Protocol (NNTP) Server
- Default World Wide Web (WWW) Server, Default Finger Server
- Default Internet Relay Chat (IRC) Server
- StreetTalk Server, StreetTalk Directory Assistance (STDA) Server
- TFTP server name, Bootfile name, Boot File Size
- Server Identifier
- Vendor class identifier
- Client-identifier
- … And few others

Options to recover (RFC 3679)

- Service Location Protocol Naming Authority
  Reason: Never published as standard and not in general use
- Relay Agent Options
  Reason: Not defined in RFC 3046 as published
- IEEE 1003.1 POSIX Timezone
  Reason: Never published as standard and not in general use
- FQDNs in DHCP Options
  Reason: Never published as standard and not in general use
- VINES TCP/IP Server
  Reason: Never published as Internet-Draft
- Server Selection
  Reason: Never published as Internet-Draft

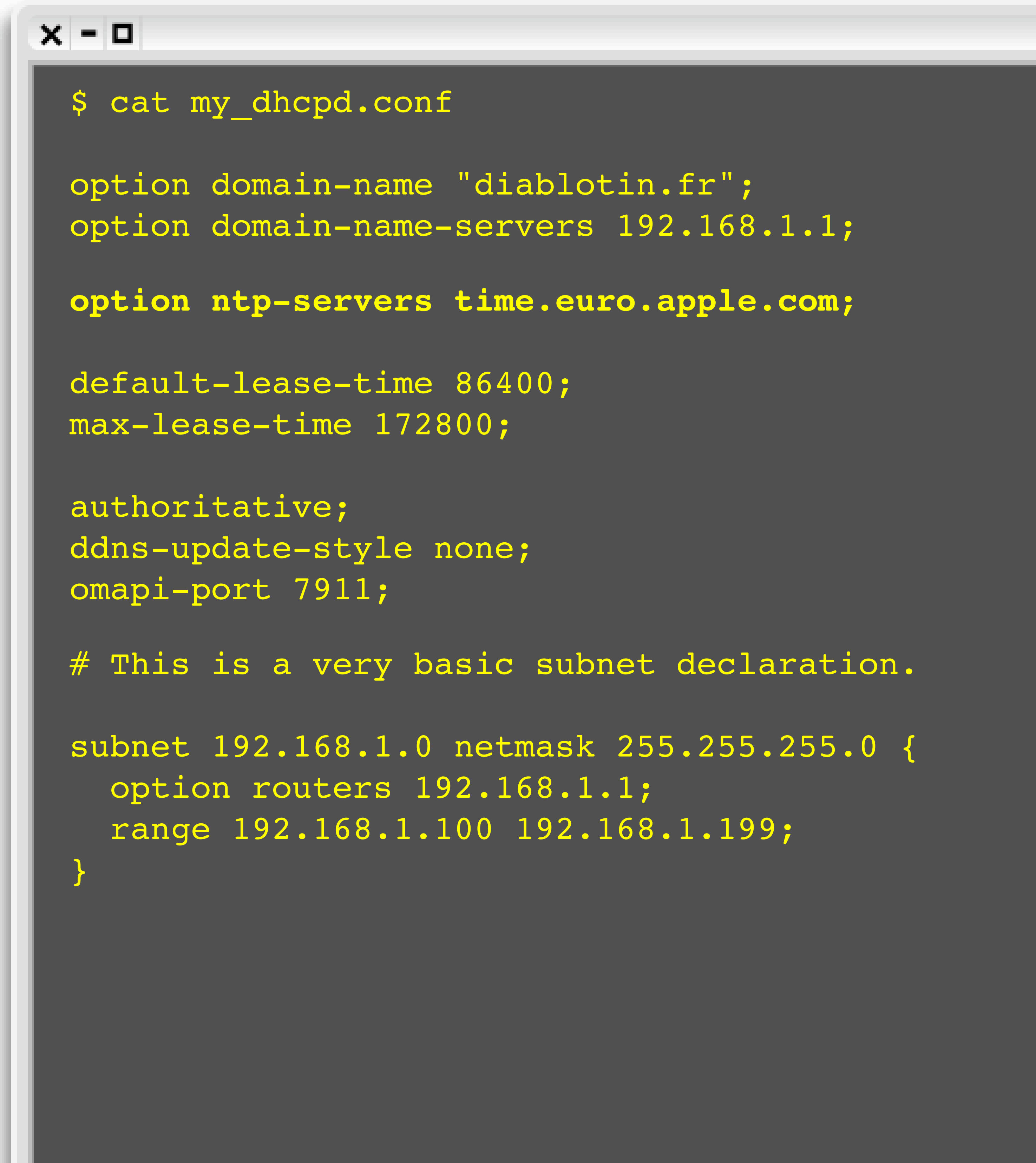
Options to recover (RFC 3679)

- IPv6 Transition
  Reason: Never published as standard and not in general use
- Printer Name
  Reason: Never published as Internet-Draft
- Multicast Assignment through DHCP
  Reason: Never published as standard and not in general use
- Swap Path
  Reason: Never published as Internet-Draft
- IPX Compatibility
  Reason: Never published as Internet-Draft
- Failover
  Reason: Current version of "DHCP Failover Protocol" does not use a DHCP option

Options

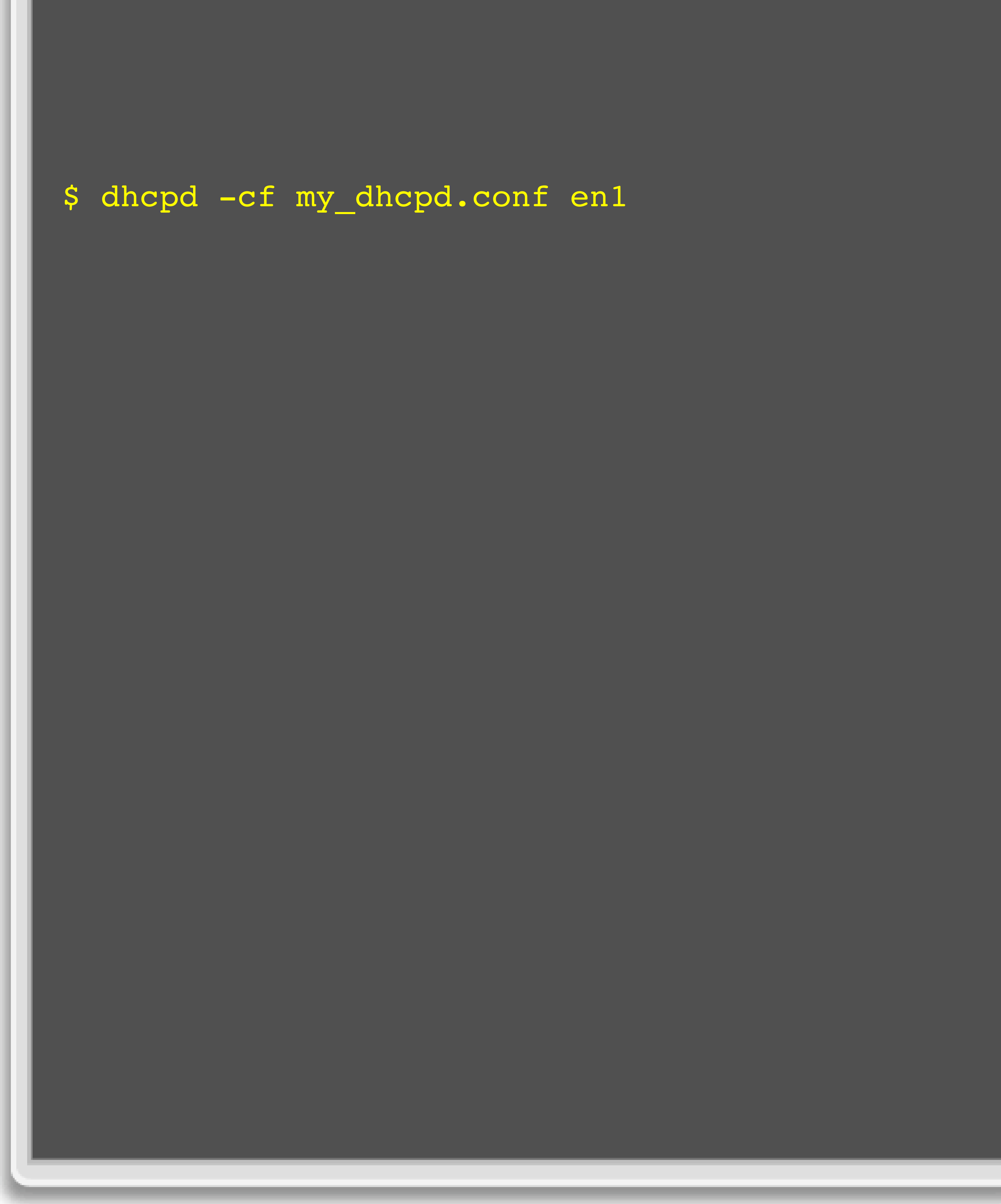
Let's practice…

Use option to get NTP server(s) — the server side



```
$ cat my_dhcpd.conf

option domain-name "diablotin.fr";
option domain-name-servers 192.168.1.1;

option ntp-servers time.euro.apple.com;

default-lease-time 86400;
max-lease-time 172800;

authoritative;
ddns-update-style none;
omapi-port 7911;

# This is a very basic subnet declaration.

subnet 192.168.1.0 netmask 255.255.255.0 {
  option routers 192.168.1.1;
  range 192.168.1.100 192.168.1.199;
}
```

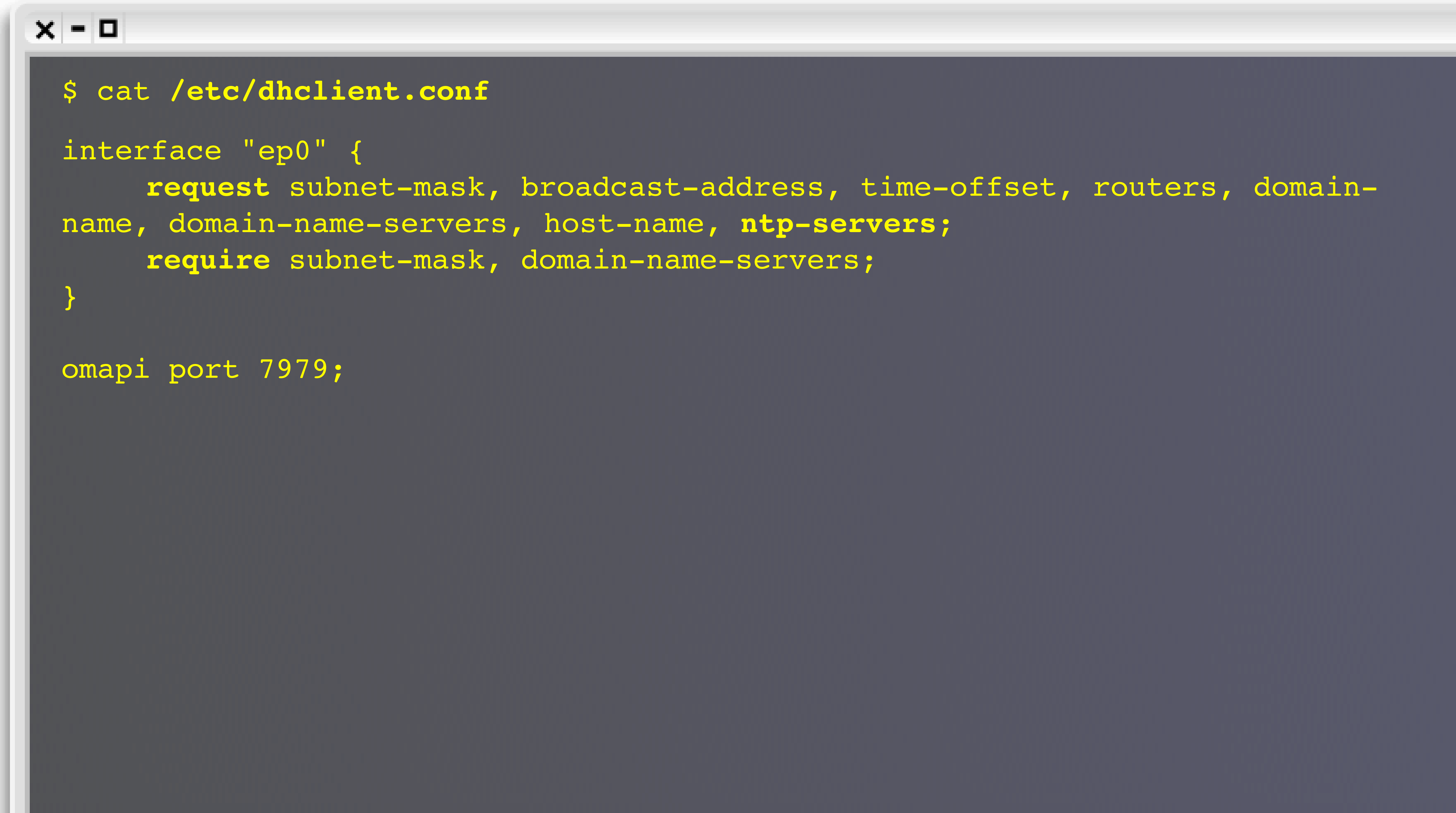
Server side



```
$ dhcpd -cf my_dhcpd.conf en1
```

Use option to get NTP server(s) — the client side

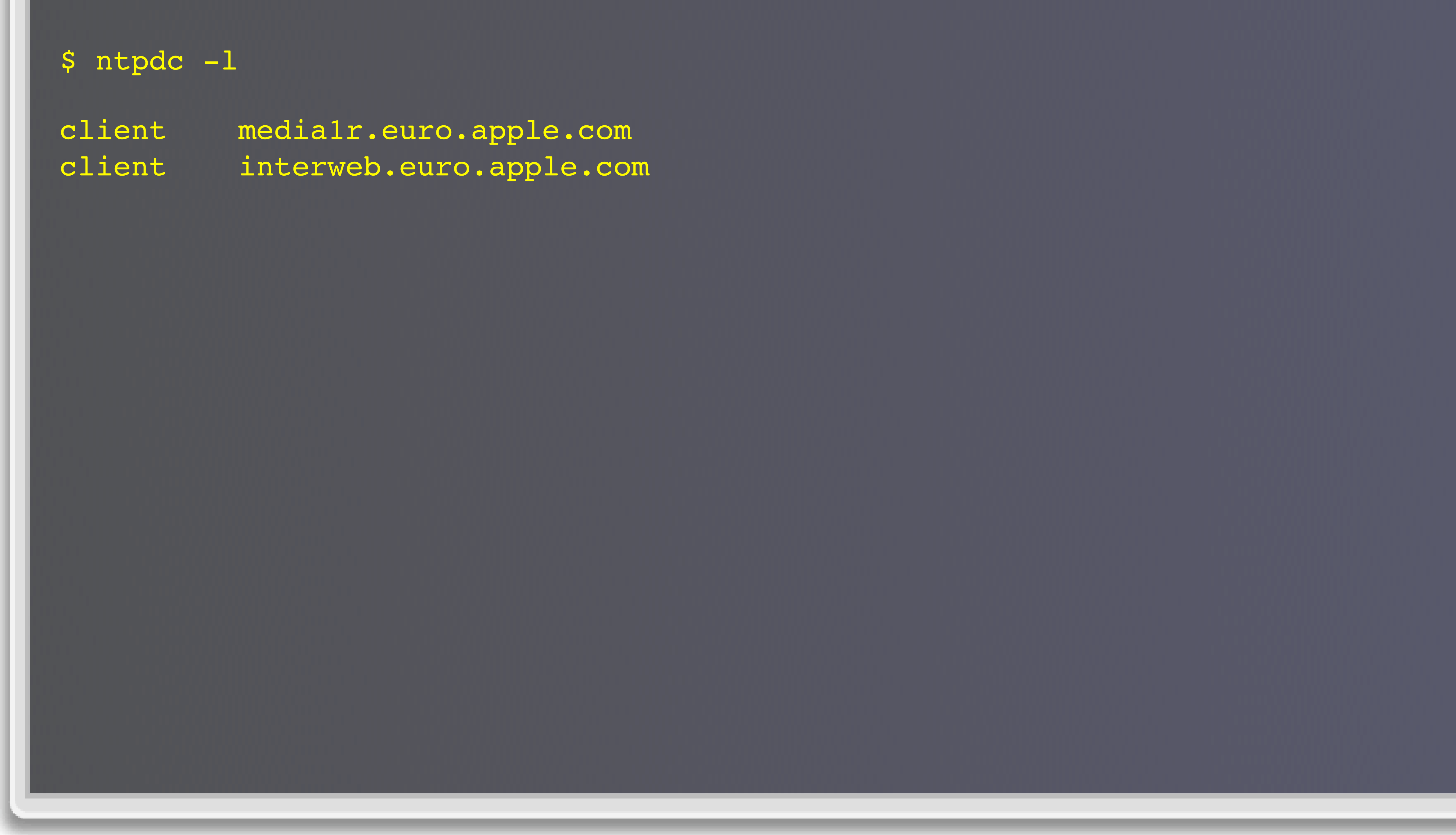
The server may or may not be configured to broadcast some of this information, and as a last resort, it's up to the DCHP client to request OR demand (a notion known as require) the data it needs



```
$ cat /etc/dhclient.conf

interface "ep0" {
     request subnet-mask, broadcast-address, time-offset, routers, domain-
name, domain-name-servers, host-name, ntp-servers;
     require subnet-mask, domain-name-servers;
}

omapi port 7979;
```

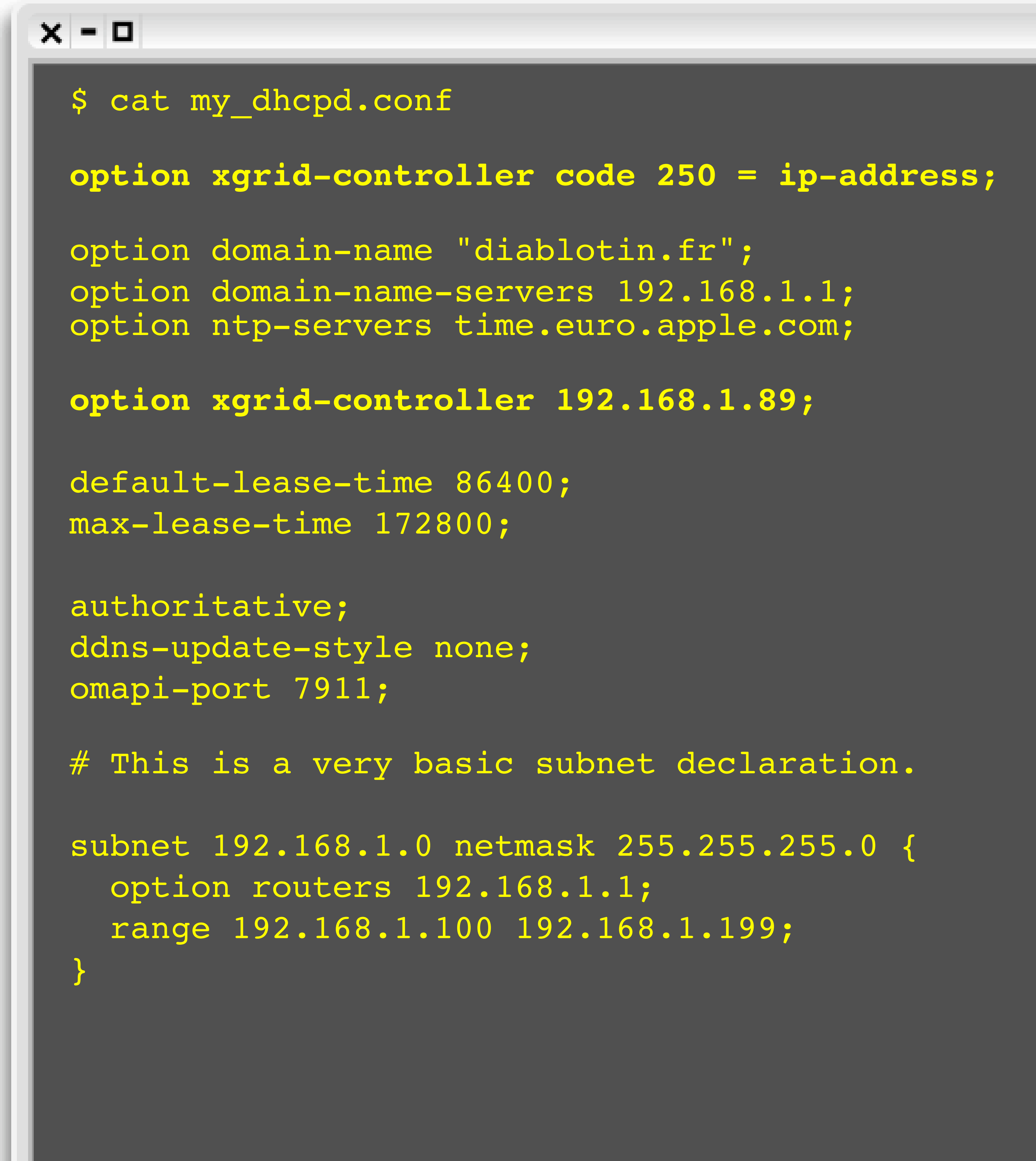
```
$ cat /etc/dhclient-exit-hooks

if [ x"$new_ntp_servers" != x ]; then
  if [ -f /etc/ntp.conf ]; then
    ( rm /etc/ntp.conf )
    exit_status=$?
  else
    ( touch /etc/ntp.conf )
    exit_status=$?
  fi
  if [ $exit_status -ne 0 ]; then
    $LOGGER "WARNING: Unable to update ntp.conf: Error $exit_status"
  else
    for ntpserver in $new_ntp_servers; do
      ( echo server $ntpserver >>/etc/ntp.conf )
    done
    ntpd -p /var/run/ntpd.pid
    echo "dhclient hooks: New Time Servers: "$new_ntp_servers
  fi
fi
```

```
$ dhclient

Internet Systems Consortium DHCP Client V3.0.1
Copyright 2004 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/products/DHCP

Listening on BPF/ep0/00:a0:24:66:29:7d
Sending on   BPF/ep0/00:a0:24:66:29:7d
Sending on   Socket/fallback
DHCPDISCOVER on ep0 to 255.255.255.255 port 67 interval 8
DHCPOFFER from 192.168.1.35
DHCPREQUEST on ep0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.35
dhclient: New Network Number: 192.168.1.0
dhclient: New Broadcast Address: 192.168.1.255
dhclient: New IP Address (ep0): 192.168.1.199
dhclient: New Subnet Mask (ep0): 255.255.255.0
dhclient: New Broadcast Address (ep0): 192.168.1.255
dhclient: New Routers: 192.168.1.1
dhclient hooks: New Time Servers: 17.72.133.42 17.72.133.45
bound to 192.168.1.199 -- renewal in 36086 seconds.
```



```
$ ntpdc -l

client    media1r.euro.apple.com
client    interweb.euro.apple.com
```

Create a user-defined option — the server side

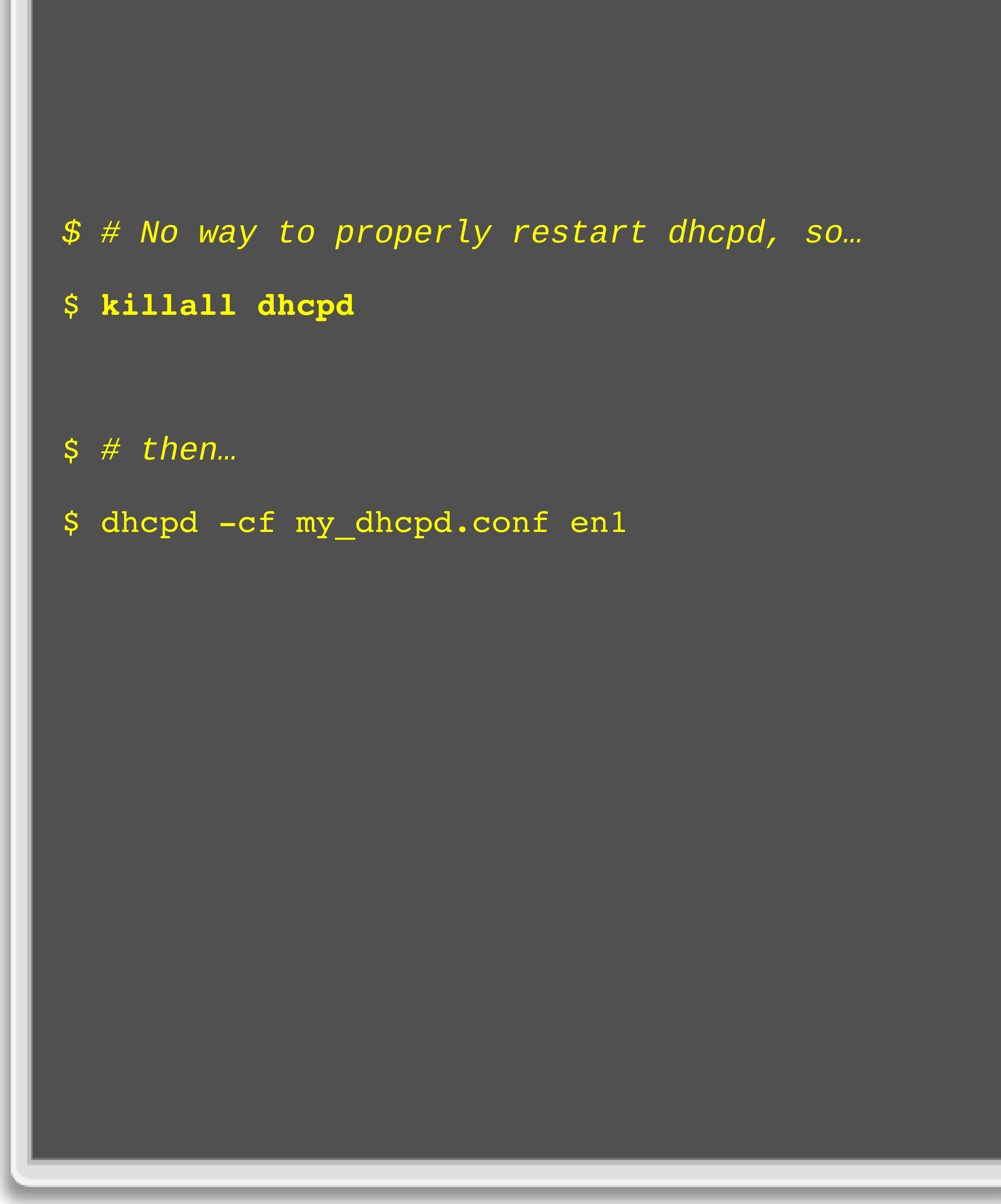


```
$ cat my_dhcpd.conf

option xgrid-controller code 250 = ip-address;

option domain-name "diablotin.fr";
option domain-name-servers 192.168.1.1;
option ntp-servers time.euro.apple.com;

option xgrid-controller 192.168.1.89;

default-lease-time 86400;
max-lease-time 172800;

authoritative;
ddns-update-style none;
omapi-port 7911;

# This is a very basic subnet declaration.

subnet 192.168.1.0 netmask 255.255.255.0 {
  option routers 192.168.1.1;
  range 192.168.1.100 192.168.1.199;
}
```

Types for custom options

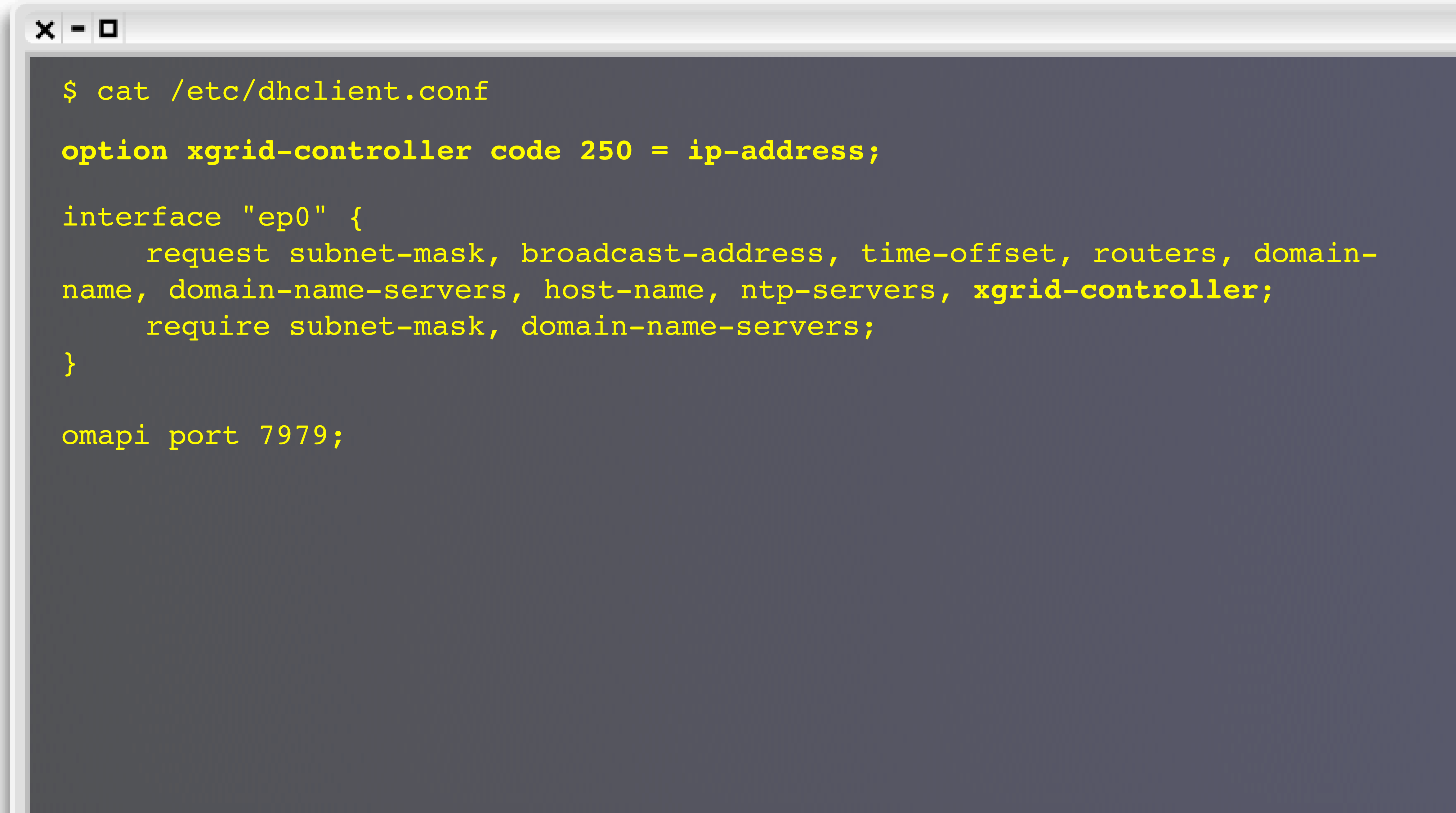
- ip-address
- [unsigned] integer [8|16|32]
- boolean (true/false or on/off)
- text (NVT ASCII string)
- string (same as text but can also be expressed as octets specified in hexadecimal, separated by colons)
- record (mix of above types)
- array (any of the above types or record except for the text and data string types - you can not specify range)

Server side



```
$ # No way to properly restart dhcpd, so…
$ killall dhcpd

$ # then…
$ dhcpd -cf my_dhcpd.conf en1
```

Get a user-defined option — the client side

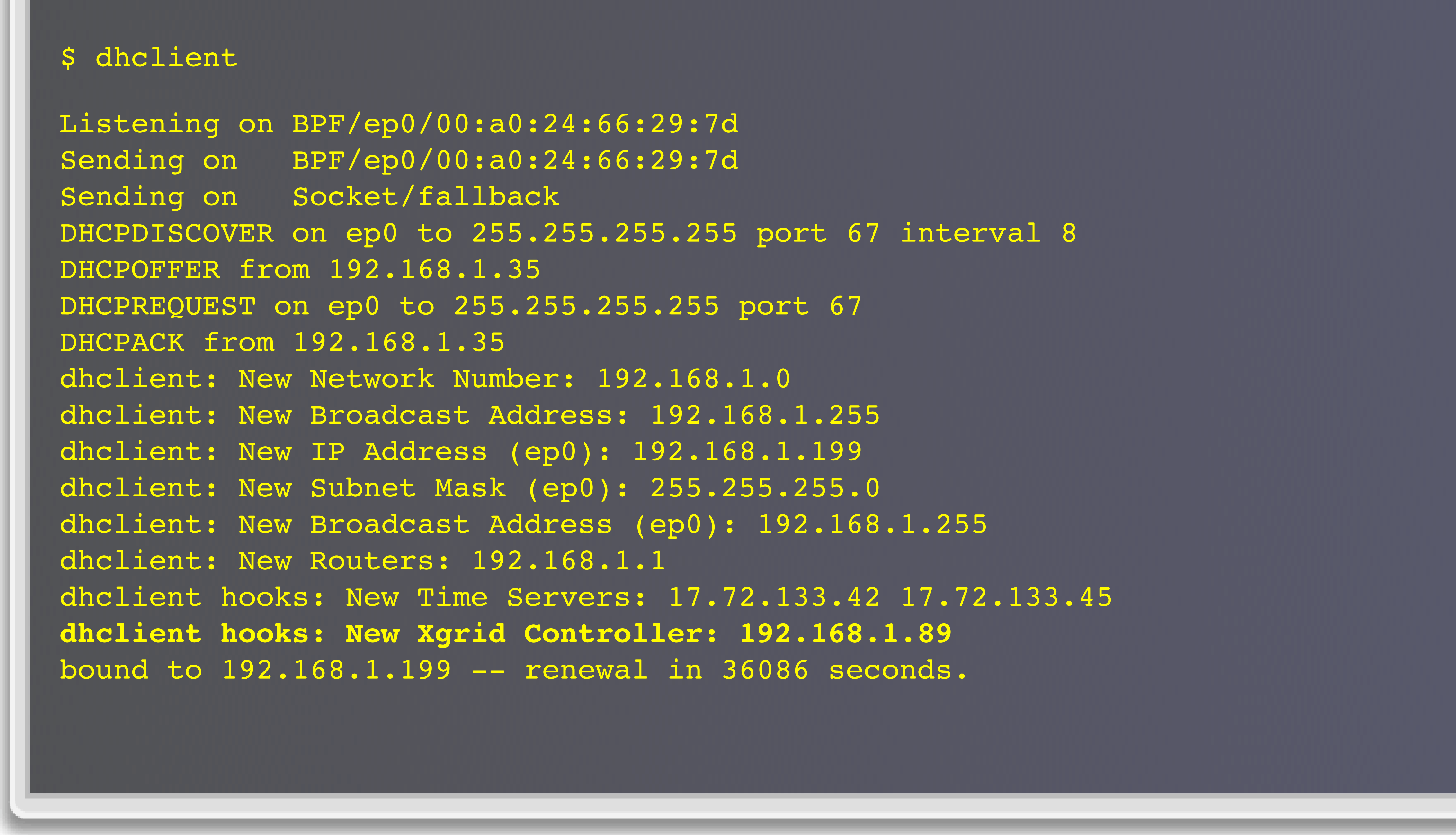


```
$ cat /etc/dhclient.conf

option xgrid-controller code 250 = ip-address;

interface "ep0" {
     request subnet-mask, broadcast-address, time-offset, routers, domain-
name, domain-name-servers, host-name, ntp-servers, xgrid-controller;
     require subnet-mask, domain-name-servers;
}

omapi port 7979;
```

User-defined option

- Excerpt from RFC2939:
  - "The DHCP option number space (1-254) is split into two parts. The site-specific option codes (128-254) are defined as "Private Use" and require no review by the DHC WG (Dynamic Host Configuration Working Group). Site-specific options codes (128-254) MUST NOT be defined for use by any publicly distributed DHCP server, client or relay agent implementations. These option codes are explicitly reserved for private definition and use within a site or an organization. The public option codes (0-127, 255) are defined as "Specification Required" and new options must be reviewed prior to assignment of an option number by IANA."

```
$ cat /etc/dhclient-exit-hooks

if [ x"$new_ntp_servers" != x ]; then
  if [ -f /etc/ntp.conf ]; then
    ( rm /etc/ntp.conf )
    exit_status=$?
  else
    ( touch /etc/ntp.conf )
    exit_status=$?
  fi
  if [ $exit_status -ne 0 ]; then
    $LOGGER "WARNING: Unable to update ntp.conf: Error $exit_status"
  else
    for ntpserver in $new_ntp_servers; do
      ( echo server $ntpserver >>/etc/ntp.conf )
    done
    ntpd -p /var/run/ntpd.pid
    echo "dhclient hooks: New Time Servers: "$new_ntp_servers
  fi
fi

if [ x"$new_xgrid_controller" != x ]; then
    echo "dhclient hooks: New Xgrid Controller: "$new_xgrid_controller
fi
```



```
$ dhclient

Listening on BPF/ep0/00:a0:24:66:29:7d
Sending on   BPF/ep0/00:a0:24:66:29:7d
Sending on   Socket/fallback
DHCPDISCOVER on ep0 to 255.255.255.255 port 67 interval 8
DHCPOFFER from 192.168.1.35
DHCPREQUEST on ep0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.35
dhclient: New Network Number: 192.168.1.0
dhclient: New Broadcast Address: 192.168.1.255
dhclient: New IP Address (ep0): 192.168.1.199
dhclient: New Subnet Mask (ep0): 255.255.255.0
dhclient: New Broadcast Address (ep0): 192.168.1.255
dhclient: New Routers: 192.168.1.1
dhclient hooks: New Time Servers: 17.72.133.42 17.72.133.45
dhclient hooks: New Xgrid Controller: 192.168.1.89
bound to 192.168.1.199 -- renewal in 36086 seconds.
```

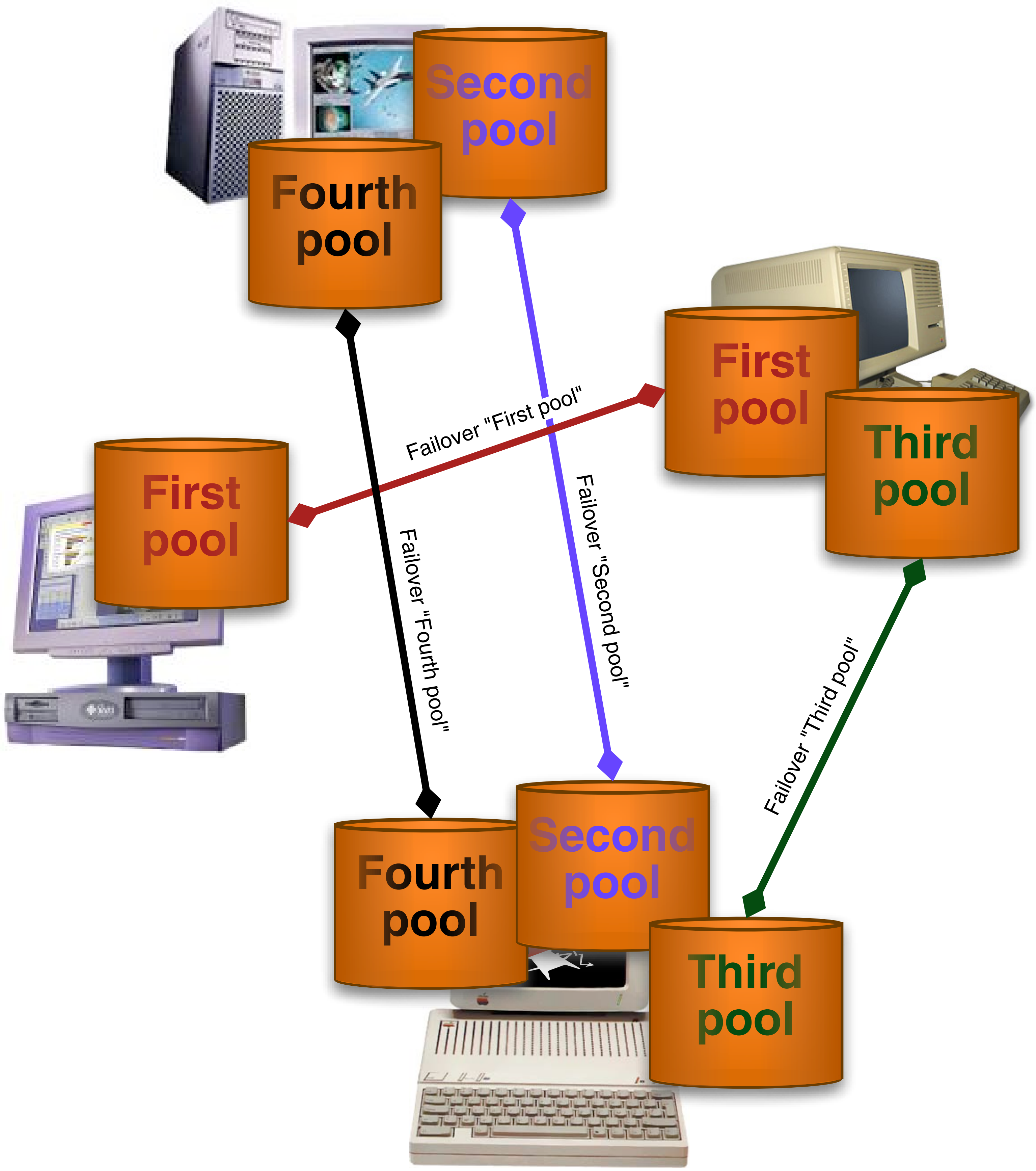
Service continuity
(High Availability)

- Failover appeared lately in the ISC DHCP server
    - It is essentially a backend problem
      (how to share the lease file between 2 servers?)
- Before this, there are only some tricks:
    - Use static addresses
      (in this case no need to keep addresses in a lease file)
      Important: consistence of information on both servers
    - Offer nonoverlapping IP address pools from 2 servers
    - You can of course mix static and dynamic addresses

Failover

- Not managed on a server basis but on address pool
- Only 2 servers for each pool
  (maybe different for each pool)
- In a primary-secondary configuration
  - either WITH load-balancing
  - or WITHOUT load-balancing
    - actually, a special case of load-balancing where primary server load is configured at 100 % activity
  - Automatic spreading of the pools of allocatable addresses

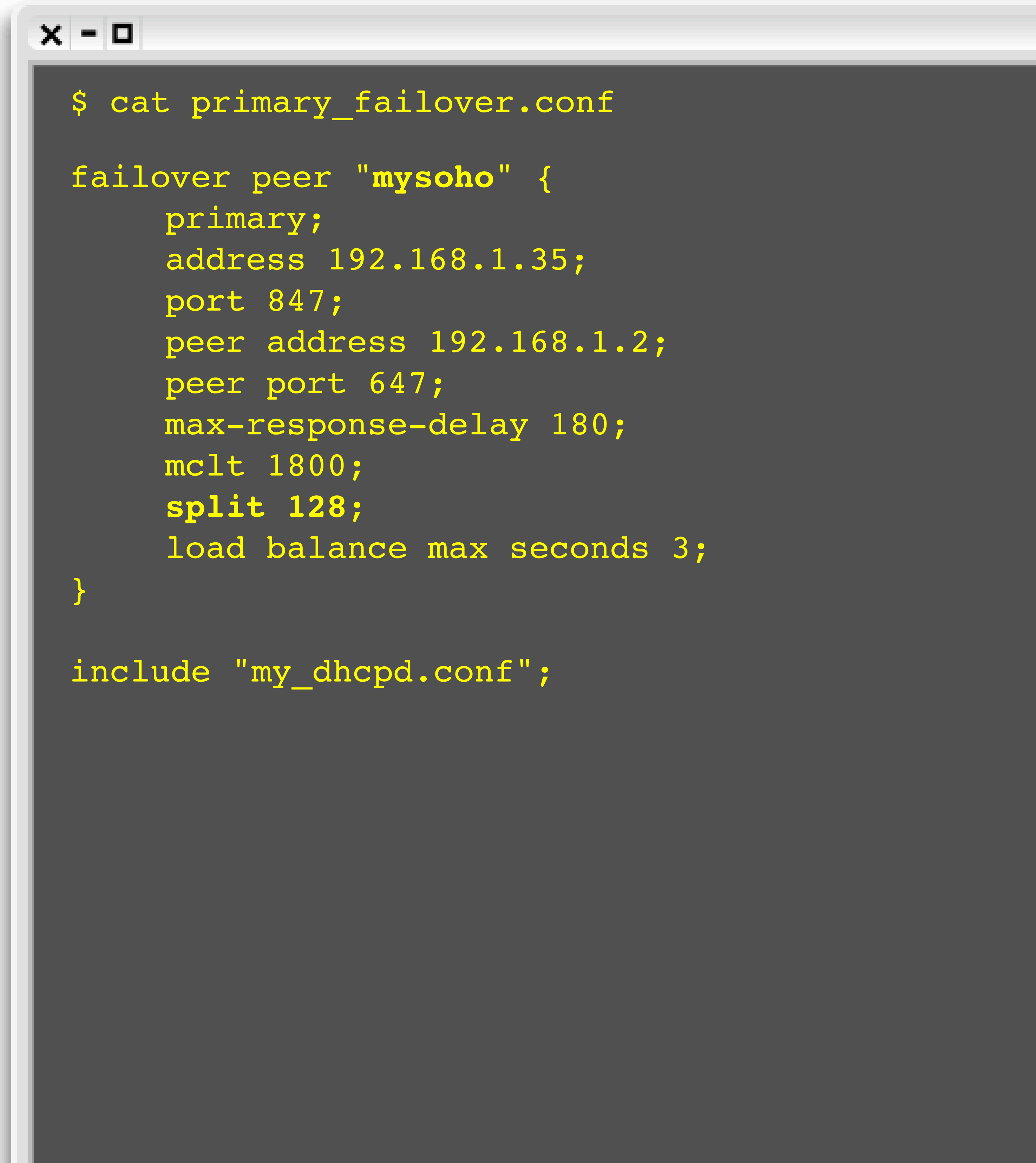
'Failover' taking-over capacity is linked to the address pools, not the servers themselves



Failover

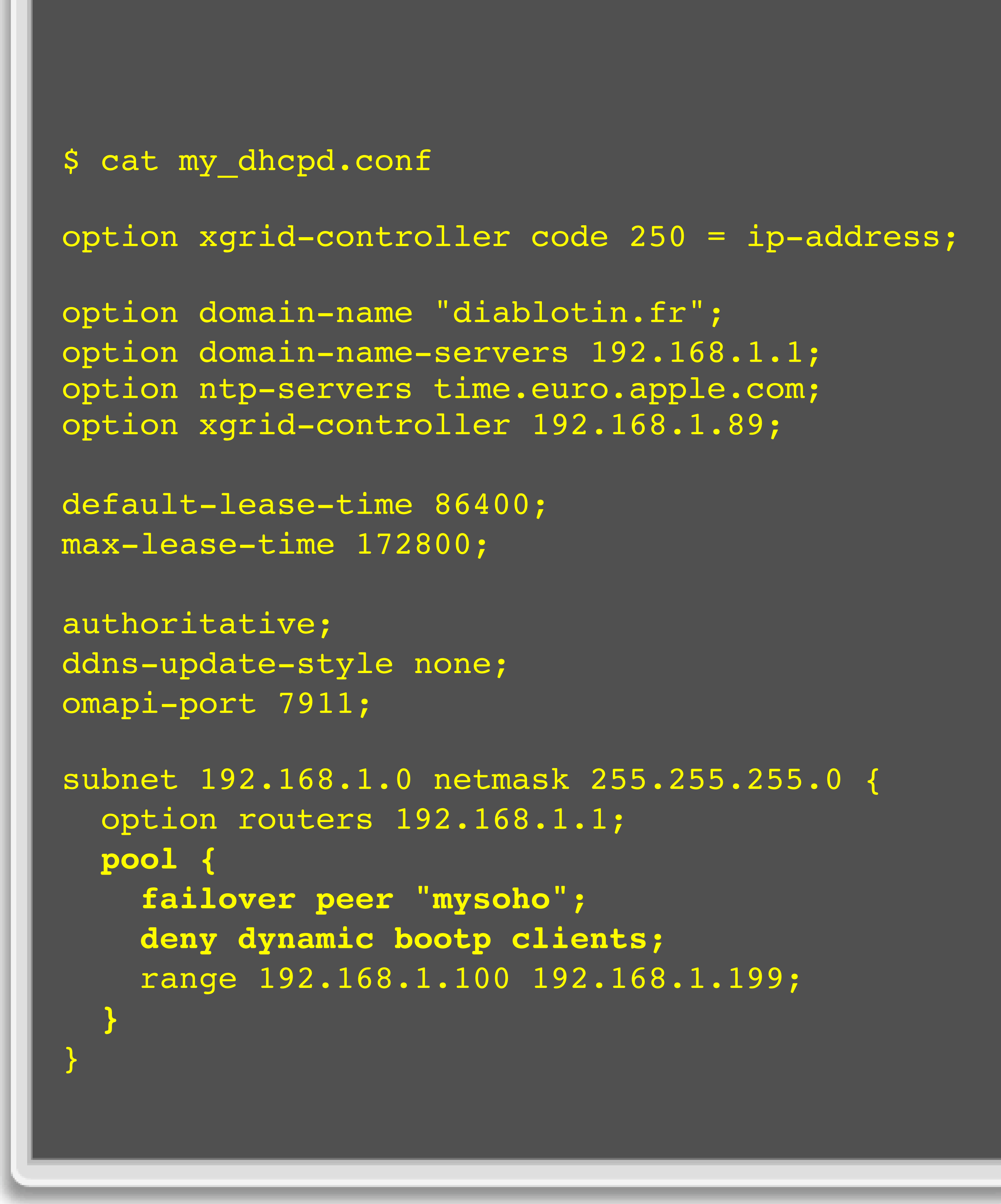
Let's practice…

Failover configuration — primary server



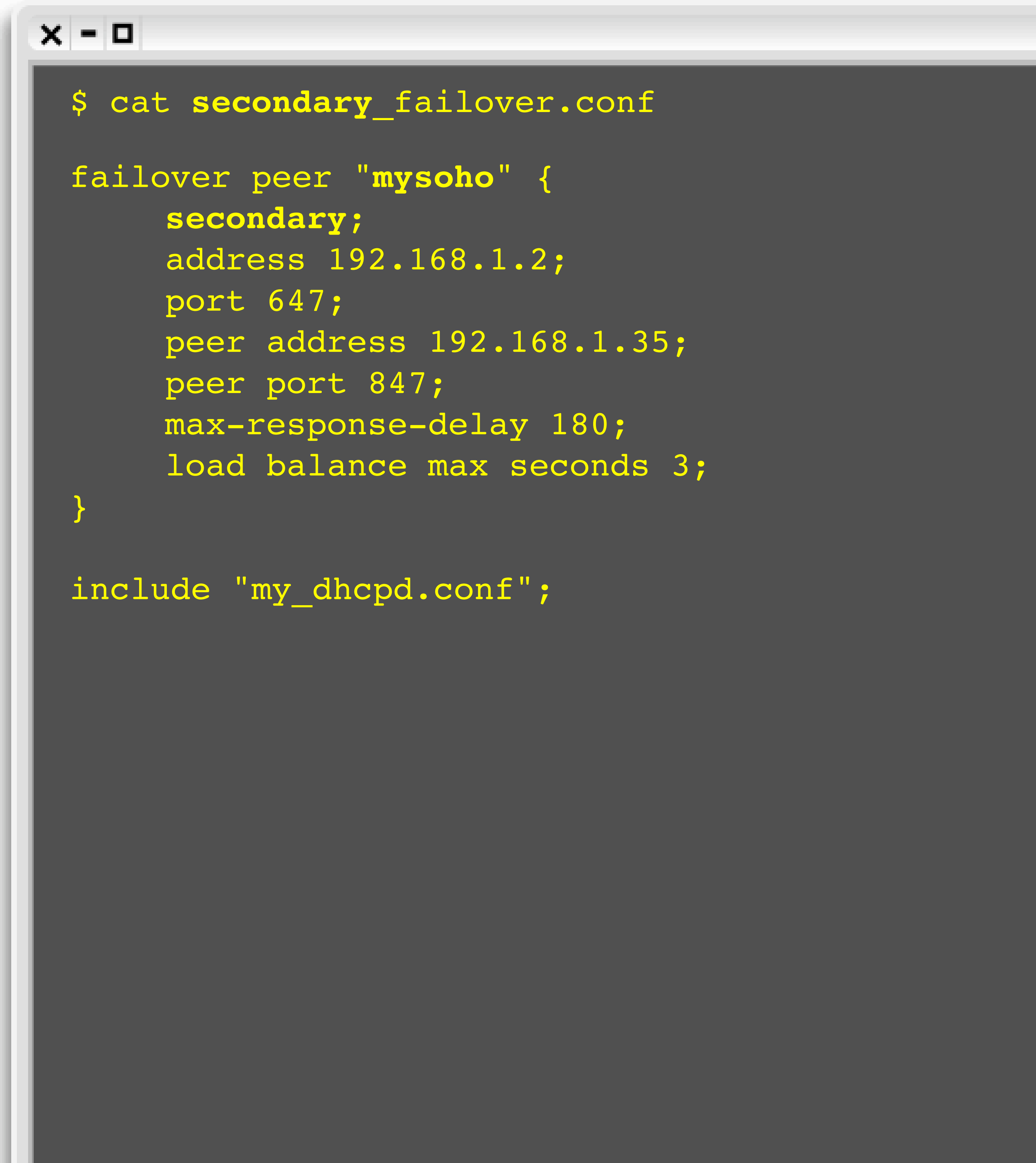
```
$ cat primary_failover.conf

failover peer "mysoho" {
     primary;
     address 192.168.1.35;
     port 847;
     peer address 192.168.1.2;
     peer port 647;
     max-response-delay 180;
     mclt 1800;
     split 128;
     load balance max seconds 3;
}

include "my_dhcpd.conf";
```

Primary server



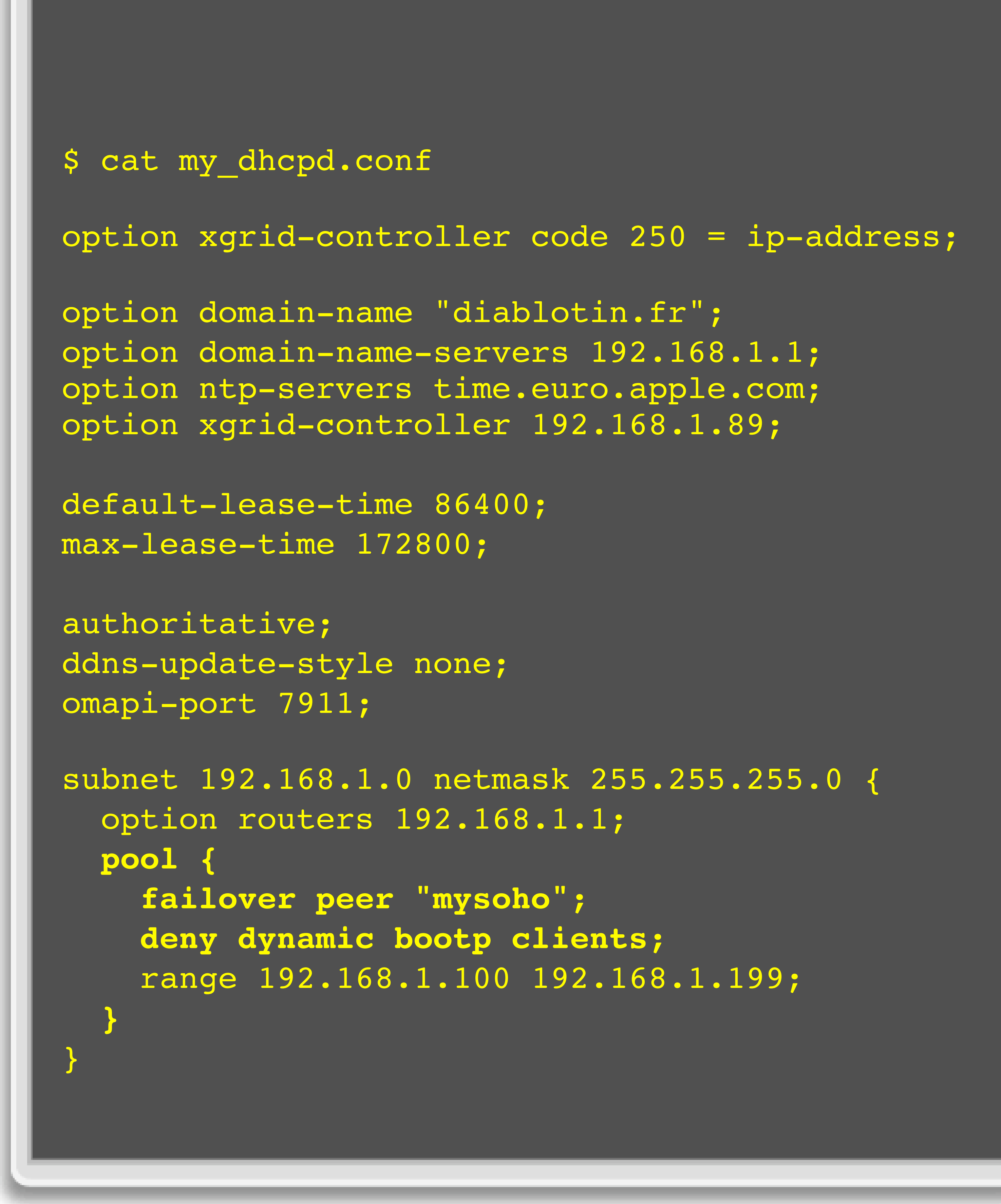
```
$ cat my_dhcpd.conf

option xgrid-controller code 250 = ip-address;

option domain-name "diablotin.fr";
option domain-name-servers 192.168.1.1;
option ntp-servers time.euro.apple.com;
option xgrid-controller 192.168.1.89;

default-lease-time 86400;
max-lease-time 172800;

authoritative;
ddns-update-style none;
omapi-port 7911;

subnet 192.168.1.0 netmask 255.255.255.0 {
  option routers 192.168.1.1;
  pool {
    failover peer "mysoho";
    deny dynamic bootp clients;
    range 192.168.1.100 192.168.1.199;
  }
}
```

Failover configuration — secondary server



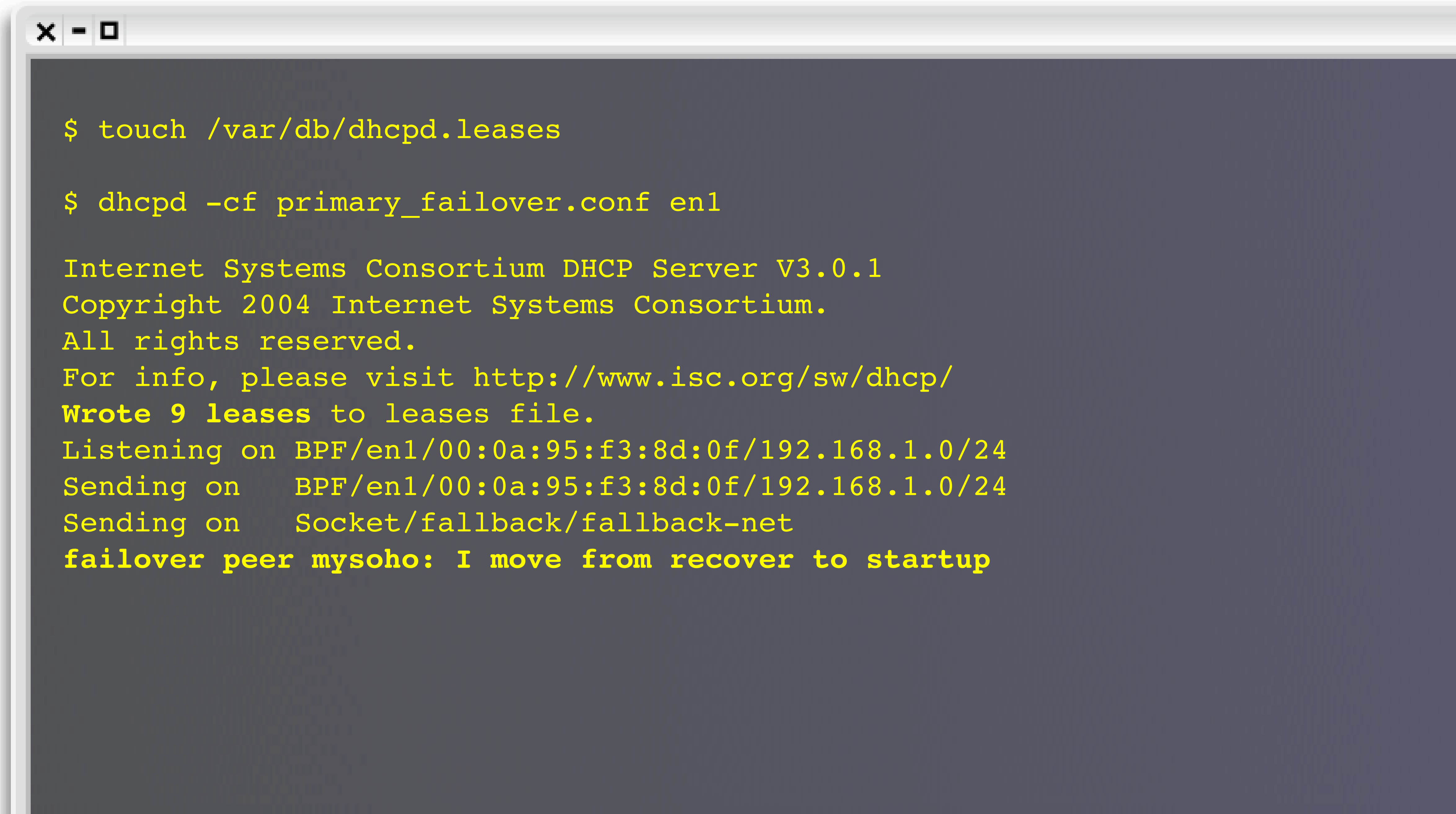
```
$ cat secondary_failover.conf

failover peer "mysoho" {
     secondary;
     address 192.168.1.2;
     port 647;
     peer address 192.168.1.35;
     peer port 847;
     max-response-delay 180;
     load balance max seconds 3;
}

include "my_dhcpd.conf";
```

Secondary server



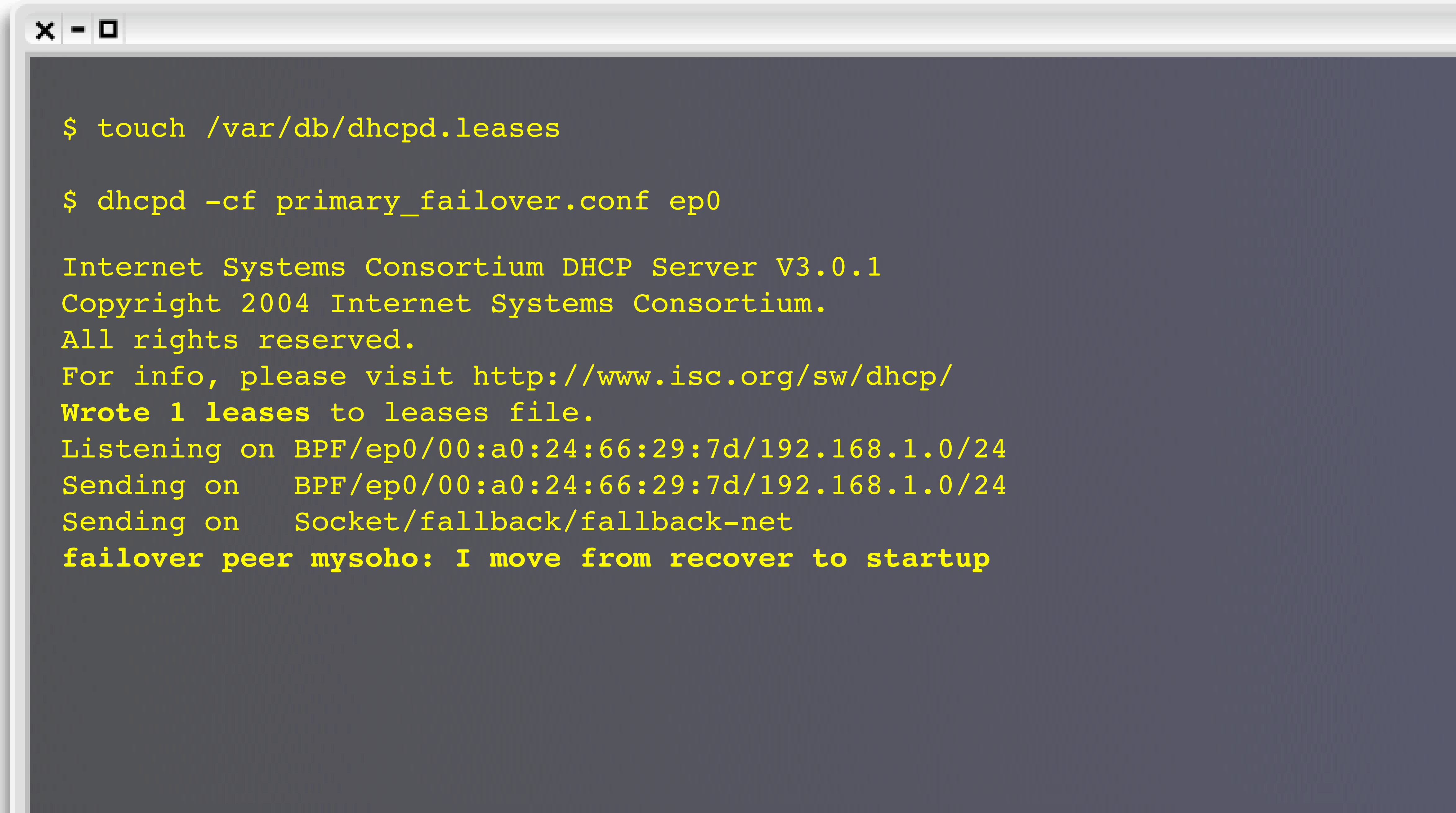
```
$ cat my_dhcpd.conf

option xgrid-controller code 250 = ip-address;

option domain-name "diablotin.fr";
option domain-name-servers 192.168.1.1;
option ntp-servers time.euro.apple.com;
option xgrid-controller 192.168.1.89;

default-lease-time 86400;
max-lease-time 172800;

authoritative;
ddns-update-style none;
omapi-port 7911;

subnet 192.168.1.0 netmask 255.255.255.0 {
  option routers 192.168.1.1;
  pool {
    failover peer "mysoho";
    deny dynamic bootp clients;
    range 192.168.1.100 192.168.1.199;
  }
}
```

Failover startup — primary server

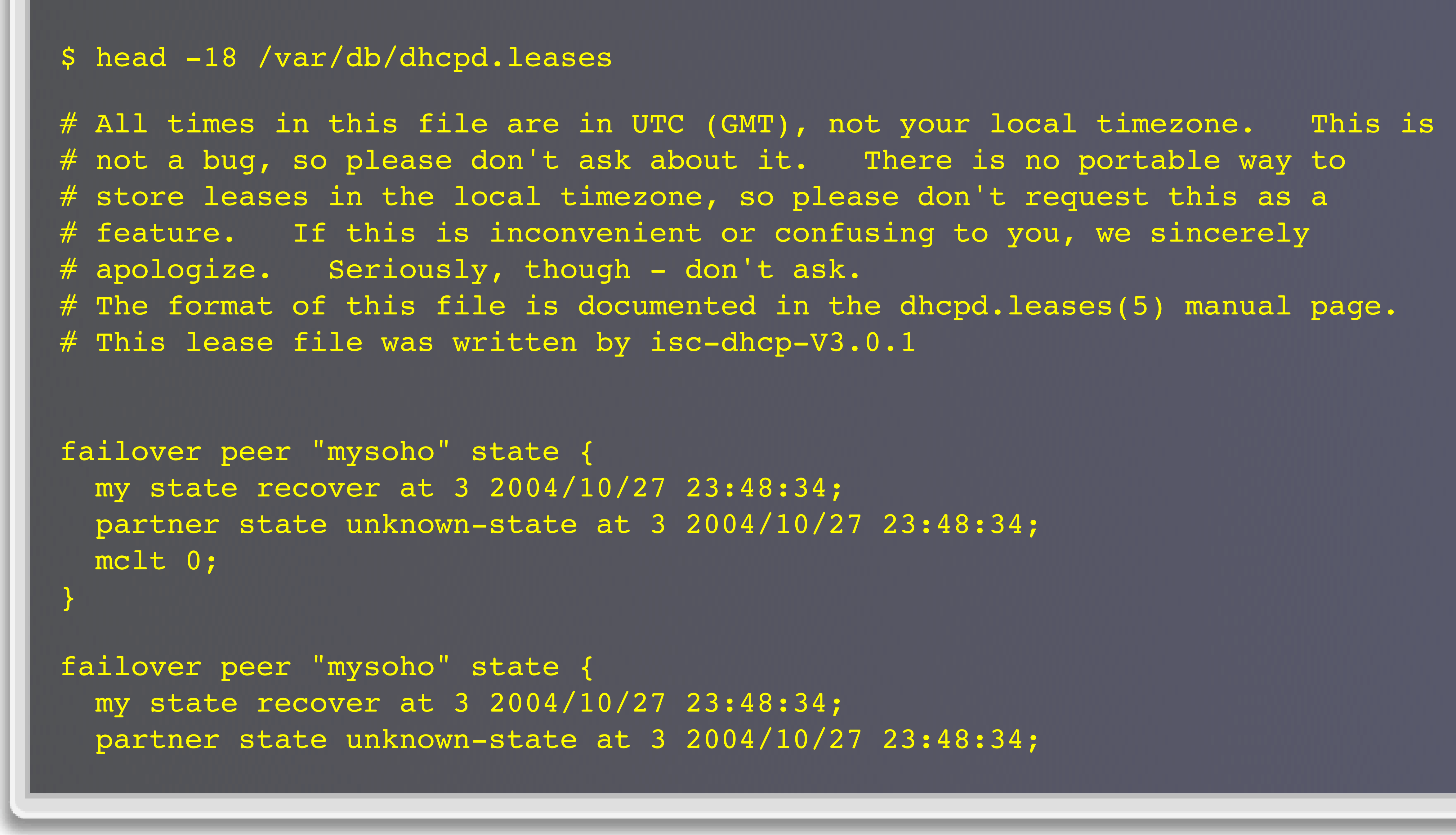


```
$ touch /var/db/dhcpd.leases

$ dhcpd -cf primary_failover.conf en1

Internet Systems Consortium DHCP Server V3.0.1
Copyright 2004 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/
Wrote 9 leases to leases file.
Listening on BPF/en1/00:0a:95:f3:8d:0f/192.168.1.0/24
Sending on   BPF/en1/00:0a:95:f3:8d:0f/192.168.1.0/24
Sending on   Socket/fallback/fallback-net
failover peer mysoho: I move from recover to startup
```

Failover startup — secondary server

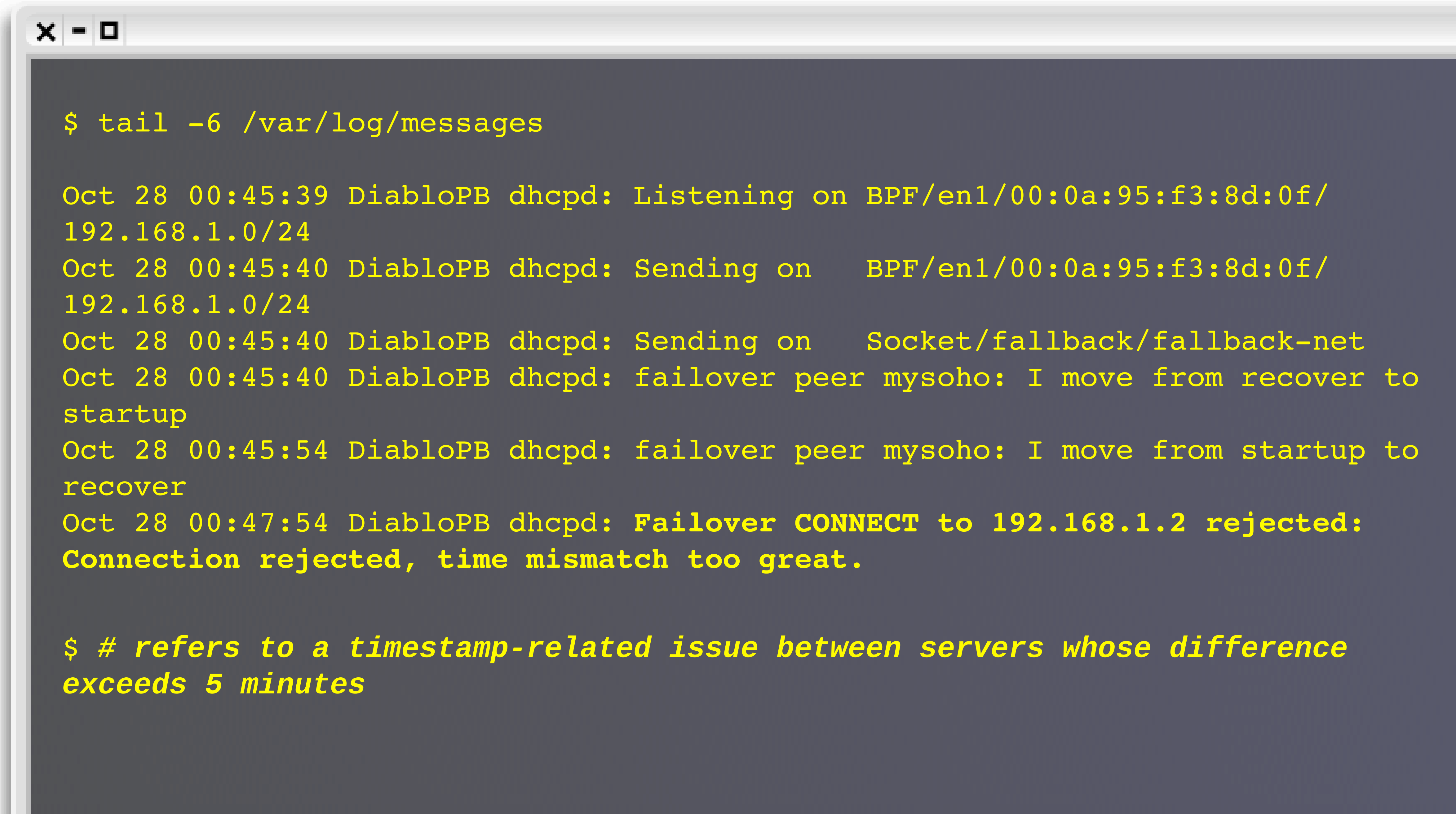


```
$ touch /var/db/dhcpd.leases

$ dhcpd -cf primary_failover.conf ep0

Internet Systems Consortium DHCP Server V3.0.1
Copyright 2004 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/
Wrote 1 leases to leases file.
Listening on BPF/ep0/00:a0:24:66:29:7d/192.168.1.0/24
Sending on   BPF/ep0/00:a0:24:66:29:7d/192.168.1.0/24
Sending on   Socket/fallback/fallback-net
failover peer mysoho: I move from recover to startup
```



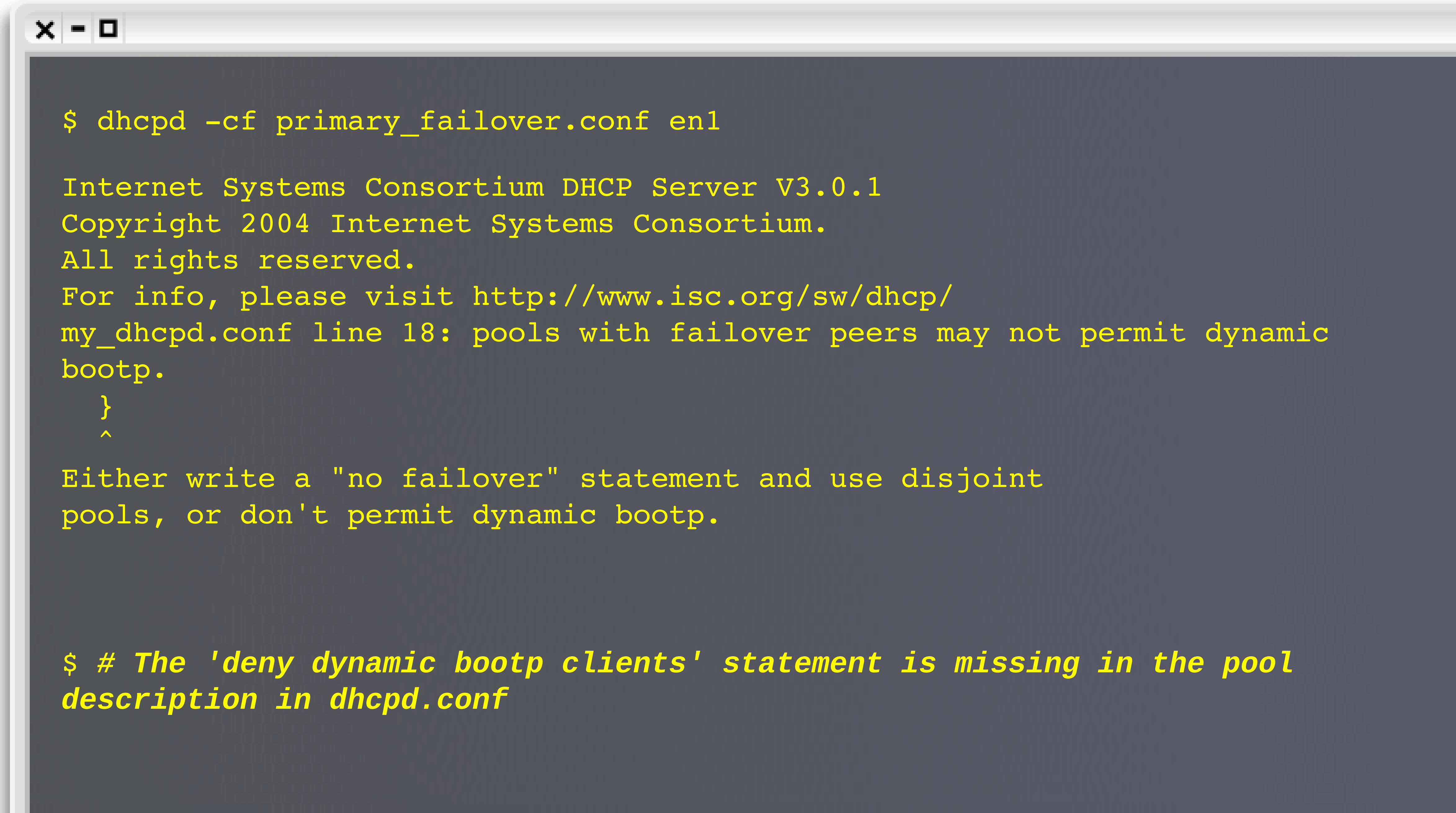
```
$ head -18 /var/db/dhcpd.leases

# 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-V3.0.1


failover peer "mysoho" state {
  my state recover at 3 2004/10/27 23:48:34;
  partner state unknown-state at 3 2004/10/27 23:48:34;
  mclt 0;
}

failover peer "mysoho" state {
  my state recover at 3 2004/10/27 23:48:34;
  partner state unknown-state at 3 2004/10/27 23:48:34;
```

When things go wrong



```
$ tail -6 /var/log/messages

Oct 28 00:45:39 DiabloPB dhcpd: Listening on BPF/en1/00:0a:95:f3:8d:0f/
192.168.1.0/24
Oct 28 00:45:40 DiabloPB dhcpd: Sending on   BPF/en1/00:0a:95:f3:8d:0f/
192.168.1.0/24
Oct 28 00:45:40 DiabloPB dhcpd: Sending on   Socket/fallback/fallback-net
Oct 28 00:45:40 DiabloPB dhcpd: failover peer mysoho: I move from recover to
startup
Oct 28 00:45:54 DiabloPB dhcpd: failover peer mysoho: I move from startup to
recover
Oct 28 00:47:54 DiabloPB dhcpd: Failover CONNECT to 192.168.1.2 rejected:
Connection rejected, time mismatch too great.

$ # refers to a timestamp-related issue between servers whose difference
exceeds 5 minutes
```

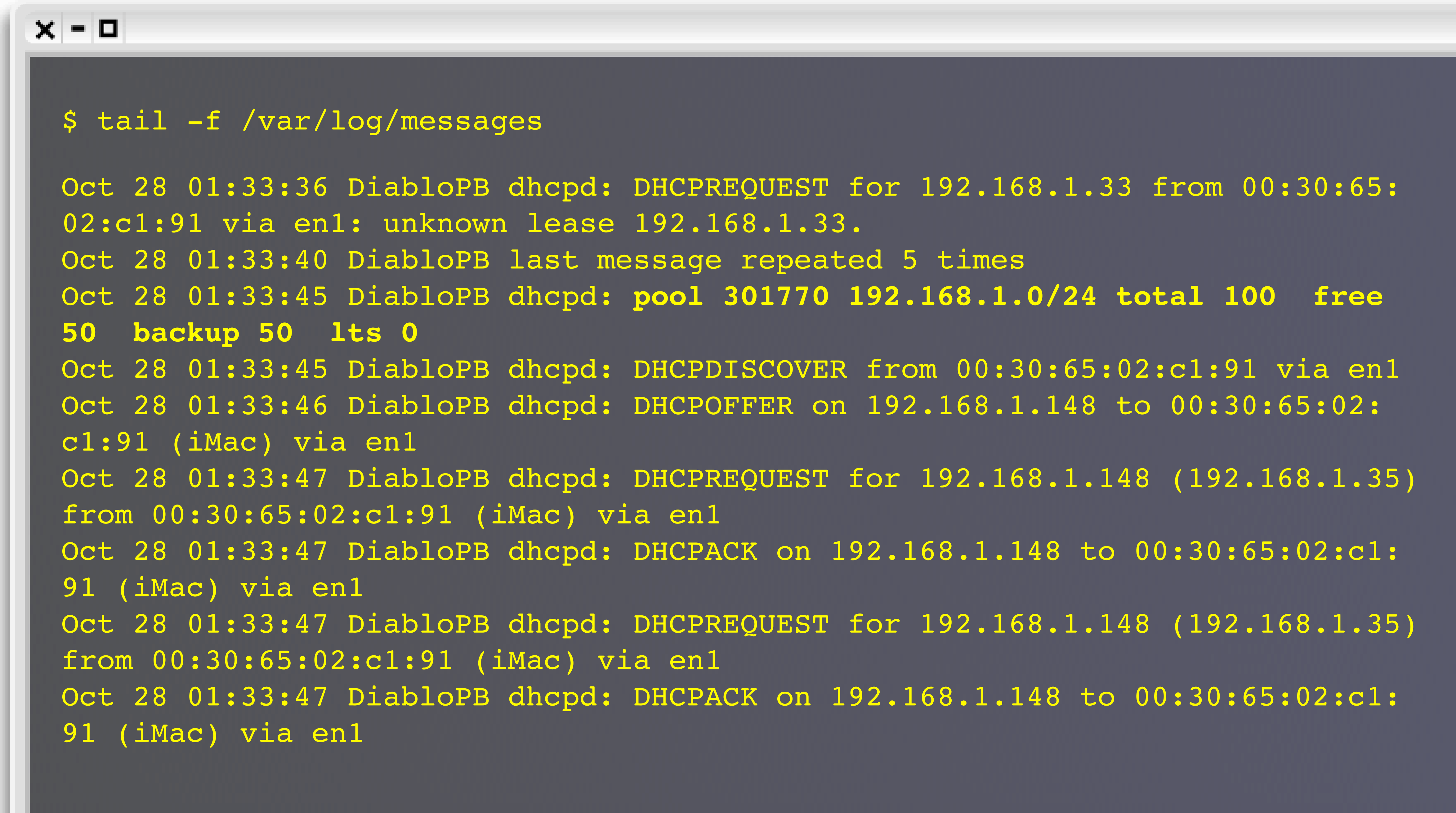
When things go wrong



```
$ dhcpd -cf primary_failover.conf en1

Internet Systems Consortium DHCP Server V3.0.1
Copyright 2004 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/
my_dhcpd.conf line 18: pools with failover peers may not permit dynamic
bootp.
  }
  ^
Either write a "no failover" statement and use disjoint
pools, or don't permit dynamic bootp.



$ # The 'deny dynamic bootp clients' statement is missing in the pool
description in dhcpd.conf
```

Load-balancing in action

A new IP address is allocated to a host that woke up from stand-by and wanted his former address (192.168.1.33) back.
One line in this lot refers to load balancing (pool 301770 ...)



```
$ tail -f /var/log/messages

Oct 28 01:33:36 DiabloPB dhcpd: DHCPREQUEST for 192.168.1.33 from 00:30:65:
02:c1:91 via en1: unknown lease 192.168.1.33.
Oct 28 01:33:40 DiabloPB last message repeated 5 times
Oct 28 01:33:45 DiabloPB dhcpd: pool 301770 192.168.1.0/24 total 100  free
50  backup 50  lts 0
Oct 28 01:33:45 DiabloPB dhcpd: DHCPDISCOVER from 00:30:65:02:c1:91 via en1
Oct 28 01:33:46 DiabloPB dhcpd: DHCPOFFER on 192.168.1.148 to 00:30:65:02:
c1:91 (iMac) via en1
Oct 28 01:33:47 DiabloPB dhcpd: DHCPREQUEST for 192.168.1.148 (192.168.1.35)
from 00:30:65:02:c1:91 (iMac) via en1
Oct 28 01:33:47 DiabloPB dhcpd: DHCPACK on 192.168.1.148 to 00:30:65:02:c1:
91 (iMac) via en1
Oct 28 01:33:47 DiabloPB dhcpd: DHCPREQUEST for 192.168.1.148 (192.168.1.35)
from 00:30:65:02:c1:91 (iMac) via en1
Oct 28 01:33:47 DiabloPB dhcpd: DHCPACK on 192.168.1.148 to 00:30:65:02:c1:
91 (iMac) via en1
```

What to verify?

- There are many running states for the failover:
  - NORMAL
  - COMMUNICATIONS-INTERRUPTED
  - PARTNER-DOWN
  - And few transient states

What to do?

- When a communication problem is detected, dhcpd automatically switches to the COMMUNICATIONS-INTERRUPTED state
- Two cases:
  - Communication comes back or the failed peer is repaired: servers synchronize their informations and state is automatically switched to NORMAL
  - The failing peer is unavailable for a period of time: it is advisable to switch 'manually' the state to PARTNER-DOWN

Switching state (first method)

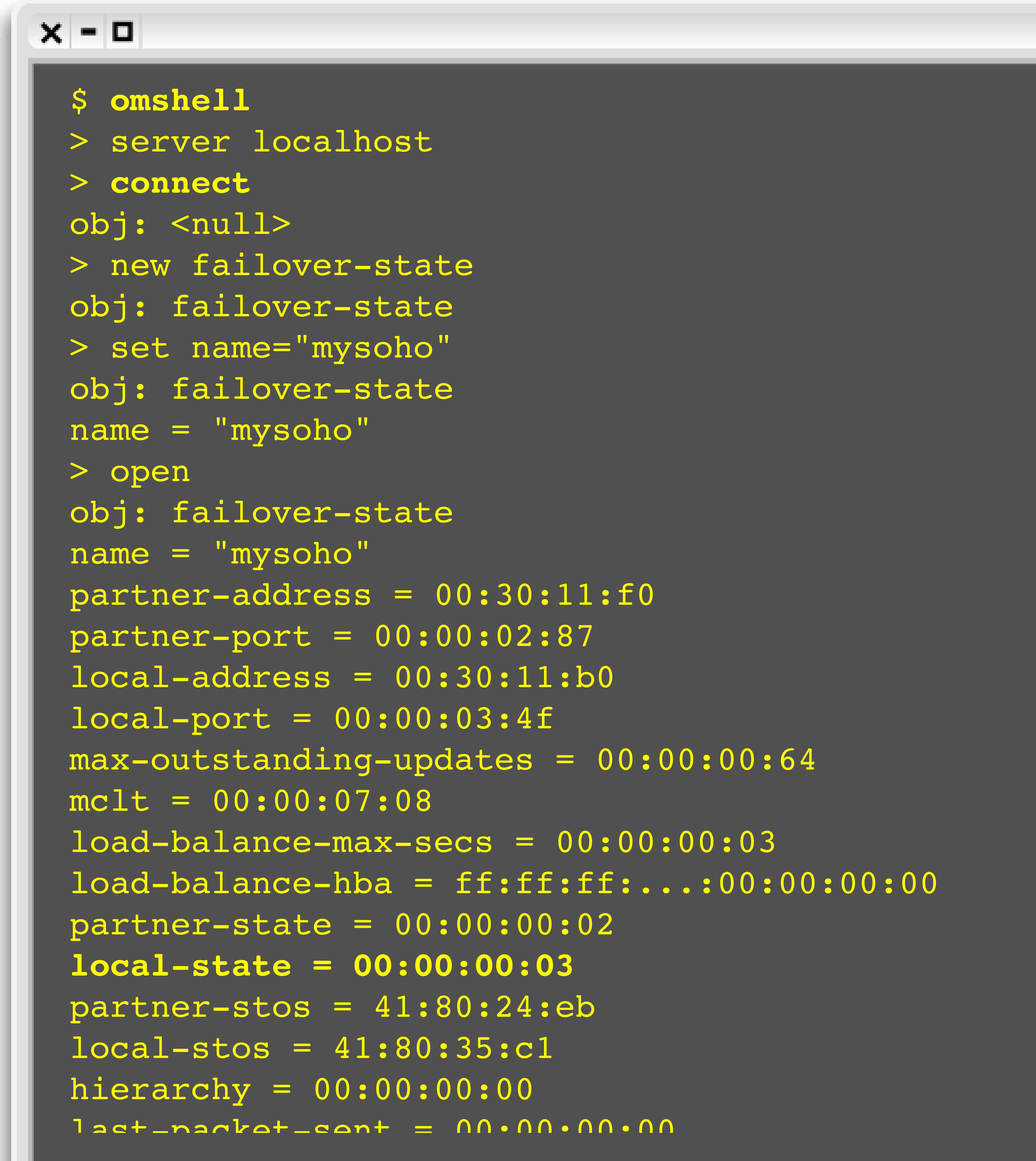
- Modifying configuration file:
  - Edit /etc/dhcpd.conf
    - Add a section

```
failover peer name state {
        my state partner-down;
        peer state state at date;
}
```

  - Stop then start dhcpd

Diablotin - Tous droits réservés

Switching state (second method)

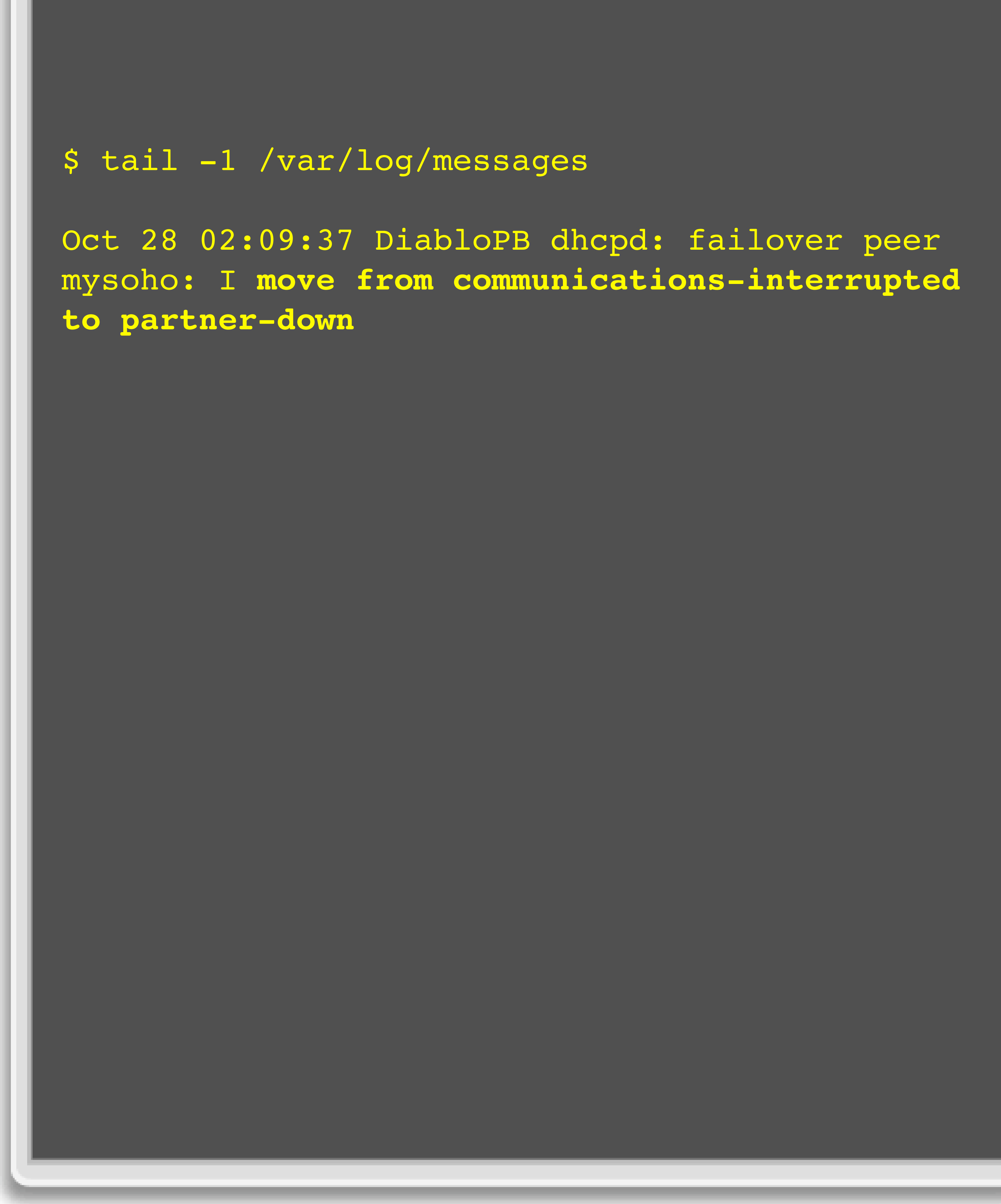


```
$ omshell
> server localhost
> connect
obj: <null>
> new failover-state
obj: failover-state
> set name="mysoho"
obj: failover-state
name = "mysoho"
> open
obj: failover-state
name = "mysoho"
partner-address = 00:30:11:f0
partner-port = 00:00:02:87
local-address = 00:30:11:b0
local-port = 00:00:03:4f
max-outstanding-updates = 00:00:00:64
mclt = 00:00:07:08
load-balance-max-secs = 00:00:00:03
load-balance-hba = ff:ff:ff:...:00:00:00:00
partner-state = 00:00:00:02
local-state = 00:00:00:03
partner-stos = 41:80:24:eb
local-stos = 41:80:35:c1
hierarchy = 00:00:00:00
last-packet-sent = 00:00:00:00
```

Switching state (second method)

```
> set local-state=1
obj: failover-state
name = "mysoho"
partner-address = 00:30:11:f0
partner-port = 00:00:02:87
local-address = 00:30:11:b0
local-port = 00:00:03:4f
max-outstanding-updates = 00:00:00:64
mclt = 00:00:07:08
load-balance-max-secs = 00:00:00:03
load-balance-hba = ff:ff:ff:...:00:00:00:00
partner-state = 00:00:00:02
local-state = 1
partner-stos = 41:80:24:eb
...
cur-unacked-updates = 00:00:00:00
> update
obj: failover-state
name = "mysoho"
partner-address = 00:30:11:f0
partner-port = 00:00:02:87
local-address = 00:30:11:b0
local-port = 00:00:03:4f
max-outstanding-updates = 00:00:00:64
mclt = 00:00:07:08
load-balance-max-secs = 00:00:00:03
load-balance-hba = ff:ff:ff:...:00:00:00:00
partner-state = 00:00:00:02
local-state = 1
partner-stos = 41:80:24:eb
...
cur-unacked-updates = 00:00:00:00
> ^C
```

Checking modification



```
$ tail -1 /var/log/messages

Oct 28 02:09:37 DiabloPB dhcpd: failover peer
mysoho: I move from communications-interrupted
to partner-down
```

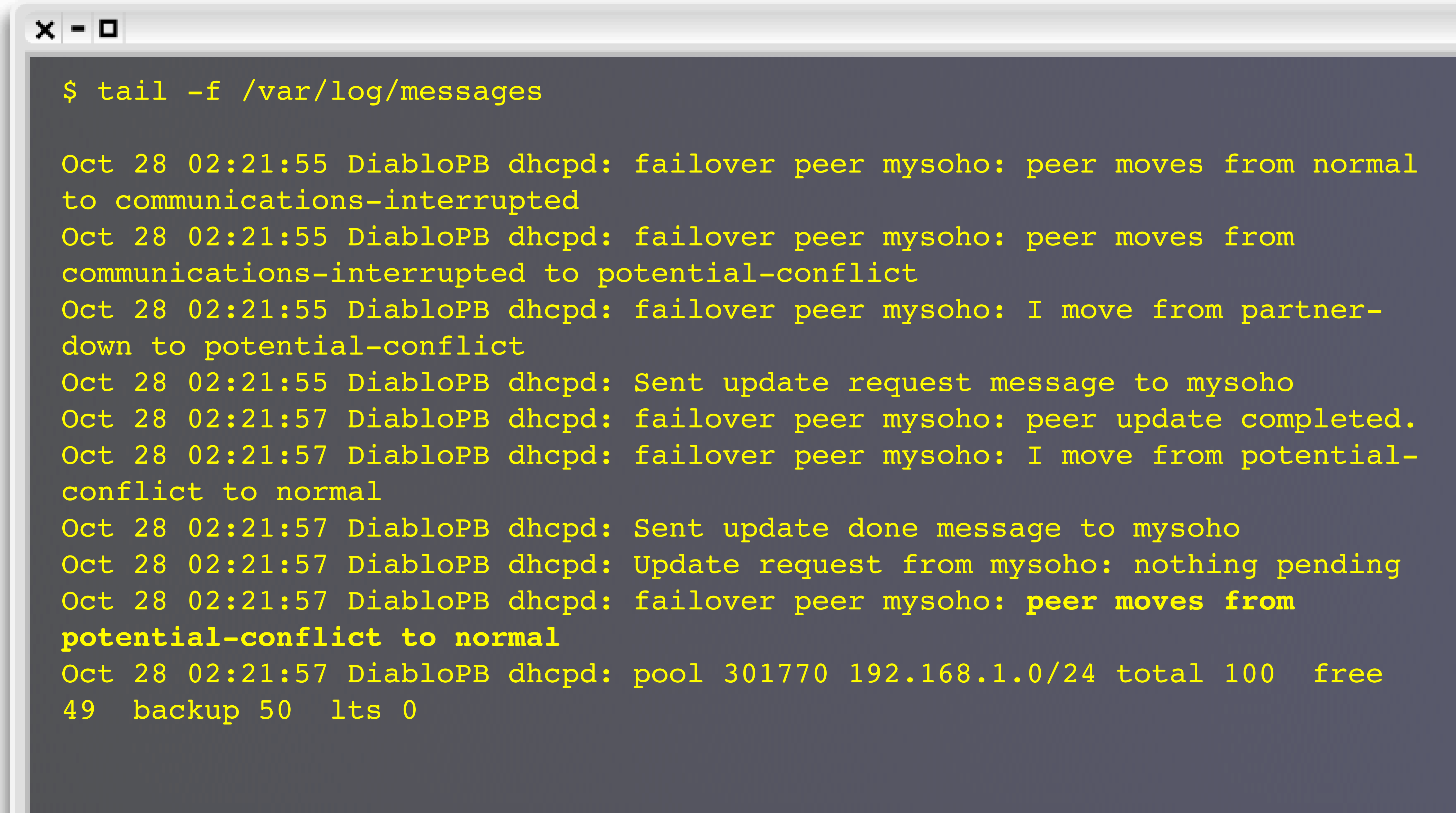
Back to normal state

- LATER ON...
  - The failing server comes back. Data are recovered -especially configuration files- as they were before its brutal stop. The lease file may happe to haven disappeared or be corrupted, which doesn't seem to be problematic.
  - THERE IS NOTHING SPECIFIC TO DO ON ANY HOST, EXCEPT LAUNCHING dhcpd on the mended 'secondary' server, with exactly the same configuration it had before it stopped.
    - When it launches, the last line mentions:

```
failover peer mysoho: I move from communications-interrupted to startup
```

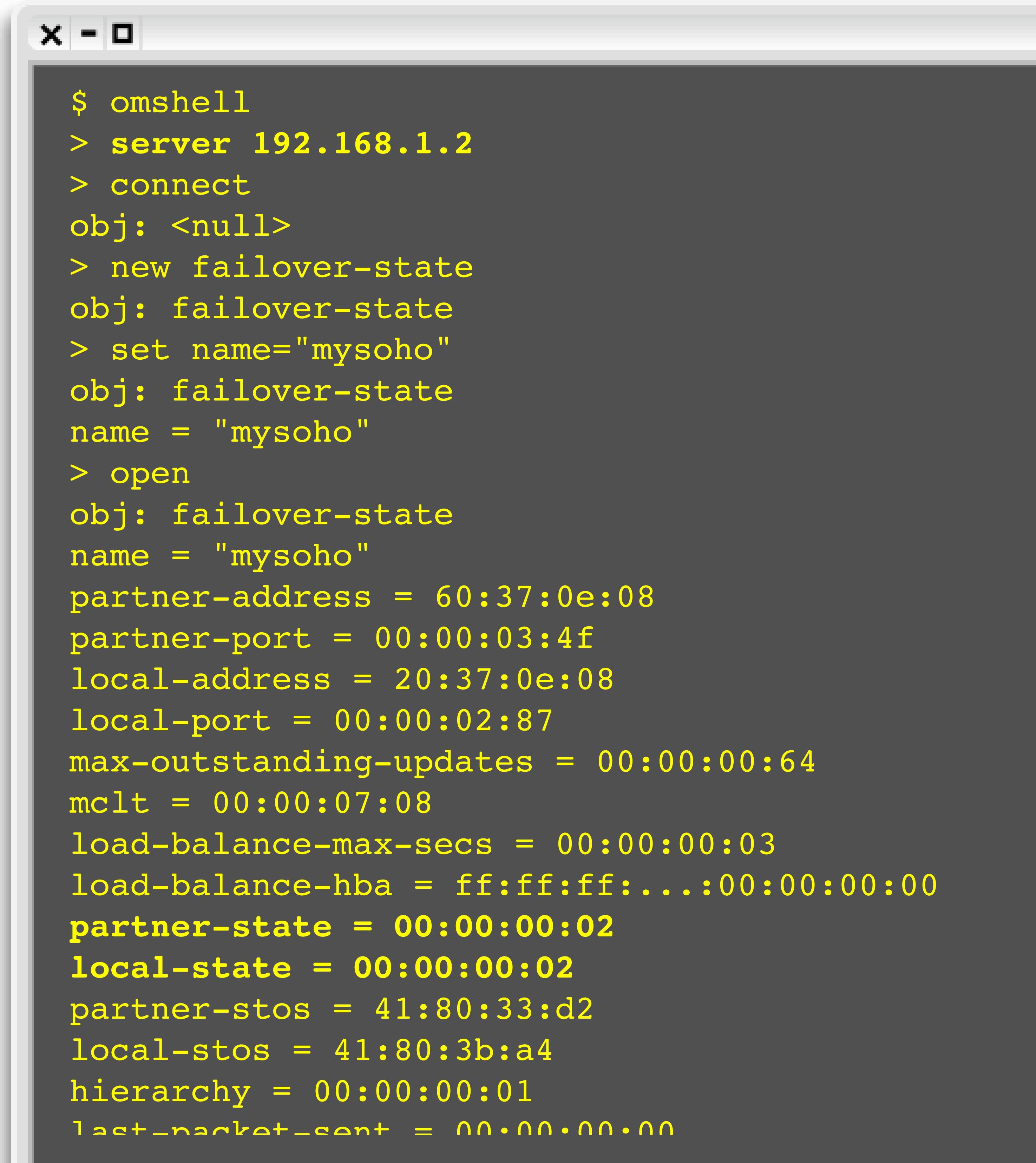
Back to normal state

The 'primary' server's log, on the other hand, mentions:



```
$ tail -f /var/log/messages

Oct 28 02:21:55 DiabloPB dhcpd: failover peer mysoho: peer moves from normal
to communications-interrupted
Oct 28 02:21:55 DiabloPB dhcpd: failover peer mysoho: peer moves from
communications-interrupted to potential-conflict
Oct 28 02:21:55 DiabloPB dhcpd: failover peer mysoho: I move from partner-
down to potential-conflict
Oct 28 02:21:55 DiabloPB dhcpd: Sent update request message to mysoho
Oct 28 02:21:57 DiabloPB dhcpd: failover peer mysoho: peer update completed.
Oct 28 02:21:57 DiabloPB dhcpd: failover peer mysoho: I move from potential-
conflict to normal
Oct 28 02:21:57 DiabloPB dhcpd: Sent update done message to mysoho
Oct 28 02:21:57 DiabloPB dhcpd: Update request from mysoho: nothing pending
Oct 28 02:21:57 DiabloPB dhcpd: failover peer mysoho: peer moves from
potential-conflict to normal
Oct 28 02:21:57 DiabloPB dhcpd: pool 301770 192.168.1.0/24 total 100  free
49  backup 50  lts 0
```

EVERYTHING IS BACK IN ORDER!



```
$ omshell
> server 192.168.1.2
> connect
obj: <null>
> new failover-state
obj: failover-state
> set name="mysoho"
obj: failover-state
name = "mysoho"
> open
obj: failover-state
name = "mysoho"
partner-address = 60:37:0e:08
partner-port = 00:00:03:4f
local-address = 20:37:0e:08
local-port = 00:00:02:87
max-outstanding-updates = 00:00:00:64
mclt = 00:00:07:08
load-balance-max-secs = 00:00:00:03
load-balance-hba = ff:ff:ff:...:00:00:00:00
partner-state = 00:00:00:02
local-state = 00:00:00:02
partner-stos = 41:80:33:d2
local-stos = 41:80:3b:a4
hierarchy = 00:00:00:01
last-packet-sent = 00:00:00:00
```

Diagnose and find problems

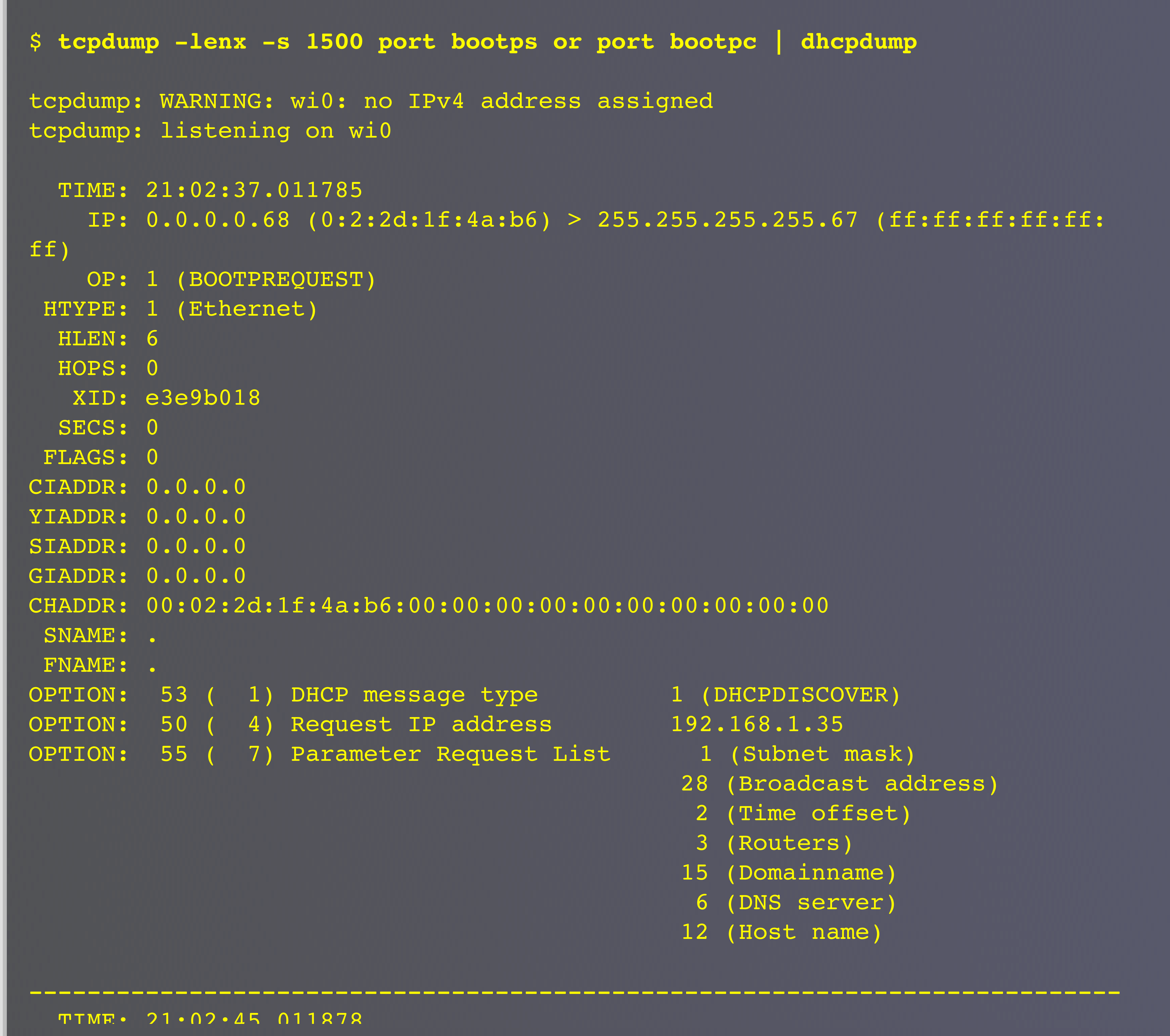
- Has the topology of the one or several networks 'attached' to the DHCP server been completely defined?
  - VERY IMPORTANT since a server cannot just deliver or stay silent, IT CAN ALSO REFUSE. Doing that, it can refuse instead of an authorized server. It can refuse in good faith, based on an incomplete or incorrect knowledge of its configuration!
    - hence the importance of 'authoritative' and 'not authoritative' (a security default meant for rookie administrators) directives.
    - ON THE CONTRARY, a server should NOT be systematically declared not authoritative as a would-be security measure. In such a case, many necessary updates do not take place, which pollutes the system with obsolete information -this information does actually not always fall within the direct competence of the DHCP server, as is the case with DNS records, for instance.

If logs are not enough

- dhcping:
  - only to check the static definitions of a DHCP server
- dhcpdump:
  - specialized tcpdump filter (Works in conjunction with this command)
- dhcp-sniff:
  - DOESN'T SEE 'releases'!
  - Also looks pretty bling regarding user-defined options! (dhcpdump to the contrary)
- And of course generic sniffing tools like tcpdump, ethereal, ettercap and so on

DHCP sniffing

Let's practice…



```
$ tcpdump -lenx -s 1500 port bootps or port bootpc | dhcpdump

tcpdump: WARNING: wi0: no IPv4 address assigned
tcpdump: listening on wi0

  TIME: 21:02:37.011785
    IP: 0.0.0.0.68 (0:2:2d:1f:4a:b6) > 255.255.255.255.67 (ff:ff:ff:ff:ff:
ff)
    OP: 1 (BOOTPREQUEST)
 HTYPE: 1 (Ethernet)
  HLEN: 6
  HOPS: 0
   XID: e3e9b018
  SECS: 0
 FLAGS: 0
CIADDR: 0.0.0.0
YIADDR: 0.0.0.0
SIADDR: 0.0.0.0
GIADDR: 0.0.0.0
CHADDR: 00:02:2d:1f:4a:b6:00:00:00:00:00:00:00:00:00:00
 SNAME: .
 FNAME: .
OPTION:  53 (  1) DHCP message type         1 (DHCPDISCOVER)
OPTION:  50 (  4) Request IP address        192.168.1.35
OPTION:  55 (  7) Parameter Request List      1 (Subnet mask)
                                             28 (Broadcast address)
                                              2 (Time offset)
                                              3 (Routers)
                                             15 (Domainname)
                                              6 (DNS server)
                                             12 (Host name)

---------------------------------------------------------------------------
  TIME: 21:02:45.011878
```

```
$ dhcp-sniff wi0

dhcp-sniff :           opened wi0 for packet capturing

--------------------------Ethernet Header-----------------------------
SRC MAC: 0:2:2d:1f:4a:b6             DST MAC: ff:ff:ff:ff:ff:ff
-------------------------------IP HDR--------------------------------
Source: 0.0.0.0                         Destination: 255.255.255.255
---------------------------------------------------------------------
----------------------------------DHCP Info--------------------------
[Operation: BootRequest ] [Hardware Type: Ethernet ]
[Hardware Addr Len: 6   ] [Hops: 0                 ] [XID: 60E22D6B ]
[Seconds: 0             ] [Flags: ]

    [Client Addr: 0.0.0.0        ]  [Your Addr: 0.0.0.0         ]
    [Server Addr: 0.0.0.0        ]  [Agent Addr: 0.0.0.0        ]

         [Client HW Addr: 00022d1f4ab600000000000000000000]
---------------------------------------------------------------------
Cookie: Good   : 99.130.83.99
---------------------------------------------------------------------
DHCP Message Type: DISCOVER
Request IP Address: 192.168.1.35.
Parameter Request List:
---------------------------------------------------------------------
Subnet Mask, Broadcast Address, Time Offset, Routers, Domain Name,
Domain Name Servers, Hostname.
---------------------------------------------------------------------


--------------------------Ethernet Header-----------------------------
SRC MAC: 0:2:2d:1f:4a:b6             DST MAC: ff:ff:ff:ff:ff:ff
-------------------------------IP HDR--------------------------------
Source: 0.0.0.0                         Destination: 255.255.255.255
```

Traps

- Timestamping synchronization is important for hosts involved in automatic update procedures (replication when it comes to hosts working together within a 'failover' context).

- Same thing regarding security: cyphering systems (cipher) or cryptography like TSIG in order to avoid 'replay'-type attacks (authorized margin: 5 minutes tops).

  - NTP protocol should be considered

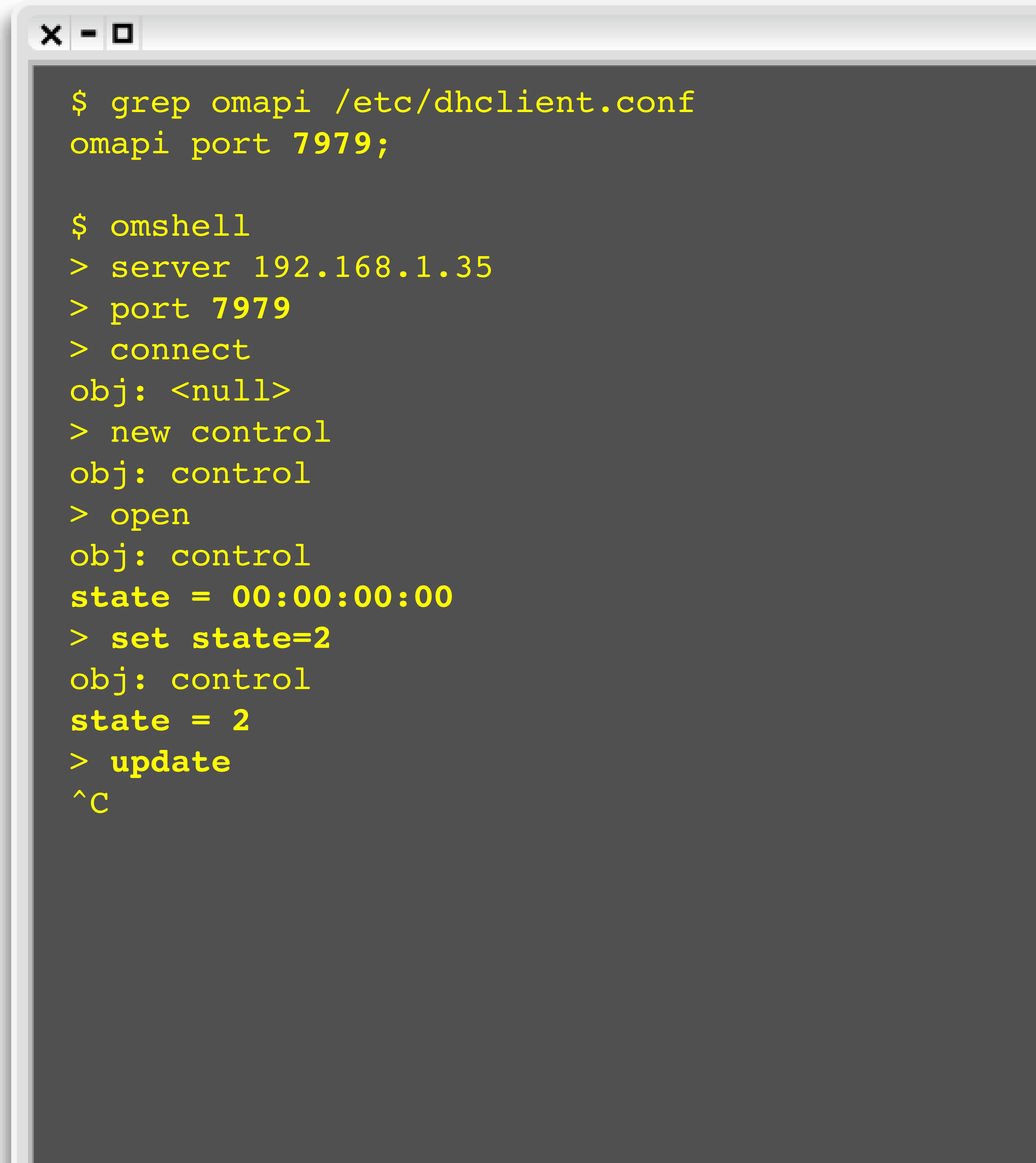
Suggestion

- It may be a good idea, at least on FreeBSD, to add a condition in /etc/rc.shutdown to invoke a dhclient *-r* if dhclient.pid is present and points to an active process
- In the same manner, one should add (using the OMAPI and the omshell) 'standby' and 'wake-up' actions for dhclient in /etc/rc.suspend and /etc/rc.resume

Change client status

Let's practice…

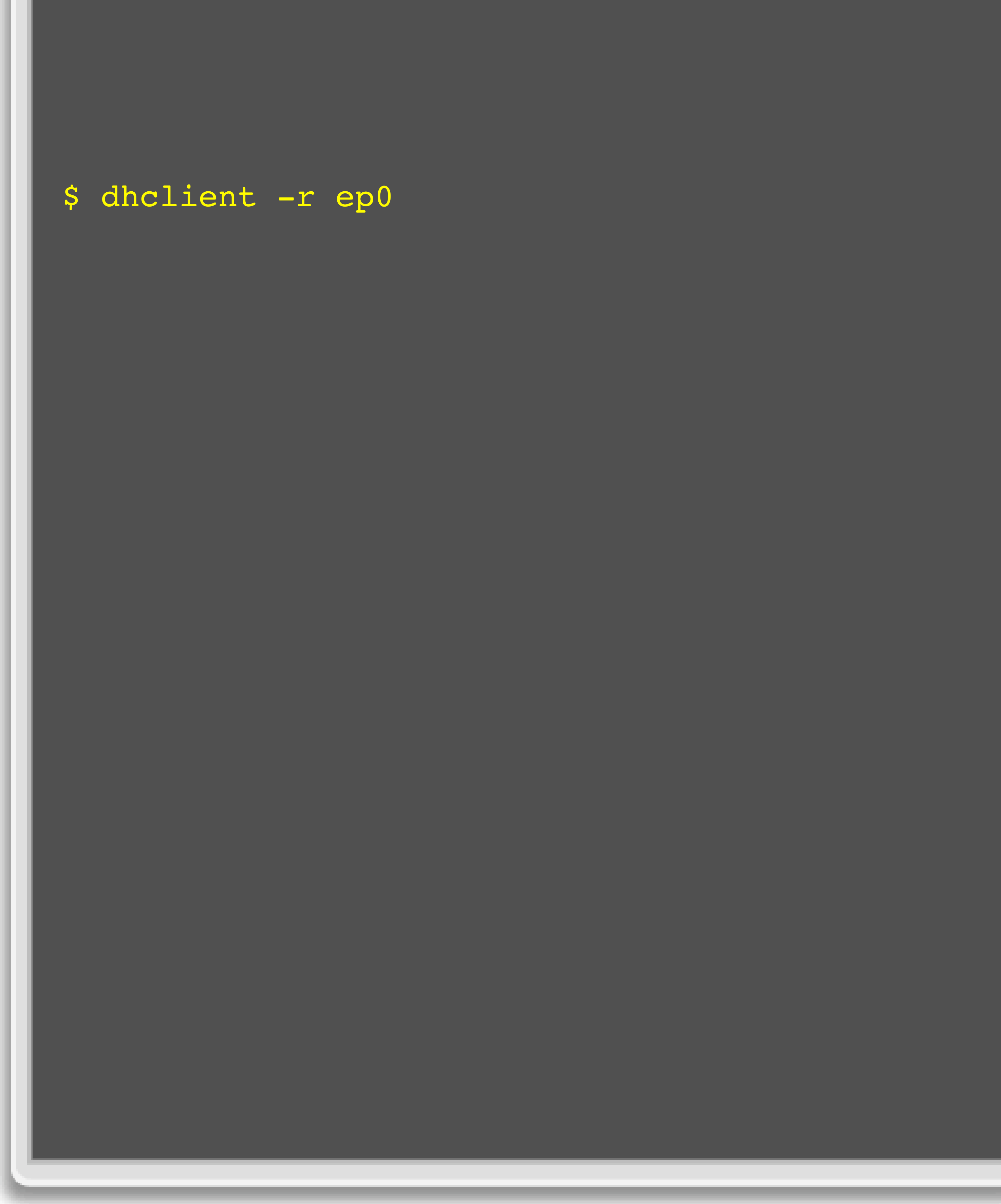
Using omshell…



```
$ grep omapi /etc/dhclient.conf
omapi port 7979;

$ omshell
> server 192.168.1.35
> port 7979
> connect
obj: <null>
> new control
obj: control
> open
obj: control
state = 00:00:00:00
> set state=2
obj: control
state = 2
> update
^C
```

Diablotin - Tous droits réservés

Using CLI…



```
$ dhclient -r ep0
```

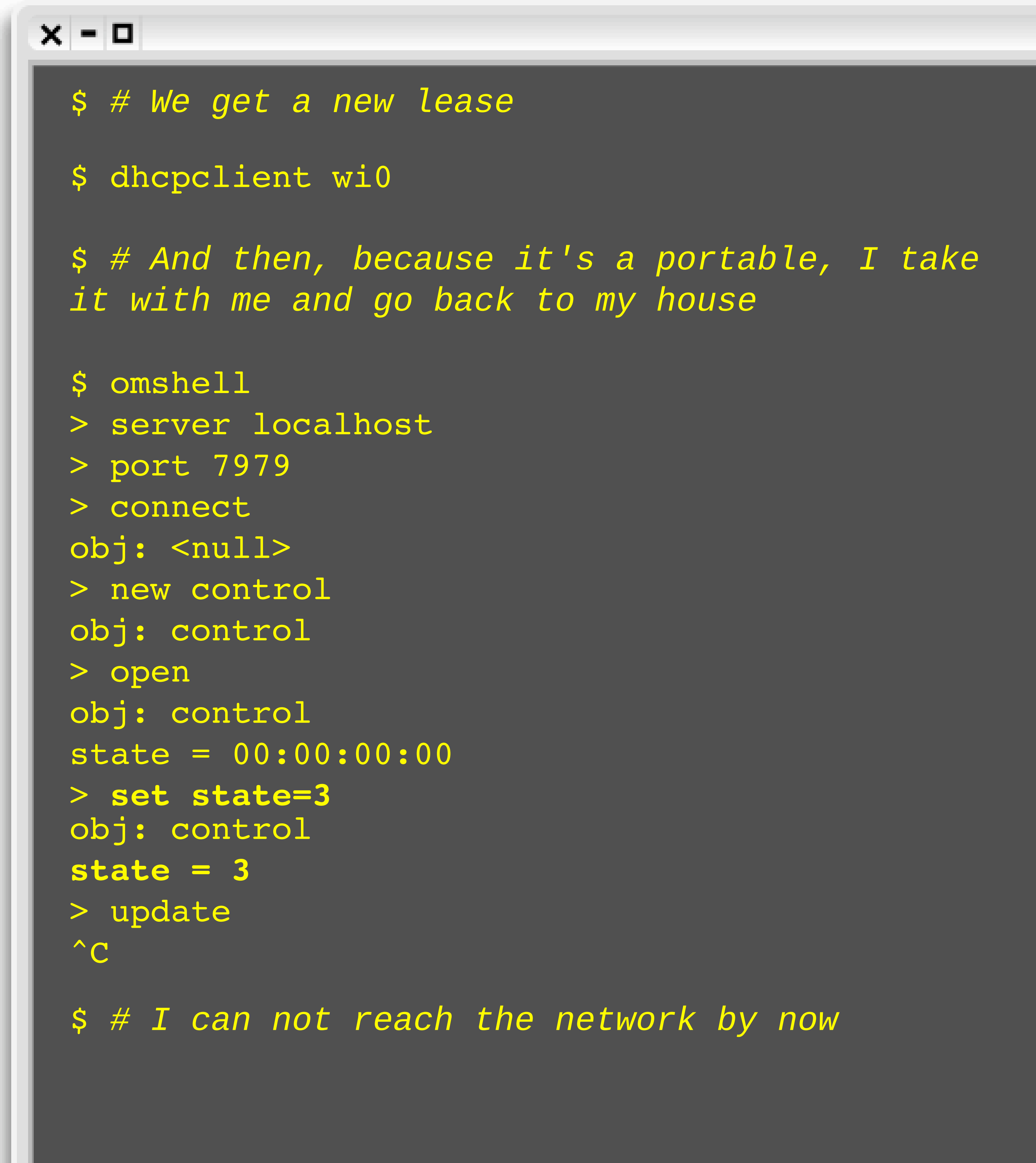
What about security?

- If you are concerned by security, you will wish some control access between omshell and dhcpclient
  - To this date (ISC-DHCP v3.0.1), UNFORTUNATELY, it SEEMS that the example shown on page 372 of the SAMS DHCP Handbook cannot work. First, the omapi key directive CANNOT work since ONLY the omapi port directive (that is, only the port 'qualifier') is accepted (this has been checked out in dhclient sources, and more specifically in the clparse.c file)
  - THEN, because using omshell, whatever the value of the key behind the key NAME_OF_KEY directive, one systematically gets an error message, either because of an unsufficient key length issue, or because of a supposedly erroneous syntax of every key (character supposedly incompatible with the base64 encoding)

Change client status

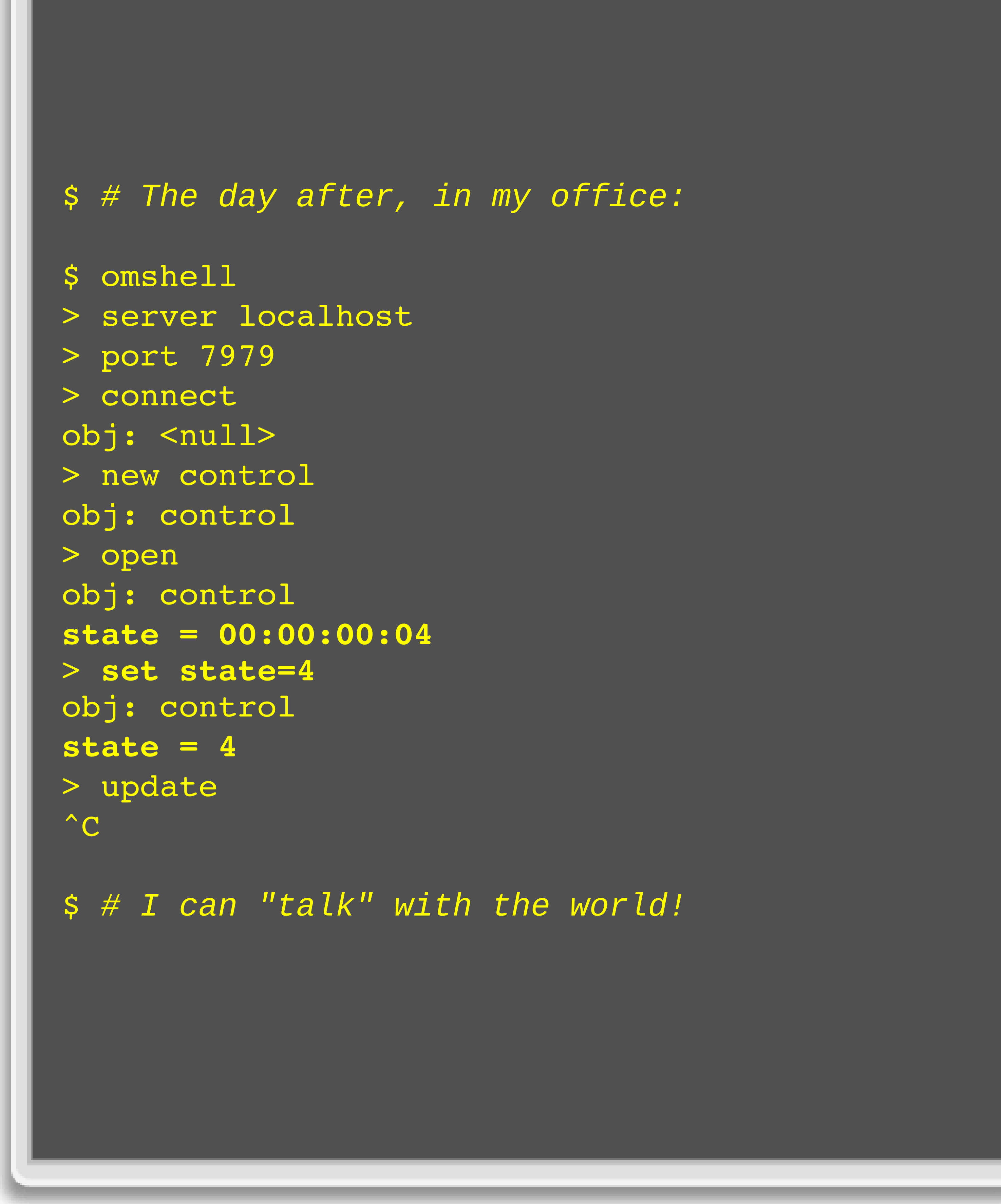
Let's practice... Again!

Hibernate



```
$ # We get a new lease

$ dhcpclient wi0

$ # And then, because it's a portable, I take
it with me and go back to my house

$ omshell
> server localhost
> port 7979
> connect
obj: <null>
> new control
obj: control
> open
obj: control
state = 00:00:00:00
> set state=3
obj: control
state = 3
> update
^C

$ # I can not reach the network by now
```

Awaken



```
$ # The day after, in my office:

$ omshell
> server localhost
> port 7979
> connect
obj: <null>
> new control
obj: control
> open
obj: control
state = 00:00:00:04
> set state=4
obj: control
state = 4
> update
^C

$ # I can "talk" with the world!
```

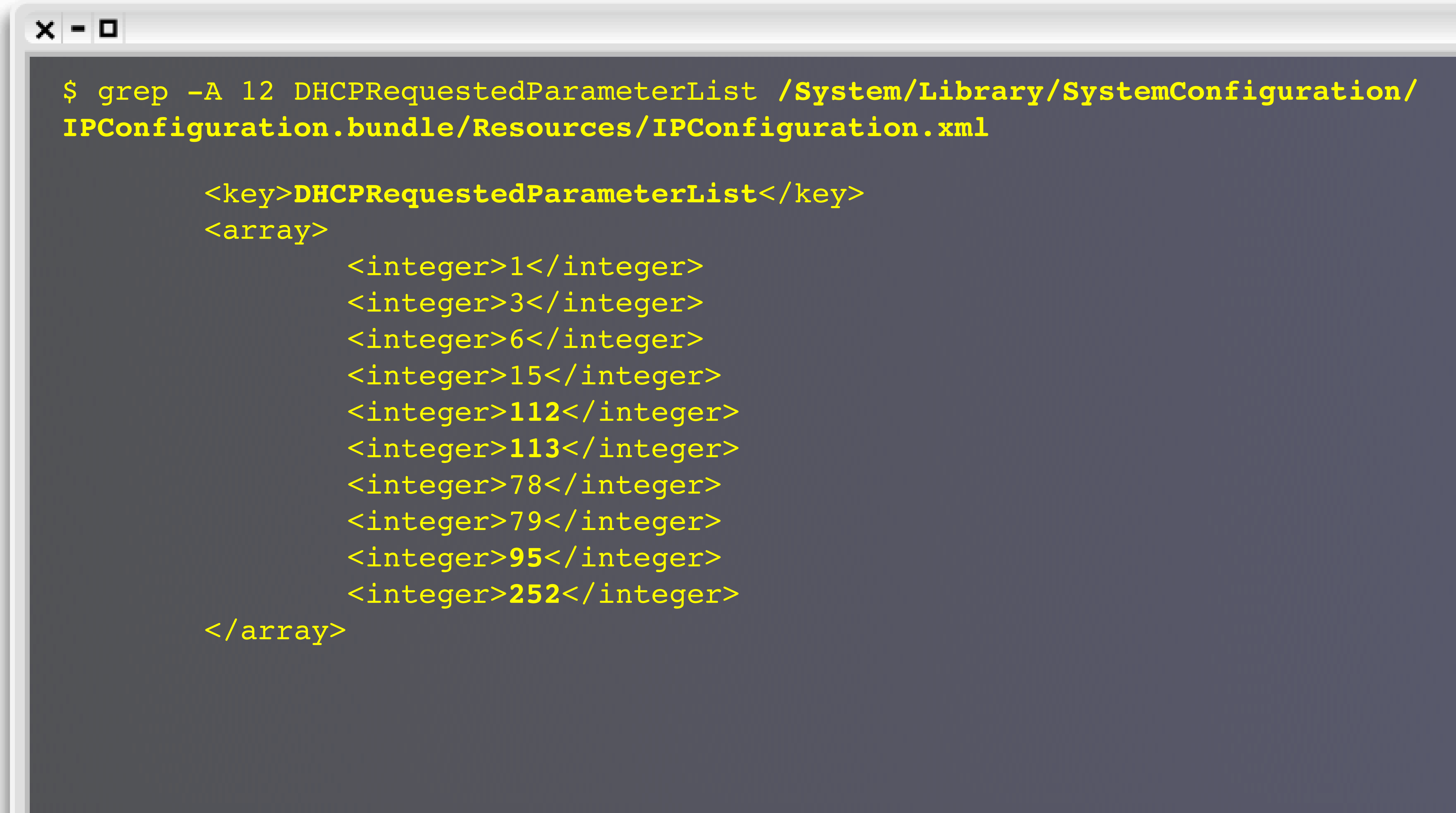
Excerpt from man dhclient (which sheds light on the former example):

> "The control object allows you to shut the client down, releasing all leases that it holds and deleting any DNS records it may have added. It also allows you to pause the client - this unconfigures any interfaces the client is using. You can then restart it, which causes it to reconfigure those interfaces. You would normally pause the client prior to going into hibernation or sleep on a laptop computer. You would then resume it after the power comes back. This allows PC cards to be shut down while the computer is hibernating or sleeping, and then reinitialized to their previous state once the computer comes out of hibernation or sleep.
>
> The control object has one attribute - the state attribute. To shut the client down, set its state attribute to 2. It will automatically do a DHCPRELEASE. To pause it, set its state attribute to 3. To resume it, set its state attribute to 4."

Special note on Mac OS X

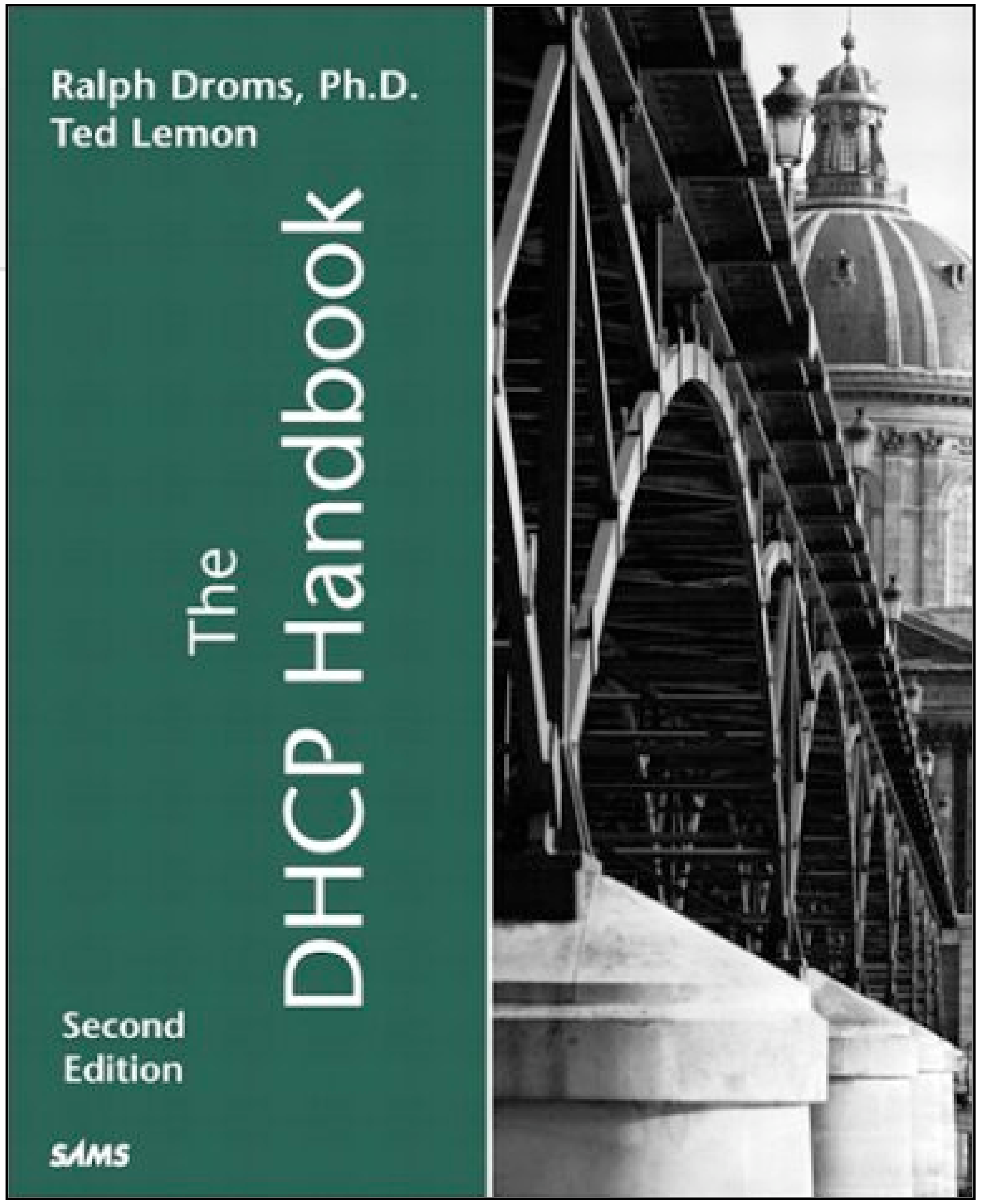
- In order to know the information returned by the DHCP server at any moment
  - ipconfig getpacket en1
  - ipconfig getoption "" router
    for a datum shared by all of the network interfaces
  - ipconfig getoption en1 lease_time
    for a datum specific to ONE network interface
  - ipconfig getoption en1 1
    by giving code rather than name of datum to be read

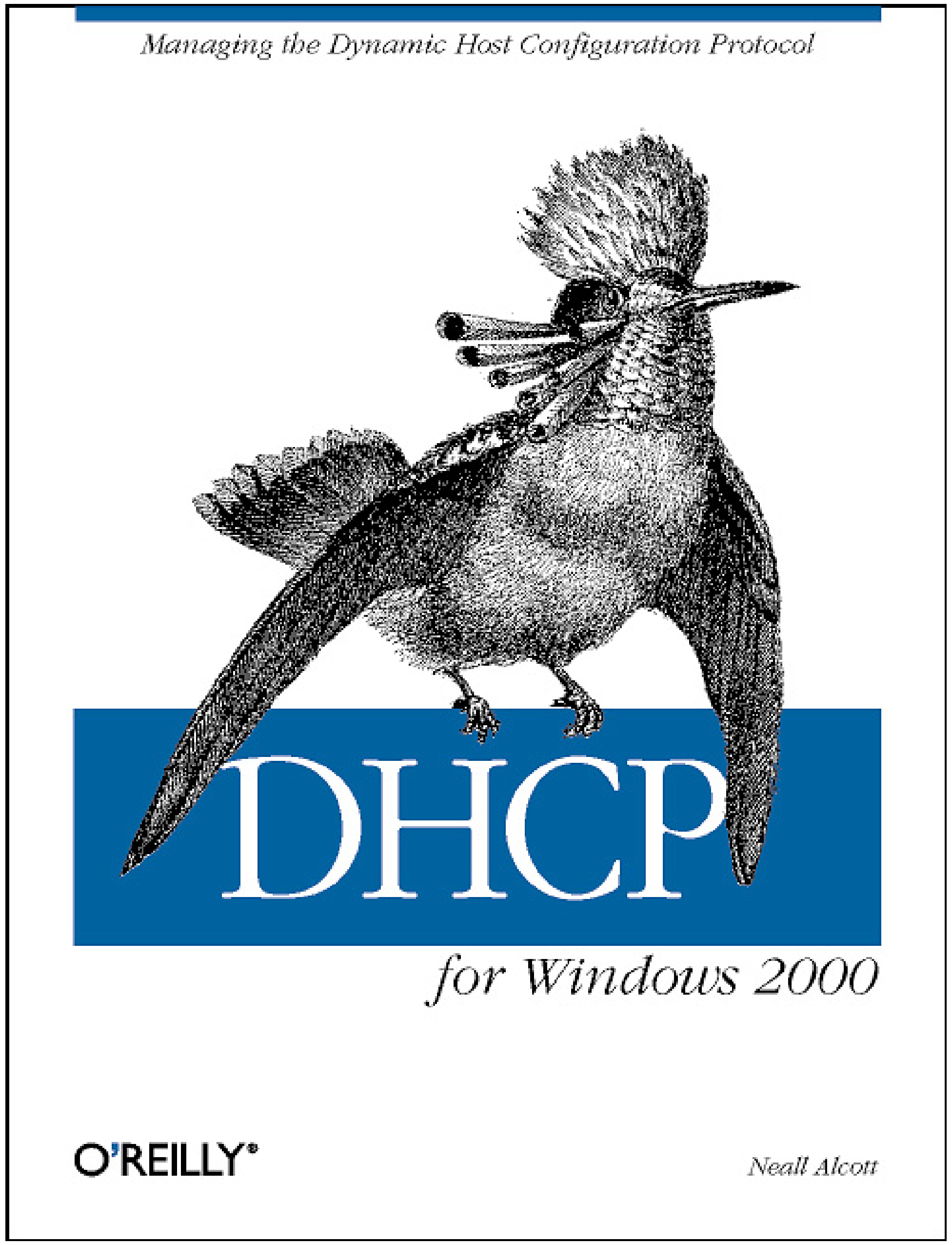
Mac OS X DHCP client configuration file



```
$ grep -A 12 DHCPRequestedParameterList /System/Library/SystemConfiguration/
IPConfiguration.bundle/Resources/IPConfiguration.xml

        <key>DHCPRequestedParameterList</key>
        <array>
                <integer>1</integer>
                <integer>3</integer>
                <integer>6</integer>
                <integer>15</integer>
                <integer>112</integer>
                <integer>113</integer>
                <integer>78</integer>
                <integer>79</integer>
                <integer>95</integer>
                <integer>252</integer>
        </array>
```

Bibliography





http://formation.diablotin.com