Basic TCP/IP in Linux

David Morgan
Third Annual Southern California Linux Expo
February 13, 2005

This presentation available at:

Configuring/using a network

- Concepts
- Manual configuration
- Automating config at bootup
- Using it

Concepts

- Packets
- Addresses
- Interfaces
- Routes

“Packets,” also known as:

- frames (esp. for ethernet and other datalink layer)
- datagrams (esp. for UDP and other transport layer)
- segments (esp. for TCP)
- packets (esp. for IP and other network layer)
- pdu’s (generally, “protocol data units”)

IP packet structure

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP’s Data Payload</td>
<td></td>
</tr>
</tbody>
</table>

TCP segment structure

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP’s Data Payload</td>
<td></td>
</tr>
</tbody>
</table>

© David Morgan 2003-2005
TCP/IP packet structure

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Destination Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
</tbody>
</table>

TCP’s Data Payload is a TCP packet

IP addresses

- 32 bit numbers
  - 11000000 10101000 00000100 00000001
- Expressed as “dot quads” or “dotted decimal”
  - 192.168.4.1

IP addresses - subnet masks

- Go with addresses
- Are also 32-bit numbers
- Operationally, like shoe sizes but for networks
  - they express the size of a network
- Netmask 255.255.255.248 is synonym for “network size is 8 addresses”

Common netmasks, small LANs

<table>
<thead>
<tr>
<th>How netmask is written</th>
<th>Size it indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>255.255.255.128 or /25</td>
<td>128 addresses</td>
</tr>
<tr>
<td>255.255.255.192 or /26</td>
<td>64</td>
</tr>
<tr>
<td>255.255.255.224 or /27</td>
<td>32</td>
</tr>
<tr>
<td>255.255.255.240 or /28</td>
<td>16</td>
</tr>
<tr>
<td>255.255.255.248 or /29</td>
<td>8</td>
</tr>
<tr>
<td>255.255.255.252 or /30</td>
<td>4</td>
</tr>
</tbody>
</table>

Interfaces

- Communication outlets to the external world
  - how many doors in your house?
  - how many interfaces in your box?
- Interface devices
  - ethernet cards /dev/eth0, /dev/eth1…
  - modems (point-to-point) /dev/ppp0, ...
  - exotic /dev/isdn0, /dev/fddi0

Routes

- Electronic location of other computers
- By IP address
- Via interfaces
- routes map addresses into interfaces
Routing – IPdest-Iface correlation

Maintained in a “routing table”:

```
[root@EMACH1 /root]# route
Kernel IP routing table

Destination     Gateway Genmask    Iface
209.233.193.22   *           255.255.255.255  ppp0
192.168.4.0      *           255.255.255.0    eth0
default          209.233.193.22  0.0.0.0    ppp0

[root@EMACH1 /root]#
```

Analogy – airport departure board

![Analogy - airport departure board](image)

Commands to config networks

- Older collection of special-purpose commands
  - `ifconfig` (for setting up addresses)
  - `route` (for setting up routes)
  - `others` (arp, netstat...)
- Newer rewritten umbrella command “ip”
  - “ip address” alternative equivalent to `ifconfig`
  - “ip route” alternative to `route`
  - “ip neighbor” alternative to `arp`
- Old commands implemented elsewhere, but “ip” is linux-only

ifconfig command

- manually configuring interfaces
  - View interface status
    - `ifconfig -a`
  - Set interface characteristics
    - `ifconfig eth0 192.168.4.1`

or “ip address” command

- manually configuring interfaces
  - View interface status
    - `ip address show`
  - Set interface characteristics
    - `ip address add 192.168.4.1 dev eth0`
“ip address” command

<table>
<thead>
<tr>
<th>File</th>
<th>Edit</th>
<th>View</th>
<th>Terminal</th>
<th>Tab</th>
<th>Help</th>
</tr>
</thead>
</table>

- `ifconfig`: `ifconfig eth0` to view current settings
- `ip addr show`: `ip addr show dev eth0`

**route command**

--- manually configuring routes

- **host route** - to a single machine
  - `route add -host 192.168.4.2 eth0`
- **network route, local** - to a group of machines
  - `route add -net 192.168.4.0 netmask 255.255.255.0 eth0`
- **network route, thru gateway** - to a group of machines
  - `route add -net 192.168.5.0 netmask 255.255.255.0 gw 192.168.4.1`
- **default route** - to “any and all” else
  - `route add default gw 192.168.4.1`

or “ip route” command

--- manually configuring routes

- **host route** - to a single machine
  - `ip route add 192.168.4.2 dev eth0`
- **network route, local** - to a group of machines
  - `ip route add 192.168.4.0/24 dev eth0`
- **network route, thru gateway** - to a group of machines
  - `ip route add 192.168.5.0/24 via 192.168.4.1`
- **default route** - to “any and all” else
  - `ip route replace default via 192.168.4.1`

Great. But that’s too hard.

- Can’t somebody else run `ifconfig/route` for me?
- To the rescue: pre-written scripts do it!
- You just feed them the values to use

Boot time automation scripts

- **Initialization script**: `/etc/rc.d/init.d/network`
- `/etc/sysconfig/network-scripts/ifup`

```
[Excerpts, Fedora3’s “network” initscript, line 98 ff.]
```

/`etc/rc.d/init.d/network`

Calls “ifup” script for each interface

```
# bring up interfaces configured to come up at boot time
for i in $interfaces; do
  action "$Bringing up interface $i: "; ./ifup $i boot
Done
```

Establishes gateway

```
ip route replace default via GATEWAY...  
...from next slide
```

[Excerpts, Fedora3’s “network” initscript, line 98 ff.]
/etc/sysconfig/network

Sets environment variables to values the scripts use for guidance

\[
\begin{align*}
\text{NETWORKING} &= \text{yes} \\
\text{FORWARD_IPV4} &= \text{no} \\
\text{GATEWAY} &= 192.168.3.1
\end{align*}
\]

Network config control at bootup

- Edit the network/ifcfg-ethX files yourself
- Use an admin tool, which does the same thing
  - `/usr/sbin/system-config-network (Fedora)
  - webmin

Fedora's system-config-network

It's up. What can you do with it?

- Test it - ping
- Watch it – tcpdump
- Interfere with it - iptables
- Work with others - services

/etc/sysconfig/network

Reads settings from ifcfg-ethX, configures interface and routes

\[
\begin{align*}
\text{if} \ ! \ \text{LC_ALL=C ip addr ls} \ $(\text{REALDEVICE}) \ | \ \text{LC_ALL=C grep -q} \\
\quad \ "\$(\text{IPADDR})/\$(\text{PREFIX})" \ ; \ \text{then} \\
\quad \ \text{if} \ ! \ \text{ip addr add} \ $(\text{IPADDR})/\$(\text{PREFIX}); \ \text{then} \\
\quad \quad \text{echo "Error adding address $(IPADDR) for $(DEVICE)."} \\
\quad \ \text{fi} \\
\quad \ \text{fi} \\
\end{align*}
\]

PSEUDOCODE

if <the interface doesn’t already have an address> ; then
  if <trying to give it one fails>; then
    <print error message>
  endif
endif

Network config control at bootup

- Edit the network/ifcfg-ethX files yourself
- Use an admin tool, which does the same thing
  - `/usr/sbin/system-config-network (Fedora)
  - webmin

Fedora's system-config-network

It’s up. What can you do with it?

- Test it - ping
- Watch it – tcpdump
- Interfere with it - iptables
- Work with others - services

/proc/sys/net/ipv4/ip_forward

Sets the ip_forward flag to 1 to enable packet forwarding

/proc/sys/net/ipv4/ip_tables

Sets up iptables rules to manage traffic

Network config control at bootup

- Edit the network/ifcfg-ethX files yourself
- Use an admin tool, which does the same thing
  - `/usr/sbin/system-config-network (Fedora)
  - webmin

Fedora's system-config-network

It’s up. What can you do with it?

- Test it - ping
- Watch it – tcpdump
- Interfere with it - iptables
- Work with others - services

/proc/sys/net/ipv4/ip_forward

Sets the ip_forward flag to 1 to enable packet forwarding

/proc/sys/net/ipv4/ip_tables

Sets up iptables rules to manage traffic

Network config control at bootup

- Edit the network/ifcfg-ethX files yourself
- Use an admin tool, which does the same thing
  - `/usr/sbin/system-config-network (Fedora)
  - webmin

Fedora's system-config-network

It’s up. What can you do with it?

- Test it - ping
- Watch it – tcpdump
- Interfere with it - iptables
- Work with others - services
ping: the “Hey! You there?” utility

- **purpose:** Tests connectivity
- **method:** Probes an address
- **output:** Reports whether there is a reply

```
[root@EMACH1 /root]# ping -c3 66.218.71.81
PING 66.218.71.81 (66.218.71.81) from 64.130.228.61 : 56(84) bytes of data.
64 bytes from 66.218.71.81: icmp_seq=0 ttl=55 time=34.5 ms
64 bytes from 66.218.71.81: icmp_seq=1 ttl=55 time=33.6 ms
64 bytes from 66.218.71.81: icmp_seq=2 ttl=55 time=34.1 ms
--- 66.218.71.81 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 33.6/34.0/34.5 ms
```

…so we know 66.218.71.81 is alive

```
[root@EMACH1 /root]# ping -c3 66.218.71.17
PING 66.218.71.17 (66.218.71.17) from 64.130.228.61 : 56(84) bytes of data.
--- 66.218.71.17 ping statistics ---
3 packets transmitted, 0 packets received, 100% packet loss
```

…so we don’t know if 66.218.71.17 is alive

```
tcpdump -i <interface>
```

```
iptables -t filter -A INPUT -i eth1 -p tcp --sport 1024:65535 --dport 23 -s 0.0.0.0/0 -d 192.168.4.1/32 -j ACCEPT
iptables -t filter -A OUTPUT -o eth1 -p tcp --sport 23 --dport 1024:65535 -s 192.168.4.0/24 -d 0.0.0.0/0 -j ACCEPT
iptables -t filter -P INPUT DROP
iptables -t filter -P OUTPUT DROP
```

A 4-rule filtering firewall

Executed in chronological sequence as shown, resultant 2-chain firewall permits telnet access between this machine 192.168.4.1 and others via eth1. And nothing else.

```
iptables -t nat -A POSTROUTING -o eth1 -p tcp --dport 23 -j SNAT --to 216.83.185.193
iptables -t nat -A POSTROUTING -o eth1 -p tcp --dport 5631 -j DNAT --to 216.83.185.193
```

nat table: rules that alter packet

- Masquerading
  - iptables -t nat -A POSTROUTING -o eth1 -p tcp --dport 23 -j SNAT --to 216.83.185.193
- Pinholing (port forwarding)
  - iptables -t nat -A POSTROUTING -o eth1 -p tcp --dport 5631 -j DNAT --to 216.83.185.193
IP masquerading

Outbound packet:

Reply:

Distinction: machine vs process

Ports and conversations

Each process has its own number, called a port number. Stating a port tells which process you want to talk to. Basis for services.

Biblio

- “IP Command Reference,” Alexey Kuznetsov (run “gv $locate ipref.ps” in your Linux GUI)
- http://www.tcpdump.org/
- http://www.iptables.org/