Basic TCP/IP in Linux

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

This presentation available at:
http://members.dslextreme.com/~dmorgan1/scale2005-networks.pdf

Configuring/using a network

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

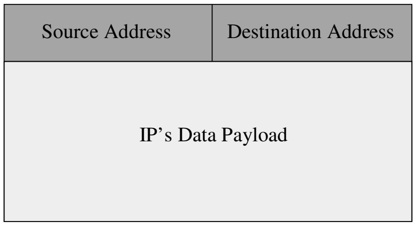
Concepts

- Packets
- Addresses
- Interfaces
- Routes

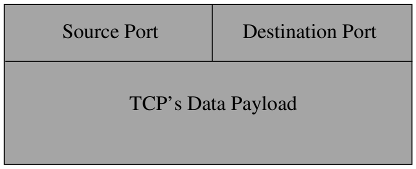
“Packets,” also known as:

● 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”)

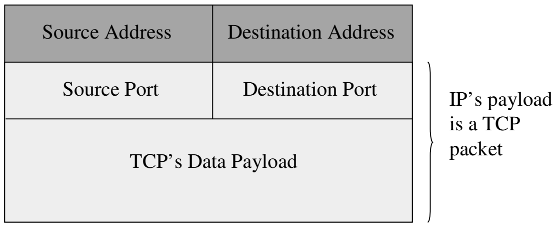
IP packet structure



TCP segment structure



TCP/IP packet structure



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 *siz*e of a network
- Netmask 255.255.255.248 is synonym for "network size is 8 addresses"

Common netmasks, small LANs

How netmask is written	Size it indicates
255.255.255.128 or /25	128 addresses
255.255.255.192 or /26	64
255.255.255.224 or /27	32
255.255.255.240 or /28	16
255.255.255.248 or /29	8
255.255.255.252 or /30	4

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

Departure board

Destination	Gate
Phoenix	33A
Portland	36B
international	Terminal 4

Local, not outside of airport

Local, not outside of computer

Routing table

Destination	Interface
209.233.193.22 /32	ppp0
192.168.4.0 /24	eth0

Commands to config networks

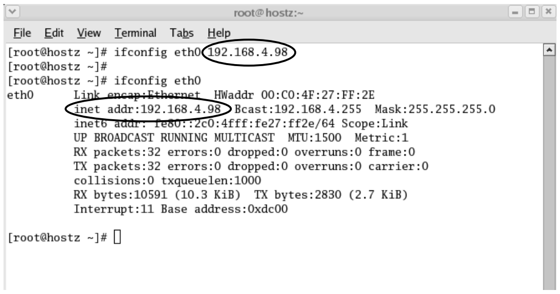
- Older collection of special-purpose commands
  - ifconfig (for setting up addresses)
  - route (for setting up routes)
  - others (arp, netstat…)
- Newer rewritten umbrella command “ip”
  - “ip address” alternative equivalent to ifconfig
  - “ip route” alternative to route
  - “ip neighbor” alternative 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

ifconfig command



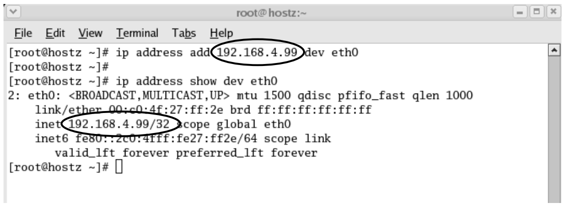
```
[root@hostz ~]# ifconfig eth0 192.168.4.98
[root@hostz ~]#
[root@hostz ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:C0:4F:27:FF:2E
          inet addr:192.168.4.98  Bcast:192.168.4.255  Mask:255.255.255.0
          inet6 addr: fe80::2c0:4fff:fe27:ff2e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:32 errors:0 dropped:0 overruns:0 frame:0
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10591 (10.3 KiB)  TX bytes:2830 (2.7 KiB)
          Interrupt:11 Base address:0xdc00

[root@hostz ~]#
```

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



```
[root@hostz ~]# ip address add 192.168.4.99 dev eth0
[root@hostz ~]#
[root@hostz ~]# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:c0:4f:27:ff:2e brd ff:ff:ff:ff:ff:ff
    inet 192.168.4.99/32 scope global eth0
    inet6 fe80::2c0:4fff:fe27:ff2e/64 scope link
       valid_lft forever preferred_lft forever
[root@hostz ~]# 
```

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

informed by

- etc/sysconfig/network
- /etc/sysconfig/network-scripts/ifcfg-ethX

/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

```
NETWORKING=yes
FORWARD_IPV4=no
GATEWAY=192.168.3.1
```

to previous slide…

/etc/sysconfig/network-scripts/ifup

Reads settings from ifcfg-ethX, configures interface and routes

```
 if ! LC_ALL=C ip addr ls ${REALDEVICE} | LC_ALL=C grep -q
"${IPADDR}/${PREFIX}" ; then
   if ! ip addr add ${IPADDR}/${PREFIX}…; then
     echo $"Error adding address ${IPADDR} for ${DEVICE}."
   fi
 fi
 [ Fedora3's "ifup" script, line 383 ff. ]
```

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

/etc/sysconfig/ network-scripts/ifcfg-eth0

Sets environment variables to values the scripts use for guidance

```
BOOTPROTO=none
DEVICE=eth0
ONBOOT=yes
IPADDR=192.168.3.2
NETMASK=255.255.255.0
```

-or-

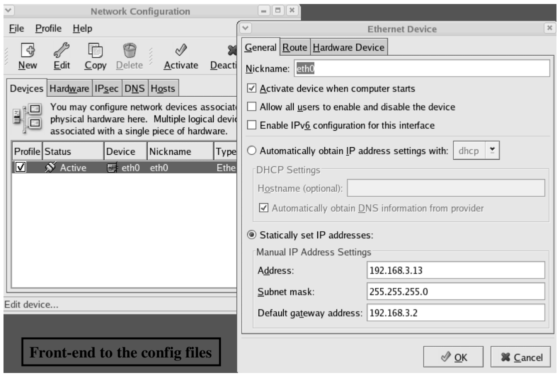
```
BOOTPROTO=dhcp
DEVICE=eth0
ONBOOT=yes
```

Documentation: /usr/share/doc/initscripts-7.93.2/sysconfig.txt

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



You may configure network devices associat physical hardware here. Multiple logical devi associated with a single piece of hardware.

Profile	Status	Device	Nickname	Type
☑	Active	eth0	eth0	Ethe

Edit device...

Nickname: eth0

☑ Activate device when computer starts

☐ Allow all users to enable and disable the device

☐ Enable IPv6 configuration for this interface

○ Automatically obtain IP address settings with: dhcp

DHCP Settings

Hostname (optional):

☑ Automatically obtain DNS information from provider

● Statically set IP addresses:

Manual IP Address Settings

Address: 192.168.3.13

Subnet mask: 255.255.255.0

Default gateway address: 192.168.3.2

Front-end to the config files

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

ping usage

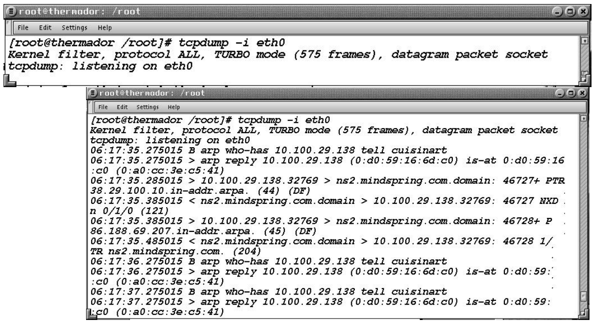
```
[root@EMACH1 /root]# ping -c3 66.218.71.81
PING 66.218.71.81 (66.218.71.81) from 64.130.228.61 : 56(84) bytes of data.
64 bytes from 66.218.71.81: icmp_seq=0 ttl=55 time=34.5 ms
64 bytes from 66.218.71.81: icmp_seq=1 ttl=55 time=33.6 ms
64 bytes from 66.218.71.81: icmp_seq=2 ttl=55 time=34.1 ms

--- 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
[root@EMACH1 /root]#
```

…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
[root@EMACH1 /root]#
```

…so we don’t know if 66.218.71.17 is alive

tcpdump -i <interface>



```
[root@thermador /root]# tcpdump -i eth0
Kernel filter, protocol ALL, TURBO mode (575 frames), datagram packet socket
tcpdump: listening on eth0
```

```
[root@thermador /root]# tcpdump -i eth0
Kernel filter, protocol ALL, TURBO mode (575 frames), datagram packet socket
tcpdump: listening on eth0
06:17:35.275015 B arp who-has 10.100.29.138 tell cuisinart
06:17:35.275015 > arp reply 10.100.29.138 (0:d0:59:16:6d:c0) is-at 0:d0:59:16
:c0 (0:a0:cc:3e:c5:41)
06:17:35.285015 > 10.100.29.138.32769 > ns2.mindspring.com.domain: 46727+ PTR
38.29.100.10.in-addr.arpa. (44) (DF)
06:17:35.385015 < ns2.mindspring.com.domain > 10.100.29.138.32769: 46727 NXD
n 0/1/0 (121)
06:17:35.385015 > 10.100.29.138.32769 > ns2.mindspring.com.domain: 46728+ P
86.188.69.207.in-addr.arpa. (45) (DF)
06:17:35.485015 < ns2.mindspring.com.domain > 10.100.29.138.32769: 46728 1/
TR ns2.mindspring.com. (204)
06:17:36.275015 B arp who-has 10.100.29.138 tell cuisinart
06:17:36.275015 > arp reply 10.100.29.138 (0:d0:59:16:6d:c0) is-at 0:d0:59:
:c0 (0:a0:cc:3e:c5:41)
06:17:37.275015 B arp who-has 10.100.29.138 tell cuisinart
06:17:37.275015 > arp reply 10.100.29.138 (0:d0:59:16:6d:c0) is-at 0:d0:59:
:c0 (0:a0:cc:3e:c5:41)
```

```
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
```

– Table for this rule

– Rule action
- -A add rule to chain/list
- -D delete rule from chain/list
- -P default policy for chain/list

– Rule chain/list (tables contain chains)
- INPUT
- OUTPUT
- FORWARD
- PREROUTING
- POSTROUTING

– Packet qualifiers
- By interface and direction
- protocol
- source port number(s)
- destination port number(s)
- source address (range)
- destination address (range)

– Packet disposition
- ACCEPT
- DROP
- REJECT
- SNAT
- DNAT

A 4-rule filtering firewall

```
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 192.168.4.1/32 -d 0.0.0.0/0 –j ACCEPT

iptables -t filter -P INPUT DROP

iptables -t filter -P OUTPUT DROP
```

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.

nat table: rules that alter packet

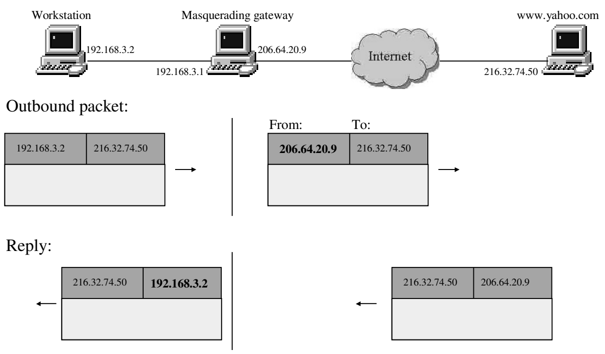
- Masquerading

```
iptables -t nat -A POSTROUTING
              -o eth1 -s 10.0.0.0/8
                          -j SNAT --to 216.83.185.193
```

- Pinholing (port forwarding)

```
iptables -t nat -A PREROUTING
              -i eth1 -d 216.83.185.193/32 -p tcp --dport 5631
                          -j DNAT --to 216.83.185.193
```

IP masquerading



Workstation

Masquerading gateway

www.yahoo.com

192.168.3.2

192.168.3.1

206.64.20.9

Internet

216.32.74.50

Outbound packet:

192.168.3.2	216.32.74.50

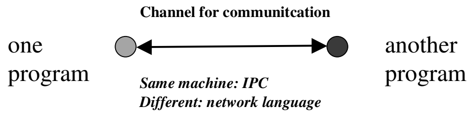
From:	To:
206.64.20.9	216.32.74.50

Reply:

216.32.74.50	**192.168.3.2**

216.32.74.50	206.64.20.9

Work with others: services



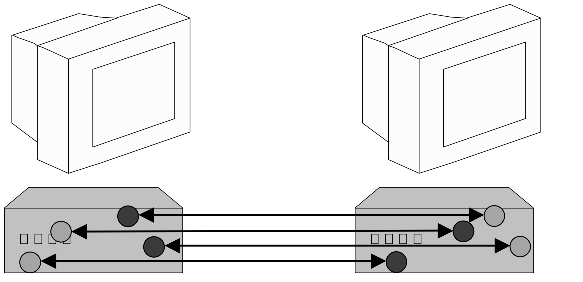
Channel for communitcation

one program

another program

Same machine: IPC
Different: network language

Distinction: machine vs process



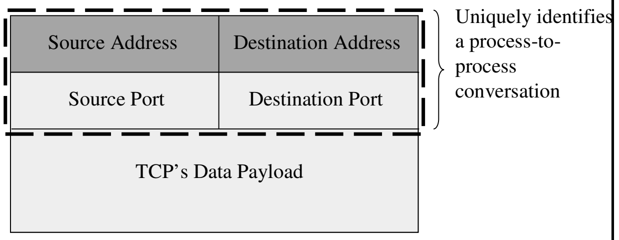
ftp server
mail client
web server
telnet client

● Client processes

● Server processes

ftp client
mail server
web client
telnet server

Ports and conversations



Source Address	Destination Address
Source Port	Destination Port

TCP's Data Payload

Uniquely identifies a process-to-process conversation

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 ipcref.ps)" in your linux GUI)
- The Linux Network Administrator's Guide, Olaf Kirsch (http://www.tldp.org/LDP/sag/html/index.html)
- http://www.tcpdump.org/
- http://www.iptables.org/

This presentation available at:
http://members.dslextreme.com/~dmorgan1/scale2005-networks.pdf