ConvexOS Network Administration Course Notes

CONVEX COMPUTER CORPORATION

P-Lw

ConvexOS Network Administration Course Notes

CONVEX Education Center

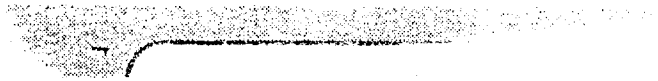
February 3, 1993

CONVEX Computer Corporation

Printed in the United States of America

Basic Networking

CONVEX COMPUTER CORPORATION



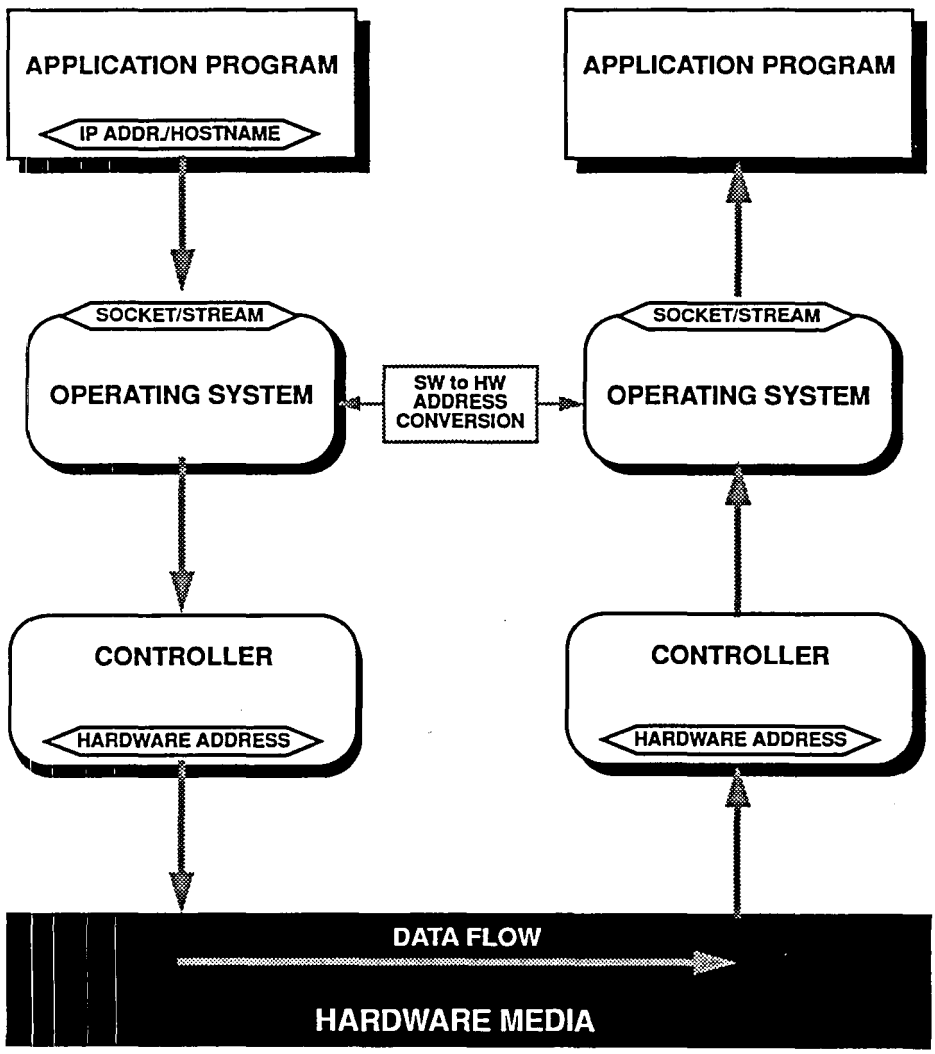
Topics:

- References
- Network Flow
- Protocol Layers
- Software Addresses
- Internet Addresses
- Address Classes
- Dot Notation
- Special Addresses
- Building Internet Addresses
- Device Interfaces
- Network Hardware Configuration [*/ioconfig*]
- Associating Network Interfaces and IP Addresses [*ifconfig(8c)*]
- Obtaining network status [*netstat(1c)*]
- Hostnames
- Hosts file [*/etc/hosts*]
- Associating a name with a host, [*hostname(1)*]
- Networks file [*/etc/networks*]
- Determining if a network interface is reachable [*ping(8c)*]
- Configuring Network Interfaces [*/etc/rc.local*]
- Listing trusted hosts [*/etc/hosts.equiv*, *.rhosts*]

References

- Convex Internet Services System Manager's Guide (chapters 2 – 6)
- Convex Network File System, System Manager's Guide
- Convex Networking Concepts
- ConvexOS Man Pages:
  - *ioconfig(4)*
  - *lo(4)*
  - *ex(4)*
  - *hy(4)*
  - *hosts(5)*
  - *ping(1c)*
  - *pong(8)*
  - *hostname(1)*
  - *netstat(1c)*

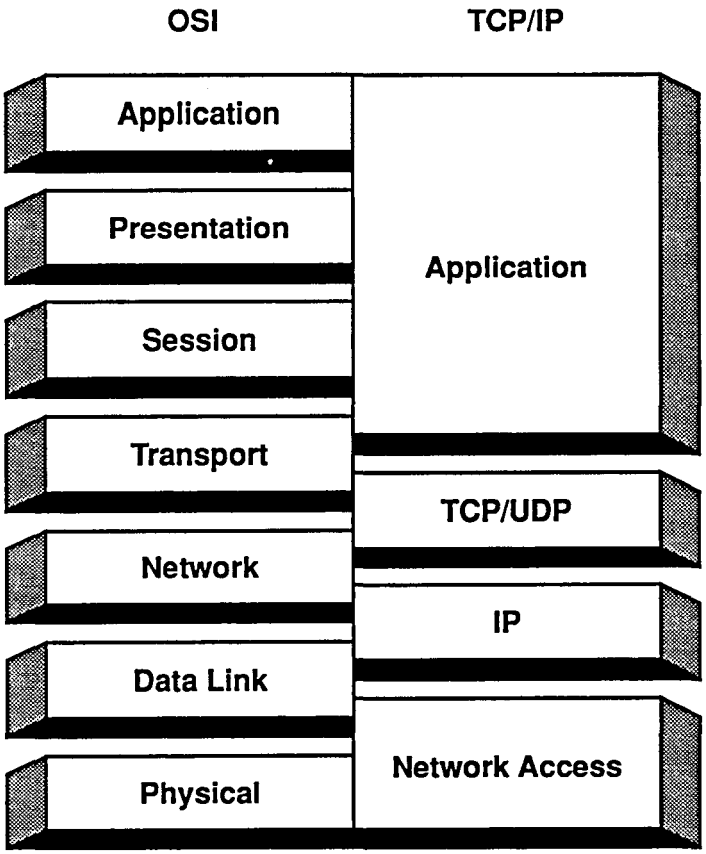
Network Flow



Protocol Layers

There are many different protocol hierarchies, the most common being the Open Systems Interconnect (OSI) model, and the Transmission Control Protocol/Internet Protocol (TCP/IP) model.

Although there is not an exact correspondence between them, this is an approximation:



Software Addresses

- Internet Addresses
  - used by all TCP/IP implementations
  - 32-bit [unsigned] integer which uniquely defines an interface on a network
  - interpreted by the software as a network number and a host number
  - automatically translated into a hardware address by the software
  - unique address required for each hardware interface
- Other address types for other protocols

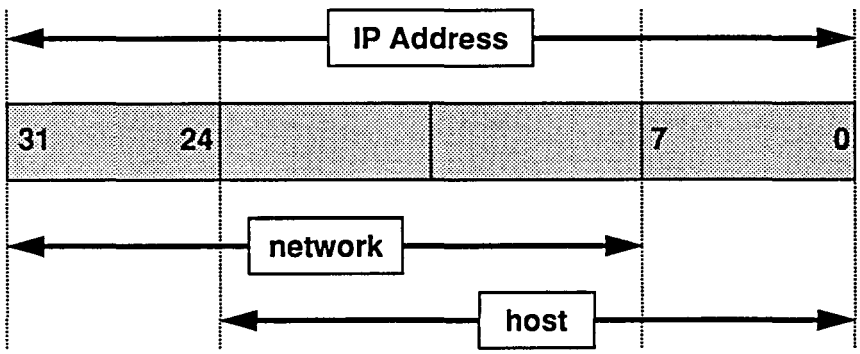
Note: Users usually deal with hostnames rather than internet addresses directly, although many applications will accept either.

Internet Addresses

Internet addresses (IP address) are 32-bit numbers divided into two components:

1. network number – 8 to 24 bits used for routing
2. host number – 8 to 24 bits used to identify an individual host's interface within a network[1]

Address field ranges:



1. A host on the network may have multiple network interfaces; each interface will have a unique interface number.

Address Classes

An IP address is divided into classes which identify how many bits in the address are allocated to the network number and how many bits are allocated to the host number.

Address classes:

- are determined by the upper 5 bits of the IP address
- determine (by the bit allocations and convention) how many networks and how many interfaces can belong to any class

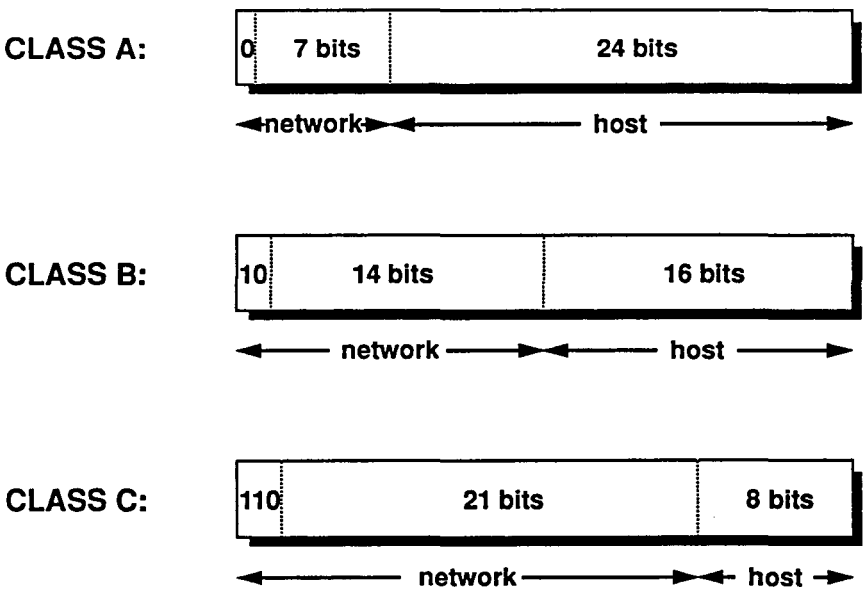
Upper 5 bits					Address Class	Bits for Network #	Possible Networks	Network # First Octet Value Range	Bits for Host #	Possible Hosts *
0	-	-	-	-	A	7	126 **	0 - 127 **	24	16,777,124
1	0	-	-	-	B	14	16,384	128 - 191	16	65,534
1	1	0	-	-	C	21	2,097,152	192-223	8	254
1	1	1	0	-	D	multicast – remaining 28 bits used to identify a host group				
1	1	1	1	0	E	reserved for future use				

* This number accounts for a host number of all 0 binary digits being reserved for netmasks and a host number of all 1 binary digits being reserved for broadcasts.

** Network numbers 0 and 127 are reserved so there are effectively 126 usable values, even though there are 128 possible values.

Address Classes (cont'd)

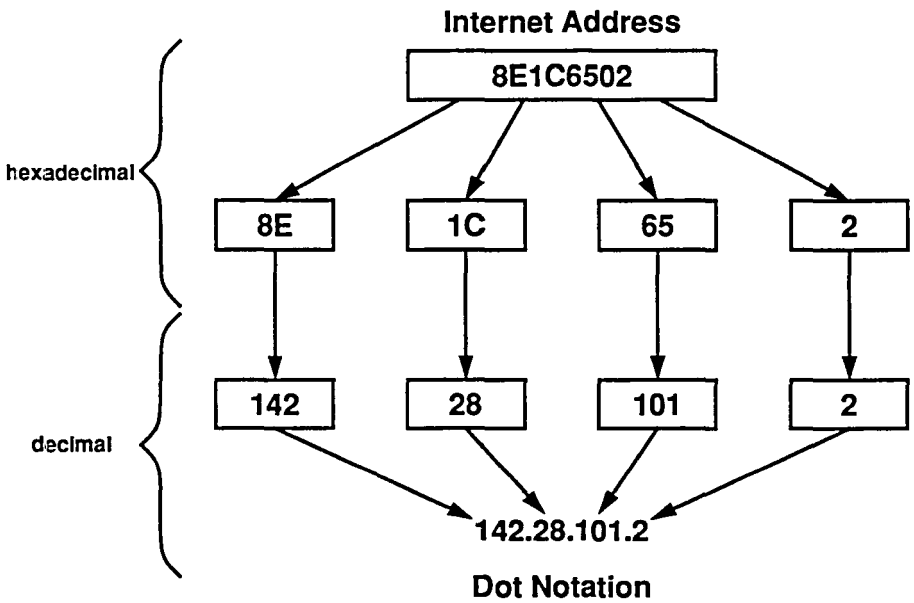
The 32-bit IP address is partitioned this way:



Dot Notation

Dot notation is an alternate method of specifying a 32-bit IP address:

- makes reading the IP address easier (for people)
- divides the 32-bit IP address into 4 8-bit sections, called "octets"
- uses a decimal representation for each octet
- accepted by most TCP/IP applications when an IP address is required
- appears in many configuration files (*/etc/hosts*, */etc/networks*, ...)



Exercises

[1] A host's IP address is 0xC142650E. What is the dot notation for this address?

_____

[2] What class address is this?

_____

[3] What is the range (in decimal) of the network portion of class A addresses?

_____

Class B? _____

Class C? _____

Special Addresses

There are two types of addresses which have special uses:

1. loopback
   - interface that allows networking commands to work on the local host
   - configured after all other network interfaces
   - has IP address 127.0.0.1[1]

   - interface name is *lo0*

1. This is by convention; there is no Request For Comments (RFC) specifying the network address for loopback.

Special Addresses (cont'd)

2. broadcast
   - a "wildcard" address, to send to all hosts on all networks

   - the networks receiving the broadcast can be restricted to a single network by providing a network number in the network portion of the address, and wildcarding the host portion

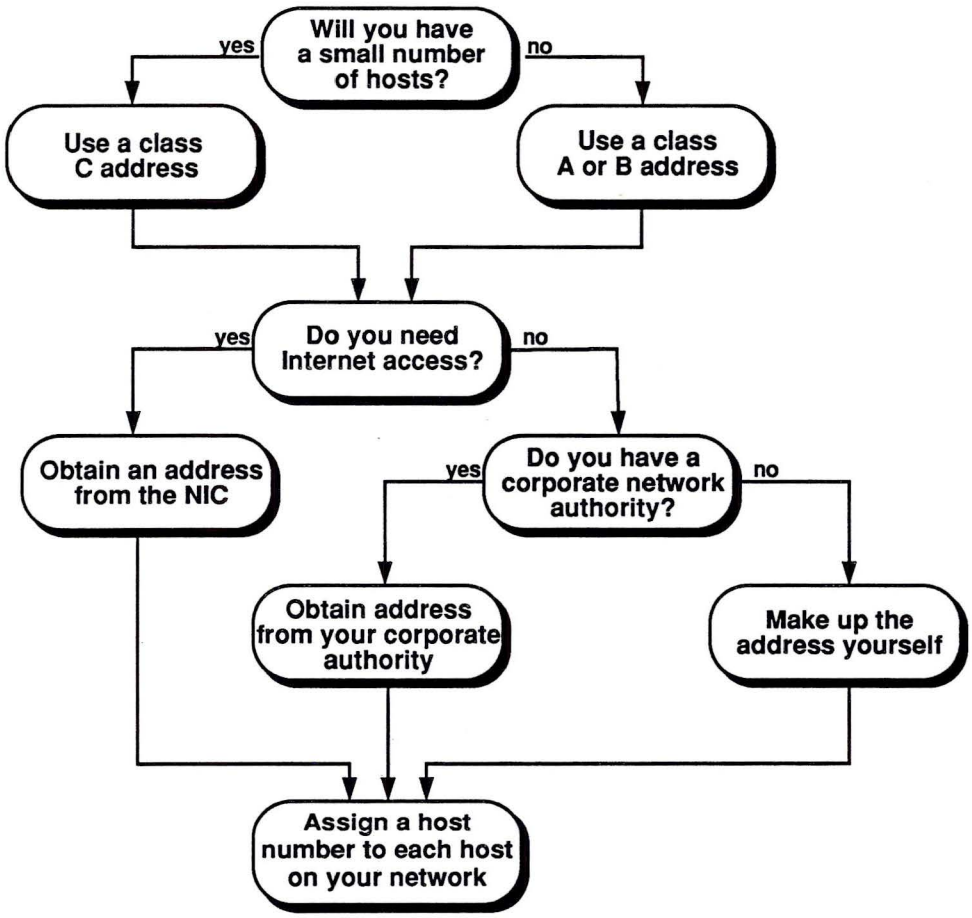
     this form of "restricted" broadcast is the default broadcast mode set by *ifconfig(8c)*, using the interface's network as the broadcast network

Building Internet Addresses

1. Determine your network needs
   - will you have many or few hosts?
   - will you need access to the Internet?
   - will you be expanding your network?
2. Determine your network layout
   - will subnets be used?
   - do high traffic areas need to be isolated?
   - are you using any vendor's product that limits your network configuration options?[1]
3. Obtain a network address for your installation.
   - do you need to contact the Network Information Center (NIC)?
   - can you determine network numbers within your corporation?
4. Assign an IP address for each host on your network using the network address you obtained.

1. Some software does not allow the same device to be both a router and network application host. There may also be other hardware constraints.

Building Internet Addresses (cont'd)



Device Interfaces

A device interface is a table entry in the kernel which is accessed either:

- by user code through a file system entry in */dev*
- directly by the kernel

The corresponding hardware devices are listed on the SPU in */ioconfig*.

If the interface is accessed through the file system, the file system entry is created using *mknod(2)*. This is usually done with the shell script */dev/MAKEDEV*.

Device Interfaces (cont'd)

Each type interface is referenced with a different device name prefix. The suffix ($n = 0 - 7$) identifies the unit number.

- **ex***n*
  - ethernet
  - Multibus (Excelan), VMEbus, compatible with IEEE 802.3
  - TCP/IP, DECnet, OSI WAN protocols
  - 10 megabits per second
  - up to 500 meters
  - DECnet applications use the */dev* interface; TCP/IP applications use an internal kernel interface
- **fd***n*
  - Fiber Distributed Data Interface (FDDI)
  - VMEbus, complies with ANSI X3T9.5 and ISO 9314
  - TCP/IP
  - 100 megabits per second
  - 2 kilometers
- **tty***XY*
  - ASYNC terminal lines, Multibus, VMEbus
  - Serial Line Internet Protocol (SLIP)
  - 19.2 kilobits per second
  - distance depends on cabling

Device Interfaces (cont'd)

- **hy***n*
  - hyperchannel
  - Multibus (IKON 10077-NSC & Network Systems A400)
  - VMEbus (IKON 10090 & Network Systems NB400)
  - 50 megabits per second
  - distance depends on cabling
  - TCP/IP (RFC 1044 – "Internet Protocol on Network System's HYPERchannel: Protocol Specification")

- **unet***n*
  - ultranet
  - VMEbus (VME/UltraNet controller, Data Link Interface, shielded cable)
    - complies with IEEE 802.2 LLC and TP4
    - 120 megabits per second
    - 107 meters
  - High Performance Parallel Interface (HiPPI)
    - complies with ANSI X3T9.3
    - fastest industry standard, 800 megabits per second
    - 25 meters
  - TCP/IP, UltraNet proprietary protocol

Network Hardware Configuration [*/ioconfig*]

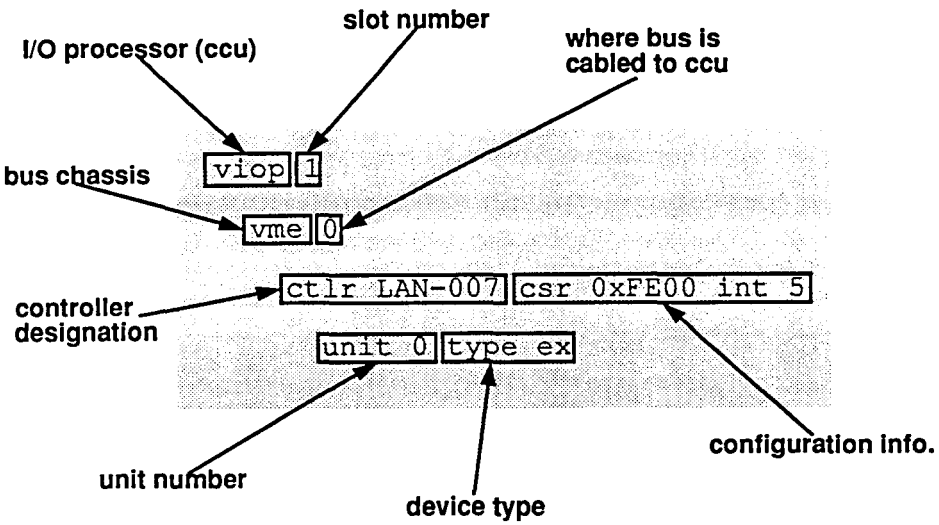
The network hardware configuration is contained in the file */ioconfig* on the SPU. This lists the controllers and devices, where the controllers are configured in linear sequence (ex0, ex1, ...) based on the order in which they appear in the file.

Note that the loopback interface, *lo0*, has no entry, since there is no physical hardware associated with it.

The */ioconfig* file can be read with the command:

```
% spu -r /ioconfig
```

Example */ioconfig* file:



Associating Network Interfaces and IP Addresses [*ifconfig(8c)*]

Network interfaces are associated with IP addresses using *ifconfig.*

ifconfig also:

- configures network interface parameters
  - enable/disable trailers – packet header information follows packet data
  - enable/disable Address Resolution Protocol (ARP) in address mapping
  - set a routing metric for the interface (a "cost" of travelling through this interface)
  - enable/disable debugging information
  - netmask
  - broadcast
- activates/deactivates an interface
- queries the status of an interface
- is used at boot time to define the network address of each interface present on the machine

Associating Network Interfaces and IP Addresses [*ifconfig(8c)*] (cont'd)

Examples:

To query the status of an interface:

```
# ifconfig ex0
```

To activate an interface using the name "myhost"[1]:

```
# ifconfig ex0 myhost up
```

To activate an interface using the IP address 135.44.26.3:

```
# ifconfig ex0 135.44.26.3 up
```

To deactivate an interface:

```
# ifconfig ex0 down
```

To set a netmask for an interface:

```
# ifconfig ex0 netmask 132.68.0.0
```

1. "myhost" would appear in the */etc/hosts* file and provide ifconfig with an IP address to use to configure the interface.

Obtaining network status [*netstat(1c)*]

A variety of network status information is displayed with *netstat*, including:

- information about active sockets
- information about all interfaces (**-i**)
- information about a particular interface (**-I**)
- a one-time snapshot
- repeated information (<*num seconds in interval*>)

Examples:

List of active sockets:

```
% netstat
```

State of interfaces configured:

```
% netstat -i
```

State of a particular interface every 3 seconds:

```
% netstat -I ex0 3
```

Obtaining network status [*netstat(1c)*] (cont'd)

Example output from a host with 2 interfaces:

```
% netstat -i
Name  Mtu   Network     Address      Ipkts   Ierrs Opkts   Oerrs Collis Queue
ex0   1500  trng-net    fetrain      220296  0     48261   0     0      0
lo0   1536  loopback-n  localhost    4483    0     4483    0     0      0
ex1*  1500  trng-net    fetrain2     308     0     4       0     0      0
```

Field Name	Contents	Unit
Name	Network interface (a "*" means the interface is down)	interface (type and unit number)
MTU	Maximum Transmission Unit (largest chunk of data that can be transmitted at one time)	bytes
Network	Network this interface connects to	name * or dot notation
Address	This interface's Internet address	name ** or dot notation
Ipkts	Input packets	# since startup
Ierrs	Input errors	# since startup
Opkts	Output packets	# since startup
Oerrs	Output errors	# since startup
Collis	Collisions	# since startup
Queue	Output queue size	# packets currently in queue

* See */etc/networks* for more information.

** See */etc/hosts* for more information.

Hostnames

A hostname is a text string that represents a particular IP address.

Hostnames and IP addresses are [usually] interchangeable in:

- Application programs (*telnet*, *ftp*, *rlogin*, *rcp*, ...)
- Output from commands (*netstat*, *syspic*, ...)
- Documentation RFC's, administrator manuals, ...)

Hostnames and IP addresses are not interchangeable in:

- Files where the association between the two is being made (*/etc/hosts*, ...)

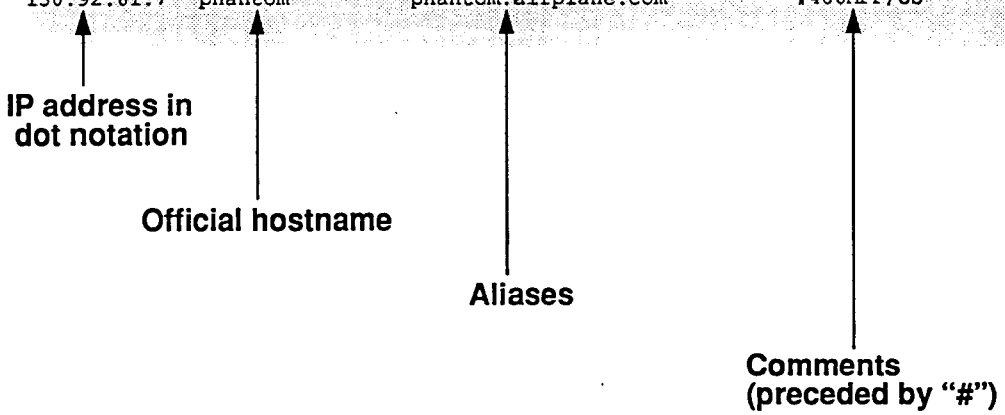
Hosts file [*/etc/hosts*]

The database, */etc/hosts*:

- maps IP addresses to hostnames
- used by utilities such as *telnet, ftp, ...*
- is an ASCII file that can be edited with any editor
- may be placed under NIS control (more later)

Sample portion of an */etc/hosts* file:

```
127.0.0.1     localhost
130.92.61.1   concorde      concorde.airplane.com      #PCMPP/OS
130.92.61.2   blackbird     blackbird.airplane.com     #PCMPP/OS
130.92.61.3   u2            u2.airplane.com            #PCMPP/OS
130.92.61.4   hornet        hornet.airplane.com        #d2w68
130.92.61.5   venison       venison.airplane.com       #MipsMPP/OS
130.92.61.7   phantom       phantom.airplane.com       #486MPP/OS
```



Associating a name with a host [*hostname(1)*]

The host may be assigned a name that can be used to identify the host. Usually, this same name is used as the name of the primary network interface.

To set the host's name to "myhost":

```
# /bin/hostname myhost
```

Then, the network interface ex0 is configured to be named the same as the host[1]:

```
# ifconfig ex0 `/bin/hostname` up
```

These two commands typically appear in the */etc/rc.local* file.

1. This assumes that there is an entry for the result of **/bin/hostname** ("myhost") in the */etc/passwd* file.

Exercises

[1] How many interfaces are available on this machine?

_____

[2] What are their IP addresses?

_____

_____

_____

[3] What is the default name for each interface?

_____

_____

_____

[4] Is there any interface with a name ("Address" in **`netstat`** output) that matches the hosts's official name?

If so, what is the interface? _____

What is it's IP address? _____

** PLEASE DO NOT PROCEED UNTIL INSTRUCTED TO DO SO **

Successful completion of these exercises requires coordination between the instructor and all the members of the class.

Thank you.

[5] Configure an interface on host A as a class C address. Take note of the steps you followed:

_____

_____

_____

[6] Configure an interface on host B as a class C address. Take note of the steps you followed:

_____

_____

_____

[7] Telnet from host A to host B using host B's class C IP address.

_____

[8] Telnet from host B to host A using host A's class C IP address.

_____

[9] What do you need to do so that you can telnet from host B to host A (and vice-versa) using host names instead of IP addresses?

_____

_____

Do it for each of the two hosts. Does it work? Why or why not?_____

_____

[10] If someone is having problems telneting to a host using its hostname, what are some of the possible problems?

_____

_____

_____

_____

_____

_____

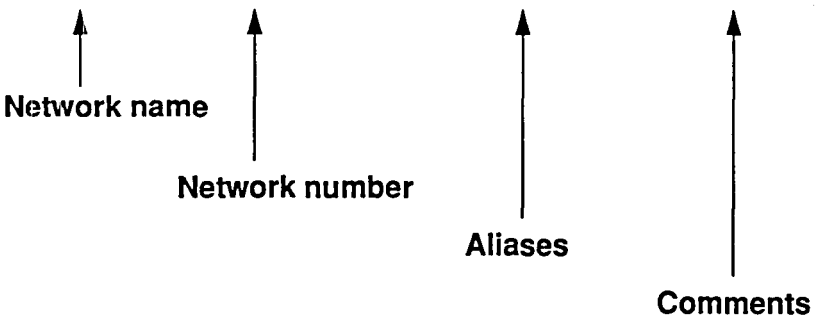
Networks file [*/etc/networks*]

The database, */etc/networks*:

- maps network addresses to names
- used by utilities such as *netstat*, ...
- is an ASCII file that can be edited with any editor
- may be placed under NIS control (more later)

Sample portion of an */etc/networks* file:

```
mktg-net        191.92
fish-net        194.92.49
trng-net        19
kitchen-net     121.2
twinkie-net     192.36.2        cake-net     # high calorie network
# logical networks
loopback-net    127
```



Determining if a network interface is reachable [*ping(8c)*]

The *ping* program:

- determines if another machine's network interface is reachable
- sends an Internet Control Message Protocol (ICMP) message to the designated interface (a reply means the interface was reached)
- has different output for different implementations
- may provide time statistics, but these should not be interpreted as reliable performance measurements

Example (on a Convex system):

```
% ping fetrain
PING fetrain.convex.com [130.168.50.3]: 56 data bytes
64 bytes from 130.168.50.3: icmp_seq=0. time=19. ms
64 bytes from 130.168.50.3: icmp_seq=1. time=15. ms
^C
%
```

Exercises

[1] Your instructor will give you a hostname or an IP address. Ping that host from a Convex machine. What response do you get?

_____

_____

[2] If available, login to a SUN-like workstation. Ping the host from the SUN. How does this response differ from exercise 1?

_____

_____

[3] How would you make the output from the SUN ping look like the output from the Convex ping?

_____

[4] How would you make the output from the Convex look like the output from a SUN ping (this is a trick question)?

_____

Configuring Network Interfaces [*/etc/rc.local*]

The */etc/rc.local* file is where the network interfaces are configured.

Sample section from */etc/rc.local*:

```
if [ -f /etc/ifconfig ]; then
        /etc/ifconfig ex1 `/bin/hostname` netmask 0xffffff00 trailers up arp
        # If UltraNet is installed, start it up
        if [ -f /etc/rc.ultra ]; then
                sh /etc/rc.ultra
        fi
       # All physical interfaces MUST be configured before lo0 is configured
        /etc/ifconfig lo0 localhost up
fi
```

Exercises

[1] What interfaces are configured in the *rc.local* file for your lab machine?

_____

_____

[2] What *ifconfig* command would be used to activate hyperchannel instead of ethernet?

_____

Listing trusted hosts [*/etc/hosts.equiv, .rhosts*]

A "trusted" host is one which does not require authentication to access your host.

Trusted hosts:

- are used by *rlogin, rsh, rcp, ...*
- are listed in */etc/hosts.equiv* and/or *.rhosts files.*
- are listed in */etc/hosts* if the host name is used
- referenced by their IP addresses if their host names are not used
- permit anyone with a valid login to access your machine without authentication, except root

Sample */etc/hosts.equiv* file:

```
concorde
concorde.airplane.com
u2
u2.airplane.com
```

host names are listed one per line

Listing trusted hosts [*/etc/hosts.equiv, .rhosts*] (cont'd)

In addition to */etc/hosts.equiv*, a *.rhosts* file on the destination host is also checked to see if there is an entry for the host originating the access.

The directory examined for the *.rhosts* file is the home directory on the destination host of the user attempting to access that host.

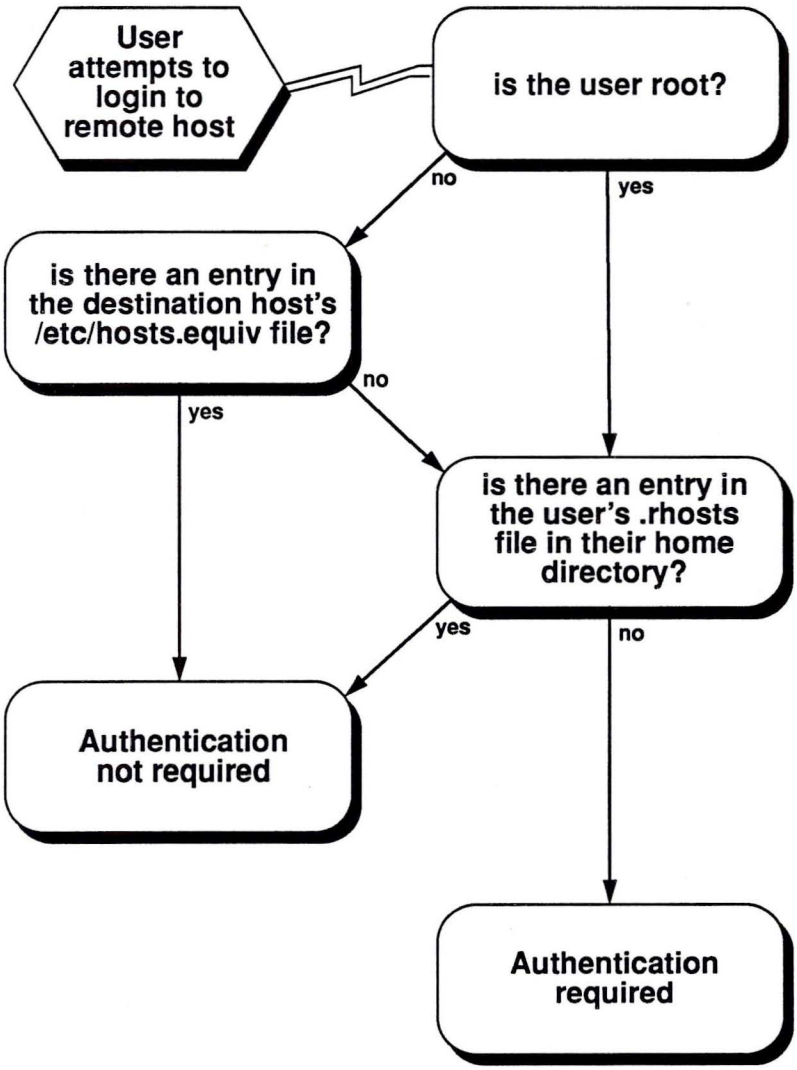
If the machine requesting access does not appear in */etc/hosts.equiv*, then the appropriate *.rhosts* file is checked. If the host requesting access is listed in either file, and the user accessing is not root, authentication is not required.

If the user accessing the destination host is root, then only the file */.rhosts* is checked for the appropriate host.

/etc/hosts.equiv and *.rhosts* files should not allow group or other write access. If they do, they are ignored.

Listing trusted hosts [*/etc/hosts.equiv*, *.rhosts*] (cont'd)

The check-for-authentication process looks like this:



Exercises

[1] What files affect whether or not a user is required to authenticate to a system?

[2] What are the permission restrictions for these files?

[3] What would you do so all users except root can access this host without authenticating when telneting from host TRUSTED?

[4] What would you do so root can also access this host without authenticating?

Network Services

CONVEX COMPUTER CORPORATION

Topics:

- References
- Client-Server Model
- Port numbers
- Internet Services
- Establishing Access to Services
- Transport Protocol Types
- The super-server, inetd

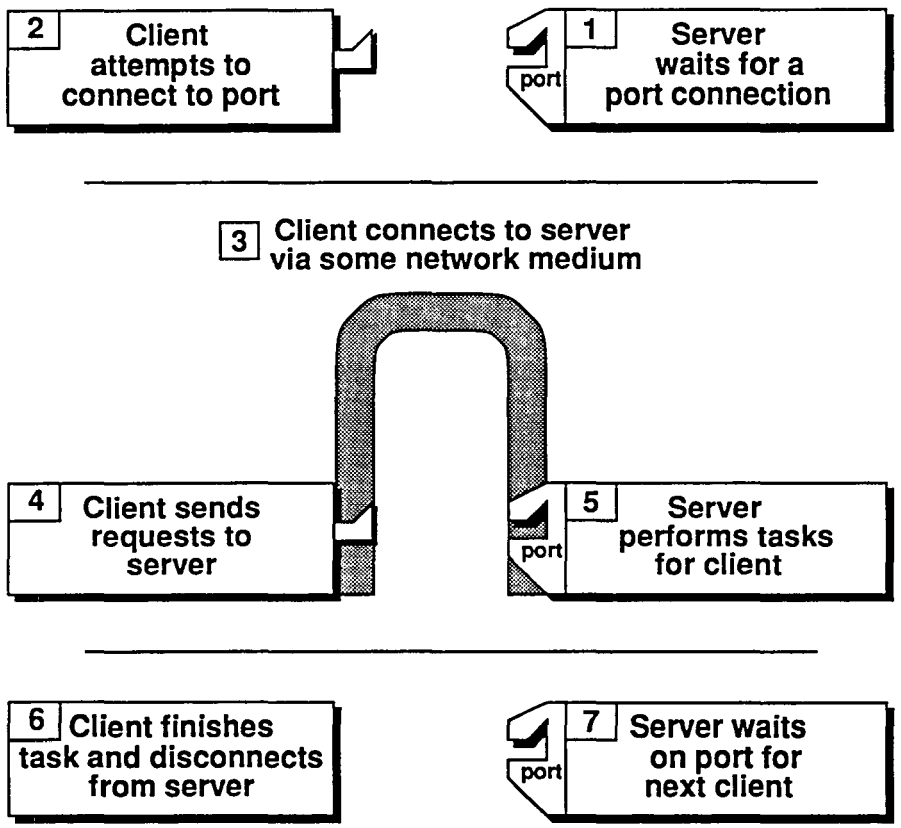
References

- Convex Internet Services System Manager's Guide
- Convex Networking Concepts
- ConvexOS Man Pages:
  - *services(5)*
  - *inted(8c)*
  - *ftpd(8c)*
  - *rlogind(8c)*
  - *rshd(8c)*
  - *telnetd(8c)*

Client-Server Model

Many network applications are based on the client-server model. Non-multiplexing servers perform tasks for one client at a time. Other clients must wait for previous clients to release the server.

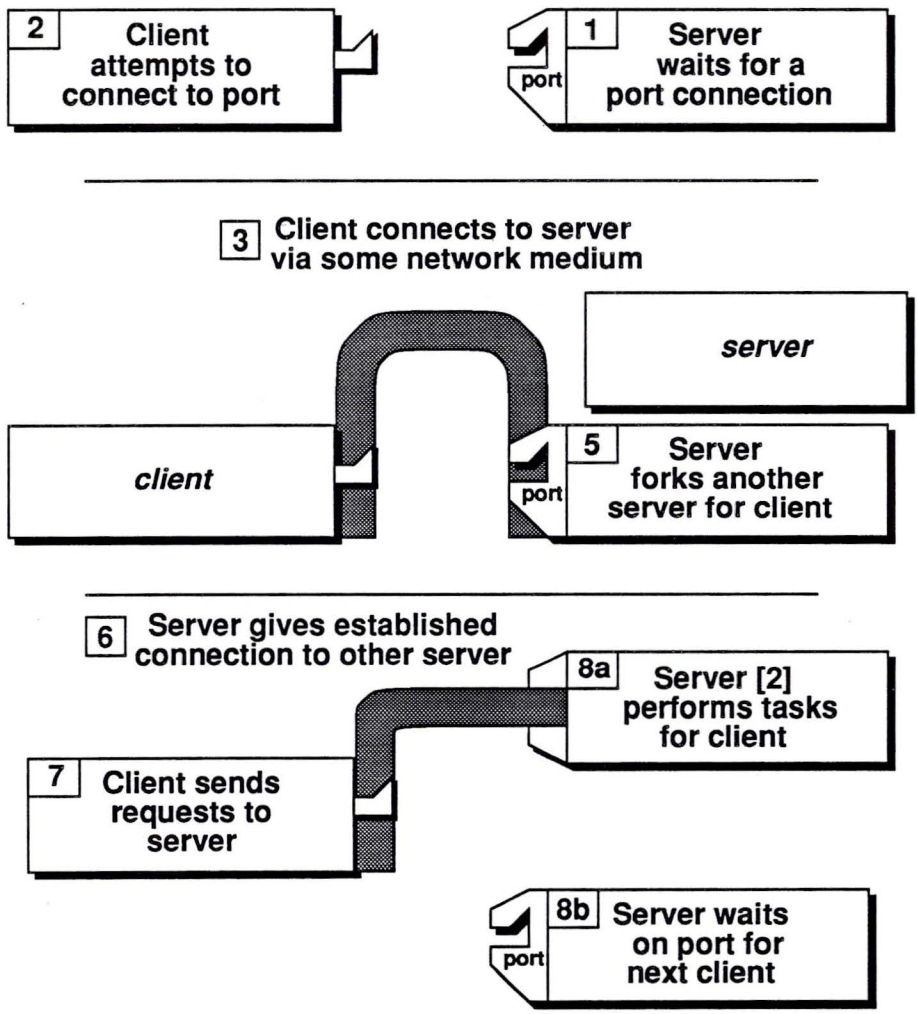
Non-multiplexing client-server paradigm



Client-Server Model (cont'd)

Multiplexing servers can handle requests from several clients simultaneously.

Multiplexing client-server paradigm



Port numbers

In both paradigms, the client must know the port number where the desired server is listening.

A single network interface may be used to communicate with several other network interfaces simultaneously. Port numbers are used by an interface to differentiate these connections.

Port numbers:

- are 16-bit unsigned integers
- have possible values 0 – 65535
- are used to multiplex connections within a protocol

A **socket** is the combination of a port number (which identifies a server) and an IP address (which identifies the host where the server resides).

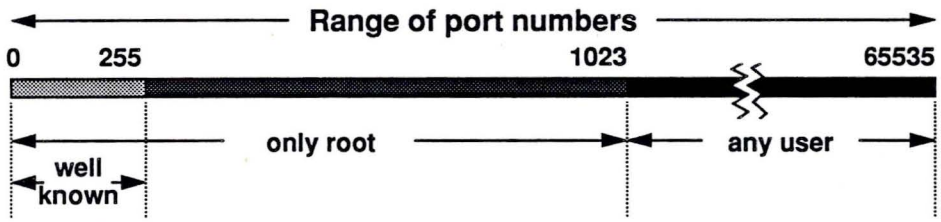
Port numbers (cont'd)

A port number is **reserved** if the application must run as root to listen on the port (server must run as root). Reserved port numbers are less than 1024.

A port number is **well-known** if there is a frequently-used service associated with that port number. Examples:

- ftp
- telnet
- smtp

Well-known port numbers are less than 256.



Internet Services

Internet services provide access and sharing capabilities for a set of hosts on a network.

DARPA Services:

- telnet – remote virtual terminal
- ftp – file transfer protocol
- tftp – trivial file transfer protocol
- SMTP – Simple Mail Transfer Protocol

Berkeley Services:

- rlogin – remote login
- rsh – remote shell
- rcp – remote copy
- ruptime – remote machine status
- rwho – who's logged on, on a remote machine
- rdist – remote file distribution
- finger – remote user information
- rex – remote execution (called "on" on most other systems)
- lpd – remote printing
- talkd – remote conversation
- comsat – mail notification daemon (biff)

Convex Services:

- taped – tape request daemon
- nqsd – Network Queuing System daemon

Establishing Access to Services [*/etc/services*]

Network services are accessed when a client connects to the port where the service daemon is listening. Each of the familiar services is associated with a well-known port.

Some of the more frequently-used services:

Service	Port
ftp	21
telnet	23
smtp	25
tftp	69
sunrpc	111

Establishing Access to Services [*/etc/services*] (cont'd)

The services a host supports are listed in */etc/services*. This file maps the service name to the port number where it is reached, and also the transport layer protocol that is used (TCP or UDP)

A portion of */etc/services*:

```
# The format of the entries are:
#       service-name    port-number/protoname   aliases...
#
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp
```

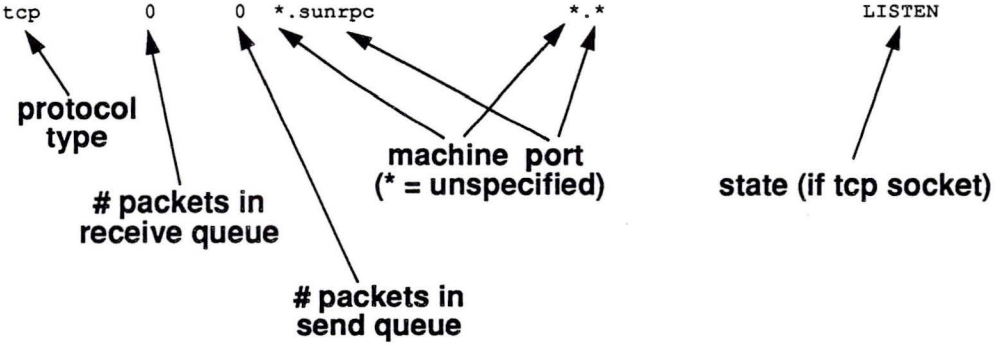
/etc/services is used by:

- inetd – the super server (more later)
- netstat – service names can be displayed rather than their corresponding port numbers

Establishing Access to Services [*/etc/services*] (cont'd)

The services in use can be obtained with the **`netstat`** command[1]:

```
# netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
udp        0      0  *.995                  *.*
tcp        0      0  *.995                  *.*                    LISTEN
udp        0      0  *.biff                 *.*
udp        0      0  *.chargen              *.*
tcp        0      0  *.chargen              *.*                    LISTEN
udp        0      0  *.crashdum             *.*
udp        0      0  *.daytime              *.*
tcp        0      0  *.daytime              *.*                    LISTEN
udp        0      0  *.discard              *.*
tcp        0      0  *.discard              *.*                    LISTEN
udp        0      0  *.echo                 *.*
tcp        0      0  *.echo                 *.*                    LISTEN
tcp        0      0  *.exec                 *.*                    LISTEN
tcp        0      0  *.finger               *.*                    LISTEN
tcp        0      0  *.ftp                  *.*                    LISTEN
tcp        0      0  *.login                *.*                    LISTEN
tcp        0      0  *.nqs                  *.*                    LISTEN
udp        0      0  *.ntalk                *.*
tcp        0      0  *.printer              *.*                    LISTEN
tcp        0      0  *.prtsserv            *.*                    LISTEN
udp        0      0  *.route                *.*
tcp        0      0  *.shell                *.*                    LISTEN
tcp        0      0  *.smtp                 *.*                    LISTEN
udp        0      0  *.sunrpc              *.*
tcp        0      0  *.sunrpc              *.*                    LISTEN
```



1. This output was edited for brevity

Transport Protocol Types

The */etc/services* file specifies a protocol for each service so that some services may communicate over multiple protocols.

Two protocols are frequently seen, TCP and User Datagram Protocol (UDP):

TCP	UDP
connection required	connectionless
reliable delivery	delivery not guaranteed
packet order preserved	packet order not preserved
more overhead	less overhead
error correction	optional checksum (tunable parameter)

Exercises

[1] How many sockets are in a "LISTENING" state? What procedure did you use to determine this?

**** PLEASE DO NOT PROCEED UNTIL INSTRUCTED TO DO SO ****

Successful completion of these exercises requires coordination between the instructor and all the members of the class.

Thank you.

[2] Your instructor will slightly modify the host's configuration. Try telnetting to the host. What happens? Why?

The super-server, inetd

The basic method for accessing a server's capabilities is the same for most servers:

- Identify the service you want by its port number
- Connect to the port
- Begin invoking the server's functions

This procedure becomes repetitive because each server must acquire the port where it listens in the same way. And, if those servers are multiplexing servers, chances are that they give the connection to their children servers the same way.

Multiplexing servers will fork/exec child processes that communicate with clients using standard I/O. The child processes do not have to address network connection issues.

inetd is the super-server, or super-daemon because it listens and establishes connections for multiple servers.

The super-server, inetd (cont'd)

inetd operates this way:

1. *inetd* is started from */etc/rc.std* at boot time.
2. The contents of */etc/inetd.conf* are read to inform inetd:
   - what services are to be assisted
   - what connection type is used
   - what protocol is used
   - if the server multiplexes or not
   - who the server runs as
   - the pathname of the server's executable
   - the command line (argv)
   - if the server is an internet service or an rpc service
3. A correlation is built by inetd matching a port number and a service:
   - For internet services, this depends on the contents of */etc/services* (*inetd* does not look at this file directly; library routines are used).
   - For RPC services, inetd will create a port and register it with *portmap*.[1]

1. This is why the portmapper must be running before *inetd* is started.

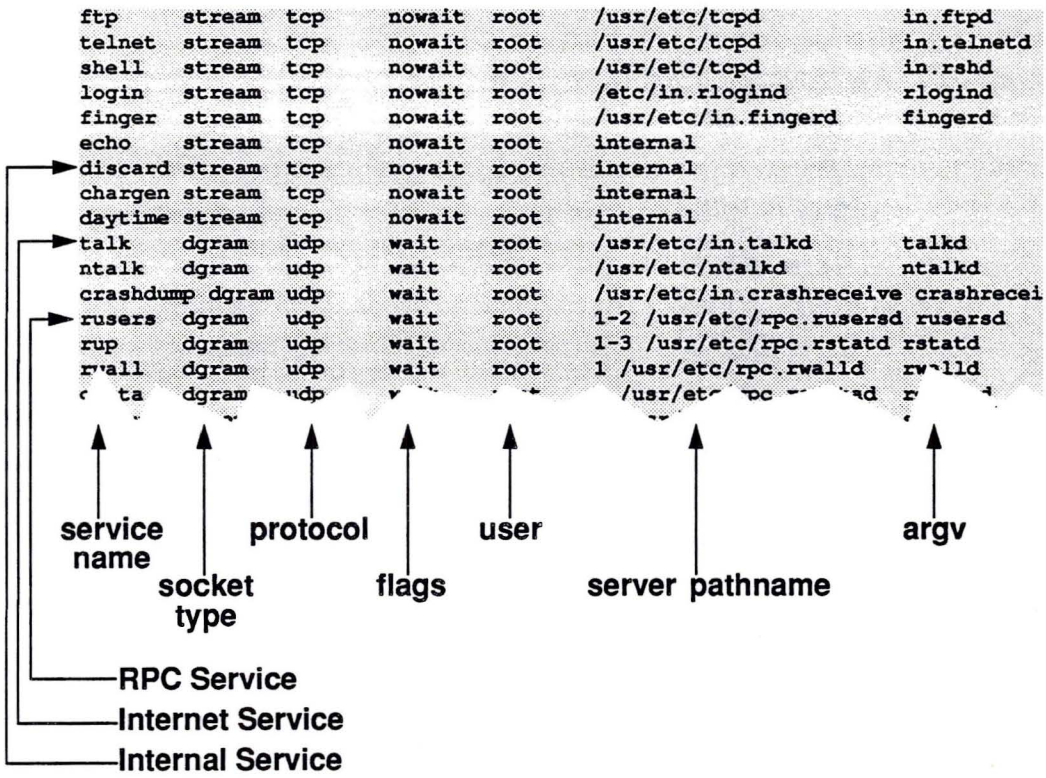
The super-server, inetd (cont'd)

4. Clients connect to ports on the server machine:
   - An internet service client connects to a port monitored by *inetd*
   - An RPC service client connects to port 111, where the portmapper is listening. The portmapper tells the client which port the particular service is using. The client then connects to this port.[1]
5. *inetd* determines what server the client requires based on the port that was connect to, and does one of two things
   - forks/execs that server, giving it the connection to the client as its standard I/O
   - handles the request directly
6. *inetd* returns to listening on the port immediately for multiplexing servers, or after the server finishes executing for non-multiplexing servers

1. For this discussion, we will assume that this is a port which was registered by *inetd*. When the client connects to this port, *inetd* will be listening there.

The super-server, inetd (cont'd)

A portion of an *inetd.conf* file:



The super-server, inetd (cont'd)

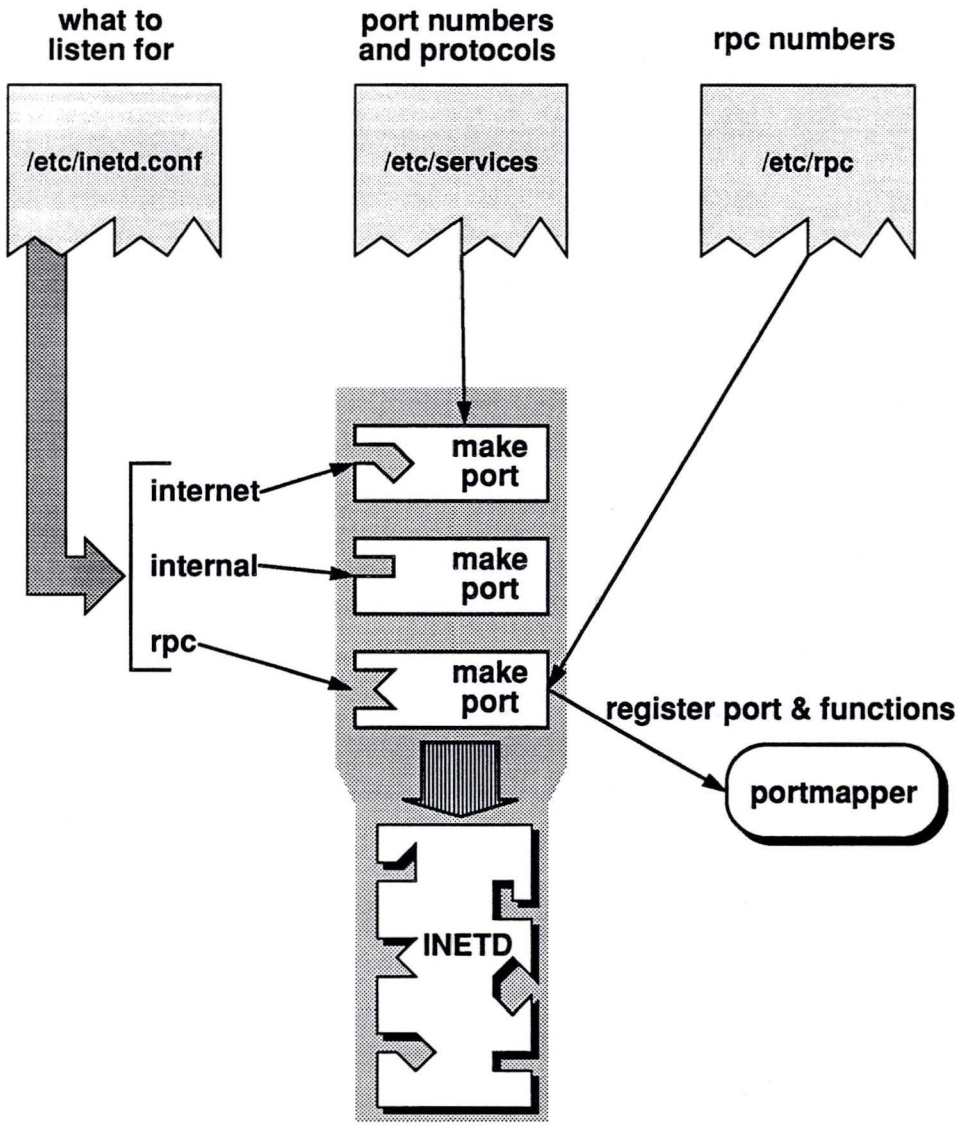
When *inetd* runs, it reads its configuration file, *inetd.conf*, and listens on a set of ports for connections.

If the configuration file changes *inetd's* tables will be out of date. *inetd* will re-read the configuration file, and update its tables, when it is sent a hangup signal (SIGHUP).

This feature lets the system administrator control *inetd*'s behavior using ascii files and scripts.

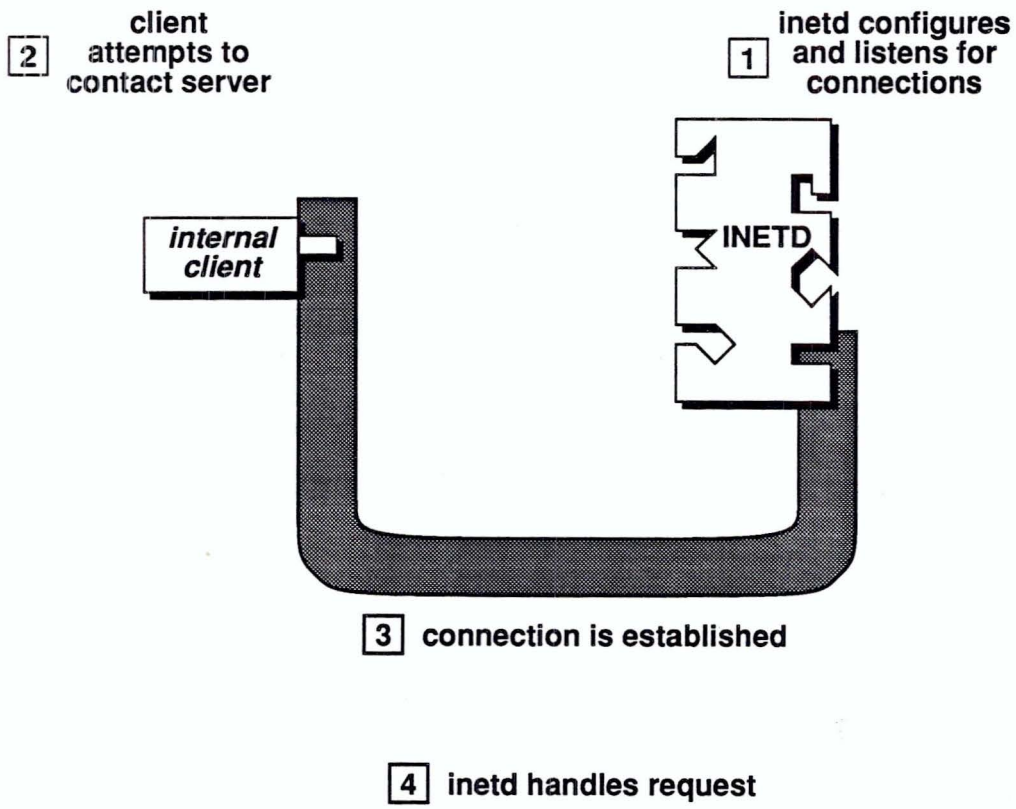
Note that if *inetd* changes its configuration, connections which already exist are not affected.

The super-server, inetd (cont'd)



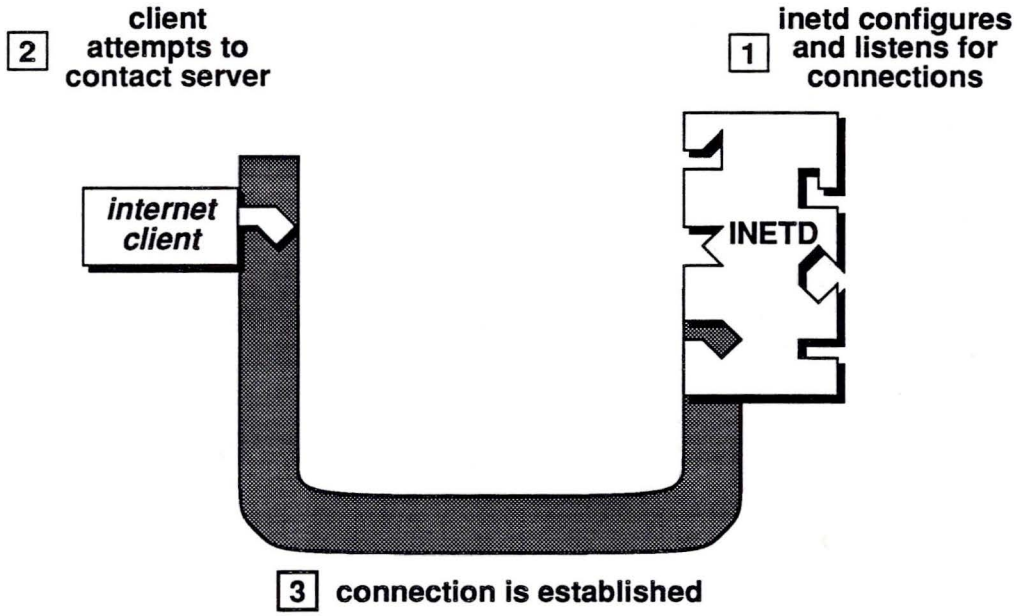
The super-server, inetd (cont'd)

Internal service:



The super-server, inetd (cont'd)

Internet service:

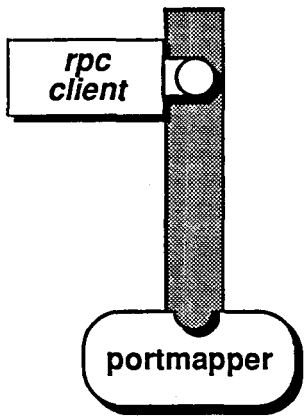


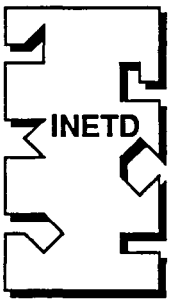
The super-server, inetd (cont'd)

RPC service

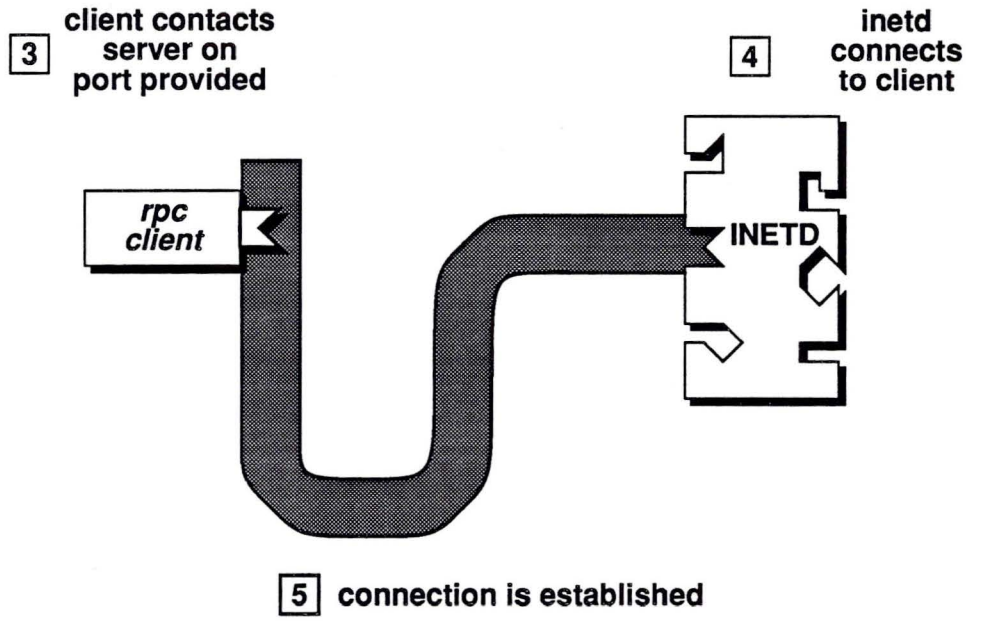
2 client contacts portmapper to get server port

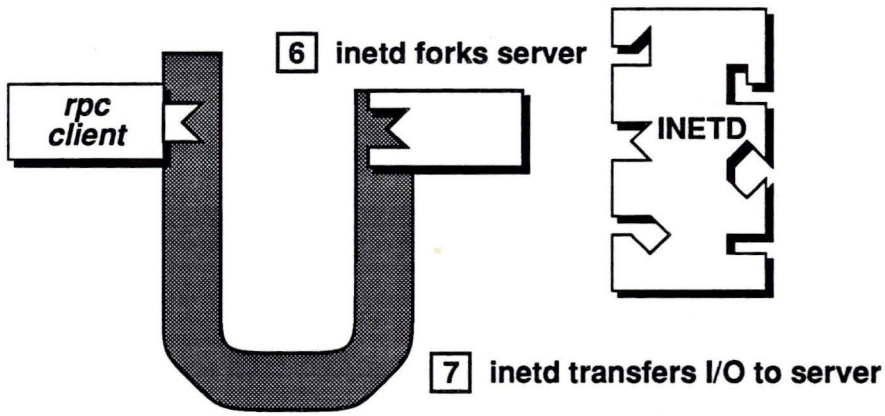
1 inetd configures and listens for connections





The super-server, inetd (cont'd)





Exercises

[1] How many internal *inetd* services are configured for your lab machine?

How many internet services? _____

How many RPC services? _____

**** PLEASE DO NOT PROCEED UNTIL INSTRUCTED TO DO SO ****

Successful completion of these exercises requires coordination between the instructor and all the members of the class.

Thank you.

[2] Create a script in */tmp* called *my_server*. This script should run the date command, and append the result to a file in */tmp* called *date_list*. What does this file look like?

[3] Edit */etc/inetd.conf* so that the ftp service no longer runs the ftp server, but runs your *my_server* script instead. What line(s) did you change?

[4] Run *ftp* to this host. What happens?

[5] Cause *inetd* to update its configuration. What do you need to do to cause this to happen?

_____

_____

[6] Now try *ftp*'ing to this host. Is there any different behavior?

_____

_____

_____

[7] Restore the correct *inetd.conf* file.

Address Resolution Protocol

CONVEX COMPUTER CORPORATION

Topics

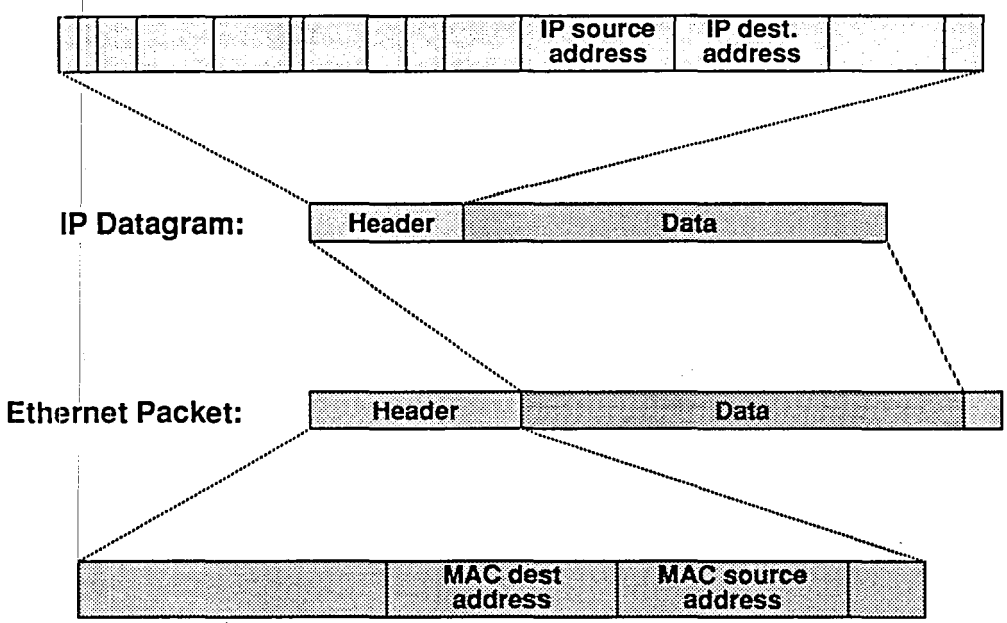
- References
- Address Resolution Protocol
- ARP Processing
- RARP Processing
- Manipulating ARP tables [*arp(8c)*]
- Loading the ARP cache from a file [*/etc/ethers*]
- Trailers

References

- Internet Services System Manager's Guide
- Convex Networking Concepts
- ConvexOS Man Pages:
  - *ioconfig(4)*
  - *arp(8c)*
  - *hosts(5)*
  - *ethers(5)*

Address Resolution Protocol

Address Resolution Protocol (ARP) is a method by which IP addresses are mapped to network hardware addresses (MAC[1] address).



1. Media Access Control

ARP Processing

The ARP protocol does the following:

- When an application requires the MAC address of a host, the ARP table is checked for an entry. If the entry is found, it is returned to the application.
- If the entry is not found, a request is broadcast for the host with the designated IP address to return its MAC address.
- This MAC address/IP address pair is then cached in the ARP table, and the MAC address is returned to the requesting application.

There are other details about what to do when a host is down, how often cache entries are flushed, etc. More information can be found in *Internetworking with TCP/IP, Volume I*, by Douglas E. Comer.

ARP processing is not enabled for an interface by default. To enable ARP processing, use the **`arp`** option of the *ifconfig* command.

When an interface must send a packet for which there is no MAC address available, the ARP protocol is used to get the MAC address.

RARP Processing

The RARP protocol does the following:

- A host which does not know its IP address broadcasts a message containing its MAC address and requests its IP address
- Hosts receiving the request consult their */etc/ethers* files to determine if there is an entry for the requestor's MAC address.
- If an entry is found, a message is sent back to the requestor containing the IP address corresponding to the MAC address.

This facility is used by diskless workstations to determine their IP addresses when they boot.

Manipulating ARP tables [*arp(8c)*]

The ARP tables are manipulated with the *arp* command.

To view the contents of the ARP table:

```
% arp -a
shakespeare.convex.com  (130.168.50.101)  at 0:0:a7:10:1e:3e  permanent
juliet.convex.com       (130.168.50.103)  at 0:0:a7:0:27:e4   permanent
andersen.convex.com     (130.168.50.27)   at 0:80:35:7:0:a0
sushi.convex.com        (130.168.49.1)    at 8:0:14:23:9:11   trailers
hamlet.convex.com       (130.168.50.104)  at 0:0:a7:10:1e:4b  permanent
edu.convex.com          (130.168.50.9)    at 0:0:c:0:b3:9c    permanent
plato.convex.com        (130.168.50.10)   at aa:0:4:0:ea:7
macbeth.convex.com      (130.168.50.105)  at 0:0:a7:10:1e:7f  permanent
o' hello    nvex        (130.  .50.106)   at 0:  a7:10:1e     erma-
                                50.1                  :0:
```

host name — IP address — MAC address — flags

*** A flag of "permanent" indicates the entry was read from a file.**

To delete an entry from the ARP table:

% **arp -d** <*hostname*>

To load the ARP table from a file (different flags indicate different file formats):

% **arp -f** <*file*>

- or -

% **arp -e** <*file*>

Loading the ARP cache from a file [*/etc/ethers*]

The **-e** and **-f** options of the *arp* command load the ARP cache with data contained in a file. This is typically done at boot time from */etc/rc.local* so that diskless nodes can use the RARP protocol to find their IP addresses.

Sample /etc/ethers file:

```
00:00:A7:00:27:E4       juliet
00:00:A7:10:1E:3E       shakespeare
00:00:A7:10:1E:4B       hamlet
00:00:A7:10:1E:76       macbeth
00:00:A7:10:1E:90       othello
```

Trailers

Trailers is an option that restructures packet delivery format on an interface. With this option enabled, packets are transmitted with the header at the end of the packet instead of the beginning. This allows the receiving interface to deposit the data portion of the packet first, aligned on a page boundary, and eliminating a copy. This was a requirement for some early hardware and is not an issue for the majority of hardware today.

This option is rarely used, and Convex recommends disabling it when an interface is configured.

The trailers option is enabled by default in ConvexOS 10.1 and earlier.

To disable trailers, use the **`-trailers`** option of the *ifconfig* command.

No trailers (recommended):

Trailers:

Exercises

[1] How would you make an ARP table entry "permanent" without loading the table entry from a file?

_____

_____

[2] Your instructor will identify a host that appears in the ARP table on your lab machine. This host should be configured "permanent." How would you verify this? Make a note of the table entry.

_____

_____

**** PLEASE DO NOT PROCEED UNTIL INSTRUCTED TO DO SO ****

Successful completion of these exercises requires coordination between the instructor and all the members of the class.

Thank you.

[3] Remove this host's entry from the ARP table and verify it is removed. Now ping the host. What happens to the ARP table?

_____

_____

[4] Is this table entry different now than from exercise 2?

Why or why not?

[5] How would you make this entry "permanent" again?

Routing and Subnets

CONVEX COMPUTER CORPORATION

Topics:

- References
- Topology
- Repeaters
- Bridges
- Routers
- Gateways
- Routing Table
- Routing Algorithm
- Routing Types
- Static Routing [*route(8c)*]
- Dynamic routing [*routed(8c)*]
- Smart Routers
- Subnets
- Routing with subnets

References:

- Internet Services System Manager's Guide
- Convex Networking Concepts
- ConvexOS Man Pages:
  - *route(8c)*
  - *routed(8c)*
  - *ping(8c)*
  - *netstat(1c)*

Topology

A network topology is the physical layout of the network; what hardware components are used, and how they are arranged.

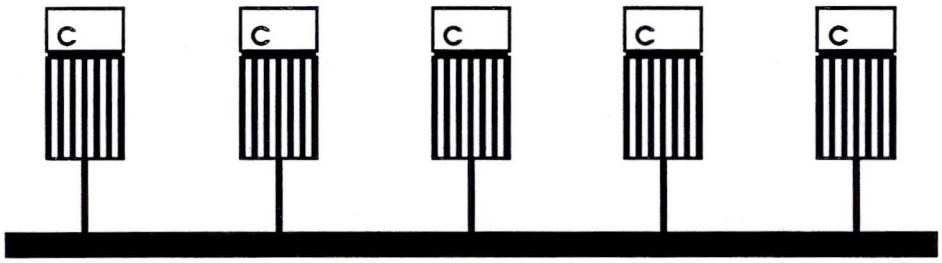
In addition to the transport media (ethernet, FDDI, etc.), a network consists of a number of interconnect devices:

- repeater
- bridge
- gateway
- router

These are used to construct complex topologies using simpler topologies as building blocks.

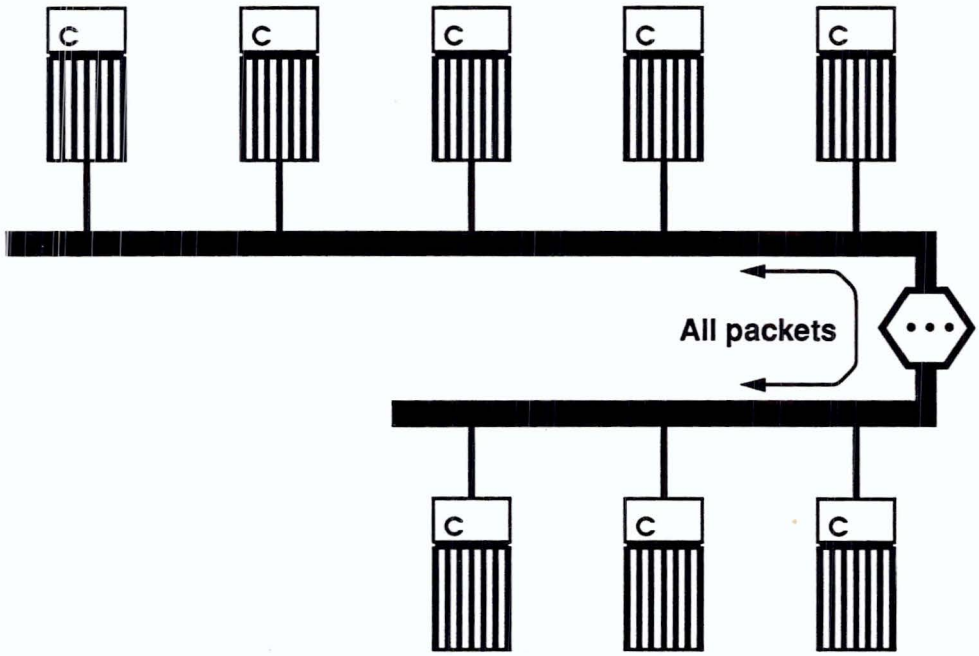
Topology (cont'd)

In the simplest case, all the hosts are on the same network and the topology consists of only the network medium. Any packet put on the hardware media by any host can be received by any other host.



Repeaters

As the distances of the medium increase[1] it becomes necessary to amplify the electronic signals as they traverse the cable. A repeater is a device which simply amplifies the electronic signal to enable extending the length of a communication line.



1. Ethernet, specifically

Bridges

To enable communications between two physically different networks, a bridge can be used. There are two classes of bridges:

1. A simple bridge
2. An adaptive (or learning) bridge

A simple bridge behaves much like a repeater, except it deals with whole packets. This serves to isolate certain types of problems (improperly-formed packets, collisions, etc.) to the physical network where they originated.

Bridges (cont'd)

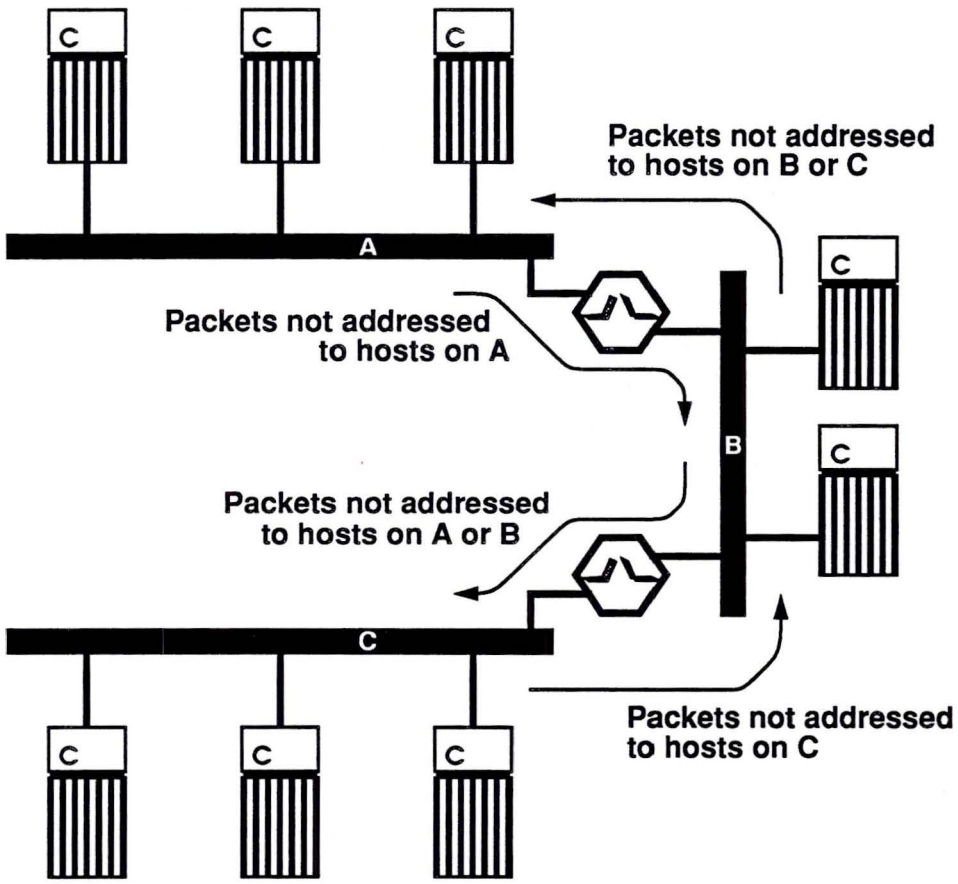
An adaptive bridge provides the advantages of the simple bridge and also serves to reduce traffic on the network:

- A list of reachable interfaces is maintained by the bridge for each of its network connections.
- When a packet enters the bridge, the MAC address of the *source* of the packet is added to the list for that interface.
- When a packet arrives on an interface, the destination address of the packet is compared to the list the bridge maintains for that interface. If the packet's destination appears in the list, the bridge does nothing.
- If the packet's destination address is not in the list, the bridge forwards the packet.

The source lists are built dynamically, based on packet traffic.

Bridges (cont'd)

A possible topology using adaptive bridges:



Routers

There are basically two classes of routers:

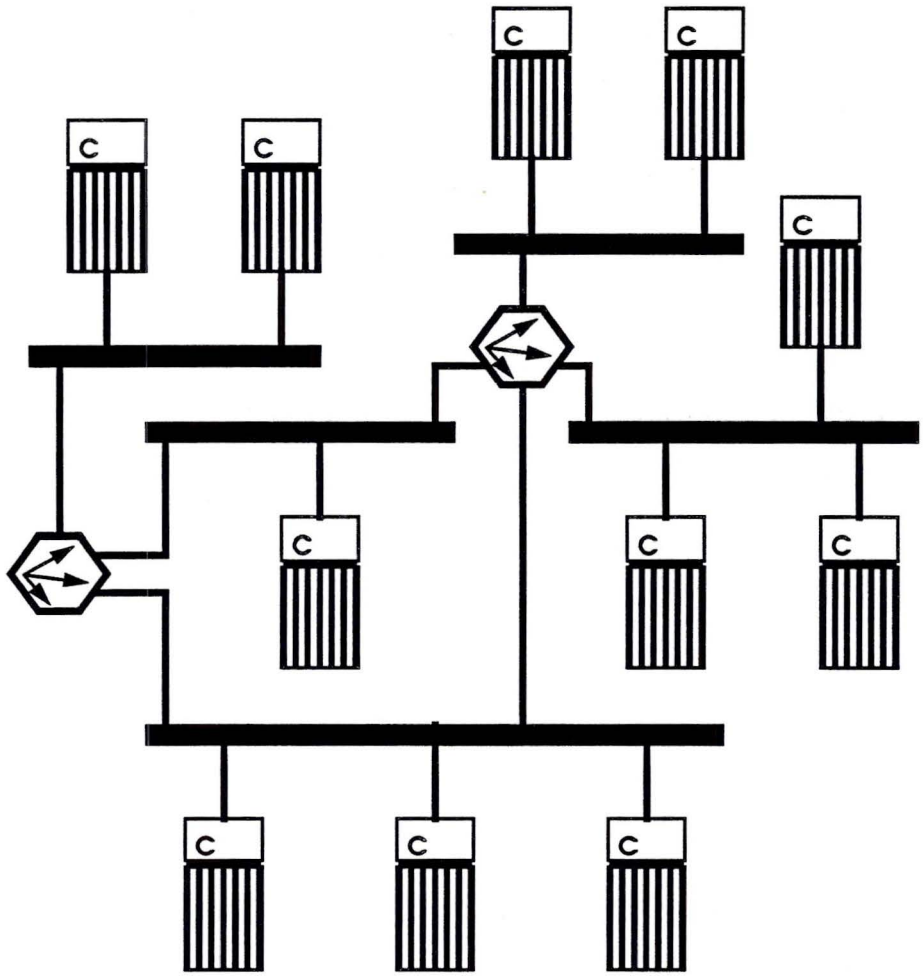
1. Dedicated routing devices (which includes hosts that do nothing but routing)
2. Hosts which perform routing functions in addition to running application programs.

In either case, the routing activity performed is the same. The function of a router is to receive a packet from one interface, and direct it to another interface, based on its destination.

A router uses the network portion of the IP address of a packet to determine the destination interface. This implies that routers connect different [IP level] networks.

Routers (cont'd)

A possible topology using routers:

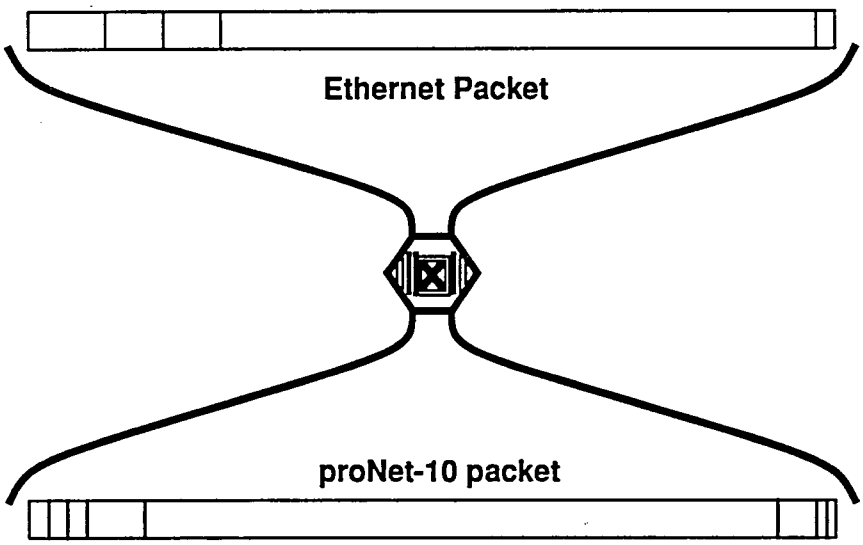


A router requires a network interface for each network it is connected to.

Gateways

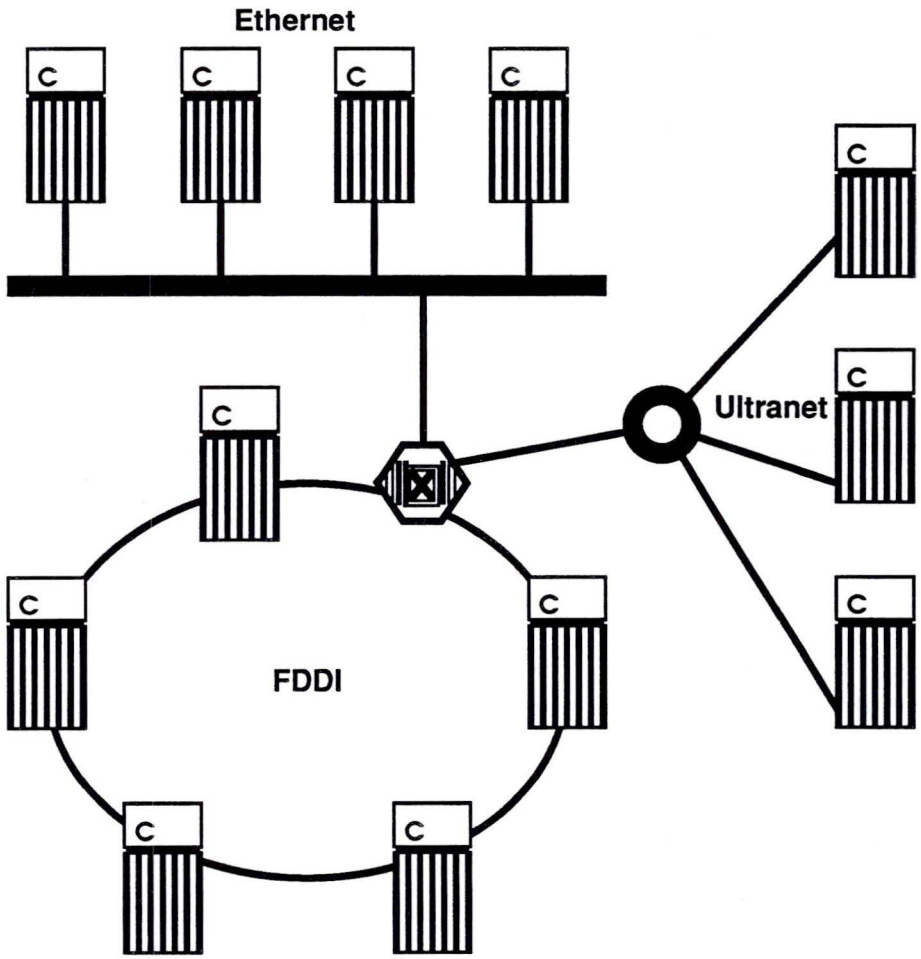
When networks of different physical types must be connected, a gateway is used.

Gateways are able to perform protocol conversions in addition to routing packets between networks.



Gateways (cont'd)

A possible topology using a gateway:



Exercises

[1] You have a situation where three hosts are all located in a machine room, and networked together using ethernet. A printer with an ethernet interface is to be located in a remote area of the building (greater than 500 meters) and needs to be located on this network. How would you go about connecting the printer to the network?

[2] Hosts on an FDDI network need to communicate with hosts on an ethernet network. What is needed to make this happen?

Method 1:

Method 2:

Routing Table

The routing table is a data structure in the kernel which maps destinations to gateways used to get to those destinations.

Destinations can be either host addresses or network addresses.

Gateways are the IP addresses of interfaces connected to the host performing the routing.

The routing table can be viewed with the *netstat* command. Example:

```
% netstat -r -n
Routing tables
Destination       Gateway            Flags     Refs     Use  Interface
127.0.0.1         127.0.0.1          UH           3  916529  lo0
130.168.192       130.168.49.254     UG           0       0  ex1
default           130.168.49.254     UG           0   23797  ex1
130.168.64        130.168.49.254     UG          20  600133  ex1
130.      .93     130.16.  49.254    UG           0       0
```

Destination

Where the next hop is for this route

Flags:
- **U = up**
- **G = gateway**
- **H = host**
- **D = dynamic creation**
- **M = dynamic modification**

Current # of active users

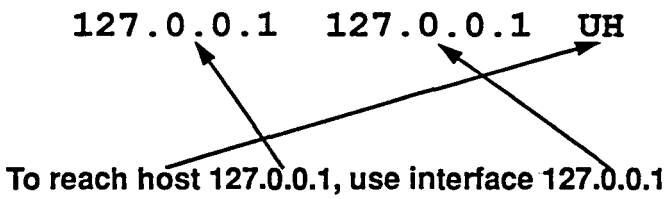
 # packets sent via that route

Network interface

Routing Table (cont'd)

When an interface is configured with *ifconfig*, an entry is made in the routing table for that interface.

The loopback interface, lo0 gets an entry as follows:



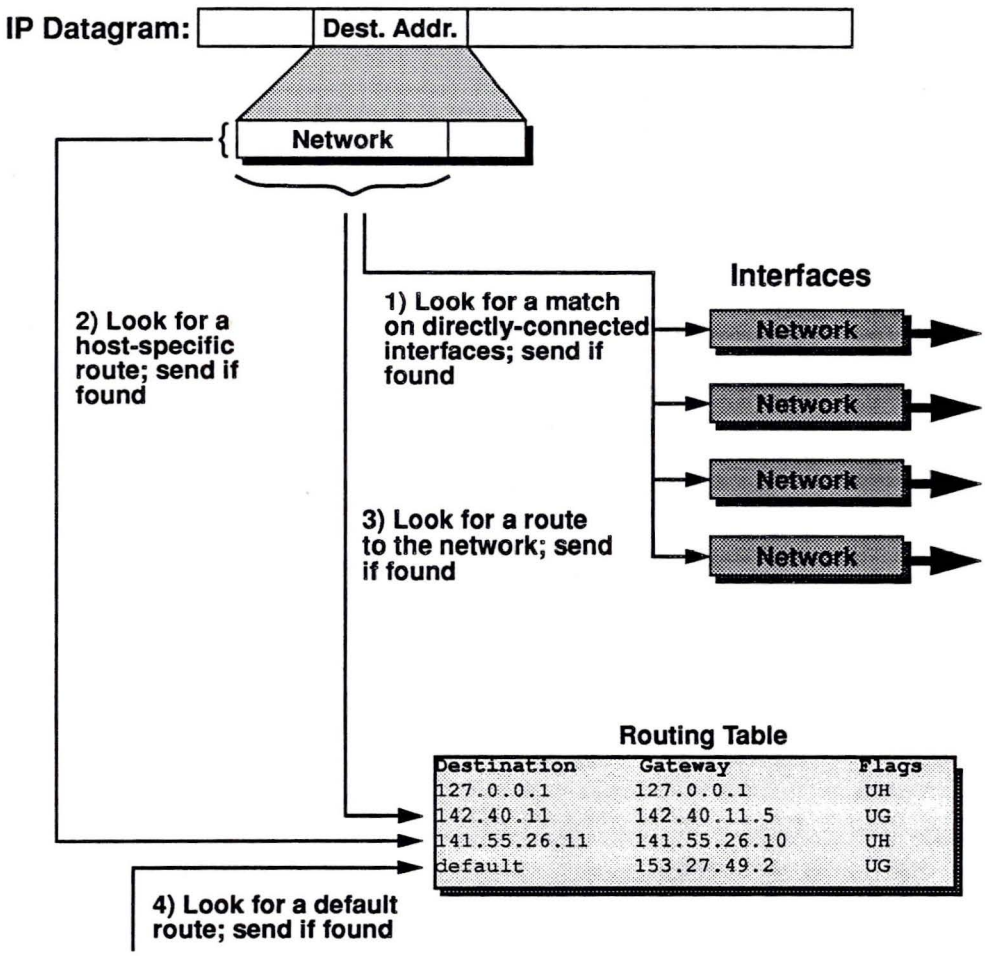
All other interfaces have an entry in the following format:

<network> *<interface>* **U**

Where *<network>* is the network portion of the address of *<interface>*.

Additional routes can be added manually after the interfaces are configured. This can be done in */etc/rc.local* with the *route* command.

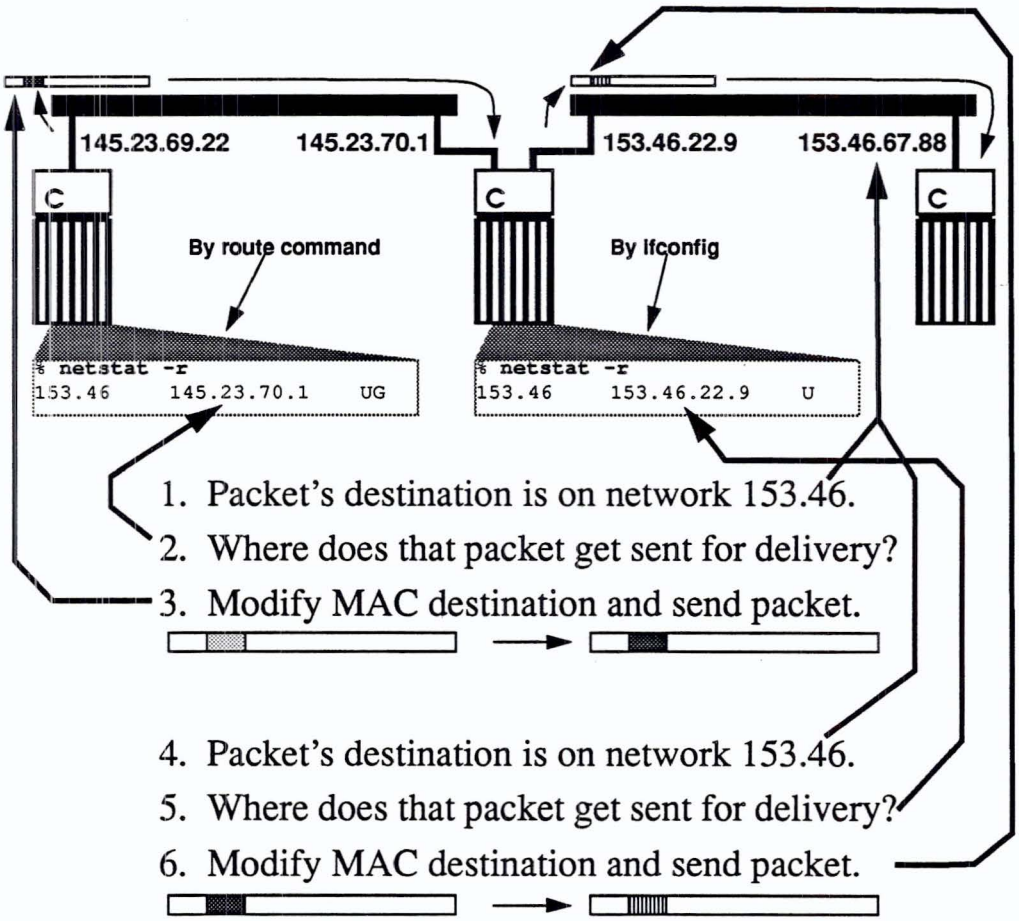
Routing Algorithm



Routing Algorithm (cont'd)

Routing example:

Send a packet from 145.23.69.22 to 153.46.67.88



Routing Types

Convex supports two forms or routing:

1. Static routing:
   - Routing table is derived from the configuration of network interfaces (*ifconfig*) and explicit table manipulation (*route*)
   - Usually used when the number of routes is small
   - Best suited to relatively stable networks
   - Requires each host to have all the routing information it needs
2. Dynamic routing:
   - Routing table is configured and updated by the **routed** daemon
   - Useful when number of routes is large
   - Well-suited for changing network configurations
   - Manual route table manipulation is not required

Static Routing [*route(8c)*]

In static routing, the system administrator maintains the host's routing tables. Changes to the network need to be made manually using the *route* command.

Syntax:

route <*command*> <*keyword*> <*destination*> <*gateway*> <*metric*>

Parameter	Description
<*command*>	**add** – to add a route **delete** – to delete a route
<*keyword*>	**net** – specifies that <*destination*> is to be interpreted as a network **host** – specifies that <*destination*> is to be interpreted as a host * This parameter is optional
<*destination*>	The network or host that will be reached via this route
<*gateway*>	The address of the interface where packets using this route should be sent
<*metric*>	A measure of the "cost" of this route; frequently the number of gateways that must be travelled to reach the destination. This parameter is only required when <*command*> is **add**.

Example:

```
# route add net twohops 130.172.33.1 2
```

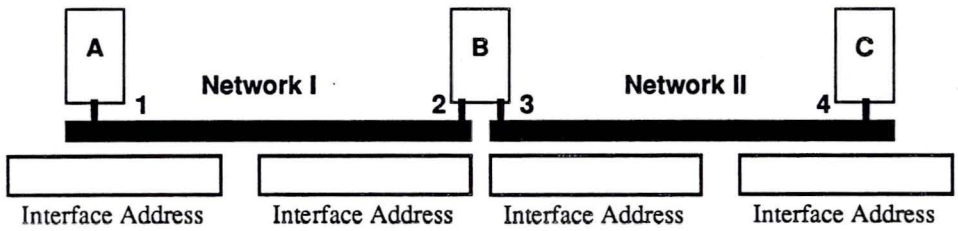
Exercises

**** PLEASE DO NOT BEGIN UNTIL INSTRUCTED TO DO SO ****

Successful completion of these exercises requires coordination between the instructor and all the members of the class.

Thank you.

[1] Configure your network so that a Convex machine is being used as a router between two different networks, with one other host on each network. What does your configuration look like?



[2] What must you do to establish this configuration?

[3] Try pinging interface 4 from host A. What happens? Why?

[4] Add a route from host A network II using dot notation only. Verify your solution. What did you do?

[5] What do you need to do to specify network II symbolically?

[6] Make all necessary entries in */etc/hosts* and */etc/networks* on all three hosts so that interfaces and networks can be specified symbolically instead of by their IP addresses.

/etc/networks changes:

/etc/hosts changes:

[7] What happens if all 4 interfaces are the same class network, and have the same network number?

[8] What happens if hosts A and C have the same IP address?

[9] Design and implement experiments to verify each of the 5 different possibilities the routing algorithm discussed earlier in this module may encounter. Log your configuration and results for each experiment here:

1: _____

2: _____

3: _____

4: _____

5: _____

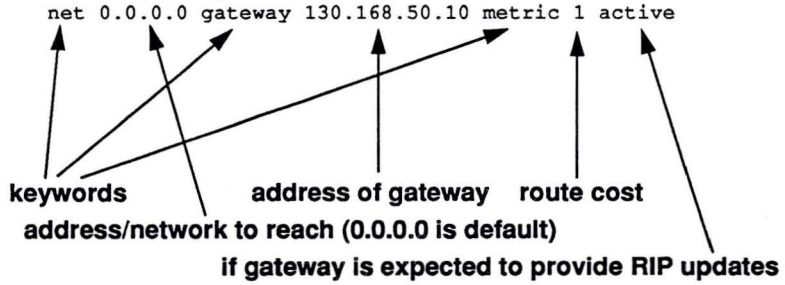
Dynamic Routing [*routed(8c)*]

In dynamic routing, the routing table is maintained by the ***routed*** daemon. Routed uses the Routing Information Protocol (RIP) to update the table:

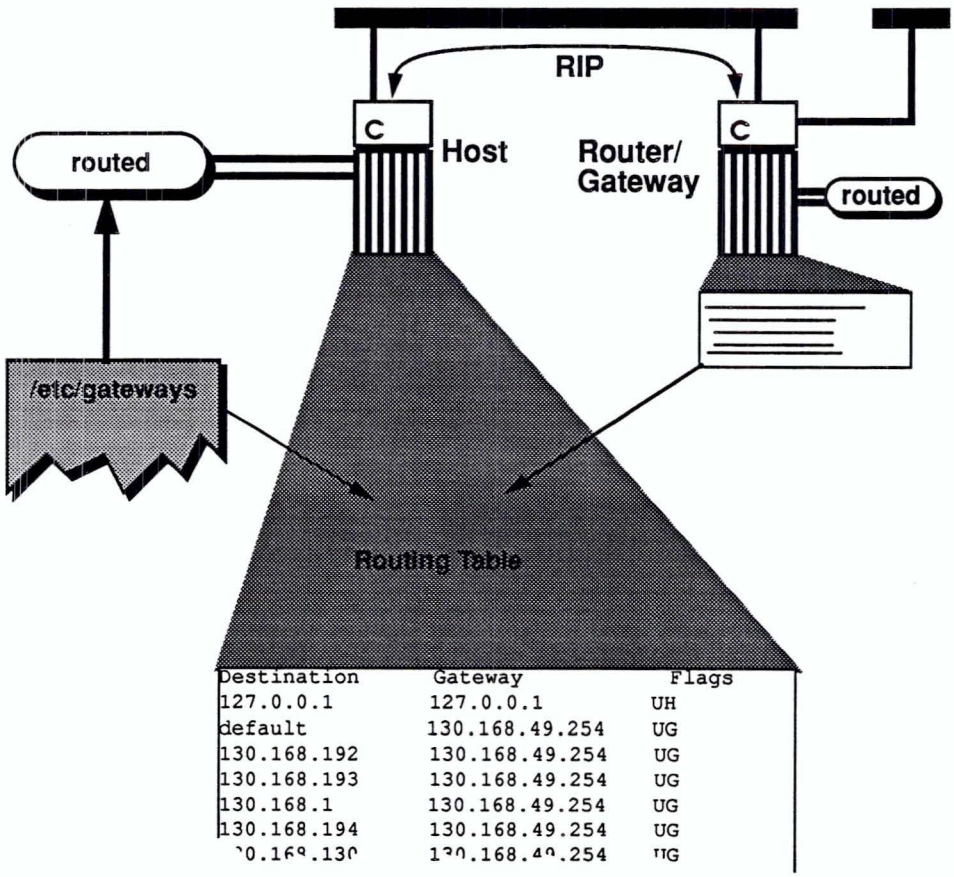
- Gateways broadcast their routing information at regular intervals (typically every 30 seconds). This is an *active* connection because it broadcasts.
- Hosts listen to the gateways and update their routing tables according the information received. This is a *passive* connection because it doesn't broadcast.
- After 180 seconds with no report from a gateway, those routes in the routing table expire and will be deleted (assume the gateway is down).
- The "metric" portion of a route is used in determining the relative cost of the route.
- Routes with a metric greater than 15 are deleted from the table.
- Initial routing table optionally supplied in */etc/gateways*.

Example /etc/gateways setting up a default route to a gateway:

```
# /etc/gateways
net 0.0.0.0 gateway 130.168.50.10 metric 1 active
```



Dynamic Routing [routed(8c)] (cont'd)



Dynamic Routing [routed(8c)] (cont'd)

Dynamic routing can be enabled from */etc/rc.local* by running the *routed* daemon:

```
if [ -f /etc/routed ]; then
  /etc/routed & echo -n ' routed'
```

Effects of dynamic routing:

- The route table is searched linearly and may become quite large causing slowdowns
- *routed* cannot assimilate ICMP redirect messages[1] potentially causing a sub-optimal route to be used
- The table must be searched every 180 seconds to expire routes

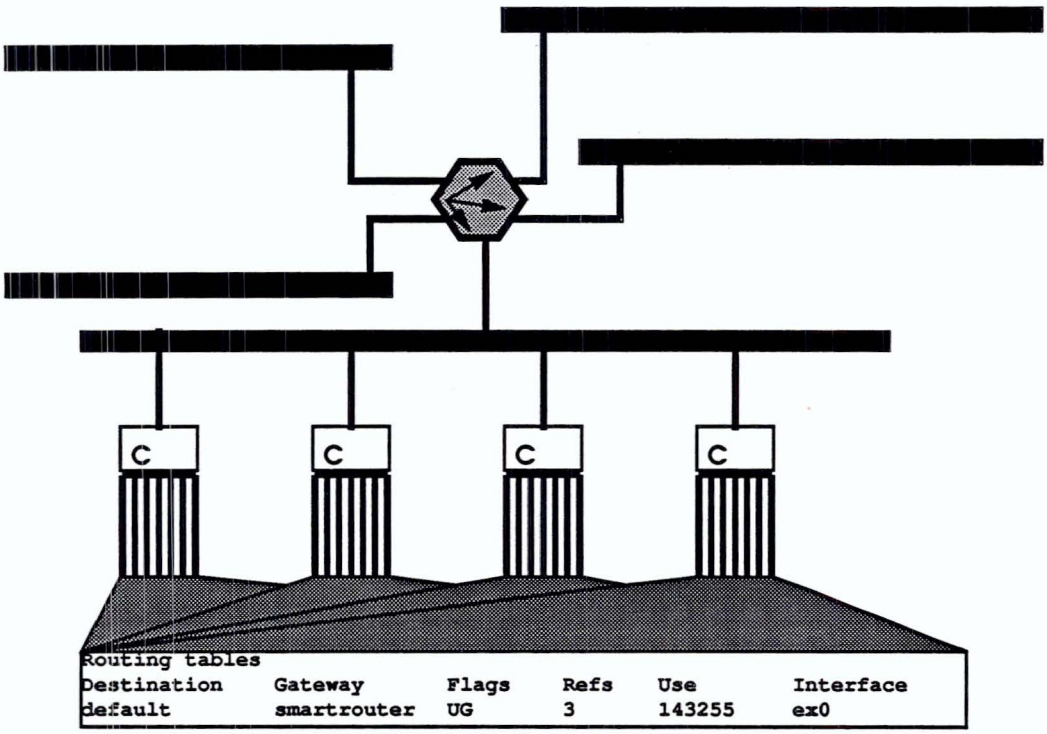
1. Indicated by the "D" or "M" flags in the output from **`netstat -r`**

Smart Routers

As an alternative to a computer providing dynamic routing, a smart router may be used.

Smart routers:

- Are designed specifically to perform dynamic routing
- Support other routing protocols besides RIP
- Require each host using the smart router to have a single static route (the default route) to the smart router



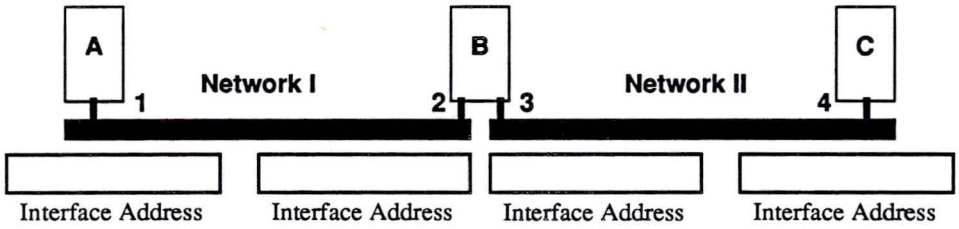
Exercises

**** PLEASE DO NOT BEGIN UNTIL INSTRUCTED TO DO SO ****

Successful completion of these exercises requires coordination between the instructor and all the members of the class.

Thank you.

[1] Configure your network the same way as you did for exercise [1] on page 20.. What does your configuration look like?



[2] Remove all static routing information from all hosts. What did you need to do?

[3] Enable dynamic routing on all hosts. How is this done without editing *rc.local*? Do it.

[4] Try to ping host C from host A. What happens? Why?

[5] Add default routes for hosts A and C. What routes are added?

For host A: _____

For host C: _____

[6] Try pinging host C from host A again. What happens? Why?

[7] For this exercise, simply pinging host C from host A, is it necessary to give host B a default route? Why or why not?

[8] Clear the routing tables from all hosts once again and kill the route daemons. Add */etc/gateways* files to hosts A and C so that the routing table will be configured with a default route when the route daemon is run. What do the */etc/gateways* files look like?

Host A:

Host C:

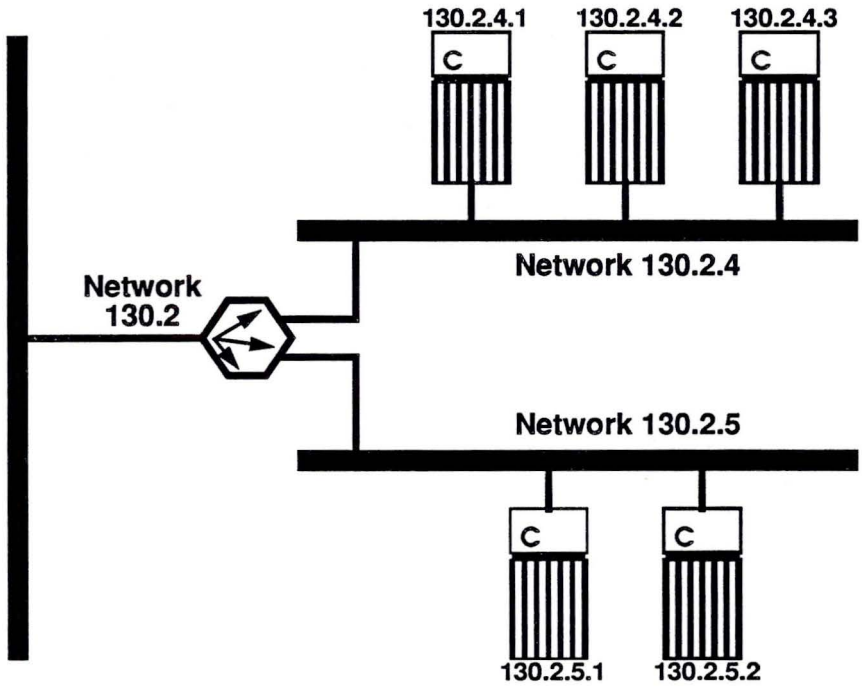
[9] Run the route daemons again and ping host C from host A. What happens?

[10] Compare the route tables on all three hosts. Explain what you see.

Subnets

Subnets:

- Allow multiple physical networks to use the same network number.
- Typically used with Class B addresses (but any address class will work)



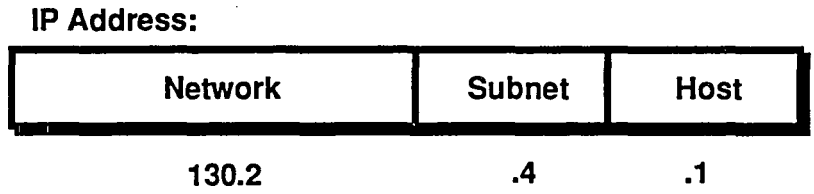
To the outside world (outside of the router), the network hierarchy is a single network.

Inside (subnetted side of the router) the network is a collection of different class networks.

Subnets (cont'd)

Usually, subnetting uses class B & C addresses derived from class A addresses, or class C addresses derived from class B addresses. The divisions don't have to be complete octets, however.

The hosts on the previous page would have their addresses partitioned this way:



Subnets (cont'd)

Subnetting is an attribute of the interface. This means the interface must be *ifconfig*'d with additional information:

- A netmask which specifically identifies what portion of the IP address is used, within the subnetted network, as the network address.
- A broadcast address which specifically identifies how to address broadcast packets.

The netmask and broadcast address[1] will complement one another unless you are trying to do something specific.

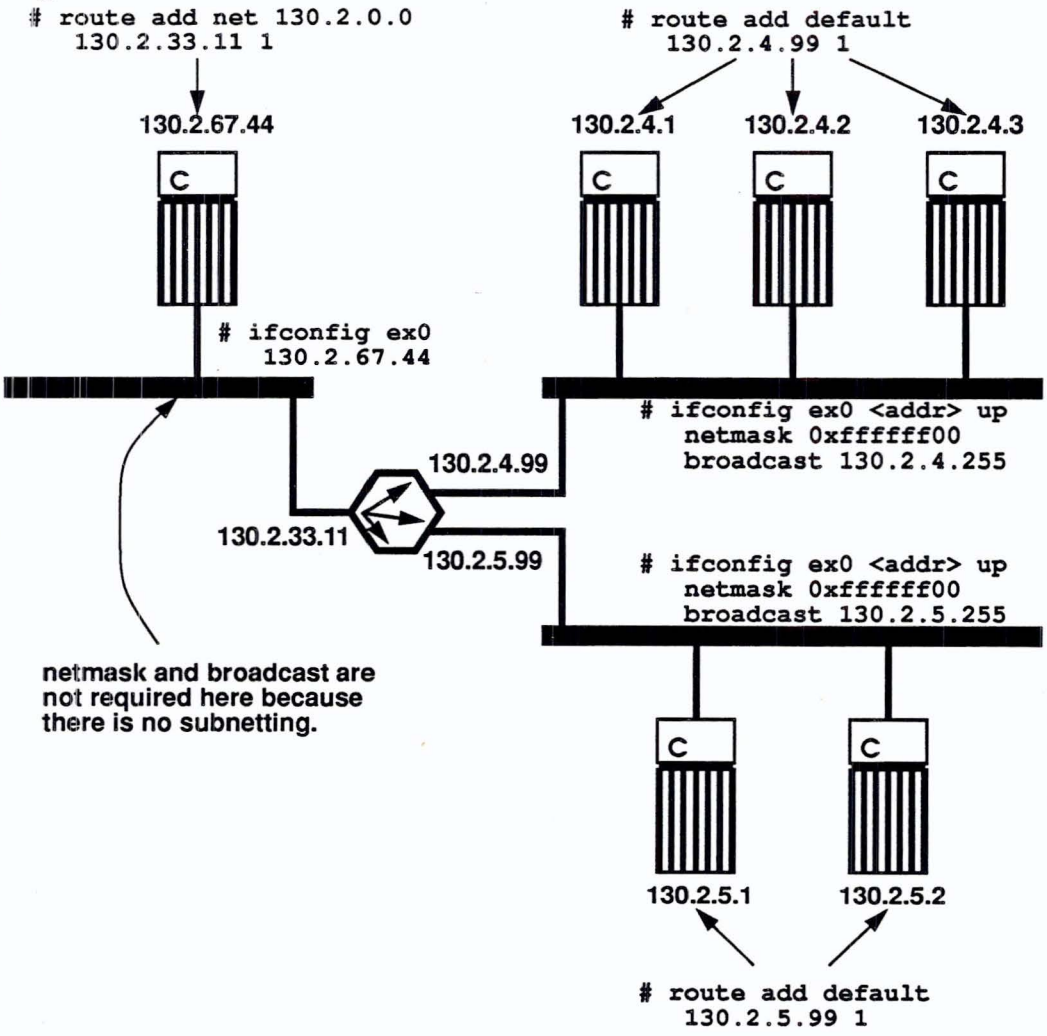
Example:

```
# ifconfig ex0 myhost up
 netmask 0xffffff00 broadcast 130.2.4.255
```

1. Netmasks are usually specified in hexadecimal; broadcast addresses are usually specified in dot notation.

Routing with subnets

Since subnetting will segment a network into different physical networks, routing is necessary for traffic to cross the different segments.



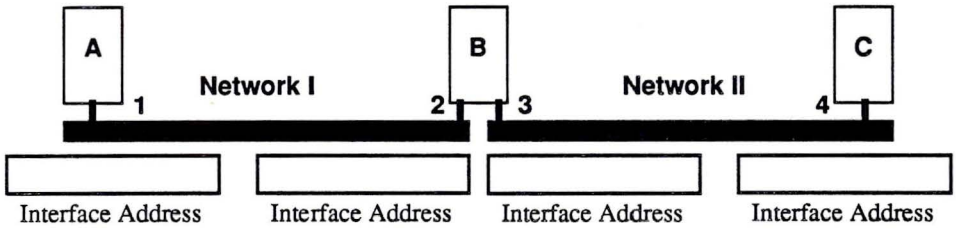
Exercises

**** PLEASE DO NOT BEGIN UNTIL INSTRUCTED TO DO SO ****

Successful completion of these exercises requires coordination between the instructor and all the members of the class.

Thank you.

[1] Configure your network as shown. Network I and Network II should use the same class B network addresses, subnetted with one octet from the host portion of the address. What does your configuration look like?



[2] Modify the necessary files so that the interfaces may be referenced symbolically using the addresses assigned. What files did you modify?

_____

What entries were made?

_____

_____

_____

_____

[3] Based on the information in this module, there are several ways to establish routing from one subnet to the other. What are they?

[4] Select a method of static routing and implement it. What was necessary to accomplish this?

[5] Select a method of dynamic routing and implement it. What was necessary to accomplish this?

[6] What happens if the two [logical] subnets are actually on the same physical network (perhaps somebody goofed in setting up the physical network)? Design and implement an experiment to find out. Does it make a difference if the interfaces are running ARP or not?

Remote Procedure Calls

CONVEX COMPUTER CORPORATION

Topics

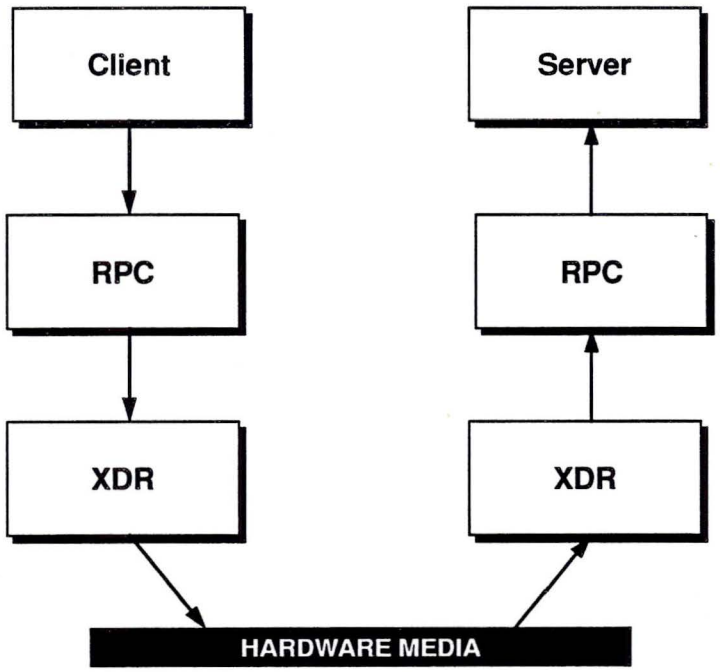
- Remote Procedure Calls (RPC)
- RPC Services
- /etc/portmap
- /usr/etc/rpcinfo
- /usr/etc/spray

Remote Procedure Calls (RPC)

RPC is a programming environment for distributed heterogeneous computing. The major components are:

- A registry service – portmap
- RPC library
- An external data representation (XDR) for architecture independent representation of binary data

RPC Conceptual Flow



Remote Procedure Calls (cont'd)

RPC Services:

1. Operate on top of both TCP and UDP
2. Must register the IP port they use with the portmapper
3. Are not in /etc/services (except *portmap*)
4. Can be started by *inetd*, or independently

Example RPC services started by *inetd* (from */etc/inetd.conf*):

```
rusers  dgram   udp     wait    root    1   /usr/etc/rpc.rusersd rusersd
rup     dgram   udp     wait    root    1-3 /usr/etc/rpc.rstatd rstatd
rwall   dgram   udp     wait    root    1 /usr/etc/rpc.rwalld   rwalld
quota   dgram   udp     wait    root    1 /usr/etc/rpc.rquotad  rquotad
spray   dgram   udp     wait    root    1 /usr/etc/rpc.sprayd   sprayd
rex     stream  tcp     wait    root    1 /usr/etc/rpc.rexd     rexd
mount   dgram   udp     wait    root    1 /usr/etc/rpc.mountd   mountd -n
b  tpa   dgr   udp     wait    root    1 usr/  /rpc.b  tparamd bor  tamd
```

Version Number
(Registered)

RPC Services

Selected RPC services:

Service	Function
rusers	information about users on remote machines
rup	more efficient version of ruptime
quota	required to support quotas on nfs mounted partitions
spray	like ping for rpc
rex	remote execution facility (called "on" by most UNIX systems)
mount	required to mount nfs partitions
bootparam	supports remote booting of workstations

RPC Services (cont'd)

Each RPC service has:

1. A program number
2. A version range
3. A Program name
4. Optional nicknames

This information is maintained in */etc/rpc*.

Example */etc/rpc*:

```
# The format of the entries are:
#       rpc-server      rpc-number      aliases...
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog nfsd
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown rwalld
yppasswdd       100009  yppasswd
etherstatd      100010  etherstat
rquotad         100011  rquotaprog quota rquota
sprayd          100012  spray
3270_mapper     100013
rje_mapper      100014
selection_svc   100015  selnsvc
database_svc    100016
rexd            10[illegible]7  r -
```

/etc/portmap

- TCP and UDP both require a port number to map internet packets to services
- */etc/services* normally contains this information, applications use a *getservbyname* library call
- *portmap* provides this function for rpc services
- *portmap* maps RPC program numbers to internet port numbers
- applications make an RPC call to *portmap* to obtain the port number for the desired service

Client	Server
• Allocate socket data structure	• Allocate socket data structure • Bind socket to localaddress.port • Send RCP registration to portmapper
• Connect to portmap on remote machine to obtain port # for RPC service	• Listen on address.port for connections
• Connect to remotehost.port	

/etc/portmap (cont'd)

- is started from */etc/rc.local*
- must be started before *inetd*

Example entry in */etc/rc.local*:

```
if [ -f /etc/portmap ]; then
        /etc/portmap & echo -n ' portmap'
fi
```

/usr/etc/rpcinfo

A utility to test RPC program registration and readiness

Example:

```
/usr/etc/rpcinfo -p myhost
```

Shows which programs are registered on **myhost**.

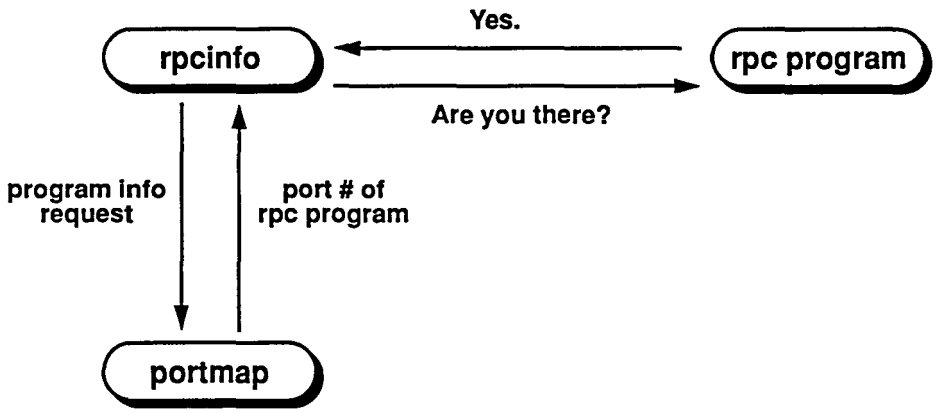
Sample output:

```
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100007    2   tcp   1028  ypbind
    100007    2   udp   1025  ypbind
    100007    1   tcp   1028  ypbind
    150001    1   udp   1023  pcnfsd
    100007    1   udp   1025  ypbind
    100003    2   udp   2049  nfs
    100028    1   tcp    608  ypupdated
    100004    2   udp    995  ypserv
 536924300    1   udp    612  nfsd_fork
    100028    1   udp    610  ypupdated
    100029    1   udp   1002  keyserv
    100004    2   tcp    996  ypserv
    100004    1   udp    995  ypserv
    100004    1   tcp    996  ypserv
    100009    1   udp   1021  yppasswdd
 591751049    1   tcp    712
 591751049    2   tcp    712
 591751049    3   tcp    712
    100002    1   udp   4951  rusersd
    100002    2   udp   4951  rusersd
    100001    1   udp   4952  rstatd
    100001    2   udp   4952  rstatd
```

/usr/etc/rpcinfo (cont'd)

rpcinfo can also test whether or not a registered program is responding.

rpcinfo flow:



You need to know:

1. the program number
2. the protocol (TCP or UDP)
3. the version (optional)

Usage:

rpcinfo *–[u|p]* <*prognum*> <*version*> <*host*>

/usr/etc/rpcinfo (cont'd)

rpcinfo can also be used to get the IP address of all servers (for a particular service) that are registered on your network. Use the -b option to retrieve this information.

Example:

`rpcinfo -b` <*service*>

/usr/etc/spray

- used to test throughput of RPC
- sends a 100k byte stream of RPC requests to a specified host

Sample output:

```
% spray -c 5 mikey
sending 5 packets of lnth 86 to mikey ...
        no packets dropped by mikey
        195 packets/sec, 16796 bytes/sec
```

Exercises

[1] Configure mountd to disallow pre-3.0 mount requests

z inetd.conf wymień parametr
-n z mountd oraz then
kill -HUP `inetd.ps`

[2] You stop getting rup information from host A. What is wrong?

rstatd does not work
inetd does not start rstatd
(rpcinfo -p - my zarejestrowany)

man rpcgen ← pozwala
pisać programy
rpc
i przejmować się
problemem XDR

Network File System

CONVEX COMPUTER CORPORATION

Topics

- Purpose of NFS
- NFS Protocol Flow
- NFS Architecture
- NFS Mount
- NFS Access
- NFS Read – Client
- NFS Access Mount Options
- Soft Mounts
- Hard Mounts
- Interruptible Mounts
- NFS Components
- NFS Setup
- Designing a File System
- NFS and UNIX Semantics
- NFS Server Setup
- NFS Client Setup
- NFS Security
- NFS Performance

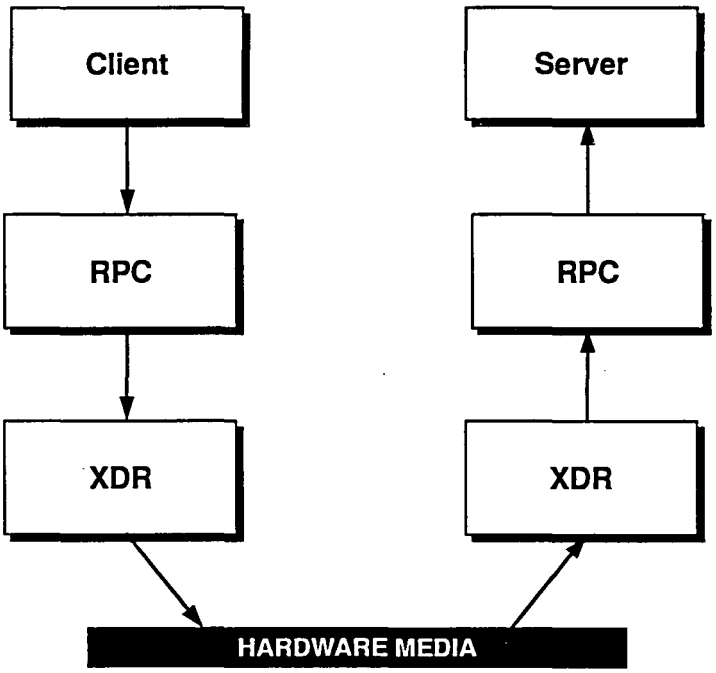
Purpose of NFS

The Network File System was designed to allow remote disk partitions and directories to appear like local disks and directories.

NFS is:

- a protocol
- and RPC application
- a collection of kernel modules and user space daemons
- heterogeneous
- stateless

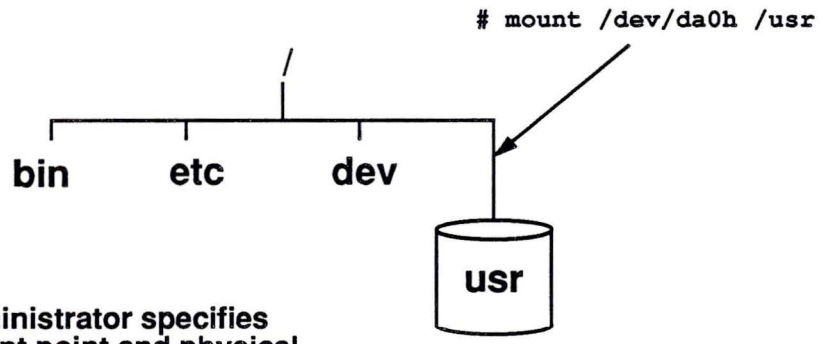
NFS protocol flow



NFS Architecture

NFS emulates local mounts but accesses data across the network. Compare NFS mounts to traditional mounts.

Traditional mount:

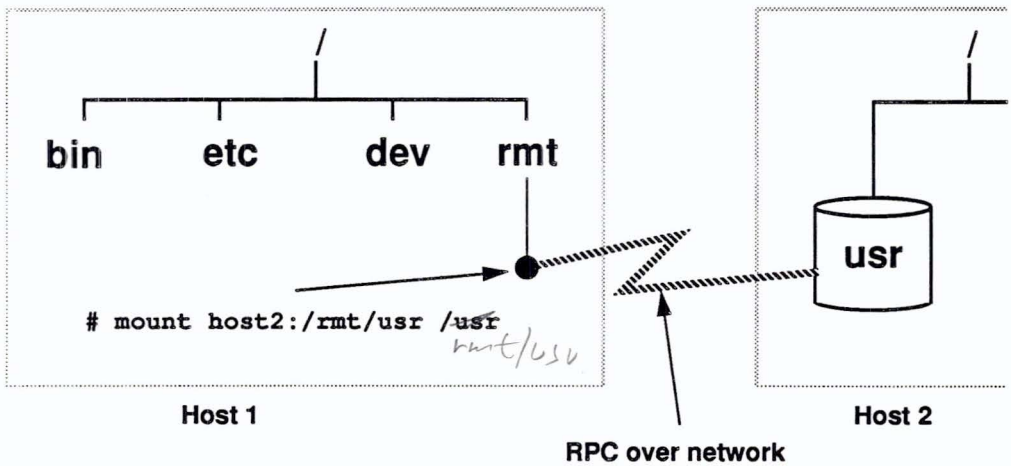


Administrator specifies mount point and physical disk partition

NFS Architecture (cont'd)

In NFS mounts the access to a physical drive is replaced with an RPC call across the network.

NFS Mount:



Administrator specifies mount point and host where files will be accessed from

***** Do NOT place NFS mount points directly under the root directory***

(do a symbolic link for root directory)

NFS Architecture (cont'd)

NFS uses several daemons to mount and transfer data across NFS mounts.

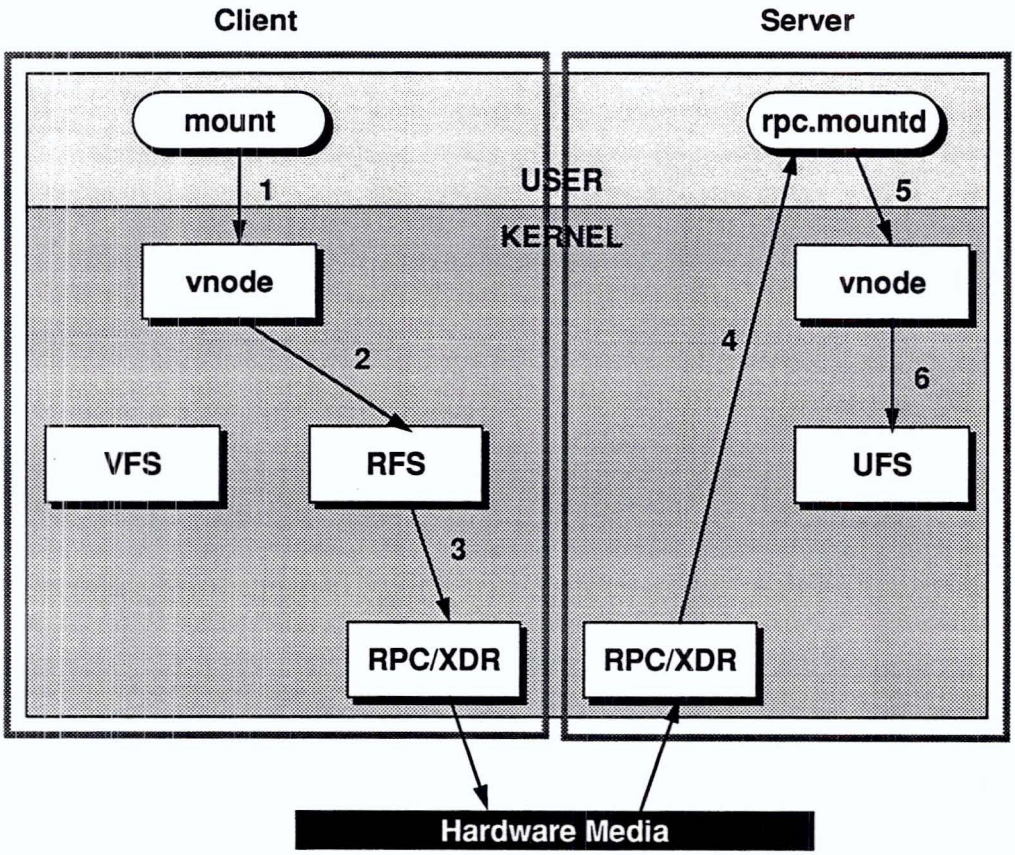
They are:

Daemon	Purpose	Runs On
rpc.mountd	Receives incoming mount requests and passes back mount handle to diskless system	Server
nfsd	Receives read and write requests from RPC fro local disk, returns read requests to diskless system via RPC	Client
biod	Buffers read and write requests into more efficient large blocks	Client

NFS Mount

There are several "file system" components in an NFS mount:

- VFS – Virtual File System
- UFS – UNIX File System
- RFS – Remote File System



NFS Mount (cont'd)

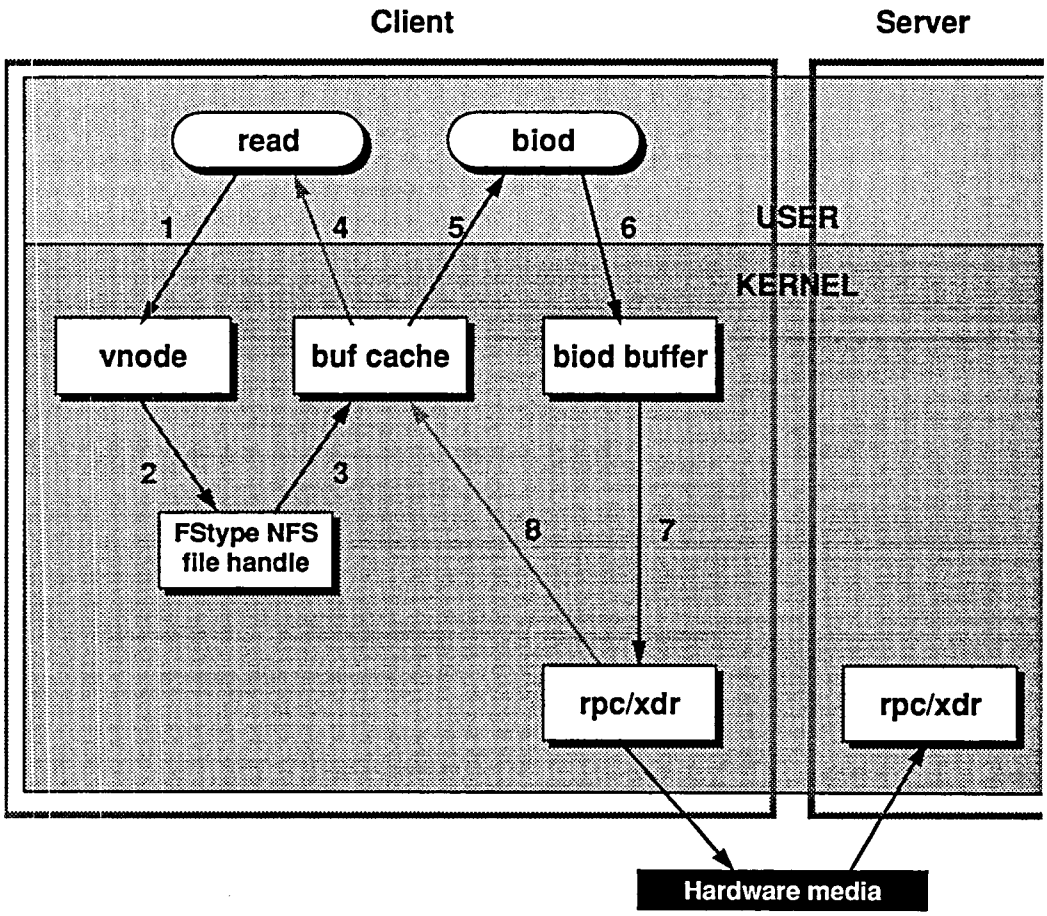
1. Client initiates request with the *mount* command
2. Client vnode layer calls remote RFS
3. Client remote RFS packages mount RPC call and sends to server
4. Server receives RPC request and sends to **`rpc.mountd`** daemon

   Authentication is performed if necessary
5. Server *rpc.mountd* accesses local file system for mount data
6. Server packages mount information into file handle and returns via RPC

NFS Access

Once a mount file handle is returned to the client, read and write requests can begin.

Example NFS read:



NFS Read – Client

1. Application generates read request and passes to kernel vnode layer
2. Kernel determines access is for NFS file and locates file handle
3. Kernel checks buffer cache for data
4. If data found, returns data to application
5. If data not found, and RPC request is passed to the block I/O daemon, *biod*
6. *Biod* buffers the request in 8k buffers
7. Then *biod* sends the RPC request to the remote system
8. If no *biod*s are available, the kernel generates and RPC request directly

NFS Access Mount Options

Notice that once the read request is handed off to the RPC layer and passed over the hardware media, there is very little the application can do until a response is returned.

How and when a response is returned depends on how the file system was mounted.

Mount options:

- Soft – an ETIMEOUT error is returned after some number of tries
- Hard – the application waits until a response is received
- Intr – accept **^C** to kill an I/O operation from the keyboard

Soft Mounts

- Used for:
  - Read-only data
  - Global data repositories
  - Infrequently accessed data
- Not all applications handle the ETIMEOUT error correctly
- Not recommended for:
  - executables
  - temporary files or directories
  - application staging areas

Hard Mounts

- Ensure update occurs even if the server crashes
- Used for;
  - executables
  - home directories
  - application data spooling
- Application pend until an operation is completed
- This includes *pwd*, *ls*, and any other utilities that **`stat`** a hard mounted directory entry.
- Applications in this state cannot be interrupted

Interruptible Mounts

- Used in conjunction with hard mounts
- Convex recommends this as the default
- Allow **^C** (INTR character) from the keyboard to interrupt an outstanding NFS I/O request
- Some applications do not behave will when I/O is interrupted

Exercises

[1] Execute the mount command and identify which types of mounts this system has. Note them below:

[2] Start an interactive editor session on a file in each of the NFS mounted partitions. Observe the behavior of the application as the sever does not respond. Note the response below:

NFS Components

In addition to the daemons we discussed earlier, additional daemons provide added functionality.

Service Name	Function	Initiated From	Runs on
nfsd	processes server requests	*rc.local*	server
biod	batches up client I/O requests	*rc.local*	client
quota	supports quotas	*inetd*	both
rpc.statd	status monitor	*rc.local*	both
rpc.lockd	file locking *	*rc.local*	both
mount	process file system mount requests	*inetd*	both

* Requires **rpc.statd**

NFS Setup

The following steps are required:

Server:

1. Install software
2. verify */etc/passwd* and */etc/group*
3. Start *nfsd*
4. Export file systems via */etc/exports*
5. Enable access to exported file systems with *exportfs*

Client:

1. Install software
2. Verify */etc/passwd* and */etc/group*
3. Start *biod*
4. Mount fie systems either on the command line, or edit */etc/fstab* for permanent mounts

Designing a File System

Before activating NFS, you need to architect your network file system.

Issues

- Speed
  - Network accesses will be slower than local access. Frequently accessed files should be local if possible. Examples:
    - */tmp*
    - application working directories
    - home directories
    - project directories
- Uids
  - User ids need to be consistent across the network. NFS makes access checks based on the uid, not username. Also, the following files must be consistent across the network:
    - */etc/passwd*
    - */etc/group*

/usr/spool/mail ← I should be shared by all hosts
/usr/mail

Designing a File System (cont'd)

- Point of failure

  You must balance the risk of network failure or system failure vs. hardware resources and convenience. Examples:

  - What if you are dependent upon NFS to access your application binaries?
  - What if you are dependent upon NFS to access your home directory?

  Conversely:

  - What if you need to send output to a device only supported on one machine? NFS eliminates the need for duplicate disk spooling space
  - Large data archives, such as the online man pages, can easily be shared avoiding the need for duplicate disk space on each machine

NFS and UNIX semantics

Some UNIX file system semantics are not preserved in NFS. These can be broken down into two categories:

1. Changes for security, particularly limiting root access
2. Changes because NFS is stateless

Root behavior on NFS:

- For access purposes uid 0, root, is converted to uid -2, anon
- Programs which are setuid or setgid will not have the effective uid and gid bits set at execution
- Root behavior can be controlled with mount options, which we will examining in the security section

NFS and UNIX semantics (cont'd)

Effects of statelessness:

- Files which are unlinked are actually just renamed until the file is closed
- Locking requires the support of the **`rpc.lockd`** and **`rpc.statd`** daemons. Without these daemons, locking is ignored
- NFS client descriptors can become out of date (stale) when the inode on the server is reused. For example, if a server is restored from tape and the NFS clients are not rebooted, stale handles result

UID mapping:

- NFS depends upon a homogeneous uid/gid interpretation across the net
- Requests from unidentified uid's are converted to the "anon" id for access control.

NFS Server Setup

1. Determine how many simultaneous NFS requests you need. Start this many copies of *nfsd* in *rc.local*.
2. Edit */etc/exports* to allow access to the file systems for exporting.

Example */etc/exports*:

```
/mnt/lang
/mnt/lang2
/tac -root=sushi
/tmp
/crash -root=sushi:saki:dhostwo:convexs:middlec:nlcvx:orion:rigel
/crash/tools -root=sushi:saki:dhostwo:convexs:middlec:nlcvx:orionT
/export -root=sunfish:sunfish.convex.com
/training -access=convext:convext.convex.com:toolman:toolman.convex.com
```

directory options

Selected */etc/exports* options:

- -access=host1:host1:..
  Allows host1 and host 2 (and hosts ...) to access this directory, if no access is specified, everyone has access.
- -ro
  Export this filesystem as read only
- -rw=host1:host2:...
  Read/write access is granted to the hosts on the list. All other hosts are granted read only access.
- -anon=<*value*>
  Change the value of anon from -2 to the value specified here. An id of -1 means do not allow access from unidentified ids.

NFS Server Setup (cont'd)

3. Run the command *exportfs* to inform clients of the file systems they can access.

   Example:

   ```
   # exportfs -a
   ```

   Note: *exportfs* also supports all the options you can set in */etc/exports*.

NFS Client Setup

1. Edit *rc.local* to start the number of **biod** daemons desired
2. Decide where in the local file system you wish to mount the remote file system
3. Decide whether you want to mount the filesystem as hard, soft, or interruptible (more later)
4. Try to mount the new file system

   # mount -o *<options>* *<host>*:*<filesystem>* *<dir>*
5. Add the new file system to */etc/fstab* for automatic mounts on reboot.

NFS Client Setup (cont'd)

Mount options:

hard
(default), retransmit until the operation succeeds. Not interruptible, nor does the request timeout even if the server does not respond (server crash or server too slow)

intr
(default), Allows the system call to be interrupted from the keyboard with the interrupt key sequence.

soft
Send an ETIMEOUT to the process and terminate the operation if you have to retransmit more than 3 or 4 times.

nolf
Don't allow creation of files over 2 gigabytes. A good idea if this is an NFS mounted file system since NFS can't handle the offsets necessary for large files.

fg
(default), If the mount fails the first time, retry in the foreground.

bg
Retry mounts in the background. Recommended for filesystems in /etc/fstab.

retry=<*num*>
(default 7), Number of times to retry a mount before failing.

timeo=<*num*>
(default 10,000), tenths of a second to elapse before timing out on this request.

NFS Client Setup (cont'd)

In addition, there are mount options which control the following:

- Root behavior
- Authenticated RPC
- Performance

These options will be discussed later.

Exercises

[1] Set up the SUN workstation to access the Convex manual pages. What other issues are there besides just setting up the NFS environment?

SUN: mkdir /mnt/man
chmod 777 /mnt/man
mount -o ro,soft convex:/usr/man /mnt/man
CONVEX: /etc/exports -> /usr/man
exportfs -a

[2] Which of the default mount options would you change and why?

NFS Security

NFS supports the following options to enhance the security of remote access:

Option	Purpose	**mount**	*fstab*	*exports*
nosuid	disallow execution of suid programs	Y	Y	N
root=<*host*>	allow root access from this host	Y	Y	Y
anon=-1	disallow access by unrecognized uids	Y	Y	Y

no entry in /etc/passwd

NFS Performance

A variety of options exist to enable optimal performance of NFS:

Option	Purpose	`mount`	*fstab*	*exports*
`async`	allow server to return once data is in buffer cache	Y	N	Y
`rsize, wsize`	set read and write packet size	Y	Y	Y
`retrans`	number of times to try this packet	Y	Y	Y
`timeo`	how long to wait before retrying	Y	Y	Y
`attribute caching`	allow the client to cache file attributes	Y	Y	N

it concerns data integrity and not security

NFS Performance (cont'd)

asynchronous operation

A server option that allows the server to take advantage of the Convex buffer cache

rsize, wsize

Mount options should be set equal to the media packet size. This is especially important for:

- WANs where fragmentation could cause excessive retries
- Serial Line ip

retrans

The current default, 5, should be raised for media with poor transmission capabilities

timeO

The current default, 10,000, should be raised for long haul networks

NFS Performance (cont'd)

Use *nfsstat* and *ps* to monitor performance and see if you have enough *nfsd*'s running.

Nfsstat sample output:

```
# nfsstat -s
Server rpc:
calls badcalls nullrecv badlen xdrcall
3827975 1 0 0 1

Server nfs:
calls badcalls
3827973 602
null getattr setattr root lookup readlink read
1751 0% 609298 15% 6334 0% 0 0% 2617372 68% 194677 5% 71913 1%
wrcache write create remove rename link symlink
0 0% 30463 0% 9114 0% 5804 0% 1066 0% 516 0% 13 0%
mkdir rmdir readdir fsstat
12 0% 0 0% 276162 7% 3478 0%
```

Ps sample output:

```
# ps -aux | grep nfsd | grep -v grep
root 436 0.0 0.0 188 4 ? S 44:16 /etc/nfsd 4 -m
root 434 0.0 0.0 188 4 ? S 43:03 /etc/nfsd 4 -m
root 435 0.0 0.0 188 4 ? S 46:32 /etc/nfsd 4 -m
root 437 0.0 0.0 188 4 ? S 50:53 /etc/nfsd 4 -m
root 433 0.0 0.0 216 4 ? D 0:00 /etc/nfsd 4 -m
```

NFS Performance (cont'd)

NFS also has the ability to cache attributes (those returned by the **getattr** system call) on the client. This:

- reduces the number of NFS calls made to the server to retrieve these attributes
- increases the risk of obtaining obsolete data
- should be used when nfsstat indicates a large percentage of **getattr** calls

Options are set on the *mount* command line or in */etc/fstab*.

NFS Performance (cont'd)

NFS **getattr** caching options:

Option	Value	Default
acregmin	hold attribute for <n> seconds after file modification	3
acregmax	hold attribute for no more than <n> seconds after file modification	60
acdirmin	hold attribute for <n> seconds after directory update	30
acdirmax	hold cached attributes for no more than <n> seconds after directory update	60
actimeo	set maximum time for files and directories *	<*n*>

* **actimeo** sets **acdirmax**, **acregman=**<*n*>, and sets **acregmin** and **acdirmin** to the defaults. Using this option is mutually exclusive with using the others.

NFS Performance (cont'd)

To determine if client attribute caching parameters need modification, use *nfsstat*:

```
# nfsstat -c
Client rpc:
calls badcalls retrans badxid timeout wait newcred
3092434 504 1774 393 2272 0 0

Client nfs:
calls badcalls nclget ncltoomany
3034574 277 3034574 0
null getattr setattr root lookup readlink read
0 0% 520331 17% 6685 0% 0 0% 1642862 54% 12226 0% 713919 23%
wrcache write create remove rename link symlink
0 0% 12344 0% 6734 0% 222 0% 66 0% 0 0% 4 0%
mkdir rmdir readdir fsstat
19 0% 3 0% 107370 3% 11789 0%
```

Note **`getattr`** calls in the range of 25-30% are normal. If yours are higher, you may wish to increase the time to live in the cache.

Attribute caching is enabled by default.

Exercises

[1] Determine the best performance options for your current setup. What options did you choose? How did you determine this was the best?

Automounter

CONVEX COMPUTER CORPORATION

Topics

- Purpose
- Operation
- Maps
- Master Map
- Direct Map
- Indirect Map
- Additional Information
- Direct Map Worked Example
- Indirect Map Worked Example
- Master Map Worked Example

Automounter Purpose

The automounter was designed to solve several common problems which occur in NFS environments. The automounter daemon:

1. allows users to control which file systems are mounted
2. is transparent, mostly, to users
3. eliminates the need for rarely used permanent mounts
4. provides a way of accessing replicated file systems from multiple servers

Automounter Operation

The automounter operates in the following way:

- A set of pathnames is tracked and an NFS mount is created when one of those paths is accessed.

  For example, assume */notes/devel* is normally mounter from server engr.com. With automounter, the following would occur:

User	Kernel/Automounter
% **ls notes** %	
% **cd /notes/devel**	(after cd, kernel maintains control...)
	Calls automounter daemon
	Automounter mounts /devel from engr.com
	Control returns to user
% **ls** foo bar junk	

- The administrator decides which file systems or directories should be mounted by NFS statically from */etc/fstab* or dynamically by the automounter
- Automounter keeps track of pathnames and servers in a set of files called maps

Automounter Maps

The automounter manages filesystems through a set of maps. There are three types of maps:

1. The master map
2. Indirect maps
3. Direct maps

Each map entry contains the folowing information:

- The pathname for automounter to manage (the **key**)
- The server(s) name(s) to mount the file system from, or the map name to find the servers in
- Any mount options required

Master Map

The master map:

- contains all the references to direct and indirect maps
- supports an optional special entry type
- is usually named */etc/auto.master*
- provides the simplest invocation of the automounter

Example:

```
# automount -f /etc/auto.master
```

Master Map (cont'd)

Example master map file:

```
# Mount-point          Map                  [ Mount Options ]
/net                   -hosts               -hard,intr,timeo=20
/-                     /etc/auto.direct     -soft,ro,intr,timeo=20
/-                     /etc/auto.SIS        -soft,ro,intr,timeo=20
/-                     /etc/mikel.direct    -soft,ro,intr,timeo=20
```

Key – the root of the mount point, or "-" for direct maps

Mapfile – the full path name of the map file except in the case of special entries

Mount Options – any standard mount options

Note:

- the maps are specified as full pathnames
- direct maps begin with "/-"
- indirect maps begin with the root of the filesystem
- a special map "-hosts" exists (more later)

Direct Maps

Direct maps:

- Manage full pathnames
- similar to */etc/fstab*

Example:

```
/usr/man          -ro,soft          oak:/usr/man
/fonts            -ro,hard,intr     willow:/usr/X11/fonts ash:/fonts
/usr/games        -ro,soft,nosui    flip:/usr/games\
                                    dolphin:/usr/games
```

full pathname of the local mount point

any standard mount options

a white-space separated list of server/pathname pairs

Note:
The */etc/fstab* equivalent for the direct map

```
/usr/man          -ro,soft      oak:/usr/man
```

is

```
oak:/usr/man      -ro,soft      /usr/man
```

There is no *fstab* equivalent for multiple servers. Automounter chooses a server by sending an RPC procedure 0 request to each of the NFS servers. The server who responds first is the one automounter uses.

Exercises

[1] Create a direct map that allows you to mount the Sun's manual pages on the Convex. Create a master map file to access the direct map. Start the automounter. What mount options did you use and why?

Indirect Maps

- Manage mount points that are not full pathnames
- Typically an entire subdirectory of automounter mounts.

Example

```
willow                                  willow:/home/willow
peach                   -rw,nosuid      peach:/export/home
```

relative pathname of the local mount point

any standard mount options

a white-space separated list of server/pathname pairs

Additional Information

- For every entry except direct map entires, automounter adds a mount entry to */etc/mtab*.
- Automounter really mounts in */tmp_mnt* and then creates a symlink from the mountpoint defined in the map to */tmp_mnt*.
- You can change the default mount directory for automounter by using the **-M** option when you start the automounter.
- Automounter times out mounts after a certain time and unmounts the file system (default 5 minutes). You can reset this with the **-t** option on the automount command line.
- Automounter uses mount options with the following precedence:
  1. automounter command line
  2. direct or indirect map entry
  3. master map
- Because automounter creates mount points for indirect maps and special maps, no other files (regular or NFS) may reside in these directories[1].

1. Or they will be shadowed by the file system automounter mounts there.

Direct Map Worked Example

Assume the following direct map entry:

```
/usr/man          -ro,soft          oak:/usr/man rose:/usr/man
```

When automounter is started it:

1. "fakes" a mount for */usr/man*

```
willow:(pid 92) /usr/man ignore ro,intr,port=679,map=/etc/auto.direct, direct 0 0
```

When */usr/man* is accessed, automounter executes the following commands:

2. contacts servers oak and rose with a null RPC call
3. **`mkdir /tmp_mnt/usr/man`**
4. **`mount oak:/usr/man /tmp_mnt/usr/man`**
5. **`ln -s /tmp_mnt/usr/man /usr/man`**

Indirect Map Worked Example

Assume the following indirect map entry and corresponding master map entry:

```
john            -rw,hard,intr       oak:/home/john
/home           auto.home           -rw,intr
```

When automounter is started, it:

1. "fakes" a mount for */home* in */etc/mtab*

```
willow:(pid 92) home nfs rw,intr,port=679,map=/etc/auto.home,indirect 0 0
```

   When */home/john* is accessed, automounter executes the following commands:

2. contacts server oak with a null RPC call
3. **`mkdir /tmp_mnt/home/john`**
4. **`mount oak:/home/john /tmp_mnt/home/john`**
5. **`ln -s /tmp_mnt/home/john /home/john`**

Master Map Worked Example

The master map contains a special map type (**-hosts**) which allows the user to mount all filesystems that are exported from a particular host:

```
/net:                    -hosts
```

Note the map file is NOT a full pathname.

When automounter encounters the -hosts map, it:

1. "fakes" a mount in */etc/mtab*:

```
willow:(pid 92) /net nfs ro,intr,port=679,map=-hosts,indirect 0 0
```

Upon access to */net/hostname*, automounter:

2. sends the null RPC call to the server (hostname)
3. requests a list of all exportable file systems
4. sorts the list by ascending size (to ensure proper mounting order)
5. mounts all available filesystems in */tmp_mnt/net/hostname/...*
6. **`ln -s /tmp_mnt/net/hostname /net/hostname`**

Network Information Service

CONVEX COMPUTER CORPORATION

Topics

- Problem
- Purpose
- Suitability
- Architecture
- Master Server
- File Access vs. Map Access
- NIS Client
- NIS – Testing a setup
- NIS Slave Servers
- NIS Passwords

Problem

Previously called "Yellow Pages," (YP), the Network Information Service (NIS) was designed by Sun to solve the following problem:

A distributed environment requires consistency in certain UNIX database files is order for function properly. For example:

- */etc/passwd*
- *etc/group*
- */etc/hosts*

Purpose

NIS was designed to provide:

- A single point of administration for common configuration files
- A client/server database access capability
- Network-wide consistency of critical information

Suitability

NIS is only suitable for files that do not contain machine-dependent information. Files which are not suitable are:

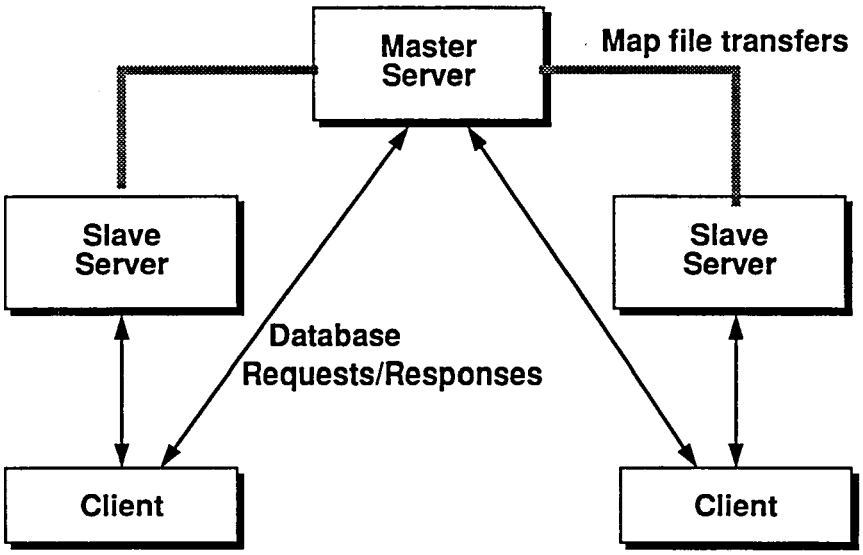
- */etc/fstab*
- */etc/rc*

Architecture

NIS is an RPC-based service that consists of:

- A master server
- Optional slave servers
- Clients

Once a file is managed by NIS, it is called a **map.**



Master Server

To create a master server:

1. Decide which files you wish to manage via NIS
2. Ensure the current versions of the files accurately reflect your network information
3. Set the domainname[1]
4. Initialize the maps
5. Start **`ypserv`** (the master server) from */etc/rc.local*[2]

** You need to be able to broadcast to every machine in the domain for NIS to work. Be careful about subnetting.

1. A group of machines working in a client/server paradigm
2. To use the maps from this machine, you also need to be a client

Master Server (cont'd)

1. Select which files to manage with NIS:

File	Recommended
/etc/ethers	Y
/etc/group	Y
/etc/hosts	Y
/etc/networks	Y
/etc/passwd	Y
/etc/protocols	N
/etc/rpc	N *
/etc/services	N **
/usr/lib/aliases	Y
/etc/netgroup	Y
/etc/netids	Y

* If you choose to manage /etc/rcp via NIS, recall that the tpdaemon is a Convex RPC service and must be added to the map if you are serving the Convex from a non-Convex machine.

** Required for non-convex master and Convex slaves.

2. Verify the contents of the selected files

3. Set the domainname[1]

   A group of systems that share the same set of NIS maps is referred to as a domain.

   The *domainname* utility sets a system's domain name.

   Example:

   # /bin/domainname mydomain

4. Initialize the maps

   - cd to */usr/etc/yp*
   - edit the "all" line in *Makefile* to match your maps

```
STDHOSTS= $(YPDIR)/stdhosts
MKNETID=$(YPDIR)/mknetid
MKALIAS=$(YPDIR)/mkalias
CHECKDOM=$(YPDIR)/checkdom

# If you wish to use the auto.master and auto.direct YP maps,
# add them as targets here.
#
all: domainname ypservers passwd group hosts ethers networks rpc services
\\p        protocols pwrestrict netgroup bootparams aliases publickey netid
#       protocols pwrestrict netgroup bootparams aliases netid

domainname:
        @$(CHECKDOM)
```

Partial /etc/yp/Makefile

   - build the maps with:

     # ypinit -m

1. Not the mail domain name used by sendmail.

5. Start the server

   Portion of */etc/rc.local*:

```
if [ -f /bin/domainname ]; then
        /bin/domainname `hostname`        # only set if Yellow Pages is used
fi


if [ -f /bin/domainname -a "`/bin/domainname`" != "" ]; then
        if [ -f /usr/etc/ypserv -a -d /etc/yp/`/bin/domainname` ]; then
                /usr/etc/ypserv -i & echo -n ' ypserv'
        fi
        if [ -f /etc/ypbind ]; then
                /etc/ypbind & echo -n ' ypbind'
        fi
fi
```

Note: We start *ypbind* since most servers are also clients of their own maps.

File Access vs. Map Access

Once you convert to NIS, it is important to understand where in formation is actually coming from.

Files always consulted before maps:

- */etc/passwd*
- */etc/group*
- */usr/lib/aliases*

Files never consulted before maps:

- */etc/services*
- */etc/protocols*
- */etc/networks*
- */etc/netgroup*
- */etc/bootparams*
- */etc/ethers*
- */etc/hosts*[1]

1. /etc/hosts is used briefly during booting, so valid local entries still reside here.

NIS Client

To convert a system from local file access to NIS maps, execute the following steps:

1. Edit local files that will be consulted during NIS operation
2. Set client's domain name
3. Start *ypbind.*

NIS Client (cont'd)

Editing local files:

1. Save a copy of all local files
2. Leave enough information so the system will function if the NIS server goes down
3. Include NIS map information with "+" operator
4. Exclude NIS map information with the "-" operator
5. Precedence:
    - local file entries
    - excluded map information
    - any included map information

NIS Client (cont'd)

Edit */etc/hosts*:

- include your host
- include localhost
- include everything else from NIS with the "+" operator

Example:

```
127.1 localhost loopback loop
#
#
139.130.1.1 sid sid.vicious.com
+
```

Note: You don't need to include the NIS master host since *ypbind* broadcasts for the server to use.

NIS Client (cont'd)

Edit */etc/group*:

- include local groups, and any groups essential to the operation of the system without NIS
- exclude individual groups from map with "-group"
- include all else with "+:*:*"[1]

Example:

```
root:*:0:
daemon:*:1:
sys:*:2:
+:*:*
```

1. We use "+:*:*" so that when NIS is down, "+" does not become a valid group

NIS Client (cont'd)

Edit /etc/passwd:

- leave in enough entries for proper single user operation
- exclude unwanted users: "-username"
- include the rest of the map: "+:*:0:0:::"

Example:

```
root:9wxntql2hHt:0:1:/:/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
uucp:*:4:4::/usr/spool/uucppublic:
+:*:0:0:::
```

Continue with all files you wish to manage under NIS.

NIS – Testing a setup

Since NIS maps are accessed through RPC calls, standard ConvexOS commands, such as *cat(1)* cannot access the information stored there. The following commands provide information about the NIS environment and maps:

- Display the domainname:

  % /bin/domainname

- Show which server we are using:

  % /usr/bin/ypwhich

- Access information out of a NIS map:

  % ypmatch *<key> <mapname>*

- Print an entire map:

  % ypcat *<mapname>*

Note: mapnames and the lookup keys are listed in */usr/etc/yp/Makefile*.

Exercises

[1] Set up the NIS master server. Which files are you going to manage and why?

[2] Set up the client without modifying any files. Does the system still operate as you expect?

[3] Set up the client's */etc/passwd* file. Ensure the proper accounts are installed in */etc/passwd.* Which accounts do you need and why?

[4] Users are complaining that they can't log in. What could be the problems?

NIS Slave Servers

To add redundancy to a NIS environment, you may designate some systems as slave servers. A slave server:

- contains the same information as the master with a slight time delay
- does not need copies of the original seed information for the maps
- can respond to NIS map requests when the master server is down

Note: You can always get the contents of a map into a format suitable for creating a slave server with *ypcat*.

NIS Slave Servers (cont'd)

To create a slave server, execute the following on the slave:

1. **`ypinit -s`** *<master_server>*
2. Start *ypserv* and *ypbind* from */etc/rc.local*
3. On the master, set up a cron entry to update the slave servers. Several scripts in */usr/etc/yp* are available. They use the command *ypxfr*.

NIS passwords

You can choose to manage passwords locally or via the passwd maps.

To manage passwords via NIS:

1. Start yppasswdd on the client from */etc/rc.local*

   Sample portion of */etc/rc.local*:

```
if [ -f /usr/etc/rpc.yppasswdd ]; then
   /usr/etc/rpc.yppasswdd /etc/passwd /etc/pwrestrict \
  -m passwd pwrestrict DIR=/etc & echo -n ' yppasswdd'
fi
```

   Note: If you are not running password aging, you must delete the information underlined in this portion of */etc/rc.local*.

2. move */bin/passwd* to */bin/passwd.local*
3. move */usr/bin/yppasswd* to */bin/passwd*

Domain Name Server

CONVEX COMPUTER CORPORATION

Topics:

- IP Addresses and Host names
- BIND
- Server Setup
- Primary Servers
- Secondary Servers
- Resolving-only Servers
- Caching-only Server
- Server Operations
- Side Effects

IP Addresses and Host Names

Every network interface on the Internet has some form of hardware address. To make these interfaces easier to use, and provide a means of logical and physical partitioning, IP addresses are used.

Another, more abstract, means of reference to interfaces is the use of a symbolic name, or hostname. An Internet site may use the */etc/hosts* file to map IP addresses to symbolic names for hosts at that site.

But what about maintaining */etc/hosts* for every device on the entire internet? To do this manually would be a monumental[1] task and prone to errors.

An additional problem is that a central authority is required to maintain the name list so names don't conflict.

1. According to Comer, *Internetworking with TCP/IP Vol. I*, p. 312, there were 137,000 hostnames on the internet in 1990.

IP Addresses and Host Names (cont'd)

The solution:

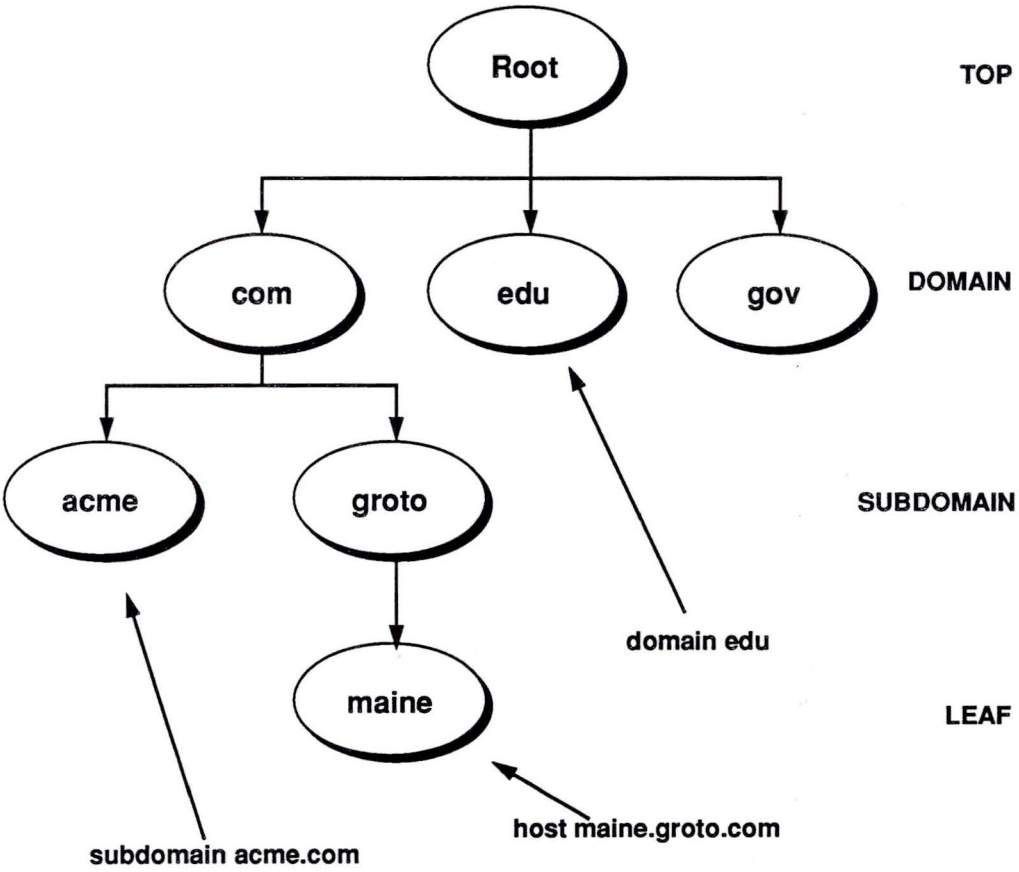
- Define a host naming hierarchy (domain names)
- Define a protocol to traverse the hierarchy and resolve a name into an internet address
- Store hostname/IP address maps in servers instead of ASCII files. (**named**, "*name -d*")
- Partition names and delegate naming authority

This methodology is referred to as the Domain Name Service.[1]

1. Note this reference to "Domain" is not related to the "domain" used by NIS.

IP Addresses and Host Names (cont'd)

Host names use a hierarchical naming system:



IP Addresses and Host Names (cont'd)

In this name hierarchy, a host name is fully qualified when it contains:

- a domain name
- subdomain names (optional, may be multiple)
- a leaf name

A host wishing to participate on the Internet must use a fully qualified name, and conform to the DNS naming conventions.

A host which is physically connected to the internet must run the **named** software.

BIND

Convex implements DNS with the Berkeley Internet Name Domain server (BIND), **routed.**

The Server of Authority (SOA) is the daemon ultimately responsible for providing the IP address for a requested hostname.

Servers in charge of large domains (for example, .com) may delegate authority to other servers.

A server that is a SOA is called a *primary* server, and loads its database from disk.

In addition to primary servers, there are also other types of servers:

- secondary
- caching
- resolving

Server Setup

Before you set up a server, you need to know:

1. The domain of authority, if any
2. The IP address of master servers to which unresolved requests are forwarded
3. Whether you intend to run the name server on all machines on the local network

Primary Servers

Primary servers convert */etc/hosts* into a database. They require disk space for the databases and can respond to requests from other servers. A makefile is provided to create the nameserver databases.

To set up a primary server:

1. Ensure that */etc/hosts* contains the correct information for the nameserver databases
2. **cd** to */usr/lib/conf/bind/setup*
3. Edit *Makefile*:
   - specify the type of server
   - your domain name
   - any other defaults you wish to override
4. Execute:
   make >& make.out
5. Copy *named.boot* to */etc* or execute **make install**
6. Start the name server daemon, **named**
7. Touch */etc/use_nameserver*

Secondary Servers

- Delegated authority by a primary server
- Receives tables from a primary server at start-up
- Periodically polls the primary server to update its tables

Resolving-only Servers

- Contain no host information themselves
- Always query a designated server to resolve hostnames
- Require less memory – designed for diskless workstations

Resolving-only Servers (cont'd)

Resolving servers always access another server for their information. They are smaller and don't require any updates, but they are slower.

To set up a resolving-only server:

- Set domain in */etc/resolv.conf*
- Set nameservers in */etc/resolv.conf*. Use dot notation for the IP address of the nameservers(s)
- Touch */etc/use_nameserver*

To test a resolver use *nslookup.*

Nslookup:

- interactive query utility
- use "?" for help
- specifying a fully-qualified hostname will determine whether the host is accessible or not.

Caching-only Server

- Not an authority
- Caches information it receives until time-to-live expires
- Requests information from other primary servers

Exercises

[1] Which type of name server would generate the most network traffic?

[2] Set up the Convex as a resolving-only server. Your users are complaining that rsh doesn't work any more. What is the problem?

Server Operations

Debugging the nameserver database

- *named* stores its pid in */etc/named.pid*
- sending SIGINT to named causes it to dump its data into */usr/tmp/named_dump.db*

```
kill -INT `cat /etc/named.pid`
```

- data in */usr/tmp/named_dump.db* should match data in the *named.hosts* file created previously
- if the databases do not match:
  1. disable named by removing */etc/use_nameserver*
  2. determine the correct information and fix */etc/hosts*
  3. go to step 2 for configuring primary servers, page 8.

Server Operations (cont'd)

The *named* service can be enabled or disabled at any time:

To enable the name server:

```
touch /etc/use_nameserver
```

To disable the name server:

```
rm /etc/use_nameserver
```

When the name server is not running, hostname resolution is handled either by the Network Information System (NIS) or */etc/hosts*. Therefore, it is important to maintain a reasonable local hosts file, even if you are using named to access the Internet.

Side Effects

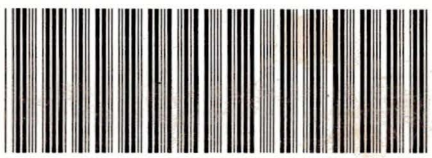
1. Hostnames are fully-qualified when the name server is running
2. Changes to a published internet number are difficult because many primary servers load their databases from their own disk images.

Exercises

[1] BIND is a protocol replacement for which ASCII file?

[2] Follow the steps listed to set up a primary server. Test access to a well-known internet address. Use nslookup, finger, or anonymous ftp to verify correct operation.

50.115
50.10



Document Number
710-026430-000