

## Daemons in the Night and Basic Networking

ITSC 1402/KRF

1

## The Story of TCP/IP

- ❑ The Army puts out a bid request for computers
  - ◆ DEC wins the bid
- ❑ The Air Force puts out a computer bid request
  - ◆ IBM wins the bid
- ❑ The Navy puts out a bid request
  - ◆ Unisys wins
- ❑ The President decides to invade Grenada
  - ◆ Oops, the armed forces discover their computer systems can't talk to each other
- ❑ DoD must find a way to integrate these separate systems

ITSC 1402/KRF

2

## Network Protocols

- ❑ Two principal protocols in use
  - ◆ TCP/IP (Transmission Control Protocol/Internet Protocol)
  - ◆ UDP/IP (User Datagram Protocol/Internet Protocol)
- ❑ Developed for the US DoD to interconnect networks designed by different vendors into a seamless and robust "network of networks"
- ❑ Due to the design, the network can sustain considerable damage and still provide robust communication
  - ◆ Of course, that same robustness can mask network problems for long periods of time

ITSC 1402/KRF

3

## The Layered Approach

- ❑ TCP/IP and UDP/IP are composed of layers
  - ◆ The Internet Protocol (IP) is responsible for moving data packets from node to node.
    - Each node is identified by a four byte destination address (the IP address)
  - ◆ The Transmission Control Protocol (TCP) is a connection-oriented service that is responsible for verifying correct delivery of data
  - ◆ The User Datagram Protocol (UDP) is a connection-less service that is mainly useful for broadcasting packets across a network

ITSC 1402/KRF

4

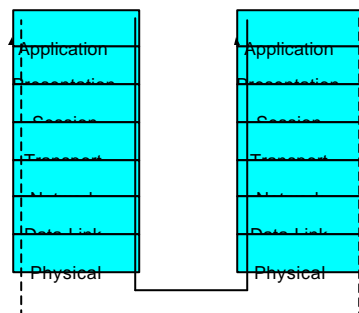
## Understanding TCP/IP Networking

- ❑ Networking is an inherently complex problem
- ❑ Much of this complexity is hidden from normal view but as administrators, you will occasionally (or regularly) have to deal with it
- ❑ One of the major issues is controlling network access
  - ◆ If network programs were allowed to control the network hardware directly, chaos would result
- ❑ The solution to this is the *network stack*

ITSC 1402/KRF

5

## OSI Network Stacks



ITSC 1402/KRF

6

## Stack Components



- ❑ Application layer – applications such as telnet, ftp, and e-mail
- ❑ Presentation layer – concerned with information representation, such as data compression and cryptography – all other layers treat the data as simply a stream of bits
- ❑ Session layer – allows users on different machines to establish sessions between them – session layer worries about synchronization and dialog control (who's turn is it to talk if communication is only allowed one way at a time)

ITSC 1402/KRF

7

- ❑ Transport layer – accepts data from the session layer, splits it into smaller packets, and ensures they all arrive correctly at the other end; reassembles packets on receipt from the other system
- ❑ Network layer – controls operation of the subnet; key issues is deciding how packets are routed from source to destination
- ❑ Data Link layer – takes a raw transmission facility and transforms it into what appears to higher layers as an error-free communications channel

ITSC 1402/KRF

8

- ❑ Physical layer – includes the network interface card (NIC) and cabling that connects to the network



ITSC 1402/KRF

9

## Hostnames

- ❑ Most computers on a TCP/IP network are given a name (the *hostname*)
- ❑ Although on the local network, the machine may be known by a simple name, such as *clyde*, or *phred*, or *unixserv*, for use on the Internet, the machine needs a *fully qualified domain name*
- ❑ A fully qualified domain name has the format: `hostname.site.domain.country`



ITSC 1402/KRF

10

## Fully Qualified Domain Names



- ❑ Hostname – a name by which the computer is known; it must be unique within the local network
- ❑ Site – a name given to the site, such as *dcccd*, or *ibm*, or *yahoo*
- ❑ Domain – each site belongs to a specific domain. Domains group sites with similar purpose together
  - ◆ *edu* – educational institutions
  - ◆ *com* – commercial companies
  - ◆ *gov* – governmental departments and agencies
  - ◆ *net* – networking companies
- ❑ Country, such as *au*, *uk*, or *fr*
  - ◆ *us* or nothing at all refers to United States

ITSC 1402/KRF

11

- ❑ In Linux, root can set the hostname via the *hostname* command
  - ◆ Non-root users can use *hostname* to view the hostname for the machine
- ❑ It isn't always necessary to use the fully qualified name
- ❑ Within a local net, you can usually use only the hostname
  - ◆ For instance, when logged onto *clyde.dcccd.edu*, you can refer to *phred.dcccd.edu* by just *phred*



ITSC 1402/KRF

12

## Network Addresses



- Although alpha names are nice for people, they are not efficient for computers
- IP addresses are currently 32 bit numbers
  - ◆ IPv6 uses 128 bit addresses
- IP addresses are written as four numbers separated by . (dots) such as 144.162.120.231
  - ◆ Since IP addresses are 32 bits, dividing them into four groups gives 8 bits per group so the numbers can range from 0 – 255 in each group
- 144.162.120.231 is stored in the computer as 10010000 10100010 01111000 11100111

ITSC 1402/KRF

13

## Binary to Decimal Conversion



- To convert an 8 bit binary number to decimal, you add the weights of each "one" bit
- X X X X X X X X
  - └─ 1
  - └─ 2
  - └─ 4
  - └─ 8
  - └─ 16
  - └─ 32
  - └─ 64
  - └─ 128

ITSC 1402/KRF

14

## Examples



- 10110010
  - ◆  $128 + 32 + 16 + 2 = 178$
- 11100111
  - ◆  $128 + 64 + 32 + 4 + 2 + 1 = 231$
- 01111000
  - ◆  $64 + 32 + 16 + 8 = 120$

ITSC 1402/KRF

15

## Net Masks



- IP addresses actually have two parts
  - ◆ Host address
  - ◆ Network address
- The network address defines to which network the host belongs
- The host portion identifies a specific machine on that network
- The network portion is the high order portion of the IP address
- The portion of the IP address that is the network portion is defined by another 32 bit number, the netmask

ITSC 1402/KRF

16

- To figure out the network address and host address, the netmask and IP address are treated as binary numbers and compared
  - ◆ For each bit that is set (a one) in the netmask and the IP, the corresponding bit is set in the network address
  - ◆ For each bit that is NOT set in the netmask but is set in the IP address, the corresponding bit is set in the host address



ITSC 1402/KRF

17

## Example



- IP Address  
144.162.120.231 = 10010000 10100010 01111000 11100111
- Netmask  
255.255.0.0 = 11111111 11111111 00000000 00000000
- Network Address  
= 10010000 10100010 00000000 00000000
- = 144 162 0 0
- Host Address  
= 00000000 00000000 01111000 11100111
- = 0 0 120 231

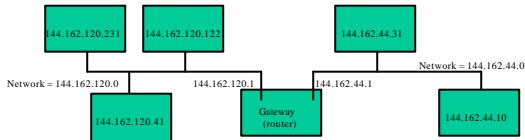
- Although this example was fairly simple and didn't really need binary, when you work with subnets it gets much more confusing if you don't use the binary technique

ITSC 1402/KRF

18

## Internet as a Network of Networks

- Each local network is a collection of machines with the same network address
- These networks are then connected together to form the Internet



ITSC 1402/KRF

19

## Network Classes

- There are four network classes defined by IP and they have been assigned the following ranges of addresses

Network Class	Netmask	Network Addresses
A	255.0.0.0	0.0.0.0 - 127.255.255.255
B	255.255.0.0	128.0.0.0 - 191.255.255.255
C	255.255.255.0	192.0.0.0 - 223.255.255.255
Multicast	240.0.0.0	224.0.0.0 - 239.255.255.255

ITSC 1402/KRF

20

## Private Network Classes

- Each of the primary network classes defined by IP also has a range of private addresses that are not legal for use on the Internet

Network Class	Netmask	Network Addresses
A	255.0.0.0	10.0.0.0 - 10.255.255.255
B	255.255.0.0	172.16.0.0 - 172.31.255.255
C	255.255.255.0	192.168.0.0 - 192.168.255.255

ITSC 1402/KRF

21

## Assigning IP Addresses

- Some IP addresses are commonly used and should not be used for normal machines
  - ◆ x.x.x.0 – network address
  - ◆ x.x.x.255 – broadcast address
  - ◆ 127.0.0.1 – loopback address

ITSC 1402/KRF

22

## Name Resolution

- How do we get from the people-friendly computer name to the efficient IP address?
  - ◆ The process of doing this is called *name resolution*
- There are two methods of name resolution
  - ◆ The `/etc/hosts` file
  - ◆ The Domain Name Service

ITSC 1402/KRF

23

## `/etc/hosts`

- Name resolution can be accomplished by maintaining a file of IP addresses and their corresponding host names
  - ◆ Unix uses `/etc/hosts` for this
- Each line has the format:
  - IP\_Address hostname aliases
  - ◆ # symbols can be used for comment lines

ITSC 1402/KRF

24



□ A portion of clyde's `/etc/hosts` file looks like:  
 # Internet host table

```
127.0.0.1      localhost
144.162.120.231 clyde.dcccd.edu  clyde
192.168.128.1  clyde-hme1 # SUNRAY ADD
144.162.120.232 bonnie.dcccd.edu  bonnie
144.162.120.233 phred.dcccd.edu  phred
144.162.120.235 r2d2.dcccd.edu   r2d2
144.162.120.237 hawkeye
144.162.120.240 proton
144.162.121.238 hp_144          #printer_254
144.162.121.237 hp_144d        #printer_253
```

## Problems With `/etc/hosts`



- What happens when you `http://www.krfrazer.com` from clyde?
- With only the `/etc/hosts` file, which doesn't contain an entry for `krfrazer.com`, you are stuck
- Of course, you could always add the entry
  - ◆ But what about all the other machines you want to access?
- You could also add each of them
  - ◆ With several million machines on the Internet, this is not feasible

## Domain Name Service (DNS)



- DNS is arranged as a hierarchy, much like a file system
- At the top is the root domain "." which is administered by the Internet Assigned Numbers Authority (IANA)
- The process of assigning a domain to an organization is known as "delegating"
  - ◆ This means that the organization must be willing and capable of providing the domain name server for its domain

## Domain Name Servers



- The Domain Name System is implemented as collection of inter-communicating nameservers
- At any given level of the DNS hierarchy, a nameserver for a domain has knowledge of all the immediate sub-domains of that domain.
- For each domain there is a primary nameserver, which contains authoritative information regarding Internet entities within that domain.
- Secondary nameservers can be configured, which periodically download authoritative data from the primary server

- Secondary nameservers provide backup to the primary nameserver when it is not operational
  - ◆ They also improve the performance of the DNS, since the nameservers of a domain that respond to queries most quickly are used in preference to any others.

## `/etc/resolv.conf`



- When performing a name resolution most UNIX machines will check their `/etc/hosts` first and then check with their name server.
- How does the machine know where its domain name server is?
- The answer is in the `/etc/resolv.conf` file.
- `resolv.conf` is a text file with three main types of entries

## /etc/resolv.conf Components



- # comments
  - ◆ Anything after a # is a comment and ignored
- Domain *name*
  - ◆ Defines the default domain. This default domain will be appended to any hostname that does not contain a dot.
- Nameserver *address*
  - ◆ This defines the IP address of the machine's domain name server
  - ◆ It is possible to have multiple name servers defined and they will be queried in order (useful if one goes down)

ITSC 1402/KRF

31

## /etc/nsswitch.conf



- *nsswitch.conf* - System Databases and Name Service Switch configuration file
- Various functions in the C Library need to be configured to work correctly in the local environment
  - ◆ Traditionally, this was done by using files (e.g., `/etc/passwd`), but other nameservices (like the Network Information Service (NIS) and the Domain Name Service (DNS)) became popular, and were hacked into the C library, usually with a fixed search order
- The Linux libc5 and the GNU C Library 2.x (libc.so.6) contain a cleaner solution of this problem
  - ◆ It is designed after a method used by Sun Microsystems in the C library of Solaris 2 called "Name Service Switch" (NSS)
  - ◆ The sources for the "databases" and their lookup order are specified in the `/etc/nsswitch.conf` file

ITSC 1402/KRF

32

## NSS Contents



- These databases are available in the NSS:
  - ◆ aliases
  - ◆ Ethers - Ethernet numbers.
  - ◆ hosts - Host names and numbers
  - ◆ passwd - User passwords
  - ◆ rpc - Remote procedure call names and numbers
  - ◆ shadow - Shadow user passwords
  - ◆ Along with numerous others

ITSC 1402/KRF

33

- An example `/etc/nsswitch.conf` file might look something like:

```
passwd:      compat
group:       compat
shadow:      compat

hosts:       dns [UNAVAIL=return] files
networks:    nis [NOTFOUND=return] files
ethers:      nis [NOTFOUND=return] files
protocols:   nis [NOTFOUND=return] files
rpc:         nis [NOTFOUND=return] files
services:    nis [NOTFOUND=return] files
```

ITSC 1402/KRF

34

## /etc/nsswitch.conf Entries



- The first column is the database
  - ◆ The rest of the line specifies how the lookup process works
    - You can specify the way it works for each database individually.
- The configuration specification for each database can contain two different items:
  - ◆ The service specification like 'files', 'db', or 'nis'.
  - ◆ The reaction on lookup result like '[NOTFOUND=return]'.
- For Linux libc5, the allowed service specifications are 'files', 'nis' and 'nisplus'
  - ◆ For hosts, you could specify 'dns' as an extra service

ITSC 1402/KRF

35

- The second item in the specification gives the user much finer control on the lookup process
- Action items are placed between two service names and are written within brackets. The general form is
  - ◆ '[ ( '!' STATUS '=' ACTION )+ ]'
- where
  - ◆ STATUS => success | notfound | unavail | tryagain
  - ◆ ACTION => return | continue

- The case of the keywords is insignificant

ITSC 1402/KRF

36

## Status Values



- The STATUS values are the results of a call to a lookup function of a specific service. They mean:
  - ◆ *success* - No error occurred and the wanted entry is returned. Default action for this is 'return'.
  - ◆ *notfound* - The lookup process works ok but the needed value was not found. Default action is 'continue'.
  - ◆ *unavail* - The service is permanently unavailable. This may mean the needed file is not available, or, for DNS, the server is not available or doesn't allow queries. Default action is 'continue'.
  - ◆ *tryagain* - The service is temporarily unavailable. This might mean a file is locked or a server currently cannot accept more connections. Default action is 'continue'.
- NOTE: Each process that uses */etc/nsswitch.conf* reads the file only once
  - ◆ If the file is later changed, the process will continue using the old configuration

ITSC 1402/KRF

37

## Routing



- Routing is the act of deciding how each individual packet finds its way through the multiple different paths to its destination.
- For most UNIX computers the routing decisions they must make are simple
  - ◆ If the packet is for a host on the local network then the data is placed on the local network and delivered to the destination host
  - ◆ If the destination host is on a remote network then the packet will be forwarded to the local gateway for forwarding
- However, a network the size of the Internet cannot be constructed with such a simple approach

ITSC 1402/KRF

38

## Routing Tables



- Routing is concerned with finding the right **network** for a packet
- Once the right network has been found the packet can be delivered to the host
- Most hosts (and gateways) on the Internet maintain a routing table
- The entries in the routing table contain the information to know where to send packets for a particular network.

ITSC 1402/KRF

39

## Building the Routing Table



- The routing table can be constructed in one of two ways
  - ◆ Created by the Systems Administrator, sometimes referred to as static routes
  - ◆ Dynamically created by a number of different available routing protocols
- The dynamic creation by routing protocols is complex and beyond the scope of this class

ITSC 1402/KRF

40

## Ports



- Ports are a way to specify a pseudo-address, or sub-address for your network card
- This allows multiple network services to work on a single Ethernet interface without their data getting all scrambled
- Most network services have pre-assigned port numbers, known as "well-known ports"
  - ◆ These are listed in RFC1700
- Ports 0-1023 are pre-assigned and should not be used for your own "special" projects

ITSC 1402/KRF

41

## Address Resolution Protocol



- The mapping of 6 byte Ethernet addresses into 4 byte Internet addresses is performed by the Address Resolution Protocol (ARP)
- ARP maintains a table that translates between IP address and Ethernet address
- When the machine wants to send data to a computer on the local Ethernet network the ARP software is asked if it knows about the IP address of the machine
- If the ARP table contains the IP address the Ethernet address is returned

ITSC 1402/KRF

42



- ❑ If the IP address is not known a packet is broadcast to every host on the local network; the packet contains the required IP address
- ❑ Every host on the network examines the packet
- ❑ If the receiving host recognizes the IP address as its own, it will send a reply back that contains its Ethernet address.
- ❑ This response is then placed into the ARP table of the original machine (so it knows it next time)



- ❑ The ARP table will only contain Ethernet addresses for machines on the local network.
- ❑ Delivery of information to machines not on the local network requires the intervention of routing software
- ❑ On a UNIX machine you can view the contents of the ARP table using the *arp* command
  - ◆ *arp -a* will display the entire table

## Configuring TCP/IP Networking



- ❑ The configuration process includes the following steps
  - ◆ Configure the devices
    - Done either at system startup time (ethernet and other permanent connections) or by a user program (dial-up connections such as PPP)
    - This process configures the network devices with the appropriate information including IP address, network address etc.
  - ◆ Configure the name resolver
    - This step sets up the DNS so that your system can translate IP addresses into hostnames and vice versa



- ❑ Configure routing
  - ◆ Informs the system how it is meant to send information from one network to another

## Configuration Tools and Files



- ❑ Command line tools such as *ifconfig*
  - ◆ The standard tool used to perform the configuration; you might use this from the command line or it might be performed by startup scripts.
- ❑ Device configuration files
  - ◆ Most versions of Unix use text-based files to store the configuration information, such as IP address, netmask and broadcast addresses for the various devices used by *ifconfig*
  - ◆ Suse uses */etc/rc.config*
  - ◆ RedHat uses the */etc/sysconfig/network* file and the */etc/sysconfig/network-scripts* directory to contain these files which are used as the system starts up



- ◆ Solaris uses */etc/hostname.<ifname>*, */etc/nodename*, */etc/inet/netmasks*, and */etc/inet/hosts*
- ❑ Network configuration files
  - ◆ These files provide details for other network services such as DNS and routing
  - ◆ */etc/route.conf* for Suse
  - ◆ */etc/sysconfig/static-routes* for RedHat
  - ◆ */etc/defaultrouter* for Solaris
- ❑ Startup scripts
  - ◆ You will want your network to be automatically configured whenever the computer is turned on

## Configuring the Device/Interface



- The loopback device is a special case
  - ◆ It is always present and is used to provide access to your own machine
  - ◆ Even if you don't have a network connection you will be able to use the loopback interface to test some of the basic networking services
  - ◆ The loopback interface always has the IP address 127.0.0.1
    - Whenever you use the IP address 127.0.0.1 you are connecting to your own computer

ITSC 1402/KRF

49

## ifconfig



- Network interfaces are configured using the *ifconfig* command
- The standard format for turning a device on is *ifconfig device\_name IP\_address netmask value up*
- For example
  - ifconfig eth0 138.77.37.26 netmask 255.255.255.0 up*
    - ◆ Configures the first ethernet address with the IP address of 138.77.37.26 and the netmask of 255.255.255.0
  - ifconfig lo 127.0.0.1*
    - ◆ Configures the loopback address appropriately

ITSC 1402/KRF

50

## Other *ifconfig* Parameters



- up and down
  - ◆ These parameters are used to take the device up and down (turn it on and off)
  - ◆ *ifconfig eth0 down* will disable the eth0 interface and will require an *ifconfig* command like the first example above to turn it back on
- -arp
  - ◆ Will turn on/off the address resolution protocol for the specified interface
- -pointtopoint *addr*
  - ◆ Used to specify the IP address (*addr*) of the computer at the far end of a point to point link

ITSC 1402/KRF

51

## Suse Configuration Files



- Suse uses the */etc/rc.configfile* to help in the configuration of network devices during the startup of the system
- The script which actually starts networking on a Suse Linux machine is */etc/init.d/network*

ITSC 1402/KRF

52

## */etc/rc.config* (Networking Only)



```
# Start loopback networking? ("yes" or "no")
# (this will be needed for all rpc services)
#
START_LOOPBACK="yes"
#
# Networking
#
# Number of network cards: "_0" for one, "_0_1_2_3" for four cards
#
NETCONFIG="_0"
#
# This variable contains all indices of active PCMCIA network devices
#
NETCONFIG_PCPCIA=""
#
```

ITSC 1402/KRF

53

```
# IP Addresses
#
IPADDR_0="172.16.1.9"
IPADDR_1=""
IPADDR_2=""
IPADDR_3=""
#
# Network device names (e.g."eth0")
#
NETDEV_0="eth0"
NETDEV_1=""
NETDEV_2=""
NETDEV_3=""
#
```

ITSC 1402/KRF

54



```
# Parameters for ifconfig, simply enter "bootp" or "dhcpcient"
# to use the respective service for configuration.
# Sample entry for ethernet:
# IFCONFIG_0="192.168.81.38 broadcast 192.168.81.63 netmask
255.255.255.224"
#
IFCONFIG_0="172.16.1.9 broadcast 172.16.1.255 netmask
255.255.255.0"
IFCONFIG_1=""
IFCONFIG_2=""
IFCONFIG_3=""
#
```

ITSC 1402/KRF

55



```
# Setup dummy network device for IPADDR_0? This is useful for non
# permanent network connections (e.g. SLIP, PPP). Some software
# needs a connection to FQHOSTNAME (e.g. plp). (yes, no)
#
SETUPDUMMYDEV="no"
#
# Do you want the "dynamic IP patch" to be enabled at bootup? (yes/no)
#
IP_DYNIP="no"
#
# Enable syn flood protection (see
# /usr/src/linux/Documentation/Configure.help)
# (yes/no)
#
IP_TCP_SYNCOOKIES="yes"
#
```

ITSC 1402/KRF

56



```
# Runtime-configurable parameter: forward IP packets.
# Is this host a router? (yes/no)
#
IP_FORWARD="no"
#
# SuSEconfig can do some checks and modifications in /etc/hosts.
# If this is not wanted, set the following variable to 'no' (yes/no).
#
CHECK_ETC_HOSTS="yes"
#
# If CHECK_ETC_HOSTS is set to yes, SuSEconfig sorts your /etc/hosts.
# But in some cases this may be unwanted. So here is a flag which tells
# SuSEconfig to "beautify" your /etc/hosts. (yes/no)
#
BEAUTIFY_ETC_HOSTS="yes"
```

ITSC 1402/KRF

57



```
# If SORT_PASSWD_BY_UID is set to yes, SuSEconfig sorts your
# /etc/passwd and /etc/group byuid/gid.
#
SORT_PASSWD_BY_UID="no"
#
# Hostname of the system (full name)
# If zero, and bootp is used above, bootp will also set the hostname
# (e.g. "riemann.suse.de" or "hugo.linuxde")
# Don't forget to edit your /etc/hosts appropriately.
#
FQHOSTNAME="pixel.krfrazier.com"
#
```

ITSC 1402/KRF

58



```
# When shutting down routing, all net connection can be closed (not
# useful in all cases). If CLOSE_CONNECTIONS is set to "true",
# /etc/init.d/route will scan /proc for active network connections and send
# term signals to the resp. processes.
#
CLOSE_CONNECTIONS="false"
```

ITSC 1402/KRF

59

## RedHat Configuration Files



- Redhat uses the `/etc/sysconfig/network` file and the contents of the `/etc/sysconfig/network-scripts` directory to help in the configuration of network devices during the startup of the system
- The script which actually starts networking on a RedHat Linux machine is `/etc/rc.d/init.d/network`

ITSC 1402/KRF

60

## /etc/sysconfig/network



```
NETWORKING=yes
HOSTNAME=unixserv.nlc.dcccd.edu
GATEWAY=144.162.122.1
```

ITSC 1402/KRF

61

## /etc/sysconfig/network-scripts



- Example *ifcfg-eth0* file

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=144.162.120.255
IPADDR=144.162.120.72
NETMASK=255.255.0.0
NETWORK=144.162.0.0
ONBOOT=yes
```

ITSC 1402/KRF

62

## Solaris Configuration Files



- Solaris uses */etc/hostname.<ifname>*, */etc/nodename*, */etc/inet/netmasks*, and */etc/inet/hosts*
- The script which actually starts networking on a Solaris machine is */etc/init.d/network*

ITSC 1402/KRF

63

## Configuring the Name Resolver



- Once the device/interface is configured you can start using the network
- However you'll only be able to use IP addresses
  - ◆ At this stage the networking system on your computer doesn't know how to resolve hostnames
- This is where the name resolver and its associated configuration files enter the picture

ITSC 1402/KRF

64

- The three files we'll be looking at are

- ◆ */etc/resolv.conf*
  - Specifies where your main domain name server is located
- ◆ */etc/host.conf* (Linux, Solaris uses *nsswitch.conf*)
  - Allows you to specify how the name resolver will operate
- ◆ */etc/hosts*
  - A local file which specifies the IP/hostname association between common or local computers

ITSC 1402/KRF

65

## */etc/resolv.conf*



- */etc/resolv.conf* is the configuration file for the resolver
- Its format is simple
  - ◆ It is a text file with one keyword per line
- There are three keywords typically used
  - ◆ *domain*
    - Specifies the local domain name
  - ◆ *search*
    - Specifies a list of alternate domain names to search
  - ◆ *nameserver*
    - This keyword, which may be used many times, specifies an IP address of a domain name server to query when resolving names

ITSC 1402/KRF

66



- ❑ An example `/etc/resolv.conf` might look like:
 

```
domain nlc.dcccd.edu
search dcccd.edu
nameserver 199.34.16.2
nameserver 144.162.10.230
nameserver 199.1.11.2
```
- ❑ This specifies that the default domain name to append to unqualified names is `nlc.dcccd.edu`
  - ◆ If the host is not found in that domain try the `dcccd.edu` domain directly
- ❑ Two nameservers entries are supplied, either of which may be called to resolve the name

## `/etc/host.conf`



- ❑ The `/etc/host.conf` file is where you configure some items that govern the behavior of the name resolver
  - ◆ The format of this file is described in detail in the 'resolver 5' man page
- ❑ In nearly all circumstances the following example will work for you:
  - ◆ `order hosts,bind`
  - ◆ `multi on`
- ❑ This tells the name resolver to check the `/etc/hosts` file before trying to query a nameserver and to return all valid addresses for a host found in the `/etc/hosts` file instead of just the first

## `/etc/hosts`



- ❑ The `/etc/hosts` file is where you put the name and IP address of local hosts
  - ◆ If you place a host in this file then you do not need to query the domain name server to get its IP Address
  - ◆ The disadvantage of doing this is that you must keep this file up to date yourself if the IP address for that host changes
- ❑ In a well managed system the only hostnames that usually appear in this file are an entry for the loopback interface and the local hosts name
 

```
# /etc/hosts
127.0.0.1 localhost loopback
192.168.0.1 this.host.name
```

- ❑ You may specify more than one host name per line as demonstrated by the first entry, which is a standard entry for the loopback interface.

## How Does Routing Work?



- ❑ Each host keeps a special list of routing rules, called a routing table
  - ◆ This table contains rows which typically contain at least three fields
    - A destination address
    - The name of the interface to which the packet is to be routed
    - Optionally the IP address of another machine which will carry the packet on its next step through the network
- ❑ In Linux you can see this table by using the following command:
  - ◆ `netstat -r`

- ❑ The routing process is fairly simple (at least in theory)

- ◆ An incoming packet is received
- ◆ The destination address is examined and compared with each entry in the table
- ◆ The entry that best matches that address is selected and the packet is forwarded to the specified interface
- ◆ If the gateway field is filled then the packet is forwarded to that host via the specified interface, otherwise the destination address is assumed to be on the network supported by the interface



- ❑ To manipulate this table the *route* command is used
  - ◆ This command takes command line arguments and converts them into kernel system calls that request the kernel to add, delete or modify entries in the routing table

## Configuring Routing



- ❑ Imagine you have an ethernet network
- ❑ You've been told it is a class-C network with an address of 192.168.1.0
- ❑ You've been supplied with an IP address of 192.168.1.10 for your use and have been told that 192.168.1.1 is a router connected to the Internet.
- ❑ The first step is to configure the interface as described earlier



- ❑ You would use a command like:
 

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```
- ❑ You now need to add an entry into the routing table to tell the kernel that packets for all hosts with addresses that match 192.168.1.\* should be sent to the Ethernet device
- ❑ You would use a command similar to:
 

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```



- ❑ Note the use of the '-net' argument to tell the route program that this entry is a network route. Your other choice here is a '-host' route which is a route that is specific to one IP address.
- ❑ This route will enable you to establish IP connections with all of the hosts on your Ethernet segment
- ❑ But what about all of the IP hosts that aren't on your Ethernet segment?
- ❑ It would be tough to have to add routes to every possible destination network, so the *default route* is a way to simplify this task



- ❑ The *default route* matches every possible destination, but poorly, so that if any other entry exists that matches the required address it will be used instead
  - ◆ The idea of the default route is simply to enable you to say "and everything else should go here"
- ❑ In the example you would use an entry like:
 

```
◆ # route add default gw 192.168.1.1 eth0
```
- ❑ The 'gw' argument tells the route command that the next argument is the IP address, or name, of a gateway or router machine which all packets matching this entry should be directed to for further routing



- ❑ So, your complete configuration would look like:
 

```
ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
route add default gw 192.168.1.1 eth0
```
- ❑ These steps are actually performed at boot time by the startup files on a properly configured Linux box

## Tools



- ❑ *nslookup*
- ❑ *netstat*
- ❑ *traceroute*

ITSC 1402/KRF

79

## *nslookup*



- ❑ The *nslookup* command is used to query a name server and is supplied as a debugging tool
  - ◆ It is generally used to determine if the name server is working correctly and for querying information from remote servers
- ❑ *nslookup* can be used from either the command line or interactively
  - ◆ Giving *nslookup* a hostname will result in it asking the current domain name server for the IP address of that machine

ITSC 1402/KRF

80

- ❑ **For example**
- ❑ *nslookup*
  
- ❑ *nslookup clyde.dcccd.edu*
  
- ❑ *nslookup www.ibm.com*



ITSC 1402/KRF

81

## *netstat*



- ❑ The *netstat* command is used to display the status of network connections to a UNIX machine
  - ◆ One of the functions it can be used for is to display the contents of the kernel routing table by using the *-r* switch
- ❑ For example
- ❑ *netstat -rn*

ITSC 1402/KRF

82

## *traceroute*



- ❑ Users on one machine can't connect to another machine or if they can any information transfer between the two machines is either slow or plagued by errors
- ❑ It's not only the machines at each end you have to check
  - ◆ If the machines are on different networks the data will flow through a number of gateways and routers
  - ◆ It might be one of the gateway machines that is causing the problem.

ITSC 1402/KRF

83

- ❑ *traceroute* provides a way of discovering the path taken by data as it goes from one machine to another
  - ◆ It can be used to identify where problems might be occurring
  - ◆ On the Internet that path may not always be the same



ITSC 1402/KRF

84

## traceroute Examples



- ❑ `traceroute clyde`
- ❑ `traceroute www.ibm.com`
- ❑ `traceroute circus.cqu.edu.au`

ITSC 1402/KRF

85

## What's a Daemon?



- ❑ A daemon (pronounced DEE-muhn) is a program that runs continuously and exists for the purpose of handling periodic service requests that a computer system expects to receive
- ❑ The daemon program forwards the requests to other programs (or processes) as appropriate
  - ◆ Each server of pages on the Web has an HTTPD or Hypertext Transfer Protocol daemon that continually waits for requests to come in from Web clients and their users
- ❑ In mythology, a daemon, according to Webster's, was "an attendant power or spirit"

ITSC 1402/KRF

86

- ❑ Daemon can be confused with demon, which has a different but similar meaning
- ❑ The New Hacker's Dictionary says that a daemon is a program that runs by itself directly under the operating system whereas a demon is part of a larger application program



ITSC 1402/KRF

87

## Super Daemons



- ❑ There are two types of daemons
  - ◆ Stand Alone
  - ◆ Super
- ❑ Stand Alone daemons are:
  - ◆ Self contained
  - ◆ Can be run from the command line
  - ◆ Are usually started from rc scripts
- ❑ Super daemons wait for requests and then call the appropriate process, which may also be a daemon
- ❑ `inetd` is an example of a super daemon

ITSC 1402/KRF

88

## Common Daemons



- ❑ `apmd` – power mgmt
- ❑ `atd` – at daemon
- ❑ `cron` – cron daemon
- ❑ `lpd` – line printer daemon
- ❑ `smbd` – smb daemon
- ❑ `klogd` – kernel log daemon
- ❑ `syslogd` – system logger
- ❑ `moused` – file system mount daemon
- ❑ `statd` – network status monitor
- ❑ `rquotad` – remote disk quota server
- ❑ `inetd` – internet daemon
- ❑ `ftpd` – ftp server
- ❑ `tcpd` – TCP server
- ❑ `sendmail` – mail server
- ❑ `routed` – network routing daemon
- ❑ `named` – name server
- ❑ `fingerd` – finger server
- ❑ `nfsd` – nfs daemon
- ❑ `timed` – time server
- ❑ `snmpd` – Simple Network Mgmt Protocol daemon

ITSC 1402/KRF

89

## Network Daemons



- ❑ The `/etc/services` file specifies which port a particular protocol will listen on
  - ◆ Examples:
    - SMTP – port 25
    - telnet – port 23
    - ssh – port 22
    - ftp – ports 20 and 21
- ❑ How do we know what program acts as the network daemon for a particular service?
- ❑ How does it get started?

ITSC 1402/KRF

90

## How Network Daemons Start



- There are two ways network daemons get started
  - ◆ Startup scripts at boot time
  - ◆ The *inetd* daemon
- *inetd* is a super daemon
- It listens to a number of ports and when activity is detected, *inetd* starts the appropriate network daemon for that port
  - ◆ Which daemon is associated with each port is defined in the */etc/inetd.conf* configuration file

ITSC 1402/KRF

91

## How Do I Know Where to Start a Daemon?



- Starting daemons with *inetd* is usually only done for services that only have a limited number of connections
  - ◆ For example, a busy web server should start the http daemon in a startup script
- The reason is overhead associated with *inetd*
  - ◆ Drawback to starting from startup scripts is that the daemon is always running, using RAM and CPU resources

ITSC 1402/KRF

92

## Starting a Service



- When a request is received on a port, *inetd* reads the */etc/services* file to determine what type of request is associated with the port
  - ◆ *inetd* searches */etc/services* for the first instance of the port number
- *inetd* then reads */etc/inetd.conf*, looking for the service name that was associated with the port number from */etc/services*
- When it finds a match, it then starts the associated process to handle the request

ITSC 1402/KRF

93

## */etc/inetd.conf* Fields



Field	Purpose
service-name	The service name, as listed in <i>/etc/services</i>
socket-type	Type of service; <i>stream</i> for TCP, <i>packet</i> for UDP, <i>raw</i> for direct IP
protocol	Transport protocol name as listed in <i>/etc/protocols</i>
flags	How <i>inetd</i> is to behave for this service
user	Username the daemon runs as
daemon-program	Full path to the daemon executable
args	Any command line arguments needed

ITSC 1402/KRF

94

## Sample of */etc/inetd.conf*



```
shell stream tcp nowait root /usr/sbin/tcpd in.rshd
login stream tcp nowait root /usr/sbin/tcpd in.rlogind
#exec stream tcp nowait root /usr/sbin/tcpd in.rexecd
#comsat dgram udp wait root /usr/sbin/tcpd in.comsat
talk dgram udp wait root /usr/sbin/tcpd in.talkd
ntalk dgram udp wait root /usr/sbin/tcpd in.ntalkd
#dtalk stream tcp wait nobody /usr/sbin/tcpd in.dtalkd
```

ITSC 1402/KRF

95

## Following an *ftp* Session



- While monitoring the computer's ports, a TCP request is received by *inetd* on port 21
- *inetd* checks */etc/services* looking for a service that matches a TCP request on port 21
  - ◆ *inetd* finds "ftp 21/tcp"
- *inetd* then searches */etc/inetd.conf* for an entry that matches ftp
  - ◆ *inetd* finds the entry:  
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd-l -a

ITSC 1402/KRF

96



- ❑ *inetd* then runs the *tcpd* program with *in.ftpd* as an argument
- ❑ *tcpd* logs the request via *syslogd* and does some additional checks
- ❑ When all is well, *tcpd* runs the *in.ftpd* server program and goes away

## tcpd Daemon



- ❑ The *tcpd* program can monitor incoming requests for *telnet*, *finger*, *ftp*, *exec*, *rsh*, *rlogin*, *tftp*, *talk* and other services that have a one-to-one mapping onto executable files
- ❑ Whenever a request for service arrives, the *inetd* daemon runs the *tcpd* program instead of the desired server
- ❑ *tcpd* logs the request and does some additional checks
- ❑ When all is well, *tcpd* runs the appropriate server program and goes away

## Super Server Considerations



- ❑ Since super servers don't keep the real server loaded all the time, you can usually update the real server without having to restart the super server
- ❑ Although super servers save memory by only loading servers when needed, they will provide slower response since they do have to load the real server
  - ◆ For things such as web servers, the delay is usually not tolerable
- ❑ Some servers, such as *sendmail* and *samba*, just don't work well from a super server

## Adding a New Server to inetd



- ❑ First, verify the service name exists in */etc/services*
  - ◆ If not, add it making sure to specify an unused port number
- ❑ Make sure the executable file for the server exists
- ❑ Add the new service to */etc/inetd.conf*
- ❑ Restart *inetd* by sending it a HANGUP signal so it will read the modified */etc/inetd.conf* file
  - ◆ `ps ax | grep "inetd"`
  - ◆ `337 ? S 0:00 inetd`
  - ◆ `kill -SIGHUP 337`
- ❑ Test your new server

## Extended Internet Services Daemon (xinetd)



- ❑ Designed as a replacement for *inetd*, providing enhanced security, support for logging, and user notifications
- ❑ *xinetd* also incorporates TCP security, providing TCP Wrappers functionality without the need for the *tcpd* daemon
- ❑ Red Hat Linux versions 7.0 and up use *xinetd* instead of *inetd*
- ❑ */etc/xinetd.conf* is the configuration file

## xinetd Features



- ❑ Access control:
  - ◆ Has built-in access control for stopping connections from bad guys, or for only allowing connections from good guys
- ❑ Can be compiled with built-in libwrap support
  - ◆ Uses `hosts.{allow|deny}`
  - ◆ More efficient than using *tcpd*
- ❑ TCP wrappers are good, but can only see one connection at a time
  - ◆ *xinetd* can limit the rate of incoming connections, number of incoming connections from specific hosts, or total number of connections for a service.

## Features



- ❑ Limit access to services based on access time of day
- ❑ You can have specific services bind to specific IPs
  - ◆ This lets you provide different services to internal clients than external clients
- ❑ Prevent denial of service attacks
  - ◆ With the access control capabilities of limiting the rate of incoming connections, xinetd can respond to "port bombs" in a reasonable fashion
  - ◆ If one host seems to be hogging your services, you can limit the number of simultaneous connections from a host

ITSC 1402/KRF

103

## Features



- ❑ You can place limits on the size of the log files it creates, so people can't fill your disk
- ❑ Extensive logging abilities
  - ◆ You can configure the *syslog* logging level for each service independently
  - ◆ If you don't want to use *syslog* logging, you can have each service log to a file, independent of any other service
  - ◆ It can log the start and stop times for the connection, so you can determine how long a client used your services
  - ◆ It can log extensive information about failed connection attempts

ITSC 1402/KRF

104

## Features



- ❑ The *redir* feature allows you to redirect a TCP stream to another host
  - ◆ This other host does not need to be an externally reachable machine
  - ◆ If you want to provide services on a NAT'd machine, run *xinetd* with the *redir* feature to redirect the service to a different host
- ❑ As of the *xinetd* 2.1.8.8pre\* series, *xinetd* supports IPv6

ITSC 1402/KRF

105

## Features



- ❑ You can print different banners to the client when they have a successful connection, when their connection attempt failed, and always regardless of connection status
  - ◆ This can help keep your users informed of changes, and why they may be having trouble accessing services

ITSC 1402/KRF

106

## *xinetd.conf*



- ❑ Entries in *xinetd.conf* define servers to be activated, along with options and security features, when a request for service is received
- ❑ Entries in *xinetd.conf* have the following format:

```
service service_name
{
  attribute assignment_operator value value ....
  ...
}
```

where assignment\_operator is =, +=, or -=

ITSC 1402/KRF

107

## Attributes



- ❑ Most attributes take a single value and use the = assignment\_operator
- ❑ Some can take a list of values
- ❑ You can assign values with =, but you can add or subtract values using += and -=
  - ◆ These are normally used to modify attributes that are assigned in the defaults block
- ❑ Certain attributes are required for a service, such as *socket\_type* and *wait*

ITSC 1402/KRF

108

## xinetd Attributes



Attribute	Description
id	Service identifier. By default, the service id is the same as the service name
type	Type of service; RPC, INTERNAL (provided by xinetd), or UNLISTED
flags	Possible flags include REUSE, INTERCEPT, NORETRY, IDONLY, NAMEINARGS, NODELAY, and DISABLE
disable	"yes" disables the service
socket_type	Stream, dgram, raw, seqpacket (for reliable sequential datagram transmission)

ITSC 1402/KRF

109

## xinetd Attributes



Attribute	Description
protocol	Specifies a protocol for the service. Protocol must exist in /etc/protocols. If attribute is not defined default service protocol is used
wait	"yes" – wait until the service completes "no" – continue to handle new requests
user	UID for the server process
group	GID for the server process
instances	Maximum number of simultaneous active server processes

ITSC 1402/KRF

110

## xinetd Attributes



Attribute	Description
nice	Server priority
server	Specifies the program name that provides the service
server_args	List of arguments to be passed to the service. This does not include the server name
only_from	A list of IP addresses that will be allowed access to the service. With no value, service is denied to all addresses
no_access	Lists hosts that are specifically denied service

ITSC 1402/KRF

111

## xinetd Attributes



Attribute	Description
access_times	Specifies the time interval during which the service is available
log_type	Where is the service's log to be sent, either SYSLOG or FILE
log_on_success	Specifies the info to be logged when a server starts and stops. Information that can be included includes PID, HOST (IP of remote host), USERID (the remote user), EXIT (server exit status), and DURATION

ITSC 1402/KRF

112

## xinetd Attributes



Attribute	Description
log_on_failure	Specifies the info to be logged when a server fails to start. Information that can be included includes HOST (IP of remote host), USERID (the remote user), ATTEMPT (logs a failed attempt), and RECORD (logs info from remote host to allow monitoring of access attempts)
rpc_version	RPC version to use for an RPC service
rpc_number	Specifies number for an UNLISTED RPC service

ITSC 1402/KRF

113

## xinetd Attributes



Attribute	Description
env	Defines environmental variables for the service
passenv	List of environmental variables from xinetd's environment to pass to the service
port	Service port
redirect	Allows a TCP service to be redirected to another host
bind	Allows service to be bound to a specific interface

ITSC 1402/KRF

114

## xinetd Attributes



Attribute	Description
interface	Synonym for <i>bind</i>
banner	Name of a file to display for a remote host when a connection is established
banner_success	Name of a file to display for a remote host when a connection is granted
banner_fail	Name of a file to display for a remote host when a connection is denied
groups	Allows access to groups the service has access to ("yes" or "no")

ITSC 1402/KRF

115

## xinetd Attributes



Attribute	Description
enabled	List of services to enable
include	Specifies a file to be included as part of the configuration file
includedir	Specifies a directory. Every file in that directory will be sequentially read and combined to form the <i>xinetd</i> configuration

- These are the more common attributes. See the *xinetd.conf* man page for a complete listing

ITSC 1402/KRF

116

## Aurora *xinetd.conf*



```
## Simple configuration file for xinetd
## Some defaults, and include /etc/xinetd.d/
defaults
{
  instances          = 60
  log_type           = SYSLOG authpriv
  log_on_success     = HOST PID
  log_on_failure     = HOST
  cps                = 25 30
}
includedir /etc/xinetd.d
```

ITSC 1402/KRF

117

## *hosts.allow* and *hosts.deny*



- *xinetd* also uses the *hosts.allow* and *hosts.deny* files for limiting access
- Wildcards
  - ALL Matches all hosts
  - LOCAL Matches any host specified with just a host name. Matches hosts on local domain
  - UNKNOWN Matches any user or host whose name or address is unknown
  - KNOWN Matches any known host or user
  - PARANOID Matches any host whose name doesn't match its IP address
  - EXCEPT Exceptions *list1* EXCEPT *list2*

ITSC 1402/KRF

118

## Example Service Config File



```
service ftp
{
  socket_type = stream
  wait = no
  user = root
  protocol = ftp
  server = /usr/sbin/in.ftpd
  server_args = -l -a
  disable = yes
}
```

ITSC 1402/KRF

119