TCP/IP Overview

A protocol that allows two computers to talk to each other

May be used with network card, modem, serial port, parallel port or wireless port.

Server: any machine that handles requests.
Provides files, news, mail, dial-up ports.

Client: any machine that makes requests.

Local: on your own machine
Remote: on some other other machine

Gateway: connects you to the rest of the network
Firewall: protects you from the rest of the network

Services supported using TCP/IP include:
mail, network file system, telnet, news,
network information system (yp), www

IP Names and Addresses

Network addresses are officially assigned.
Use the one defined by your service provider or network administrator.

32-bit (4 octet) number
usually given in dot notation:
`134.139.248.22`
each octet is given in decimal (0..255)

Some addresses have special meanings and uses.

Network names are officially assigned.
Coordinate names with your network administrator
`lab18.net.cecs.csulb.edu`

User on the network
Specify which user and which machine
format: `user@machine`
example: `sam@lab99.net.cecs.csulb.edu`

Internet Numbering

An internet number has two parts,
a subnet part and a host part

1) Use the subnet part of the internet number to get the packet to the correct internet subnet.

2) On the local subnet use the host part of the internet number to get the packet to the correct machine.
(Actually this step uses the whole number.)

Rule: IP numbers of all interfaces of all hosts on the same subnet must have the same subnet part.

Packets always move from a host to another host on the same subnet.

At a host, a packet can enter the host on one subnet and leave on another.
(Provided the host is connected to two subnets and the packet is to be forwarded.)

Definition: a host that is connected to more than one subnet is called a *gateway*.

Each card in a gateway has a different Internet number.

Determining if a Host is on the Local Subnet

Needed: a way to distinguish the subnet part from the host part.
Solution: netmasks

Examples:

1) `0xffffff00` first 24 bits: subnet, last 8 bits: host
2) `0xffffffc0` first 26 bits: subnet, last 6 bits: host
3) `0xffff0000` first 16 bits: subnet, last 16 bits: host

Two host parts are reserved:
all 1's (8,6,16 bits)–broadcast to all on subnet
all 0's–refers to the subnet

Examining a Network Interface

Each interface must be configured with a unique IP number.

`ifconfig` – Examine the configuration of the network interfaces.

Reports: ethernet numbers, internet numbers, netmasks, hardware configuration

Linux Reports: interface statistics.

The ARP Table

We use IP numbers, the hardware needs ethernet numbers.

We need to translate.
This is called the Address Resolution Problem.
Solved by the Address Resolution Protocol.

IP number – hardware number mapping table

`arp -a -n`: display the table

```
134.139.248.65 ether 00:00:C0:9E:DD:5A C * eth0
```

IP address, interface type, his hardware adddress, flag, mask, which ethernet card

Arp entries are generated automatically

```
arp -s 134.139.248.61 01:01:01:01:01:01
```

force an entry into the arp table
(only if the other machine is brain-dead)

The administrator does not need to maintain this table, it is useful for seeing what else is on the subnet.

The Routing Table

Needed: a way to specify forwarding.
Solution: routing tables

partial route table for cheetah

destination	gateway	Genmask
134.139.248.32	134.139.248.18	255.255.255.224
0.0.0.0	134.139.248.1	0.0.0.0

For machines on `134.139.248.32` subnet send to machine `134.139.248.18`

For all other machines send to machine `134.139.248.1`

`netstat`–examine network information

`route`–examine routing information (uses `netstat`)

Although `ifconfig` configures the subnets you are directly attached to; the route contains entries for these subnets. Having everything put into a single table speeds the lookups.

Configuring for Networking

Internet Protocol

Plug and Play: On modern cards Linux will find the card if the drivers for that card are loaded.

If you have more than one card, you must be careful.

The drivers are probed in the order they are loaded into the kernel. So certain types of cards will be found first and labeled `eth0`. (On the first boot).

If cards are identical, they are loaded in their order on the PCI bus.

On the first boot udev records the hardware addresses and which eth numbers they were assigned.
These will be assign the same eth numbers on subsequent boots, even if they are removed from the machine.

Consequence: need to erase the udev (`/etc/udev/rules.d`) when you swap ethernet cards.

Typical action: load all cards, assign IPs and ping to see which ones have the blinking lights.

Now reassign IPs (or switch around the subnet).

Name and Number Lookup

Two files have principle control over looking up names and numbers.

`resolv.conf`:
Used only in the look of internet name/number information.

Specifies the name server to use for name and number lookups.
It also specifies the search order for names.

`nsswitch.conf`:
Used for internet name/number lookup
as well as for other lookups
(passwords, group, netmasks, boot params)

Specifies what order the lookups are to be done in.
Files, NIS, DNS are used in the order specified by this file.

Configuring An Interface

`ifconfig` – you must specify an interface and its new configuration.

The following is one line:

```
ifconfig eth0 134.139.248.2
                broadcast 134.139.248.3
                netmask 255.255.255.252
```

The internet number, netmask and broadcast address are set.
Usually the broadcast address is omitted, because it is the subnet number with all 1's in the host part. You only need to include it if there is an unsual broadcast address.

Adding A Route

`route` – specify a subnet and a gateway to use to get there.

The following is one line:

```
route add -net 134.139.248.32
         netmask 255.255.255.224
         gw 134.139.248.18
         metric 1
```

The metric, if omitted, is assumed to be 1.

Configuring TCP/IP

1) Build networking into your kernel

```
cd /usr/src/linux
make config
```

1b) Select a kernel with networking in it

2) Set up `/etc/hosts` (optional)

```
127.0.0.1 localhost
134.139.248.19 puma.net.cecs.csulb.edu puma
```

assigned number, full name, nick-name

`/etc/networks` is optional

The following may be done by hand, but
are usual done at boot

`/etc/rc.d/rc.inet1`

`/etc/rc.d/rc.inet2` – starts the daemons that provide various network services.

`/etc/rc.d/rc.inet1.config` – data used by `rc.inet1`

3) Set up you host name

`hostname [`*your name*`]`

```
hostname puma
```

4) configure the localhost (loopback) interface

```
ifconfig lo 127.0.0.1
```

5) configure the network interface

```
ifconfig [parameters]
ifconfig eth0 134.139.248.19 netmask 255.255.255.248
```

6) Set up the route to the gateway

```
route [parameters]
route add default gw 134.139.248.17
```

7) Add other routes if necessary.

```
route add -net 134.139.248.32 gw 134.139.248.18
                 netmask 255.255.255.224
```

Hostnames and Internet Numbers

Principle: Each machine has a name, it may have several internet numbers.

Needed: name-number and number-name lookup.

`/etc/hosts` file – list of names and number

```
127.0.0.1    localhost
134.139.248.18 jaguar.net.cecs.csulb.edu jaguar
```

Needed a way to handle millions of names.

Domain Name Service. Principle–contact a network database and have them do the the names-number and number-name lookups.

Needed: the IP number of a DNS (network database) server.

You have several ways to a lookups

Needed: which order to apply the lookups.

8) Point name resolution at a DNS server

`/etc/resolv.conf` – (a file)

```
domain net.cecs.csulb.edu
nameserver 134.139.248.17
search net.cecs.csulb.edu cecs.csulb.edu
```

9) Set the resolver parameters

`/etc/nsswitch.conf`

```
hosts: files nis dns
passwd: files nis
```

Testing

`ping 127.0.0.1`
failure ⇒ loopback not setup

`ping 134.139.248.99` – (self)
failure ⇒ interface not configured

`ping 134.139.248.65` – (gateway)
failure ⇒ incorrect netmask

`ping 134.139.248.17` – (some distant machine)
failure ⇒ bad netmask or missing gateway route

`ping puma` – (ourselves)
failure ⇒ bad `nsswitch.conf` or `/etc/hosts`

`ping prep.ai.mit.edu` – (some distant machine)
failure ⇒ bad `resolv.conf` or `nsswitch.conf`

Network Services
inetd

The `inetd` super-server is responsible for providing/starting many standard services such as: ftp telnet time finger rlogin

`/etc/inetd.conf` – controls what inetd does

```
ftp stream tcp nowait root /usr/sbin/tcpd proftp
```

service name (ftp): should be in `services`
socket type/protocol: should be `dgram udp` or `stream tcp`
fork: `nowait` fork a process to handle this request `wait` handle the request yourself
UID: run service with this priviledge (root uucp)
base server: tcpd–called tcp wrappers this program does a security check, logs the access and starts the service
real server (name of executable): `proftp`
options: allows - options after real server

Commenting out the entry turns off the service.

To make inetd re-read it's configuration file use
`kill -HUP`

TCP Wrappers

Two control files.

`/etc/hosts.allow`: permission is granted to these.
`/etc/hosts.deny`: permission is denied to these.
Not found: permission is granted

Example: `/etc/hosts.allow`

```
in.fingerd : aardvark.cecs.csulb.edu
.net.cecs.csulb.edu
in.tftpd : 134.139.248.64/255.255.255.224
wu.ftpd : 134.139.0.0/255.255.0.0
in.telnetd : ALL
```

Subnet/netmask format is allowed, starting . is a wild card

The key word `ALL` is recognized.

Example: `/etc/hosts.deny`

```
ALL : ALL
```

Everything not explicitly allowed in `hosts.allow` is denied.

On the left, `ALL` refers to all inetd services.

Services List

`/etc/services` — Provides a list of names and numbers for common services.

```
telnet  23/tcp
```

Port 23 is assigned to the telnet server

Note: inetd uses this file to find the port numbers, if the service is not in here, you can't name the service in `inetd.conf`. Usually: don't change this file or `/etc/protocols` which lists the allowed IP protocols.

The r Commands

`rlogin`: login into another machine.
`rsh`: run a command on another machine
`rcp`: copy a file to/from another machine

`rlogin lab85 -l sam` —login to lab85 as sam

Authorization: requires a password or preauthorization.

`~sam/.rhosts` — user authorization granted.

```
lab88.net.cecs.csulb.edu joe
```

joe from lab88 doesn't need a password to login as sam

`/etc/hosts.equiv` — machine authorization granted for same user name.

```
lab89.net.cecs.csulb.edu
```

joe from lab89 doesn't need a password to login as joe

`rsh`, `rcp` — require pre-authorization

Remote Procedure Calls

You can get information about remote procedure calls

Command: `rpcinfo`

Options:

`-p [host]` – probe the host to see what calls are available.

`-u host program` – see if a particular RPC is running on the specified machine (UDP)
`-t` same only TCP

`-b program version` – broadcast to see which machines on the subnet are running a program.

`-d program version` – deregister (remove) a remote program from your system

Example:

```
rpcinfo -p cheetah.cecs.csulb.edu
```

List the remote programs that cheetah provides.

ssh Capabilities

The ssh suite provides both login, copy and tunneling facilities.
All facilities use the `sshd` server on the remote machine.

Command: `ssh`

A machine may be specified by either it's name of IP number.

A user to login as on the remote machine may be specified.
If no user is specified the name of the user originating the connection is used.

Sample: `ssh bob@lab99`

Command: `scp`

In addition to specifying a machine and user a source and destination file or path are specified.

Sample: `scp myfile bob@lab99:Hisfile`

ssh Security Features

Each host running `sshd` has a private and public key. (Actually it has 3 such pairs)

These are stored in the `/etc/ssh` directory.

Also in that directory are the configuration files for the ssh clients and server.

If you use an ssh client a `.ssh` directory is created inside your home directory. Public keys of known hosts are stored here.

If you attempt to connect to a known host and it's private key does not match the public key you have stored for it, the connection is broken. (configurable)

If you attempt to connect to a host for which you do not have a public key you will be asked if you want to proceed. (configurable)

For each connection a session key is created and the network data is encrypted with that key.

Client Configuration

Options on the client side (e.g. ssh) can be set in three ways:

1) Command-line (overrides all other settings)

2) User configuration file (`˜/.ssh/config`) (overrides system-wide settings)

3) System-wide configuration file (`/etc/ssh/ssh_config`)

Thus the administrator can set default values, but the user can override anything (standard Unix philosophy).

Configuration file format: *keyword arguements*

Some options:

`CheckHostIP`: *yes* tells ssh to lookup the host key in the known hosts file.

`Cipher`: tell ssh what encryption method to use.

`Port`: use a port other than 22.

`StrictHostKeyChecking`: how to handle the known hosts file. Warning, if your hosts are frequently reloaded, this option is a pain.

Server Configuration

Options on ther server side are set in
`/etc/ssh/sshd_config`
(The user cannot tell the server how to behave).

`AllowUsers`: can be used to restrict which users can login and can also restrict the machines they can login from.

`ListenAddress`: Only accept client requests on the interface with the specified IP address. This line can be repeated to specify several IP addresses.

Existing config files:
Commented out keywords with default values

Access Without a Password

In some cases you may need to set up access without a password.

The steps:

1) setup a public/private key for your account

2) send the public key to the other account

3) if your public key is installed in the other account's *authorized keys* list then your account can ssh to that account without a password.
This access right applies to the other ssh commands as well such as `scp`.

Note: updates are distributed using this method in the main labs.

Access Details

1) `ssh-keygen -t rsa`
Generates a public/private key pair,
Places them in your `.ssh` directory,
`ir_rsa` (private key), `id_rsa.pub` (public key)
It also creates the directory if necessary.
If you are going to use this for automatic login, don't use a passphrase.

2) scp the public key to the other account,
You will need the password
Call it something safe like newkey

3) ssh to the other account
create the `.ssh` for that account
make sure the directory is mod 700 (`rwx------`)
Add the key to that accounts list of authorized keys
A safe way to do this is to use `cat` to append the key:

```
cat newkey >> ~/.ssh/authorized_keys
```

Both scp and ssh should now work without a password.