Focus: Basics and Standard Tools

- Solving network problems depends a lot on your understanding
- Simple tools can tell you what you need to know
- Example: `ping` is incredibly useful!

Troubleshooting

- Avoid it by:
  - redundancy
  - documentation
  - training
- Try quick fixes first
  - simple problems often have big effects:
    - is the power on?
    - is the network cable plugged into the right socket? Is LED flashing?
    - has anything changed recently?
- Change only one thing at a time
  - test thoroughly after the change
- Be familiar with the system
  - maintain documentation
- Use tools that automatically document what you are doing, e.g.,
  - `sysreport` in Red Hat Linux
  - descriptions of your normal behaviour

Documentation

- Maintain an inventory of equipment and software
  - a list mapping MAC addresses to machines can be very helpful
- Maintain a change log for each major system, recording:
  - each significant change
  - each problem with the system
  - each entry dated, with name of person who made the entry
- Two categories of documentation:
  - Configuration information
    - describes the system
  - use system tools to obtain a snapshot, e.g., `sysreport` in Red Hat Linux
    - procedures
    - How to do things
  - use tools that automatically document what you are doing
    - `script` - better
    - `plod` from `http://bullwinkle.deer-run.com/~hal/plod/`
      - lets you record a worksheet easily
    - `Perl`, so fine on any platform

Problem Solving

1. Define problem
2. Gather facts
3. Consider possibilities based on facts
4. Create action plan
5. Implement action plan
6. Observe results
   - YES, Symptoms stop
   - NO, symptoms persist
7. Document process

General Troubleshooting

- Problem is reported by a person or by software
  - Often involves communicating with others
  - Somewhat like gathering requirements in software design
  - An iterative process
- Possible questions to ask:
  - What does not work?
  - What does work?
  - Are the things that do and do not work related?
  - Has the thing that does not work ever worked?
  - When the problem was first noticed?
  - What has changed since the last time it did work?
  - Did anything unusual happen since the last time it worked?
- When exactly does the problem occur?
  - Can the problem be reproduced and if so, how can it?
Gather the Facts

You probably need to find out more about the problem from other sources, including:
- Asking other people: affected users, network administrators, managers, and other key people
- Network management systems, such as Nagios
  [http://nagios.org/](http://nagios.org/)
- Tools such as Ethereal, tcpdump, ntop
- Server log files
- Documentation about your servers and network created by local staff
- Documentation about software and hardware that are provided by the vendors

Consider Possibilities based on Facts

Using the information you have gathered, try to eliminate some potential problems from your list.

Create an Action Plan

Start with the most likely... and those that are easiest to test
- Plan needs to be methodical
- Plan to change only one thing at a time
- You can then understand the cause of the problem
- Aim to understand the problem so you can learn from it, solve (or prevent) similar problems in the future
- Aim higher than just removing the symptoms!

Implement Action Plan

- Perform each step carefully
- Test to see if symptoms go away

Observe Results

Gather results as you change each variable
- Use same techniques you used in slide 11:
  - Check with the key people
  - Check with your tools

If Solved: Document Solution

- Record the problem and its resolution in the documentation you maintain for your company.
- Ensure others in your team can benefit from the insight you have gained

Otherwise, Modify Action Plan

- Go back to the steps in slide 13:
  - Modify your action plan, selecting the next most likely action from your list
  - You may have discovered more information in your investigation, so this can help you focus on likely causes.
  - If you have exhausted all the items on your list, and cannot think of what else to do:
    - Get help from the vendor
    - Get help from mailing lists
    - Discuss the problem with your network of colleagues (e.g., the people who are now studying with you, but who move on to work in a similar field!)
    - You could even track me down and ask me! Quite a few of my ex-students do.

TCP/IP
**IP Header**

- **Version** — this is a 4-bit IP header length field that indicates the version of IP currently used. The current version of IP is 4 (IPv4) but IPv6 is already being implemented experimentally and will be supported on future versions of the IOS.
- **IP Header Length (IHL)** — this indicates the datagram header length in 32-bit words.
- **Type of Service (ToS)** — ToS specifies how a particular upper-layer protocol would like the current datagram to be handled. Datagrams can be assigned various levels of importance with this field.
- **Total length** — this specifies the length of the entire IP packet, including data and header, in bytes.
- **Identification** — this field contains an integer that identifies the current datagram. This field is used to help piece together datagram fragments.

**TCP Header**

- **Source Port**
- **Destination Port**
- **Sequence Number**
- **Acknowledgement Number**
- **Header length**
- **Reserved**
- **Window**
- **Checksum**
- **Urgent Pointer**
- **Options (0 to 40 bytes)**
- **Padding**
- **Your data starts here**

**UDP Header**

- **Source Port**
- **Destination Port**
- **Length**
- **Checksum**
- **Your data starts here**

**IP Header**

- **Options (0 to 40 bytes)**
- **Padding**
- **Your data starts here**
Troubleshooting TCP/IP

Step 1 First, determine whether your local host is properly configured (for instance, correct subnet mask and default gateway configuration).

Step 2 Next, use the `ping` or `traceroute` commands to determine whether the routers through which you must communicate can respond. Start with the most local router and progressively ping outwards through the Internet or use `traceroute`.

Step 3 If you cannot get through a particular node, examine the node configuration and use the various show commands to determine the state of the router (these include show ip route, show ip arp, show running-configuration, and so on.)

Step 4 If you can get to all the routers in the path, check the host configuration at the remote host (or get someone’s help to do so and check its configuration).

Checking (and Setting) Host Configuration

- Solving Boot problems: §32, §33
- Determine IP address, netmask, broadcast address: §32
- Determine correct MAC -> IP address mapping: §33, §34
- Examine routing table: §47
- Examine access controls: §23
- Examine web proxy settings: check web browser
- Examine DNS resolver settings: §23
- Determine services provided: §10, §11
- Determine CPU, memory load conditions (is the server overloaded?) §52

Boot problems: Linux

- Use `grub` to interactively boot the computer (see my extensive grub handout: [http://steve.cis.unt.edu/~grub/grub.pdf](http://steve.cis.unt.edu/~grub/grub.pdf))
- Verify that `/etc/fstab` mounts the correct filesystems
- Use a rescue disk such as Knoppix or the Red Hat installation CDROM.
- This gives you full access to the system and repairing boot problems is pretty straightforward.

Boot problems: Windows

- Use the installation Windows CD to enter the (extremely limited) system repair mode. I believe this is called the recovery console.
- Use the Linux floppy bootdisk at [http://home.eunet.no/~pnordahl/ntpasswd/](http://home.eunet.no/~pnordahl/ntpasswd/) to replace the Administrator password
  - Gives full access to the NTFS file system.
  - Not as good with Windows as Knoppix is with Linux, but better than another reinstall.
  - Takes some time to build.
- Henry Leung (in A204d) has built some.

Determine Addresses

Linux: On Linux, these commands all show the IP address, MAC address, netmask and broadcast address for all (or the specified) interface. The commands `ip` and `ifconfig` are in the directory `/sbin`; `netstat` is in `/bin`.

```
$ ip addr
$ ip addr show eth0
$ ifconfig
$ ifconfig eth0
$ netstat -i
```

Windows:

```
C:\> ipconfig /all
```

Cisco IOS: these are both privileged commands, as shown by the prompt:

```
Router# show running-config
```
MAC ↔ IP mapping — 1

Linux:

```bash
$ arp -a
$ ip neig show
```

The lifetime of the ARP cache entries is settable in `/proc/sys/net/ipv4/neigh/<interface>/gc_stale_time` and is normally 60 seconds.

Cisco IOS:

```
Router# show ip arp
```

Access controls can block access mysteriously unless you think to check for it.

Linux: There are two main things to check. The `iptables` command is in the `/sbin` directory.

```bash
$ iptables -L -n
```

Note that Linux and many other POSIX systems implement the tcpwrappers access control in `/etc/hosts.allow` and `/etc/hosts.deny`. See `man hosts.allow` and `man hosts.deny`.

Windows:

```
C:\> route print
C:\> netstat -nr
```

Recent versions of Windows provide the program:

```
C:\> ipconfig /all
```

Windows: Check network connections with `ipconfig /all`.

Cisco IOS:

```
Router# show ip access-list
```

Checking services provided

Linux: There are four main ways to check:

- Verify the processes with `ps` (see §41)
- Verify the services that are configured to start when the system boots:
  ```bash
  $ chkconfig --list | grep on
  ```
- Check that the service is listening on the network interface:
  ```bash
  $ netstat -tua
  ```
- The `lsof` program can be helpful in diagnosing problems with network services. See §32.

Windows: Check network connections with `ipconfig /all`.

Using `top` to see Resource Hogs

The program `top` shows:

- `load average` (the average number of processes that are ready to run, but for which CPU is available)
- `a load average of 4 or more is “quite high”`
- `processes that use the most resources`

Using `ps` to See If Server Running

- `Is the network service running on the server?`
- `Is the web server running?`
  ```bash
  $ ps aux | grep httpd
  ```
- `Is the DHCP server running?`
  ```bash
  $ ps aux | grep dhcpd
  ```
- `Is the directory server running?`
  ```bash
  $ ps aux | grep slapd
  ```
- Windows: use the task manager

DNS resolver settings

Linux: The configuration for the resolver is `/etc/resolv.conf`. This determines what name servers the system will ask. It also tells what domain will be appended to a hostname.

The `/etc/hosts` file is usually the first way hostname ↔ IP address mappings are made, but this can be changed in `/etc/nsswitch.conf`. To ask the operating system for what it can see there, do:

```bash
$ getent hosts
```

Linux provides three tools for troubleshooting DNS and DNS servers: `dig`, `host` and `nslookup`.

Windows: See the output of

```
C:\> ipconfig /all
```

Recent versions of Windows provide the program:

```
C:\> ipconfig /all
```

for the names of the DNS server the resolver will use.

Routing Table

Linux: The commands `ip` and `route` are in `/sbin`, the command `netstat` is in `/bin`.

```bash
$ ip route
$ route -n
$ netstat -nr
```

Windows:

```
C:\> route print
C:\> netstat -nr
```

Cisco IOS:

```
Router# show ip route
```

Using `netstat` to See If Server Running

- `Is the network service running on the server?`
- `Is the web server running?`
  ```bash
  $ ps aux | grep httpd
  ```
- `Is the DHCP server running?`
  ```bash
  $ ps aux | grep dhcpd
  ```
- `Is the directory server running?`
  ```bash
  $ ps aux | grep slapd
  ```
- Windows: use the task manager

Windows:

```
C:\> arp -a
C:\> ip neigh show
```

You may wish to clear the ARP cache on a Windows machine with:

```
C:\> arp -d (IP address)
```

or clear the entire ARP cache with:

```
C:\> arp -d *
```

since the Windows ARP cache lives 10 minutes by default, a rather (too?) long time.

It can be changed by two registry entries under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`.

Access Controls

Access controls can block access mysteriously unless you think to check for it.

Linux: There are two main things to check. The `iptables` command is in the `/sbin` directory.

```bash
$ iptables -L -n
```

Note that Linux and many other POSIX systems implement the tcpwrappers access control in `/etc/hosts.allow` and `/etc/hosts.deny`. See `man hosts.allow` and `man hosts.deny`.

Windows:

```
C:\> route print
C:\> netstat -nr
```

Cisco IOS:

```
Router# show ip access-list
```

Checking services provided

Linux: There are four main ways to check:

- Verify the processes with `ps` (see §41)
- Verify the services that are configured to start when the system boots:
  ```bash
  $ chkconfig --list | grep on
  ```
- Check that the service is listening on the network interface:
  ```bash
  $ netstat -tua
  ```
- The `lsof` program can be helpful in diagnosing problems with network services. See §32.

Windows: Check network connections with `ipconfig /all`.

Using `top` to see Resource Hogs

The program `top` shows:

- `load average` (the average number of processes that are ready to run, but for which CPU is available)
- `a load average of 4 or more is “quite high”`
- `processes that use the most resources`
**netstat -tua: See Network Connections**

- netstat -tua shows all network connections, including those listening.
- sudo netstat -tup shows all network connections, including those listening, and the processes responsible.
- netstat -tu shows all network listeners.
- netstat -t shows only TCP connections that are established.
- netstat -i is like ifconfig, shows info and stats about each interface.
- netstat -nr shows the routing table, like route -n.
- Windows provides netstat also.

**lsof: List Open Files**

- An amazingly useful tool.
- Available for almost any Unix system.
- lsof -i shows output to Internet and X.25 files, but won’t show connections that have terminated.
- lsof -i@nicku.org will show only connections to that machine.
- Can monitor progress of an FTP transfer, many, many other applications.
- See manpage, FAQ and quick start guide.
- Apparently, no equivalent tool available on Windows.

**ifconfig**

- ifconfig eth0 — show stats on network interface eth0.
- sudo ifconfig lo 127.0.0.1 — configure the loopback interface, start it up.
- sudo ifconfig eth0 172.19.233.5 netmask 255.255.255.0 — configure eth0 with IP address 172.19.233.5/24.
- ifconfig — show all configured network interfaces.
- ifconfig -a — show all interfaces, including those not configured yet.

**route**

- route -n — print routing table.
- route add 127.0.0.1 — add a route to localhost; should have been done automatically when created device with ifconfig.
- route add -net 172.19.233.0 — add a route to the eth0 configured on previous slide.
- route add 172.19.64.0 gw 172.19.233.254 — add a static route to network 172.19.64.0 through router 172.19.233.254.
- route add default gw 172.19.233.253 — add a default route to 172.19.233.253 through eth0.

**Connectivity Testing: Cabling**

- Label cables clearly at each end.
- Cable testers.
  - ensure wired correctly, check:
  - attenuation.
  - length — is it too long?
  - 100BaseT: less than 100m.
  - Is the activity light on the interface blinking?

**Ping**

- Most useful check of connectivity.
- Universal.
- If ping hostname, includes a rough check of DNS.
- Sends an ICMP (Internet Control Message Protocol) ECHO_REQUEST.
- Waits for an ICMP ECHO_REPLY.
- Most pings can display round trip time.
- Most pings can allow setting size of packet.
- Can use to make a crude measurement of throughput—see §61.

**What ping Result is Good, Bad?**

- A steady stream of consistent replies indicates probably okay.
- Usually first reply takes longer due to ARP lookups at each router.
- After that, ARP results are cached.
- ICMP error messages can help understand results:
  - Destination Network Unreachable indicates the host doing ping cannot reach the network.
  - Destination Host Unreachable may come from routers further away.
**How to Use ping?**

- Ensure local host networking is enabled first: ping localhost, local IP address
- ping a known host on local network
- ping local and remote interfaces on router
- ping by IP as well as by hostname if hostname ping fails
- confirm DNS with `dig` (or `nslookup`) — see slide 51B
- Ping from more than one host

**fping: flood ping**

- Designed to test a large number of hosts
- more efficient than ping
- take care not to flood too much!
- RPMs are available, I built one (a long time ago) and put it on ictlab under ∼:ftp/pub/redhat/contrib

**hping2: ping anything with anything**

- able to send custom TCP/IP packets and display target replies like ping program does with ICMP replies.
- Can install with $ `yum -y install hping2` on Fedora Core 1.
- See [http://www.hping.org/](http://www.hping.org/)

**arping: uses ARP requests**

- Limited to local network
- Can work with MAC or IP addresses
- use to probe for ARP entries in router (very useful!)
- packet filtering can block ICMP pings, but won’t block ARP requests

**Path Discovery: traceroute**

- Sends UDP packets
  (Microsoft `tracert` sends ICMP packets)
- increments Time to Live (TTL) in IP packet header
- Sends three packets at each TTL
- records round trip time for each
- increases TTL until enough to reach destination

**traceroute: How it Works**

- As IP packets pass through each router, TTL in IP header is decremented
- Packet is discarded when TTL decrements to 0
- ROUTER sends ICMP `TIME_EXCEEDED` message back to traceroute host
- When UDP packet reaches destination, gets ICMP `PORT_UNREACHABLE`, since uses an unused high UDP port

**traceroute Limitations**

- Each router has a number of IP addresses
- but `traceroute` only shows the one it used
- get different addresses when run `traceroute` from other end
- sometimes route is asymmetric
- router may be configured to not send ICMP `TIME_EXCEEDED` messages
- get stars: * instead of round-trip time in `traceroute` output

**Performance Measurements: delay**

- Three sources of delay:
  - `transmission delay` — time to put signal onto cable or media
  - depends on transmission rate and size of frame
  - `propagation delay` — time for signal to travel across the media
  - determined by type of media and distance
  - `queuing delay` — time spent waiting for retransmission in a router
Is Bandwidth == Throughput?

- **bandwidth** — the difference between the upper frequency and the lower frequency that a channel can carry
- **throughput** — amount of data that can be sent over link in given time
- is not the same as bandwidth, which really has no direct meaning with digital information
- bandwidth is related to throughput by the Shannon-Hartley Theorem; throughput \( \propto \) bandwidth if signal to noise ratio is fixed:

\[
C_{\text{max}} = B \log_2 \left(1 + \frac{S}{N}\right) \text{ bits/sec}
\]

where \( C_{\text{max}} = \) maximum channel capacity, \( B = \) bandwidth in Hz

ping Roughly Estimating Throughput

Example: measuring throughput between this machine and one remote machine.
ping with packet size = 100 bytes, round-trip time = 30ms
ping with packet size = 1100 bytes, round-trip time = 60ms
So takes 30ms extra (15ms one way) to send additional 1000 bytes, or 8000 bits
Throughput is roughly 8000 bits per 15ms, or about 530,000 bits per second
A very crude measurement:
- no account for other traffic, treats all links on path, there and back, as one.
- Routers sometimes send packets onwards with much higher priority than with which they answer pings. See slide §68.

Throughput: ping Two Remote Hosts

Measure throughput between two remote hosts: may use tools like ping
ping two locations with two packet sizes (4 pings altogether, minimum)
Many ping programs calculate average ping time:
- better to make a number of pings, use the average ping time.
- First ping time may be longer due to the time to get an answer to the arp request
- May be better to ping once, then start pinging again, and use the average ping time.
Example:

<table>
<thead>
<tr>
<th>Address</th>
<th>RTT 100 bytes</th>
<th>RTT 1100 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>205.153.61.1</td>
<td>1.380 ms</td>
<td>5.805 ms</td>
</tr>
<tr>
<td>205.153.60.2</td>
<td>4.985 ms</td>
<td>12.623 ms</td>
</tr>
<tr>
<td>165.166.36.17</td>
<td>8.621 ms</td>
<td>26.713 ms</td>
</tr>
</tbody>
</table>

**Throughput:** ping Two Remote Hosts — 2

Throughput: ping Two Remote Hosts — 3

Example:

\[
TP = 16 \times \frac{P_l - P_s}{t_f - t_s + t_{nl} + t_{ns}} \text{ bits per second}
\]

where:
- \( P_l \) = large packet size, bytes
- \( P_s \) = small packet size, bytes
- \( t_{nl} \) = ping time for larger packet to the near link, seconds
- \( t_{ns} \) = ping time for smaller packet to the near link, seconds
- \( t_f \) = ping time for larger packet to the far link, seconds
- \( t_s \) = ping time for smaller packet to the far link, seconds

**Throughput:** ping Two Remote Hosts — 4

Other measurements needed
- i.e., for quality of service for multimedia

Throughput: ping One Remote Host

This can be expressed as a simple formula:

\[
TP = 16 \times \frac{P_l - P_s}{t_f - t_s} \text{ bits per second, where}
\]

- \( P_l \) = size of large packet
- \( P_s \) = size of small packet
- \( t_f \) = round-trip time for large packet
- \( t_s \) = round-trip time for small packet

Here we have:

\[
TP = 16 \times \frac{1100 - 100}{(60 - 30) \times 10^{-3}} = 16 \times \frac{1000}{30} \times 10^6 = \frac{16}{30} \times 10^6 = 533.3 \text{ Mbits/second}
\]


**Path Performance: Other tools**

- Could use a tool like pathchar, bing, clink, pchar, or metric that performs this calculation for you
- Use `http://www.google.com/` to locate these tools
- pathchar is available in binary form
- Others in source form, need compile with commands something like this:
  - `cd bing-1.1.3`
  - `make`
  - `sudo make install`

**Measuring Throughput**

- Test may be affected by caching in the web browser
- Pathchar is available in binary form
- Better: measure using traffic similar to that created by the application.

**Path measurement with pathchar**

Example: first, start receiver on `ictlab`

```
$ sudo ./pathchar sina.com.hk
```

```
<table>
<thead>
<tr>
<th>hop</th>
<th>ip address</th>
<th>latency</th>
<th>throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>127.0.0.1</td>
<td>0.94 ms</td>
<td>52 Mb/s</td>
</tr>
<tr>
<td>2</td>
<td>192.168.83.2</td>
<td>4.98 ms</td>
<td>26 Mb/s</td>
</tr>
<tr>
<td>3</td>
<td>202.85.129.136</td>
<td>15.79 ms</td>
<td>6 Mb/s</td>
</tr>
<tr>
<td>4</td>
<td>cw7204.vtc.edu.hk</td>
<td>6.31 ms</td>
<td>45 Mb/s</td>
</tr>
<tr>
<td>5</td>
<td>g5-0-0.wttbr01.imsbiz.com</td>
<td>66.53 ms</td>
<td>33 Mb/s</td>
</tr>
<tr>
<td>6</td>
<td>210.87.254.61</td>
<td>0.72 ms</td>
<td>45 Mb/s</td>
</tr>
<tr>
<td>7</td>
<td>210.87.254.129</td>
<td>72.10 ms</td>
<td>52 Mb/s</td>
</tr>
<tr>
<td>8</td>
<td>iadvantage3-RGE.hkix.net</td>
<td>9.04 ms</td>
<td>33 Mb/s</td>
</tr>
<tr>
<td>9</td>
<td>v005-m02.hk01.iadvantage.net</td>
<td>9.88 ms</td>
<td>33 Mb/s</td>
</tr>
<tr>
<td>10</td>
<td>202.85.139.11</td>
<td>0.72 ms</td>
<td>52 Mb/s</td>
</tr>
<tr>
<td>11</td>
<td>202.85.129.136</td>
<td>6.31 ms</td>
<td>52 Mb/s</td>
</tr>
</tbody>
</table>
```

**Measuring Throughput with ttcp**

- Use ttcp, not affected by disk I/O
- Consists of a client and server
- Need have installed at both ends
- May be affected by caching in the web browser
- Better: measure using traffic similar to that created by the application.

```
$ sudo make
$ sudo make install
```

The **ip** program, `iproute`

- The ip program in the iproute package provides complete control over TCP/IP networking in a Linux system
- Provides more networking control facilities than other TCP/IP implementations
- Supports tunneling in many forms
- `iproute` documentation is in two manuals, one for IP routing, the other for tunneling
iproute and iptables

Between these software packages, you can:

- throttle bandwidth for certain computers
- throttle bandwidth to certain computers
- fairly share bandwidth
- protect your network from DoS attacks
- protect Internet from your customers
- multiplex many servers into one, for load balancing or for high availability
- restrict access to your computers
- limit access of your users to other hosts
- do routing based on user id, MAC address, source IP, port, type of service, time of day or content

See the Linux Advanced Routing and Traffic Control HOWTO at [http://itip.org](http://itip.org) for details.

Traffic Measurements: netstat -i

The netstat program can show statistics about network interfaces.

Linux netstat shows lost packets in three categories:
- errors
- drops (queue full: shouldn’t happen)
- overruns (last data overwritten by new data before old data was read: shouldn’t happen)

These values are cumulative (since interface was up)

Could put a load on interface to see current condition, with ping -l, to send large number of packets to destination

See the difference in values

Traffic measurements: ifconfig, ip

`ifconfig` and `ip` give more information than `netstat -i`.

```
$ ifconfig
```

```
eth0 Link encap:Ethernet  HWaddr 00:00:E2:35:AF:EE
     inet addr:172.19.64.52  Bcast:172.19.127.255  Mask:255.255.192.0
     UP  BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     inet6 addr: fe80::200:0:e2ff:fe35af/64 Scope:Link
     Link Rolls:18  Base Address:0xd000
     RX packets:407579600  errors:0  dropped:0  overruns:0  frame:0
     TX packets:1605655688  errors:0  dropped:0  overruns:3  carrier:0
     collisions:0  txqueuelen:100
     RX bytes:3070792102  (2913.7 Mb) RX errors:0  RX frame:0
     TX bytes:2048217058  (1953.3 Mb) TX errors:0  TX carrier:0
     Interrupt:18  Base address:0xd000
     RX: bytes packets errors dropped overrun multicast
     TX: bytes packets errors dropped delivery collisions
```

```
$ ip -s link list
```

```
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    RX packets:2445799644  errors:0  dropped:0  overruns:0  frame:0
    RX bytes:1606151878  (153.6 Mb) RX errors:0  RX frame:0
    TX packets:2404217058  (2313.7 Mb) TX errors:0  TX carrier:0
    Interrupt:19  Base address:0x9b00
    RX bytes packets errors dropped delivery collisions
    TX bytes packets errors dropped delivery collisions
```

Here we run `netstat -i` on ictlab:

```
$ netstat -i
```

```
Kernel Interface table

                  Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0              1500   0 407027830 0 0 0 1603191764 0 0
```

Notice that of the 1.6 billion bytes transmitted, there were 3 overruns.

Next, blast the path you want to test with packets using `ping -l` or the spray program, and measure again.

Getting more info using ip

The `ip -s (-statistics)` option to `ip` provides statistics.

Adding a second gives you even more:

```
$ ip -s -s link list
```

```
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    RX packets:2445799644  errors:0  dropped:0  overruns:0  frame:0
    RX bytes:1606151878  (153.6 Mb) RX errors:0  RX frame:0
    TX packets:2404217058  (2313.7 Mb) TX errors:0  TX carrier:0
    Interrupt:19  Base address:0x9b00
```

Quick Guide to using `ip`: set up interface

Here we set up a network interface and give it the IP address 192.168.0.124:

```
$ ip link set dev eth1 up
$ ip addr add 192.168.0.124/24 dev eth1
```

Two important points:

- If you do not specify the netmask, a netmask of /32 is assumed.
- `brd` + `dev` means obtain broadcast address by setting the host bits.

Quick Guide to using `ip`: set up routes

```
$ ip route add default dev eth1 via 192.168.0.254
$ ip route add 192.168.0.10/24 via 192.168.0.10
```

The last adds a static route to another network

the first adds the default route.

You can omit the device if the network can be reached through a particular interface without any ambiguity

i.e., `ip` is smart enough to figure out which network device to use, though specifying it doesn’t hurt.

Packet Capture
tcpdump, Ethereal and Ntop
What is Packet Capture?

- Real time collection of data as it travels over networks
- Tools called:
  - packet sniffers
  - packet analysers
  - protocol analysers, and sometimes even
  - traffic monitors

When Packet Capture?

- Most powerful technique
- When need to see what client and server are actually saying to each other
- When need to analyse type of traffic on network
- Requires understanding of network protocols to use effectively

Warning: Don’t Get Sacked!

- Be sure that your boss agrees with you capturing packets on your company’s network
- People have been sacked for doing this without permission!
- Some have suffered long lawsuits and criminal records:
- Do not invade the privacy of others
- Capturing passwords with insecure protocols such as telnet, ftp, http (that is not encrypted with TLS) is very easy
- DON’T DO IT!

How to use tcpdump

- Can just type its name (as root):
  - `$ sudo tcpdump`
  - but get a huge amount of data!
  - Can restrict the data collected using a filter
  - A filter may select addresses, protocols, port numbers, ...

tcpdump Filters: host and port

- Show all network traffic to and from 192.168.0.1:
  - `$ tcpdump host 192.168.0.1`
- Show packets to 192.168.0.1:
  - `$ tcpdump dst 192.168.0.1`
- Show packets to port 68 on 192.168.0.1:
  - `$ tcpdump dst 192.168.0.1 and port 68`

tcpdump Filters: networks

- Capture traffic to or from 205.153.60/24:
  - `$ tcpdump net 172.19.64/18`
  - can specify network as source or destination:
  - `$ tcpdump src net 205.153.60/24`
  - `$ tcpdump dst net 172.19.64/18`

tcpdump: some options

- `-c ⟨n⟩` capture a count of ⟨n⟩ packets then stop
- `-w ⟨file⟩` write raw data to ⟨file⟩.
  - Very useful — can filter and analyse this later with tcpdump, ethereal or other tools
- `-i ⟨interface⟩` collect from ⟨interface⟩ instead of lowest numbered network interface
- `-s ⟨bytes⟩` collect no more than ⟨bytes⟩ of data from each packet instead of default 68 bytes
- `-e` show link level info, e.g., Ethernet addresses
- `-x` gives a hexadecimal dump of packets excluding link level data
- `-X` display ASCII as well as hexadecimal if have `-x` option too
**Writing data to a file**

```bash
$ sudo tcpdump -c 1000 -w ~/tmp/tcpdump.pcap
```

tcpdump: listening on eth0

1014 packets received by filter
0 packets dropped by kernel

**Reading a Dumped File**

```bash
$ tcpdump -nr ~/tmp/tcpdump.pcap
```

```
22:32:41.863173 arp who-has 172.19.64.52 tell 172.19.64.52
22:32:41.863198 arp reply 172.19.64.52 is-at 0:0:e2:35:af:ee
22:32:42.082584 arp who-has 172.19.125.229 tell 172.19.125.127
22:32:43.113655 arp who-has 172.19.123.211 tell 172.19.64.63
22:32:44.635149 arp who-has 172.19.127.106 tell 172.19.125.229
22:32:44.874117 arp who-has 172.19.65.45 tell 172.19.124.174
22:32:45.147178 arp who-has 172.19.126.240 tell 172.19.126.50
22:32:45.209507 arp who-has 172.19.127.106 tell 172.19.125.127
22:32:45.239445 arp who-has 172.19.65.16 tell 172.19.127.254
22:32:45.540507 arp who-has 172.19.126.50 (44:30:54:59:43:4d) tell 172.19.65.10
22:32:45.562004 arp who-has 172.19.126.50 tell 172.19.65.2
```

**HTTP**

```bash
$ tcpdump -nr ~/tmp/tcpdump.pcap port http
```

```
22:43:32.633636 192.168.25.9.14075 > 172.19.64.52.http:
   S 1015952778:1015952778(0) win 6144 <mss 1460> (DF)
22:43:32.633693 172.19.64.52.http > 192.168.25.9.14075:
   S 1929920485:1929920485(0) ack 1015952779 win 5840
   <mss 1460> (DF)
22:43:32.635828 192.168.25.9.14075 > 172.19.64.52.http:
   P 1:590(589) ack 1 win 6144 (DF)
22:43:32.635906 172.19.64.52.http > 192.168.25.9.14075:
   . ack 590 win 6479 (DF)
22:43:32.636758 172.19.64.52.http > 192.168.25.9.14075:
   P 1:217(216) ack 590 win 6479 (DF)
22:43:32.636982 172.19.64.52.http > 192.168.25.9.14075:
   F 217:217(0) ack 590 win 6479 (DF)
22:43:32.639080 192.168.25.9.14075 > 172.19.64.52.http:
   R 590:590(0) ack 217 win 0 (DF)
```

**Window**

- `win (nnn)` specifies data window the sending host will accept in future packets
- I.e., the maximum number of bytes
- TCP flow-control:
  - host reduces this number if congested or overloaded
  - will sometimes set to 0 to temporarily halt incoming traffic in this connection

**Ethereal**

King of the Packet Analysers!
Available for Linux, Unix, Windows
Ethereal

- Ethereal can read data captured by `tcpdump`, e.g.,
  `% ethereal -r tcpdump.pcap`
- Or File → Open
- Can capture data itself
- Uses same filter language as `tcpdump`

You can expand any protocol:
- If we click on the + next to Bootstrap Protocol, we can see the details of the DHCP Request:

Display Filters

- Note the box at the bottom of Ethereal for display filters
- Select only some of the packets captured for display
- See man `ethereal` and search for DISPLAY FILTER SYNTAX
- Different syntax than the syntax for capture filters
- Example:
  ```
  ip.src==172.19.64.52 and ip.dest==172.19.64.57
  ```

Tools → Follow TCP Stream

- Can view the contents of an entire TCP stream conversation, in ASCII or in hexadecimal.
- Be careful not to invade your customers' privacy.
- Can use to check if a communications stream is really encrypted

Ntop: monitoring data at a point

- The Ntop program:
  - Listens on a network interface
  - Puts an Ethernet interface into promiscuous mode and displays statistics through a web interface
  - Shows:
    - Percentages of protocols
    - Which machines generate most traffic
    - Which traffic is purely local, which traffic comes from outside, which traffic goes from inside to outside of network
Ntop: Installing

Installation is pretty easy. On my Fedora Core 1 machine:

```
$ rpmbuild --rebuild ntop-3.0-0.src.rpm
$ sudo rpm -Gnv /home/nicku/RPM/RPM/RPMs/i386/ntop-3.0-0.i386.rpm
$ ls -l /etc/ntop.conf
-rwx------ 1 root root 13203 Apr 27 03:47 /etc/ntop.conf
$ sudo cp -a /etc/ntop.conf.sample /etc/ntop.conf
$ sudo emacs /etc/ntop.conf &
# temporarily comment out the line --daemon
$ sudo /usr/bin/ntop @/etc/ntop.conf -A
$ sudo service ntop start
```

Then open the web browser on http://localhost:3000/

Switched Networks

Using Ethereal, tcpdump, Ntop in a switched network

Port Monitoring: Switched Networks

**Problem:** a switched network is really a point-to-point network
- You cannot normally capture the unicast traffic from other hosts on a single switch port
- How do you use Ethereal, tcpdump or Ntop to monitor traffic between a number of hosts?

**Solution:** many switches support port monitoring, where one port can monitor all traffic on a specified VLAN

**Example:** Cisco 3500XL switches provide the `port monitor` command:

```
port monitor vlan VLAN1
```

Are switched networks secure?

Is all unicast traffic on one port of a switch private?
- No, there are tools (dsniff and Ettercap) freely available to automate ARP spoofing and man-in-the-middle attacks, that provide various ways to compromise switch security.

Port Scanning

What is a port scanner?

- Sends packets to various ports on a network device
- Best one available everywhere is nmap
- can identify the OS of the target machine
- Do not port scan arbitrary machines in your company’s network without permission!
- May be interpreted as a cracking attempt

How does nmap identify OS?

- RFCs leave interpretation of some things up to the implementer
- RFCs do not specify how should work if get contradictory flags, strange sequences of inconsistent packets
- Most TCP/IP implementations are not complete
- Every implementation of TCP/IP is different; the “grey areas” are different from one OS to another.
- nmap sends “strange” packets to the machine, detects how reacts, matches this against a file of OS fingerprints
Running `nmap`: Use `xnmap`

```
$ sudo -v
$ sudo xnmap 
```

- Enter the IP address of machine(s) to identify
- select other choices from buttons
- press Start
- `xnmap` is simply a way to easily generate command line options to `nmap` using a graphical interface

**Uses of `nmap`**

- Identify the type of a computer that is causing trouble on the network
- Check what network services a computer is really offering
- compare with `netstat -tua` output
- A cracked computer may be hiding some services with trojaned utilities
- `nmap` can help you discover such services

---

**DNS troubleshooting**

**Troubleshooting DNS Servers**

**The people who write the most common name server (Bind) promote `dig`, deprecate `nslookup`**

- `dig` output is in form of DNS resource records
- can copy and paste straight into DNS database files

---

**`dig`: Checking forward DNS lookup**

```
$ dig nicku.org
```

```
$ dig -x 202.69.77.139
```

**`dig` syntax**

```
dig [(options)] @@[server] (name) (type)
```

- main option is `-x`
- `server` is the name server to query
  - by default, use first server in `/etc/resolv.conf`
- `(name)` is what you want to look up
- `(type)` can be: `a`, `mx`, `axfr`, `soa`, etc.
- default is to get A record(s)
dig: axfr (Zone Transfer)

`dig @ns tyict.vtc.edu.hk axfr`

result can be copied and pasted as a master file in a DNS server

nslookup: an interactive program

```
slookup
```

Note: `nslookup` is deprecated and may be removed from future releases. Consider using the 'dig' or 'host' programs instead. Run `nslookup` with the `'-silent'` option to prevent this message from appearing.

```
> nicku.org
```

Non-authoritative answer:
Name: nicku.org
Address: 202.69.77.139

```
slookup: reverse lookups
```

```
> 202.69.77.139
```

Non-authoritative answer:
Name: 077-139.onebb.com
Authoritative answers can be found from:
ns1.onebb.com internet address = 202.180.160.1
ns2.onebb.com internet address = 202.180.161.1

nslookup: an interactive program

```
> 202.69.77.139
```

Non-authoritative answer:
Name: nicku.org
Address: 202.69.77.139

Telnet: Troubleshooting Email and Other Protocols

```
smt
```

Email: testing with telnet

```
telnet smtp.vtc.edu.hk 25
```

```
telnet pop.vtc.edu.hk 110
```

SMTP commands for sending mail

```
he10 iden1ify your computer
mail from specify sender
rcpt to specify receiver
data indicates start of message body
quit terminate session
```

SMTP: port 25
POP3: port 110

Testing the VTC pop3 server 1

```
telnet pop.vtc.edu.hk 110
```

```
telnet smtp.vtc.edu.hk 25
```

```
telnet smtp.vtc.edu.hk
```

```
telnet pop.vtc.edu.hk
```

```
>> telnet smtp.vtc.edu.hk 25
```

```
Trying 192.168.79.191...
```

Connected to smtp.vtc.edu.hk (192.168.79.191).

Escape character is '^]'.

```
220 pandora.vtc.edu.hk ESMTP Mirapoint 3.2.2-GA; Tue, 25 Feb 2003 11:15:30 +0800 (HKT)
helo nickpc.tyict.vtc.edu.hk
250 pandora.vtc.edu.hk Hello [172.19.32.30], pleased to meet you
```

```
mail from: nicku@nicku.org
```

```
250 nicku@nicku.org... Sender ok
```

```
rcpt to: nicku@nicku.org
```

```
250 nicku@nicku.org... Recipient ok
```

```
data
```

```
354 Enter mail, end with '.' on a line by itself
```

```
My message body.
```

```
.
```

```
250 AFF21826 Message accepted for delivery
```

```
quit
```

```
221 pandora.vtc.edu.hk closing connection
```

Connections closed by foreign host.

Telnet: Troubleshooting Email and Other Protocols

```
smt
```

Email: testing with telnet

```
telnet smtp.vtc.edu.hk 25
```

```
telnet pop.vtc.edu.hk 110
```

SMTP commands for sending mail

```
he10 iden1ify your computer
mail from specify sender
rcpt to specify receiver
data indicates start of message body
quit terminate session
```

Use names, not IP addresses, to specify destination
Testing the pop3 server 2

cmds:

- `retr <messagenum>` retrieve the message with number `<messagenum>`
- `dele <messagenum>` delete the message with message number `<messagenum>`
- `quit` exit POP3 mode

See RFC 1939 for easy-to-read details

First, must authenticate:

- `user (username)`
- `pass (password)`

stat shows number of messages and total size in bytes

list list all the message numbers and size in bytes of each message

Pop3 commands: retrieving mail

Telnet: Testing Other Applications

- Many network protocols are text. Telnet can be helpful in checking:
  - IMAP servers:
    - `$ telnet (hostname) 143`
  - Web servers:
    - `$ telnet (hostname) 80`
  - Ftp servers:
    - `$ telnet (hostname) 21`
  - Even ssh (can check version, if responding):
    - `$ telnet (hostname) 22`

Conclusion

- Check the simple things first
- Be methodical
- Document what you do
- Become familiar with common tools
- Use the tools to become familiar with your network before troubles strike
- Know what is "normal"
- Get permission from the boss before using packet sniffing and port scanners

Return-path: <nicku@nicku.org>
From: Nick Urbanik <nicku@nicku.org>
Date: Tue, 25 Feb 2003 11:15:30 +0800 (HKT)
Message-id: <200302250316.AFF21826@pandora.vtc.edu.hk>
My message body.
My message body.
My message body.