Network Troubleshooting

Troubleshooting Tools

Nick Urbanik <nicku(at)nicku.org>

References

- Joseph D. Sloan, *Network Troubleshooting Tools*, O'Reilly, August 2001, ISBN: 059600186-X, TK5105.5 .S557 2001
- Cisco's *Troubleshooting Overview* at: http://www.cisco.com/univercd/cc/td/doc/cisintwk/itg_v1/tr1901.htm
- Craig Hunt, *TCP/IP Network Administration*, Third Edition O'Reilly, April 2002, ISBN: 059600297-1. Chapter 13 covers troubleshooting TCP/IP.
- Martin A. Brown, *Guide to IP Layer Network Administration with Linux*, http://linux-ip.net/
- Noah Davids, *Don't forget to check your ARP cache*, http://members.cox.net/~ndav1/self_published/The_ARP_cache.doc

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*
- Be familiar with your tools
  - *before trouble strikes*

Troubleshooting: Learn as you go

- Study and be familiar with the *normal behaviour* of your network
- Monitoring tools can tell you when things are wrong
  - if you know what things look like when they are right
- Using tools such as Ethereal can help you understand
  - your network, and
  - TCP/IP— better

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
  - Procedural information
    - How to do things
    - use tools that automatically document what you are doing, e.g., `script`

Documentation Tools

- Use `script`:

```
$ script ~/logs/logfile-$(date +%F-%R).log
```

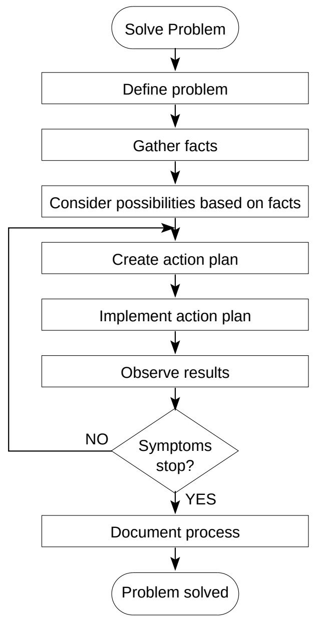
  - starts a new shell
  - all you type, all output goes into the file
  - Add comments with `# I tried this...`
- Use `tee`:

```
$ arp -a | tee outfile
```

- Use `sudo`: all commands are recorded in `/var/log/secure`
- Use `plod` from `http://bullwinkle.deer-run.com/~hal/plod/`
  - lets you record a worksheet easily
  - Perl, so fine on any platform

General Troubleshooting

Problem Solving



Identify the Problem

- 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 be reproduced?

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/`
    - Tools such as Ethereal, `tcpdump`, `ntop` (`http://ntop.org/`) — see slides starting at §83
    - 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

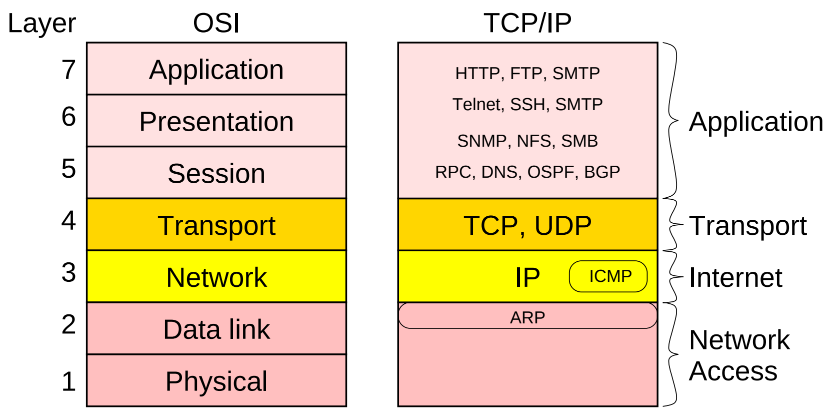
- Record the problem and its resolution in the documentation you maintain for your company.
- Ensure others in your team can benefit from the insight you have gained

Otherwise, Modify Action Plan

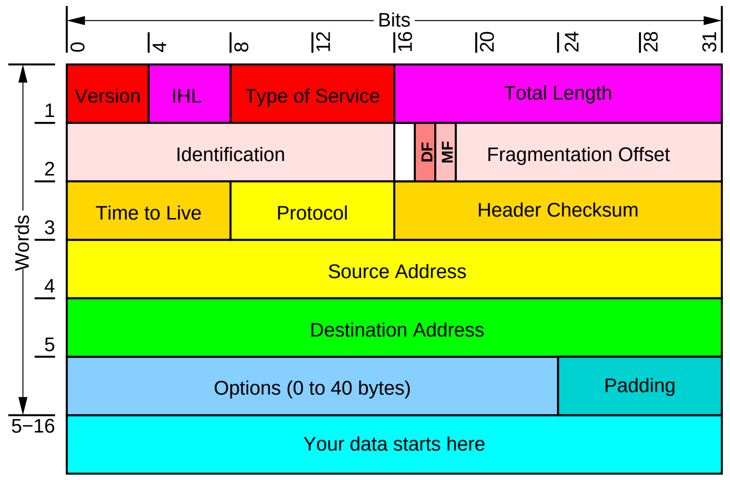
- Go back to the steps in slide §13:
  - Modify your action plan, selecting the next most likely action from your list
  - You may have discovered more information in your investigation, so this can help you focus on likely causes.
- If you have exhausted all the items on your list, and cannot think 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

OSI—TCP/IP



IP Header—Layer 3



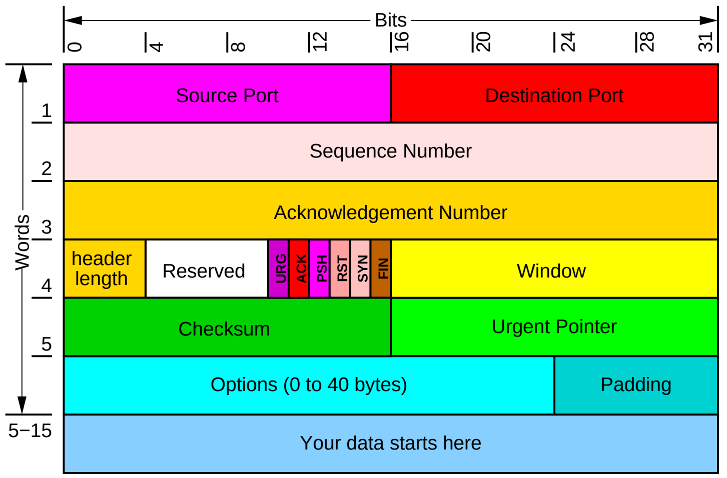
IP Header

- 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.

IP Header (continued)

- Flags — a flag is a 3-bit field of which the 2 low-order bits control fragmentation. One bit specifies whether the packet can be fragmented; the second bit specifies whether the packet is the last fragment in a series of fragmented packets.
- Time-to-Live — this field maintains a counter that gradually decrements down to zero, at which point the datagram is discarded. This prevents packets from looping endlessly.
- Protocol — protocol indicates which upper-layer protocol receives incoming packets after IP processing is complete.
- Header Checksum — this field helps ensure IP header integrity.
- Source Address — this field specifies the sending node.
- Destination Address — this field specifies the receiving node.
- Options — this field allows IP to support various options, such as security.
- Data — this field contains upper-layer information.

TCP Header—Layer 4



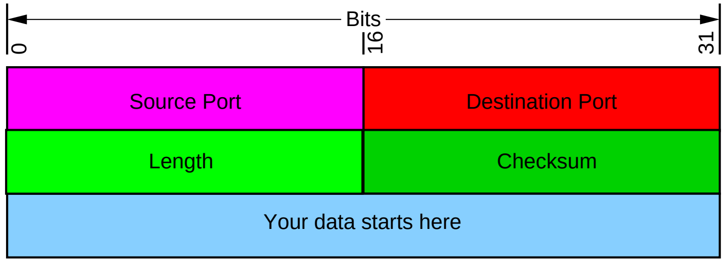
TCP Header

- Source port and destination port — these fields identify the points at which upper-layer source and destination processes receive TCP services.
- Sequence number — this field usually specifies the number assigned to the first byte of data in the current message. Under certain circumstances, it can also be used to identify an initial sequence number to be used in the upcoming transmission.
- Acknowledgment number — this field contains the sequence number of the next byte of data the sender of the packet expects to receive.
- Data offset — this field indicates the number of 32-bit words in the TCP header.
- Reserved — this field is reserved for future use.
- Flags — this field carries a variety of control information.

TCP Header (continued)

- Window — this field specifies the size of the sender's receive window (that is, buffer space available for incoming data).
- Checksum — this field indicates whether the header was damaged in transit.
- Urgent pointer — this field points to the first urgent data byte in the packet.
- Options — this field specifies various TCP options.
- Data — this field contains upper-layer information.

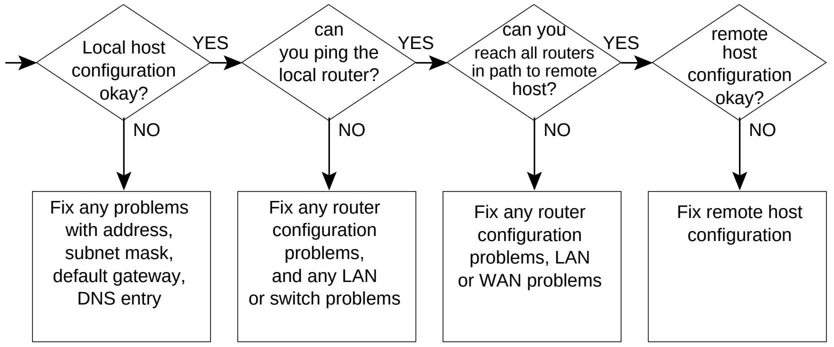
UDP Header—Layer 4



- Source and Destination Port fields serve the same functions as they do in the TCP header.
- Length field specifies the length of the UDP header and data
- Checksum field allows packet integrity checking. It is optional.

Troubleshooting TCP/IP

Troubleshooting TCP/IP


Local host configuration okay?
YES
NO
Fix any problems with address, subnet mask, default gateway, DNS entry
can you ping the local router?
YES
NO
Fix any router configuration problems, and any LAN or switch problems
can you reach all routers in path to remote host?
YES
NO
Fix any router configuration problems, LAN or WAN problems
remote host configuration okay?
NO
Fix remote host configuration

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.

Host Network Configuration tools

`ps` — information about processes

`top` — dynamic information about processes

`netstat` — show connections and services, routing

`lsof` — list open files

`ifconfig` — shows and changes network interfaces

`route` — shows, changes routing table

`ip` — show, change, set network configuration

`arp` — shows MAC addresses

`nmap` — portscanner: shows open ports, identifies OS

Checking (and Setting) Host Configuration

- Solving Boot problems: §32, §33
- Determine IP address, netmask, broadcast address: §34
- Deterine correct MAC ↔ IP address mapping: §35, §36
- Examine routing table: §37
- Examine access controls: §38
- Examine web proxy settings: check web browser
- Examine DNS resolver settings: §39
- Determine services provided: §40, §41
- Determine CPU, memory load conditions (is the server overloaded?) §42

Boot problems: Linux

- Use `grub` to interactively boot the computer (see my extensive grub handout: http://nicku.org/ossi/lab/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/ to replace the Administrator password
- Use the bootable Windows CDROM: http://www.nu2.nu/pebuilder/;
  - 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
Router# show ip interface brief
Router# show ip interface
```

MAC ↔ IP mapping — 1

Linux:

```
$ arp -a
$ ip neigh 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
```

Note that this document: http://members.cox.net/~ndav1/self_published/The_A
has a good discussion on troubleshooting ARP.
The online book at http://linux-ip.net/ has an excellent chapter on ARP.
ARP is probably the most dangerously exposed protocol in a LAN, and can easily be subverted by tools such as `dsniff` and `Ettercap`. Hard-code important ARP entries to avoid attack.

MAC ↔ IP mapping — 2

Windows:

```
C:\> arp -a
```

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`.

Routing Table

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

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

Windows:

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

Cisco IOS:

```
Router# show ip route
```

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.

```
$ 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`.

Cisco IOS:

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

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:

```
$ getent hosts
```

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

Windows: See the output of

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

for the names of the DNS server the resolver will use. Recent versions of Windows provide the program `nslookup` to help with DNS troubleshooting.

See Slides starting at §118 for details of `dig` and `nslookup`.

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:

```
$ chkconfig --list | grep on
```

- Check that the service is listening on the network interface:

```
$ netstat -tua
```

  will show all network connections to this machine.
- The `lsof` program can be helpful in diagnosing problems with network services. See §44.

Windows: Check network connections with

```
C:\> netstat -a
```

Using ps to See If Server Running

- Is the network service running on the server?
- Is the web server running?

```
$ ps aux | grep httpd
```

- Is the DHCP server running?

```
$ ps aux | grep dhcpd
```

- Is the directory server running?

```
$ ps aux | grep slapd
```

- Windows: use the task manager

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 no 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 -tuap` shows all network connections, including those listening, and the processes responsible
- `netstat -tul` 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
- should have been done automatically by `ifconfig`
- `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

Software tools: `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 `ping`s can display round trip time
- Most `ping`s 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 §118
- Ping from more than one host

`fping`: flood ping

- Designed to test a large number of hosts
- more efficient than `ping`
- Used extensively by monitoring software such as `mon`: `http://www.kernel.org/software/mon/`, **nagios**: `http://www.nagios.org/`
- 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/.

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

Quality of a Link

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

Is Bandwidth == Throughput?

- *bandwidth* — the difference between the upper frequency and the lower frequency that a channel can carry
  - measured in Hertz
- *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:

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

$$\begin{aligned} \text{where } C_{\max} &= \text{maximum channel capacity,} \\ B &= \text{bandwidth in Hz,} \\ S &= \text{signal power in W,} \\ N &= \text{noise power in W.} \end{aligned}$$

`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` One Remote Host

- This can be expressed as a simple formula:

$$\begin{aligned} TP &= 16 \times \frac{P_l - P_s}{t_l - t_s} \text{ bits per second, where} \\ P_l &= \text{size of large packet} \\ P_s &= \text{size of small packet} \\ t_l &= \text{round-trip time for large packet} \\ t_s &= \text{round-trip time for small packet} \end{aligned}$$

Here we have:

$$\begin{aligned} TP &= 16 \times \frac{1100 - 100}{(60 - 30) \times 10^{-3}} \\ &= 16 \times \frac{1000}{30 \times 10^{-3}} \\ &= \frac{16}{30} \times 10^{6} \\ &\approx 530{,}000 \text{ bps} \end{aligned}$$

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:

Address	**RTT 100 bytes**	**RTT 1100 bytes**
205.153.61.1	1.380 ms	5.805 ms
205.153.60.2	4.985 ms	12.823 ms
165.166.36.17	8.621 ms	26.713 ms

Throughput: `ping` Two Remote Hosts — 2

Address	RTT 100 bytes	RTT 1100 bytes
205.153.61.1	1.380 ms	5.805 ms
205.153.60.2	4.985 ms	12.823 ms
165.166.36.17	8.621 ms	26.713 ms

- Time difference / 2 (round trip time (RTT) $\rightarrow$ one way)
- Divide by size difference in bits: $8000 = 8 \times (1100 - 100)$
- Multiply by 1000 (ms $\rightarrow$ seconds)
- Mbs = bps/10^6

Near link	Far link	Time difference	Est. Throughput
205.153.61.1	205.153.60.2	3.413 ms	4.69 Mbps
205.153.60.2	165.166.36.17	10.254 ms	1.56 Mbps

$$3.413\,\text{ms} = (12.823 - 4.985) - (5.805 - 1.380)\,\text{ms}$$
$$10.254\,\text{ms} = (26.713 - 8.621) - (12.823 - 4.985)\,\text{ms}$$

Throughput: `ping` Two Remote Hosts — 3

$$TP = 16 \times \frac{P_l - P_s}{t_{fl} - t_{fs} - t_{nl} + t_{ns}} \quad \text{bits per second}$$

where:

$$\begin{aligned}
P_l &= \textbf{\textit{l}}\text{arge packet size, bytes}\\
P_s &= \textbf{\textit{s}}\text{mall packet size, bytes}\\
t_{nl} &= \text{ping time for } \textbf{\textit{l}}\text{arger packet to the } \textbf{\textit{n}}\text{ear link, seconds}\\
t_{ns} &= \text{ping time for } \textbf{\textit{s}}\text{maller packet to the } \textbf{\textit{n}}\text{ear link, seconds}\\
t_{fl} &= \text{ping time for } \textbf{\textit{l}}\text{arger packet to the } \textbf{\textit{f}}\text{ar link, seconds}\\
t_{fs} &= \text{ping time for } \textbf{\textit{s}}\text{maller packet to the } \textbf{\textit{f}}\text{ar link, seconds}
\end{aligned}$$

Throughput: `ping` Two Remote Hosts — 4

$$\begin{aligned}
P_l &= \textbf{\textit{l}}\text{arge packet size, bytes} = 1100\,\text{bytes}\\
P_s &= \textbf{\textit{s}}\text{mall packet size, bytes} = 100\,\text{bytes}\\
t_{nl} &= \text{ping time for } \textbf{\textit{l}}\text{arger packet to the } \textbf{\textit{n}}\text{ear link, seconds}\\
&= 5.805 \times 10^{-3}\,\text{seconds}\\
t_{ns} &= \text{ping time for } \textbf{\textit{s}}\text{maller packet to the } \textbf{\textit{n}}\text{ear link, seconds}\\
&= 1.380 \times 10^{-3}\,\text{seconds}\\
t_{fl} &= \text{ping time for } \textbf{\textit{l}}\text{arger packet to the } \textbf{\textit{f}}\text{ar link, seconds}\\
&= 12.823 \times 10^{-3}\,\text{seconds}\\
t_{fs} &= \text{ping time for } \textbf{\textit{s}}\text{maller packet to the } \textbf{\textit{f}}\text{ar link, seconds}\\
&= 4.985 \times 10^{-3}\,\text{seconds}\\
TP &= 16 \times \frac{P_l - P_s}{t_{fl} - t_{fs} - t_{nl} + t_{ns}} \quad \text{bits per second}\\
&= 16 \times \frac{1100 - 100}{(12.823 - 4.985 - 5.805 + 1.380) \times 10^{-3}} \quad \text{bits per second}\\
&= 16 \times \frac{1000}{3.413 \times 10^{-3}}\\
&\approx 4{,}687{,}958\\
&\approx 4.69\,\text{Megabits per second}
\end{aligned}$$

Throughput: `ping` Two Remote Hosts — 5

$$
\begin{aligned}
P_l &= \textbf{\textit{l}}\text{arge packet size, bytes} = 1100\,\text{bytes}\\
P_s &= \textbf{\textit{s}}\text{mall packet size, bytes} = 100\,\text{bytes}\\
t_{nl} &= \text{ping time for } \textbf{\textit{l}}\text{arger packet to the } \textbf{\textit{n}}\text{ear link, seconds}\\
&= 12.823 \times 10^{-3}\,\text{seconds}\\
t_{ns} &= \text{ping time for } \textbf{\textit{s}}\text{maller packet to the } \textbf{\textit{n}}\text{ear link, seconds}\\
&= 4.985 \times 10^{-3}\,\text{seconds}\\
t_{fl} &= \text{ping time for } \textbf{\textit{l}}\text{arger packet to the } \textbf{\textit{f}}\text{ar link, seconds}\\
&= 26.713 \times 10^{-3}\,\text{seconds}\\
t_{fs} &= \text{ping time for } \textbf{\textit{s}}\text{maller packet to the } \textbf{\textit{f}}\text{ar link, seconds}\\
&= 8.621 \times 10^{-3}\,\text{seconds}\\
TP &= 16 \times \frac{P_l - P_s}{t_{fl} - t_{fs} - t_{nl} + t_{ns}} \quad \text{bits per second}\\
&= 16 \times \frac{1100 - 100}{(26.713 - 8.621 - 12.823 + 4.985) \times 10^{-3}} \quad \text{bits per second}\\
&= 16 \times \frac{1000}{10.254 \times 10^{-3}}\\
&\approx 1{,}560{,}366\\
&\approx 1.56\,\text{Megabits per second}
\end{aligned}
$$

Limitations of measuring with `ping`

- Most modern routers give high priority to routing
  - Expecially switching routers, such as the Cisco 6509 in our Campus
  - Many give much lower priority to answering pings
  - The difference can be so great that the ping reply sometimes comes sooner from a more distant router, which according to our formula, indicates a negative throughput!
  - Do not blindly apply this formula!
- Measurements may not match the kind of traffic created by the application you support
- The big *advantages* of these ICMP measurements are:
  - you do not need access to the machines, and
  - you do not need to install any special software on them.

Path Performance: Other tools

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

```
$ cd bing-1.1.3
$ make
$ sudo make install
```

Path measurement with pathchar

```
$ sudo ./pathchar sina.com.hk
pathchar to sina.com.hk (202.85.139.140)
 can't find path mtu - using 1500 bytes.
 doing 32 probes at each of 45 sizes (64 to 1500 by 32)
 0 localhost (127.0.0.1)
 |   106 Mb/s,   293 us (698 us),  +q 1.18 ms (15.7 KB)
 1 172.19.35.246 (172.19.35.246)
 |    28 Mb/s,   488 us (2.10 ms)
 2 192.168.83.2 (192.168.83.2)
 3  * 1   448 798       2
 |    20 Mb/s,   273 us (3.25 ms)
 4 cw7204.vtc.edu.hk (202.40.210.220)
 |   6.8 Mb/s,   521 us (6.04 ms)
 5 210.176.123.37 (210.176.123.37)
 |    52 Mb/s,   20 us (6.31 ms)
 6 210.87.254.61 (210.87.254.61)
 |   136 Mb/s,   116 us (6.63 ms)
 7 g5-0-0.wttbr01.imsbiz.com (210.87.254.129)
 |    33 Mb/s,   0.94 ms (8.88 ms),  +q 1.48 ms (6.10 KB) *6
 8 iadvantage3-RGE.hkix.net (202.40.161.172)
 |   164 Mb/s,   45 us (9.04 ms),  +q 1.74 ms (35.6 KB) *6
 9 v005-m02.hk01.iadvantage.net (202.85.129.53)
 |   ?? b/s,   -66 us (8.88 ms)
10 202.85.129.136 (202.85.129.136)
 |   ?? b/s,   459 us (9.79 ms)
11 202.85.139.11 (202.85.139.11)
11 hops, rtt 6.18 ms (9.79 ms), bottleneck 6.8 Mb/s, pipe 9361 bytes
```

Measuring Throughput

- May use ftp to transfer a large file, measure time
  - tests whole path
  - problem: affected by disk I/O, xinetd
- May use a *web browser* and measure download time
  - Problem: may be affected by caching in the web browser
  - May be affected by caching in web proxies
- *Better*: measure using traffic similar to that created by the application.

Measuring Throughput with ttcp

- Use ttcp, not affected by disk I/O
- Consists of a client and server
- Need have installed at both ends
- Part of Red Hat Linux, Cisco IOS
- Example: first, start receiver on ictlab:

```
$ ttcp -r -s
ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp
ttcp-r: socket
ttcp-r: accept from 172.19.32.30
ttcp-r: 16777216 bytes in 1.45 real seconds = 11285.88 KB/sec +++
ttcp-r: 9704 I/O calls, msec/call = 0.15, calls/sec = 6684.46
ttcp-r: 0.0user 0.2sys 0:01real 14% 0i+0d 0maxrss 0+2pf 0+0csw
```

- Second, start transmitter on nickpc:

```
$ ttcp -t -s ictlab
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp  -> ictlab
ttcp-t: socket
ttcp-t: connect
ttcp-t: 16777216 bytes in 1.45 real seconds = 11335.64 KB/sec +++
ttcp-t: 2048 I/O calls, msec/call = 0.72, calls/sec = 1416.95
ttcp-t: 0.0user 0.0sys 0:01real 4% 0i+0d 0maxrss 0+2pf 0+0csw
```

iproute

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 tunnelling

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://tldp.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!)
  - drops and overruns indicate faulty flow control — bad!
- 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

Measuring Traffic: netstat -i

- 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      3 BMRU
lo    16436   0   2858402      0      0      0    2858402      0      0      0 LRU
```

- Notice that of the 1.6 billion bytes transmitted, there were 3 overuns.
- Next, blast the path you want to test with packets using ping -l or the spray program, and measure again.

Traffic measurements: `ifconfig`, `ip`

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

```
$ ifconfig eth0
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
          IPX/Ethernet 802.2 addr:33001601:0000E235AFEE
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          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:3055300191 (2913.7 Mb)  TX bytes:2048217058 (1953.3 Mb)
          Interrupt:18 Base address:0xd000
$ ip -s link list eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:00:e2:35:af:ee brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    3058362227 407610495 0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    2140511920 1605768150 0       0       0       0
```

Getting more info using `ip`

- The `-s` (`-statistics`) option to `ip` provides statistics. Adding a second gives you even more:

```
$ ip -s -s link list eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 100
    link/ether 00:00:e2:35:af:ee brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets   errors  dropped overrun mcast
    3070792102 407726727  0       0       0       0
    RX errors: length    crc     frame   fifo    missed
               0          0       0       0       0
    TX: bytes  packets   errors  dropped carrier collsns
    2445799644 1606151878 0       0       0       0
    TX errors: aborted   fifo    window  heartbeat
               0          3       0       0
```

Quick Guide to using `ip`: set up interface

- Here we set up a network interface and give it the IP address 192.168.0.1/24:

```
$ ip link set dev eth1 up
$ ip addr add 192.168.0.1/24 brd + dev eth1
```

- Two important points:
  - If you do not specify the netmask, a netmask of /32 is assumed
  - `brd +` 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 192.168.1.0/24 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*:
  - See `http://www.stonehenge.com/merlyn/`, and `http://www.lightlink.com/spacenka/fors/` for a famous example
- 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!

`tcpdump`

- Available everywhere
- Windows: `http://windump.polito.it/`
- Syntax also used by other programs (such as Ethereal)
- Often it is the only tool available, so good to know
- Works by putting network interface into *promiscuous* mode
  - normal Ethernet interface will ignore packets not addressed to it
  - in *promiscuous mode*, will examine *all packets* that arrive, even those not addressed to 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`: 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
- but you cannot see what you are capturing till later!

`-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

Many more options: `man tcpdump`

`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` filters: protocol

- `tcpdump ip`
- `tcpdump tcp`
- `tcpdump ip proto ospf`
- This will catch DNS name lookups, but not zone transfers (which use tcp):
- `tcpdump udp port 53`

tcpdump filters: combining

- This will not work as you might expect:
- `tcpdump host ictlab and udp or arp`
- Instead, need group with parentheses, and quote:
- `tcpdump "host ictlab and (udp or arp)"`
- many more ways of filtering: `man tcpdump`

Writing data to a file

```
$ 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

```
$ tcpdump -nr ~/tmp/tcpdump.pcap arp
22:32:41.751452 arp who-has 172.19.127.254 tell 172.19.127.29
22:32:41.863173 arp who-has 172.19.64.52 tell 172.19.64.63
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 172.19.65.16 tell 172.19.125.229
22:32:43.113655 arp who-has 172.19.123.211 tell 172.19.65.2
22:32:44.635149 arp who-has 172.19.65.16 tell 172.19.127.106
22:32:44.874117 arp who-has 172.19.65.6 tell 172.19.126.174
22:32:45.147178 arp who-has 172.19.65.16 tell 172.19.126.240
22:32:45.209507 arp who-has 172.19.127.254 tell 172.19.125.127
22:32:45.212484 arp who-has 172.19.127.175 tell 172.19.125.127
22:32:45.239445 arp who-has 172.19.127.254 tell 172.19.125.212
22:32:45.455863 arp who-has 172.19.65.16 tell 172.19.126.194
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

```
$ 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)
```

tcpdump: When reading TCP

- format:
- src ¿ dst: flags data-seqno ack window urgent options
- Flags are some combination of S (SYN), F (FIN), P (PUSH) or R (RST) or a single '.' (no flags).
- The first time tcpdump sees a tcp 'conversation', it prints the sequence number from the packet.
- On subsequent packets of the conversation, the difference between the current packet's sequence number and this initial sequence number is printed.

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

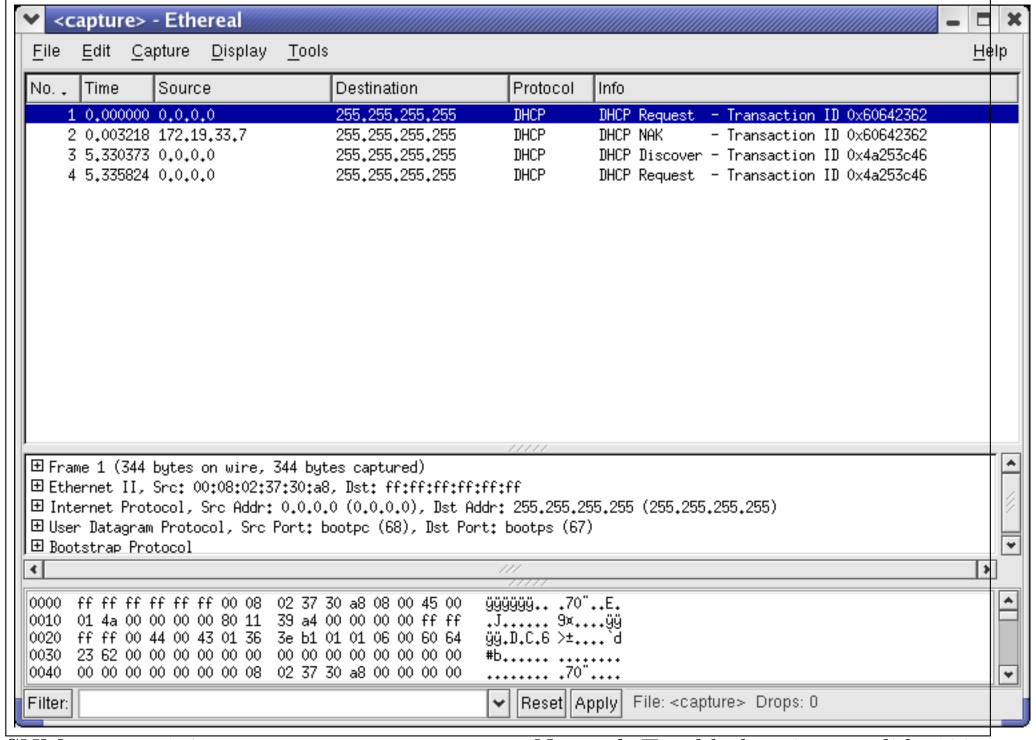
Ethereal
King of the Packet Analysers!
Available for Linux, Unix, Windows

Ethereal

- Ethereal can read data captured by `tcpdump`, e.g.,

```
$ ethereal -r tcpdump.pca
```

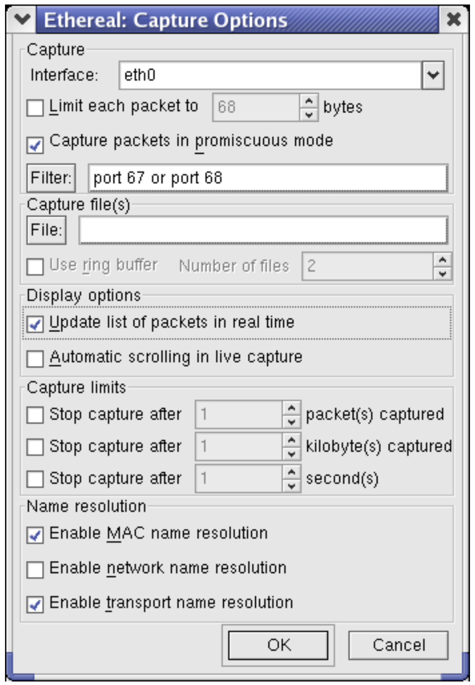
- or File → Open
- Can capture data itself
- Uses same filter language as `tcpdump`



```
No.   Time     Source       Destination      Protocol Info
    1 0.000000 0.0.0.0      255.255.255.255  DHCP     DHCP Request  - Transaction ID 0x60642362
    2 0.003218 172.19.33.7  255.255.255.255  DHCP     DHCP NAK      - Transaction ID 0x60642362
    3 5.330373 0.0.0.0      255.255.255.255  DHCP     DHCP Discover - Transaction ID 0x4a253c46
    4 5.335824 0.0.0.0      255.255.255.255  DHCP     DHCP Request  - Transaction ID 0x4a253c46
```

```
⊞ Frame 1 (344 bytes on wire, 344 bytes captured)
⊞ Ethernet II, Src: 00:08:02:37:30:a8, Dst: ff:ff:ff:ff:ff:ff
⊞ Internet Protocol, Src Addr: 0.0.0.0 (0.0.0.0), Dst Addr: 255.255.255.255 (255.255.255.255)
⊞ User Datagram Protocol, Src Port: bootpc (68), Dst Port: bootps (67)
⊞ Bootstrap Protocol
```

```
0000  ff ff ff ff ff ff 00 08  02 37 30 a8 08 00 45 00   ÿÿÿÿÿÿ.. .70"..E.
0010  01 4a 00 00 00 00 80 11  39 a4 00 00 00 00 ff ff   .J...... 9¤....ÿÿ
0020  ff ff 00 44 00 43 01 36  3e b1 01 01 06 00 60 64   ÿÿ.D.C.6 >±....`d
0030  23 62 00 00 00 00 00 00  00 00 00 00 00 00 00 00   #b...... ........
0040  00 00 00 00 00 00 00 08  02 37 30 a8 00 00 00 00   ........ .70"....
```



Ethereal: Capture Options

Capture
Interface: eth0
[ ] Limit each packet to 68 bytes
[✓] Capture packets in promiscuous mode
Filter: port 67 or port 68

Capture file(s)
File:
[ ] Use ring buffer   Number of files 2

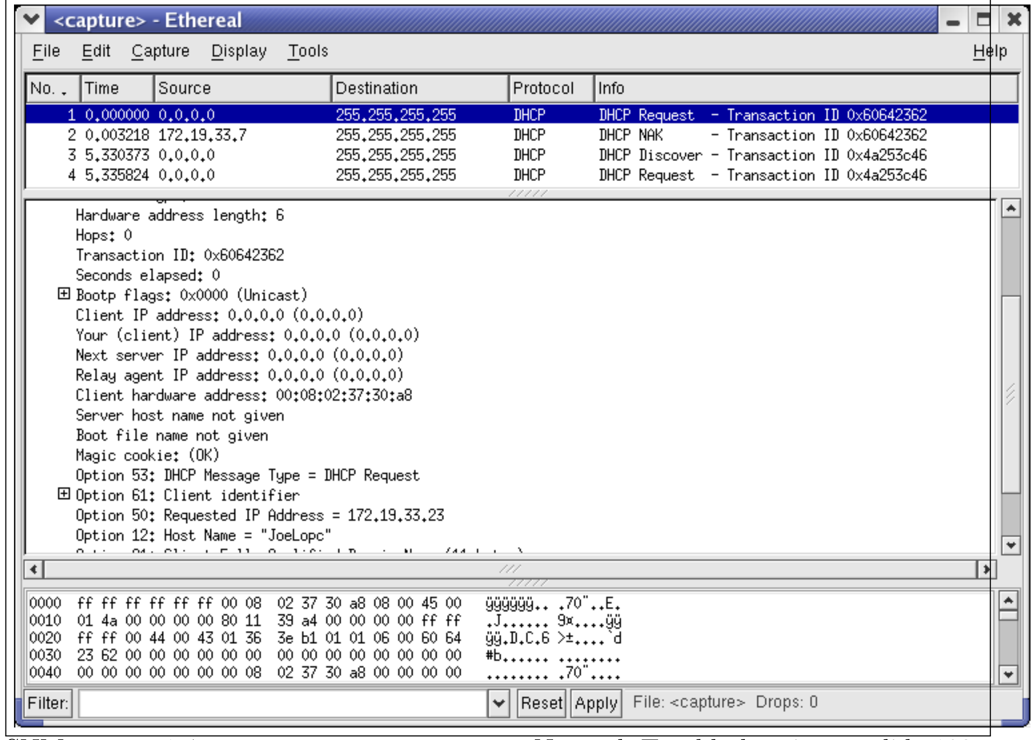
Display options
[✓] Update list of packets in real time
[ ] Automatic scrolling in live capture

Capture limits
[ ] Stop capture after 1 packet(s) captured
[ ] Stop capture after 1 kilobyte(s) captured
[ ] Stop capture after 1 second(s)

Name resolution
[✓] Enable MAC name resolution
[ ] Enable network name resolution
[✓] Enable transport name resolution

You can expand any protocol:

- If we click on the ⊞ next to `Bootstrap Protocol`, we can see the details of the DHCP Request:



```
No.  Time     Source       Destination      Protocol Info
  1 0.000000 0.0.0.0      255.255.255.255  DHCP     DHCP Request  - Transaction ID 0x60642362
  2 0.003218 172.19.33.7  255.255.255.255  DHCP     DHCP NAK      - Transaction ID 0x60642362
  3 5.330373 0.0.0.0      255.255.255.255  DHCP     DHCP Discover - Transaction ID 0x4a253c46
  4 5.335824 0.0.0.0      255.255.255.255  DHCP     DHCP Request  - Transaction ID 0x4a253c46
```

```
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x60642362
    Seconds elapsed: 0
  ⊞ Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0 (0.0.0.0)
    Your (client) IP address: 0.0.0.0 (0.0.0.0)
    Next server IP address: 0.0.0.0 (0.0.0.0)
    Relay agent IP address: 0.0.0.0 (0.0.0.0)
    Client hardware address: 00:08:02:37:30:a8
    Server host name not given
    Boot file name not given
    Magic cookie: (OK)
    Option 53: DHCP Message Type = DHCP Request
  ⊞ Option 61: Client identifier
    Option 50: Requested IP Address = 172.19.33.23
    Option 12: Host Name = "JoeLopc"
```

```
0000  ff ff ff ff ff ff 00 08  02 37 30 a8 08 00 45 00   ÿÿÿÿÿÿ.. .70¨..E.
0010  01 4a 00 00 00 00 80 11  39 a4 00 00 00 00 ff ff   .J...... 9¤....ÿÿ
0020  ff ff 00 44 00 43 01 36  3e b1 01 01 06 00 60 64   ÿÿ.D.C.6 >±....`d
0030  23 62 00 00 00 00 00 00  00 00 00 00 00 00 00 00   #b...... ........
0040  00 00 00 00 00 00 00 08  02 37 30 a8 00 00 00 00   ........ .70¨....
```

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 -Uhv /home/nicku/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.sample
$ 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?

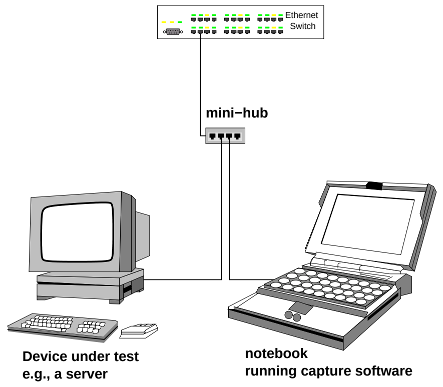
Solution: many switches support *port monitoring*, where one port can monitor all traffic on a specified VLAN

Example:

- Cisco 3500XL switches provide the `port monitor` command:
- `port monitor vlan VLAN1`

How monitor one machine?

- You are asked to check out a server on a switched network: The switch does not support port monitoring (§109), or you do not have administrative access to the switch: what to do?
- Use a small hub, and use a notebook running the capture software



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

DNS troubleshooting

- Suspect DNS when get long timeouts before see any response
- ping name, IP address, see if only IP address works
- tools on Linux, Unix:
  - `dig`, `nslookup`, `host`
- tools on Windows:
  - `nslookup`

DNS: `dig`

- 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 9.2.1 <<>> nicku.org
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23568
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 3
;; QUESTION SECTION:
;nicku.org.            IN      A
;; ANSWER SECTION:
nicku.org.     60      IN      A     202.69.77.139
;; AUTHORITY SECTION:
no-ip.com.              60      IN      NS      nf1.no-ip.com.
no-ip.com.              60      IN      NS      nf2.no-ip.com.
no-ip.com.              60      IN      NS      nf3.no-ip.com.
;; ADDITIONAL SECTION:
nf1.no-ip.com.          60      IN      A       66.185.166.131
nf2.no-ip.com.          60      IN      A       66.185.162.100
nf3.no-ip.com.          60      IN      A       216.66.37.10
;; Query time: 254 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Feb 24 10:55:26 2003
;; MSG SIZE  rcvd: 154
```

`dig`: reverse lookup 1

```
$ dig -x 202.69.77.139
; <<>> DiG 9.2.1 <<>> -x 202.69.77.139
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22117
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 0
;; QUESTION SECTION:
;;139.77.69.202.in-addr.arpa.    IN      PTR
;; ANSWER SECTION:
139.77.69.202.in-addr.arpa. 3600 IN     PTR     077-139.onebb.com.
;; AUTHORITY SECTION:
77.69.202.in-addr.arpa. 3600    IN      NS      ns2.onebb.com.
77.69.202.in-addr.arpa. 3600    IN      NS      ns1.onebb.com.
;; Query time: 310 msec
;; SERVER: 172.19.64.52#53(172.19.64.52)
;; WHEN: Mon Feb 24 11:07:04 2003
;; MSG SIZE  rcvd: 111
```

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: `any`, `a`, `mx`, `axfr`, `soa`, etc.
- default is to get A record(s)

dig: axfr (Zone Transfer)

- `dig` can request a complete zone transfer:

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

; <<>> DiG 9.2.2-P3 <<>> @ns tyict.vtc.edu.hk axfr
;; global options:  printcmd
tyict.vtc.edu.hk.       86400   IN      SOA     ns.tyict.vtc.edu.hk.
 nicku.vtc.edu.hk. 2004031000 3600 1800 604800 600
tyict.vtc.edu.hk.       86400   IN      NS      ns.tyict.vtc.edu.hk.
tyict.vtc.edu.hk.       86400   IN      NS      ns1.tyict.vtc.edu.hk.
tyict.vtc.edu.hk.       86400   IN      NS      dns1.vtc.edu.hk.
tyict.vtc.edu.hk.       86400   IN      NS      dns2.vtc.edu.hk.
000081667.tyict.vtc.edu.hk. 86400 IN    A       172.19.64.92
...
```

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

nslookup: an interactive program

```
$ nslookup
Note:  nslookup is deprecated and may be removed
from future releases.  Consider using the ‘dig’
or ‘host’ programs instead.  Run nslookup with
the ‘-sil[ent]’ option to prevent this message
from appearing.
> nicku.org
Server:         127.0.0.1
Address:        127.0.0.1#53

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

nslookup: reverse lookups

```
> 202.69.77.139
Server:         127.0.0.1
Address:        127.0.0.1#53
Non-authoritative answer:
139.77.69.202.in-addr.arpa      name = 077-139.onebb.com.
Authoritative answers can be found from:
77.69.202.in-addr.arpa  nameserver = ns1.onebb.com.
77.69.202.in-addr.arpa  nameserver = ns2.onebb.com.
ns1.onebb.com   internet address = 202.180.160.1
ns2.onebb.com   internet address = 202.180.161.1
>
```

Telnet: Troubleshooting Email and Other Protocols

Email: testing with `telnet`

- Email protocols SMTP, POP3 are text
- `telnet` a good tool to test them
- syntax:

  `$ telnet` ⟨*server*⟩ ⟨*portnumber*⟩

- SMTP: port 25
- POP3: port 110

Test the VTC mail server:

```
$ 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
Connection closed by foreign host.
```

SMTP commands for sending mail

`helo` identify 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 VTC pop3 server 1

```
$ telnet pop.vtc.edu.hk 110
Trying 192.168.79.12...
Connected to pop.vtc.edu.hk (192.168.79.12).
Escape character is '^]'.
+OK carme.vtc.edu.hk POP3 service (iPlanet Messaging Server 5.2
 Patch 1 (built Aug 19 2002))
user nicku
+OK Name is a valid mailbox
pass password
+OK Maildrop ready
stat
+OK 1 673
```

Testing the pop3 server 2

```
retr 1
+OK 673 octets
Return-path: <nicku@nicku.org>
Received: from pandora.vtc.edu.hk (pandora.vtc.edu.hk [192.168.79.191])
 by carme.vtc.edu.hk (iPlanet Messaging Server 5.2 Patch 1 (built Aug 19 2002))
 with ESMTP id <0HAU00I35H3HGL@carme.vtc.edu.hk> for nicku@ims-ms-daemon
 (ORCPT nicku@nicku.org); Tue, 25 Feb 2003 11:16:29 +0800 (CST)
Received: from nickpc.tyict.vtc.edu.hk ([172.19.32.30])
        by pandora.vtc.edu.hk (Mirapoint Messaging Server MOS 3.2.2-GA)
        with SMTP id AFF21826; Tue, 25 Feb 2003 11:16:01 +0800 (HKT)
Date: Tue, 25 Feb 2003 11:15:30 +0800 (HKT)
From: Nick Urbanik <nicku@nicku.org>
Message-id: <200302250316.AFF21826@pandora.vtc.edu.hk>
My message body.
.
dele 1
+OK message deleted
quit
+OK
Connection closed by foreign host.
```

pop3 commands: retrieving mail

- 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

`retr` ⟨*messagenum*⟩ retrieve the message with number ⟨*messagenum*⟩

`dele` ⟨*messagenum*⟩ delete the message with message number ⟨*messagenum*⟩

`quit`

`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