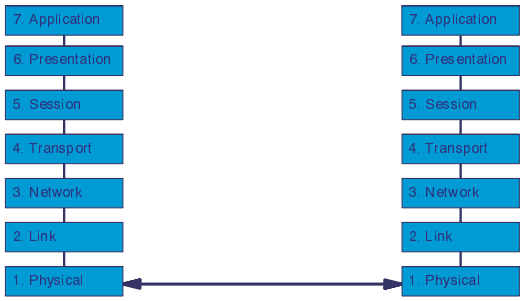
Brian Candler
NSRC

Introduction to Internetworking

The OSI model (Open Systems Interconnection)



The OSI model

- A generic model, not a specific protocol like TCP/IP or X25
- Breaks down networking into simpler parts
- Helps us understand, discuss and compare networks

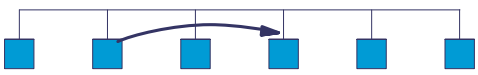
Layer 1 - Physical Layer

- Transfers stream of *bits* from A to B
- Defines connectors, type of cable, maximum length, topology, voltages for 0 and 1, speed (bits per second)

- No concept of bytes or frames

Layer 2 - Link Layer

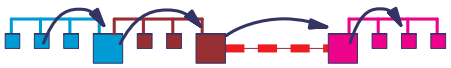
- Organise bits into bytes and frames
  - Special bit patterns as delimiters
- Address frames to a specific machine on a shared (broadcast) medium



- Some layer 2's detect corrupted frames
- Some layer 2's retransmit corrupted frames
  (but *not* ethernet)

Layer 3 - Network Layer

- Send data through multiple hops to far distant networks
- Move data between different Layer 2 types
- Uniform numbering scheme
- Globally scalable



Layer 4 - Transport Layer

- Breaks large streams of data into smaller chunks for layer 3 to carry
- Performs end-to-end error correction and flow control (if required)
- Identifies which *service* (as opposed to which *machine*) you wish to communicate with

7

Layer 5 - Session Layer

- Keeps a session running even if transport layer connection has to be broken and reconnected
- Multiplex data through multiple transport connections for higher throughput
- *NOT USED IN TCP/IP (Application layer is responsible for these functions if required)*

Layer 6 - Presentation Layer

- Performs conversion of data formats, e.g. ASCII to EBCDIC
- *NOT USED IN TCP/IP (Application layer is responsible for this function, if required)*

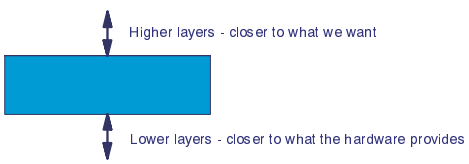
9

Layer 7 - Application Layer

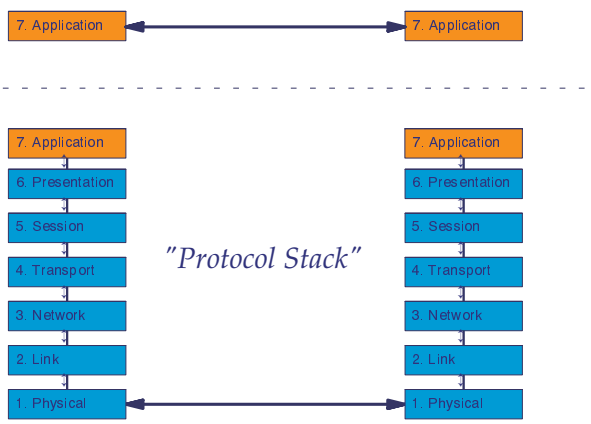
- Performs the useful task we are interested in, e.g. sending mail, transferring web pages
- Application-specific protocols (e.g. SMTP, HTTP) carried through the stack to the remote machine
- Applications think they are talking directly to each other - of course we know different!

Interaction between layers

- Each layer provides services to the layer directly above
- Each layer makes use of services provided by the layer directly below



11



The OSI Model

- Who has seen this before?
- Any questions?

Examples of layer 1/layer 2

- Ethernet
  - Layer 1: 10baseT, 100baseTX, 1000baseTX etc
  - Layer 2: Media Access Control (MAC)
- Other local area networks
  - FDDI, Token Ring, Wireless 802.11
- Asynchronous serial links (e.g. PC/modem)
  - Layer 1: RS232
  - Layer 2: Async PPP, SLIP
- Synchronous serial links
  - Layer 1: RS232, X21, HSSI, POS (STM1, OC3)
  - Layer 2: PPP, Cisco HDLC, Frame Relay...
- Various DSL technologies / ATM

Examples of Layer 3

- IP - the Internet Protocol
- Provides a "best effort" datagram delivery service
- Machines are identified by IP numbers, which are globally unique
  - IP version 4: 32-bit IP numbers
  - IP version 6: 128-bit IP numbers
  - Still doubtful as to whether IPv6 will ever be widely deployed
- IPv4 is the only internetworking protocol we'll consider

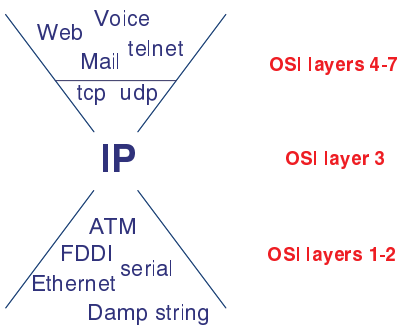
Examples of Layer 4

- TCP: Transmission Control Protocol
  - Guarantees reliable delivery of data, in order
  - Assigns sequence numbers, performs automatic retransmission
  - Connection-based ("virtual circuit")
  - Flow control
  - Allows you to select which process you are communicating with (port number)
- UDP: User Datagram Protocol
  - Connectionless, no delivery guarantees
  - Used when the whole payload can fit inside a single datagram and data loss is acceptable

Examples of Layer 7 protocols

- HTTP: Hyper Text Transfer Protocol
  - For web browser to retrieve pages from web server
  - Runs over TCP
- SMTP: Simple Mail Transfer Protocol
  - For one machine to deliver mail messages to another machine
  - Runs over TCP
- DNS: Domain Name System
  - For machine to issue name-to-address queries
  - Runs over UDP (mostly)
- Many, many more

The Hourglass Model



Putting it into practice

- Let's see how we can use the OSI model to test and debug our network

Testing at layer 1

- Check link status lights
- A hub can identify a malfunctioning device and isolate that port (look for "partition light")
- Neither of these is foolproof
  - A cable may be good enough to make the link status light come on, but not good enough to transfer data reliably
- Use a cable tester
- Visual inspection

Ask your machine for its interface status

```
$ ifconfig fxp0
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.0.201 netmask 0xffffff00 bcast 192.168.0.255
        ether 00:00:00:60:2a:85
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
```

This is layer 1 status information as detected by the hardware

MAC address (layer 2 configuration)
IP address (layer 3 configuration)

Testing at layer 2

- Ethernet doesn't have a direct way to perform tests at layer 2
  - So use a layer 3 test (ping), but direct it to another machine on the same LAN as you
  - Look at interface error counters: **netstat -i**
- Most serial links have layer 2 testing built in
  - e.g. PPP performs an initial exchange of packets to establish the connection (LCP) and can send "keepalive" packets to continuously test it
  - On many routers, the interface status will tell you whether layer 2 is up

Testing at layer 3

- "ping" sends a special "echo request" packet. If it arrives, an "echo response" will be sent back
- Proves network working in both directions

```
$ ping -c5 147.28.0.39
PING 147.28.0.39 (147.28.0.39): 56 data bytes
64 bytes from 147.28.0.39: icmp_seq=0 ttl=51 time=391.264 ms
64 bytes from 147.28.0.39: icmp_seq=1 ttl=51 time=394.113 ms
64 bytes from 147.28.0.39: icmp_seq=2 ttl=51 time=392.129 ms
64 bytes from 147.28.0.39: icmp_seq=3 ttl=51 time=396.275 ms
64 bytes from 147.28.0.39: icmp_seq=4 ttl=51 time=349.356 ms

--- 147.28.0.39 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 349.356/384.627/396.275/17.720 ms
```


Testing at layer 3 (contd)

- Sending large-sized pings can be useful
  - larger packets are more sensitive to bit errors and ethernet collisions
- ping -s1472 -c100 147.28.0.39
  - 20 bytes IP header + 8 bytes ICMP header + 1472 bytes data = 1500 byte datagram
  - You need to be "root" to do this
- pings are not TCP or UDP, but ICMP
  - Internet Control Message Protocol

Testing at layer 3 (contd)

- "traceroute" sends a series of packets and uses this to show the intervening routers
- Unix: **traceroute -n 147.28.0.39**
- Windows: **tracert -d 147.28.0.39**
- The -n/-d flag is used to prevent DNS lookups. Use it!
  - If you have a networking problem, you don't want to have the additional uncertainty of whether your DNS servers are working or reachable

Testing at layer 3 (contd)

- "tcpdump" shows you packets going in and out of an interface
- tcpdump -i fxp0 -n -s1500 -X
- Use a filter to select packets of interest
  - tcpdump -i fxp0 -n -s1500 -X tcp port 80
    - TCP packets to or from port 80 only
  - tcpdump -i fxp0 -n -s1500 -X host 192.168.0.1
    - all packets to or from that host only
- The '-n' flag prevents DNS lookups; use it.

Testing at layer 4/7

- Many Internet layer 7 protocols use plain-text messages. You can therefore drive them directly from a keyboard.
- "telnet" can be used to open a TCP connection to a remote server
  - there is no equivalent for UDP
  - the port number selects which server you want
- Once the connection is open, you type layer 7 messages for whichever application protocol is running on this port

Example: fetching web pages without a web browser!

```
$ telnet www.nsrc.org 80          <-- Standard port number for HTTP
Trying 128.223.162.27...
Connected to www.nsrc.org.
Escape character is '^]'.
GET / HTTP/1.0 [Enter]            <-- HTTP request (note: first line is case-sensitive)
Host: www.nsrc.org [Enter]
[Enter]
HTTP/1.1 200 OK                   <-- HTTP response headers followed by HTML data
Server: JavaWebServer/1.1.1
Content-Length: 6307
Content-Type: text/html
Last-Modified: Mon, 27 Dec 2004 13:55:20 GMT
Connection: close
Date: Thu, 30 Dec 2004 14:34:45 GMT

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN"> ...
```

This is an enormously useful tool for debugging

- e.g. when testing a web server
- eliminates any possible errors with web browser (because you're not using one!)
- see all the raw HTTP response headers, which the web browser usually hides from the user
- also check in your web server's log file to see how the request was handled

If you don't know where to start?

- Do a layer 3 test (ping the remote host)
- If ping works: you likely have an application problem. Perform layer 4/7 tests.
- If ping doesn't work, then there is a networking problem
  - check you can ping other machines on the same LAN as you
  - ping nearby routers and ones further away until you find the area of the problem

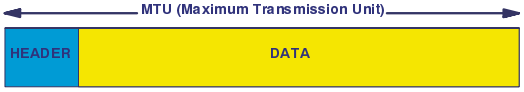
Testing and debugging

- Any questions?
- Reminder: if you've forgotten what the "-c" flag to "ping" does, how can you find out?
- Simple practical exercise

The Internet Protocol (IP)

- Layer 3 in the TCP/IP stack
- Delivers chunks of data - "datagrams" - across an internetwork
- Scales to global network - The Internet
- Integrates different LAN technologies
- RFC 791 (IPv4)

IP Datagram Structure



- Header
  - Source IP address - *where it came from*
  - Destination IP address - *where it is going to*
  - Header checksum
  - Other fields (TTL, Layer 4 protocol identifier, Fragmentation information)
- Data
  - The actual data you want to carry
- Total size up to MTU bytes
  - What limits the MTU?

IP Addresses / IP Numbers

- IP number identifies a device (host)
- Globally unique for every host
  - Why?
- Independent of layer 2 addresses
- 32 bit binary number

Example:

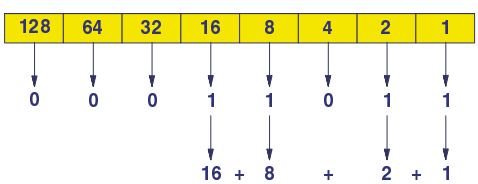
11001110000110111110111000000101

IP numbers (continued)

- Convert to decimal for convenience
- Group into bytes (8 bits) and convert each in turn; separate with periods

- There is nothing special about the 8 bit boundaries; to the computer it is still a single 32 bit number

Binary to Decimal conversion



= 27

Class Examples

- Convert the following IP numbers to decimal:

  10000010001110110000101000011110

  10010011000111000000000000100111

Decimal to Binary conversion

- Keep subtracting to find all the bits

103	128		0
103	64	64	1
39	32	32	1
7	16		0
7	8		0
7	4	4	1
3	2	2	1
1	1	1	1

→ 01100111

- Or cheat by using a conversion table :-)

Class Example

- Convert the following IP number to binary

  128.223.162.27

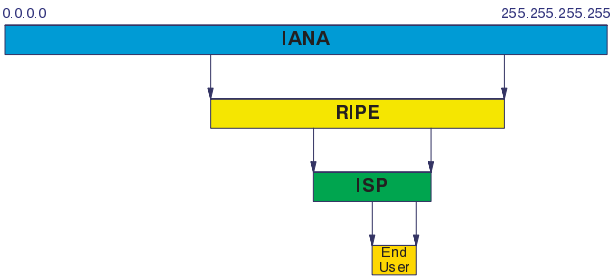
Properties of IP numbers

- What is the smallest IP number?
- What is the largest IP number?
- How many IP numbers are there altogether?

Where do you get IP numbers?

- We must ensure they are unique
- Get a block of IP addresses from upstream provider (Local Internet Registry)
- Local registry gets larger block from regional registry
  - ARIN (Americas), RIPE (Europe), APNIC (Asia/Pacific), LACNIC (Latin America/Carribean), AFRINIC (Africa)
- Regional registry gets them from IANA - the Internet Assigned Numbers Authority

Allocation of IP addresses



"Provider-based allocation"

- Ensures uniquess of IP addresses
- Addresses used by one ISP are contiguous
- This helps keep routing tables small - see later
- Unfortunately means that if you change ISP, you have to renumber your network

IP address ranges

- IP ranges are given as PREFIXES
- First "n" bits of number are fixed, remaining bits you are free to allocate
- n is called the Prefix Length and is written /**n**

Example:
206.27.244.64/27

Prefix Example

- 206.27.244.64/27

1. Convert to binary

11001110 00011011 11110100 01000000

2. Treat as 32 bit number, divide at prefix

11001110000110111111010001000000

27 bits of prefix - Fixed

5 bits host number - free to use

Prefix Example (continued)

3. Calculate lowest IP number and convert back to decimal

11001110000110111111010001000000

11001110 00011011 11110100 01000000

206.27.244.64

4. Calculate highest IP number and convert back to decimal

11001110000110111111010001011111

11001110 00011011 11110100 01011111

206.27.244.95

Prefix Example (continued)

- How many IP numbers does a /27 prefix give you?

Prefixes and IP numbers

- Notice the difference between a Prefix and an IP number

206.27.244.64 - One IP number

206.27.244.64/27 - A whole *range* of IP numbers

- A /32 prefix is also a single IP number

The "Golden Rules" for allocating IP numbers

1. All hosts on the same network (layer 2) must have the SAME unique prefix
2. All hosts must have DIFFERENT host numbers (the part after the prefix)
3. Host numbers of all 0's and all 1's are reserved and must not be used for hosts

Using the Golden Rules

- 206.27.244.64/27

```
11001110000110111111010001000000  ✘  (Network Address)
11001110000110111111010001000001  ✓
11001110000110111111010001000010  ✓
11001110000110111111010001000011  ✓
11001110000110111111010001000100  ✓
                    ⋮
11001110000110111111010001011110  ✓
11001110000110111111010001011111  ✘  (Broadcast Address)
```

Why the Golden Rules?

- Just looking at *one* prefix gives you the path to *all* the hosts on a network
- Ensure uniqueness of IP addresses

More Examples

"Classful" addresses

- In the old days, IP space was broken up into fixed "classes"
  - Class A: addresses beginning with '0' (binary)
    - 0-127.x.x.x, Prefix length /8
  - Class B: addresses beginning with '10'
    - 128-191.x.x.x, Prefix length /16
  - Class C: addresses beginning with '110'
    - 191-223.x.x.x.x, Prefix length /24
- This was wasteful of IP addresses
  - Now have CIDR: Classless Interdomain Routing
  - But Class D (224-239), Class E (240-255) still special cases

Nowadays, prefixes must be given explicitly

- Some software lets you enter e.g. "/27"
- Older software requires you to enter a "netmask"
  - /27 = 27 ones followed by 5 zeros
  - 11111111 11111111 11111111 11100000
  - 255.255.255.224 (decimal)
  - 0xffffffe0 (hex)
- Just use a conversion table