CS551
Layering and Addressing

Bill Cheng

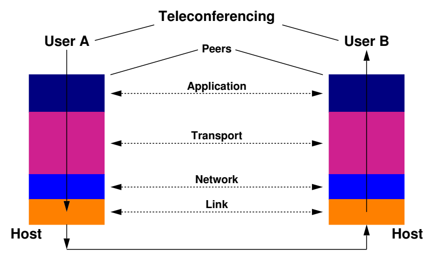
http://merlot.usc.edu/cs551-f12

Protocols

- Set of rules governing communication between network elements (applications, hosts, routers)
- Protocols define:
  - Format and order of messages
  - Actions taken on receipt of a message
- Protocols are hard to design
  - We need design guidelines!

Layering

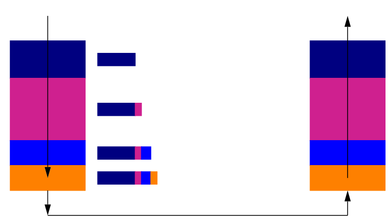
- Layering: technique to simplify complex systems



Layering Characteristics

- Each layer relies on services from layer below and exports services to layer above
- Interface defines interaction
- Hides implementation - layers can change without disturbing other layers (black box)

Layer Encapsulation



OSI Model: 7 Protocol Layers

- Physical: how to transmit bits
- Data link: how to transmit frames
- Network: how to route packets hop2hop
- Transport: how to send packets end2end
- Session: how to tie flows together
- Presentation: byte ordering, security
- Application: everything else!

Layering General Issues

- Reliability
- Flow control
- Fragmentation
- Multiplexing
- Connection setup (handshaking)
- Addressing/naming (locating peers)

Example: Transport Layer

- First end-to-end layer
- End-to-end state
- May provide reliability, flow control, and congestion control

Example: Network Layer

- Point-to-point communication
- Network and host addressing
- Routing

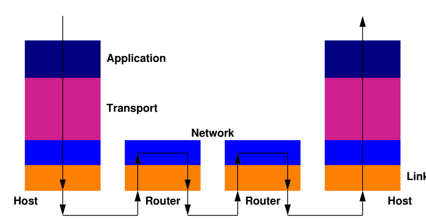
Is Layering Harmful?

- Sometimes..
  - Layer N may duplicate lower level functionality (e.g., error recovery).
  - Layers may need same info (timestamp, MTU).
  - Strict adherence to layering may hurt performance.
  - Naïve layer implementation frequently hurts performance.

Course Focus

IP & TCP

IP



IP Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source IP Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                Example Internet Datagram Header
```


IP Functions

- Type of service
  - Not used until recently
- Identification, flags and fragment offset
  - Fragmentation
- Time to live
  - Bounded delivery
- Protocol
  - (De)multiplexing higher layer protocols (analogous to *port numbers* in TCP)
- Length
  - IP packet length limited to 64K
- Header checksum
  - Ensures some degree of header integrity

Fragmentation

- Forwarding costs per packet
  - Nice if we can send large chunks of data
- Different link-layers have different MTUs
- Fragmentation
  - Intra-network
  - Inter-network

Fragmentation Is Harmful

- Uses resources poorly
  - Example of packet just bigger than MTU
- Poor end-to-end performance
  - Loss of a fragment
- Reassembly is hard
  - Buffering constraints

Path MTU Discovery

- Hosts dynamically discover MTU of path
  - Send message with don't fragment bit
  - Get ICMP message indicating size
- What happens if path changes?
  - Increasing/decreasing path MTU
- Usually implemented by the *transport* layer
  - Expected that future routing protocols will provide MTU information

Path MTU

- Algorithm:
  - Initialize MTU to MTU to next hop
  - Send datagrams with DF bit set
  - If *"datagram too big"*, decrease MTU
  - Periodically (>5mins, or >1min after previous increase), increase MTU
- Some routers will return proper MTU
- MTU values cached in routing table

Addressing in IP

- IP addresses are names of interfaces
- DNS names are names of hosts
- DNS binds host names to interfaces
- Routing binds interface names to paths

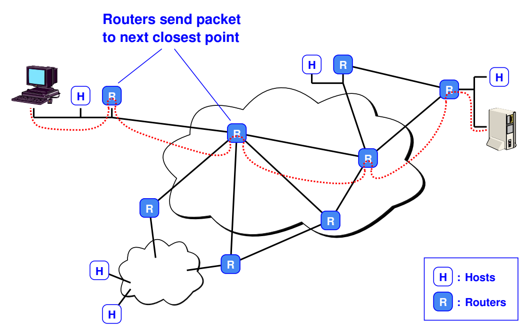
Addressing Considerations

- Fixed length or variable length?
- Issues:
  - Flexibility
  - Processing costs
  - Header size
- Engineering choice: IP uses fixed length addresses

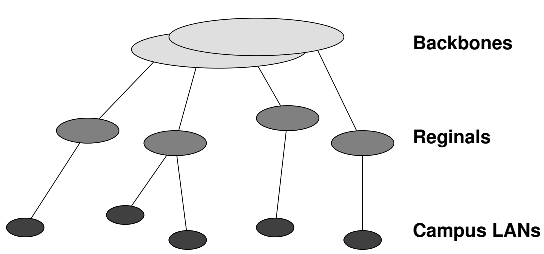
Addressing Considerations

- Structured vs flat
- Issues
  - Need structure for designing scalable binding from interface name to route!
  - How many levels? Fixed? Variable?

Packet Traveling Through the Internet



IP Addressing Hierarchy



Some Special IP Addresses

- 127.0.0.1: local host (a.k.a. The loopback address.
- 127.X.X.X: same as above.
- Host bits all set to 0: network address.
- Host bits all set to 1: broadcast address.
- 0.0.0.0: this host on this network.

IP Addresses

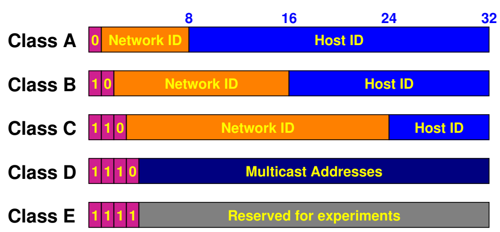
- Fixed length: 32 bits
- Initial classful structure

```
High Order Bits  Format                             Class
---------------  ---------------------------------  -----
      0           7 bits of net, 24 bits of host      a
      10         14 bits of net, 16 bits of host      b
      110        21 bits of net,  8 bits of host      c
      111        escape to extended addressing mode
```


Class Sizes

- Total IP address size: 4 billion
  - Class A: 128 networks, 16M hosts
  - Class B: 16K networks, 64K hosts
  - Class C: 2M networks, 256 hosts

IP Address Classes (Some Are Obsolete)

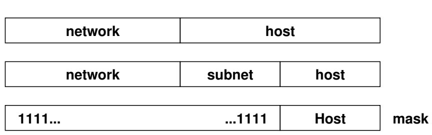


Subnet Addressing

- Very few LANs have close to 64K hosts
  - for networks with more than 255 hosts
- Variable length subnet masks
  - could subnet a class B into several chunks

Subnetting

- Simple and elegant way to reduce the total number of network addresses that are assigned.



Subnetting Example

- Assume an organization was assigned address 150.100 (10010110 01100100)
- Assume < 100 hosts per subnet
- How many host bits do we need?
  - seven
- What is the network mask?
  - 11111111 11111111 11111111 10000000
  - 255.255.255.128

Using Subnet Mask

- Assume a packet arrives with address 150.100.12.176 (10010110 01100100 00001100 10110000)
- Step 1: AND address with subnet mask
  - (150.100.12.176) *AND* (255.255.255.128)
  - result: 150.100.12.128 which is the target network
- Target network has hosts in the range
  - 150.100.12.129 - 150.100.12.254

IP Address Problem (1991)?

- Address space depletion
  - in danger of running out of classes A and B
- Routing table explosion

Some Problems

- Class B sparsely populated
  - but people refuse to give it back
- One solution: assign class C addresses
  - how do you allocate to avoid routing table explosion?
- Addresses not geographically related
  - addresses given by your ISP
  - blocks assigned to various countries

Classless Inter-domain Routing (CIDR)

- Do not use classes to determine network ID
- Use common part of address as network number
  - i.e., use netmask (/xx bits) for network address
- E.g., addresses 192.4.16 - 192.4.31 have the first 20 bits in common. Thus, we use this as the network number
  - 192.4.16: 11000000 00000100 00010000
  - 192.4.31: 11000000 00000100 00011111
  - netmask is /20
- In CIDR /xx is valid for almost any xx

CIDR Addressing

- A block of addresses is described by
  - address prefix
  - mask
- Examples:
  - 10/8 denotes addresses from 10.0.0.0 to 10.255.255.255
  - /xx indicates number of significant bits

Classless Inter-Domain Routing (CIDR)

- Several key ideas
  - allocate addresses to organizations in ***power-of-two blocks***
  - organizations get addresses from ***provider's block***
  - provider ***aggregates***
- Addresses:
  - address utilization
  - routing table size

Old classes and CIDR

- Class A network is a /8
- Class B network is a /16
- Class C network is a /24

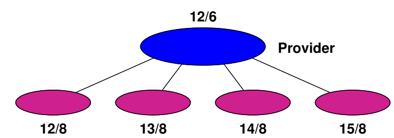
CIDR prefixes

CIDR Blk Prfx	# Eqiv Class C	# of Hosts
/28	1/16	16
/27	1/8	32
/26	1/4	64
/25	1/2	128
/24	1 class C	256
/23	2	512
/22	4	1,024
/21	8	2,048
/20	16	4,096
/19	32	8,192
/18	64	16,384
/17	128	32,768
/16	256=1 class B	65,536
/15	512	131,072
/14	1,024	262,144
/13	2,048	524,288

CIDR example

- Network admin is allocated 8 class C chunks, 201.10.0.0 to 201.10.7.255 (11001001 00001010 00000000 00000000 to 11001001 00001010 00000111 11111111)
- Allocation uses 3 bits of class C space
- Remaining 21 bits are network number, written as 201.10.0.0/21
- 21 is prefix indication which must be carried with address
- Routing protocols carry this prefix

CIDR Illustration



12/6 = 0 0 0 0 1 1 0 0 .0.0.0

12/8 = 0 0 0 0 1 1 0 0 .0.0.0

13/8 = 0 0 0 0 1 1 0 1 .0.0.0

14/8 = 0 0 0 0 1 1 1 0 .0.0.0

15/8 = 0 0 0 0 1 1 1 1 .0.0.0

CIDR Shortcomings

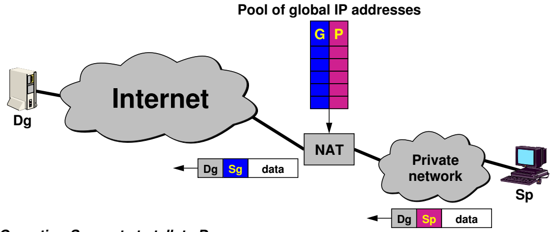
- Multi-homing
- Customer selecting a new provider
- Some other ideas
  - geographic addressing
- Is it enough? Do we need a new IP?

Network Address Translation (NAT)

- Kludge (but useful)
- Sits between your network and the Internet
- Translates local network layer addresses to global IP addresses
- Has a pool of global IP addresses (less than number of hosts on your network)

NAT Illustration

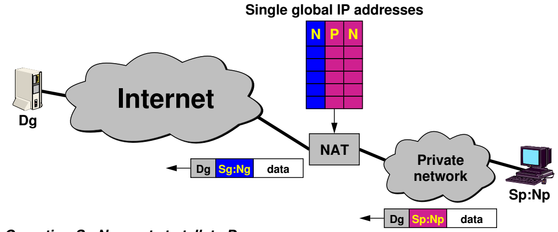
Pool of global IP addresses



Operation:Sp wants to talk to Dg:
- Create *Sg-Sp* mapping (*g* for global and *p* for private)
- Replace *Sp* with *Sg* for outgoing packets
- Replace *Sg* with *Sp* for incoming packets

NAT Illustration - Overloading

Single global IP addresses



Operation:Sp:Np wants to talk to Dg:
- Create *Ng-Sp:Np* mapping
- Replace *Sp:Np* with *Sg:Ng* for outgoing packets
- Replace *Sg:Ng* with *Sp:Np* for incoming packets

NAT Disadvantages

- Breaks end-to-end semantics
  - internal computers cannot be addressed from the outside
  - more on ***End-to-end Argument*** later [Saltzer81a]
- NAT box modifies packets on the fly
  - sometimes needs to modify app-level info, not just packet headers
    - ex. if IP address is in packet data (not just header), as in FTP
  - therefore forces application-specific gateways (for protocols that do not work behind NAT box)
  - state information stored in NAT box
  - new failure modes

NAT Advantages

- Breaks end-to-end semantics
  - internal computers cannot be addressed from the outside
  - an effective security kludge!
- Cheap, relatively easy, relatively fast
- Don’t have to tell your cable modem company :-)

IPv6

- The Right Way
  - just make bigger addresses
  - and fix a bunch of other stuff
- But... requires a whole new protocol stack
  - slow adoption
  - but but... seems to be gaining momentum
- Cell phones
- Asia
- We will not talk about IPv6 in this class

IPv6

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Things to Think About

- How much IP functionality is really useful?
- Was IP a success by design or by accident?
- More on this later [Clark88a]

Hints for Computer System Design

Where? \ Why?	*Functionality (Does it work?)*	*Speed (Is it fast enough?)*	*Fault-tolerance (Does it keep working?)*
Completeness	Separate normal and worst case	Shared load End-to-end Safety first	End-to-end
Interface	Do one thing well: Don't generealize Get it right Don't hide power Use procedure arguments Leave it to the client Keep basic interfaces stable Keep a place to stand	Make it fast Split resources Static analysis Dynamic translation	End-to-end Log updates Make actions atomic
Implementation	Plan to throw one away Keep secrets Use a good idea again Divide the conquer	Cache answers Use hints Use brute force Compute in background Batch processing	Make actions atomic Use hints