Review of Internet Architecture and Protocols

Professor Guevara Noubir

Northeastern University

noubir@ccs.neu.edu

Reference Textbooks:

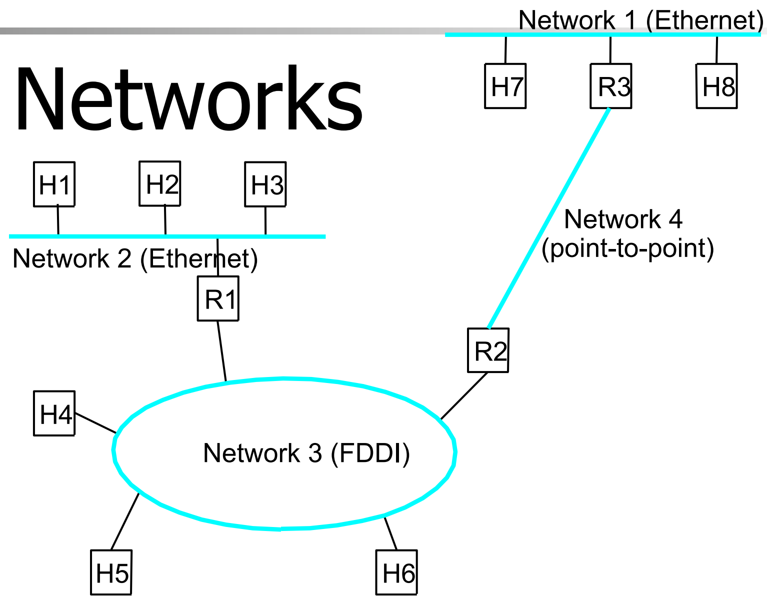
Computer Networks: A Systems Approach, L. Peterson, B. Davie, Morgan Kaufmann

Outline

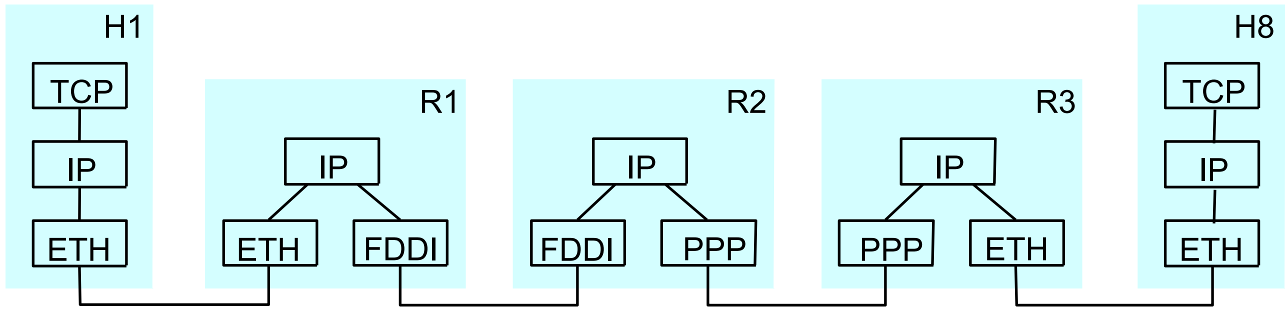
- Internet Protocol
- Addressing
- IP over LAN
- Routing
- End-to-End protocols
- Naming

IP – The Internet

- Concatenation of Networks

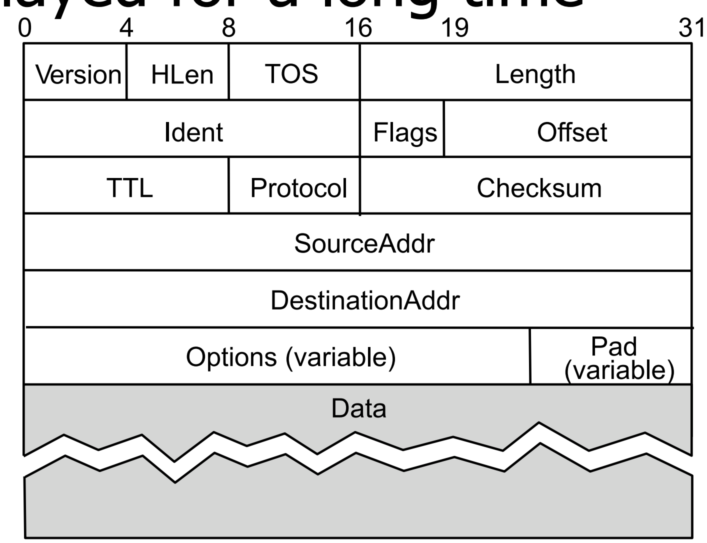


- Protocol Stack



Service Model

- Connectionless (datagram-based)
- Best-effort delivery (unreliable service)
  - packets are lost
  - packets are delivered out of order
  - duplicate copies of a packet are delivered
  - packets can be delayed for a long time
- Datagram format

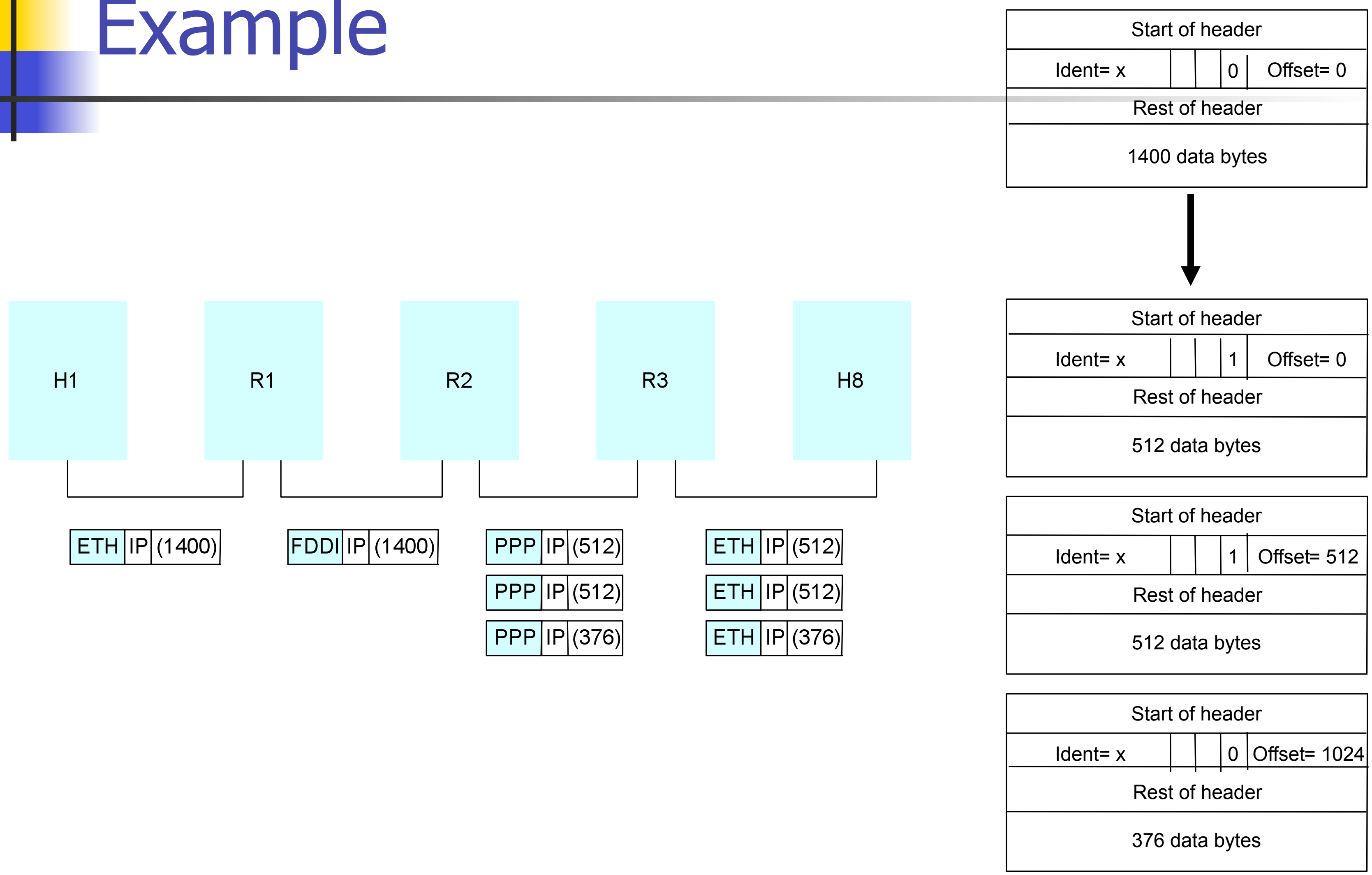


Fragmentation and Reassembly

- Each network has some MTU
- Strategy
  - fragment when necessary (MTU < Datagram)
  - re-fragmentation is possible
  - fragments are self-contained datagrams
  - use CS-PDU (not cells) for ATM
  - delay reassembly until destination host
  - do not try to recover from lost fragments

  - hosts are encouraged to perform "path MTU discovery"

Example



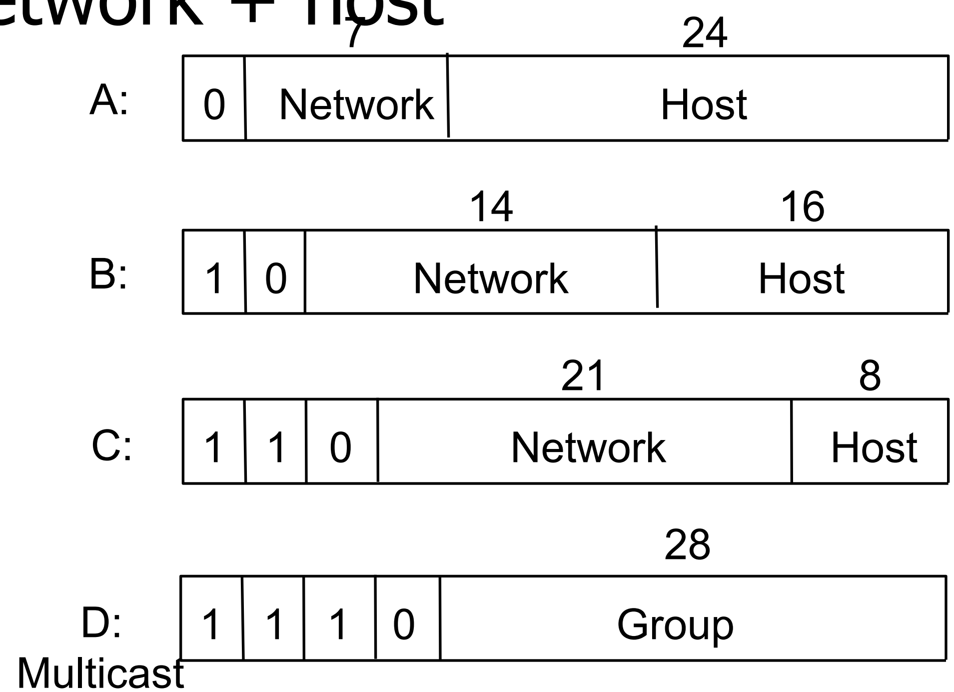
H1
R1
R2
R3
H8
ETH IP (1400)
FDDI IP (1400)
PPP IP (512)
PPP IP (512)
PPP IP (376)
ETH IP (512)
ETH IP (512)
ETH IP (376)
Start of header
Ident= x 0 Offset= 0
Rest of header
1400 data bytes
Start of header
Ident= x 1 Offset= 0
Rest of header
512 data bytes
Start of header
Ident= x 1 Offset= 512
Rest of header
512 data bytes
Start of header
Ident= x 0 Offset= 1024
Rest of header
376 data bytes

Internet Control Message Protocol (ICMP) RFC 792

- Integral part of IP but runs as ProtocolType = 1 using an IP packet
- Codes:
  - Echo (ping)
  - Redirect (from router to inform source host of better route)
  - Destination unreachable (protocol, port, or host)
  - TTL exceeded (so datagrams don't cycle forever)
  - Checksum failed
  - Reassembly failed

Global Addresses

- Properties
  - globally unique
  - hierarchical: network + host



- Dot Notation
  - 10.3.2.4
  - 128.96.33.81
  - 192.12.69.77

Datagram Forwarding

- Strategy
  - every datagram contains destination's address
  - if directly connected to destination network, then forward to host
  - if not directly connected to destination network, then forward to some router
  - forwarding table maps network number into next hop
  - each host has a default router
  - each router maintains a forwarding table
- Example (R2)

Network Number	Next Hop
1	R3
2	R1
3	interface 1
4	interface 0

Address Translation

- Map IP addresses into physical addresses
  - destination host
  - next hop router
- Techniques
  - encode physical address in host part of IP address
    - not reasonable
  - table-based
- ARP
  - table of IP to physical address bindings
  - broadcast request if IP address not in table
  - target machine responds with its physical address
  - table entries are discarded if not refreshed

ARP Details

- Request Format
  - HardwareType: type of physical network (e.g., Ethernet)
  - ProtocolType: type of higher layer protocol (e.g., IP)
  - HLEN & PLEN: length of physical and protocol addresses
  - Operation: request or response
  - Source/Target-Physical/Protocol addresses
- Notes
  - table entries timeout in about 15 minutes
  - update table with source when you are the target
  - update table if already have an entry
  - do not refresh table entries upon reference

ARP Packet Format

0 8 16 31

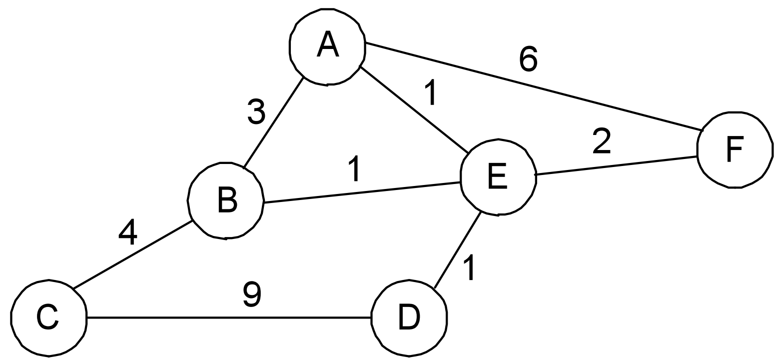
<table>
<tr><td colspan="2">Hardware type = 1</td><td>ProtocolType = 0x0800</td></tr>
<tr><td>HLen = 48</td><td>PLen = 32</td><td>Operation</td></tr>
<tr><td colspan="3">SourceHardwareAddr (bytes 0 – 3)</td></tr>
<tr><td colspan="2">SourceHardwareAddr (bytes 4 – 5)</td><td>SourceProtocolAddr (bytes 0 – 1)</td></tr>
<tr><td colspan="2">SourceProtocolAddr (bytes 2 – 3)</td><td>TargetHardwareAddr (bytes 0 – 1)</td></tr>
<tr><td colspan="3">TargetHardwareAddr (bytes 2 – 5)</td></tr>
<tr><td colspan="3">TargetProtocolAddr (bytes 0 – 3)</td></tr>
</table>

Dynamic Host Configuration Protocol (DHCP)

- IP addresses of interfaces cannot be configured when manufactured (like for Ethernet)
- Configuration is an error-prone process
- Solution: centralize the configuration information in a DHCP server:
  - DHCP server discovery: broadcast a DHCPDISCOVER request
  - Requests are relayed (unicast) to the server by DHCP relays
  - DHCP server broadcast replies with <HWADDR, IPADDR, lease-info>
  - DHCP runs on top of UDP (broadcast IP and MAC addresses,)

Routing Overview

- Forwarding vs Routing
  - forwarding: to select an output port based on destination address and routing table
  - routing: process by which routing table is built
- Network as a Graph



- Problem: Find lowest cost path between two nodes
- Factors
  - relatively static: topology
  - dynamic: load

Distance Vector

- Each node maintains a set of triples
  - `(Destination, Cost, NextHop)`
- Exchange updates directly connected neighbors
  - periodically (on the order of several seconds)
  - whenever table changes (called *triggered* update)
- Each update is a list of pairs:
  - `(Destination, Cost)`
- Update local table if receive a “better” route
  - smaller cost
  - came from next-hop
- Refresh existing routes; delete if they time out

Example

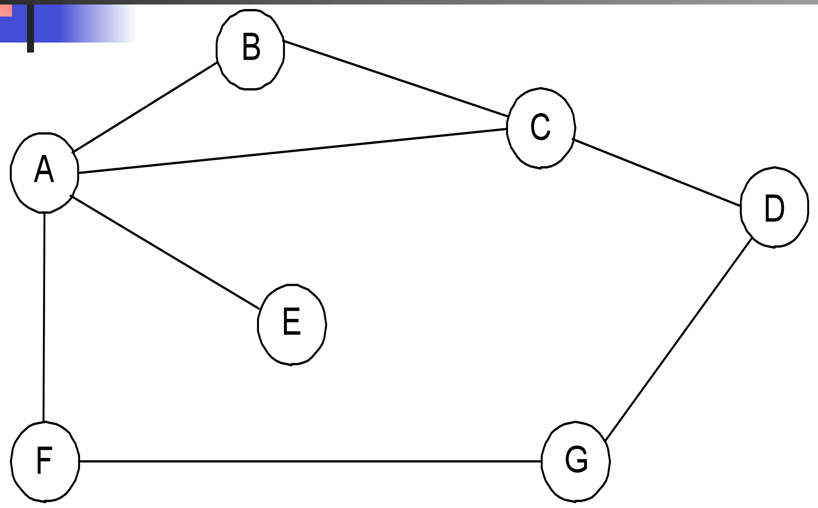


Table for node B

Destination	Cost	NextHop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

Routing Information Protocol (RIP)

- Uses Bellman-Ford's algorithm
- Protocol over UDP, port 520
- Distance-vector protocol
- Protocol overview:
  - Init: send a request packet over all interfaces
  - On response reception: update the routing table
  - On request reception:
    - if request for complete table (*address family*=0) send the complete table
    - else send reply for the specified address (infinity=16)
  - Regular routing updates:
    - every 30 seconds part/entire routing table is sent (broadcast) to neighboring routers
  - Triggered updates: on metric change for a route
  - Simple authentication scheme

Link State

- Strategy
  - send to all nodes (not just neighbors) information about directly connected links (not entire routing table)
- Link State Packet (LSP)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet

Link State (cont)

- Reliable flooding
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - do no forward already received LSPs
  - generate new LSP periodically
    - increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP
    - discard when TTL=0

Route Calculation

- Dijkstra's shortest path algorithm
- Let
  - *N* denotes set of nodes in the graph
  - *l* (*i*, *j*) denotes non-negative cost (weight) for edge (*i*, *j*)
  - *s* denotes this node
  - *M* denotes the set of nodes incorporated so far
  - *C*(*n*) denotes cost of the path from *s* to node *n*

```
M = {s}
for each n in N - {s}
   C(n) = l(s, n)
while (N != M)
   M = M union {w} such that C(w) is the minimum for
       all w in (N - M)
   for each n in (N - M)
      C(n) = MIN(C(n), C (w) + l(w, n ))
```

Open Shortest Path First

- IP protocol (not over UDP), reliable (sequence numbers, acks)
- Protocol overview: link state protocol
  - The link status (cost) is sent/forwarded to all routers (LSP)
  - Each router knows the exact topology of the network
  - Each router can compute a route to any address
  - simple authentication scheme
- Advantages over RIP
  - Faster to converge
  - The router can compute multiple routes (e.g., depending on the type of services, load balancing)
  - Use of multicasting instead of broadcasting (concentrate on OSPF routers)

Popular Interior Gateway Protocols

- RIP: Route Information Protocol
  - distributed with Unix
  - distance-vector algorithm
  - based on hop-count
- OSPF: Open Shortest Path First
  - more recent Internet standard
  - uses link-state algorithm
  - supports load balancing
  - supports basic integrity check http://www.faqs.org/rfcs/rfc2328.html

How to Make Routing Scale

- Flat versus Hierarchical Addresses
- Inefficient use of Hierarchical Address Space
  - class C with 2 hosts (2/256 = 0.78% efficient)
  - class B with 255 hosts (255/65536 = 0.39% efficient)
- Still Too Many Networks
  - routing tables do not scale
  - route propagation protocols do not scale

Subnetting

- Add another level to address/routing hierarchy: *subnet*
- *Subnet masks* define variable partition of host part
- Subnets visible only within site

Network number	Host number

Class B address

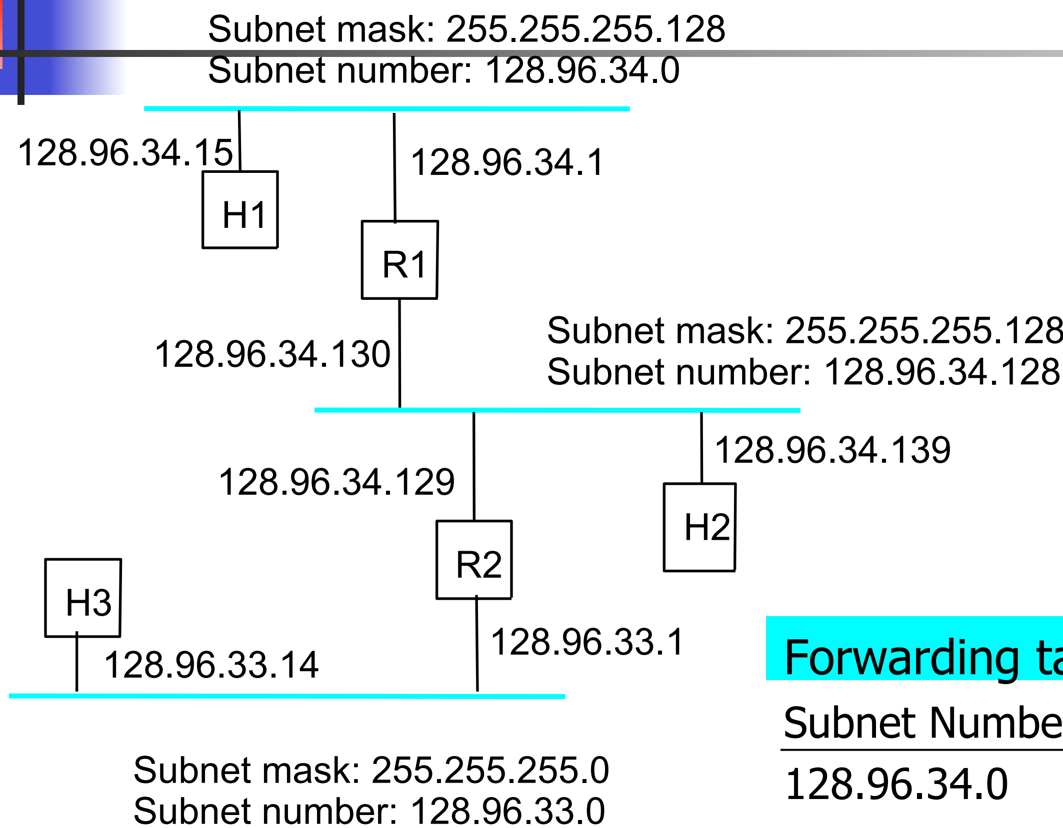
111111111111111111111111	00000000

Subnet mask (255.255.255.0)

Network number	Subnet ID	Host ID

Subnetted address

Subnet Example



Forwarding table at router R1

Subnet Number	Subnet Mask	Next Hop
128.96.34.0	255.255.255.128	interface 0
128.96.34.128	255.255.255.128	interface 1
128.96.33.0	255.255.255.0	R2

Forwarding Algorithm

```
D = destination IP address
for each entry (SubnetNum, SubnetMask, NextHop)
   D1 = SubnetMask & D
   if D1 = SubnetNum
      if NextHop is an interface
         deliver datagram directly to D
      else
         deliver datagram to NextHop
```

- Use a default router if nothing matches
- Not necessary for all 1s in subnet mask to be contiguous
- Can put multiple subnets on one physical network
- Subnets not visible from the rest of the Internet

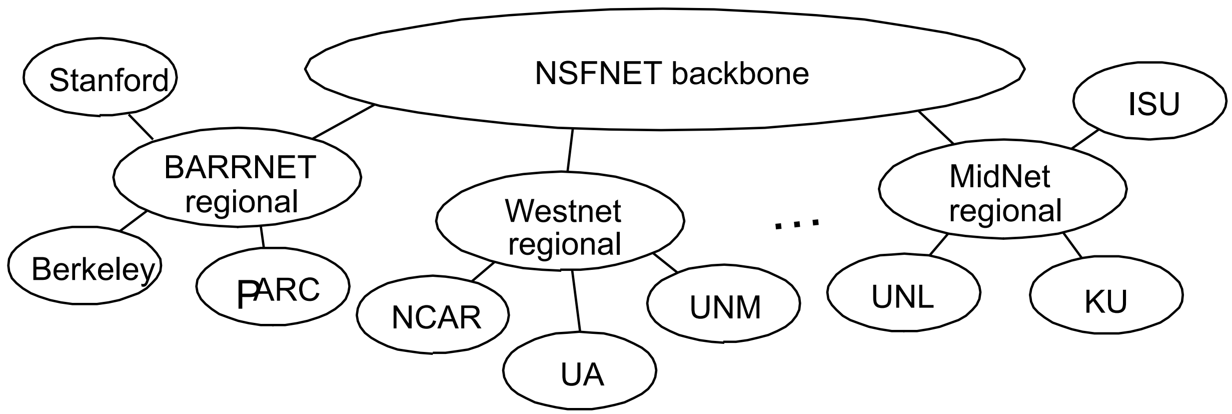
Supernetting: Restructuring IP Addresses

- Assign block of contiguous network numbers to nearby networks
- Called CIDR: Classless Inter-Domain Routing
- Represent blocks with a single pair

  `(first_network_address, count)`
- Restrict block sizes to powers of 2
  - E.g., 192.4.16 – 192.4.31: /20
- Use a bit mask (CIDR mask) to identify block size
- All routers must understand CIDR addressing

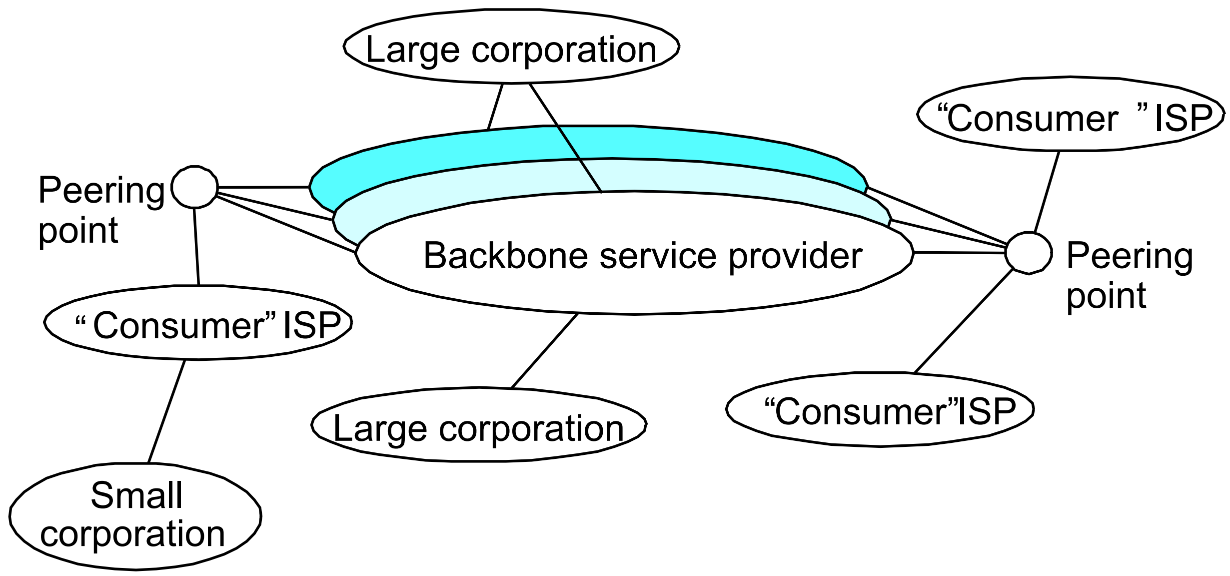
Internet Structure

Past



Internet Structure

Yesterday



Route Propagation

- Know a smarter router
  - hosts know local router
  - local routers know site routers
  - site routers know core router
  - core routers know almost everything
- Autonomous System (AS)
  - corresponds to an administrative domain
  - examples: University, company, backbone network
  - assign each AS a 16-bit number
- Two-level route propagation hierarchy
  - interior gateway protocol (each AS selects its own)
  - exterior gateway protocol (Internet-wide standard)

EGP: Exterior Gateway Protocol

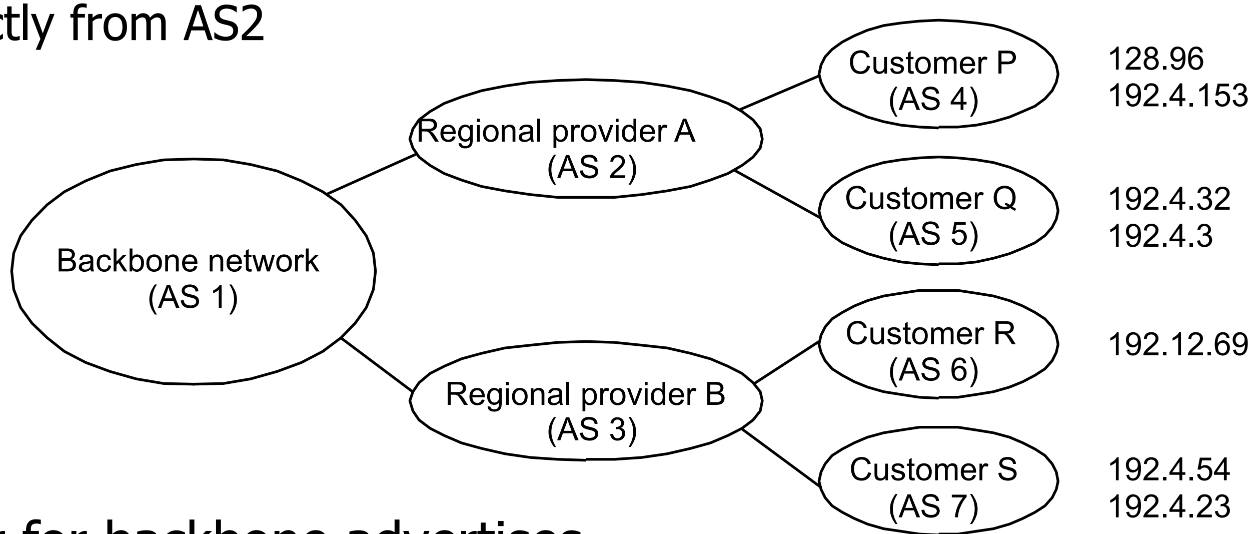
- Overview
  - designed for tree-structured Internet
  - concerned with *reachability*, not optimal routes
- Protocol messages
  - neighbor acquisition: one router requests that another be its peer; peers exchange reachability information
  - neighbor reachability: one router periodically tests if the other is still reachable; exchange HELLO/ACK messages; uses a k-out-of-n rule
  - routing updates: peers periodically exchange their routing tables (distance-vector)

BGP-4: Border Gateway Protocol

- AS Types
  - stub AS: has a single connection to one other AS
    - carries local traffic only
  - multihomed AS: has connections to more than one AS
    - refuses to carry transit traffic
  - transit AS: has connections to more than one AS
    - carries both transit and local traffic
- Each AS has:
  - one or more border routers
  - at least one BGP *speaker* that advertises:
    - local networks
    - other reachable networks (transit AS only)
    - advertise complete *path* of AS to reach destination
    - Possibility to withdraw path
  - in the backbone BGP speakers inject learned information using IBGP + intradomain routing protocol to reach border routers

BGP Example

- Speaker for AS2 advertises reachability to P and Q
  - network 128.96, 192.4.153, 192.4.32, and 192.4.3, can be reached directly from AS2



- Speaker for backbone advertises
  - networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path (AS1, AS2).
- Speaker can cancel previously advertised paths

END TO END PROTOCOLS

End-to-End Protocols

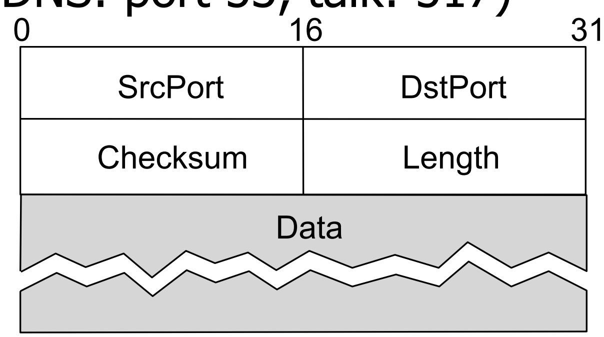
- Goal: turn host-to-host packet delivery into process-to-process communication channel
- Underlying best-effort network
  - drop messages
  - re-orders messages
  - limits messages to some finite size
  - delivers messages after an arbitrarily long delay
  - delivers duplicate copies of a given message
- Common end-to-end services
  - guarantee message delivery
  - deliver messages in the same order they are sent
  - deliver at most one copy of each message
  - support arbitrarily large messages
  - support synchronization
  - allow the receiver to flow control the sender
  - support multiple application processes on each host

Types of End-to-End Protocols

- Simple asynchronous demultiplexing service (e.g., UDP)
- Reliable byte-stream service (e.g., TCP)
- Request rePly Service (e.g., RPC)

Simple Demultiplexor (UDP)

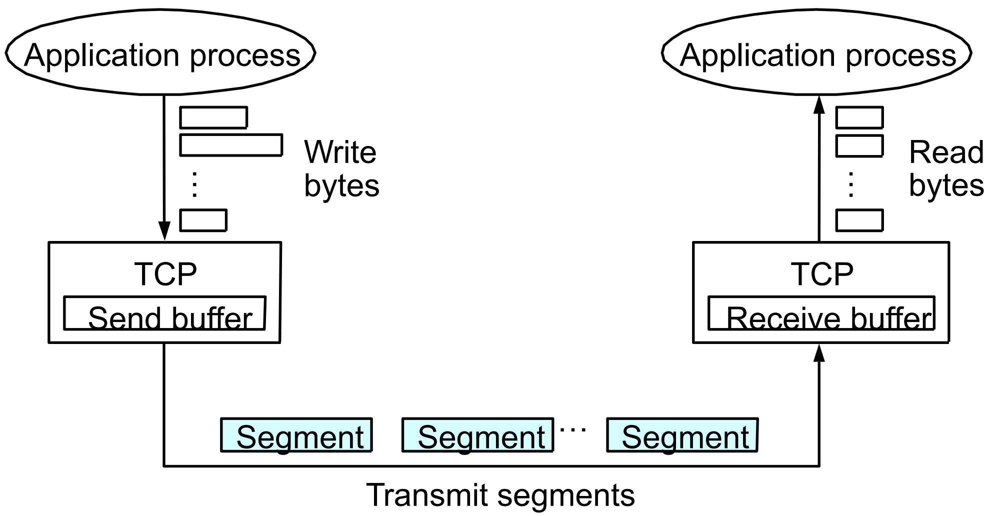
- Unreliable and unordered datagram service
- Adds multiplexing
- No flow control
- Endpoints identified by ports
  - servers have *well-known* ports (e.g., DNS: port 53, talk: 517)
  - see **`/etc/services`** on Unix
- Header format



- Optional checksum
  - pseudo header + UDP header + data
  - Pseudo header = protocol number, source IP addr, dest IP addr, UDP length

TCP Overview

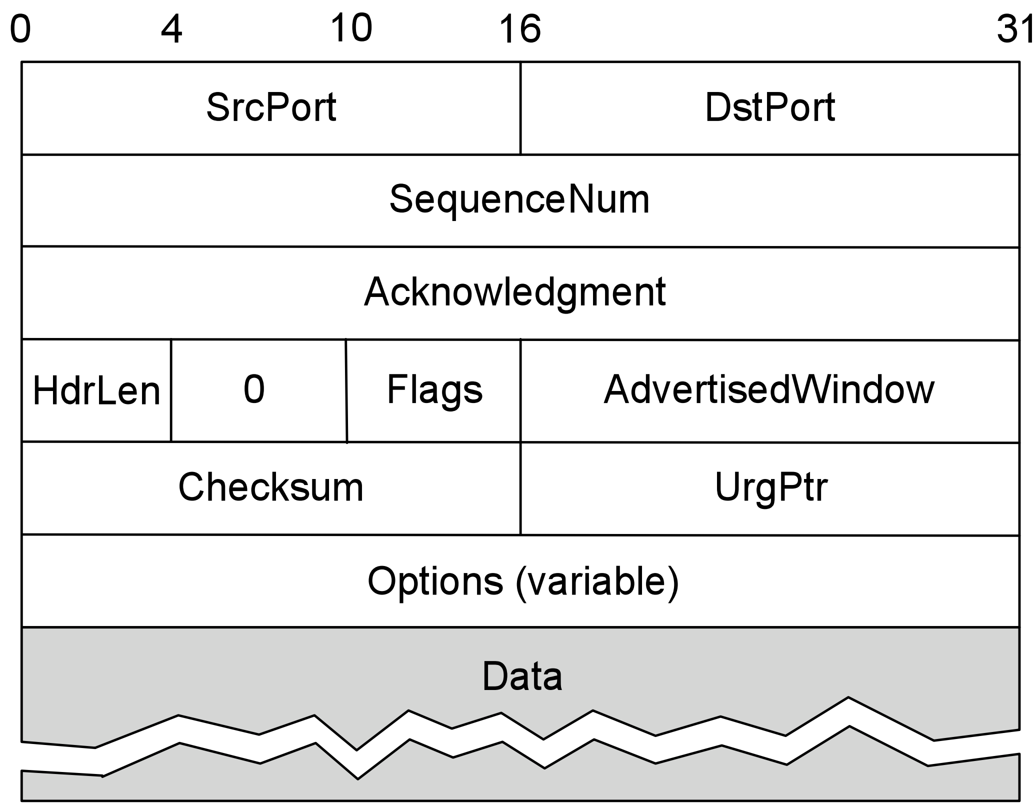
- Reliable
- Connection-oriented
- Byte-stream
  - app writes bytes
  - TCP sends *segments*
  - app reads bytes
- Full duplex
- Flow control: keep sender from overrunning receiver
- Congestion control: keep sender from overrunning network



Data Link Versus Transport

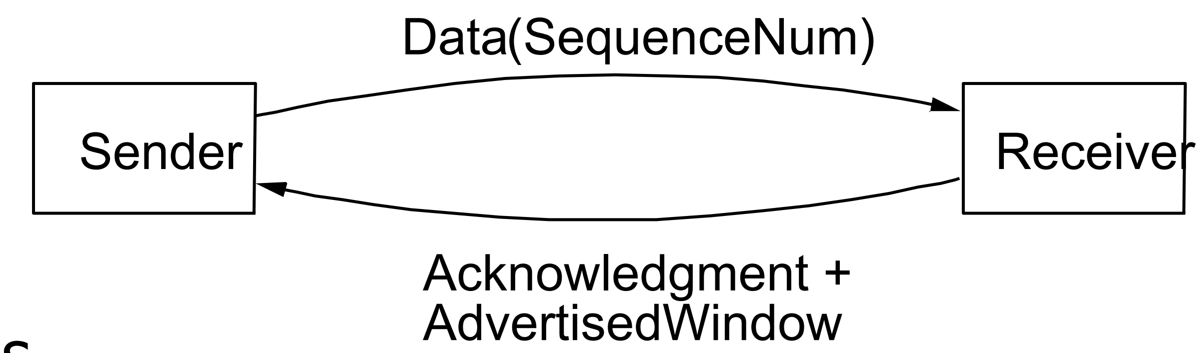
- Potentially connects many different hosts/applications
  - need explicit connection establishment and termination
- Potentially varying RTT
  - need adaptive timeout mechanism
- Potentially long delay in network
  - need to be prepared for arrival of very old packets
- Potentially varying capacity at destination
  - need to accommodate different node capacity
- Potentially varying network capacity
  - need to be prepared for network congestion

Segment Format



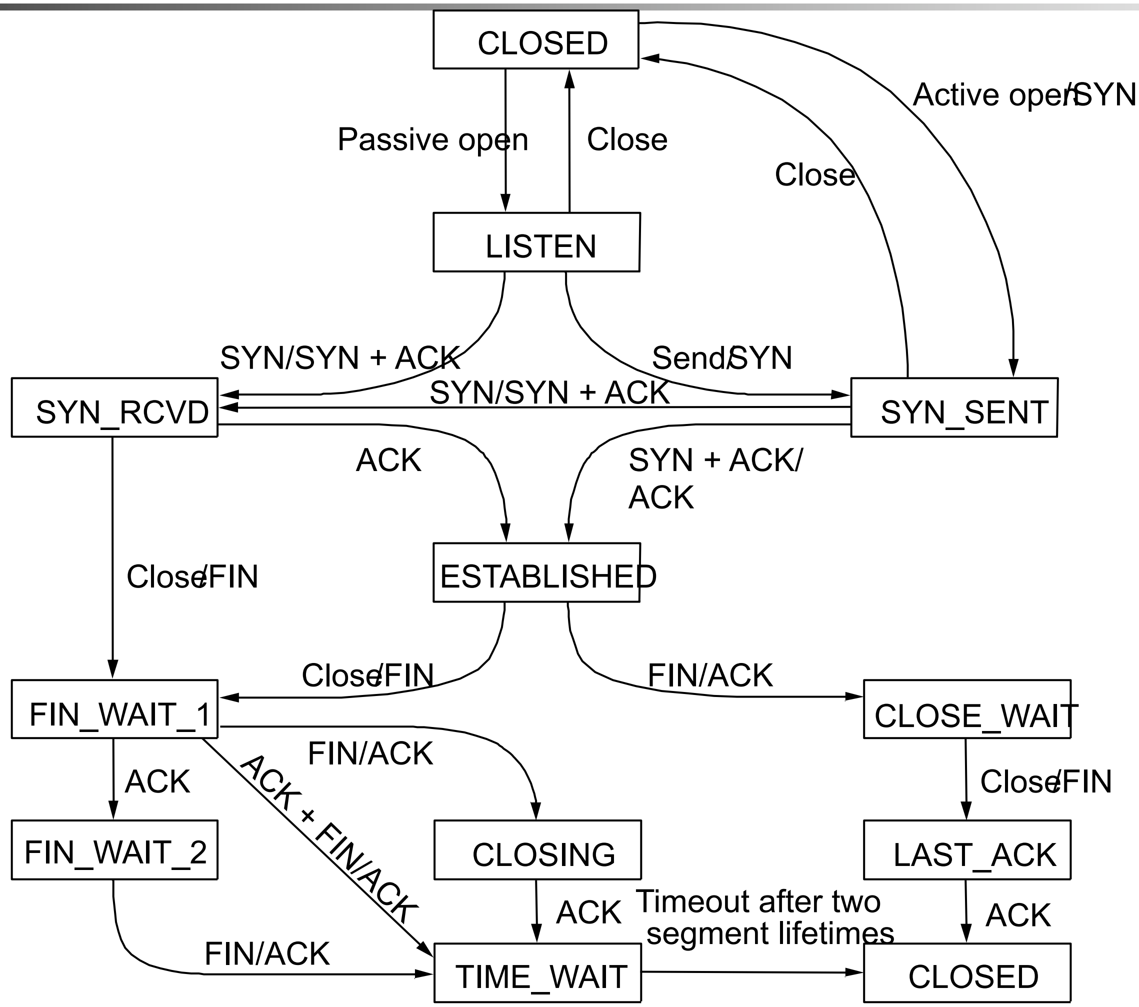
Segment Format (cont)

- Each connection identified with 4-tuple:
  - **`(SrcPort, SrcIPAddr, DsrPort, DstIPAddr)`**
- Sliding window + flow control
  - **`acknowledgment, SequenceNum, AdvertisedWindow`**



- Flags
  - **`SYN, FIN, RESET, PUSH, URG, ACK`**
- Checksum
  - pseudo header + TCP header + data

State Transition Diagram



Sliding Window in TCP

- Purpose:
  - Guarantees a reliable delivery of data (ARQ)
  - Ensures that data is delivered in order (SeqNum)
  - Enforces flow-control between sender and receiver (AdvertisedWindow field)

NAMING

Naming in the Internet

- Hosts

  `cheltenham.cs.princeton.edu` → `192.12.69.17`

  `192.12.69.17` → `80:23:A8:33:5B:9F`

- Files

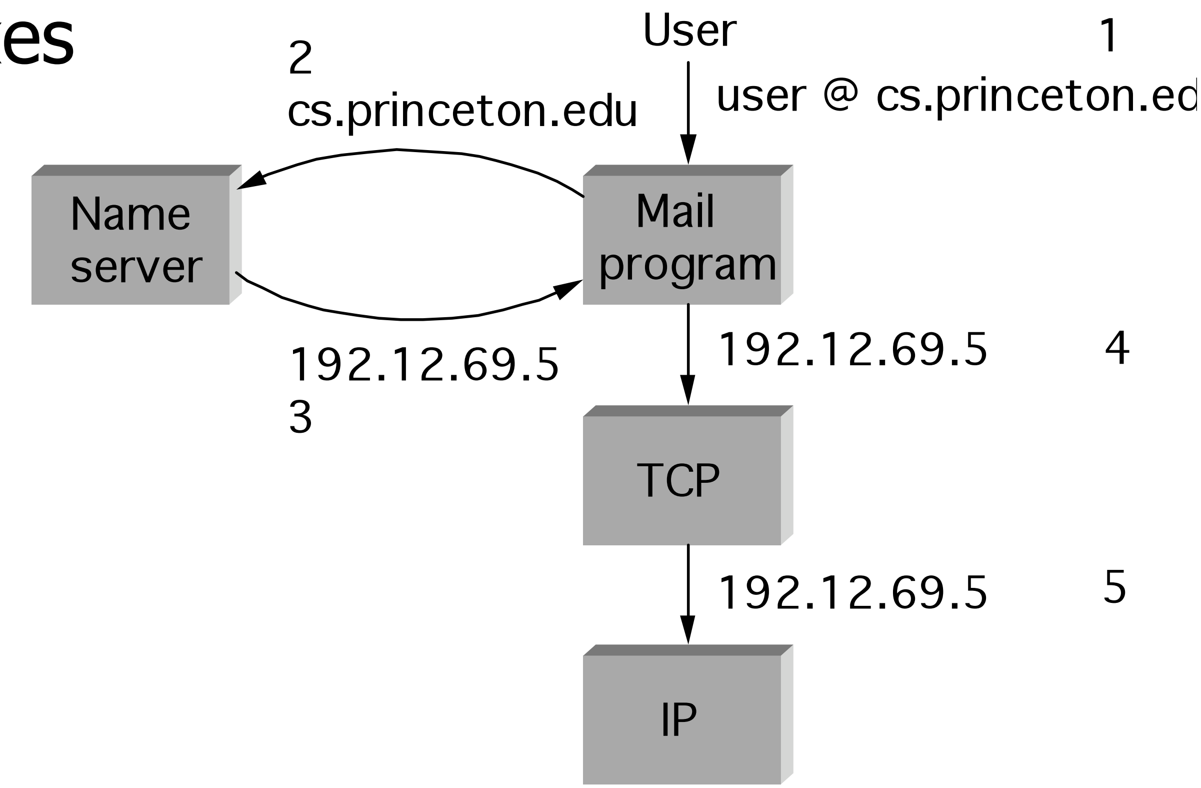
  `/usr/llp/tmp/foo` → `(server, fileid)`

- Users

  `Larry Peterson` → `llp@cs.princeton.edu`

Examples (cont)

- Mailboxes

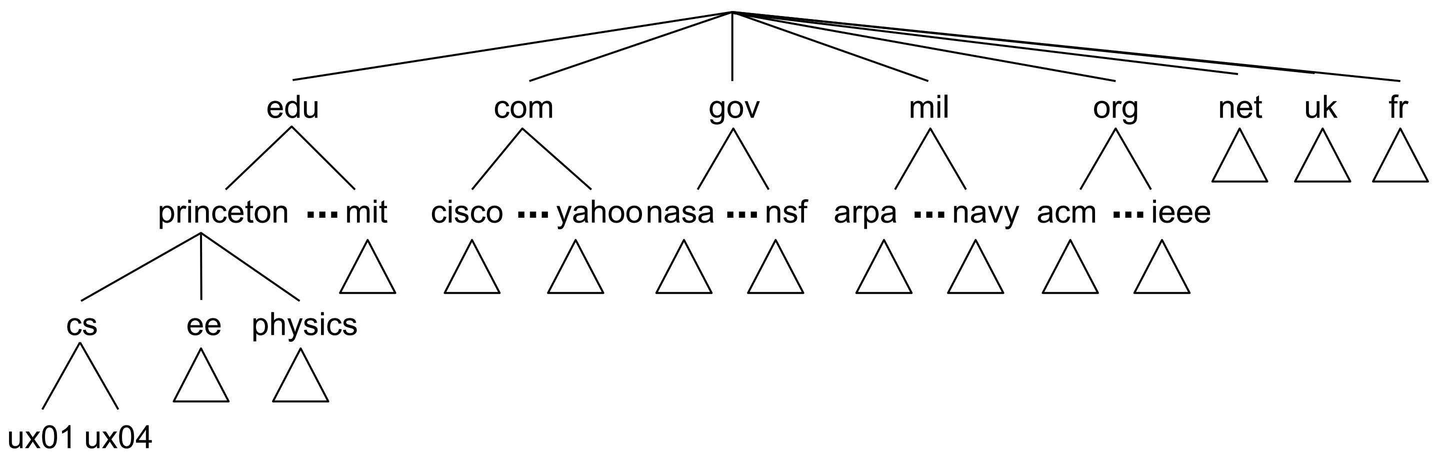


- Services

`nearby ps printer with short queue and 2MB`

Domain Naming System

- Hierarchy

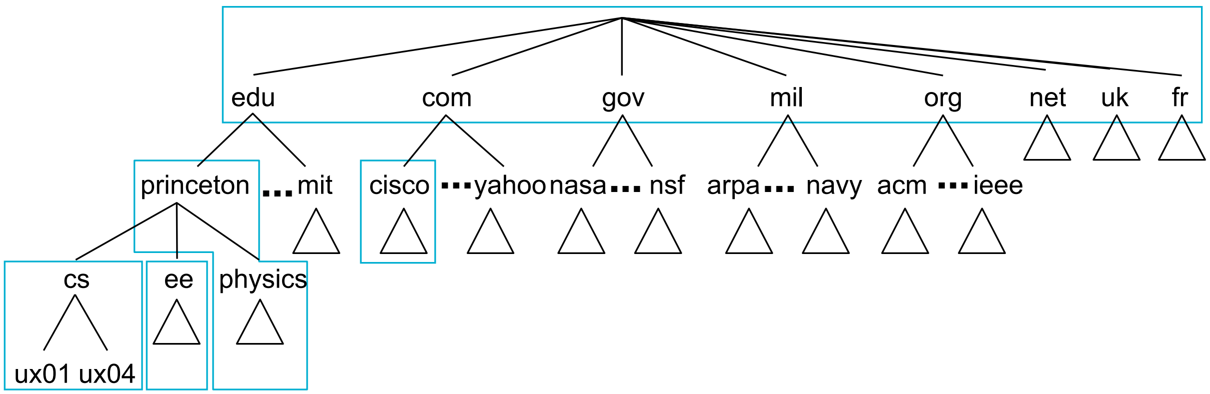


- Name

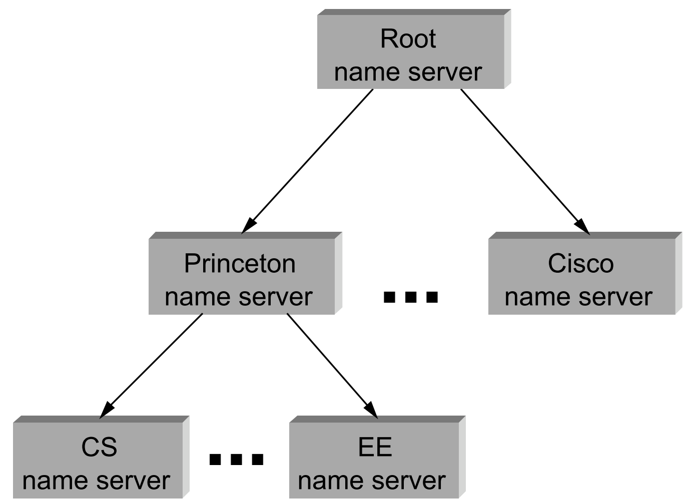
`chinstrap.cs.princeton.edu`

Name Servers

- Partition hierarchy into *zones*



- Each zone implemented by two or more *name servers*



Resource Records

- Each name server maintains a collection of *resource records*

  `(Name, Value, Type, Class, TTL)`
- Name/Value: not necessarily host names to IP addresses
- Type
  - A: Value is an IP address
  - NS: Value gives domain name for host running name server that knows how to resolve names within specified domain.
  - CNAME: Value gives canonical name for particle host; used to define aliases.
  - MX: Value gives domain name for host running mail server that accepts messages for specified domain.
- Class: allow other entities to define types
  - IN: Means Internet
- TTL: how long the resource record is valid

Root Server

```
(princeton.edu, cit.princeton.edu, NS, IN)
(cit.princeton.edu, 128.196.128.233, A, IN)

(cisco.com, thumper.cisco.com, NS, IN)
(thumper.cisco.com, 128.96.32.20, A, IN)
```

…

Princeton Server

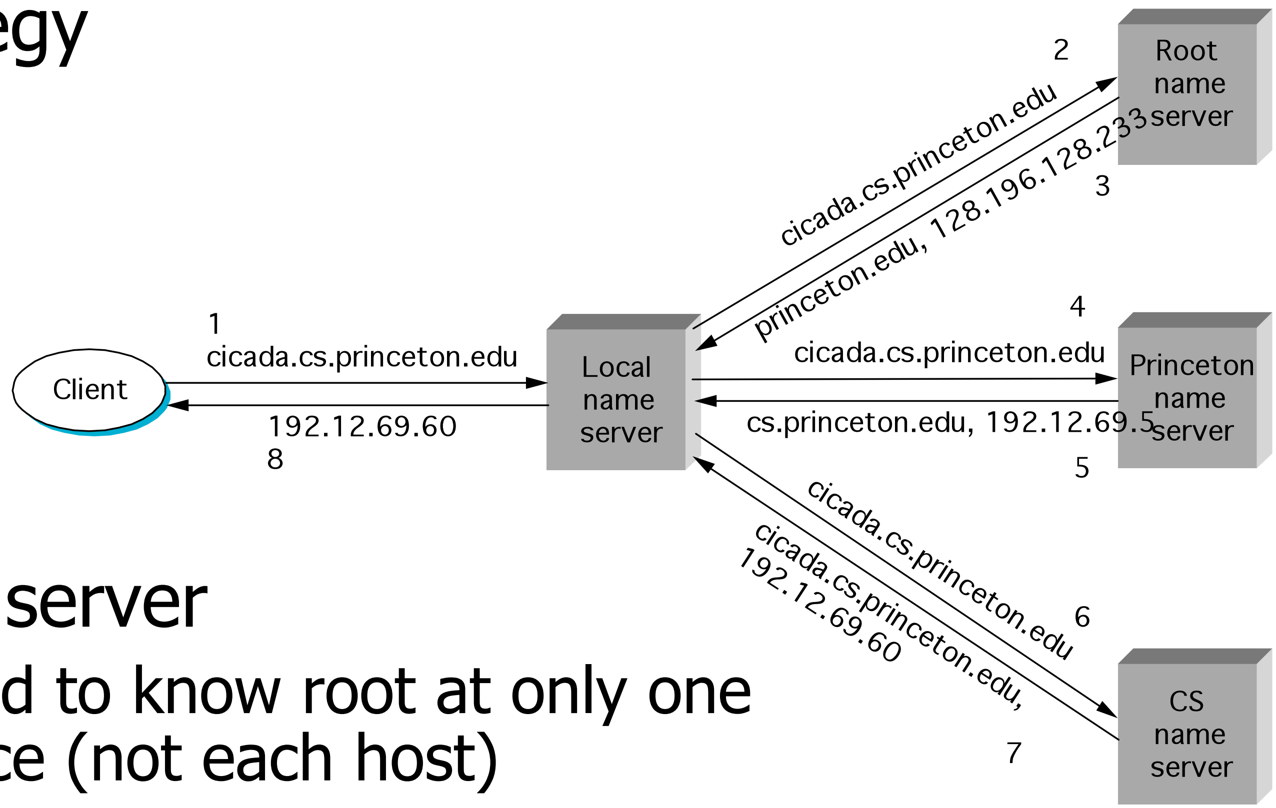
```
(cs.princeton.edu, optima.cs.princeton.edu, NS, IN)
(optima.cs.princeton.edu, 192.12.69.5, A, IN)
(ee.princeton.edu, helios.ee.princeton.edu, NS, IN)
(helios.ee.princeton.edu, 128.196.28.166, A, IN)
(jupiter.physics.princeton.edu, 128.196.4.1, A, IN)
(saturn.physics.princeton.edu, 128.196.4.2, A, IN)
(mars.physics.princeton.edu, 128.196.4.3, A, IN)
(venus.physics.princeton.edu, 128.196.4.4, A, IN)
```

CS Server

```
(cs.princeton.edu, optima.cs.princeton.edu, MX, IN)
(cheltenham.cs.princeton.edu, 192.12.69.60, A, IN)
(che.cs.princeton.edu, cheltenham.cs.princeton.edu,
  CNAME, IN)
(optima.cs.princeton.edu, 192.12.69.5, A, IN)
(opt.cs.princeton.edu, optima.cs.princeton.edu,
  CNAME, IN)
(baskerville.cs.princeton.edu, 192.12.69.35, A, IN)
(bas.cs.princeton.edu, baskerville.cs.princeton.edu,
  CNAME, IN)
```

Name Resolution

- Strategy



- Local server
  - need to know root at only one place (not each host)
  - site-wide cache

Summary

- Multi-layer stack of protocols:
  - Link Layer: ethernet (IEEE802.3), FDDI, ATM, wlan (IEEE802.11)
  - Network Layer:
    - Internet Protocol (IP) is a focal point
    - Routing protocols: RIP, OSPF, BGP-4
  - Transport Layer: UDP, TCP
  - Naming: DNS
- How do these protocols fit with each other?
- What is the syntax and semantic of typical packets (e.g., TCP, IP, UDP)
- What are the important mechanisms (e.g., TCP handshake, DNS resolution)