Networking

- Middleware gives guarantees not provided by networking
- How do you connect computers?
  - LAN
  - WAN
- Let us consider the example of the Internet

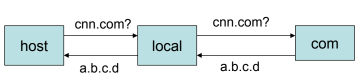
Internet: Example

- Click -> get page
- specifies
  - protocol (http)
  - location

  (www.cnn.com)

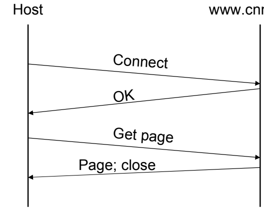
Internet: Locating Resource

- www.cnn.com
  - name of a computer
  - Implicitly also a file
- Map name to IP address
  - DNS



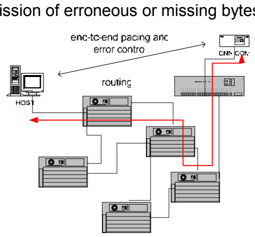
Internet: Connection

- Http sets up a connection (tcp)
  - between the host and cnn.com to transfer the page
- The connection transfers page as a byte stream
  - without errors: flow control + error control



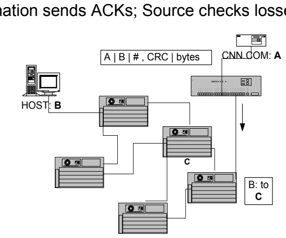
Internet: End-to-end

- Byte stream flows end to end across many links/switches:
  - routing (+ addressing)
- That stream is regulated and controlled by both ends:
  - retransmission of erroneous or missing bytes; flow control



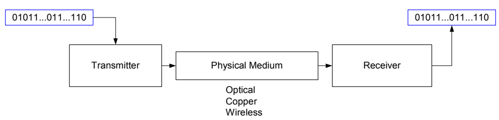
Internet: Packets

- The network transports bytes grouped into packets
- Packets are “self-contained”; routers handle them 1 by 1
- The end hosts worry about errors and pacing
  - Destination sends ACKs; Source checks losses



Internet: Bits

- Equipment in each node sends packets as string of bits
- That equipment is not aware of the meaning of the bits
- Frames (packetizing) vs. streams



Internet: Points to remember

- Separation of tasks
  - send bits on a link: transmitter/receiver [clock, modulation,…]
  - send packet on each hop [framing, error detection,…]
  - send packet end to end [addressing, routing]
  - pace transmissions [detect congestion]
  - retransmit erroneous or missing packets [acks, timeout]
  - find destination address from name [DNS]
- Scalability
  - routers don't know full path
  - names and addresses are hierarchical

Internet : Challenges

- Addressing ?
- Routing ?
- Reliable transmission ?
- Interoperability ?
- Resource management ?
- Quality of service ?

Concepts at heart of the Internet

- Protocol
- Layered Architecture
- Packet Switching
- Distributed Control
- Open System

Protocol

- Two communicating entities must agree on:
  - Expected order and meaning of messages they exchange
  - The action to perform on sending/receiving a message
- Asking the time

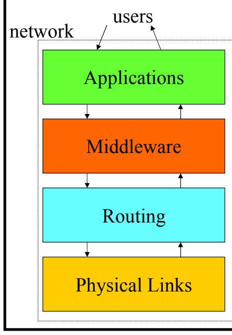
Layered Architectures

- Human beings can handle lots of complexity in their protocol processing.
  - Ambiguously defined protocols
  - Many protocols all at once
- How computers manage complex protocol processing?
  - Specify well defined protocols to enact.
  - Decompose complicated jobs into layers;
    - each has a well defined task

Layered Architectures

- Break-up design problem into smaller problems
  - More manageable
- Modular design: easy to extend/modify.
- Difficult to implement
  - careful with interaction of layers for efficiency

Layered Architecture



Web, e-mail, file transfer, ...

Reliable/ordered transmission, QOS, security, compression, ...

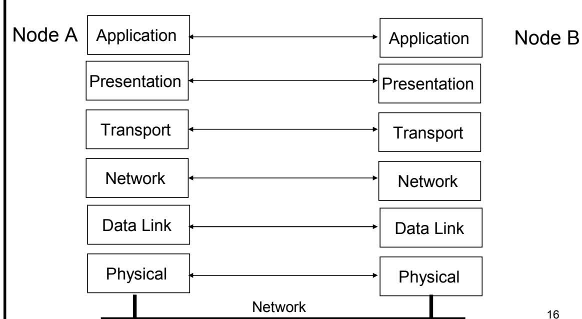
End-to-end transmission, resource allocation, routing, ...

Point-to-point links, LANs, radios, ...

The OSI Model

- Open Systems Interconnect model is a standard way of understanding conceptual layers of network comm.
- This is a model, nobody builds systems like this.
- Each level provides certain functions and guarantees, and communicates with the same level on remote notes.
- A message is generated at the highest level, and is passed down the levels, encapsulated by lower levels, until it is sent over the wire.
- On the destination, it makes its way up the layers,until the high-level msg reaches its high-level destination.

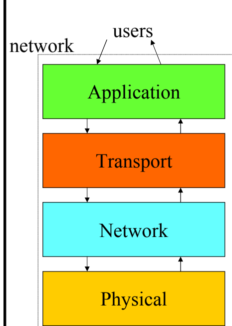
OSI Levels



OSI Levels

- Physical Layer: electrical details of bits on the wire
- Data Link: sending “frames” of bits and error detection
- Network Layer:” routing packets to the destination
- Transport Layer: reliable transmission of messages, disassembly/assembly, ordering, retransmission of lost packets
- Session Layer; really part of transport, typ. Not impl.
- Presentation Layer: data representation in the message
- Application: high-level protocols (mail, ftp, etc.)

Internet protocol stack



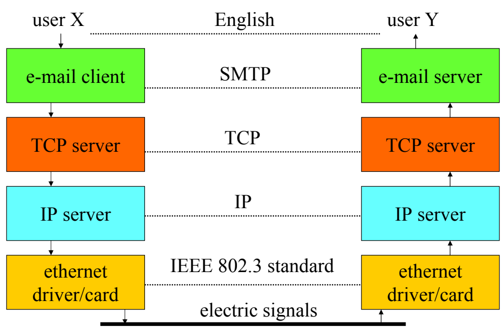
HTTP, SMTP, FTP, TELNET, DNS, …

TCP, UDP.

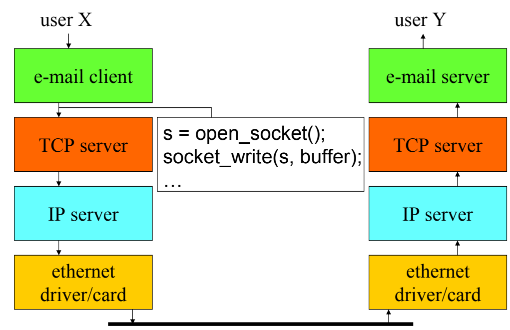
IP

Point-to-point links, LANs, radios, ...

Protocol stack



Protocol interfaces



Addressing

- Each network interface has a hardware address
  - Multiple interfaces ⇒ multiple addresses
- Each application communicates via a *port*
  - Port is a logical connection endpoint
  - Allows multiple local applications to use network resources
  - Up to 65535
    - < 1024 : used by privileged applications
    - 1024 ≤ available for use ≤ 49151
    - 49152 ≤ Dynamic ports/private ports ≤ 65535
  - http ports 80 and 8080
  - telnet 23, ftp 21, etc
- Think of a telephone network …

Addressing and Packet Format

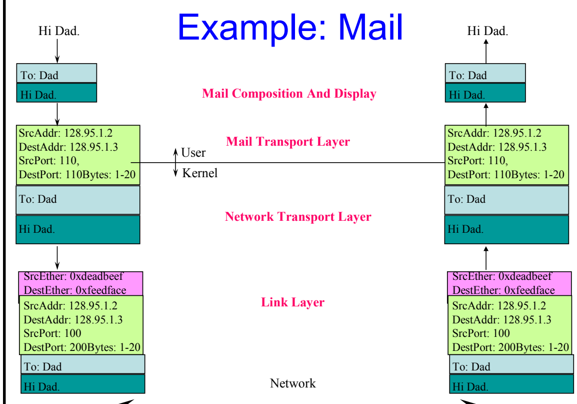
- The ``Data'' segment contains higher level protocol information.
  - Which protocol is this packet destined for?
  - Which process is the packet destined for?
  - Which packet is this in a sequence of packets?
  - What kind of packet is this?
- This is the stuff of the OSI reference model.

Start (7 bytes)
Destination (6)
Source (6)
Length (2)
Msg Data (1500)
Checksum (4)

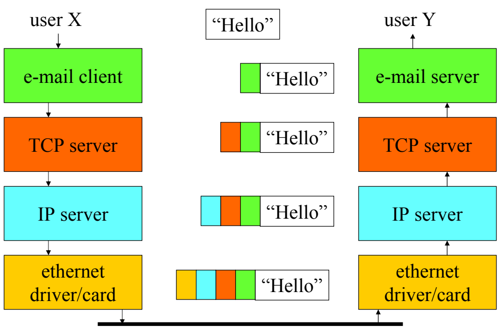
Ethernet packet dispatching

- An incoming packet comes into the Ethernet controller.
- The Ethernet controller reads it off the network into a buffer.
- It interrupts the CPU.
- A network interrupt handler reads the packet out of the controller into memory.
- A dispatch routine looks at the Data part and hands it to a higher level protocol
- The higher level protocol copies it out into user space.
- A program manipulates the data.
- The output path is similar.
- Consider what happens when you send mail.

Example: Mail



Protocol encapsulation



End-to-End Argument

- What function to implement in each layer?
- Saltzer, Reed, Clarke 1984
  - A function can be correctly and completely implemented only with the knowledge and help of applications standing at the communication endpoints
  - Argues for moving function upward in a layered architecture
- Should the network guarantee packet delivery ?
  - Think about a file transfer program
  - Read file from disk, send it, the receiver reads packets and writes them to the disk

End-to-End Argument

- If the network guaranteed packet delivery
  - one might think that the applications would be simpler
    - No need to worry about retransmits
  - But need to check that file was written to the remote disk intact
- A check is necessary if nodes can fail
  - Consequently, applications need to perform their retransmits
- No need to burden the internals of the network with properties that can, and must, be implemented at the periphery

End-to-End Argument

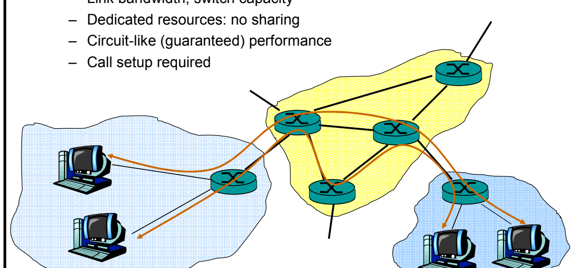
- An Occam's razor for Internet design
  - If there is a problem, the simplest explanation is probably the correct one
- Application-specific properties are best provided by the applications, not the network
  - Guaranteed, or ordered, packet delivery, duplicate suppression, security, etc.
- The internet performs the simplest packet routing and delivery service it can
  - Packets are sent on a best-effort basis
  - Higher-level applications do the rest

Two ways to handle networking

- Circuit Switching
  - What you get when you make a phone call
  - Dedicated circuit per call
- Packet Switching
  - What you get when you send a bunch of letters
  - Network bandwidth consumed only when sending
  - Packets are routed independently
- Message Switching
  - It's just packet switching, but routers perform store-and-forward

Circuit Switching

- End-to-end resources reserved for "call"
  - Link bandwidth, switch capacity
  - Dedicated resources: no sharing
  - Circuit-like (guaranteed) performance
  - Call setup required



Packet Switching

- Each end-to-end data stream divided into packets
  - User's packets *share* network resources
    - Compared to dedicated allocation
  - Each packet uses full link bandwidth
    - Compared to dividing bandwidth into pieces
  - Resources are used as needed
    - Compared to resource reservation
- Resource contention:
  - Aggregate demand can exceed amount available
  - Congestion: packets queue, wait for link use
  - Store and forward: packets move one hop at a time
    - Transmit over link
    - Wait turn at next link

Routing

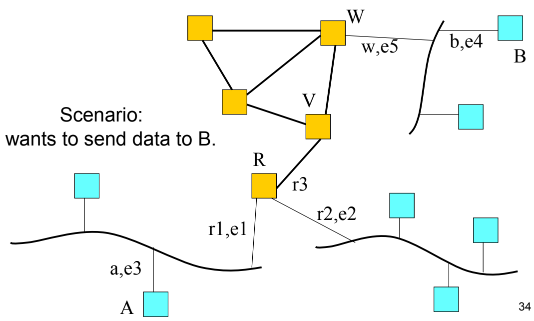
- Goal: move data among routers from source to dest.
- Datagram packet network:
  - Destination address determines next hop
  - Routes may change during session
  - Analogy: driving, asking directions
  - No notion of call state
- Circuit-switched network:
  - Call allocated time slots of bandwidth at each link
  - Fixed path (for call) determined at call setup
  - Switches maintain lots of per call state: resource allocation

Packet vs. Circuit Switching

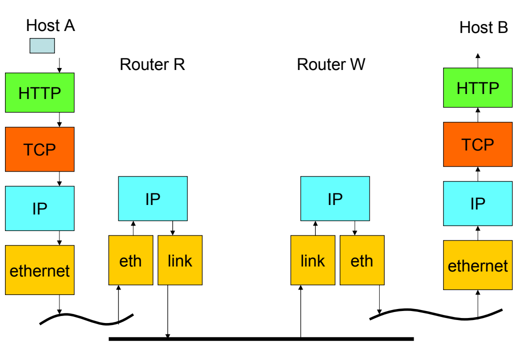
- Reliability: no congestion, in-order data in circuit-switch
- Packet switching: better bandwidth use
- State, resources: packet switching has less state
  - Good: less control plane processing resources along the way
  - More data plane (address lookup) processing
- Failure modes (routers/links down)
  - Packet switch reconfigures sub-second timescale
  - Circuit switching: more complicated
    - Involves all switches in the path

A small Internet

Scenario:
A wants to send data to B.



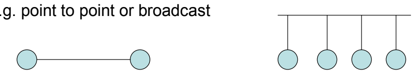
Packet forwarding



The Link Layer

What is purpose of this layer?

- Physically encode bits on the wire
- Link = pipe to send information
  - E.g. point to point or broadcast



- Can be built out of:
  - Twisted pair, coaxial cable, optical fiber, radio waves, etc
- Links should only be able to send data
  - Could corrupt, lose, reorder, duplicate, (fail in other ways)

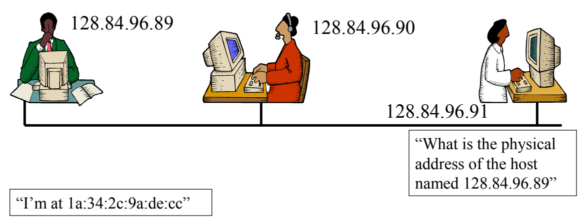
How to connect routers/machines?

- WAN/Router Connections
  - Commercial:
    - T1 (1.5 Mbps), T3 (44 Mbps)
    - OC1 (51 Mbps), OC3 (155 Mbps)
    - ISDN (64 Kbps)
    - Frame Relay (1-100 Mbps, usually 1.5 Mbps)
    - ATM (some Gbps)
  - To your home:
    - DSL
    - Cable
- Local Area:
  - Ethernet: IEEE 802.3 (10 Mbps, 100 Mbps, 1 Gbps)
  - Wireless: IEEE 802.11 b/g/a (11 Mbps, 22 Mbps, 54 Mbps)

Link level Issues

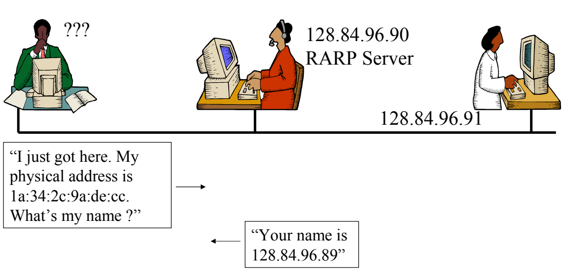
- Encoding: map bits to analog signals
- Framing: Group bits into frames (packets)
- Arbitration: multiple senders, one resource
- Addressing: multiple receivers, one wire

Addressing & ARP



- ARP is used to discover physical addresses
  - ARP = Address Resolution Protocol

Addressing & RARP

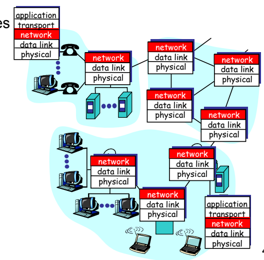


- RARP is used to discover virtual addresses
  - RARP = Reverse Address Resolution Protocol

The Network Layer

Purpose of Network layer

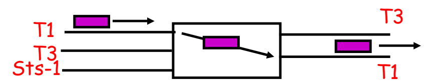
- Given a packet, send it across the network to destination
- 2 key issues:
  - Portability:
    - connect different technologies
  - Scalability
    - To the Internet scale



What does it involve?

Two important functions:

- *routing:* determine path from source to dest.
- *forwarding:* move packets from router's input to output



Network service model

Q: What *service model* for "channel" transporting packets from sender to receiver?

service abstraction

- guaranteed bandwidth?
- preservation of inter-packet timing (no jitter)?
- loss-free delivery?
- in-order delivery?
- congestion feedback to sender?

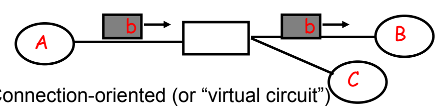
The most important abstraction provided by network layer:

virtual circuit or datagram?

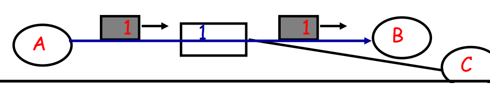
Which things can be "faked" at the transport layer?

Two connection models

- Connectionless (or "datagram"):
  - each packet contains enough information that routers can decide how to get it to its final destination

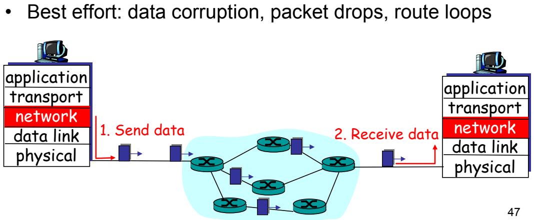


- Connection-oriented (or "virtual circuit")
  - first set up a connection between two nodes
  - label it (called a virtual circuit identifier (VCI))
  - all packets carry label



Datagram networks

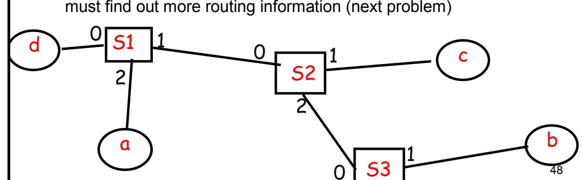
- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths
- Best effort: data corruption, packet drops, route loops



Datagrams: Forwarding

How does packet get to the destination?

- switch creates a "forwarding table", mapping destinations to output port (ignores input ports)
- when a packet with a destination address in the table arrives, it pushes it out on the appropriate output port
- when a packet with a destination address not in the table arrives, it must find out more routing information (next problem)



Datagrams

- Plusses:
  - No round trip connection setup time
  - No explicit route teardown
  - No resource reservation ⇒ each flow could get max bandwidth
  - Easily handles switch failures; routes around it
- Minuses
  - Difficult to provide resource guarantees
  - Higher per packet overhead
- Internet uses datagrams: IP (Internet Protocol)

IP addressing: CIDR

- Classless InterDomain Routing
  - network portion of address of arbitrary length
  - address format: a.b.c.d/x, where x is # bits in network portion

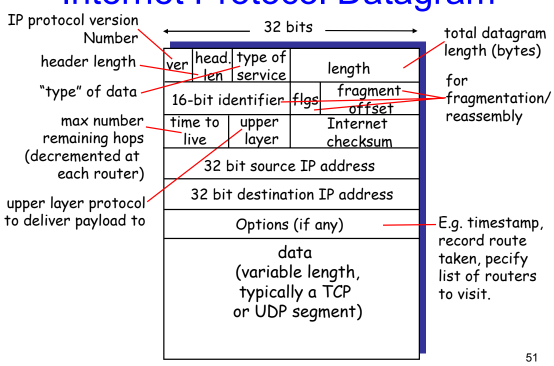
← network part → ← host part →

11001000 00010111 00010000 00000000

200.23.16.0/23

  - Examples:
    - Class A: /8
    - Class B: /16
    - Class C: /24

Internet Protocol Datagram

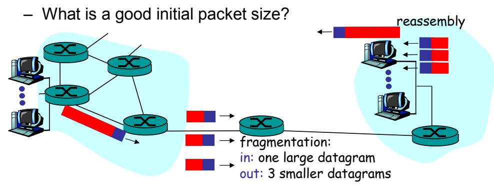


Datagram Portability

- IP Goal: To create one logical network from multiple physical networks
  - All intermediate routers should understand IP
  - IP header information sufficient to carry the packet to destination
  - Goal: Run over anything!
- Problem:
  - Physical networks have different MTUs
  - "max. transmission unit": 1500 for Ethernet, 48 for ATM
- Solution 1:
  - Fit everything in the MTU (!)

IP Fragmentation & Reassembly

- Solution 2: (the one used)
  - If packet size > MTU of network, then fragment into pieces
    - Each fragment is less than MTU size
    - Each has IP headers + frag bit set + frag id + offset
  - Packets may get refragmented on the way to destination
  - Reassembly only done at the destination
  - What is a good initial packet size?



Internet: Names and Addresses

Naming in the Internet

- What are named? All Internet Resources.
  - Objects: www.cs.cornell.edu/pages/ranveer
  - Services: weather.yahoo.com/forecast
  - Hosts: planetlab1.cs.cornell.edu
- Characteristics of Internet Names
  - human recognizable
  - unique
  - persistent
- Universal Resource Names (URNs)

Locating the resources

- Internet services and resources are provided by end-hosts
  - ex. www1.cs.cornell.edu and www2.cs.cornell.edu host Ranveer's home page.
- Names are mapped to Locations
  - Universal Resource Locators (URL)
  - Embedded in the name itself: ex. weather.yahoo.com/forecast
- Semantics of Internet naming
  - ✓ human recognizable
  - ✓ uniqueness
  - x persistent

Locating the Hosts?

- Internet Protocol Addresses (IP Addresses)
  - ex. planetlab1.cs.cornell.edu → 128.84.154.49
- Characteristics of IP Addresses
  - 32 bit fixed-length
  - enables network routers to efficiently handle packets in the Internet
- Locating services on hosts
  - port numbers (16 bit unsigned integer) 65536 ports
  - standard ports: HTTP 80, FTP 20, SSH 22, Telnet 20

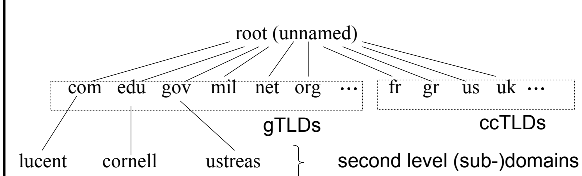
Mapping Not 1 to 1

- One host may map to more than one name
  - One server machine may be the web server (www.foo.com), mail server (mail.foo.com)etc.
- One host may have more than one IP address
  - IP addresses are per network interface
- But IP addresses are generally unique!
  - two globally visible machines should not have the same IP address
  - Anycast is an Exception:
    - routers send packets dynamically to the closest host matching an anycast address

How to get a name?

- Naming in Internet is Hierarchical
  - decreases centralization
  - improves name space management
- First, get a domain name then you are free to assign sub names in that domain
  - How to get a domain name coming up
- Example: weather.yahoo.com belongs to yahoo.com which belongs to .com
  - regulated by global non-profit bodies

Domain name structure



gTLDs= Generic Top Level Domains
ccTLDs = Country Code Top Level Domains

Top-level Domains (TLDs)

- Generic Top Level Domains (gTLDs)
  - .com - commercial organizations
  - .org - not-for-profit organizations
  - .edu - educational organizations
  - .mil - military organizations
  - .gov - governmental organizations
  - .net - network service providers
  - New: .biz, .info, .name, …
- Country code Top Level Domains (ccTLDs)
  - One for each country

How to get a domain name?

- In 1998, non-profit corporation, Internet Corporation for Assigned Names and Numbers (ICANN), was formed to assume responsibility from the US Government
- ICANN authorizes other companies to register domains in com, org and net and new gTLDs
  - Network Solutions is largest and in transitional period between US Govt and ICANN had sole authority to register domains in com, org and net

How to get an IP Address?

- Answer 1: Normally, answer is get an IP address from your upstream provider
  - This is essential to maintain efficient routing!
- Answer 2: If you need lots of IP addresses then you can acquire your own block of them.
  - IP address space is a scarce resource - must prove you have fully utilized a small block before can ask for a larger one and pay $$ (Jan 2002 - $2250/year for /20 and $18000/year for a /14)

How to get lots of IP Addresses? Internet Registries

RIPE NCC (Riseaux IP Europiens Network Coordination Centre) for Europe, Middle-East, Africa

APNIC (Asia Pacific Network Information Centre)for Asia and Pacific

ARIN (American Registry for Internet Numbers) for the Americas, the Caribbean, sub-saharan Africa

Note: Once again regional distribution is important for efficient routing!

Can also get Autonomous System Numnbers (ASNs from these registries

Are there enough addresses?

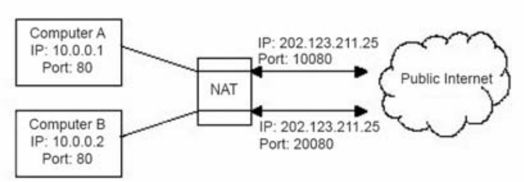
- Unfortunately No!
  - 32 bits $\rightarrow$ 4 billion unique addresses
  - but addresses are assigned in chunks
  - ex. cornell has four chunks of /16 addressed
    - ex. 128.84.0.0 to 128.84.255.255
    - 128.253.0.0, 128.84.0.0, 132.236.0.0, and 140.251.0.0
- Expanding the address space!
  - IPv6 128 bit addresses
  - difficult to deploy (requires cooperation and changes to the core of the Internet)

DHCP and NATs

- Dynamic Host Control Protocol
  - lease IP addresses for short time intervals
  - hosts may refresh addresses periodically
  - ♥ only live hosts need valid IP addresses
- Network Address Translators
  - Hide local IP addresses from rest of the world
  - only a small number of IP addresses are visible outside
  - ♥ solves address shortage for all practical purposes
  - ✣ access is highly restricted
    - ex. peer-to-peer communication is difficult

NATs in operation

- Translate addresses when packets traverse through NATs
- Use port numbers to increase number of supportable flows



DNS: Domain Name System

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function implemented as application-layer protocol
  - complexity at network's "edge"

DNS name servers

How could we provide this service? Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

- no server has all name-to-IP address mappings

Name server: process running on a host that processes DNS requests

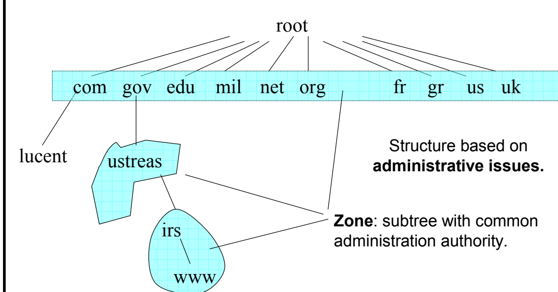
local name servers:

- each ISP, company has *local (default) name server*
- host DNS query first goes to local name server

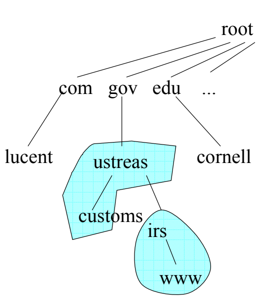
authoritative name server:

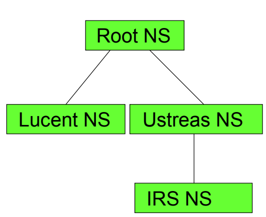
- can perform name/address translation for a specific domain or zone

Name Server Zone Structure



Name Servers (NS)





Name Servers (NS)

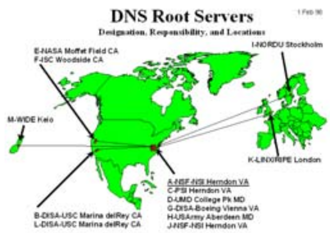
- NSs are **duplicated** for reliability.
- Each domain must have a primary and secondary.
- Anonymous ftp from:

  ftp.rs.internic.net, netinfo/root-server.txt

  gives the current root NSs (about 10).
- Each host knows the IP address of the **local** NS.
- Each NS knows the IP addresses of all root NSs.

DNS: Root name servers

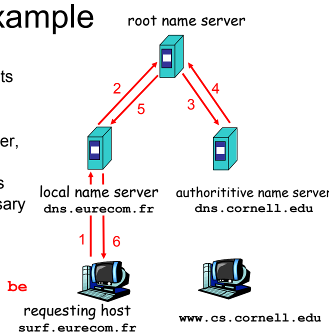
- contacted by local name server that can not resolve name
- root name server:
  - Knows the authoritative name server for main domain
- ~ 60 root name servers worldwide
  - real-world application of anycast



Simple DNS example

host **surf.eurecom.fr** wants IP address of **www.cs.cornell.edu**

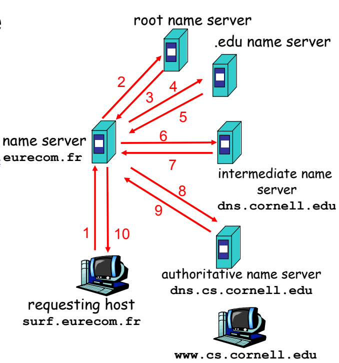
1. Contacts its local DNS server, **dns.eurecom.fr**
2. **dns.eurecom.fr** contacts root name server, if necessary
3. root name server contacts authoritative name server, **dns.cornell.edu**, if necessary **(what might be wrong with this?)**



DNS example

Root name server:

- may not know authoritative name server
- may know *intermediate name server:* who to contact to find authoritative name server



DNS Architecture

- Hierarchical Namespace Management
  - domains and sub-domains
  - distributed and localized authority
- Authoritative Nameservers
  - server mappings for specific sub-domains
  - more than one (at least two for failure resilience)
- Caching to mitigate load on root servers
  - time-to-live (ttl) used to delete expired cached mappings

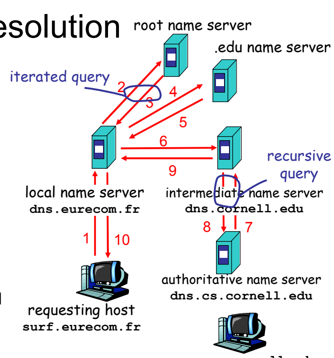
DNS: query resolution

iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"
- Takes burden off root servers

recursive query:

- puts burden of name resolution on contacted name server
- reduces latency



DNS records: More than Name to IP Address

DNS: distributed db storing resource records (RR)

RR format: `(name, value, type,ttl)`

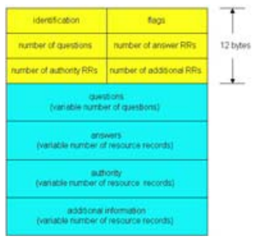
- Type=A
  - **name** is hostname
  - **value** is IP address
  - One we've been discussing; most common
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is IP address of authoritative name server for this domain
- Type=CNAME
  - **name** is an alias name for some "cannonical" (the real) name
  - **value** is cannonical name
- Type=MX
  - **value** is hostname of mailserver associated with **name**

DNS protocol, messages

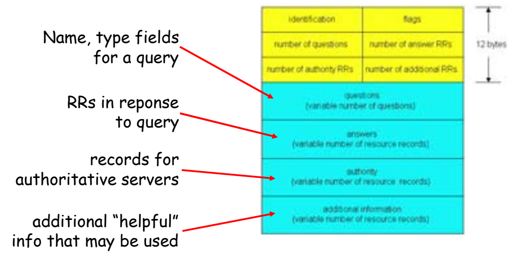
DNS protocol : *query* and *repy* messages, both with same *message format*

msg header

- identification: 16 bit # for query, repy to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative
  - reply was truncated



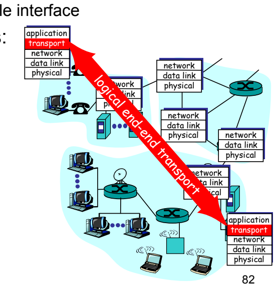
DNS protocol, messages



The Transport Layer

Purpose of this layer

- Interface end-to-end applications and protocols
  - Turn best-effort IP into a usable interface
- Data transfer b/w processes:
  - Compared to end-to-end IP
- We will look at 2:
  - TCP
  - UDP

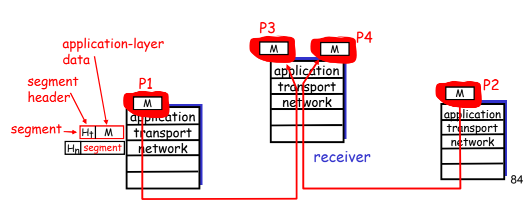


UDP

- **U**nreliable **D**atagram **P**rotocol
- Best effort data delivery between processes
  - No frills, bare bones transport protocol
  - Packet may be lost, out of order
- Connectionless protocol:
  - No handshaking between sender and receiver
  - Each UDP datagram handled independently

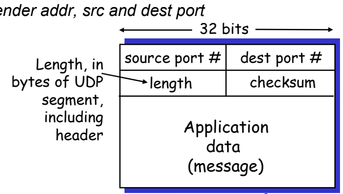
UDP Functionality

- Multiplexing/Demultiplexing
  - Using ports
- Checksums (optional)
  - Check for corruption



Multiplexing/Demultiplexing

- Multiplexing:
  - Gather data from multiple processes, envelope data with header
  - Header has src port, dest port for multiplexing
    - Why not process id?
- Demultiplexing:
  - Separate incoming data in machine to different applications
  - Demux based on *sender addr, src and dest port*



UDP segment format

Implementing Ports

- As a message queue
  - Append incoming message to the end
  - Much like a mailbox file
- If queue full, message can be discarded
- When application reads from socket
  - OS removes some bytes from the head of the queue
- If queue empty, application blocks waiting

UDP Checksum

- Over the headers and data
  - Ensures integrity end-to-end
  - 1's complement sum of segment contents
- Is optional in UDP
- If checksum is non-zero, and receiver computes another value:
  - Silently drop the packet, no error message detected

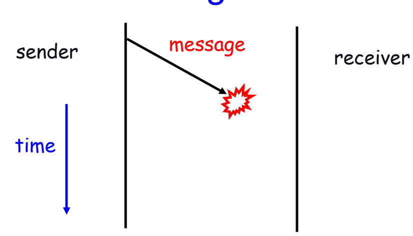
UDP Discussion

- Why UDP?
  - No delay in connection establishment
  - Simple: no connection state
  - Small header size
  - No congestion control: can blast packets
- Uses:
  - Streaming media, DNS, SNMP
  - Could add application specific error recovery

TCP

- Transmission Control Protocol
  - Reliable, in-order, process-to-process, two-way byte stream
- Different from UDP
  - Connection-oriented
  - Error recovery: Packet loss, duplication, corruption, reordering
- A number of applications require this guarantee
  - Web browsers use TCP

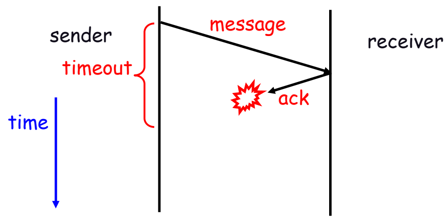
Handling Packet Loss



There are a number of reasons why the packet may get lost:
- router congestion, lossy medium, etc.

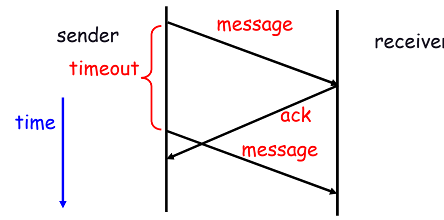
How does sender know of a successful packet send?

Lost Acks



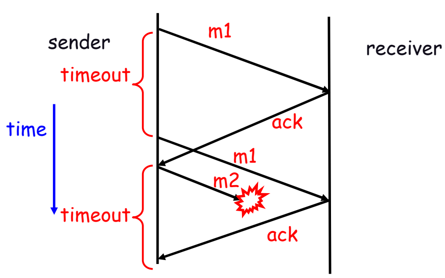
What if packet/ack is lost?

Delayed ACKs



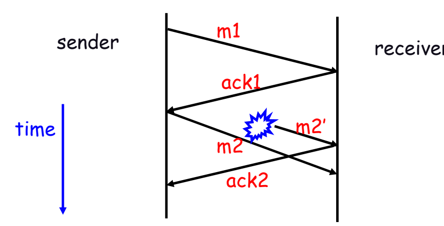
What will happen here? Due to congestion, small timeout, …
Delayed ACKs ⇒ duplicate packets

Delayed ACKs



How to solve this scenario?

Insertion of Packets

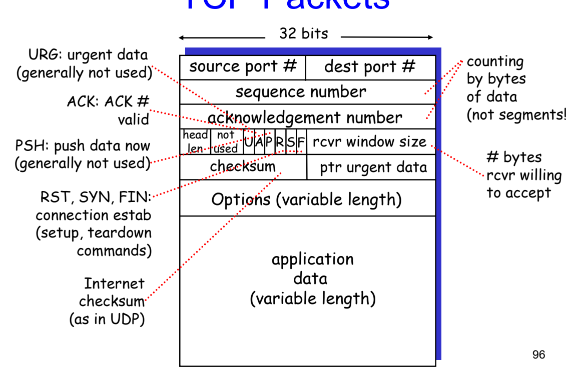


m2' could be from an old expired session!

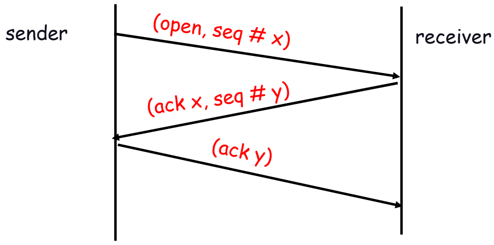
Message Identifiers

- Each message has <message id, session id>
  - Message id: uniquely identifies message in sender
  - Session id: unique across sessions
- Message ids detect duplication, reordering
- Session ids detect packet from old sessions
- TCP's sequence number has similar functionality:
  - Initial number chosen randomly
  - Unique across packets
  - Incremented by length of data bytes

TCP Packets

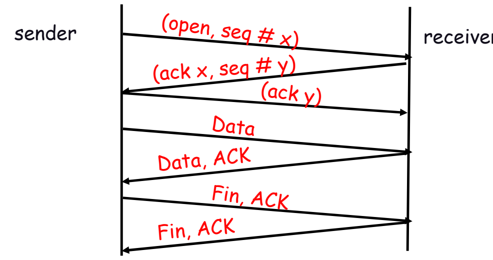


TCP Connection Establishment



TCP is connection-oriented. Starts with a 3-way handshake. Protects against duplicate SYN packets.

TCP Usage



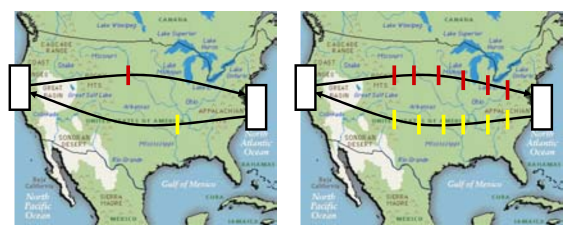
TCP timeouts

- What is a good timeout period ?
  - Want to improve throughput without unnecessary transmissions

$$\text{NewAverageRTT} = (1 - \alpha)\ \text{OldAverageRTT} + \alpha\ \text{LatestRTT}$$
$$\text{NewAverageDev} = (1 - \alpha)\ \text{OldAverageDev} + \alpha\ \text{LatestDev}$$
where $\text{LatestRTT} = (\text{ack\_receive\_time} - \text{send\_time})$,
$\text{LatestDev} = |\text{LatestRTT} - \text{AverageRTT}|$,
$\alpha = 1/8$, typically.
$$\text{Timeout} = \text{AverageRTT} + 4 * \text{AverageDev}$$

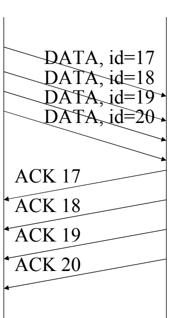
- Timeout is thus a function of RTT and deviation

TCP Windows



- Multiple outstanding packets can increase throughput

TCP Windows

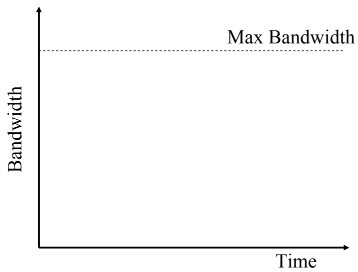


- Can have more than one packet in transit
- Especially over fat pipes, e.g. satellite connection
- Need to keep track of all packets within the window
- Need to adjust window size

TCP Congestion Control

- TCP increases its window size when no packets dropped
- It halves the window size when a packet drop occurs
  - A packet drop is evident from the acknowledgements
- Therefore, it slowly builds to the max bandwidth, and hover around the max
  - It doesn't achieve the max possible though
  - Instead, it shares the bandwidth well with other TCP connections
- This linear-increase, exponential backoff in the face of congestion is termed *TCP-friendliness*

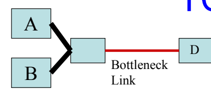
TCP Window Size

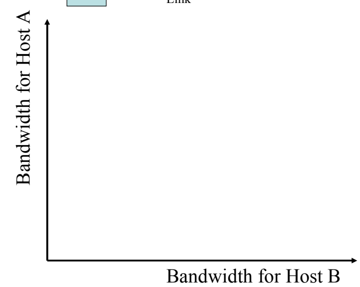


- Linear increase
- Exponential backoff

- Assuming no other losses in the network except those due to bandwidth

TCP Fairness



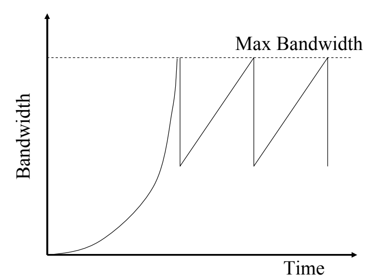


- Want to share the bottleneck link fairly between two flows

TCP Slow Start

- Linear increase takes a long time to build up a window size that matches the link bandwidth*delay
- Most file transactions are not long enough
- Consequently, TCP can spend a lot of time with small windows, never getting the chance to reach a sufficiently large window size
- Fix: Allow TCP to build up to a large window size initially by doubling the window size until first loss

TCP Slow Start



- Initial phase of exponential increase

- Assuming no other losses in the network except those due to bandwidth

TCP Summary

- Reliable ordered message delivery
  - Connection oriented, 3-way handshake
- Transmission window for better throughput
  - Timeouts based on link parameters
- Congestion control
  - Linear increase, exponential backoff
- Fast adaptation
  - Exponential increase in the initial phase