Midterm Review

EE 122: Intro to Communication Networks

Fall 2007 (WF 4-5:30 in **Cory 277**)

Vern Paxson

TAs: Lisa Fowler, Daniel Killebrew & Jorge Ortiz

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, and colleagues at Princeton and UC Berkeley

Announcements / Pending Questions

- Homework #2 due 11AM Weds Oct 10
  - No late assignments accepted!
- No lecture on Weds Oct 10
- Additional office hours for me: Fri Oct 12 1-2:30PM
  - And by appointment next Monday
  - No office hours Weds Oct 10

Moving From Switches to Routers

- Advantages of switches over routers
  - Plug-and-play
  - Fast filtering and forwarding of frames
- Disadvantages of switches over routers
  - Topology restricted to a spanning tree
  - Large networks require large ARP tables
  - Broadcast storms can cause the network to collapse
  - Can't accommodate non-Ethernet segments (why not?)

3

Comparing Hubs, Switches & Routers

	hubs	switches	routers
traffic isolation	no	yes	yes
plug & play	yes	yes	no
optimized routing	no	no	yes
cut-through	yes	yes	no

4

Midterm Review

- In-class next Friday
- Closed book
- You can have one regular-sized (8.5”x11”) sheet of paper with notes on both sides
- No PDAs, calculators, electronic/Internet gadgets, smart cell phones, etc.
- No Blue Books - all answers on exam sheets
- Ensure legibility (pencil + eraser)

5

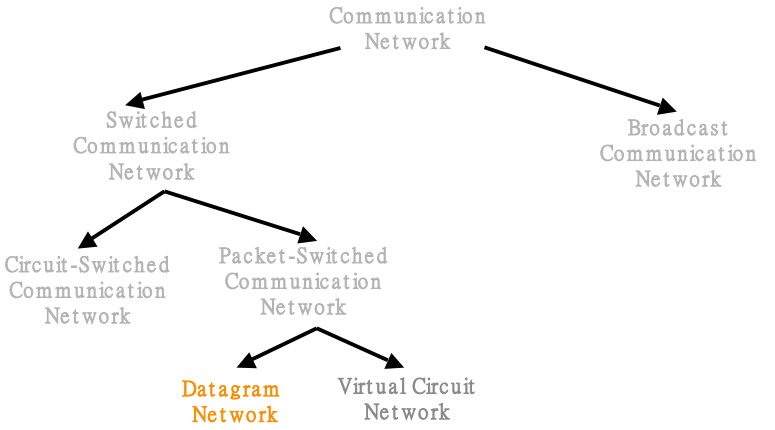
Fundamental Challenges for Networking

- Speed-of-light
- Desiring a pervasive global network (*scaling*)
- Need for it to work efficiently/cheaply
- Failure of components
- Enormous dynamic range
  - “no such thing as typical”
- Disparate parties must work together
- Rapid growth/evolution
- Crooks & other bad guys

6

Taxonomy of Communication Networks

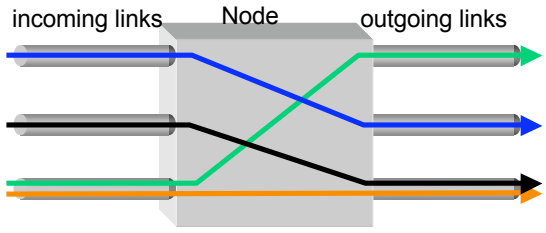
- Communication networks can be classified based on the way in which the nodes exchange information:



7

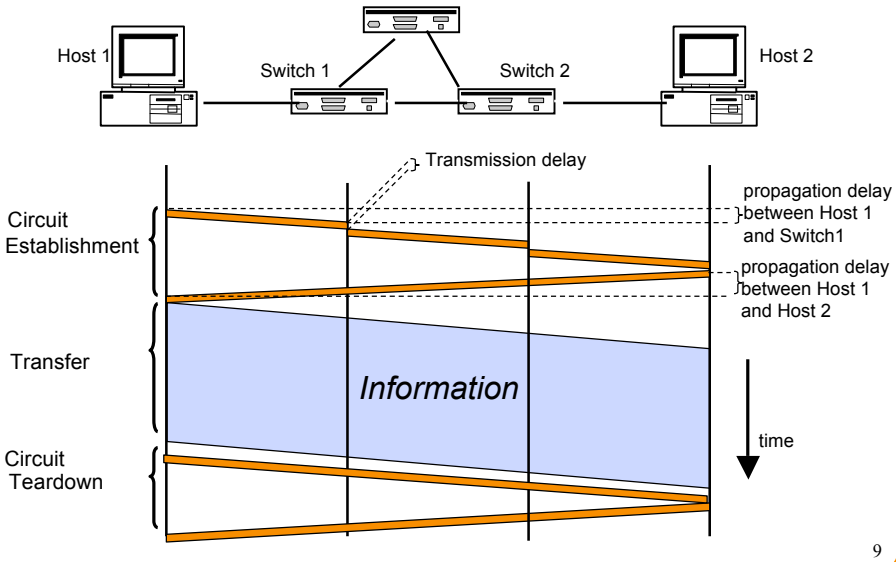
Circuit Switching (e.g., Phone Network)

- Establish: source creates circuit to destination
  - Nodes along the path store connection info
  - Nodes generally reserve resources for the connection
  - If circuit not available: “Busy signal”
- Transfer: source sends data over the circuit
  - No destination address, since nodes know path
- Teardown: source tears down circuit when done



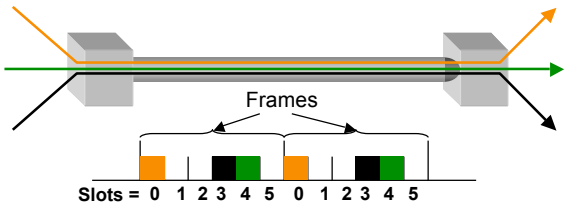
8

Timing in Circuit Switching



9

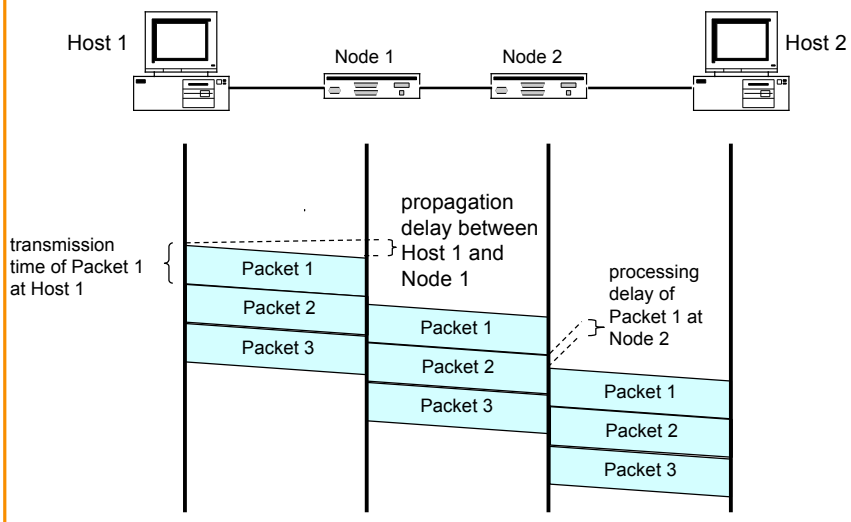
Time-Division Multiplexing/Demultiplexing



- Time divided into frames; frames into slots
- Relative slot position inside a frame determines to which conversation data belongs
  - E.g., slot 0 belongs to orange conversation
- Requires synchronization between sender and receiver—surprisingly non-trivial!
- In case of non-permanent conversations
  - Need to dynamically bind a slot to a conversation
  - How to do this?
- If a conversation does not use its circuit **the capacity is lost!**

10

Timing of Datagram Packet Switching



11

Packet-Switching vs. Circuit-Switching

- Critical advantage of packet-switching over circuit switching: Exploitation of statistical multiplexing
- Another: since routers don't know about individual conversations, when a router or link fails, it's: Easy to fail over to a different path
- A third: easier for different parties to link their networks together because they're **not** promising to reserve resources for one another
- However, packet-switching must handle congestion:
  - More complex routers
  - Harder to provide good network services (e.g., delay and bandwidth guarantees)
- In practice, sometimes combined, e.g., IP *over* SONET

12

Protocol Standardization

- Ensure communicating hosts speak the same protocol
  - Standardization to enable multiple implementations
  - Or, the same folks have to write all the software
- Standardization: Internet Engineering Task Force
  - Based on working groups that focus on specific issues
  - Produces "Request For Comments" (RFCs)
    - Promoted to standards via rough consensus and running code
  - IETF Web site is **http://www.ietf.org**
  - RFCs archived at **http://www.rfc-editor.org** (per Homework #1)
- De facto standards: same folks writing the code
  - P2P file sharing, Skype, <your protocol here>…

13

Layering: A Modular Approach

- Partition the system
  - Each layer solely relies on services from layer below
  - Each layer solely exports services to layer above
- Interface between layers defines interaction
  - Hides implementation details
  - Layers can change without disturbing other layers

14

Drawbacks of Layering

- Layer N may duplicate lower level functionality
  - E.g., error recovery to retransmit lost data
- Layers may need same information
  - E.g., timestamps, maximum transmission unit size
- Layering can hurt performance
  - E.g., hiding details about what is really going on
- Some layers are not always cleanly separated
  - Inter-layer dependencies for performance reasons
  - Some dependencies in standards (header checksums)
- Headers start to get really big
  - Sometimes header bytes >> actual content

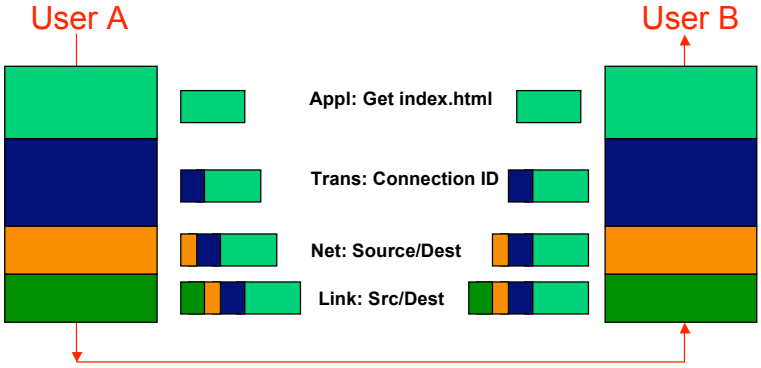
15

Layer Violations

- Sometimes the gains from not respecting layer boundaries are too great to resist
- Can occur with higher-layer entity inspecting lower-layer information:
  - E.g., TCP-over-wireless system that monitors wireless link-layer information to try to determine whether packet loss due to congestion or corruption
- Can occur with lower-layer entity inspecting higher-layer information
  - E.g., firewalls, NATs (network address translators), "transparent proxies"
- Just as with in-line assembly code, can be **messy** and paint yourself into a corner (you know too much)

16

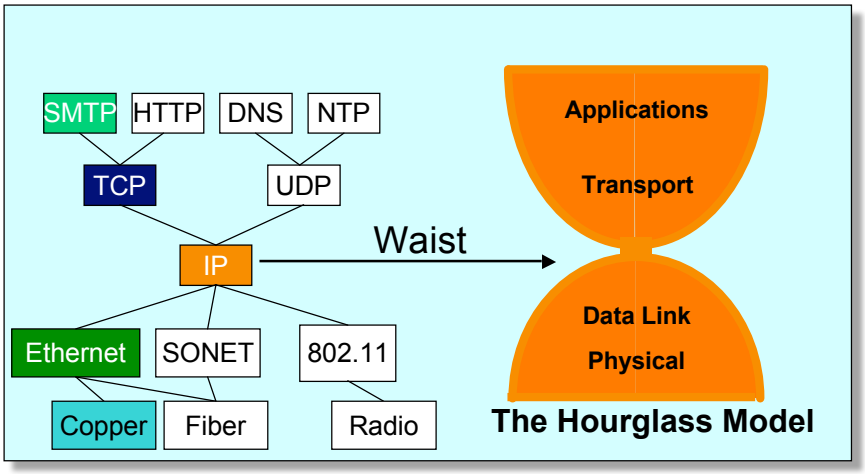
Layer Encapsulation



Common case: 20 bytes TCP header + 20 bytes IP header + 14 bytes Ethernet header = *54 bytes overhead*

17

The Internet *Hourglass*



There is just one network-layer protocol, **IP**.
The "narrow waist" facilitates interoperability.

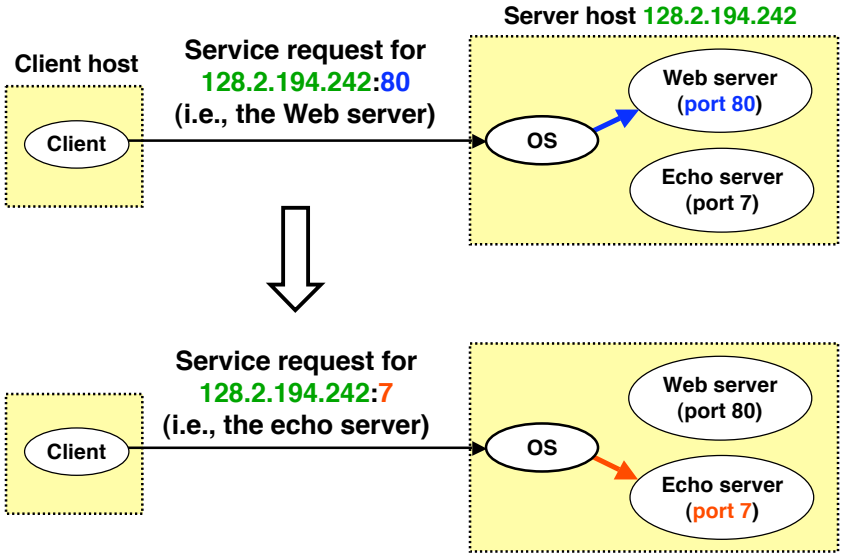
18

End-to-End Principle (Moderate Interpretation)

- Think twice before implementing functionality in the network
- If hosts can implement functionality correctly, implement it in a lower layer only as a performance enhancement
- But do so only if it does not impose burden on applications that do not require that functionality

19

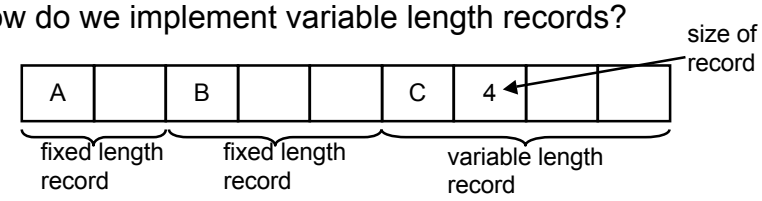
Using Ports to Identify Services



20

Recovering message boundaries

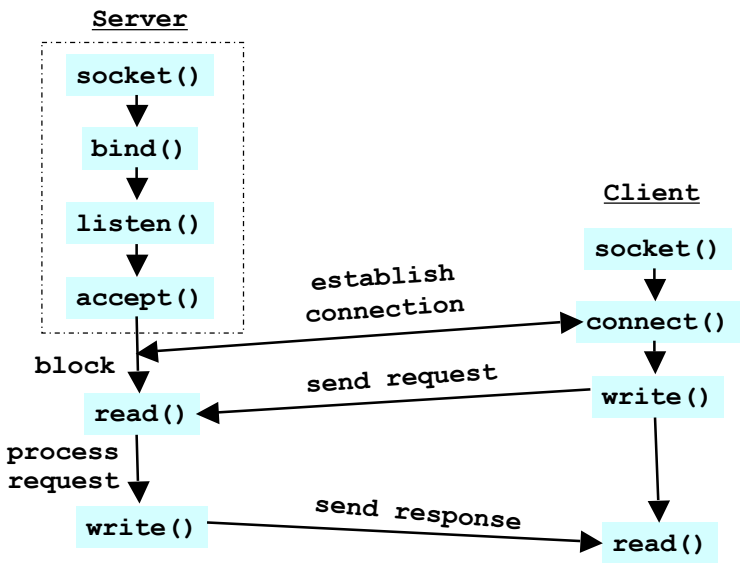
- Stream socket data separation:
  - Use records (data structures) to partition data stream
  - How do we implement variable length records?



  - What if field containing record size gets corrupted?
    - Not possible! Why?
- Structuring the byte stream so you can recover the original data boundaries is termed **framing**

21

Putting it All Together



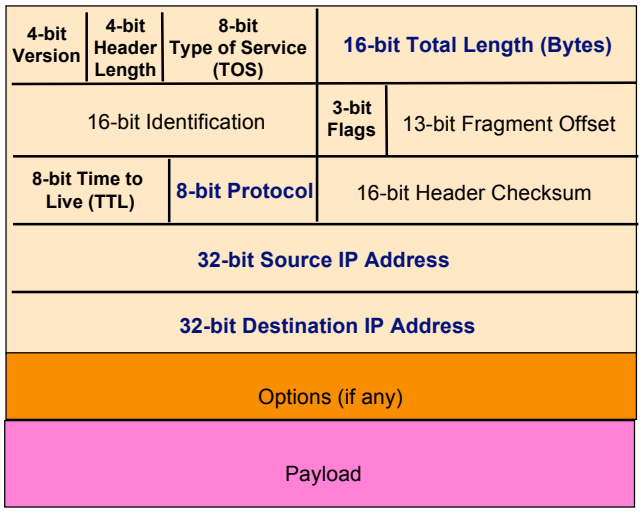
22

IP Service: "Best Effort" Suffices

- No error detection or correction
  - Higher-level protocol can provide error checking
- Successive packets may not follow the same path
  - Not a problem as long as packets reach the destination
- Packets can be delivered out-of-order
  - Receiver can put packets back in order (if necessary)
- Packets may be lost or arbitrarily delayed
  - Sender can send the packets again (if desired)
- No network congestion control (beyond "drop")
  - Sender can slow down in response to loss or delay

23

IP Packet Structure

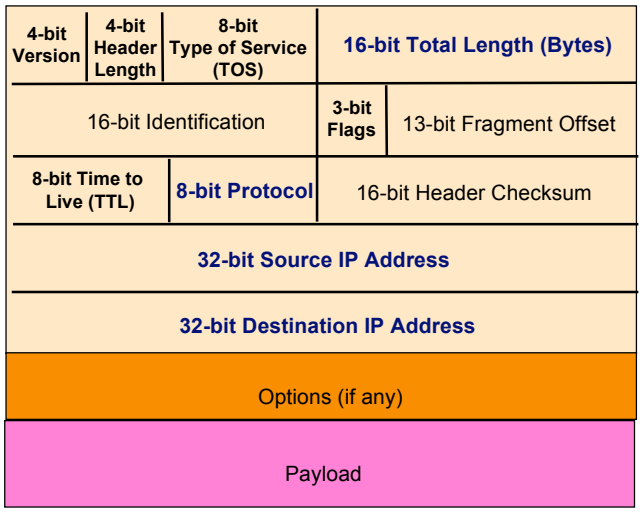


Fragmentation

- Identifier (16 bits): used to tell which fragments belong together
- Flags (3 bits):
  - Reserved **(RF):** unused bit (why "reserved"?)
  - Don't Fragment **(DF):** instruct routers to **not** fragment the packet even if it won't fit
    - Instead, they drop the packet and send back a "Too Large" ICMP control message
    - Forms the basis for "Path MTU Discovery", covered later
  - More (**MF**): this fragment is not the last one
- Offset (13 bits): what part of datagram this fragment covers in 8-byte units
- Thus, a fragment has either **MF** set or Offset > 0

25

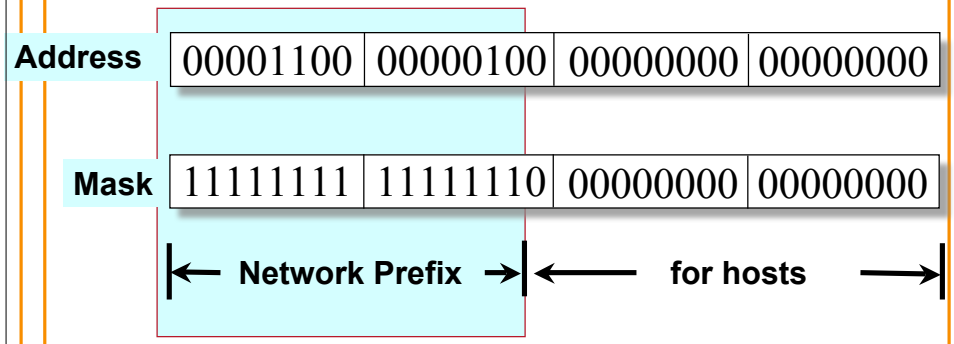
Security Implications of IP's Design



Classless Inter-Domain Routing (CIDR)

Use two 32-bit numbers to represent a network.
Network number = IP address + Mask

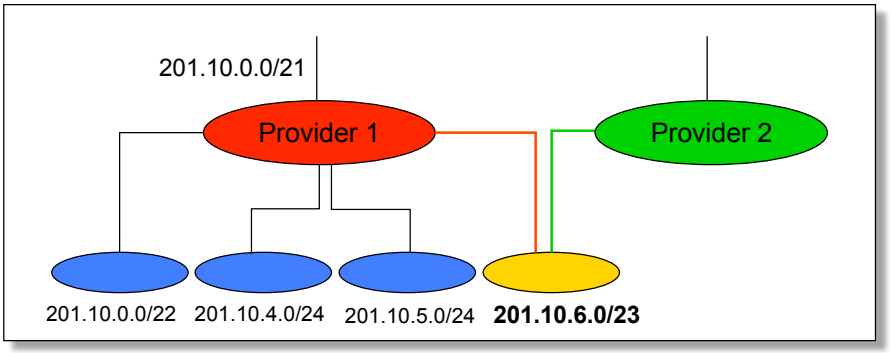
IP Address : 12.4.0.0 IP Mask: 255.254.0.0



Written as 12.4.0.0/15 or 12.4/15

27

But, Aggregation Not Always Possible



Multi-homed **customer with 201.10.6.0/23 has two providers. Other parts of the Internet need to know how to reach these destinations through *both* providers.**
⇒ /23 route must be globally visible

28

Obtaining a Block of Addresses

- Separation of control
  - Prefix: assigned *to* an institution
  - Addresses: assigned *by* the institution to their nodes
- Who assigns prefixes?
  - Internet Corporation for Assigned Names and Numbers
    - Allocates large address blocks to *Regional Internet Registries*
    - ICANN is **politically charged**
  - Regional Internet Registries (RIRs)
    - E.g., ARIN (American Registry for Internet Numbers)
    - Allocates address blocks within their regions
    - Allocated to Internet Service Providers and large institutions ($$)
  - Internet Service Providers (ISPs)
    - Allocate address blocks to their customers (could be recursive)
      - Often w/o charge

Longest-Prefix-Match Forwarding

Forwarding Table

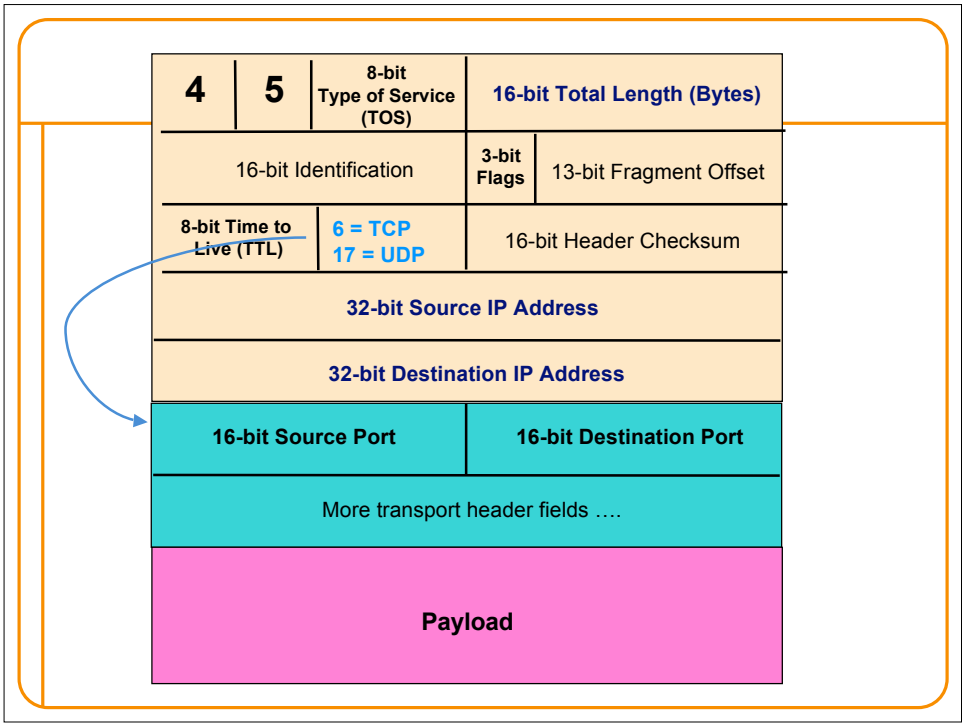
prefix	outgoing link
192.0.0.0/4	Serial3/0.2
4.83.128.0/17	Serial0/0.1
201.10.0.0/21	Serial2/2.3
201.10.6.0/23	Serial0/0.1
126.255.103.0/24	Serial1/1.1

destination

201.10.7.17 →

```
201: 11001001
 10: 00001010
  7: 00000111
 17: 00010001
```

```
  4: 00000100
 83: 01010011
128: 10000000
```



Why Would Anyone Use UDP?

- Finer control over what data is sent and when
  - As soon as an application process writes into the socket
  - … UDP will package the data and send the packet
- No delay for connection establishment
  - UDP just blasts away without any formal preliminaries
  - … which avoids introducing any unnecessary delays
- No connection state
  - No allocation of buffers, sequence #s, timers …
  - … making it easier to handle many active clients at once
- Small packet header overhead
  - UDP header is only 8 bytes

32

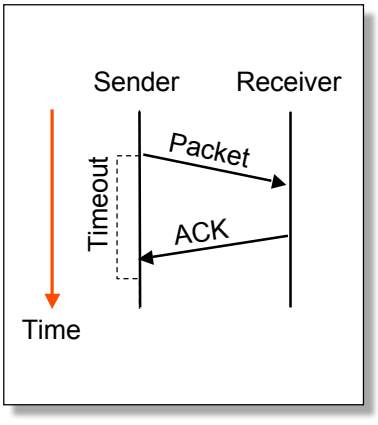
Transmission Control Protocol (TCP)

- Connection oriented
  - Explicit set-up and tear-down of TCP session
- Stream-of-bytes service
  - Sends and receives a stream of bytes, not messages
- Congestion control
  - Dynamic adaptation to network path's capacity
- Reliable, in-order delivery
  - TCP tries very hard to ensure byte stream (eventually) arrives intact
    - In the presence of corruption and loss
- Flow control
  - Ensure that sender doesn't overwhelm receiver

33

Automatic Repeat reQuest (ARQ)

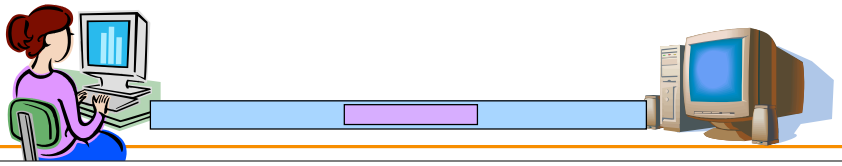
- Automatic Repeat Request
  - Receiver sends acknowledgment (ACK) when it receives packet
  - Sender waits for ACK and ***times out*** if does not arrive within some time period
- Simplest ARQ protocol
  - **Stop and Wait**
  - Send a packet, stop and wait until ACK arrives



34

How Fast Can Stop-and-Wait Go?

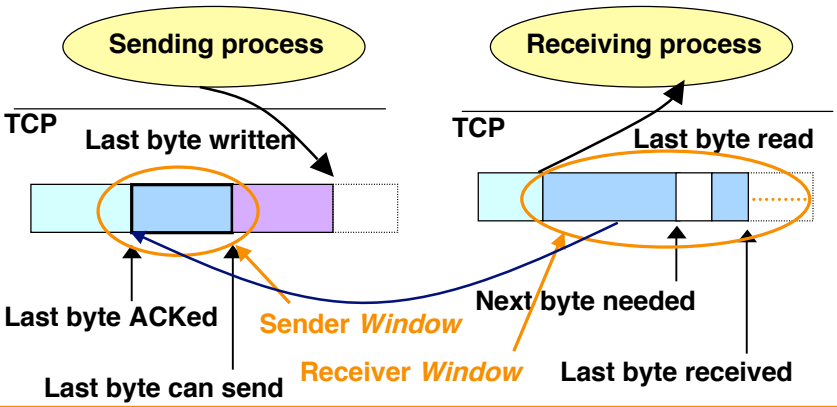
- Suppose we're sending from UCB to New York:
  - Bandwidth = 1 Mbps (megabits/sec)
  - RTT = 100 msec
  - Maximum Transmission Unit (MTU) = 1500 B = 12,000 b
  - No other load on the path and no packet loss
- What (approximately) is the fastest we can transmit using Stop-and-Wait?
- How about if Bandwidth = 1 Gbps?



35

Sliding Window

- Allow a larger amount of data "in flight"
  - Allow sender to get ahead of the receiver
  - … though not *too far* ahead



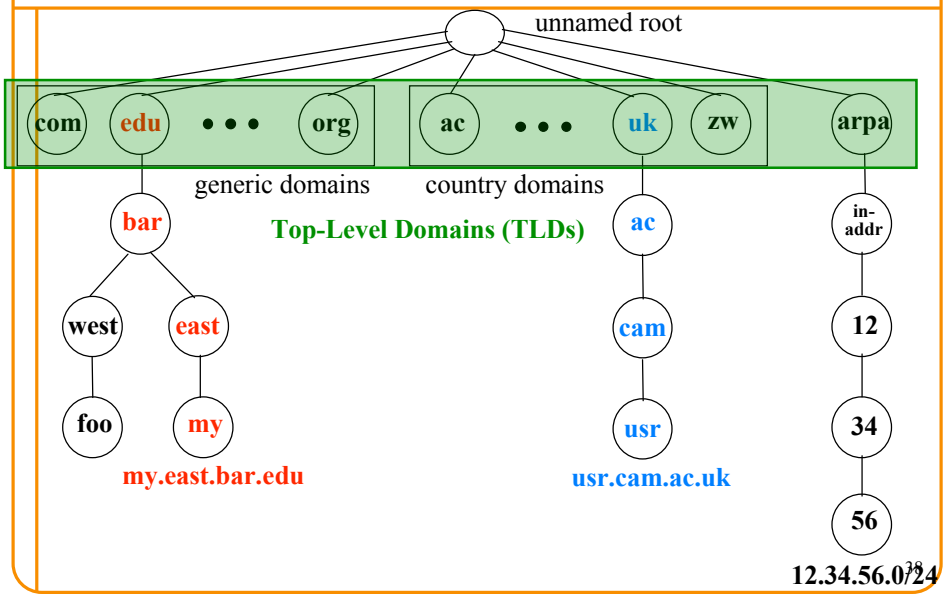
36

Performance with Sliding Window

- Given previous UCB ↔ New York 1 Mbps path with 100 msec RTT
  and Sender (and Receiver) window = 100 Kb = 12.5 KB
- How fast can we transmit?
- What about with 12.5 KB window & 1 Gbps path?
- Window required to fully utilize path:
  - **Bandwidth-delay product** (or "delay-bandwidth product")
  - 1 Gbps * 100 msec = 100 Mb = 12.5 MB
  - Note: large window = **many** packets in flight

37

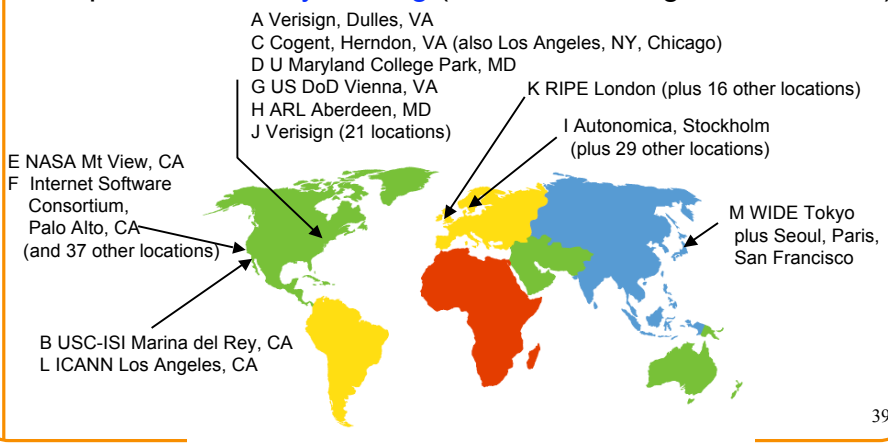
Distributed Hierarchical Database



38

DNS Root Servers

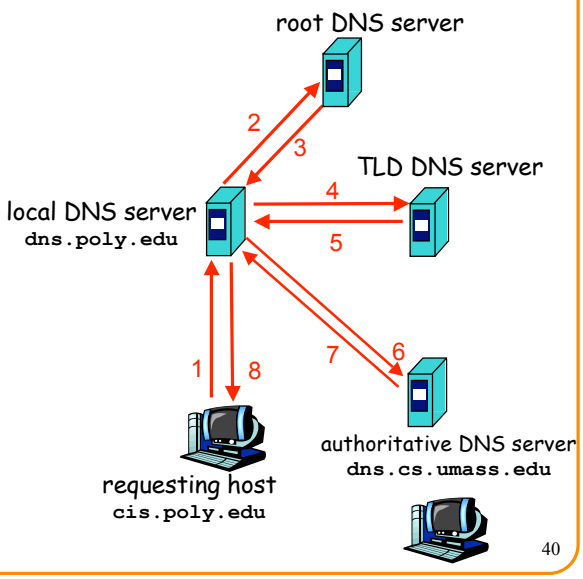
- 13 root servers (see http://www.root-servers.org/)
  - Labeled A through M
- Replication via any-casting (localized routing for addresses)



39

Recursive vs. Iterative Queries

- Recursive query
  - Ask server to get answer for you
  - E.g., request 1 and response 8
- Iterative query
  - Ask server who to ask next
  - E.g., all other request-response pairs



40

Reverse Mapping (Address → Host)

- How do we go the other direction, from an IP address to the corresponding hostname?
- Addresses already have natural "quad" hierarchy:
  - 12.34.56.78
- But: quad notation has most-sig. hierarchy element on left, while www.cnn.com has it on the right
- Idea: reverse the quads = 78.56.34.12 …
  - … and look that up in the DNS
- Under what TLD?
  - Convention: **in-addr.arpa**
  - So lookup is for `78.56.34.12.in-addr.arpa`

41

DNS Caching

- Performing all these queries takes time
  - And all this before actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can greatly reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a "time to live" (TTL) field
  - Server deletes cached entry after TTL expires

42

DNS Resource Records

DNS: distributed DB storing resource records (RR)

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

- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
- Type=PTR
  - **name** is reversed IP quads
    - E.g. 78.56.34.12.in-addr.arpa
  - **value** is corresponding hostname
- Type=CNAME
  - **name** is alias name for some "canonical" name
    E.g., **www.cs.mit.edu** is really **eecsweb.mit.edu**
  - **value** is canonical name
- Type=MX
  - **value** is name of mailserver associated with **name**
  - Also includes a weight/preference

43

unix> dig +norecurse @a.root-servers.net www.cnn.com

```
; <<>> DiG 9.2.2 <<>> +norecurse @a.root-servers.net www.cnn.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21041
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 14

;; QUESTION SECTION:
;www.cnn.com.                   IN      A

;; AUTHORITY SECTION:
com.                    172800  IN      NS      A.GTLD-SERVERS.NET.
com.                    172800  IN      NS      G.GTLD-SERVERS.NET.
com.                    172800  IN      NS      H.GTLD-SERVERS.NET.
com.                    172800  IN      NS      C.GTLD-SERVERS.NET.
com.                    172800  IN      NS      I.GTLD-SERVERS.NET.
com.                    172800  IN      NS      B.GTLD-SERVERS.NET.
com.                    172800  IN      NS      D.GTLD-SERVERS.NET.
com.                    172800  IN      NS      L.GTLD-SERVERS.NET.
com.                    172800  IN      NS      F.GTLD-SERVERS.NET.
com.                    172800  IN      NS      J.GTLD-SERVERS.NET.
com.                    172800  IN      NS      K.GTLD-SERVERS.NET.
com.                    172800  IN      NS      E.GTLD-SERVERS.NET.
com.                    172800  IN      NS      M.GTLD-SERVERS.NET.
```

Note, no "**ANSWER**" section

Cache Poisoning

- Suppose you are a Bad Guy and you control the name server for foobar.com. You receive a request to resolve *www.foobar.com* and reply:

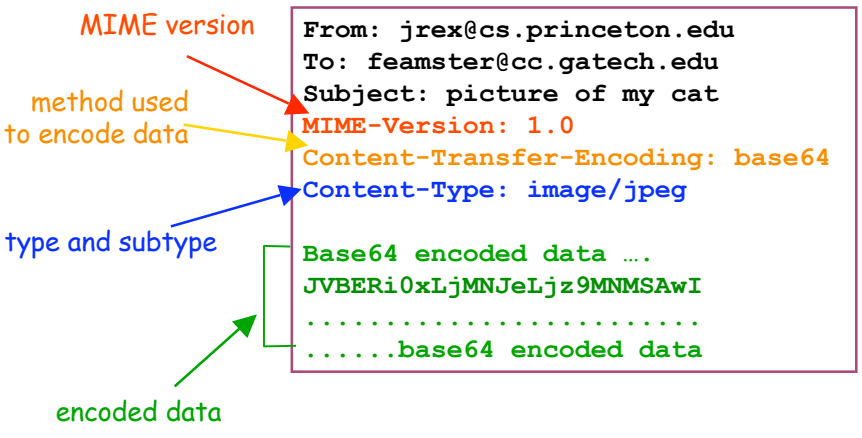
```
;; QUESTION SECTION:
;www.foobar.com.                IN      A

;; ANSWER SECTION:
www.foobar.com.         300     IN      A       212.44.9.144

;; AUTHORITY SECTION:
foobar.com.             600     IN      NS      dns1.foobar.com.
foobar.com.             600     IN      NS      google.com.

;; ADDITIONAL SECTION:
google.com.             5       IN      A       212.44.9.155
```

Evidence of the attack disappears 5 seconds later!

A foobar.com machine, *not* google.com

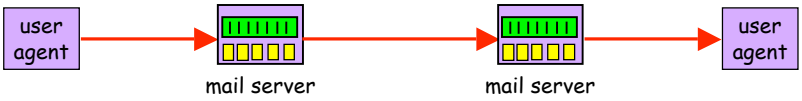
45

Example: E-Mail Message Using MIME



46

SMTP *Store-and-Forward* Protocol



- SMTP = ***Simple Mail Transfer Protocol***
- Messages sent through a series of servers
  - A server stores incoming messages in a queue
  - … to await attempts to transmit them to the next hop
- If the next hop is not reachable
  - The server stores the message and tries again later
- Each hop adds a "**`Received`**" header w/ its identity
  - Helpful for diagnosing problems with e-mail

47

Example With Received Header

```
Return-Path: <casado@cs.stanford.edu>
Received: from ribavirin.CS.Princeton.EDU (ribavirin.CS.Princeton.EDU [128.112.136.44])
      by newark.CS.Princeton.EDU (8.12.11/8.12.11) with SMTP id k04M5R7Y023164
      for <jrex@newark.CS.Princeton.EDU>; Wed, 4 Jan 2006 17:05:37 -0500 (EST)
Received: from bluebox.CS.Princeton.EDU ([128.112.136.38])
      by ribavirin.CS.Princeton.EDU (SMSSMTP 4.1.0.19) with SMTP id M2006010417053607946
      for <jrex@newark.CS.Princeton.EDU>; Wed, 04 Jan 2006 17:05:36 -0500
Received: from smtp-roam.Stanford.EDU (smtp-roam.Stanford.EDU [171.64.10.152])
      by bluebox.CS.Princeton.EDU (8.12.11/8.12.11) with ESMTP id k04M5XNQ005204
      for <jrex@cs.princeton.edu>; Wed, 4 Jan 2006 17:05:35 -0500 (EST)
Received: from [192.168.1.101] (adsl-69-107-78-147.dsl.pltn13.pacbell.net [69.107.78.147])
      (authenticated bits=0)
      by smtp-roam.Stanford.EDU (8.12.11/8.12.11) with ESMTP id k04M5W92018875
      (version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT);
      Wed, 4 Jan 2006 14:05:32 -0800
Message-ID: <43BC46AF.3030306@cs.stanford.edu>
Date: Wed, 04 Jan 2006 14:05:35 -0800
From: Martin Casado <casado@cs.stanford.edu>
User-Agent: Mozilla Thunderbird 1.0 (Windows/20041206)
MIME-Version: 1.0
To: jrex@CS.Princeton.EDU
CC: Martin Casado <casado@cs.stanford.edu>
Subject: Using VNS in Class
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit
```

48

Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: alice@crepes.fr
C: To: hamburger-list@burger-king.com
C: Subject: Do you like ketchup?
C:
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Message header

Message body

Lone period marks end of message

URL Syntax

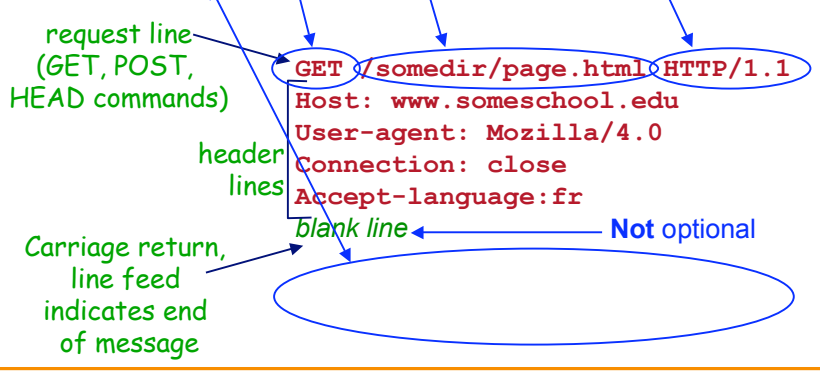
Content: How?

protocol : / / *hostname* [: *port*] / *directorypath* / *resource*

protocol	http, ftp, https, smtp, rtsp, *etc.*
hostname	FQDN, IP address
port	Defaults to protocol's standard port *e.g.* http: 80/tcp https: 443/tcp
directory path	Hierarchical, often reflecting file system
resource	Identifies the desired resource
	Can also extend to program executions: `http://us.f413.mail.yahoo.com/ym/ShowLetter?box=%40B%40Bulk&MsgId=2604_1744106_29699_1123_1261_0_28917_3552_1289957100&Search=&Nhead=f&YY=31454&order=down&sort=date&pos=0&view=a&head=b`

HTTP Request Message

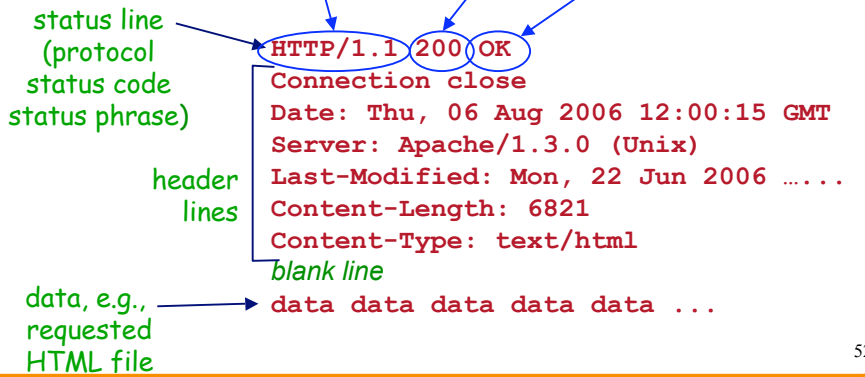
- Request message sent by a client
  - Request line: method, resource, and protocol version
  - Request headers: provide information or modify request
  - Body: optional data (e.g., to "POST" data to the server)



51

HTTP Response Message

- Response message sent by a server
  - Status line: protocol version, status code, status phrase
  - Response headers: provide information
  - Body: optional data



52

Stateless Operation

- Stateless protocol
  - Each request-response exchange treated independently
  - Clients and servers not required to retain state
- Statelessness improves scalability
  - Avoid need for server to retain info across requests
  - Enable server to handle a higher rate of requests
- However, some applications need persistent state
  - To uniquely identify the user or store temporary info
  - E.g., personalize a Web page, compute profiles or access statistics by user, track a shopping cart ….
  - Done using "cookies"

53

HTTP is Stateless

Client-Server: How?

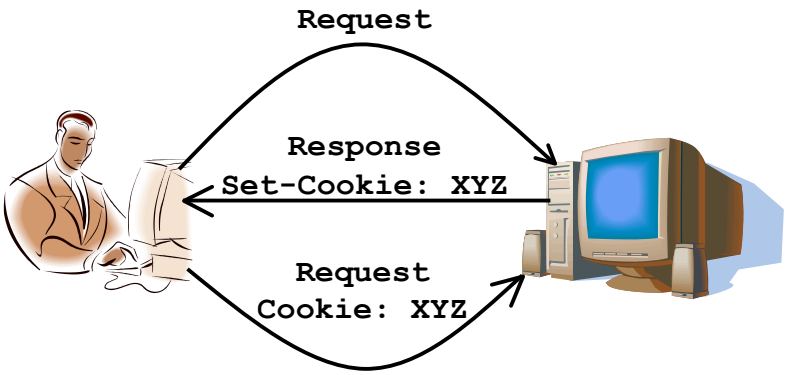
- **Stateless** protocol
  - Each request-response exchange treated independently
  - Servers *not* required to retain state
- This is good
  - Improves scalability on the server-side
    - Don't have to retain info across requests
    - Can handle higher rate of requests
    - Order of requests doesn't matter
- This is bad
  - Some applications need persistent state
    - Need to uniquely identify user or store temporary info
    - *e.g.,* Shopping cart, user preferences and profiles, usage tracking, …

54

State in a Stateless Protocol:

Cookies

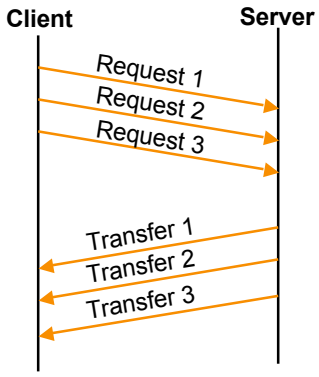
- *Client-side* state maintenance
  - Client stores small state on behalf of server
  - Client sends state in future requests to the server
- Can provide authentication



55

Pipelined Requests/Responses

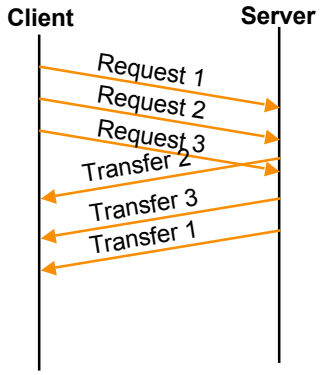
- Batch requests and responses to reduce the number of packets
- Multiple requests can be contained in one TCP segment
- Small items (common) can also share segments
- Note: maintains order of responses
  - Item 1 always arrives before item 2
- HTTP 1.1 feature (not in 1.0)



56

Concurrent Requests/Responses

- Use multiple connections to issue requests and responses in parallel
- Does not necessarily maintain order of responses
- Raises question of fairness
  - Set of ***N*** parallel connections "grabs" bandwidth ***N*** times more aggressively than just one
  - What's a reasonable/fair limit as traffic competes with that of other users?



57

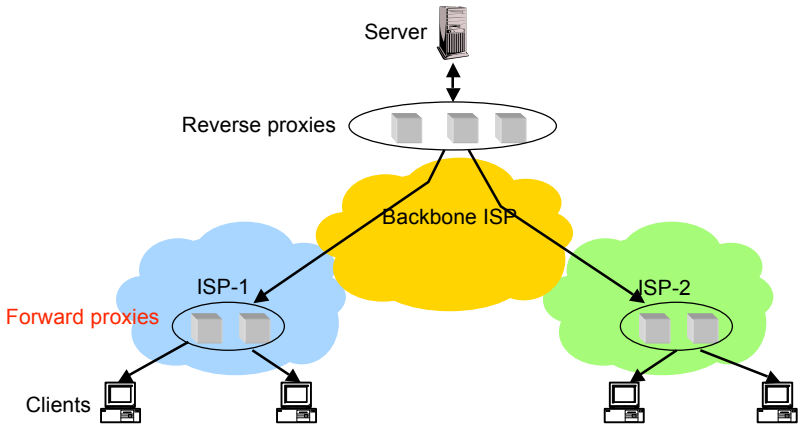
Persistent Connections

- Handle multiple transfers per connection
  - Including transfers subsequent to imaging current page
  - Maintain TCP connection across multiple requests
  - Either client or server can tear down the connection
- Performance advantages
  - Avoid overhead of connection set-up and tear-down
  - Allow TCP to learn more accurate RTT estimate
  - Allow the TCP congestion window to increase
    - I.e, leverage previously discovered bandwidth

58

Forward & Reverse Proxies

- Cache documents close to clients ◊ reduce network traffic and decrease latency
- Typically done by ISPs or corporate LANs



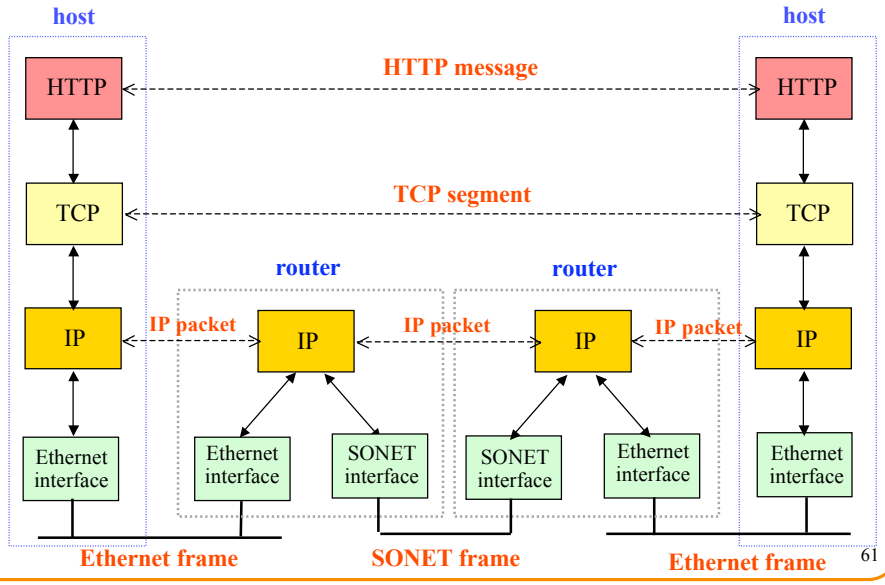
59

Caching vs. Replication (CDNs)

- Motivations for moving content close to users
  - Reduce latency for the user
  - Reduce load on the network and the server
- Caching
  - Replicating the content "on demand" after a request
  - Storing the response message locally for future use
  - May need to verify if the response has changed
  - … and some responses are not cacheable
- Replication
  - Planned replication of the content in multiple locations
  - Updating of resources handled outside of HTTP
  - Can replicate scripts that create dynamic responses

60

Message, Segment, Packet, and Frame



61

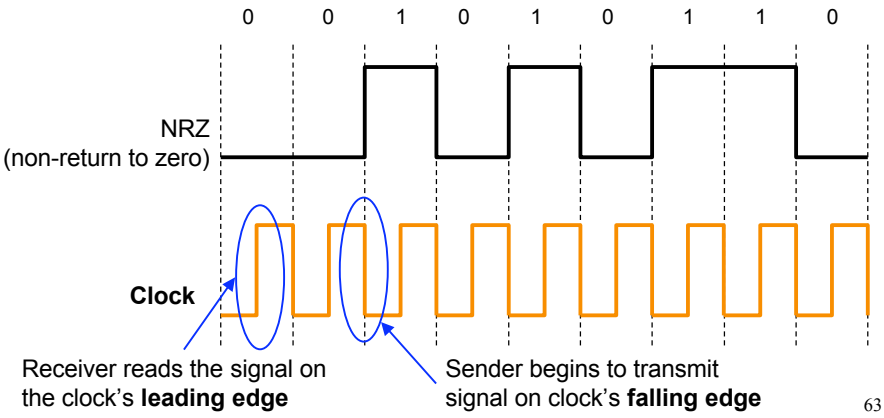
Link-Layer Services

- Encoding
  - Representing the 0s and 1s
- Framing
  - Encapsulating packet into frame, adding header, trailer
  - Using MAC addresses rather than IP addresses
- Error detection
  - Errors caused by signal attenuation, noise
  - Receiver detects presence, may ask for repeat (ARQ)
- Resolving contention
  - Deciding who gets to transmit when multiple senders want to use a shared media
- Flow control (pacing between sender & receiver)

62

Non-Return to Zero (NRZ)

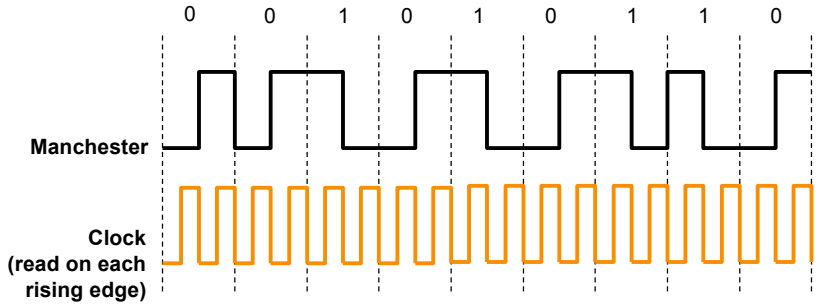
- 1 → high signal; 0 → low signal
  - (Actual signals are of course not so sharp)
- *How does receiver know where one bit stops and another begins?*



63

Manchester Encoding

- 1 → high-to-low transition; 0 → low-to-high transition
- Addresses clock recovery problems
- But: physical signaling must be twice as fast
  - To support 2 transitions/cycle ⇒ *Efficiency of 50%*



64

4-bit/5-bit (100Mb/s Ethernet)

- Goal: address inefficiency of Manchester encoding, while avoiding long periods of no transition
- Solution:
  - Use 5 bits to encode every sequence of four bits such that
    - No 5 bit code has more than one leading 0 or two trailing 0's
  - Use NRZI to then encode the 5 bit codes
  - Efficiency is 80%

4-bit	5-bit
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111

4-bit	5-bit
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

65

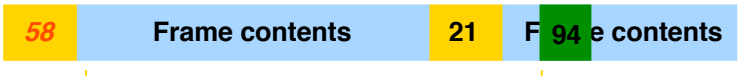
Simple Approach to Framing: Counting

- Sender: begin frame with byte(s) giving length
- Receiver: extract this length and count

53 bytes of data

21 bytes of data

- How can this go wrong?
- On occasion, the count gets **corrupted**



58 bytes of data misdelivered

Bogus new frame length; desynchronization

66

Framing: Sentinels

- Delineate frame with special pattern
  - e.g., **01111110** ⇒*start*, **01111111** ⇒*end*

- Problem: what if sentinel occurs within frame?
- Solution: escape the special characters
  - E.g., sender always inserts a 0 after five 1s
  - … receiver always removes a 0 appearing after five 1s
- Similar to escaping special characters in C programs

67

Error Detection

- Errors are unavoidable
  - Electrical interference, thermal noise, etc.
- Error detection
  - Transmit extra (redundant) information
  - Use redundant information to detect errors
  - Extreme case: send two copies of the data
  - Trade-off: accuracy vs. overhead
- Techniques for detecting errors
  - Parity checking
  - Checksum
  - Cyclic Redundancy Check (CRC)

68

Three Ways to Share the Media

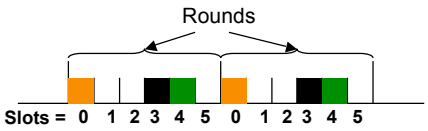
- Channel partitioning MAC protocols (TDMA, FDMA):
  - Share channel efficiently and fairly at high load
  - Inefficient at low load (where load = # senders):
    - 1/N bandwidth allocated even if only 1 active node!
- "Taking turns" protocols (discussed in **Section**)
  - Eliminates empty slots without causing collisions
  - Overhead in acquiring the token
  - Vulnerable to failures (e.g., failed node or lost token)
- Random access MAC protocols
  - Efficient at low load: single node can fully utilize channel
  - High load: collision overhead

69

Channel Partitioning: TDMA & FDMA

TDMA: time division multiple access

- Access to channel in "rounds"
  - Each station gets fixed length slot in each round
- Time-slot length is packet transmission time
  - *Unused slots go idle*
- Example: 6-station LAN with slots 0, 3, and 4



FDMA: frequency division multiple access

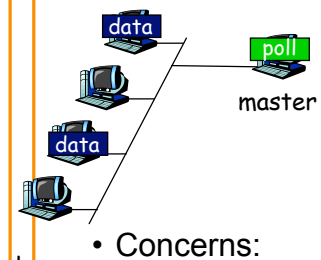
- Each station assigned fixed frequency band

70

"Taking Turns" MAC protocols

Polling

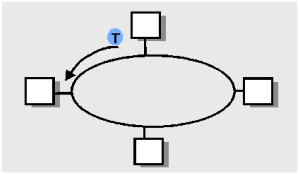
- Master node "invites" slave nodes to transmit in turn



- Concerns:
  - Polling overhead
  - Latency
  - Single point of failure (master)

Token passing

- Control token passed from one node to next sequentially
- Node must have token to send
- Concerns:
  - Token overhead
  - Latency
  - Single point of failure (token)



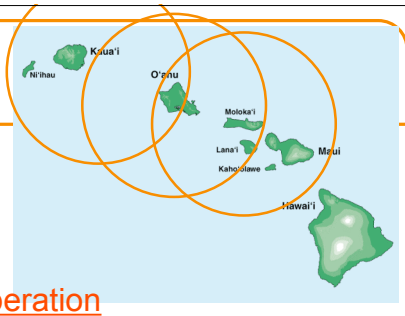
71

Key Ideas of Random Access

- Carrier sense
  - *Listen before speaking, and don't interrupt*
  - Checking if someone else is already sending data
  - … and waiting till the other node is done
- Collision detection
  - *If someone else starts talking at the same time, stop*
  - Realizing when two nodes are transmitting at once
  - …by detecting that the data on the wire is garbled
- Randomness
  - *Don't start talking again right away*
  - Waiting for a random time before trying again

72

Slotted ALOHA



Assumptions

- All frames same size
- Time divided into equal slots (time to transmit a frame)
- Nodes are synchronized
- Nodes begin to transmit frames *only at start of slots*
  - No *carrier sense*
- If two or more nodes transmit, all nodes detect collision

Operation

- When node obtains fresh frame, transmits in next slot
- No collision: node can send new frame in next slot
- Collision: node retransmits frame in each subsequent slot with probability **p** until success

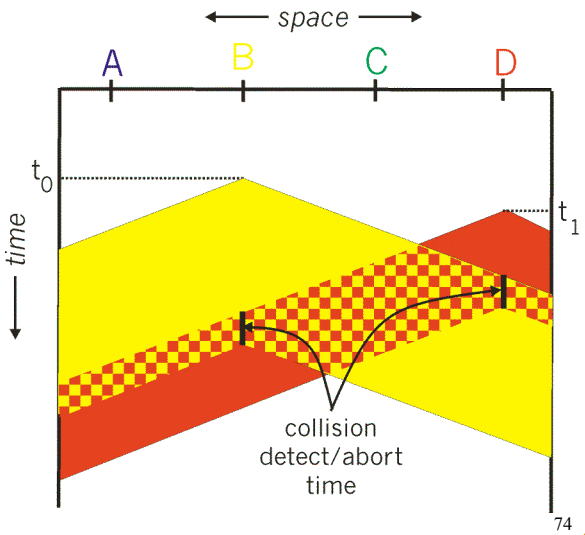
73

CSMA/CD Collision Detection

Both **B** and **D** can tell that collision occurred.

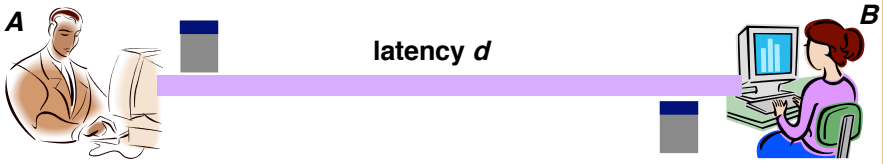
This lets them (1) know that they need to resend the frame, and (2) recognize that there's **contention** and adopt a strategy for dealing with it.

Note: for this to work, we need restrictions on **minimum frame size** and **maximum distance**



74

Limits on CSMA/CD Network Length

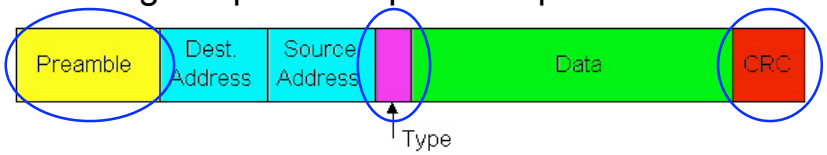


- *A* needs to wait for time ***2d*** to detect collision
  - So, *A* should keep transmitting during this period
  - … and keep an eye out for a possible collision
- Imposes restrictions. E.g., for 10 Mbps Ethernet:
  - Maximum length of the wire: 2,500 meters
  - Minimum length of a frame: 512 bits (64 bytes)
    - 512 bits = 51.2 μsec (at 10 Mbit/sec)
    - For light in vacuum, 51.2 μsec ≈ 15,000 meters
      vs. 5,000 meters "round trip" to wait for collision

75

Ethernet Frame Structure

- Sending adapter encapsulates packet in frame

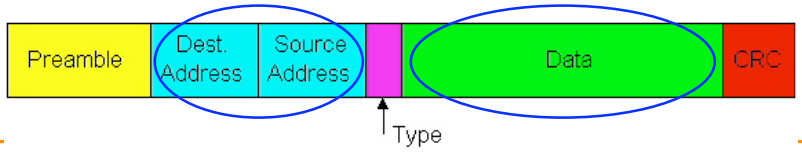


- Preamble: synchronization
  - Seven bytes with pattern **10101010**, followed by one byte with pattern **10101011**
  - Used to synchronize receiver & sender
- Type: indicates the higher layer protocol
  - Usually IP (but also Novell IPX, AppleTalk, …)
- CRC: cyclic redundancy check
  - Receiver checks & simply drops frames with errors

76

Ethernet Frame Structure (Continued)

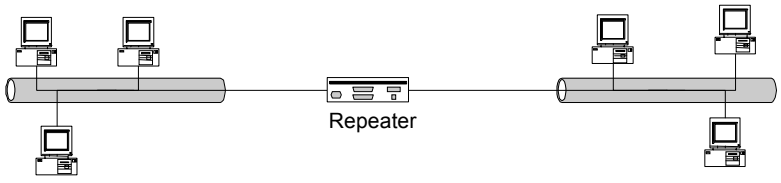
- Addresses: 48-bit source and destination MAC addresses
  - Receiver's adaptor passes frame to network-level protocol
    - If destination address matches the adaptor's
    - Or the destination address is the broadcast address (**ff**:**ff**:**ff**:**ff**:**ff**:**ff**)
    - Or the destination address is a multicast group receiver belongs to
    - Or the adaptor is in promiscuous mode
  - Addresses are **globally unique**
    - Assigned by NIC vendors (top three octets specify vendor)
      - During any given week, > 500 vendor codes seen at LBNL
- Data:
  - Maximum: 1,500 bytes
  - Minimum: 46 bytes (+14 bytes header + 4 byte trailer = 512 bits)



77

Physical Layer: Repeaters & Hubs

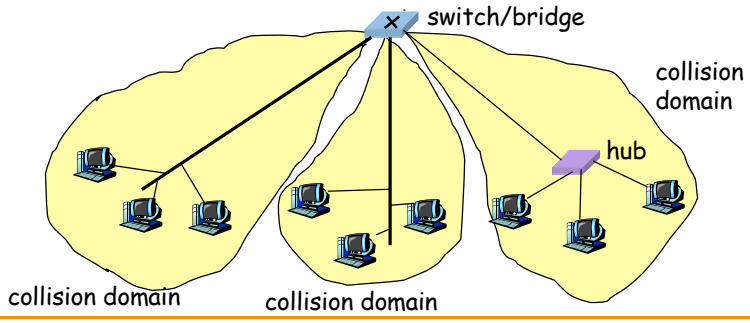
- Distance limitation in local-area networks
  - Electrical signal becomes weaker as it travels
  - Imposes a limit on the length of a LAN
    - In addition to limit imposed by collision detection
- Repeaters & Hubs join LANs together
  - Analog electronic device
  - Continuously monitors electrical signals on each LAN
  - Repeater transmits an amplified copy



78

Link Layer: Switches / Bridges

- Connect two or more LANs at the link layer
  - Extracts destination address from the frame
  - Looks up the destination in a table
  - Forwards the frame to the appropriate LAN segment
    - Or point-to-point link, for higher-speed Ethernet
- Each segment is its own collision domain



79

Advantages Over Hubs & Repeaters

- Only forwards frames as needed
  - Filters frames to avoid unnecessary load on segments
  - Sends frames only to segments that need to see them
- Extends the geographic span of the network
  - Separate collision domains allow longer distances
- Improves privacy by limiting scope of frames
  - Hosts can "snoop" the traffic traversing their segment
  - … but not all the rest of the traffic
- If needed, applies carrier sense & collision detection
  - Does not transmit when the link is busy
  - Applies exponential back-off after a collision
- Joins segments using different technologies

80

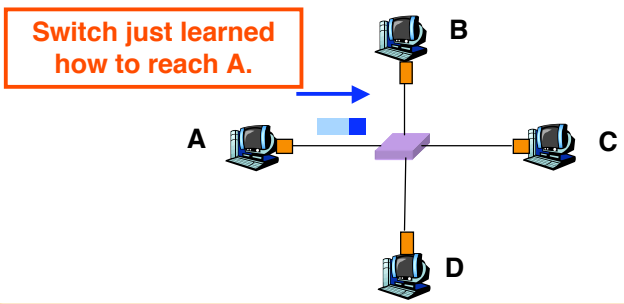
Disadvantages Over Hubs & Repeaters

- Higher cost
  - More complicated devices that cost more money
- Delay in forwarding frames
  - Bridge/switch must receive and parse the frame
  - … and perform a look-up to decide where to forward
  - Introduces store-and-forward delay
    - Can ameliorate using *cut-through switching*
      - Start forwarding after only header received
- Need to learn where to forward frames
  - Bridge/switch needs to construct a *forwarding table*
  - Ideally, without intervention from network administrators
  - Solution: self-learning

81

Self Learning: Building the Table

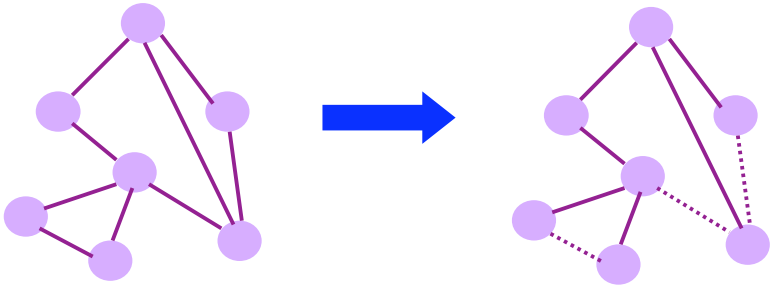
- When a frame arrives
  - Inspect *source* MAC address
  - Associate address with the *incoming* interface
  - Store mapping in the switch table
  - Use time-to-live field to eventually forget the mapping
    - ***Soft state***



82

Avoiding Loops: Spanning Trees

- Ensure the forwarding topology has no loops
  - Avoid using some of the links when flooding
  - … to prevent loop from forming
- Spanning tree (K&R pp. 406-408)
  - Sub-graph that covers all vertices but contains no cycles
  - Links not in the spanning tree do not forward frames



83

Comparing Hubs, Switches & Routers

	hubs	switches	routers
traffic isolation	no	yes	yes
plug & play	yes	yes	no
optimized routing	no	no	yes
cut-through	yes	yes	no

84