Computer Network and Distributed Systems

Transport Layer (TCP, UDP)

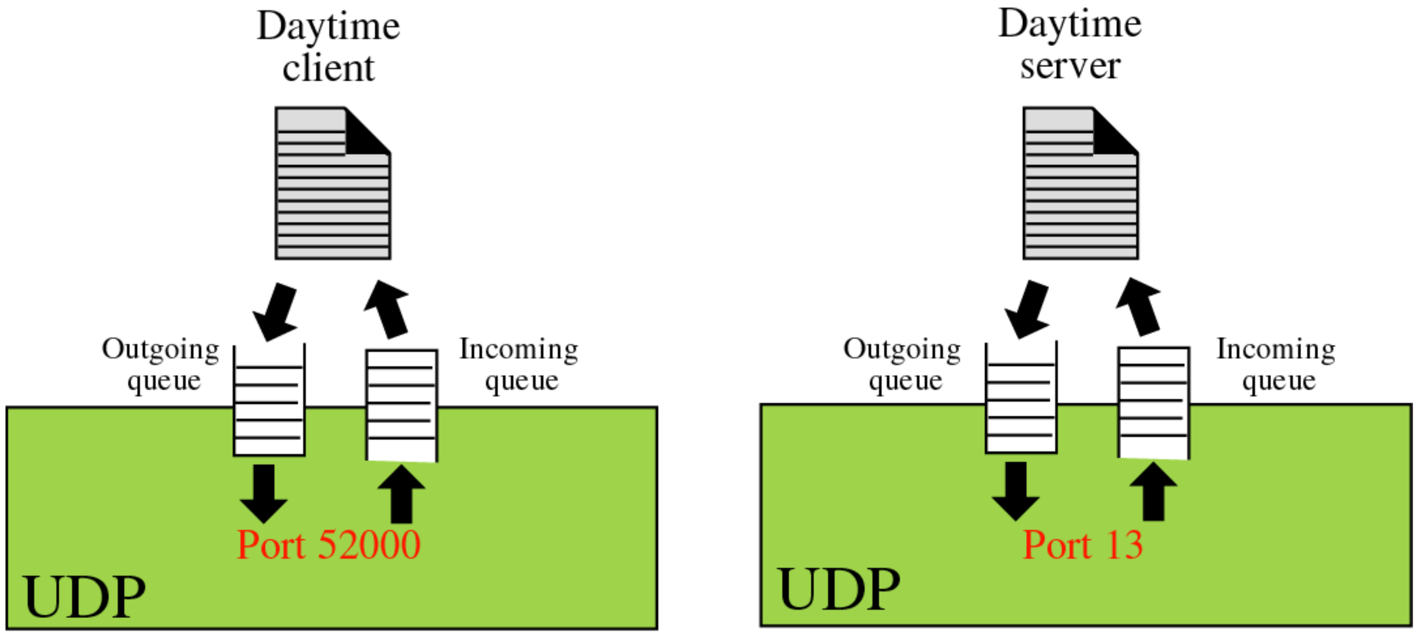
Transport Layer functions

- Possible transport layer functions:
  - Process-to-process delivery
  - Establish and maintain end-to-end connections among processes (explicitly establish and terminate connections)
  - Guarantee reliable, in-order end-to-end transfer
  - Provide end-to-end flow control
  - Congestion control

- UDP provides only the first function
- TCP provides all of the above

Ports

- Port: a 16-bit integer, used to identify an application
- Ports typically implemented as a pair of queues: an incoming queue and an outgoing queue



Classification of Ports

- Well-known / reserved ports: ports up to 1023
  - Normally used for protocols with wide applicability
  - Registered by ICANN (Internet Corporation for Assigned Names and Numbers)
  - E.g. ftp – 21, 20, telnet – 23, web server – 80
- Registered ports: ports 1024 – 49151
  - Used to avoid port collisions between user-level applications developed independently when installed on the same machine
  - Registered by IANA (Internet Assigned Numbers Authority)
  - E.g. vlc media player – 1234, RADIUS authentication protocol - 1812
- Dynamic or private ports: ports 49152 – 65535
  - Can be used by any application
  - cannot be registered with IANA
  - temporary purposes and for automatic allocation

Endpoints and Connections

- Endpoint
  - a 2-tuple <host IP, port>
  - Commonly called a socket
- Connection
  - Defined by two endpoints
  - Two connections will have at least one endpoint different (but can have one endpoint same)
- Messages de-multiplexed based on
  - Connections in TCP (TCP maintains connections)
  - Endpoint in UDP (no connection maintained)

Transmission Control Protocol (TCP)

Transmission Control Protocol

- Guarantees reliable, in-order delivery of a stream of bytes between sender and receiver applications
- Connection oriented (establishment, termination)
- Full-duplex protocol
  - Each TCP connection supports a pair of byte streams, one flowing in each direction
- Flow-control mechanism for both byte streams
  - Receiver can limit how many bytes the sender can send at a given time
- Congestion-control mechanism
  - Control how fast sender can send data, to prevent the sender from overloading the network

Stream, segment, sequence number

- Data sent by application process is viewed as a stream (sequence) of bytes

- Segment – the unit of transfer between TCP modules on two hosts
  - stream of bytes divided into segments, each segment given a TCP header, and given to IP module to send

- Each TCP segment has a sequence number to specify position of the first byte in the segment within the byte stream

Maximum Segment Size (MSS)

- TCP *usually* sets MSS to the size of the largest segment that can be sent, without causing the local IP to fragment it

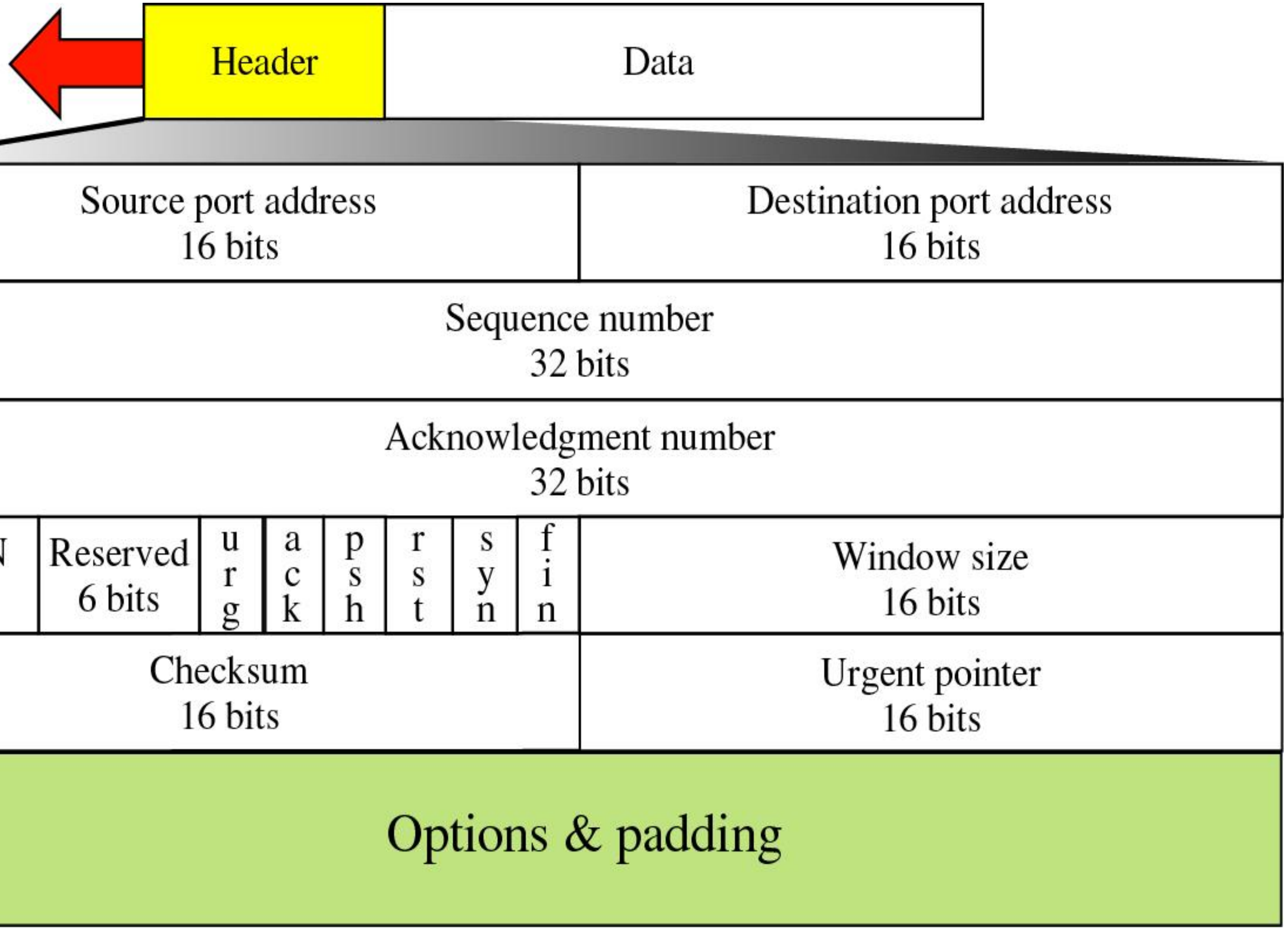
- For example, if both nodes lie in same network,
  - MSS = (MTU of underlying network – IP header size – TCP header size)

- A TCP module can negotiate with the TCP module at the other end to specify the MSS that it is willing to receive

Maximum Segment Lifetime (MSL)

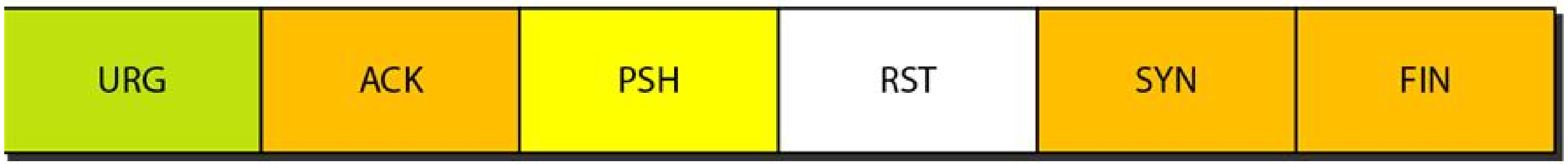
- How late can a segment arrive at destination?
  - Routers using IP discard packets if TTL expires
  - TCP assumes each packet has a maximum lifetime
  - MSL currently taken to be 120 seconds
  - Just an estimate used by TCP, not guaranteed by IP

TCP segment header



Flags in TCP header

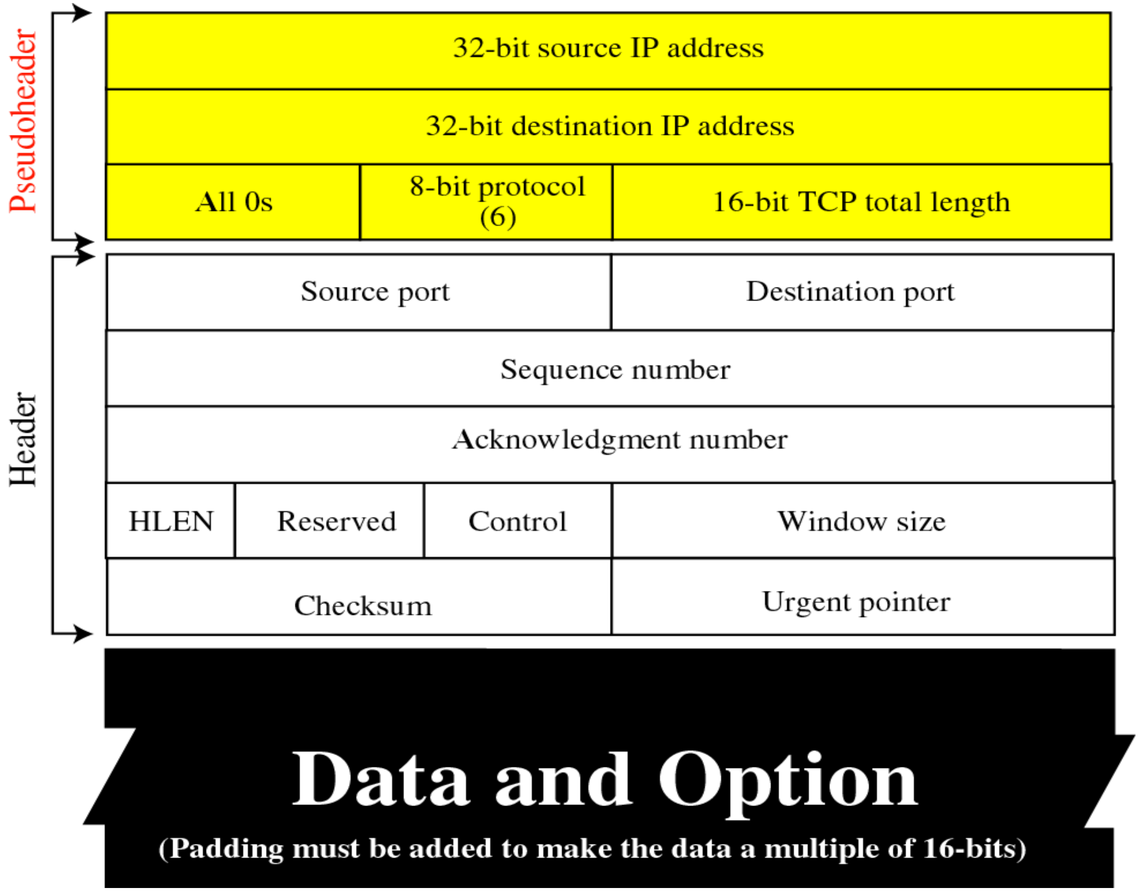
- **URG** – urgent pointer is valid in this segment
- **ACK** – the acknowledgement number is valid
- **PSH** – sender application requests sending TCP module to send all data accumulated in buffer
- **RST** – indicates the connection has been reset
- **SYN** – synchronize the sequence numbers to establish a connection
- **FIN** – application has finished sending data, wants to close connection



Checksum in TCP

- TCP checksum algorithm similar to that in IP
- TCP computes checksum over data, TCP header and a pseudo-header
- TCP pseudo-header consists of
  - TCP length field
  - Three fields from the IP header: protocol number (6 for TCP), source IP address, destination IP address
  - One octet of zeroes to pad the segment to an exact multiple of 16 bits

TCP pseudo-header



TCP pseudo-header (contd.)

- Pseudo-header (and the octet used for padding) are NOT transmitted along with the TCP segment, nor are they included in the length

- Receiver TCP module also prepends pseudo-header and compares checksum

- Why use pseudo-header?
  - To verify that this segment has been delivered between the correct two endpoints

Connection Establishment

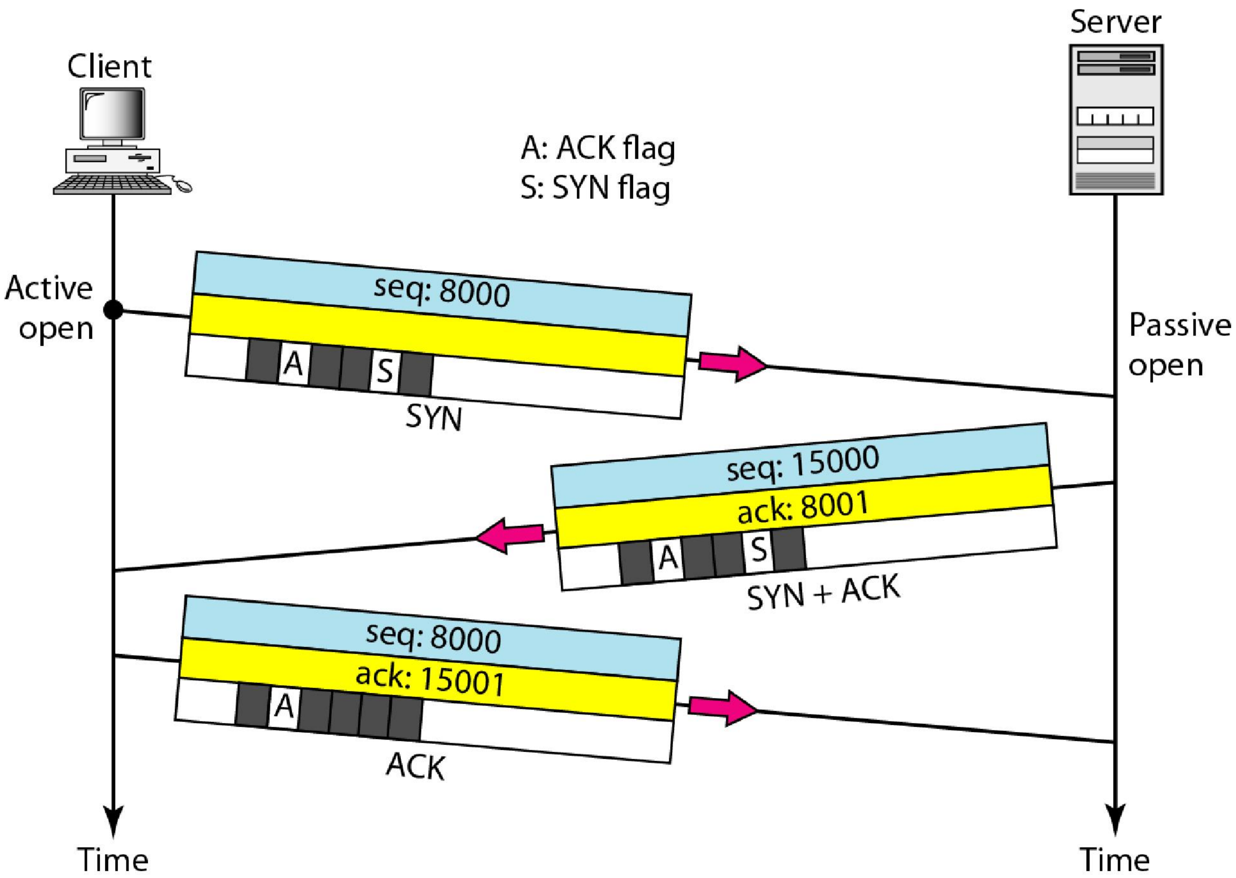
- Purpose: both sides needs to know
  - that the other side is ready for data transfer, the port used by the other side, MSS, etc
  - the other side's Initial Sequence Number (ISN)
- First byte address cannot always be 0 or 1
  - Protect against two incarnations of the same connection re-using the same ISN too soon
  - TCP specification: each side of a connection selects an ISN at random
  - For example, ISN set from a 32-bit clock that ticks every 4 microseconds (wraps around once every 4.55 hours)

Connection establishment (contd.)

- **3-way handshake**

- Client sends SYN segment (no data)
  - specifies port numbers, etc
  - Sequence Number: client's ISN isn_c
- Server responds with a SYN + ACK segment
  - Sequence Number: server's ISN isn_s
  - Acknowledgement number: isn_c + 1
- Client sends ACK segment
  - Acknowledgement number: isn_s + 1
- The side sending the first SYN is said to perform an active open. The other side performs a passive open

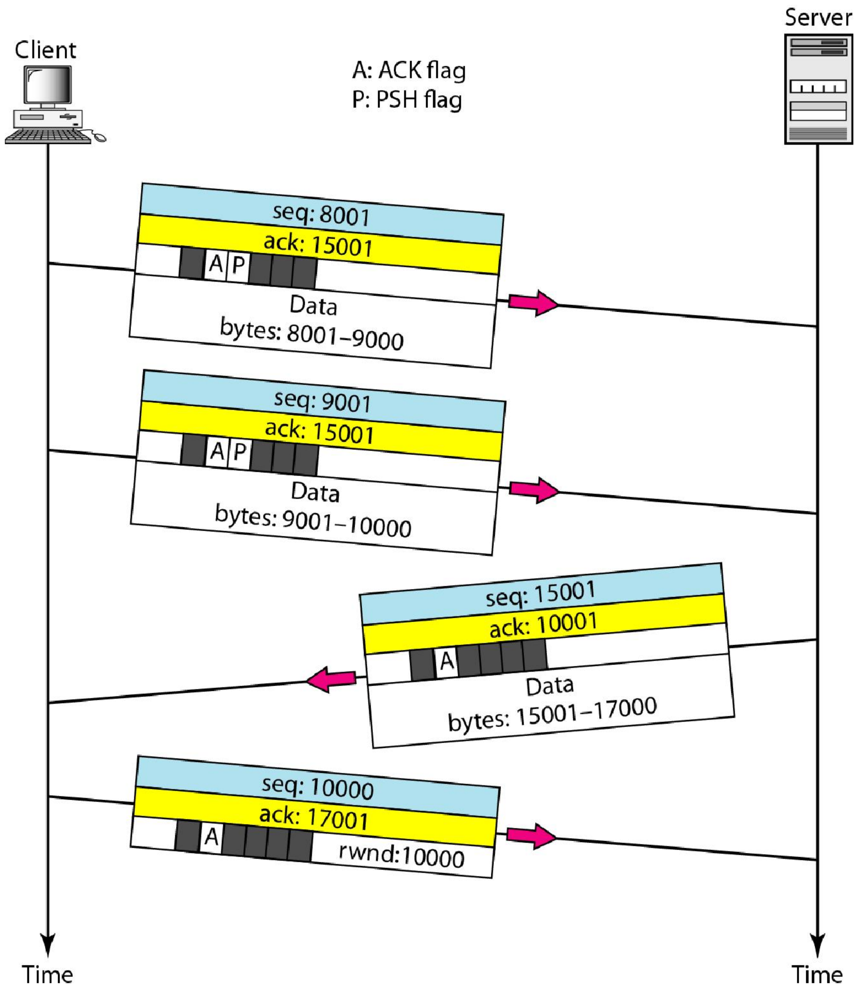
Connection establishment using three-way handshaking



Basic TCP data transfer

- Once connection is established between sender TCP module and receiving TCP module …
- TCP modules send and receive TCP segments with data
  - Each segment contains a sequence number, source and destination port
- Acknowledgments sent by TCP modules to other side
  - Indicated by ACK flag set to 1 and a valid acknowledgement number field: the next sequence number the receiver expects
  - TCP uses cumulative, positive ACK
  - ACK frame may have no data, or ACK may be piggybacked over data segments
- When data transfer over, close connection

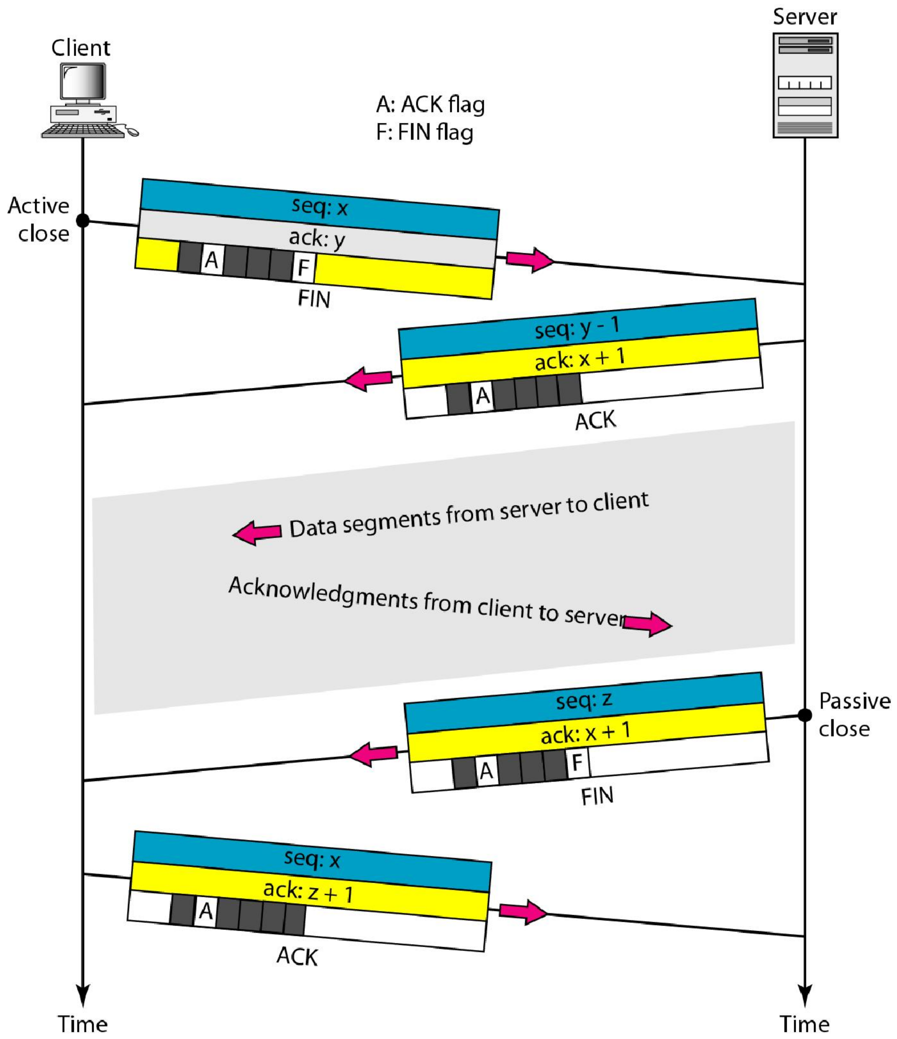
Once connection is established



Normal Connection Termination

- A duplex connection treated as two one-way connections, each to be closed separately
- Either side can initiate termination
  - TCP module sends FIN segment with a sequence num
  - Indicates that the application process on the side sending FIN will not send any more data
- The other side acknowledges FIN segment
  - TCP module receiving FIN sends an ACK segment with ACK number = FIN segment sequence number + 1
  - This side can go on sending data
- The above process repeats when the other side wants to close connection

Connection Termination - Half-close



TIME-WAIT state

- After both sides have sent FIN segments and both sides have been ACK-ed by the other side, connection placed in TIME-WAIT state

- Connection remains in the TIME-WAIT state for 2*MSL before data structures for this connection are removed

- Why TIME-WAIT state?
  - try to ensure that any outstanding data transmitted from both sides are received

States for TCP Connection

State	Description
CLOSED	There is no connection.
LISTEN	The server is waiting for calls from the client.
SYN-SENT	A connection request is sent; waiting for acknowledgment.
SYN-RCVD	A connection request is received.
ESTABLISHED	Connection is established.
FIN-WAIT-1	The application has requested the closing of the connection.
FIN-WAIT-2	The other side has accepted the closing of the connection.
TIME-WAIT	Waiting for retransmitted segments to die.
CLOSE-WAIT	The server is waiting for the application to close.
LAST-ACK	The server is waiting for the last acknowledgment.

RST flag

- Used to reset a connection (abnormal close)

- TCP module on one side sends a segment with the RST bit set
  - TCP module on the other side responds to a RST segment by aborting the connection immediately, and informs the application that a reset occurred

- RST flag cannot be set by the applications, it is set by the TCP modules

Out of Band Data

- Sender application may want some data to be delivered to the receiving application urgently (before what was sent previously)
  - TCP module sends segment with URG flag set to 1
  - Location of urgent data identified by the Urgent Pointer field

- When this segment reaches receiver TCP module
  - Receiver TCP module will notify receiver application and hands over the urgent data

TCP Flow Control and Error Control

Sliding window flow control

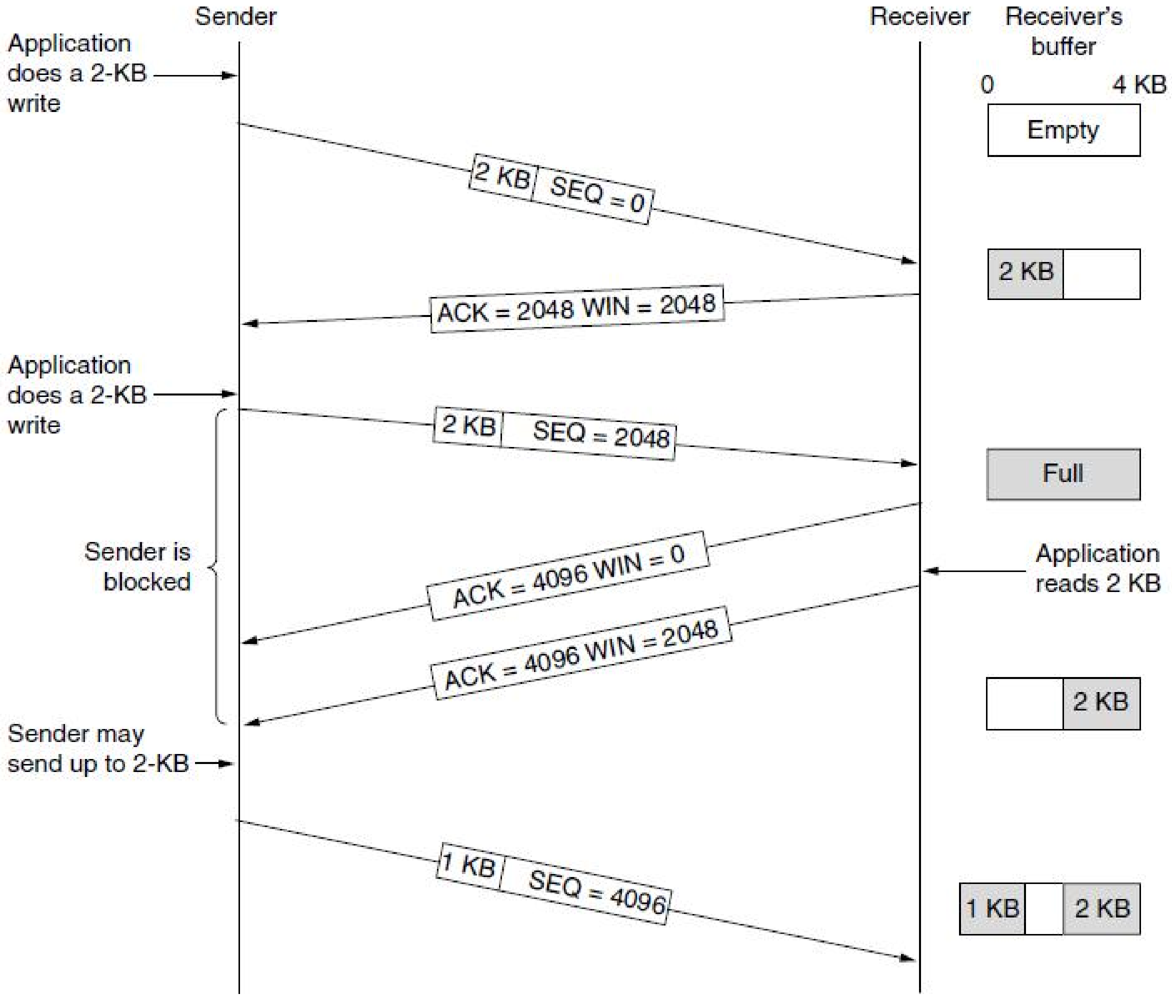
- TCP uses a byte-level sliding window flow control scheme with cumulative positive ACKs
  - Sender TCP module maintains a window of size *w* and start of window *X*
  - Can send up to *n* bytes starting from byte *X* without receiving an acknowledgement
- Window size at sender determines how much unacknowledged data can the sender send
- Also takes care of error control (segments re-transmitted if corrupted or lost)

Problem at the receiver

- Data remains in buffer of receiver TCP module till the receiver application reads data
- So, depending on how fast the application is reading the data, the receiver's window size may change

- Solution: advertised window
  - In every segment (data or ACK or both) sent to the sender, receiver tells its current window size
  - Sender uses this value instead of a fixed receiver window size

Advertised window – example



Silly window syndrome

- Receiver TCP module advertising small window, sender TCP module sending segments with few bytes of data, …
  - large overhead
- Avoiding silly window at receiver
  - Once zero window size has been advertised, receiver must wait until 'significant amount' of buffer empty
  - Receiver delays sending ACK if the window size is not sufficiently large to advertise (up to 500 msec)
- Avoiding silly window at sender
  - When to send segments? Nagle's algorithm

Nagle's algorithm

- When the application produces additional data to send:

if no data in flight

send segment immediately (with as much data as allowed by advertised window W)

else *# there is un-ACKed data in flight*

buffer new data until available data >= MSS, and then send segment with as much data as allowed by W

if ACK received while waiting to send

send all data accumulated in buffer (or as much as is allowed by W) now

end if

end if

Push flag in TCP header

- Normally, TCP module decides when to send a segment

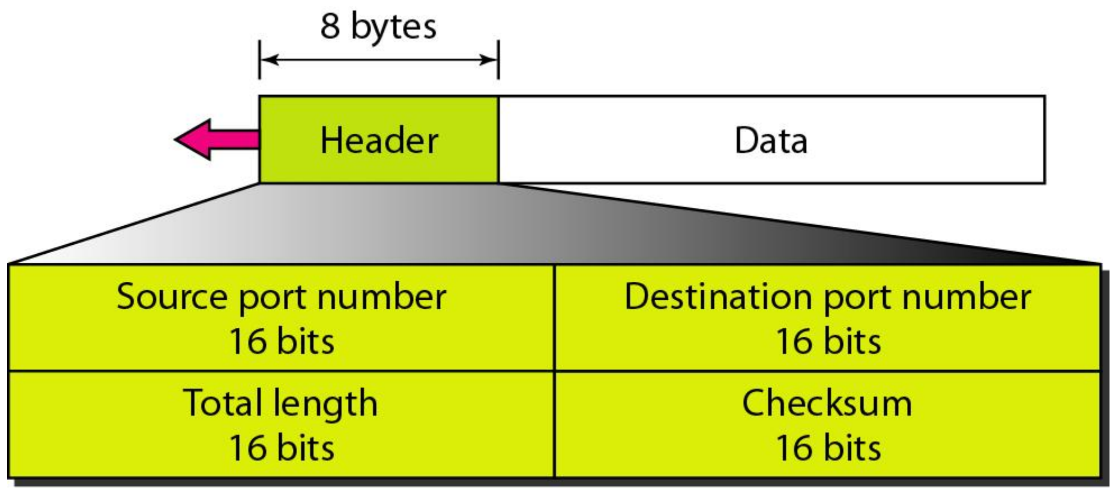
- Sender application can tell sender TCP module to send (flush) whatever bytes it has collected
  - TCP module sends segment with PSH flag set to 1
  - When segment with PSH=1 reaches receiver TCP module, data given to receiving application immediately

User Datagram Protocol (UDP)

UDP

- Simply extends the unreliable host-to-host delivery service of underlying network into a process-to-process communication channel
- Identification for each process: <host IP, port>
- Connectionless, each UDP segment handled independent of others
- UDP does NOT implement flow control, reliable or ordered delivery

UDP header



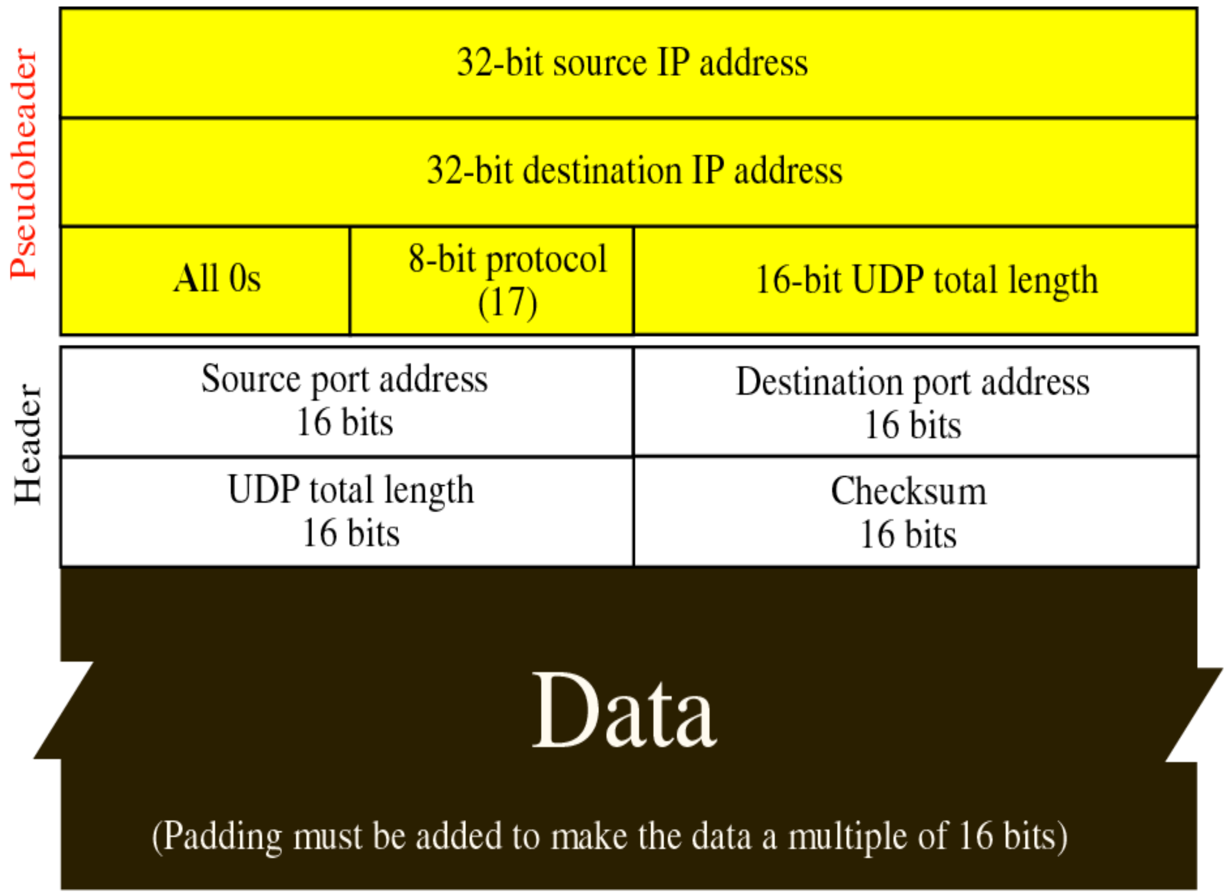
Fields in the UDP header

- ❑ Source port, destination port
  - 16-bit UDP protocol port numbers used to demultiplex datagrams (along with IP address)

- ❑ Length field: number of octets in the UDP datagram, including the UDP header and data
  - ➢ Minimum value is 8

- ❑ Checksum: same method as in TCP
  - ➢ Checksum computed over the UDP header, the contents of the message body, and a pseudo-header

Pseudo-header added to UDP datagram



Why is there a UDP?

- Since TCP is decidedly better than UDP, why is there a UDP at all?
  - UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application; UDP avoids the overhead of such processing at the level of the network interface.
  - Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets (VoIP), which may not be an option in a real-time system
- Some advantages of UDP
  - Low overhead
  - Simple, requires very little resources
  - Fine control over when data is sent

Computer Network and Distributed Systems

Name / Address resolution
Domain Name System

Three kinds of identifiers

- Host name (e.g., www.google.com)
  - Mnemonic name appreciated *by humans*
  - Provides little (if any) information about location
  - Hierarchical, variable number of alpha-numeric characters
- IP address (e.g., 64.236.16.20)
  - Numerical address appreciated *by routers*
  - Related to host's current location in the topology
  - Hierarchical name space of 32 bits
- MAC address (e.g., 00-15-C5-49-04-A9)
  - Numerical address appreciated *within local area network*
  - Unique, hard-coded in the adapter when it is built
  - Flat name space of 48 bits

Mapping between identifiers

- Domain Name System (DNS)
  - Given a host name, provide the IP address
  - Given an IP address, provide the host name

- Dynamic Host Configuration Protocol (DHCP)
  - Given a MAC address, assign a unique IP address … and tell host other stuff about the LAN
  - To automate the boot-strapping process

- Address Resolution Protocol (ARP)
  - Given an IP address, provide the MAC address
  - To enable communication within the LAN

Domain Name System

Naming and Resolving names

- Problems
  - How to assign meaningful high-level names to a very large number of hosts
  - How to resolve names i.e. how to map between high-level names and IP addresses

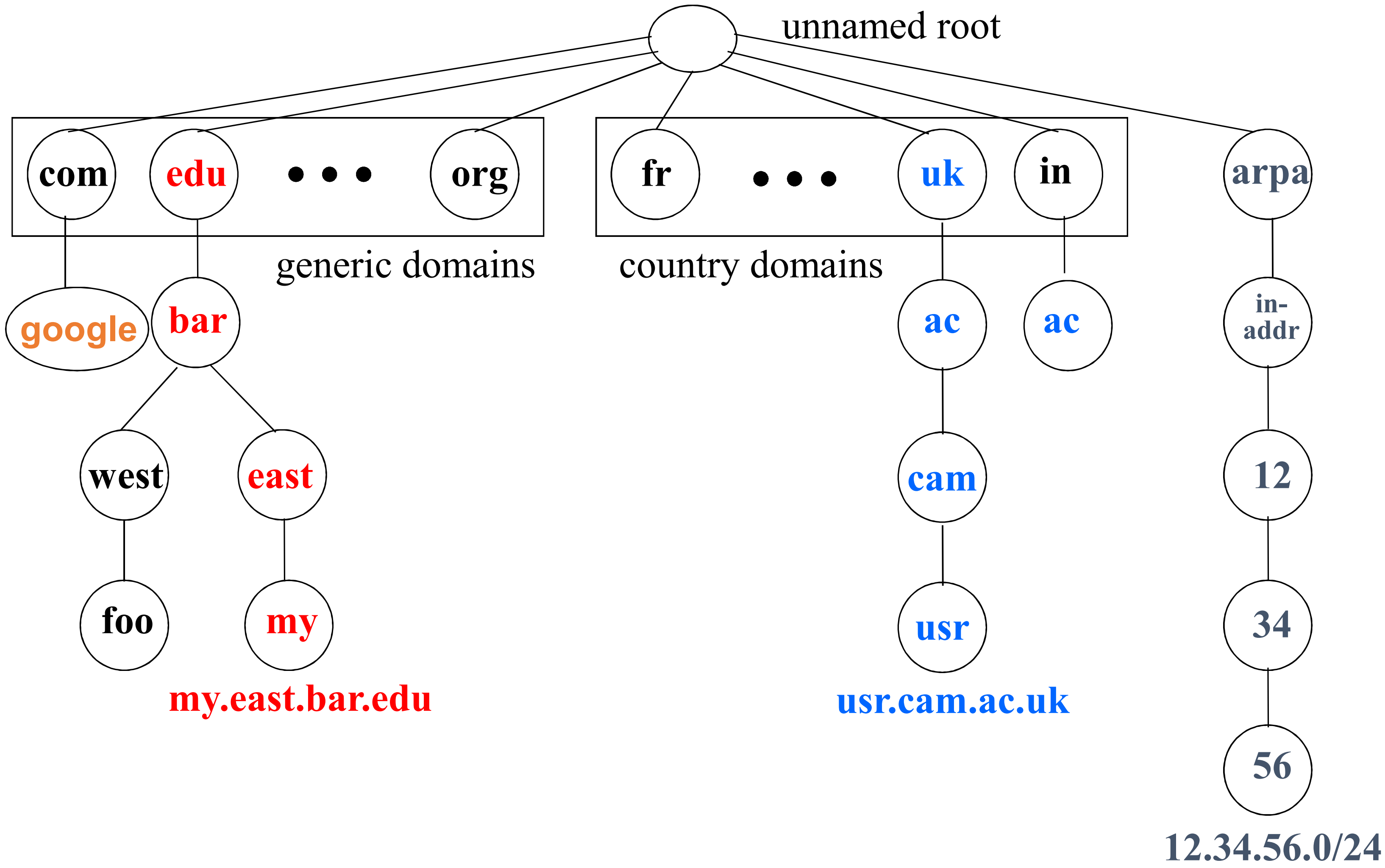
- Initial years of the Internet: name-IP mappings of all hosts stored in a file hosts.txt at a server maintained at NIC
  - Flat namespace, each host given a unique name (just a sequence of characters without further structure)
  - All hosts used to download this file periodically

The current approach - DNS

- Two aspects of DNS
  - Specifies syntax of names and rules for delegating authority over names
  - Specifies implementation of a distributed mechanism that efficiently maps names to IP addresses
- A domain name consists of a sequence of labels separated by a delimiter character, the period

- Uses a hierarchical namespace of domain names
  - Any suffix of a label in a domain name is also a domain
  - DNS namespace can be visualized as a tree

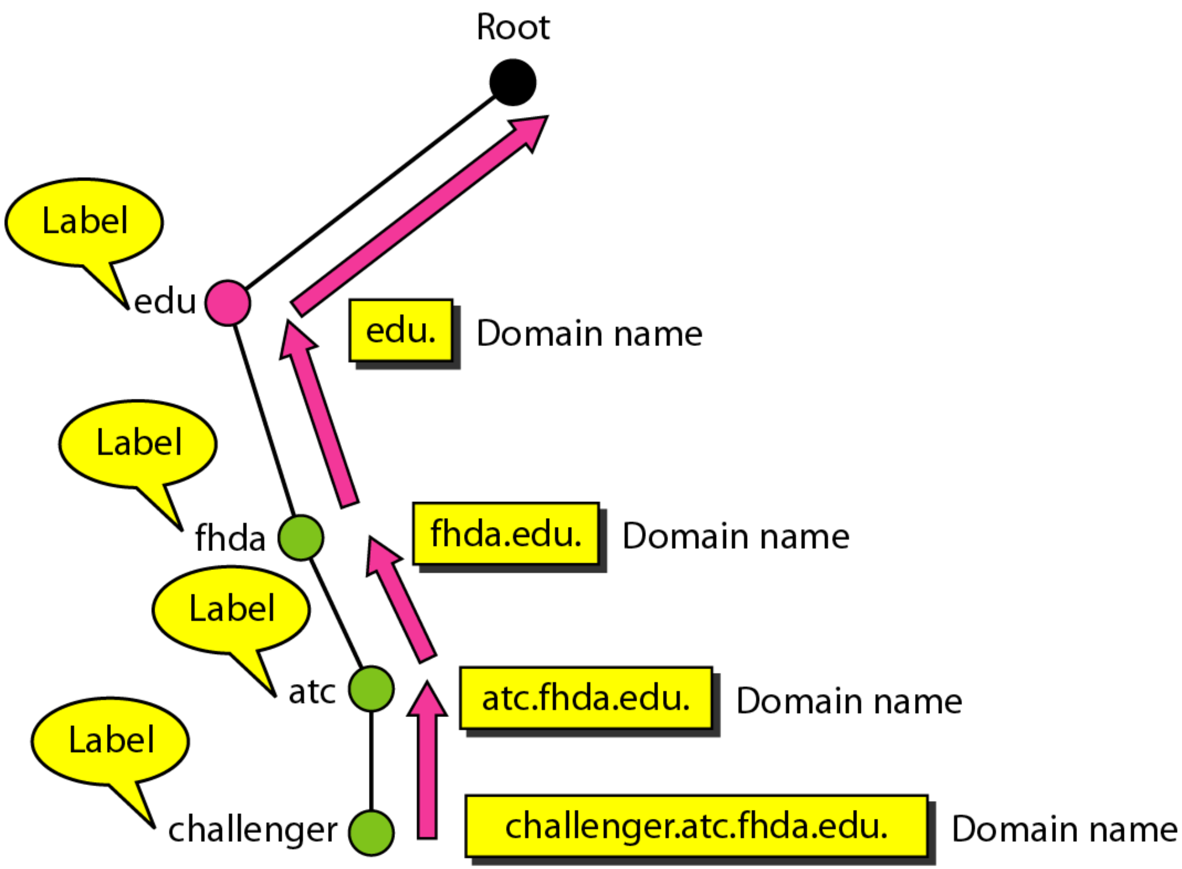
DNS Namespace



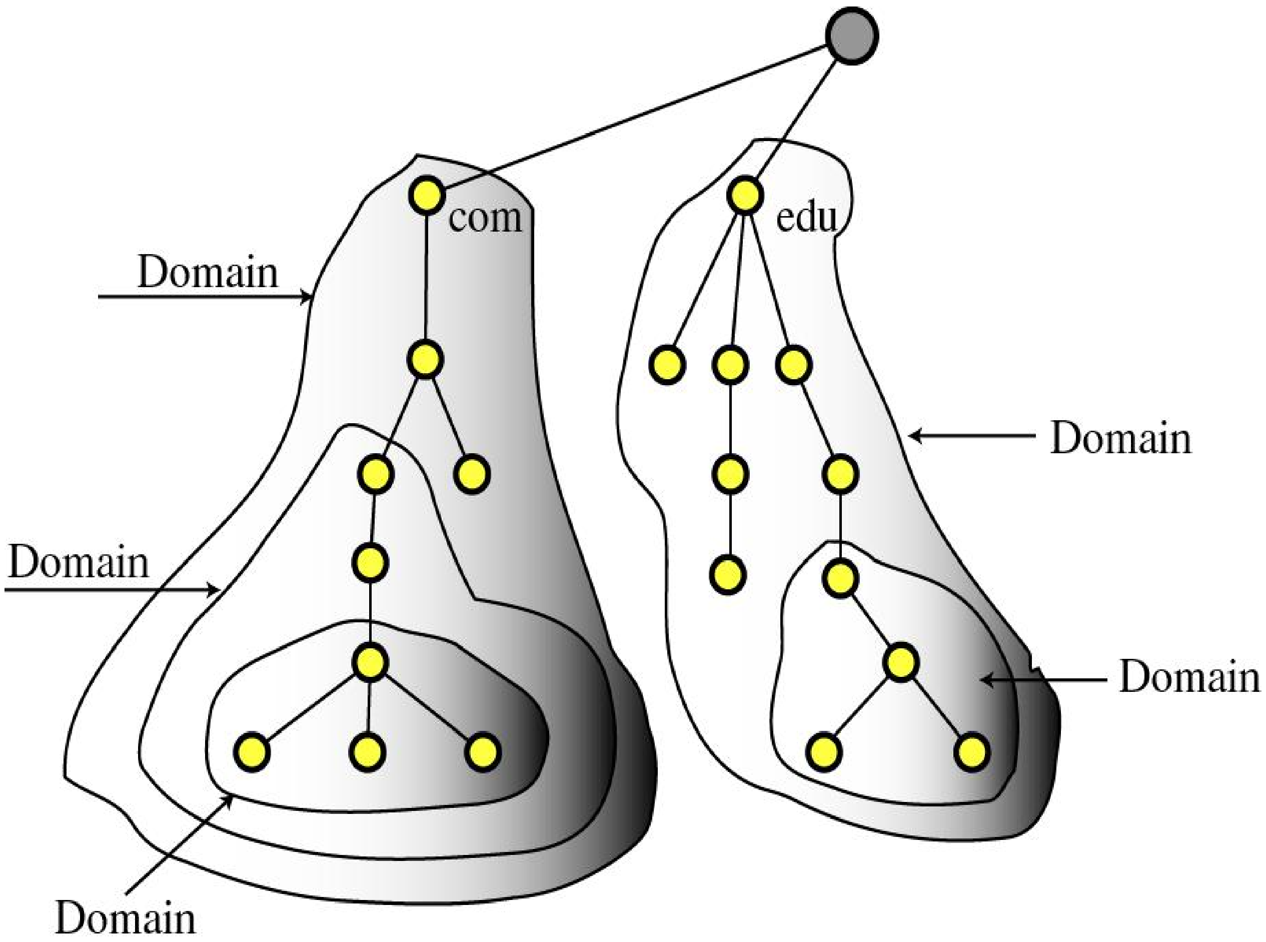
Top-level Internet Domains

Domain name	Description
com	Commercial organizations
edu	Educational institutions
gov	Government institutions
int	International organizations
mil	Military group
net	Network Support Centres
org	Organizations other than those above
arpa	Temporary ARPANET domain (obsolete)
country code	Domain for each country

Domain names and labels



Domains (sub-trees)



Mapping names to addresses

Mapping names to addresses

- DNS includes an efficient, reliable, distributed system for mapping names to addresses
  - Name-address mapping information (say, a table) partitioned into disjoint sub-tables and distributed throughout the Internet

- Mapping mechanism consists of independent, cooperative systems called name servers
  - A name server is a server program that supplies domain name to IP address translation service
  - Listens on UDP or TCP port 53

Zones and Name Servers

- Namespace hierarchy partitioned into disjoint sub-trees called zones
  - A zone can be a single domain, or a set of domains

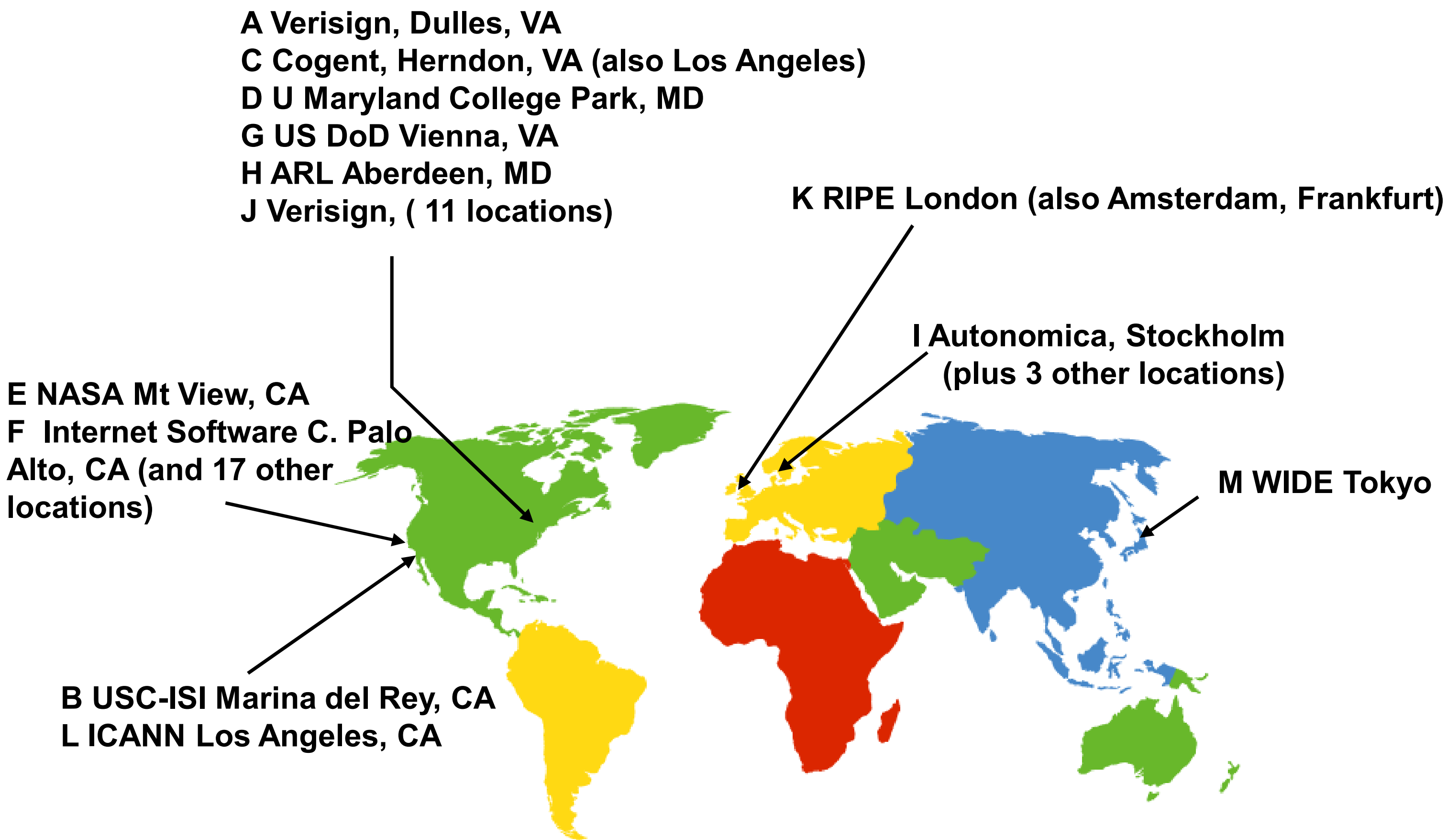
- Each zone corresponds to some administrative authority that has authority over this sub-tree
  - Name-address mappings for hosts in a zone is stored in two or more name servers maintained by the administrative authority for this zone

What does a name server store?

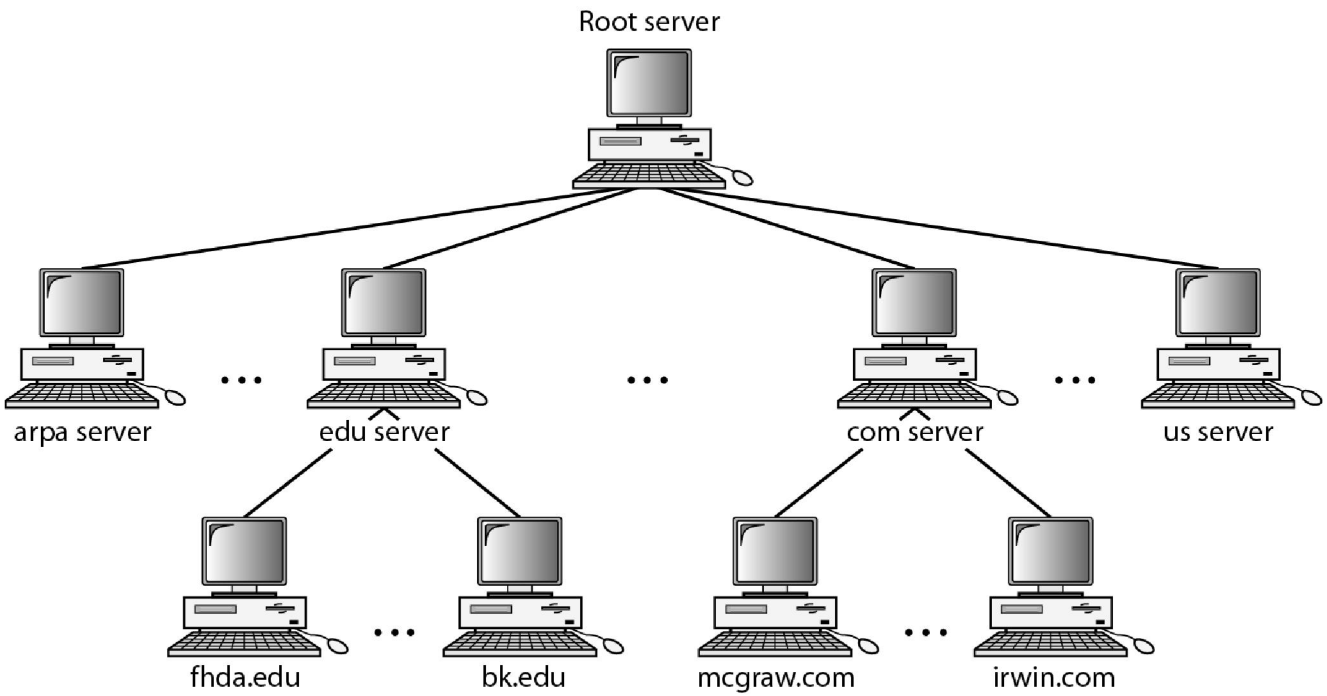
- Name-address mapping of all hosts in the corresponding zone
- Name-address mappings for all name servers corresponding to the domains immediately below itself (which are separate zones)

- Root name servers
  - 13 root servers (IP addresses well-known)
  - Each root name server stores the name-IP mappings of the name servers for the top level domains
  - DNS requires that every name server knows how to contact one or more root servers

DNS Root Name Servers



Hierarchy of name servers



Name Resolution

- Each host has a software for resolving names – DNS clients called "name resolvers"

- Client software, called a name resolver, uses one or more name servers while translating a name
  - Clients send queries to name servers, name servers respond with the requested information

- A client must know how to contact at least one name server (usually a "local name server")

Name resolution: what client does

- Client (name resolver) forms a domain name query that contains
  - The name to be resolved
  - A declaration of the class of the name (which is almost always the “Internet” class)
  - The type of answer desired
  - A code that specifies whether the name server should translate the name completely (recursive or iterative)

Name resolution: what server does

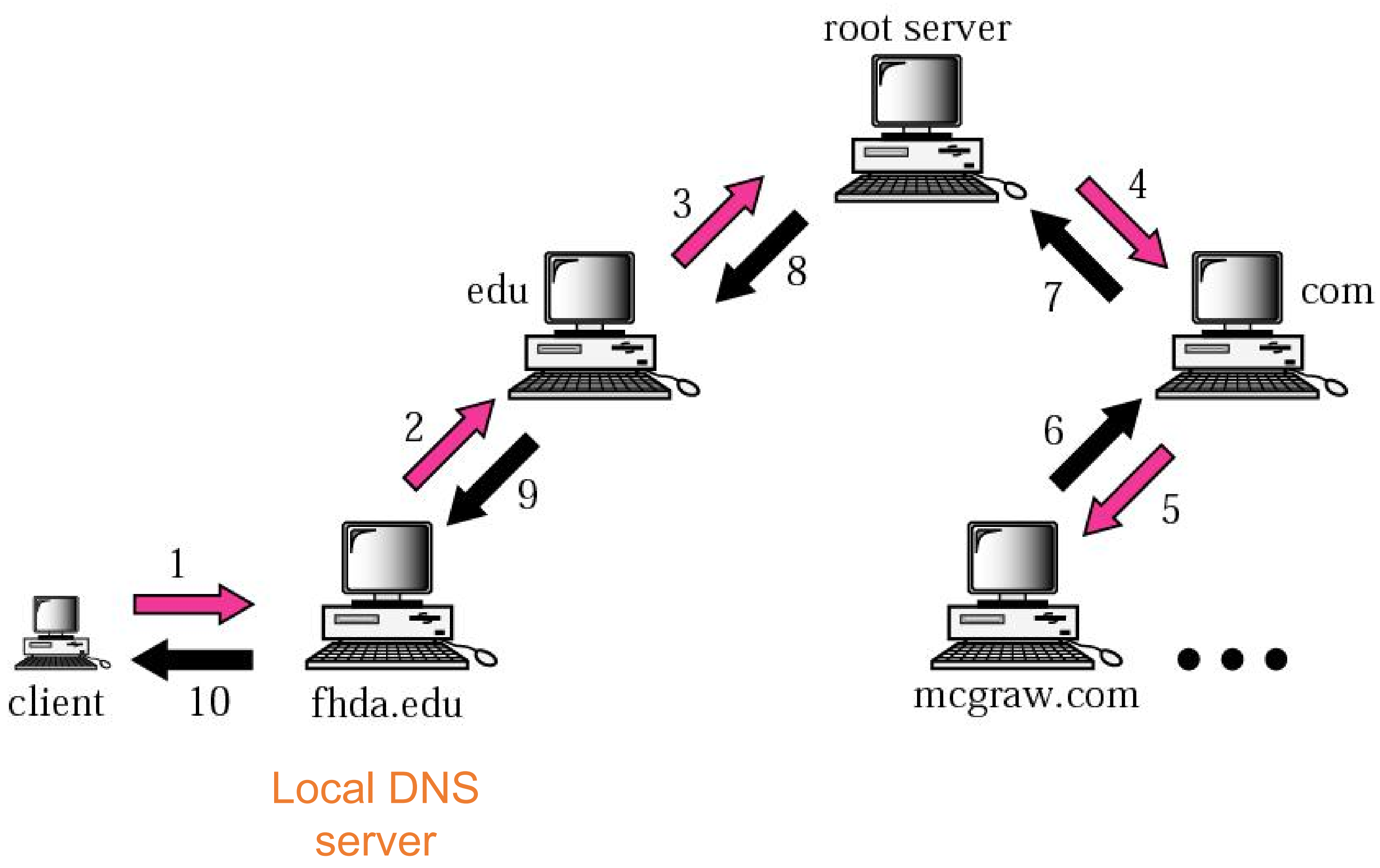
- When a name server receives a query:
- If the name lies in the domain for which it is an authority, translate the name to an address and send answer back to the client
- Otherwise, check what type of interaction the client specified
  - If recursive resolution, contact a name server that can resolve the name, get the answer and return the answer to the client
  - If iterative resolution, inform the client the name server that the client should contact next to resolve the name

Recursive vs. Iterative resolution

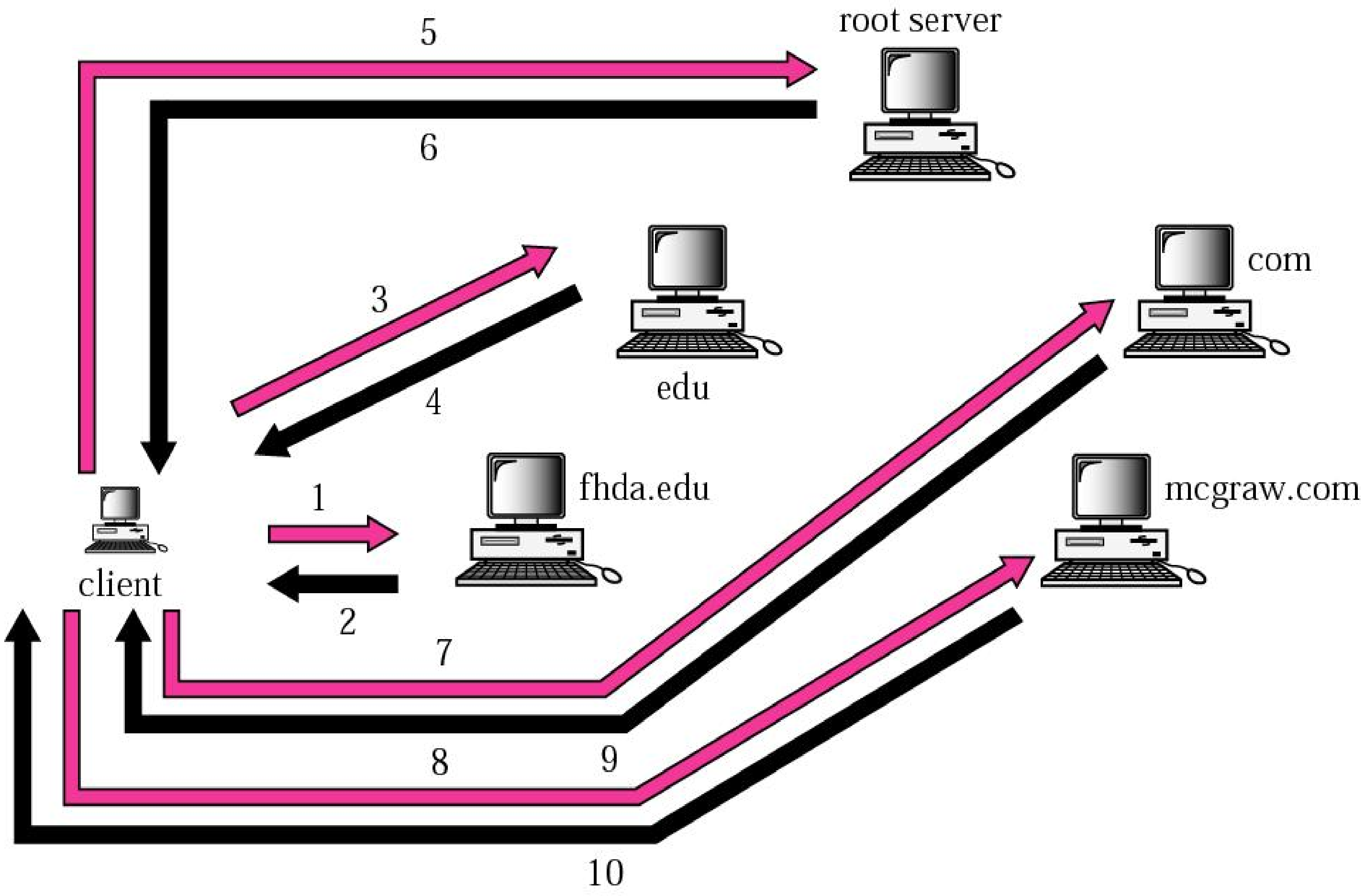
- Recursive resolution
  - Each contacted name server contacts the next name server (if necessary) itself
  - Local name server gets final answer directly
  - Puts more burden on higher level name servers

- Iterative resolution
  - Each contacted name server replies with name of server to contact next
  - Local name server contacts each remote server

Recursive DNS



Iterative DNS



DNS Caching

- Each name server caches recently resolved names, as well as a record of where the mapping information for that name was obtained

- Servers report cached information to clients, but mark it non-authoritative and give the name-address of the server S from which the binding was obtained
- Client may choose to use the non-authoritative information, or may choose to contact S to verify if binding is still valid

How long are cached entries valid?

- When an authoritative name server responds to a request, it includes a time to live (TTL) value in the response
  - Specifies how long it guarantees the binding to remain valid

- Other servers / hosts can cache these mappings, but must dispose of them once the TTL expires

Type associated with domain names

- Each named item is assigned a type
  - Specifies whether it is the address of a machine, a mailbox, a user, and so on

- When a client asks the DNS to resolve a name, it must specify the type of answer desired

DNS Resource Records

RR format: `(name, value, type, class, 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 = CNAME
  - **`name`** is alias name for some "canonical" (the real) name `www.ibm.com` is really `servereast.backup2.ibm.com`
  - **`value`** is canonical name
- Type = MX
  - **`value`** is name of mailserver associated with **`name`**

DNS Resource Records (contd.)

- NS records

  <princeton.edu, cit.princeton.edu, NS>

  <cit.princeton.edu, 64.45.192.233, A>

- CNAME records used to define aliases
  - www.cs.princeton.edu is an alias for the host named cicad.cs.princeton.edu

  <www.cs.princeton.edu, cicad.cs.princeton.edu, CNAME>

  <cicad.cs.princeton.edu, 192.12.69.55, A>

- MX records specify the host running the mail server for the domain
  - Mail server for cs.princeton.edu domain runs on the host named gnat.cs.princeton.edu

  <cs.princeton.edu, gnat.cs.princeton.edu, MX>

  <gnat.cs.princeton.edu, 192.12.69.60, A>

Contents of DNS messages

- Each question consists of a domain name for which the client seeks an IP address, a specification of the query class (usually Internet class) and the type of object desired
- Server responds by sending a similar message that contains
  - answers to the questions for which the server has bindings
  - If the server cannot answer all questions, response contains information about other name servers that the client can contact to get the mappings
  - Information about the servers that are authorities for the mappings given as replies, and the IP addresses of those servers

References

- *Data Communications & Networking, 5th Edition, Behrouz A. Forouzan*
- *Computer Networks, Andrew S. Tanenbaum and David J. Wetherall*
- *Wikipedia*