CS2105 CheatSheet

by Zachary Chua

Internet

- The Internet is a network of connected computing devices
- Devices are known as **hosts** or **end systems**
- Hosts run network applications (like browsers) and communicate over links

Network Edge (Access Network)
- Hosts access the Internet through **access network**
- eg Home/ Institute access networks

Wireless Access Network
1. Wireless LAN (WIFI): Short range (100 ft)
2. Wide-area wireless access (3G/ 4G): Long range (10s km)

Physical Media
Host connect to access network via physical media - Guided media: signals propogate in solid media (ethernet cable/ fibre optics)
- Unguided media: signals propagate freely (radio)

Network Core
A mesh of interconnected routers
Data transmitted by
1. Circuit Switching: dedicated circuit per call
2. Packet Switching: data sent through net in discrete "chunks"

Circuit Switching
End-to-end resources **allocated to and reserved for** "call" between source and dest
- call setup required
- circuit-like (**guaranteed**) performance
- circuit segment idle if not used by call
- used in traditional telephone networks
- limited capacity

Packet Switching
Resources are used on demand → no reservation and excessive congestion is possible
Performance not guaranteed
Host sending function
- breaks application message into smaller chunks, known as **packets** of length **L** bits
- transmits packets onto the link at **transmission rate R**
- link transmission rate is known as **link capacity** or **link bandwidth**

Packet Transmission Delay = $\frac{L}{R}$, assuming packet size L bits and link bandwidth R bits/sec

Store and Forward: entire packet must arrive at a router before it can be transmitted on the next link (check packet integrity; if corrupted, drop packet)

Routing and Addressing
Where to forward the packet
- Routers determine the source-destination route taken by the packet (using routing algorithms)
- Addressing: each packet needs to carry source and destination information

Delay and Loss
Loss: - Packets queue in router buffers - wait for turn to be sent out one by one
- if packet arrival rate exceeds departure rate → buffer full and drop packet

4 sources of delay

1. Nodal Processing - check bit error, determine output link
2. Queueing Delay - time waiting in queue for transmission
3. Transmission Delay - L/R, where L is packet length in bits, R is link bandwidth in bps
4. Propagation Delay - d/s, where d is length of link, s is propagation speed in medium

Throughput
How many bits can be transmitted per unit time
- Measured for end-to-end commmunication
- Link capacity (bandwidth) is meant for a specific link

Protocols
Defines format and order of messages exchanged and the actions taken after messages are sent or received

Internet Protocol
1. Application - supports network applications
2. Transport - process to process data transfer
3. Network - routing of datagrams from source to destination
4. Link - data transfer between neighbouring network elements
5. Physical - bits on wire

Application Layer

Structure of Network Application
- Client-Server
- Peer-to-peer

Client-Server
Server:
- Wait for incoming request
- Provides requested service to client
Client:
- Initiates contact with and request service from server

P2P
- No always on server
- peer request service from other peers, provides service to other peers
- self-scalable
- complex management - peers are intermittently connected and change IP addresses

Transport Service does an app need
- Data Integrity
- Throughput - Some apps may need minimum bandwidth (streaming videos)
- Timing - might need low latency
- Security - excryption

Transport Layer Protocol
App-layer protocols ride on transport layer protocols

1. TCP
- Reliable data
- Flow Control: sender won't overwhelm receiver
- Congestion Control: throttle sender when network overloaded
- does not provide: timing, minimum throughput, security

2. UDP
- Unreliable data
- no flow control
- no congestion control
- does not provide: timing, throughput, security

Web
Webpage typically consists of
- base HTML file
- several referenced objects
Each object is addressable by a URL

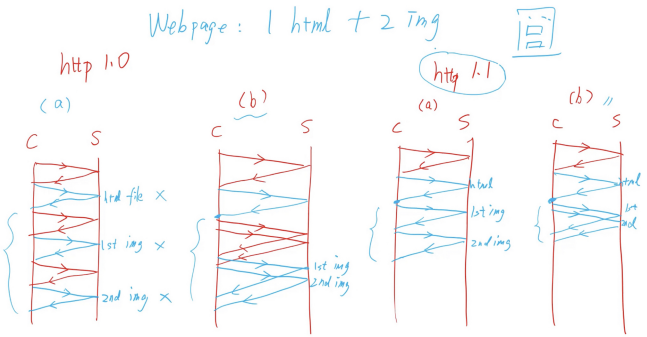
HTTP - Web app layer protocol
C/S model
- client in browser
- server on port 80 (default port number for web servers)
Uses TCP 1. Client initiates TCP connection to server
2. Server accepts TCP connection request
3. HTTP Messages are exchanged over TCP connection
4. TCP connection closed
1 and 2 are known as TCP handshaking
RTT: Round trup time, time for a packet to travel from client to server and go back.

Non-Persistent HTTP
- At most one object sent over TCP connection; TCP connection closed after object sent
- downloading multiple objects requires multiple connections

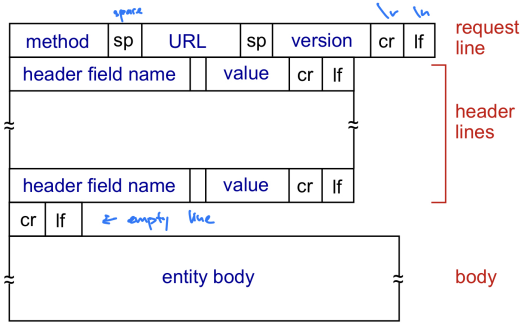
Persistent HTTP
- multiple objects sent over single TCP connection between client and server

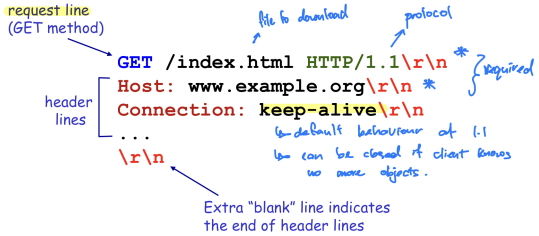
(a): Non-persistent
- $2 * RTT+$ file transmission time for each object (1 RTT for TCP, 1 RTT for object)
(b): Non-persistent with pipelining
- $2 * RTT+$ file transmission time for HTML page
- as little as $2 * RTT+$ file transmission time for other objects (concurrently)

(c): Persistent
- 1 RTT for TCP
- 1 RTT for each object
(d): Persistent with pipelining
- 1 RTT for TCP
- 1 RTT for HTML
- as little as 1 RTT for all referenced object



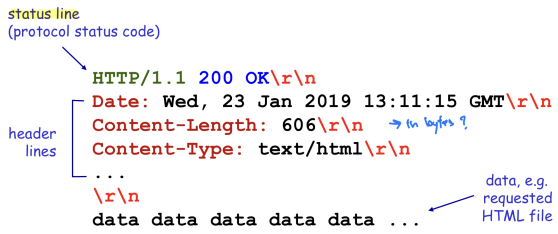
HTTP Request





Request Method Types:
1.0: GET, POST (upload input to server), HEAD (leave out requested object)
1.1: GET, POST, HEAD, PUT (uploads file to path specified in URL field), DELETE (deletes file specified in URL field)

HTTP Response
Response Status Codes:
- 200: OK
- 301: Moved Permanently (new location specified later in message)
- 403: Fobidden
- 404: Not found
- 304: Not Modified



Cookies
HTTP designed to be stateless - server maintains no information about past client requests
Cookie: http messages carry state (forever, expiry, memory only)
1. cookie header field of http request/ response messages
2. cookie file kept on user's host, managed by user's browser
3. backend database at website

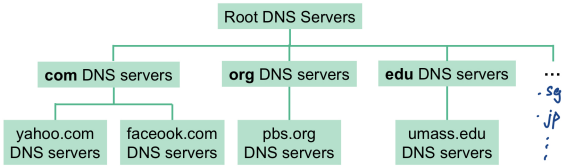
Conditional Get
- Goal: don't send object if client cache has up to date cached version
- cache: specify date of cached copy in http request; `If-modified-since: <date>`
- server: response contains no object if cached copy is up to date; response is 304

Domain Name System
Identify host by: 1. Hostname: eg, www.example.org
2. IP address: eg, 93.184.216.34 (32-bit int)
DNS translates between the two
Client carries out DNS query to determine the IP address corresponding to server name prior to connection
NOTE: 1 hostname could map to many IP addresses (for load balancing purposes)

DNS Resource Record (RR)
Mappings are stored as RR
RR format: `(name, value, type, ttl)`

Types: 1. A - name is hostname; value is IP addr
2. NS (name server) -name is domain (eg. nus.edu.sg); value is hostname of authoritative name server for domain.
3. CNAME - name is alias name (eg. www.nus.edu.sg) for some canonical (real) name; value is canonical name (eg. mgnzsqc.x.incapdns.net)
4. MX (mail exchanger) - value is the name of mail server associated with name

Distributed Hierarchical Databases
DNS is stored in RR in distributed databases implemented in hierarchy of many name servers



A client wants IP address for www.facebook.com:
- client queries root server to find .com DNS server
- client queries .com DNS server to get facebook.com DNS server
- client queries facebook.com DNS server to get IP address for www.facebook.com

Root servers
- Answers request for records in the root zone by returning a list of the authoritative NS for the appropriate top level domain (TLD)

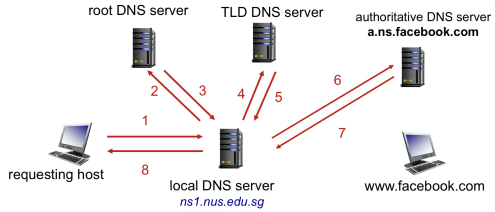
Top Level Domain
- responsible for com, org, net, edu, etc and all top-level country domains, eg. sg, jp, etc.

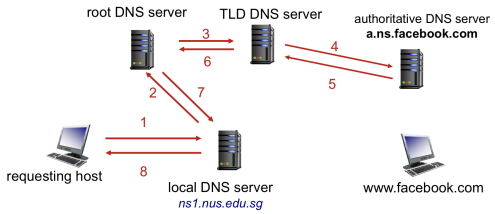
Authoritative Servers
- Organisation's own DNS server(s), providing authoritative hostname to IP mappings for organisation's named hosts (eg. web, mail)

Local DNS server
- Does not strictly belong to hierarchy
- Each ISP has one local DNS server (aka default name server)
- When host makes DNS query, query is sent to local DNS server
- Retrieve name-to-address translation from local cache
- Local DNS server acts as proxy and forwards query into hierarchy if answer not found locally

DNS Caching
Caches mapping once it learns of it
- may be out of date
- cached entries expire after some time (TTL)
- if name host changes IP address, may not be known Internet-wide until TTL expire
Runs over UDP

DNS name resolution
iterative and recursive





NOTE: Recursive rarely used as servers cannot respond to other queries until it recieves an answer, eg root DNS cannot answer other queries until TLD server replies

Socket Programming
Process: program running within a host
- Within host, two processes communicate using inter-process communication (defined by OS)
- Processes in different hosts communicate by exchanging messages (according to protocols)
- Identified by (IP address, port number) (port number is a 16 bit int, 1-1023 are reserved)

Sockets
Socket is the software interfrace between app processes and transport layer protocols - Process send/ receives messages to/ from its socket
- Programming-wise: a set of APIs
Application treat the Internet as a black box, sending and receiving messages through sockets

Types of sockets:
1. TCP: reliable, byte stream-oriented socket
2. UDP: unreliable datagram socket

Socket Programming with UDP
UDP: no connection between client and server
- Sender explicitly attaches destination IP addr and port number to each packet
- Receiver extracts sender IP addr and port number from the received packet
1 socket listens to multiple clients.

Socket Programming with TCP
- When client creates socket, client TCP establishes a connection to server TCP
- When contacted by client, server TCP creates new socket for server process to communicate with that client
This allows server to talk with multiple clients individually.
TCP socket pairs (?) are uniquely identified by (server IP, server port, client IP, client port)

TCP socket vs UDP socket
- TCP: two processes communicate as if there is a "pipe" between them. Send data → write data to pipe, no need to attach dest IP and port to each packet. Pipe is there until one of the two processes closes it. Pipe is also reliable
- UDP: Form UDP packet explicitly and attach dest IP and port no to every packet.

Transport Layer
Deliver messages between application processes running on different hosts
- TCP and UDP

What do they do
- Sender: break app message into segments, passes them to network (IP) layer
- Receiver: reassembles segments into message, passes to app layer
- Routers in between: check dest IP to determine routing

Note: - Each IP datagram contains source and dest IP addr
- Each IP datagram carries one transport layer segment
- Each segment contains src and dest port numbers

Connectionless Transport: UDP
UDP adds the following services (very few)
- Multiplexing at sender: UDP gathers data from processes, forms packets and passes them to IP
- De-multiplexing at receiver: UDP receives packets from lower layer and sends them to right processes
- Checksum
NOTE: UDP transmission is **unreliable**
Reasons for UDP:
- No connection establishment (less delay)
- Simple: no connection state at sender, receiver
- small header size
- No congestion control: UDP can blast away as fast as desired

Connectionless Demultiplexing
When UDP receiver recieves UDP segment
- Checks dest port no of segment
- Directs UDP segment to the socket with that port number
- IP datagrams (from different soucres) with same destination port no. will be directed to the same UDP socket at destination

UDP Header
Comprises of (8 bytes / 64 bits)
- Source port no. (16)
- Dest port no. (16)
- length in bytes, including header(16)
- checksum (16)

UDP Checksum
To detect errors (flipped bits) in transmitted segment

Sender:
- Compute checksum value
- put checksum value in UDP checksum field
Receiver:
- Compute checksum of received segment
- Check if computed value = checksum
NOTE: Just because checksum = computed value does not mean that there is no error

Computation:
1. Treat segment as a sequence of 16-bit integers
2. Apply binary addition on every 16 bit integer (checksum field is currently 0)
3. Carry (if any) from the MSB will be added to the result
4. Compute 1s complement to get UDP checksum

Reliable Data Transfer
Network layer is unreliable, need to build reliable data transfer over this
Network layer may:
- corrupt, drop, re-order (not considered in this mod) packets, and deliver packets after an arbitrarily long delay

rdt 2.0: Bit Errors
Solution: Use checksum to detect bit errors, receiver replies with ACK/NAK to acknowledge/ negative acknowledge
- ACK: send next packet
- NAK: resend current packet
Problem: ACK may be corrupted. Sender would resend packet, receiver does not know it is duplicate

rdt 2.1: Bit Errors
Solution: rdt 2.0 with packet sequence number
- ACK/NAK corrupted: Sender retransmits current packet
- Sender adds sequence number to each packet.
- Receiver discards (doesnt deliver to application) duplicate packets.

rdt 2.0: Bit Errors
Modification: NAK free. Receiver sends ACK for last packet received OK. ACK now contains seq no.
Duplicate ACK: resend current packet

rdt 3.0: Errors and Loss
Channel may now:
- flip bits
- lose packets
- arbitrarily long delay between packets
- but WON'T reorder packets
Solution: Sender waits for some time for ACK. Sender retransmits if no ACK is received till timeout
NOTE: If packet or ACK is just delayed not lost;
- Timeout triggers retransmission
- Retransmission generates duplicates, receiver uses seq no to detect
- Reciever must specify the packet seq no of the packet being ACKed
NOTE: Sender does not retransmit on ACK with wrong packet no. (just waits for timeout)

Performance of rdt 3.0 (bad)
- Example: packet size = 8000 bits, link rate = 1 Gbps:

$$d_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 0.008 \text{ msec}$$

  - If RTT = 30 msec, sender sends 8000 bits every 30.008 msec.

$$\text{throughput} = \frac{L}{RTT + d_{trans}} = \frac{8000}{30.008} = 267 \text{ kbps}$$

  - U_{sender}: *utilization* – fraction of time sender is busy sending

$$U_{sender} = \frac{d_{trans}}{RTT + d_{trans}} = \frac{0.008}{30 + 0.008} = 0.00027$$

Pipelined Protocols
Allows multiple, yet-to-be-ACKed packets
Requires more seq no, and buffering at both sides
2 generic forms:
- Go-Back-N (GBN)
- Selective Repeat (SR)
Same assumptions as rdt 3.0

GBN
GBN Sender:
- Can have up to N unACKed packets in pipeline
- insert k-bits seq no in packet header
- use "sliding window" to keep track of unACKed packets
- keep a timer for the oldest unACKed packet
- on timeout, retransmit all packets in sender window
- `timeout(n)`: retransmit packet n and all subsequent packets in the window

GBN Receiver:
- In order packets: ACK (simple receiver - only needs to keep track of the next expected seq no)
- Out of order packets: Discard and ACK the last in order packet
- Cumulative ACK: `ACK m` means all packets up to m are received

SR
SR Sender:
- maintains timer for each unACKed packet
- When timer expires, just retransmit only that unACKed packet

SR Receiver:
- Individually acknowledges all correctly received packets
- buffers out of order packets, as needed, for eventual in-order delivery to upper layer

sender
data from above:
- if next available seq # in window, send pkt

timeout(n):
- resend pkt n, restart timer

ACK(n) in [sendbase, sendbase+N]
- mark pkt n as received
- if n is smallest unACKed pkt, advance window base to next unACKed seq. #

receiver
pkt n in [rcvbase, rcvbase+N-1]
- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N, rcvbase-1]
- ACK(n)

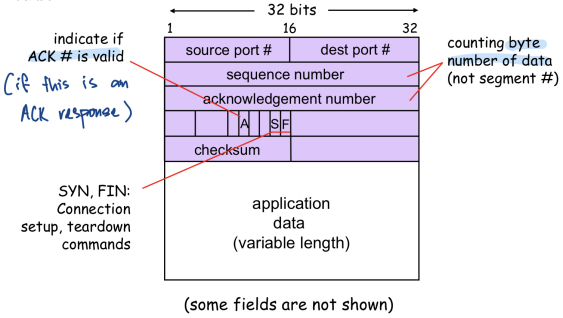
otherwise:
- ignore

TCP
Overview:
- Point-to-Point: 1 sender, 1 receiver
- Connection-oriented: handshaking before sending app data
- Full duplex service: bi-directional data flow in the same connection
- Reliable, in-order byte stream: use seq no. to label bytes

Connection Oriented De-mux:
A TCP connection socket is identified by the 4-tuple (srcIP, srcPort, destIP, destPort)

Segment Size:
A TCP segment typically has an MSS (maximum segment size) of 1460 bytes (app data, without header)

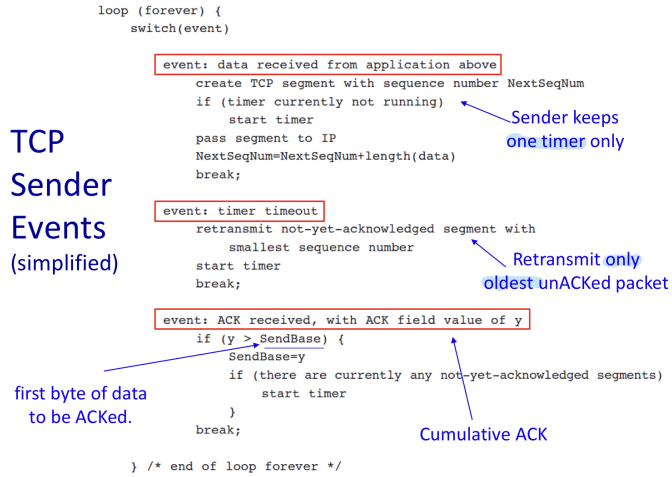
Header:



Sequence Number:
Byte number of the first byte of data in a segment

ACK (cumulative):
TCP ACKs up to the first missing byte in the stream (next expected byte number)
Eg. If segment is [0-999], receiver would ACK 1000
NOTE: TCP does not specify how receiver should handle out of order segments, it is up to implementer.

TCP Sender Events

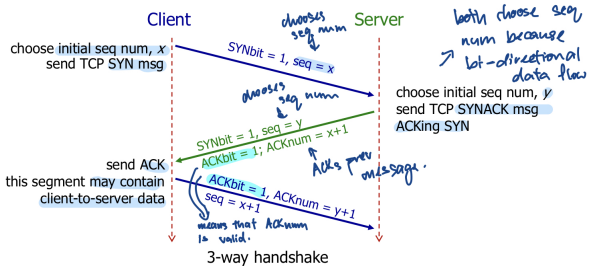


TCP Receiver Events

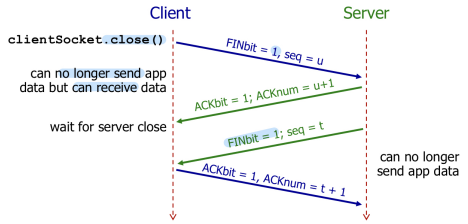
Event at TCP receiver	TCP receiver action	
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK: wait up to 500ms for next segment. If no next segment, send ACK	
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments	
Arrival of out-of-order segment higher-than-expect seq. # (gap detected)	Immediately send duplicate ACK, indicating seq. # of next expected byte	
Arrival of segment that partially or completely fills gap	Immediately send ACK, provided that segment starts at lower end of gap	

TCP Fast Retransmission
If sender receives 4 duplicate ACKs (ACK for the same segment), it supposes the segment is lost.
Immediately resend segment, even before timer expires

Establishing connection - 3-way Handshaking



Closing Connection
- Client, server each close their side of connection.
  - send TCP segment with FIN bit = 1



Network Layer
host to host

IP Address
IP Address is used to identify a host or router
32-bit integer expressed in either binary or decimal
Hosts get IP addresses via:
1. Manual configuration by system administrator
2. Automatically assigned by Dynamic Host Configuration Protocol (DHCP) server
DHCP is more scalable
NOTE: IP addresses are associated with a network interface
- each network interface has its own IP, eg laptop with ethernet port and wifi will have 2 IP addresses
- routers typically have many interfaces

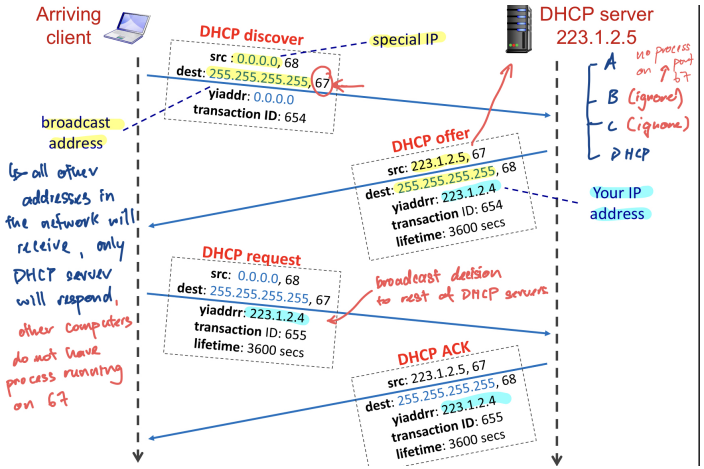
DHCP

Allows a host to dynamically obtain its IP address from DHCP server when it joins network

- IP addr is renewable
- allows reuse of IP addresses
- supports mobile users joining network

4 step process:

1. Host boradcasts "DHCP discover" message
2. DHCP server responds with "DHCP offer" message (note that a host may get more than 1 offer)
3. Host requests IP addr with "DHCP request" message
4. DHCP server sends address with "DHCP ACK" message



NOTE: DHCP may also provide additional network information

- IP address of first-hop router (default gateway)
- IP address of local DNS server
- Network mask

NOTE: DHCP runs over UDP, server on port 67, client port 68

Special IP Addresses

Special Addresses	Present Use
0.0.0.0/8	Non-routable meta-address for special use
127.0.0.0/8	Loopback address. A datagram sent to an address within this block loops back inside the host. This is ordinarily implemented using only 127.0.0.1/32.
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	Private addresses, can be used without any coordination with IANA or an Internet registry.
255.255.255.255/32	Broadcast address. All hosts on the same subnet receive a datagram with such a destination address.

Subnets

IP addresses comprises of two parts:

1. network (subnet) prefix
2. host ID

Subnet is a network formed by a group of "directly" interconnected hosts

- hosts in the same subnet have the same network prefix
- hosts in the same subnet can physically reach each other without intervening router
- They connect to the outside world through a router

CIDR

Classless Inter-domain Routing is the Internet's IP address assignment strategy

- subnet prefix is of arbitrary length
- address format: **a.b.c.d/x**, where x is the number of bits in subnet prefix

Subnet Mask

Subnet Mask is used to determine which subnet an IP address belongs to

- Made by setting all subnet prefix bits to 1s and host ID bits to 0s

and with IP address to get subnet prefix

Hierarchical Addressing

Allows efficient advertisement of routing information

- instead of having a routing table with 2^{32} entries, use block allocations
- Eg. A router could broadcast "send me anything with addresses beginning with 200.23.16.0/20"

Can have smaller and more efficient routing tables

NOTE: These are not subnet masks/ prefixes!!

Routers use **longest prefix match** to determine where to route to

Routing Algorithms - Intra AS routing

- AS: Autonomous System
- View network as a graph of nodes (routers) and edges (links)
- Each link can have an associated cost eg. money, congestion, avail bandwidth etc.

Goal: find least cost path between two vertices in graph

"link state" algorithms

- All routers have complete knowledge of network topology and link cost;

Routers periodically broadcast link costs to each other

- Use Djikstra's Algo to compute least path (using this global map)

"distance vector" algorithms

- Routers know physically-connected neighbours and link costs to neighbours
- Routers exchange local views with neighbours and update own local views based on neighbours'
- Iterative process of computation: Swap local view with neighbours, update own local view, repeat until no change
- Key Difference: Only broadcast link costs to immediate neighbours

Graph Notations:

- $c(x, y)$: cost of link between routers x, y, ∞ if not direct neighbours
- $d(x, y)$: cost of least-cost path between routers x, y

Bellman-Ford Equation

$$d_x(y) = min_v c(x, y) + d_v(y)$$

where min is taken over all direct neighbours v of x

Intuitively, shortest path from x to y is the minimum of least cost to each neighbour of x + least cost path from neighbour to y

- To find the least cost path, x needs to know the cost from each of its direct neighbours to y
- Each neighbour v sends its distance vector (y, k) to x, telling x that the cost from v to y is k

Algorithm Steps

1. Every router **x, y, z** sends its distance vectors to its directly connected neighbours.
2. When x finds out that y is advertising a path to z that is cheaper than x currently knows, x will
   2.1 update its distance vector to z accordingly
   2.2 note that all packets for z should be sent to y. This info is used to create x's forwarding table
3. After several rounds, all routers know least-cost paths to all other routers

Routing Information Protocol (RIP)

- RIP implements DV algo, using hop count as cost metric (insensitive to network congestion)
- Exchange routing table every 30 seconds over UDP port 520
- "self-repair": is no update from a neighbour router for 3 mins, assume neighbour has failed, remove neighbour entry from table

Network Address Translation (NAT)

Public IP: unique, 2^{32} addresses, many are reserved

Private IP: used within a network, NOT globally unique, ie. can't be used outside network

NAT router

NAT routers sit between a local network and wider internet

- Replace (src IP addr, port no.) of every outgoing datagram to (NAT IP addr, new port no.)
- Remember in NAT translation table, the mapping from (src IP addr, port no.) to (NAT IP addr, new port no.)
- Replace (NAT IP addr, new port no.) in dest fields of every incoming datagram with corresponding (src IP addr, port no.) stored in NAT translation table

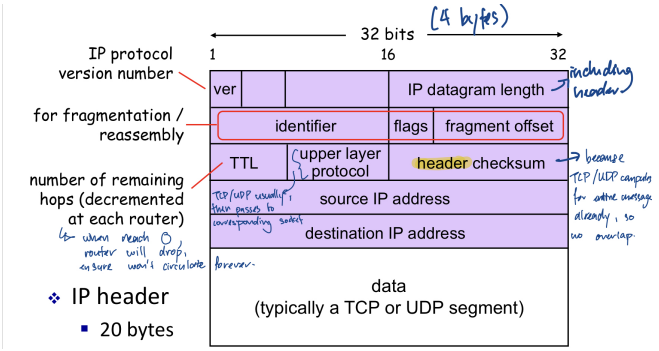
Effect: all datagrams leaving local network have the same src NAT IP addr (new port no. are not the same)

Benefits of using NAT

- No need to rent a range of public IP addr from ISP; just one for NAT router
- All hosts use private IP addr. Can change address of hosts (within network) without notifying outside world
- Can change ISP without changing addresses of hosts in local network
- Hosts insde local network are not explicitly addressable and visible by outside world (security)

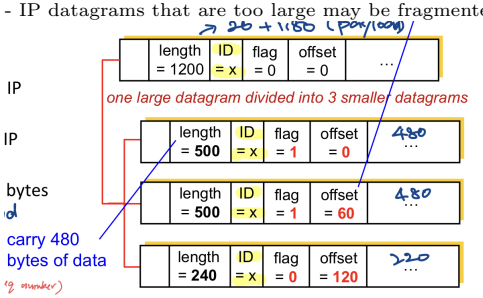
IPv4 Datagram Format

header size: 20 bytes



IP Fragmentation and Reassemby

- Different links may have different MTU (Max Transfer Unit): max amount of data a link layer frame can carry (max size of IP datagram)
- IP datagrams that are too large may be fragmented by router



- Flag = 1 if there is next fragment in same segment, 0 if last fragment
- Offset is expressed in units of 8-bytes; first byte num / 8 (relative ordering between fragments)
- ID remains the same for all fragments

Internet Control Message Protocol (ICMP)

Used by hosts and routers to communicate network-level info

- Error reporting eg. unreachable host/ network/ port/ protocol
- Echo request / reply (`ping`)

ICMP messages are carried in IP datagrams (ICMP header starts after IP header)

- ICMP header: Type + Code + Checksum + others.

Type	Code	Description
8	0	echo request (ping)
0	0	echo reply (ping)
3	1	dest host unreachable
3	3	dest port unreachable
11	0	TTL expired
12	0	bad IP header

Selected ICMP Type and subtype (Code)

eg. TTL reaches 0, packet discarded, ICMP error message sent to datagram's src addr