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

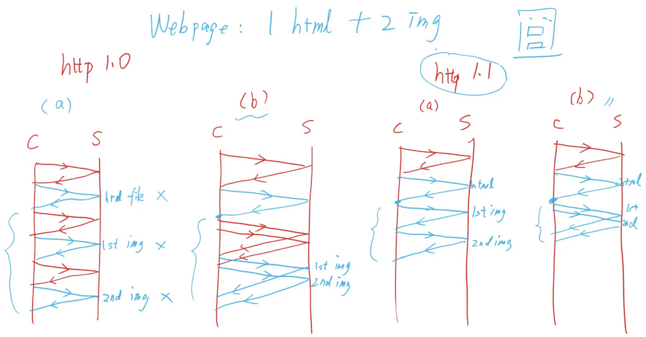
Persistent HTTP

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

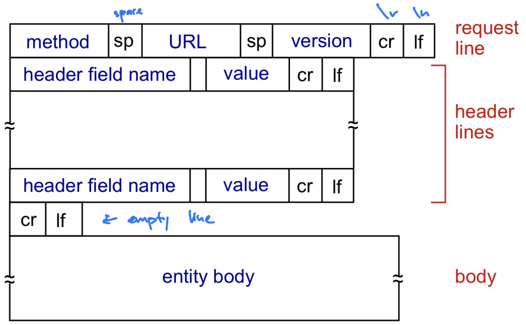
(a): Non-persistent
- $2 * RTT+$ file transmission time for each object (1 RTT for TCP, 1 RTT for object)
(b): Non-persistent with pipelining
- $2 * RTT+$ file transmission time for HTML page
- as little as $2 * RTT+$ file transmission time for other objects (concurrently)

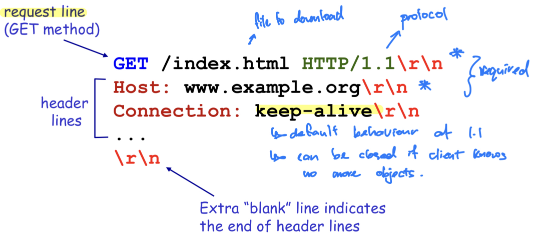
(c): Persistent
- 1 RTT for TCP
- 1 RTT for each object

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



HTTP Request





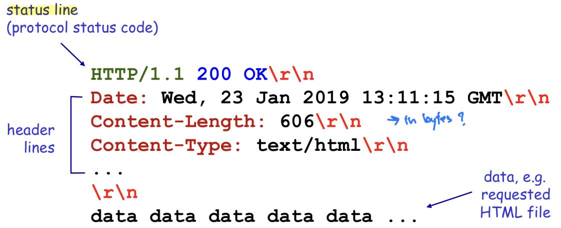
Request Method Types:

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

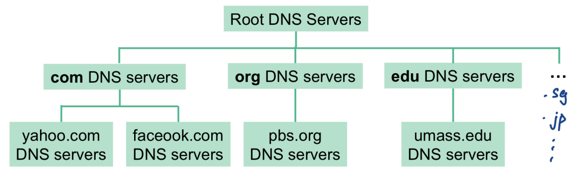
2. NS (name server) -name is domain (eg. nus.edu.sg); value is hostname of authoritative name server for domain.

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

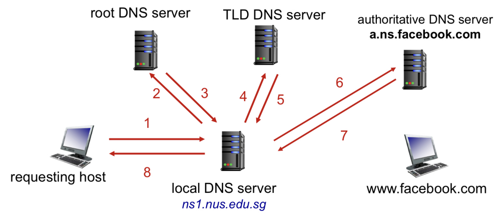
DNS Caching

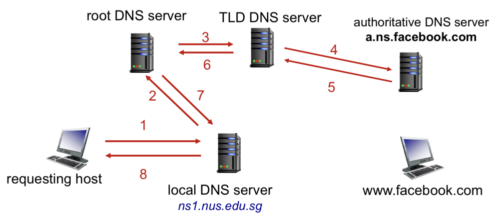
Caches mapping once it learns of it
- may be out of date
- cached entries expire after some time (TTL)
- if name host changes IP address, may not be known Internet-wide until TTL expire

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**