CS2105

AY22/23 Sem 2

github.com/jasonqiu212

01. Introduction

- **Network Edge** - Hosts (Clients and servers)
- **Access Networks** - Wired and wireless communication links
- **Network Core** - Network of interconnected routers

Network Core

Packet-Switching

- Host breaks messages into packets of L bits
- Transmits packets into access network at transmission rate R (aka Link bandwidth, capacity)

$$\text{Packet Transmission Delay} = \frac{\text{Packet Size (bits)}}{\text{Transmission Rate (bits/sec)}}$$

- **Store and Forward** - Entire packet must arrive at router before being transmitted to next link

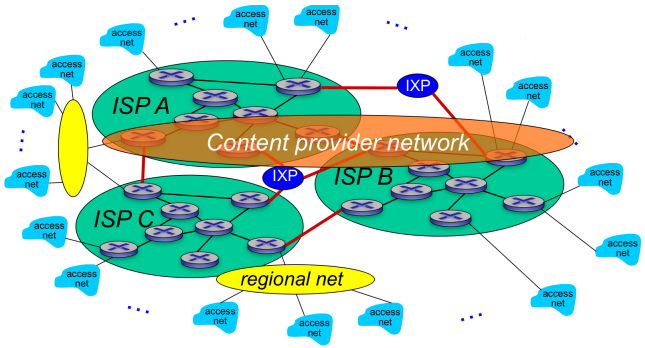
Key Functions of Network Core

- **Routing** - Determines source-destination routes taken by packets (How we get the hashtable)
- **Forwarding** - Move packets from router's input to correct router output

Circuit Switching

- Resources reserved for call between source and destination
- Pros: Betrer performance
- Cons: More resources

Internet Structure



- End systems connect to Internet via **Access Internet Service Providers (ISPs)**
- ISPs connect to larger global ISPs (usually competitors)
- Large ISPs connect via **peering links** or **internet exchange points (IXP)**
- **IXP** - Physical place with routers from different ISPs
- **Regional Networks** - Smallers ISPs
- **Content Provider Networks** - Provide content close to end users

Loss, Delay, and Throughput

Packet Loss

- If Arrival Rate $>$ Transmission Rate, packets will queue and can be dropped if queue fills up
- Solutions: Lost packets can be retransmitted

Packet Delay

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

- **Nodal Processing** - (d_{proc}) Check for bit errors and determine output link
- **Queueing Delay** - (d_{queue}) Time at queue waiting for transmission
- **Transmission Delay** - (d_{trans}) Time to load packet onto link
  - $d_{\text{trans}} = \frac{L}{R}$ where L is packet length and R is link bandwidth
- **Propagation Delay** - (d_{prop}) Time for 1 bit to reach end of link
  - $d_{\text{prop}} = \frac{d}{s}$ where d is length of link and s is propagation speed

Throughput

- Rate at which bits transferred between hosts
  - Different from transmission rate (Theoretical upper bound)
- Average: Rate over long period of time
- Instantaneous: Rate at given point in time

Protocol Layers and Service Models

- **Protocol** - Defines format, order of messages sent and received, and actions taken on message transmission
- Networks are complex with many components. How can we organize its structure?
- **Layering** - Each layer implements a service by doing something within layer and relying on services provided by layer below it
  - Explicit structure allows us to make sense of complex components
  - Easy maintenance (Like OOP, change in 1 layer should not affect others)

Internet Protocol Stack

1. Application
2. Transport
3. Network
4. Link
5. Physical

02. Application Layer

- Programs that run on end systems, and not on network-core devices

Client-server Architecture

- Server: Always-on host, Permanent IP address
- Clients: Communicates with server, Intermittently connected, Dynamic IP addresses, Do not communicate with each other directly

P2P Architecture

- Peers request service from other peers and provide service in return
- No always-on server, Intermittently connected, Dynamic IP addresses
- **Self Scalability** - New peers offer new services and demands

Process

- **Process** - Program running in host
- **Inter-process Communication** - How 2 processes in 1 host communicate
- **Messages** - Processes in different hosts communicate by exchanging this
- **Client Process** - Process that initiates communication
- **Server Process** - Process that waits to be contacted
- **Socker** - Process sends/receives messages to/from its socket (like a door)
  - Outside of socket, transport layer delivers message

Addressing Processes

- Motivation: IP address is not enough to address process, since many processes can be running on same host
  - **Identifier** - IP address and port number
  - **Port Number:** - Associated with process on host

Transport Protocol Services

1. **TCP** - Transmission Control Protocol
   - Reliable transport
   - Flow control: Sender does not overwhelm receiver
   - Congestion control
   - Connection-oriented: Setup required between client and server
2. **UDP** - User Datagram Protocol
   - Unreliable data transfer
   - Fast

App-layer Protocol

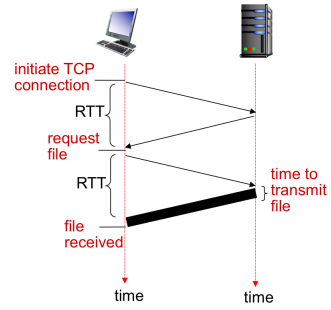
- Types of messages exchanged (e.g. Request or response)
- Message syntax: How fields are delineated
- Messages semantics: Meaning of information in fields

HTTP

- **Hypertext Transfer Protocol** - Web's application layer protocol
- Motivation: Web page consists of objects (HTML, images). Need method to request/send web objects.
- Follows client/server model
- Uses TCP
- **Stateless** - Server maintains no information about past requests

Non-persistent HTTP

- At most 1 object sent over TCP connection
- Downloading multiple objects requires multiple TCP connections

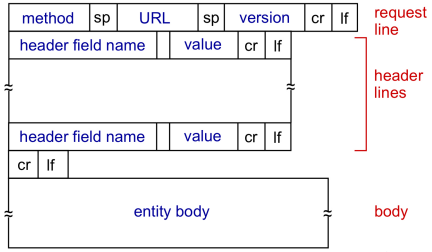


- Server closes TCP connection after sending file
- **Return Trip Time** - (RTT) Time for small packet to travel from client to server and back
- Response Time: 2 RTT + File transmission time

Persistent HTTP

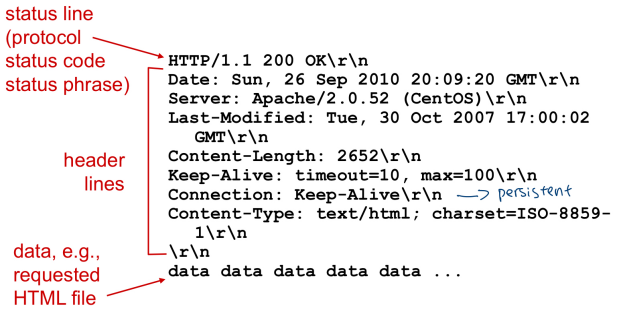
- Multiple objects can be sent over single TCP connection
- Server leaves TCP connection open after sending response
- As little as one RTT for all referenced objects

HTTP Request Message



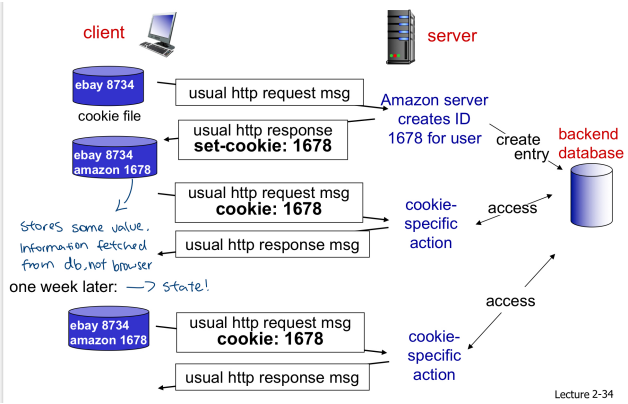
- To upload form input: **POST method** - Input uploaded via entity body and **URL method** - Input uploaded in URL field of GET method
- **HTTP/1.0** - GET, POST, HEAD (Ask server to leave request object out of response)
- **HTTP/1.1** - GET, POST, HEAD, PUT, DELETE

HTTP Response Message



Cookies

- Maintains state on client side
- Components: Cookie header for HTTP response, Cookie header for HTTP, request, Cookie file on user's host (Key-value pair), Database on server

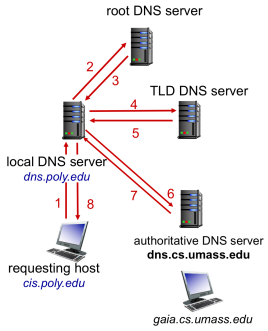


Web Cache (Proxy Server)

- Goal: Fulfill request without involving origin server via caching
- Browser sends all HTTP requests to cache
- Pros: Faster, Reduces traffic to origin server
- Cons: What if origin server updates?
  - **Conditional GET** - Origin server doesn't send object if cache has updated version
  - Cache: Specifies date of cached copy in HTTP request to origin
  - Origin Server: Response contains no object if cached object is updated

Domain Name System

- Maps between IP address and name (e.g. yahoo.com)
- Implemented using distributed and hierarchical databases
- Application-layer protocol
- **Local DNS Name Server** - Local cache of name-to-address mapping. Forwards query into hierarchy.
  - **Time to Live** - (TTL) Cached mappings disappear after some time
- **Root Name Server** - Contacted by local name server that cannot resolve name. Provides IP address of TLD servers.
- **Top-level Domain Server** - (TLD) Provides IP address of authoritative server
- **Authoritative DNS Server** - Organization's own DNS server. Provides mappings for organization's named hosts.
- Iterated query: "Not sure, ask this server"



- Recursive query: "Okay, let me find for you"
  - Heavy load on upper levels of hierarchy