COMP3331 WEEK 3 LECTURE 1

Application Layer: Outline

# 2.3 Electronic mail

## SMTP Protocol {RFC2821}

There are three elements in the modern electronic mail system:

* User agents: user agents, mail reader, mail client on your device.
* Mail servers: mail provider, for example, Hotmail, Gmail.
* Simple mail transfer protocol: SMTP

**Key features:**

1. Reliable transfer of email message from client to server, port 25
2. Direct transfer: sending server to receiving server

* Three phases of transfer:
* Handshaking
* Transfer of message closure

1. Command/response interaction (like HTTP, FTP)
2. Everything is in 7-bit- ASCII

*Note: SMTP is used for delivery and storage, the mail access protocol which used in the retrieval from server is POP/IMAP/HTTPs*

Question: why do we need sender's mail server?

To ensure that the mail can always be delivered even when the receiver server is temporary unavailable.

Question: why do we need receiver's mail server?

Because the user agent is not always online, the user may want to retrieve a file in later time. If a user's machine is not online, we still want to deliver the mail.

# 2.4 DNS (discussed in lecture 2)

# 2.5 P2P Applications (Important in this lecture)

Minimum time to distribute File F to N clients:

**Client-server approach:**

*Time to distribute F to N clients using client server approach*

where N is the number of users, F is the file size, dmin is the minimum client download rate (smallest bandwidth), us is the server upload time. This is the lower bound.

Note: when the number of clients increase, the server needs to push more copies into the network, for very large N, the time to transfer increase linearly in N. (This is not what we want)

**P2P approach:**

*Time to distribute F to N clients using P2P*

Where max upload rate is (all the peers are uploading chunks to other peers), F is the file size, dmin is the minimum client download rate (smallest bandwidth), us is the server upload time.

**Key features:**

* Requesting chunks: rarest first, we want to balance out the chunks in the network, even when someone with the rarest chunk leaves the network, we can still finish downloading the file
* Sending chunks: tit-for-tat. Send chunks to those who sent user chunks at highest rate (try to match). BitTorrent ranks top 4 peers every 10 seconds. Other peers are choked by the user.
* Optimistically unchoke: every 30 seconds, randomly select another peer (to allow new peer to join the network)

## DHT (Distributed Hash Table)

Issue with centralised database: we don't want to have centralised database (becomes bottle neck) but distributed database across multiple location.

**Circular DHT**

* Each peer only aware of immediate successor + predecessor
* Can only query successor, before getting response for tracker.
* Wort case all peers probed, N messages, on average N/2

**Circular DHT with shortcuts**

* Each peer keeps track of IP addresses of predecessor, successor, short cuts
* Possible to design shortcut so O(log N) neighbours, O(log N) msgs in query

**Peer churn**

* Peers may come + go (churn)
* Each peer knows address of its two successors
* Each peer periodically pings its two successors to check aliveness
* If immediate successor abruptly leaves, choose next successor as new immediate successor

Chart, radar chart

Description automatically generated

How to assign keys to peers:

* convert each key to an integer
* assign integer value to each peer
* put key value pair in the peer that is closest to the key

common convention: closest is the immediate successor of the key

e.g., peers: 1,3,4,5,8,10,12,14. If key 13 wants to join, then successor peer is 14. if key 15 wants to join, then successor peer is 1.

# 2.6 video streaming and content distribution networks

**Challenge**: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

Service caching and replication – amortise cost of infrastructure

**Option**: Store and serve multiple copies of videos at multiple geographically distributed sites

* Enter deep: push CDN servers deep into many access networks close to users, used by Akamai
* Bring home: small number of larger clusters in IXPs near access networks used by Limelight

**Example with Netflix:**

* CDN – stores copies of content at CDN nodes
* Subscriber requests content from CDN
* “Over the top” – overlay networks out from existing nodes, disregard the nodes that are supporting this existing network

# 2.7 socket programming (see lab 2 and lab 3)