Notes on Networks

50.012 Networks, Elective 2019

Tey Siew Wen

06 Feb 2020

Contents

1 Lecture 1: Transport Layer

Transport service requirements

- Data loss
- Throughput
- Time sensitivity

1.1 Protocols

1.1.1 TCP/UDP

TCP	UDP
Reliable transport	Unreliable data transfer between sending/receiving process
Flow control: sender won't overwhelm receiver	NIL
Congestion control: throttle sender when network overloaded	NIL
Connection-oriented: setup required between client/server	No need

For both TCP & UDP, there is no encryption of data. Hence we have SSL (Secure Sockets Layer)/ TLS (Transport Layer Security) for providing an encrypted TCP connection, ensuring data integrity and end-point authentication. Apps can use SSL/ TLS APIs to do so.

1.2 Client-server Architecture

Server is always on with permanent IP address. Hosted in data centers for scaling. Clients communicate with the server and may be intermittently (irregularly) connected with the server. Could have dynamic IP addresses. Clients usually do not communicate directly with each other.

1.3 Peer-to-peer Architecture

- Server is not always on
- Arbitrary End Systems directly communicate: Good for file sharing, overlay-routing
- Principles:
  - Fault-tolerant
  - Fate-sharing: It's okay to fail if it's your own mistake?
- Self Scalability
  - Peers request service from other peers and provide service in return
  - New peers bring new service capacity and new service demands
- Challenges:
  - Peers are intermittently connected and change IP addresses
  - Chunk Poisoning

1.4 Electronic Mail

3 major components:

1. User Agents
2. Mail Servers
    i. Mailbox: contain incoming messages for user
    ii. Message Queue: Messages to be sent (outgoing)
3. Simple Mail Transfer Protocol: SMTP
    - uses TCP for sending emails from client to server via port 25
    - has 3 phases of transfer: handshake, transfer, closure.
    - Requires message to be in 7-bit ASCII
    - Uses CRLF.CRLF for determining end of message

Compared to HTTP, SMTP is a push rather than a pull server. Both have ASCII command/res interaction, status codes.

User Agent → SMTP → Sender Mail Server → SMTP → Receiver's Mail Server → Mail Access Protocol → User Agent

Mail server: forward mail from sender to receiver mail server. If both the client and the receiver mail server are offline, the sender's mail server will keep retrying to send the mail until it works. Back then the mail servers are not so reliable to on all the time.

1.4.1 Mail Access Protocol

Mail Access Protocol is used for retrieval from server

- POP (Post Office Protocol)
- IMAP (Internet Mail Access Protocol)
- HTTPs

1.5 Processes

Definition of a Process: A program running within the host. It must have an identifier that includes IP address, port numbers associated with the process on host. e.g. HTTP: 80, Mail: 25

Types of Process Communication

- Inter-process communication: Dependent on OS
    - unless the process is on the application layer, then it is controlled by the app developer.
- Host-to-host communication: Exchange Messages
    - Messages are sent/received via sockets.

Types of Processes

- Client Process: Initiate Communication
- Server Process: Waits to be contacted

1.6 Message Segmentation

- Reducing end-to-end delay.
- More efficient recovery from bit error. Otherwise the whole message needs to be retransmitted.
- Huge message may block other smaller packets
- Header overhead linear to the no. of packets
- Cause new problems e.g. out-of-order arrival of packets

2 Lecture 2: HTTP

Each HTTP message designed to be *self-contained*:

- it bring as much detail as the server needs to serve that request
- server does not maintain state

Protocols that maintain state are complex

- Past History must be maintained
- If server/client crash, the state that is stored on either host will be inconsistent
- however doing so is likely to improve performance

2.1 HTTP Methods

Safe Methods e.g. GET, HEAD:

- enable caching and loading distribution
- does not modify resources on server

Idempotent Methods e.g. Multiple DELETE:

- *Definition*: An effort that can be applied multiple times without changing the result beyond the initial application.
  - Handle lost confirmations by re-sending
  - May modify resources on the server
  - Can be executed multiple times without changing outcome
- Counter e.g. Multiple POST

2.2 Proxy

Definition: An entity authorized to act on behalf of another e.g. an intermediatory server performing requests for us

- Serve as a single point access of control to enforce security protocols

Common traits of proxies:

- Single access of control
- Load Balancing: Distribute incoming requests to a cluster of servers, all provide the same kind of service

2.2.1 Forward Proxy

When a client makes a connection attempt to that file transfer server on the Internet, its requests usually have to *pass through the forward proxy first*, where a firewall will be behind it.

1. Depending on the forward proxy's settings, a request can be allowed or denied.
2. If allowed, then the request is forwarded to the firewall and then to the file transfer server.
3. From the point of view of the file transfer server, it is the proxy server that issued the request, not the client. So when the server responds, it addresses its response to the proxy.

4. When the forward proxy receives the response, it recognizes it as a response to the request that went through earlier. And so it in turn sends that response to the client that made the request.

Applications:

- Content Logging & Eavesdropping
- Accessing Services Anonymously

2.2.2 Reverse Proxies

The reverse proxy does the exact opposite of what a forward proxy does. It accepts requests from external clients on behalf of servers stationed behind it. The firewall is between the client and reverse proxy instead of being in between the forward proxy and the servers.

1. Depending on the reverse proxy's settings, a request can be allowed or denied.
2. From the perspective of the client, it is the reverse proxy that is providing file transfer services.

Applications:

- A/B testing, Multivariate testing
- Distribute load

3 Lecture 3: Web API

Application programming interface (API) specifies how 2 software components should interact.

Types of API Protocols

3.1 Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is the specific protocol for XML-based data exchange, popular in enterprise M-M communication.

- However, specific protocols seem to add overhead in many cases.

Web service definition language (WSDL) is used to specify the available services to clients.

- Client-side functions to call API can be automatically generated
- Auto-completion for API calls

3.2 Representational State Transfer (REST)

Representational State Transfer (REST) is not a protocol, but rather an architectural style.

- Pros: Simple, scalable, general, high performance
- Cons: No-built-in ACID & Under-fetching/Over-fetching
  - ACID: Atomicity, Consistency, Isolation, Durability
  - Over-fetching: You might get more data than you need, but the end point is designed to give you that specific data.
  - Underfetching: You might get less data than you need, because there is no end point designed to give you the data you need from that server.

3.2.1 Resources in REST

- Types
  i. Collections
  ii. Instances
- Referenced in the HTTP header

Simple Static Settings	Dynamic Settings
Each resource corresponds to single file.	Server will interpret the URL as parameters, dynamically create content for provided parameters. Content at resource URL may not exist yet.

3.2.2 Running HTTP Requests with curl

At any point of time, if you want to understand more about the flags that you pass into curl to test your http requests, run `curl -help`. Examples of sending a get request and giving the `-v` flag to show more information on exchanged messages. For a patch request, you can run: `curl -H "Content-Type: application/json" -X PATCH -d '"title":"test"' http://jsonplaceholder.typicode.com/todos/199`

The result will be:

```
{
    "userId": 10,
    "id": 199,
    "title": "test",
    "completed": true
}
```

3.2.2.1 PUT vs POST

- PUT: Used to update an existing resource. Reply will be 200.
- POST: Used to create an element in a collection. Reply will be 201 with URL of created element.

3.3 Multipurpose Internet Mail Extensions (MIME)

Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of email messages to support text in character sets other than ASCII, as well attachments of audio, video, images, and application programs. Message bodies may consist of multiple parts, and header information may be specified in non-ASCII character sets.

4 Lecture 5: Network Applications

4.1 Streaming stored video

- Use Redundancy within and between images to decrease # bits required to encode image
  - Spatial (within image)
  - Temporal (from one image to next)
- Encoding Rate
  - CBR (Constant Bit Rate): Fixed encoding rate
  - VBR (Variable Bit Rate): Changes as amount of spatial, temporal coding changes
- Challenges
  - Continuous playout constraint: Once client playout begins, playback must match original timing
    - Network Delay e.g. queue delay are variable → need client side buffer to match playout requirements
  - Client Interactivity: Allow pause, fast-forward, rewind and jump through video
  - Video packets may be lost, need to retransmit
    - Packets may be received slower than it is being sent, so some packets might be skipped.

4.1.1 Streaming multimedia: DASH

DASH stands for Dynamic, Adaptive Streaming over HTTP (DASH).

Other adaptive solutions: Apple's HTTP Live Streaming (HLS) solution, Adobe Systems HTTP Dynamic Streaming, Microsoft Smooth Streaming

- Server
  - encodes video file into multiple versions
  - each version stored, encoded at a different rate
  - manifest file: provide URLs for different versions
- Client
  - Handles most of the streaming logic:
    - When to request chunk
    - What encoding rate to request
    - Where to request

4.2 Voice over IP (VoIP)

VoIP end-end-delay requirement: <150 ms good, >400ms bad

- For delay jitter, we aim to minimize client playout delay
- Receiver attempts to playout each chunk exactly q msecs after chunk is generated
  - Large q: less packet loss
  - Small q: better interactive experience

4.2.1 Adaptive Playout Delay

4.2.1.1 Adaptively Estimate Packet Delay

Exponentially Weighted Moving Average (EWMA)

$$d_i = (1-\alpha)d_{i-1} + \alpha(r_i - t_i)$$

where:

- d_i is the delay estimate after i-th packet
- α is a small constat
- r_i is the time received, t_i is the time sent
- so $r_i - t_i$ is the measured delay of i-th packet

Average deviation of delay

$$v_i = (1-B)v_{i-1} + B|r_i - t_i - d_i|$$

d_i, v_i are calculated for every received packet, but used only at the start of talk spurt.

- 1st packet in talkspurt: playout – time$_i = t_i + d_i + Kv_i$
  - longer playback delay
- Remaining packets are played out periodically

5 Lecture 6: CDN and PNP

5.1 Content Distribution Networks (CDN)

Problem Background: Streaming Content to hundred of thousands of simultaneous users

Possible solutions:

1. Single mega server: doesn't scale
    - point of network congestion
    - long path to distant clients
    - single point of failure
    - multiple copies of vid sent over outgoing link
2. Store/serve multiple copies of videos at multiple geographically distributed sites (CDN)

CDN Operators stores copies of content at CDN Nodes.

5.1.1 CDN Server

5.1.1.1 Types of CDN

- Commercial CDN: e.g. Akamai, Cloudflare
- Content provider's own CDN: e.g. Google, netflix
- Telco CDN

5.1.1.2 Choice of Placement

- Push CDN servers deep into many access ISPs so that they are close to users
- Smaller no. of larger clusters in IXPs near access ISPs

5.1.1.3 Server Selection

Points of consideration:

- Geographically close
- Performance: Real-time measurement
- Load-balancing
- Cost: CDN may need to pay its provider ISP
- Fault-tolerance

5.1.1.4 Routing the Content

After selection, we still have a routing problem. We can route content access in 3 different ways:

1. DNS-based
2. Application Driven
    - Multiple connection setup, name lookups
3. Routing (anycast)-based

5.1.1.5 Content Replication

Mechanisms:

- Push: Use of off-peak bandwidth optimization
- Pull: More Adapative

e.g. Netflix

- has prepared content
- so push content to CDN during off-peak hours whose CDN servers pull & cache content by user demand.

e.g. Youtube

- people can upload content anytime
- so CDN servers pull content by user demand at all time

5.2 Peer-to-peer architecture

- Arbitrary end systems directly communicate as peers
- Peers are intermittently connected and may change IP addresses

e.g. VoIP (Skype), Multimedia Streaming (Kankan.com), File Distribution (BitTorrent)

5.2.1 File distribution time (lower bound)

- d_{min}: min client download rate
- $\frac{F}{d_{min}}$: min client download time
- $\frac{F}{u_s}$: server upload time for one copy of file

5.2.1.1 Client-server

- Server must sequentially upload N file copies (to N clients)
- Each client must download file copy

Time to distribute F to N clients:

$$D_{c-s} \geq \max\left(N\frac{F}{u_s}, \frac{F}{d_{min}}\right)$$

5.2.1.2 P2P

- Server must upload at least one copy
- Each client must download one file copy
- Clients as aggregate must download NF bits

Time to distribute F to N clients

$$D_{p2p} \geq \max\left(\frac{F}{u_s}, \frac{F}{d_{min}}, N\frac{F}{u_s + \sum u_i}\right)$$

Another simplification of the third term of the equation:

$$N\frac{F}{u_s + \sum u_i} = \frac{F}{u_s/N + u}$$

6 Lecture 7: Reliable Data Transport

6.1 Principles of Reliable Data Transport

1. Physical channels are never completely reliable
    - Wireless links subjected to interference
    - Transmission noise → lead to bit errors
    - Routers & Switches may drop packets due to buffer overflow
2. Reliable communication relies on **detection and retransmissions as necessary**.
    - Receiver: Acknowledges packets from the sender.
    - Sender: Transmits & retransmits based on information provided by the receiver. If the sender times out (ack/nack not received within a certain time-frame). An action will be triggered.
    - Packets: Contain sequence numbers

6.1.1 Model for Reliable Communication

- Provides send/receive methods for applications to transfer packets
- Fields in packet header is used to coordinate with peer

6.2 Basic RDT Protocols

- RDT 1.0: Assume reliable channel
- RDT 2.0: Considers that channel may be corrupted.
- RDT 2.1: RDT 2.0 + 1-bit `seq_num` for identifying packet loss.
- RDT 2.2: RDT 2.1 + Stop and wait.
- RDT 3.0: RDT 2.2 + timeouts

6.2.1 RDT 2.0 & 2.1

Assumptions

1. Channel may corrupt but never lose packets.
    - Receiver always get something whenever a packet is sent
    - Sender always receives an ack after sending a packet (Feedback)
2. Receiver can always detect if packet has been corrupted.
    - Checksum to detect bit errors
        - May be flawed as the checksum could still tally after modification to data

Communication Flow

- Sender: Transmits one packet at a time and waits for ack/nack
- Receiver: Sends ack when it receives packet correctly, and sends nack when it receives erroneous packet.

Challenges

- Corruption: The reality is that packets, acks, nacks can all get corrupted.
  - This means that the sender may not be able to tell which packet was received correctly or not.
  - If sender retransmits the packet, the receiver could deliver the same packet to the application twice. If it transmits a new one, the receiver can fail to deliver a packet to the application.
- Lost Packets
  - Receiver does not know that a packet was sent to it, so it won't send an ack response
  - Sender left waiting for ack, but ack never comes :(

Hence the 1-bit `seq_num` is introduced in RDT2.2, such that nack is not necessary anymore.

6.2.2 RDT 2.2

Stop and wait (Alternating-bit protocol)

- Sender sends one packet and waits for receiver response

6.2.2.1 Importance of Sequence Numbers (`seq_num`)

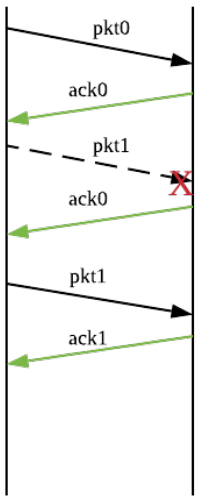
- `seq_num` in ack identifies received packet(s)
- Size of `seq_num`: Determines num of packets that can be sent before acknowledgements must be received

Sender	Receiver
Adds `seq_num` to packet	Must check if received packet's `seq_num` != previously stored `seq_num`
Must check if received ACK matches correct `seq_num`	Receiver cannot know if its last ACK received OK at sender

6.2.2.2 Scenarios

Case: Corrupted packet

- Receiver not receiving pkt1 as intended

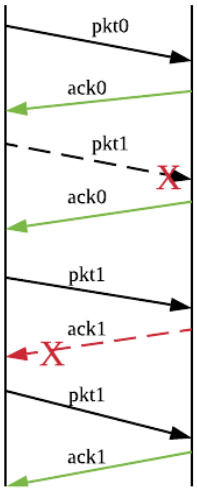


Case: Corrupted ack

- Sender not receiving ack0, so it retransmit pkt0

Case: Corrupted packet + Corrupted ack

- 1st packet and its ack not corrupted
- 2nd packet is corrupted so it is sent a 2nd time.
- During the 2nd time, the packet is not corrupted but the ack is corrupted



Corrupted ack
ack0 missing

6.2.3 RDT 3.0

Previously RDT 2.2 did not address the issue of lost packets. There are a few ways of recovering from lost packets:

1. Keep sending packet repeatedly until sender gets an ack
    - Sender and receiver both will do extra work
2. Sender waits for ack for a specified time, then re-sends the packet if ack fails to arrive in time (timeout)
    - Works well if maximum ack delay is known and **does not change**. Otherwise will lead to premature timeout.

6.2.3.1 Features of RDT 3.0

RDT3.0 is the correct way of implemeting RDT but inefficient.

- Involves timeout before retransmission
- Sends only 1 packet at a time
- works well if delay between sender & receiver is small
- inefficient if $RTT \gg t_{pkt}$ where $t_{pkt} = \frac{L}{C}$

Example of ineffiency:
Consider a 1 Gb/s link with 10 ms delay in each direction, where RDT 3.0 sends only 1 packet every 20ms. The link is actually capable of sending 20 million bits in 20ms, so for typical packet sizes, only tiny fraction of link's capacity is used.

6.2.3.2 Calculating the Performance of RDT 3.0 Important variables

- $t_{pkt} = \frac{L}{C}$
- t_{out}: RTT
- C: link speed
- L: average packet size
- q: packet loss/corruption probability

If given p, the time to travel from sender to receiver and p', the time to travel from receiver to sender,

$$q = p + (1-p)p'$$

Expected time between successful transmissions (T_{succ})

$$\begin{aligned} T_{succ} &= \sum_{k=0}^{\infty}(k+1)(RTT + t_{pkt})(q^k)(1-q)(RTT + t_{pkt})\frac{1}{(1-q)^2} \\ &= \frac{RTT + t_{pkt}}{1-q} \end{aligned}$$

Throughput

$$\frac{L}{T_{succ}} = C(1-q)\left(1 + \frac{RTT}{t_{pkt}}\right)$$

Thoroughput improves if

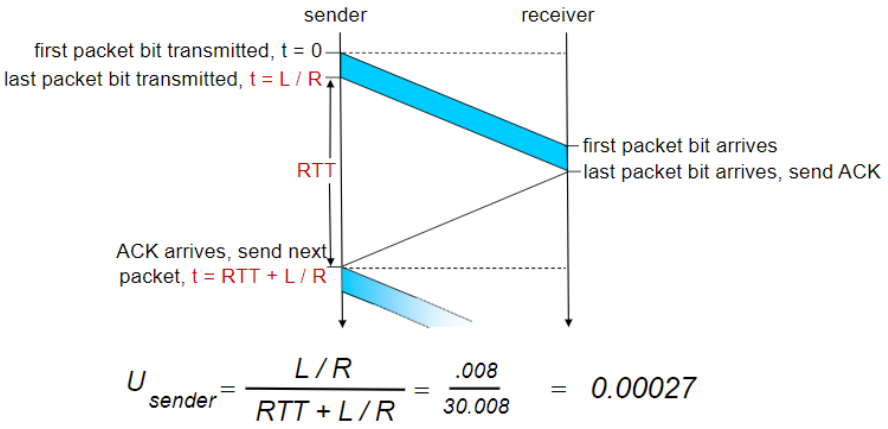
- corruption probabiliy/loss get smaller
- RTT gets smaller compared to t_{pkt}

Utilization

Fraction of time sender is busy sending

$$U = \frac{D}{RTT + D}$$

Space-time diagram



$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

6.2.3.3 Requirements of Timeout

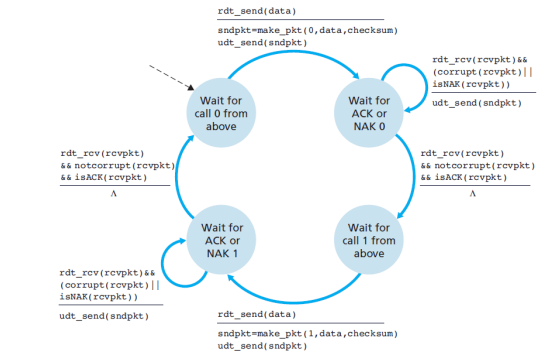
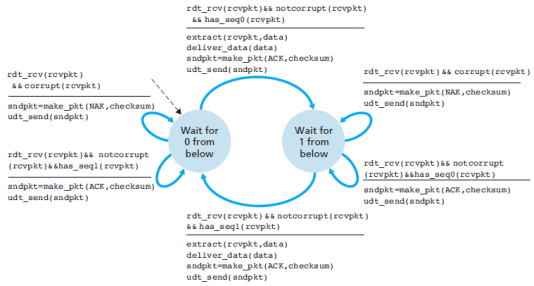
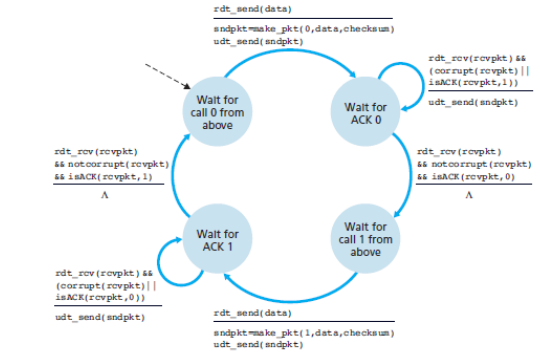
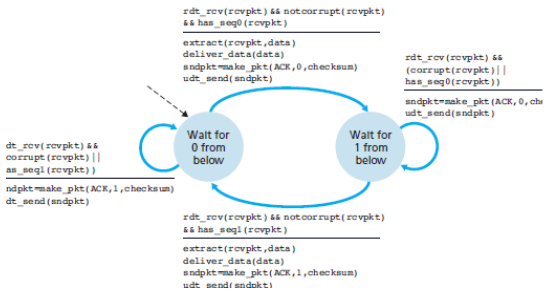
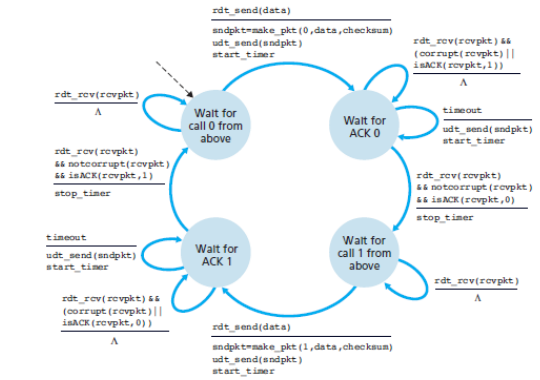
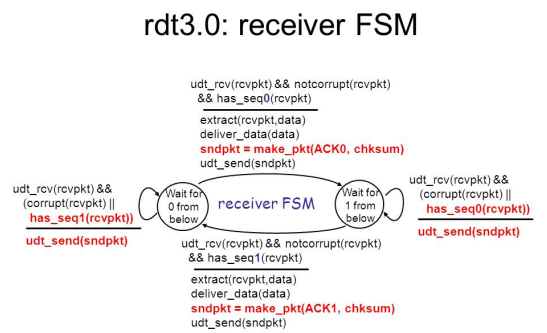
$$\text{Timeout} \geq RTT \text{ where } RTT = d_{nodal(data)} + d_{nodal(ack)}$$

Recall that $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$.

Challenges

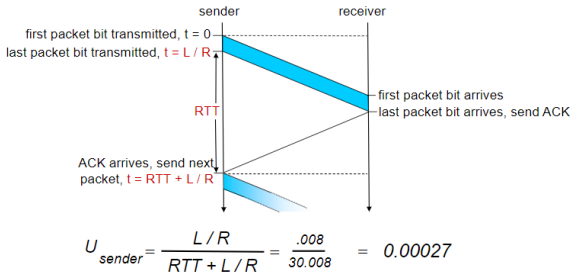
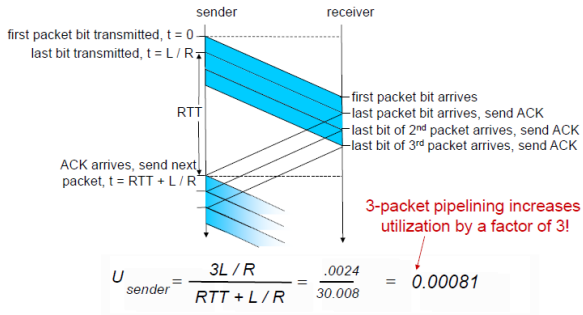
- Premature timeout: sender receives ack late, after it has retransmitted the packet that it did not receive ack for.
- Estimating RTT: Components are unknown and variable

6.2.4 Finite State Machine Diagrams

Protocol	Sender	Receiver
RDT 2.1	 Figure 3.11 ♦ rdt2.1 sender	 Figure 3.12 ♦ rdt2.1 receiver
RDT 2.2	 Figure 3.13 ♦ rdt2.2 sender	
RDT 3.0	 Figure 3.15 ♦ rdt3.0 sender	

7 Lecture 8: RDT Pipelines

Pipelining allows for increased utilization of the link.

Without pipeline	With pipeline
	

7.1 Consequences of pipelining

- The range of `seq_num` must be increased, since each in-transit packet must have a unique number and there may be multiple, in-transit, unack packets. (not counting retransmissions)
- Minimally, the sender will have to buffer packets that have been transmitted but not ack yet.

There are 2 basic approaches towards pipelined error recovery:

1. Go-Back-N (GBN): Interactive Animation
2. Selective Repeat (SR): Interactive Animation

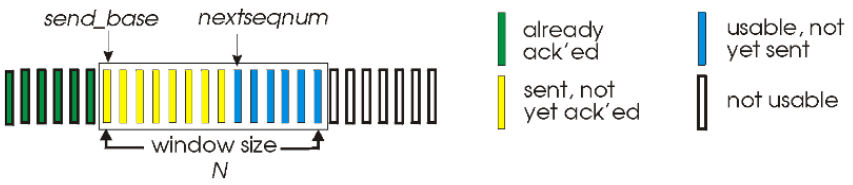
7.2 Types of Pipeline Protocols

7.2.1 Go-Back-N (GBN)

Sender

The sender is allowed to transmit multiple packets without waiting for an ack, where:

- `N`: the no. unack packets in the pipeline / the window size.
- `base`: the `seq_num` of the oldest unack packet.
- `nextseqnum`: smallest unused `seq_num`, the index of the next packet to be sent.
- `seq_num_space`: range of $[0, 2^k - 1]$
- `seq_num` with k number of bits for the packet sequence number field in the packet header.



This results in the following array of packets at the sender's side: The result will be:

```
data = seq_num_space
ack_pkts = data[0:base]
window = data[base:base+N]
sent_unack_pkts = data[base:nextseqnum]
avail_pkts_to_send = data[nextseqnum:base+N]
outside_window = data[base+N::]
```

As such, it is referred to the **sliding-window protocol**.

Receiver

The receiver always send ack for pkt with highest in-order `seq_num`. E.g. If the receiver receives packets 1,2,4,5 (pkt 3 is lost)

- Assuming window size is 5. (packets 1-5)
- It will keep resending the ack for pkt 2 and discard packets above 3 (4-5).
- Base is updated to 3, so after the timeout, the sender will resend packets 3-7.
- If these packets are successfully received, then the sender can update base to 8 and send 8-12th packets.

7.2.2 Selective Repeat (SR)

7.2.2.1 Comparison with GBN

Similarities	Differences
Fixed window size	-
Initialize timeout for packet at sender's side	Timeout initalized for each packet in SR, instead of one timeout for one packet in GBN
Sender allowed to transmit multiple packets without waiting for an ack	For out of order packets due to lost packets, they are buffered instead in SR instead of being discarded.

7.2.2.2 Limitations of small range of `seq_num`

- The receiver may intepret duplicate data as new data.
- The `seq_num_size` should be $2n$ to avoid this problem.

8 Lecture 9: TCP and RDT Principles

8.1 Transmission Control Protocol (TCP)

8.1.1 TCP Segment Structure

- Cumulative ACK: TCP sends an ACK with `seq_num` of next byte expected from the other side, instead of replying which packet it has received
    - Similar to Go-back-N
- Out-of-order packets: TCP will buffer, and not discard them.

8.1.2 Round-trip time (RTT) and Timeout

Estimating RTT

- $SampleRTT$: an average of recent measurements of time from segment transmission until ACK receipt
    - Ignore retransmissions

Exponential Moving Average Equation:

$$EstimatedRTT = (1-\alpha) \cdot EstimatedRTT + \alpha \cdot SampleRTT$$

- Influence of past samples decreases exponentially fast
- Typically $\alpha = 0.125$

Estimating SampleRTT

$$DevRTT = (1-\beta)DevRTT + \beta|SampleRTT - EstimatedRTT|$$

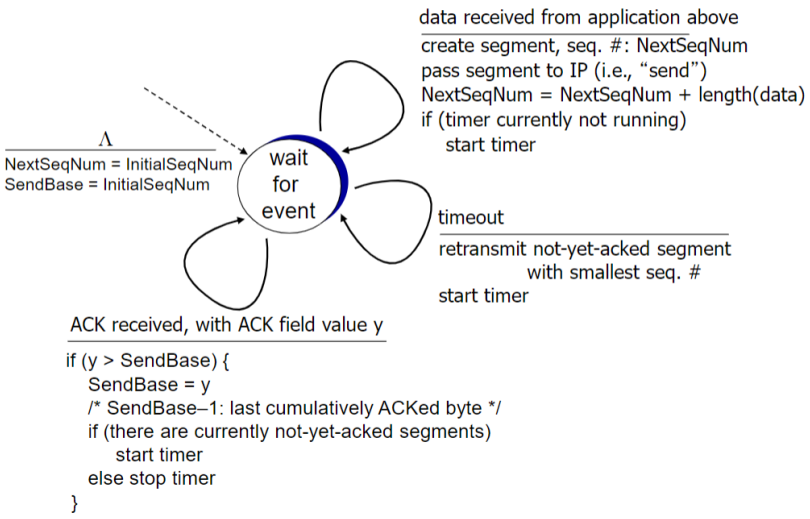
- Typically $\beta = 0.25$

Timeout Interval

$$TimeoutInterval = EstimatedRTT + 4 \times DevRTT$$

- $4 \times DevRTT$ is a safety margin
- Too short will result in premature timeout and unnecessary retransmissions
- Too long will result in slow reactions to segment loss

8.1.3 TCP RDT



8.1.3.1 Fast retransmit

- If sender receives 3 duplicate (extra) ACKs for same data, it will resend unACKed segment with smallest `seq_num`.

8.2 Other Reliable Data Transfer (RDT) Protocols

8.2.1 Go-Back-N (GBN)

- Requirement for k-bit `seq_num` in packet header: $2^k > N$
- Receiver window size: 1 (`expected_seq_num`).
- Relationship among `expected_seq_num`, `send_base`, `next_seq_num`:
  - `send_base` ≤ `next_seq_num` ≤ `expected_seq_num`

8.2.2 Selective Repeat (SR)

`rcv_base` = `next_seq_num` if all packets up to `next_seq_num` are received

- Requirement for k-bit `seq_num` in packet header : $2^k > 2N$
- Receiver window size: 1 (`expected_seq_num`).
- Relationship among `expected_seq_num`, `send_base`, `next_seq_num`:
  - `send_base` < `next_seq_num` < `expected_seq_num`

9 Lecture 10: Congestion Control

9.1 Principles of Flow Control

- Receiver controls sender so the sender won't overflow receiver's buffer by transmitting too much/too fast
  - Application may remove data from TCP socket buffers
- Receiver includes a `rwnd` (receiver window) value in TCP header of receiver-to-sender segments
  - `RcvBuffer`

9.2 Principles of Congestion Control

- Congestion Control ≠ Flow Control!
- Mainfestations
  - Buffer Overflow at Routers: Lost Packets
  - Queueing in Router buffers: Long Delay

When packets are lost, any upstream transmission capacity used for that packet is wasted.

9.2.1 Scenario 1: One Router w/ Infinite Buffers

- Assuming no retransmission

9.2.2 One Router w/ Finite Buffers

Assumptions for idealized case

- Sender knows when router buffers available
- Sender sends only when router buffers available

Transfer rates

- $\lambda_{\text{in}} = \lambda_{\text{out}}$: Application-layer input = output
- $\lambda'_{\text{in}} \geq \lambda_{\text{in}}$: Transport-layer input includes retransmissions

9.2.3 TCP Congestion Controls

Increase sender's transmission rate until loss occurs

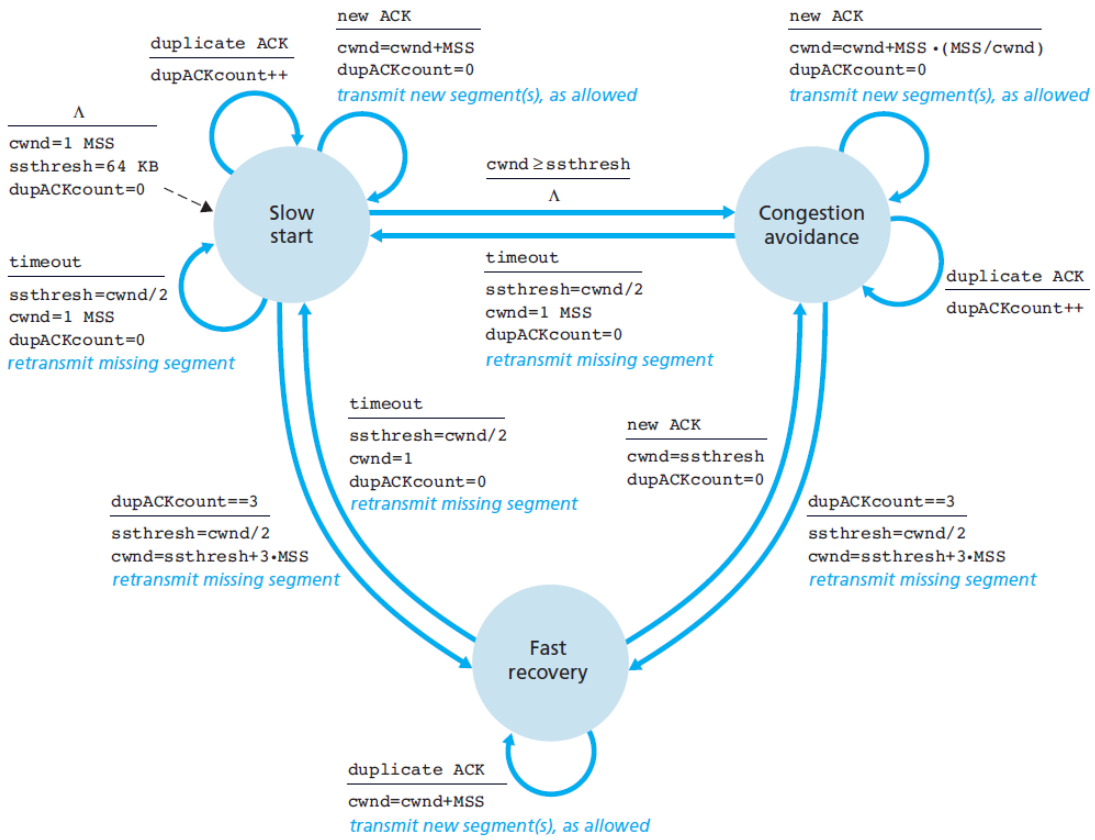
- Additive Increase: Increase cwnd (congestion window) by 1 MSS every RTT until loss detected
- Multiplicate Increase: Reduce cwnd in half after loss

10 Lecture 11: TCP Wrapup

10.1 TCP Congestion Control

A summary code for the sections discussed below:

```
cwnd = MSS
while connected:
if time < slow_start_duration:
    cwnd *= 2
    if receive_ACK:
    cwnd += MSS
else: # congestion-avoidance state
    if receive_ACK:
    cwnd += MSS*(MSS/cwnd)
```



10.1.1 Start Connection: Slow Start

When connection begins, increase rate exponentially until first loss event. (Initial rate is slow but ramps up very fast)

1. Initial `cwnd`: 1 MSS (maximum segment size)
2. Double `cwnd` every RTT
    - It is doubled with the formula: $cwnd = cwnd + MSS * \frac{cwnd}{mss}$
3. Increment `cwnd` for every ACK received

10.1.2 Congestion-avoidance state

Window grows exponentially in **slow start** to threshold, then grows linearly during the congestion-avoidance (CA) state.

- The inexponential increase is switched to linear when `cwnd` gets to half of its value before timeout.
- On loss event, `ssthresh`=0.5∗`cwnd` before loss event.

In the CA state, the congestion window is increased by $\frac{1}{k}$.

- where $k = \frac{cwnd}{mss}$

10.1.3 Explicit Congestion Notification (ECN)

10.1.3.1 Network-assisted congestion control

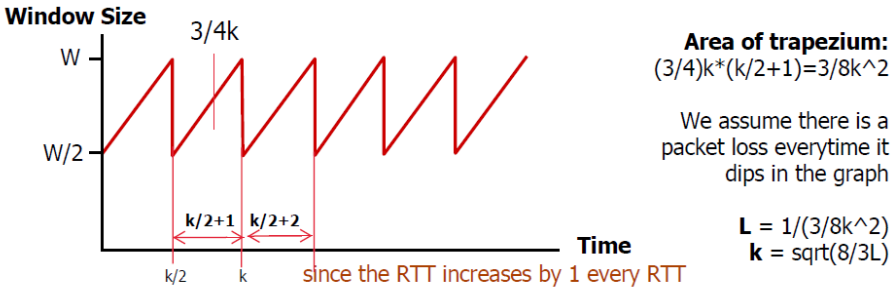
- 2 bits in IP header (ToS field) marked by network router to indicate congestion
- Receiver sets ECE bit on ACK to notify sender of congestion

10.1.4 Calculating TCP Throughput

Ignoring slow start and assuming there is always data to send,

$$\text{TCP Throughput} = \frac{3}{4} * \frac{W}{RTT}$$

- where W: window size in bytes where loss occurs



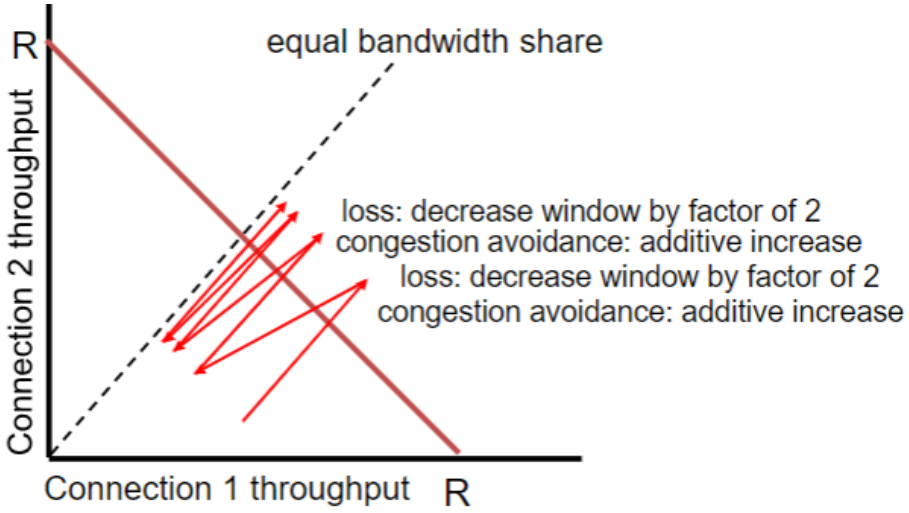
10.1.4.1 Calculating Segment Loss Probability, L

e.g. given 1500 byte segments, 100ms RTT, 10Gbps throughput, requires average 83,333 in-flight segments

$$10^7 = \frac{1.22 \times 1500}{100 \times 10^{-3} \times \sqrt{L}}$$
$$L = 2 \times 10^{-10}$$

10.1.5 TCP Fairness

- Goal: For n TCP sessions sharing same bottleneck link of bandwidth R, each should have $\frac{R}{K}$ rate.
- Implementation via Additive Increase and Multiplicate Decrease



11 Lecture 13: Network Layer Overview

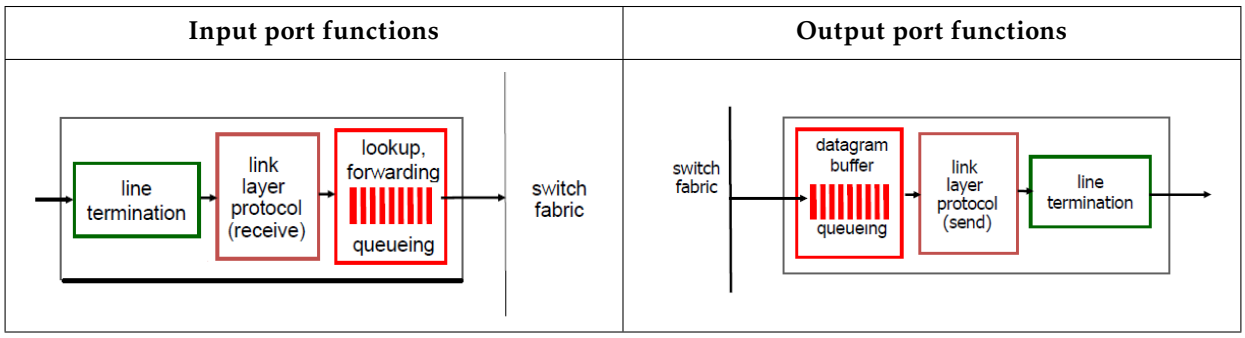
11.1 Two Key Network-Layer Functions

1. *Forwarding*: Router-local action of transferring a packet from an input link interface to the appropriate output link interface. (Data Plane)
2. *Routing*: Network-wide process that determines the end-to-end paths that packets take from source to destination. (Control Plane)

Control Plane Approaches:

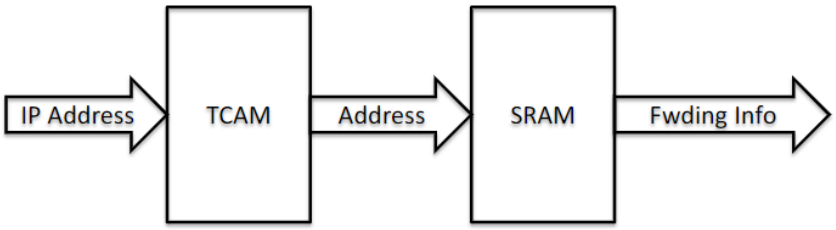
1. Traditional Routing Algorithms: Implemented in Routers
   - Per-router control plane: Individual routing algorithm components in each and every router interact in the control plane
2. Software-defined Networking (SDN): Implemented in remote servers
   - A distinct controller interacts with local control agents

Overview



11.2 Input Processing

Typical Forwarding Process:



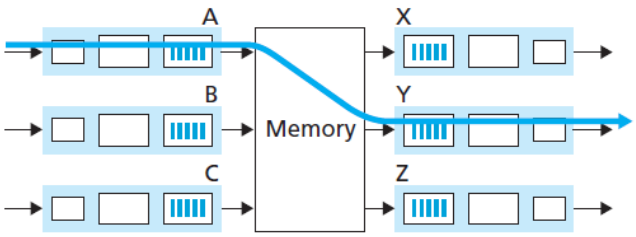
1. Every router has a **forwarding table**, which is computed and updated by its routing processor, and **stored at the input port**.
   - A router forwards a packet by examining the value of a field in the arriving packets header, and then using this header value to index into the routers forwarding table.
   - The value stored in the forwarding table entry for that header indicates the routers outgoing link interface to which that packet is to be forwarded.
   - The value to be examined in the packet's header is determined by the routing algorithm.

2. The routing processor also stores a **shadow copy**, typically stored at each input port.
    - With a shadow copy, forwarding decisions can be made locally, at each input port, without invoking the centralized routing processor on a per-packet basis
    - This helps to avoid a centralized processing bottleneck.
3. After determining the outgoing link interface, the packet is forwarded from the the router's input ports to its output ports via **switching fabrics**.

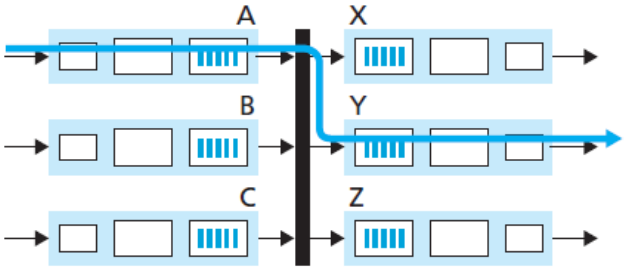
11.2.1 Switching Fabrics

There are 3 types.

1. Memory
    - An input port with an arriving packet first signaled the routing processor via an interrupt. The packet was then copied from the input port into processor memory.
    - The routing processor then extract the destination addr from the header, looked up the appropriate output port in the forwarding table, and copied the packet to the output ports buffers.
    - If the memory bandwidth is such that B packets per second can be written into, or read from, memory, then the **overall forwarding throughput** (the total rate at which packets are transferred from input ports to output ports) **must be less than** $\frac{B}{2}$.
    - Two packets cannot be forwarded at the same time, even if they have different destination ports, since only one memory read/write over the shared system bus can be done at a time.

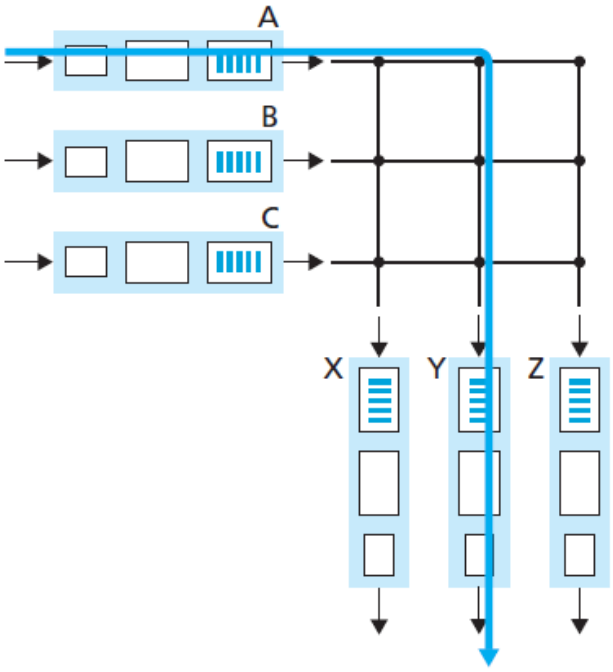


2. Bus
    - The packet is transferred directly from the input port to the output port over a shared bus, without intervention by the routing processor
    - The input port pre-pend a switch-internal label (header) to the packet to indicate the local output port for the packet
    - The packet is received by all output ports, but only the port that matches the label will keep the packet. The label is then removed at the output port.
    - **Bus Contention**: Only one packet can board the bus at a time. Hence, the switching speed of the router is limited to the bus speed and transmitting the packet onto the bus



3. Crossbar

   - An interconnection network consisting of $2N$ buses that connect N input ports to N output ports
   - There is a switch fabric controller to open/close cross points of the horizontal and vertical bus
   - Capable of forwarding multiple packets in parallel.
   - Initially developed to connect processors in multiprocessor.

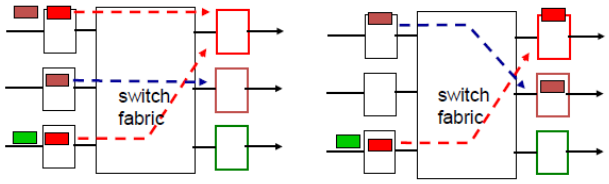


11.2.2 Input Port Queueing

Since the switching fabric operates slower than the input ports, queueing at input queues may cause the input buffer to overflow. Packet loss will then occur when no memory is available to store arriving packets.

Head of the Line (HOL) Blocking

Queued datagram at front of queue prevents others in queue from moving forward



11.2.3 Output Processing

Output port processing takes packets that have been stored in the output ports memory and transmits them over the output link. This includes selecting and de-queueing packets for transmission, and performing the needed linklayer and physical-layer transmission functions.

If the datagrams arrive from fabric faster than the transmission rate, buffering will be required. **Packet loss may occur** due to congestion and lack of buffers.

11.3 Routing Algorithms

- *Destination-based forwarding*
  - Destination address range is the index for a particular link interface
  - However there are times ranges don't divide up nicely
- *Longest Prefix Matching*
  - Longest address prefix is the index for matching destination addr
  - Performed using Ternary Content Addressable Memories (TCAMs)

e.g of longest prefix matching. given

Destination Address Range	Link Interface
1100 1000 0001 0111 0010*** **** ****	0
1100 1000 0001 0111 0011*** **** ****	1
1100 1000 0001 0111 0011000 **** ****	2
otherwise	3

a) **1100 1000 00010111 0010**110 1011 0110: 0

b) **1100 1000 00010111 0011000** 1011 0110: 2

c) **1100 1000 00010111 0011**001 1011 0110: 1

11.3.1 Ternary Content-addressable memory (TCAMs)

- A specialized type of high-speed memory that searches its entire contents in a single clock cycle.
  - More power, more area and higher latency than SRAM
  - To retrieve data on RAM, the operating system (OS) must provide the memory address where the data is stored.
  - Data stored on CAM can be accessed by performing a query for the content itself, and the memory retrieves the addresses where that data can be found.
  - Performs parallel processing
- Able to store and query data using three different inputs: 0, 1 and X (Ternary).

12 Lecture 14: IPv4 Addressing

Hosts/Routers typically has interfaces such as wired Ethernet, wireless 802.11. Interfaces are connections between the host/router and the physical link.

Since every host and router is capable of sending and receiving IP datagrams, IP requires each host and router interface to have its own IP address. Thus, an IP address is technically associated with an interface, rather than with the host or router containing that interface.

- Each IP is 32 bits long (4 bytes)
  - Each byte of the address is written in its decimal form, separated by a period from other bytes in the address. Example: 193.32.216.9
  - Total of 2^{32} possible IP addresses.
- Each IP address can be assigned to a **subnet**. Example 223.1.1.0 /24
  - The leftmost 24 bits of the 32-bit quantity define the subnet address
  - So any host that has an IP address of 223.1.1.x can be identified to be under the same subnet. Any additional hosts will also be required to have the address of this format.

To determine the subnets, detach each interface from its host/router, creating islands of isolated networks. An example is shown below.

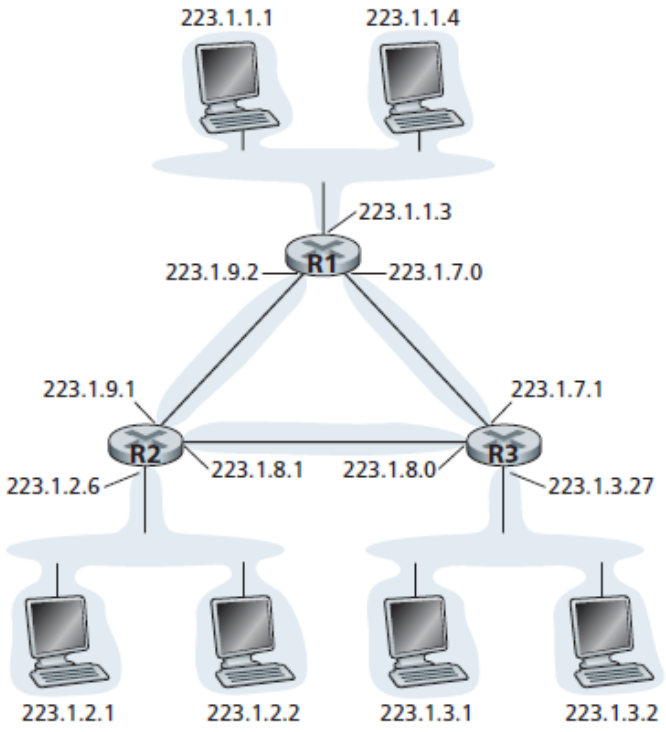


Figure 4.17 ♦ Three routers interconnecting six subnets

> Core routers in Singapore are running on IPv4. Singtel currently has Singtel has OLT vendors which operate on IPv6, and most of their services are IPv6 ready. They have a tunneling protocol for translating IPv6 internally within the Singtel network to IPv4 to these core routers.

12.1 Classless InterDomain Routing (CIDR)

- Address format: `a.b.c.d/x`
    - x most significant bits in the address (the prefix) constitute the network portion of the IP address.
        - Organizations usually use a range of addresses with a common prefix.
        - Hence, when a router outside the organization forwards a datagram whose destination address is inside this organization, they only care about the prefix.
        - This **reduces the size of the forwarding table** in the routers, since a single entry of the form `a.b.c.d/x` will be sufficient to forward packets to any destination within organization.
        - Sometimes organizations may come together under a subnet for **route aggregation/ route summarization**.
    - The remaining $(32 - x)$ bits are used to distinguish among the devices within the organization.

> Currently there is around 800k prefixes! 60k networks around the world. Singapore comprise of about 1k networks.

12.2 Dynamic Host Configuration Protocol (DHCP)

DHCP allows host to dynamically obtain its IP address from a network server when it join its network.

How it works - DORA

1. Host broadcast "DHCP **D**iscover" msg [optional]
2. DHCP server hears the broadcast and responds with "DHCP **O**ffer" msg [optional]
3. Host requests IP address: "DHCP **R**equest" msg
4. DHCP server sends address: "DHCP **A**ck" msg

Example

1. New connecting host sends a DHCP *request* encapsulated in UDP, IP and Ethernet.
2. The Ethernet frame *broadcast* on LAN, received at router running DHCP server
3. Ethernet demuxed to IP demuxed, UDP demuxed to DHCP
4. DHCP server formulates an ACK containing
    - client's IP address
    - First-hop router for client (if requested)
    - Name & IP Address of DNS Server (if requested)
    - MASK to determine which portion is the network/host (if requested)
5. Encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client
6. Client obtains the info inside ACK.

12.3 Network Address Translation (NAT)

- All datagrams leaving local network must have the same single source NAT IP address, but with different src port numbers
  - Replace (`src IP addr, port#`) of every outgoing datagram to (`NAT IP addr, new port#`)
  - Record every (`src IP addr, port#`) to (`NAT IP addr, new port#`) translation pair in NAT translation table.
- Allows local network to use just 1 IP for all devices in the network
  - The local network can change address of devices in local network without notifying outside world.
  - Security: Devices inside local net not explicitly addressable & visible by outside world.
- Allow local network to change ISP without changing addresses of devices in local network

13 Lecture 15: Routing Algorithms

13.1 Link-state Broadcast Algorithm (LS Algorithm)

13.1.1 Properties of the Algorithm

- Centralized: Use global information. Requires each node to first obtain a complete map of the network before running the Dijkstra algorithm.

13.1.2 How it works

Watch this for explanation of how it the algorithm works: https://www.youtube.com/watch?v=ue-BDS-7Ikw.

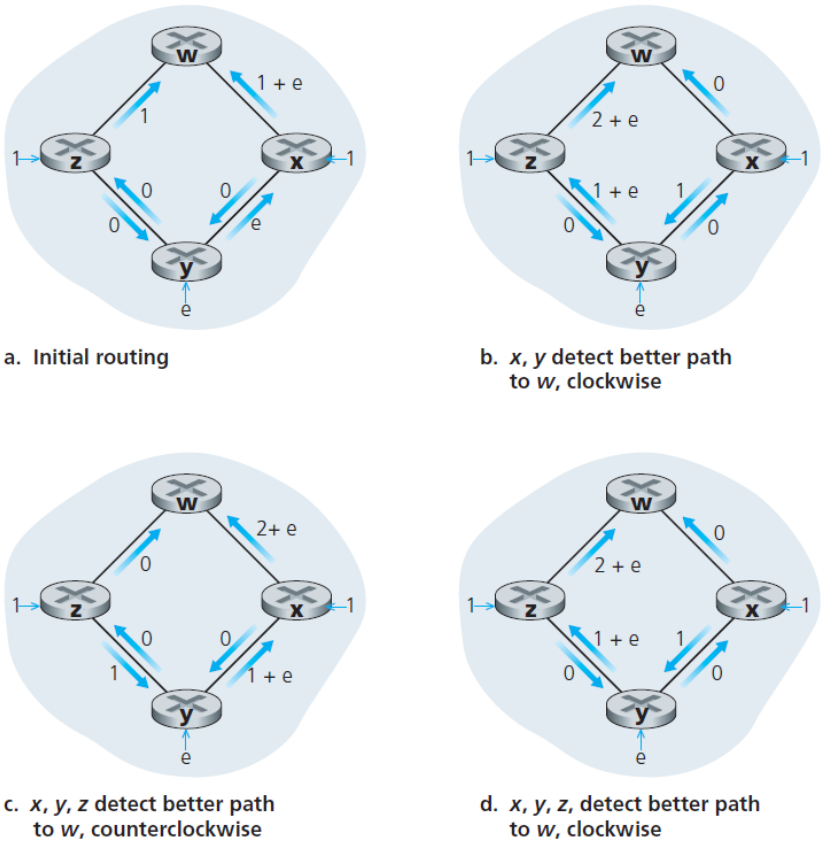
13.1.3 Time Complexity

- Total number of nodes we need to search through over all the iterations is $\frac{n(n+1)}{2}$
  - First iteration n, 2nd iteration $n-1$ and so on.
  - $n + n - 1 + ... + 1 = \frac{n(n+1)}{2}$
  - Hence, **worst case time complexity** = $O(n^2)$
  - If we implement the data structure as a heap, can reduce to logarithmic complexity.

13.1.4 Possible Pitfall Scenario - Unequal link costs

$$c(u,v) \neq c(v,y)$$

For most algorithms, it is assumed that the link costs depend on the traffic carried. And here, the load carried on both directions are not equal.



a. Initial routing

b. *x*, *y* detect better path to *w*, clockwise

c. *x*, *y*, *z* detect better path to *w*, counterclockwise

d. *x*, *y*, *z*, detect better path to *w*, clockwise

Figure 4.29 ♦ Oscillations with congestion-sensitive routing

- Initialization
  - Both x and z originates a unit of traffic destined for w,
  - y injects an amount of traffic equal to e, also destined for w.
- 1st run
  - y
    - determines that the clockwise path to w has a cost of 1, while the counterclockwise path to w (which it had been using) has a cost of $1 + e$.
    - So the least-cost path for y is now clockwise.
  - x
    - determines that the least-cost path is also clockwise.
- 2nd run
  - x, y, and z all detect a zero-cost path to w in the counterclockwise direction, and all route their traffic to the counterclockwise routes.
- 3rd run
  - x, y, and z all detect a zero-cost path to w in the clockwise direction, and all route their traffic to the clockwise routes.

Solutions

1. Mandate that link costs do not depend on the amount of traffic carried
2. Not all routers run the LS algorithm at the same time
   - Even though they initially execute the algorithm with the same period but at different instants of time, the algorithm execution instance can eventually become, and remain, synchronized at the routers. To avoid this, randomize the time it sends out a link advertisement.

13.2 Distance Vector (DV)

13.2.1 Properties of the algorithm

- Iterative
  - this process continues on until no more information is exchanged between neighbors
  - requires no signal to ask it to stop
- Asynchronous
  - does not require all of the nodes to operate in lockstep with each other
- Distributed
  - each node receives some information from one or more of its directly attached neighbors, performs a calculation, and then distributes the results of its calculation back to its neighbors.

13.2.2 Bellman-Ford equation for Least Cost

$$d_{x,y} = \min_v [c(x,v) + d_v(y)]$$

13.2.3 How it works

Watch this: https://www.youtube.com/watch?v=x9WIQbaVPzY

- A node x updates its distance-vector estimate when it either sees a cost change in one of its directly attached links or receives a distancevector update from some neighbor.
- To update its own forwarding table for a given destination y
    - what node x really needs to know is not the shortest-path distance to y but instead the neighboring node $v*(y)$ that is the next-hop router along the shortest path to y.

13.2.4 Pitfall - Count to infinity problem when a link cost increases

Good news travel fast (Lower cost), Bad news travel slow (Higher cost)

Solution

- Poisoned Reverse: if z routes through y to get to destination x, then z will advertise to y that its distance to x is infinity. z will continue telling this little white lie to y as long as it routes to x via y.
    - However, does not work for loops involving three or more nodes (rather than simply two immediately neighboring nodes)

14 Lecture 16: Scalable Routing

Routers are usually aggregated into domains known as *autonomous systems* (AS).

Intra-AS Routing	Inter-AS Routing
Gateway has links to other routers in other AS	Gateway performs inter-domain routing and intra-domain routing
All routers in AS must run same intra-domain protocol	Routers in different AS can run different intra-domain routing, but all routers would need to run the same inter-domain protocol
Single admin, so no policy decisions needed	Admin wants to control over how its traffic is routed, who routes through its net
Scalablity less of an issue, can always use hierarchical routing to reduce scale	Scalability is critical
Focus on performance	Policy may dominate over performance

14.1 Inter-AS: Border Gateway Protocol (BGP)

- It is a **path-vector** protocol
  - Link cost is not the priority; **the routing policy is the most important** for BGP Routing.
  - `CIDRized Prefix + AS-path + Next-hop`
    - Next-hop: IP addr of gateway router to enter the path (You cannot route AS, only route routers.)
    - AS-path: Enforce import-policy for paths, avoid looping
- Provides each AS a means to
  - **eBGP**: Obtain subnet reachability information from neighbor AS
  - **iBGP**: Propagate reachability information to all AS-internal routers
- BGP messages are exchanged between peers over semi-permanent TCP connection to advertise paths to different destination network prefixes
  - OPEN: *open TCP connection* to remote BGPpeer and authenticate sending BGP peer
  - UPDATE: Advertises new path / withdraw old path
  - KEEPALIVE: Keeps connection alive *in absence of UPDATES*. ACKs OPEN request
  - NOTIFICATION: Report errors in previous msg / Close Connection.

14.1.1 BGP Route Selection

- BGP *sequentially* invokes these rules to select a route
  1. Local Preference value attribute: Policy Decision
  2. Shortest AS-Path
  3. Closest Next-Hop router: Hot Potato Routing
  4. Additional Criteria
- **Hot Potato Routing**: Choose local gateway that has the **least intra-domain cost**, without caring about inter-domain cost.
- *Habit of Keeping quiet*: If a network X does not want to route from B to C via X, X will not advertise to B a route to C.

14.2 Intra-AS: Interior Gateway Protocols (IGP)

Routing with Interior Gateway Protocols (IGP).

- RIP: Routing Information Protocol
- OSPF: **Open Shortest Path First**
  - Area Border: Summarize distances to nets in own area then advertise to other Area Border routers
  - Backbone: run OSPF routing limited tobackbone
  - Boundary: Connect to other AS
- IGRP: Interior Gateway Routing Protocol

15 Lecture 17: Software Defined Networking (SDN)

- Separation of Control plane and Data plane
- **Centralized Control Plane:** A distinct remote controller interact swith local control agents (CAs) in routers to compute forwarding tables and distribute
  - Easier network management: Avoid Router misconfigurations, greater flexibility of traffic flows
  - Table-based forwarding allows "programmable" routers
- Data Plane Switches that implement generalized data-plane forwarding in its hardware can focus on making routing fast
- Incorporated for *Service Function Chaining*
  - Enterprise Virtual Services:
    - Subscriber, application or destination based granular traffic steering
  - Consumer Virtual Home Gateway
    - PNF: Firewall
    - VNFs: Parent Control, TCP/HTTP optimization, IPv4/IPv6 NAT

15.1 OpenFlow Protocol

- TCP is used to exchange messages, with default port `6653`.
- There are 3 classes of messages
  - Controller-to-switch
  - Asynchronous (Switch to Controller)
  - Symmetric (misc)

16 Lecture 19: Link Layer

16.1 Introduction

- **Main Responsiblity of the Link Layer**
  - Transfer datagram from one node to *physically adjacent* node over a link.
  - Over different links, the datagram is transferred by different link protocols
    - e.g. First Link 802.11, Second Link PPP, ... Last link Ethernet
  - Each link protocol provide different service.
    - e.g. may or may not provide RDT over link
- **Whether the Link Layer is implemented**
  - Implemented via the combination of hardware, software and firmware in *every host*
  - Particularly in the *network interface card (NIC)* such as the Ethernet card/802.11 card or a Ethernet chipset. It has a physical interface (physical layer) and a controller (link layer)
  - Attaches to the host's system buses to connect to the CPU of the host (application, transport, network and link layers), so that it is easier manage the link layer.

16.1.1 Two types of links

1. Point-to-point (PPP)
   - Dial-up Access
   - Link between Ethernet switch and host
2. Broadcast
   - Ethernet
   - upstream HFC
   - 802.11 wireless LAN

16.2 Link Layer Services

- **Framing, link access**
  - Encapsulate datagram into frame, add header and trailer
  - Channel access if *shared medium* [optional]
  - "MAC" Address used in frame headers to identify `src, dest`.
- **Reliable delivery** between adjacent nodes
  - Usually used on wireless links that would have *high error rates*. Seldomly used on low bit-error links incude Fiber, Twisted pair, Coaxial Cables.
- **Duplexing**
  - Half-duplex: Nodes at both ends of link can transmit *but not at the same time*.
    - There may be collision if the node receive ≥ 2 signals at the same time, due to simultaneous transmissions (interference).
  - Duplex: Nodes at both ends of link can transmit at the same time. There will be 1 wire for 1 direction.
    - Utilize Multiple Access Protocol.

16.3 Addressing: IP vs MAC

IP address	MAC address
32-bit	48-bit
Organizations reserve certain IP address ranges	MAC address allocation administered by IEEE, manufacturer buys a portion of MAC address space, so first 24 bits configured by manufacturer
Dynamically assigned by the network	MAC address burned in the NIC ROM. But some are also software configurable.
Uses Decimal Notation e.g. 224.12.13.2/24	Uses Hexadecimal Notation e.g. IA-2F-BB-76-09-AD
Network layer forwarding	To get frame from one interface to another physically connected interface
Not portable, as address depends on IP subnet to which the node is attached.	Portable, as LAN card containing fixed MAC flat address can be moved from one LAN to another.

16.4 Address Resolution Protocol (ARP)

IP address: Each IP node (host/router) on LAN has a ARP Table that contains IP/MAC address mappings for some LAN nodes, and the TTL time after which address mapping will be forgotten. These ARP tables are created by nodes without intervention from net adminstrator.

So if we know an interface's IP address, we can determine an interface's MAC Address.

16.4.1 Within the same Local Access Network (LAN)

1. A wants to send datagram to B, but A's ARP table does not have B's MAC Address
2. A broadcast ARP query packet, containing B's IP Address. All nodes on same LAN will receive this query.
3. B receives this packet and replies to A with its MAC Address as a frame.
4. A Caches the IP-to-MAC Address pair in its ARP table until timeout.

16.4.2 Across LANs

1. All the nodes in the same LAN as A will not know the MAC address of B, who is inside another LAN. But there is a router R between the 2 LANs
2. A create IP datagram with `IP src A, dest B`, and put it inside a link-layer frame with `dest R`. This frame is thus sent from A to R.
3. R receives this datagram, take it out of the frame and pass it to IP.
4. R can now create a link-layer frame with `dest B`, containing the A-to-B IP datagram.

16.5 Multiple Access Protocol (MAP)

- MAP is a distributed algorithm that determines how nodes share channel (when the node can transmit).
- Communication about channel sharing uses the channel itself.
- No out-of-band channel for coordinated.

16.6 Ethernet

The sending adaptor encapsulates IP datagram in an **Ethernet Frame structure** as such:

Preamble	Destination Address	Source Address	Type	Payload	CRC

- Preamble: 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- Address: 6 byte source, destination MAC Address
- Type: Indicates higher layer protocol (Mostly IP but others possible, e.g. Novell IPX, AppleTalk)
- CRC: Cyclic Redundancy Check at receiver

16.7 Switches

Switches and Routers are rather similar:

- Both are store-and-forward
- Both have forwarding tables

Switches	Routers
Link-layer devices	Network-layer devices
Learn forwarding table using flooding, learning, MAC addresses	Compute table using routing algorithms, IP addresses

16.7.1 Self-learning

- When the switch receives a frame, it will record sender/location pair in switch table and learn of the location of the sender.

16.8 Virtual Local Area Network (VLANs)

- Made up of switch(es) supporting VLAN capabilities to define multiple virtual LANs over single physical LAN infrastructure
- **Port-based VLANs**
  i. *Single physical switch can operate as multiple virtual switches:* Switch ports are grouped by switch management software
  ii. *Traffic Isolation:* Frames to/from ports 1-8 can only reach ports 1-8
  iii. *Dynamic Membership:* Ports can be dynamically assigned among VLANs
  iv. *Trunk port:* Carries frames between VLANs defined over multiple physical switches

802. IQ VLAN frame format

Preamble	Destination	Source	NEW	Type	Data (Payload)	CRC

NEW:

- 2-byte Tag Protocol Identifier
- Tag Control Information
  - 3-bit priority field like IP TOS
  - 1 bit drop eligible indicator
  - 12 bit VLAN ID field

16.9 Example of all layers together!

A typical scenario:

1. Laptop tries to connect to the Internet and requires the following:
    i. its IP address
    ii. First-hop router address
    iii. DNS server address
2. Make a DHCP request encapsulated in IP & 802.3 Ethernet
3. Broadcast Ethernet frame on LAN, and the router running the DHCP router receives it.
    i. This frame is demuxed to IP, UDP demuxed to DHCP
    ii. DHCP Server formulate DHCP ACK containing what the client wants
    iii. Encapsulate frame at DHCP server and forward (Switch Learning) through LAN
4. Client demultiplexes the DHCP frame and receive the ACK reply

17 Lecture 21: Wireless Networks

Examples:

	Single hop	Multiple hop
Infrastructure	*WiFi, cellular* Host connect to base station which connects to larger internet	*Mesh Net* Host have to relay through several wireless nodes to connect to larger internet
No infrastructure (**No base station + no connection to larger internet**)	*Bluetooth*	*MANET, VANET*

17.1 802.11 Wireless LAN

- Basic Service Set (BSS) in infrastructure mode consists of wireless hosts and access point
- Host communicates with a base station, known as an access point (AP).
- 802.11b spectrum divided into 11 channels at different frequency
  - Admin choose frequency for AP
- Channel can be same as that chosen by neighboring AP, which may result in **interference**
  - Under **Carrier Sense Multiple Access** (CSMA)
    - *Collision Detection*: Collisions are detected within short time to reduce channel wastage
      - Easy in wired LANs: can measure signal strength, and compare transmitted, received signals
      - Difficult in wireless LANs: Received signal strength overwhelmed by local transmission strength
    - *Collision Avoidance:*
      - Sender Channel sensed idle → transmit entire frame
      - Sender Channel sensed busy → defer transmission by starting random *backoff time* and transmit when timer expires. *This interval increases exponentially if no ACK.*
- Host associates with an AP by scanning channels, listening for **beacon frames** containing AP's name (SSID) and MAC address and selecting one to associate with
  - There are 2 types of scanning: Passive, Active
    - Passive: APs will send beacon frames
    - Active: Host broadcast Probe Reuqest Frame + AP will send Probe Response Frames
  - Then, the host will send Association Request Frame to the selected AP and the selected AP will send Association Response Frame back.
  - Typically run DHCP to get IP address in the selected AP's subnet

17.1.1 Mobility within same subnet

- When hosts are **moving between diff APs in the same subnet**, then the self-learning switch will see frame from the host and remember which switch port can be used to reach the hosts.
  - If diff subnets, then the host will probably lose connection.
  - Otherwise, OUT OF SYLLABUS

17.1.2 Rate Adaptation

Rate adaptation allows transmission to be done at different rates within the wireless network, depending upon the network conditions.

- Adaptor can detect channel condition in real-time
- Plotted in a Signal Noise Ratio (SNR) vs Bit Error Rate (BER) graph
- As the mobile moves away from base station
    - BER ↑, SNR ↓
    - When BER becomes too high, switch to a lower transmission rate but with lower BER.

17.1.3 Power Management

Beacon Frame: Contains list of mobiles with AP-to-mobile frames waiting to be sent

1. Node stays awake if AP-to-mobile frames are to be sent.
2. Otherwise node tells AP "I am going to zzz until next **beacon frame**"
3. AP say "ok then i don't transmit frames to you"
4. Node autowakes up before next beacon frame.

17.2 Mobile Networks

17.2.1 Cellular Network Architecture

Components of the architecture

2G:

- BTS: Base Transceiver Station
- BSC: Base Station Controller
- MSC: Mobile Switching Center

3G:

- SGSN: Serving GPRS Support Node
- GGSN: Gateway GPRS Support Node

4G:

- UE: User Element
- MME: Mobility Management Entity
- HSS: Home Subscriber Server
- S-GW: Serving Gateway
- P-GW: Packet data network Gateway
- EPC: Evolved Packet Core

Network	What changed	Components
3G	New Cellular Data Network operates in parallel with existing cellular voice network. Voice network is unchanged in core.	SGSN, GGSN
4G	No more separation between voice and data. All traffic is carried over IP core to gateway.	UE, MME, HSS, S-GW, P-GW, EPC

17.2.2 Handling Mobility

17.2.2.1 Handoff with common MSC

1. Old BSS: "Oi MSC, i want to HOTO, this is the list of new BSSs"
2. MSC: "Ok i set up new path and allocate resources for these new BSS"
3. New BSS tries to prepare to take over and allocates radio channel for use by mobile
4. After finishing, the New BSS signals MSC and old BSS "I am ready"
5. Old BSS tells new BSS: "ok now i really HOTO to you"
6. New BSS signal to activate new channel
7. Mobile signals via new BSS to MSC: "Handoff complete"

✓ Allows stronger signal to/from new BSS: Continuing connectivity and less battery drain

✓ Load Balancing: Free up channel in current BSS

17.2.2.2 Handoff Between MSCs

Distinguishing networks beyond a single MSC

1. Home Network
   - Network of cellular provider that a mobile is subscribed to
   - Network to which the mobile user's permanent phone number belongs
   - Data Stored in **Home Location Register** (HLR): The database containing permanent cell phone #, profile info, info of current location (even outside the home network)
2. Visited Network
   - Network in which mobile currently resides
   - Data is stored in **Visitor Location Register** (VLR): Database with entry for each user currently in network.
   - A visited network can also be home network

Multi-MSC Chain
Made up by the Anchor MSC (1st MSC visited during call) + other MSCs that are added onto the end of the chain as the mobile moves to new MSC.

- Optional path minimization step to shorten multi-MSC Chain

17.2.2.3 Consequences of mobility

- Performance
  - There may be packet loss/delay due to handoff / link-layer retransmissions
  - TCP may interpret this loss as congestion, and *decrease cwnd unnecessarily*
  - *Limited bandwidth* of wireless links; wireless behind wired networks by 1-2 orders in terms of speed
- Aside from Performance, there should be minimal impact
  - Best Effort Service Model remained unchanged.
  - TCP and UDP can run over wireless mobile