Contents

Contributors:
- Daniel Fitz (Sanchez)

1 Sem Outline

Week (dates)	Lecture
1	Computer Networks and the Internet
2	Principles of Nw Apps: HTTP, SMTP, DNS
3	Application Layer: P2P, CDN, Sockets
4	Networking at UQ
5	Transport Layer: UDP
6	Transport Layer: TCP
7	Network Layer: Data Plane
8	Network Layer: Control Place
9	Link Layer
11	Wireless and Mobile
12	Security
13	Multimedia

Table 1: Week Outline

2 Exam Notes

The exam will consist of:

- A number of analytical questions, similar to the tutorial questions. You won't be asked any complex analytic problems which are completely different to those in tutorials
- A number of short answer questions of the type: compare XXX to YYY and explain the differences, or advantages/disadvantages of these protocols/algorithms/applications/techniques
- Questions about different protocols, their functions and where they fit in the network protocol stack. You won't be asked about protocols you have not seen in lectures
- Questions about packet exchanges in some common protocols (e.g. DHCP, DNS, ARP, TCP, HTTP)

No multiple choice questions this year ☹

2.1 Chapter 1

- What is the Internet
- Network Edge
- Network Core
- Delay, Loss Throughput
- Protocol Layers and their service models

Not Examinable: Networks under attack, history of networking

2.2 Chapter 2: Application Layer

- Principles of Networked Applications
- Web and HTTP (including options covered in lectures/labs)
- Electronic Mail
- DNS (but no detailed message/packet format)
- Peer-to-peer
- Internet Video

Not Examinable: Detailed message formats for DNS and for email, case studies, socket programming

2.3 Chapter 3: Transport Layer

All Material

2.4 Chapter 4: Network Layer – Data Plane

All Material

2.5 Chapter 5: Network Layer – Control Plane

Most of the material covered, except as below, including a general overview of what SNMP does. You should understand link-state and distance vector routing. You won't be asked any numerical questions with distance-vector. For routing protocols, you should know about BGP, OSPF, IS-IS, RIP (which isn't in lectures, but is an example of an intra-AS distance-vector algorithm). All you really need to know about these algorithms are whether they are inter-AS or intra-AS, link-state or distance-vector.
Not Examinable: Details of SNMP architecture and packet formats. Details of BGP (5.4.2, 5.4.3, 5.4.5 are not examinable)

2.6 Chapter 6: Link Layer

- General Principles
- Error Detection and Correction – services provided, differences between correction and detection
- Multiple Access Links and Protocols, but NOT DOCSIS
- Switched Local Area Networks
- "Day in the Life of a Web Page Request" – details of each stage are covered in the earlier sections

Not Examinable: Exactly how to calculate parity, checksum, CRC, DOCSIS, MPLS, Data Center Networking

2.7 Chapter 7: Wireless

- General Principles
- Wireless characteristics
- WiFi (IEEE 802.11) except as below

Not Examinable: Mobility in WiFi, advanced features in WiFi (Ch 7.3.5). Personal area Networks. Cellular Internet Access. Mobility Management, Mobile IP, Mobility effects on higher layers

2.8 Chapter 8: Security

- What is network security – confidentiality, integrity, authentication
- Cryptographic principles – symmetric and public key algorithms (you won't be asked to calculate any ciphers)
- Names, types and uses of common cyphers, at least: Diffie-Hellman, RSA, DES, 3DES, AES, MD5, SHA-1

- Message integrity and signatures
- SSL and TLS
- IP Sec and VPN
- Firewalls and Intrusion Detection Systems – general principles

Not Examinable: Details of cipher algorithms, key lengths. Securing Email. Wireless security

2.9 Chapter 9: Multimedia

- Properties of multimedia
- UDP and HTTP streaming
- Voice over IP
- Protocols – RTP, SIP

Not Examinable: Case Studies (e.g. Skype). Network Support for multimedia, such as token-bucket, diffserv, QoS

2.10 Packet Formats

Must understand and decode the packet contents if given a byte stream for:

Link Layer: Ethernet (but not VLAN packets)

Network Layer: IPv4 (not IPv6), you won't be asked to decode option fields, but they may be present. These IPv4 packets may contain protocols like DNS or ICMP, but you won't be asked to decode the contents of those packets

Transport Layer: TCP, UDP.. You won't be asked to decode option fields, by they may be present

Application Layer: Simple HTTP request and reply. If you are required to decode text messages you will be given a table of ASCII codes

3 Chapter 1

- billions of connected computing devices
- transmission rate: **bandwidth**
- **Packet Switches:** Forward packets
  - **routers** and **switches**
- **Internet: "network of networks"** (Interconnected ISPs)
- **Protocols** control sending, receiving (e.g. TCP, IP, HTTP, Skype, 802.11)
- **Internet standards**
  RFC: Request for comments
  IETF: Internet Engineering Task Force

3.1 Network Structure

- **Network Edge**
  - hosts: clients and servers
  - servers often in data centers
- **Access networks, physical media:** wired, wireless communication links
- **network core:**
  - interconnected routers
  - network of networks

3.2 Access Network

3.2.1 Digital Subscriber Line (DSL)

- use **existing** telephone line to central office DSLAM
  - data over DSL phone line goes to Internet
  - voice over DSL phone line goes to telephone net
- < 2.5 Mbps upstream transmission rate (typically < 1 Mbps)
- < 24 Mbps downstream transmission rate (typically < 10 Mbps)

3.2.2 Cable Network

> **frequency division multiplexing:** different channels transmitted in different frequency bands

- **HFC: hybrid fiber coax**
  - asymmetric: up to 30Mbps downstream transmission rate, 2 Mbps upstream transmission rate
- **network** of cable, fiber attaches homes to ISP router
  - homes **share access network** to cable head-end
  - unlike DSL, which has dedicated access to central office

wireless LANS:

- within building (30 meters)
- 802.11b/g/n (WiFi): 11,54,450 Mbps transmission rate

wide-area wireless access:

- provided by telco (cellular) operator, 10's km
- between 1 and 10 Mbps
- 3G, 4G, LTE

3.3 Sending

- takes application message
- breaks into smaller chunks, known as **packets**, of length L bits
- transmites packet into access network at **transmission rate** R
  - link transmission rate, aka link **capacity, aka link bandwidth**

Note 1: Packet Transmission Delay

$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

3.4 Physical Media

- **bit:** propagates between transmitter/receiver pairs
- **physical link:** what lies between transmitter and receiver
- **guided media:** signals propagate in solid media (copper, fiber, coax)
- **unguided media:** signals propagate freely, e.g. radio
- **twisted pair (TP):** two insulated copper wires
  - Category 5: 100 Mbps, 1 Gbps Ethernet
  - Category 6: 10 Gbps

3.4.1 Coax

- two concentric copper conductors
- bidirectional
- broadband: multiple channels on cable, HFC

3.4.2 Fiber Optic Cable

- glass fiber carrying light pulses, each pulse a bit
- high-speed operation: high-speed point-to-point transmission (e.g. 10's - 100's Gbps transmission rate)
- low error rate
  - repeaters spaced far apart
  - immune to electromagnetic noise

3.4.3 Radio

- signal carried in electromagnetic spectrum
- no physical "wire"
- bidirectional
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

Radio Link Types:

- **terrestrial microwave:** up to 45 Mbps channels
- **LAN** (e.g. WiFi) 54 Mbps
- **wide-area** (e.g. cellular) 4G cellular: ˜10 Mbps
- **satellite**
  - Kbps to 45 Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low altitude

3.5 Packet-switching

3.5.1 Store-and-forward

> L bits per packet
> Source to destination: R bps

- takes $\frac{L}{R}$ seconds to transmit (push out) L-bit packet into link at R bps
- **store and forward:** entire packet must arrive at router before it can be transmitted on next link

Note 2: End-End delay

$$\text{delay} = 2\frac{L}{R}$$

(assuming zero propagation delay)

3.5.2 Packet switching versus circuit switching

Is packet switching a "slam dunk winner?"

- great for bursty data (resource sharing, simpler, no call setup)
- excessive congestion possible: packet delay and loss (protocols needed for reliable data transfer, congestion control)

3.6 Packet Loss

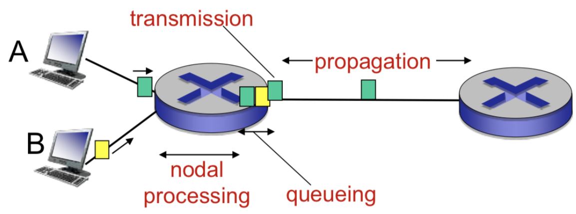

Figure 1: Packet Delay Algorithm Explanation

Note 3: Packet Delay Algorithm

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

3.6.1 Nodal Processing

$$d_{\text{proc}}$$

- check bit errors
- determine output link
- typically < msec

3.6.2 Queuing Delay

$$d_{\text{queue}}$$

- time waiting at output link for transmission
- depends on congestion level of router

3.6.3 Transmission Delay

$$d_{\text{trans}}$$

- L: packet length (bits)
- R: link bandwidth(bps)
- $d_{\text{trans}} = \frac{L}{R}$

3.6.4 Propagation Delay

$$d_{\text{prop}}$$

- d: length of physical link
- s: propagation speed ($\approx 2 \times 10^8$ m/sec)
- $d_{\text{prop}} = \frac{d}{s}$

3.7 Throughput

Rate (bits/time unit) at which bits transferred between sender/receiver

Instantaneous: rate at given point in time
Average: rate over longer period of time

Note 4: Bottleneck Link

Link on end-end path that constrains end-end throughput

3.8 Layering

3.8.1 Why Layering?

Dealing with complex systems:

- Explicit structure allows identification, relationship of complex system's pieces (layered **reference model** for discussion)
- Modularization eases maintenance, updating system
  - change of implementation of layer's service transparent to rest of system
  - e.g. change in gate procedure doesn't affect rest of system
- layering considered harmful?

3.8.2 Internet Protocol Stack

Application: supporting network applications (FTP, SMTP, HTTP)
Transport: process-process data transfer (TCP, UDP)
Network: routing of datagrams from source to destination (IP, routing protocols)
Link: data transfer between neighboring network elements (Ethernet, 802.111 (WiFi), PPP)
Physical: bits "on the wire"

3.8.3 ISO/OSI Reference Model

Internet stack "missing" these layers. These services, if needed, must be implemented in application.

Application:
Presentation: allow applications to interpret meaning of data, e.g. encryption, compression, machine-specific conventions
Session: synchronization, check-pointing, recovery of data exchange
Transport:
Network:
Link:
Physical:

3.9 Security

- Malware can get in host from:
  Virus: self-replicating infection by receiving/executing object (e.g. e-mail attachment)
  Worm: self-replicating infection by passively receiving object that gets itself executed
- **Spyware malware** can record keystrokes, web sites visited, upload info to collection site
- Infected host can be enrolled in **botnet**, used for spam. DDoS attacks

3.9.1 DoS: Denial of Service

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (botnet)
3. send packets to target from compromised hosts

3.9.2 Sniffing

- broadcast media (shared Ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g. including passwords) passing by

3.9.3 IP Spoofing

Send packet with false source address

4 Chapter 2

4.1 Application Architectures

4.1.1 Client-Server

Server: Always-on host, Permanent IP address
Clients: Do not communicate directly with each other, May have dynamic IP addresses

4.1.2 Peer-to-Peer (P2P)

- No always-on server
- Peers request service from other peers, provide service in return to other peers
- **Self Scalability** – new peers bring new service capacity, as well as new service demands

- Pers are intermittently connected and change IP addresses

Note 5: App-layer protocol defines

- **type of messages exchanged** – e.g. request, response
- **message syntax** – what fields in messages and how fields are delineated
- **message semantics** – meaning of information in fields
- **rules** for when and how processes send and respond to messages
- **open protocols** – defined in RFCs, allows for interoperability (e.g. HTTP, SMTP)
- **proprietary protocols** – e.g. Skype

4.2 Transport Service is needed

Data Integrity: Some programs need 100% reliable data transfer (e.g. file transfer, web transactions), others can tolerate loss (e.g. audio)

Timing: Some programs require low delay to be "effective" (e.g. online games)

Throughput: Some programs require minimum amount of throughput to be "effective" (e.g. multimedia), some use whatever they have available ("elastic apps")

Security: Encryption, Data Integrity

4.3 Transport Protocol Services

4.3.1 TCP

Reliable Transport between sending and receiving process

Flow Control: sender won't overwhelm receiver

Congestion Control: throttle sender when network overloaded

Connection-Oriented: setup required between client and server processes

Does Not Provide: timing, minimum throughput guarantee, security

4.3.2 UDP

Unreliable Data Transfer between sending and receiving process

Does Not Provide: reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

4.3.3 Securing TCP

TCP and UDP

- no encryption
- cleartext passwords sent into socket traverse Internet in cleartext

SSL

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- app use SSL libraries, that "talk" to TCP

SSL socket API

- cleartext passwords sent into socket traverse Internet encrypted

4.4 HTTP: Hypertext Transfer Protocol

- Web's application layer protocol
- client/server model. Client request website and server serves HTTP object in response
- Uses TCP
- HTTP is stateless. Server maintains no information about past client requests
- **non-persistent HTTP:** one object sent over one TCP connection, downloading multiple object required multiple connections
- **persistent HTTP:** multiple object sent over single TCP connection

Non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

4.4.1 Method Types

HTTP/1.0: GET, POST, HEAD (asks server to leave requested object out of response)

HTTP/1.1: GET, POST, HEAD, PUT (uploads file in entity body to path specified in URL field), DELETE (deletes file specified in the URL field)

4.4.2 Response Codes

200 OK: request succeeded, requested object later in this msg
301 Moved Permanently: requested object moved, new location specified later in this msg
400 Bad Request: request msg not understood by server
404 Not Found: requested document not found on this server
505 HTTP Version Not Supported

4.5 Cookies

Uses: authorization, shopping carts, recommendations, user session state (Web, email)

4.6 Web Caches (proxy server)

> **Goal:** satify client request without involving origin server

- Browsers requests object from cache, if in cache the object is sent back otherwise cache requests object from origin
- Cache acts as both client and server
- Reduce response time for client request
- Reduce traffic

4.6.1 Conditional GET

> **Goal:** don't send object if cache has up-to-date cached version (lower link usage)

- **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: `HTTP/1.0 Not Modified`

4.7 Electronic Mail: SMTP

RFC 2821

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer: handshaking, transfer of messages, closure
- command/response interaction
- messages must be in 7-bit ASCII
- uses persistent connections
- requires message to be in 7-bit ASCII
- uses `CRLF.CRLF` to determine end of message

Difference to HTTP being, HTTP is server sending data, SMTP is client connection sending data

SMTP: protocol for exchanging email messages
RFC 822: standard for text message format (To, From, Subject, Body)

4.7.1 Mail Access Protocols

> **SMTP:** delivery/storage to receiver's server

POP: Post Office Protocol *(RFC 1939)*: authorization, download
- POP3 is stateless across sessions
- Two main modes; download and delete, download and keep (allows multiple clients to read the same email)

IMAP: Internet Mail Access Protocol *(RFC 1730)*: more features, including manipulation of stored message on server
- All messages stored on server
- Supports folders
- Keeps user state across sessions: names of folders and mappings between message IDs and folder name

HTTP: gmail, Hotmail, Yahoo, etc

4.8 DNS: Domain Name System

- Lookup between names (e.g. google.com) and IP addresses
- **Distributed Database** implemented in hierarchy of many **name servers**
- **Application-layer protocol:** hosts, name servers communicate to **resolve** names (address/name translation)

Why not centralize DNS? Single point of failure, traffic volume, doesn't scale

4.8.1 DNS Services

- hostname to IP address translation
- host aliasing (canonical, alias names)
- mail server aliasing
- load distribution (many IP addresses correspond to one name)

4.8.2 TLD, authoritative servers

top-level domain (TLD) servers:
- responsible for com, org, net, edu, aero, jos, io
- and top-level country domains au, uk, ca
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

Authoritative DNS servers:
- organization's own DNS server(s), providing authorative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

4.8.3 Local DNS name server

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one (also called "default name server")
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

4.8.4 DNS Name Resolution

Iterated query: contacted server replies with name of server to contact. So root dns sends the ip of the next dns server to contact

Recursive query: puts burden of name resolution on contacted name server. So root dns server contacts the next levels down which contacts next level down.

4.8.5 Caching

Once (any) name server learns mapping, it **caches** mapping. Cache entries timeout (disappear) after some time (TTL). If name host changes IP address, the name servers might not update until TTLs expire.

> update/notify mechanisms proposed IETF standard RFC 2136

4.8.6 DNS Records

Note 6: RR Format

`(name, value, type, ttl)`

type=A `name` is hostname, `value` is IP address

type=NS `name` is domain (e.g. google.com), `value` is hostname of authoritative name server for this domain

type=CNAME `name` is alias name for some "canonical" (the real) name (`www.ibm.com` is really `servereast.backup2.ibm.com`), `value` is canonical name

type=MX `value` is name of mailserver associated with `name`

4.8.7 Protocol

Query and reply messages both follow same format

Table 2: Protocol Layout

2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

4.8.8 Attacking DNS

DDoS attacks
- bombard root servers with traffic. Not successful to date, traffic filtering, local DNS servers cache protecting root DNS
- bombard TLD server. Potentially more dangerous

Redirect Attacks
- man-in-middle (Intercept queries)
- DNS Poisoning (Send bogus relies to DNS server, which caches)

Exploit DNS for DDoS
- send queries with spoofed source address: target IP
- requires amplification

4.9 File Distribution Time

4.9.1 Client-server

Server Tranmission: must sequentially send (upload) N file copies. Time to send one copy: $\frac{F}{u_s}$. Time to send N copies: $\frac{NF}{u_s}$

Client: each client must download file copy. d_{min} = min client download rate. min client download time $\frac{F}{d_{\text{min}}}$

Note 7: Client-server File Distribution

time to distribute F to N clients using client-server approach

$$D_{c-s} \geq \max\{\frac{NF}{u_s}, \frac{F}{d_{\text{min}}}\}$$

4.9.2 P2P

Server Tranmission: must upload at least one copy. Time to send one copy: $\frac{F}{u_s}$

Client: each client must download file copy. Min client download time: $\frac{F}{d_{\text{min}}}$

Clients: as aggregate must download NF bits. Max upload rate (limiting max download rate) is $u_s + \sum u_i$

Note 8: P2P File Distribution

time to distribute F to N clients using P2P approach

$$D_{\text{P2P}} \geq \max\{\frac{F}{u_s}, \frac{F}{d_{\text{min}}}, \frac{NF}{u_s + \sum u_i}\}$$

4.9.3 BitTorrent

File divided into 256Kb chunks

Tracker: tracks peers participating in torrent

Torrent: group of peers exchanging chunks of a file

4.10 Multimedia

4.10.1 Video

Coding: used redundancy **within** and **between** images to decrease # bits used to encode image

Spatial: within image

Temporal: from one image to next

CBR (constant bit rate): video encoding rate fixed

VBR (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes

4.10.2 DASH

DASH: Dynamic, Adaptive Streaming over HTTP

Server: Divides video file into multiple chunks. Each chunk stored, encoded at different rates. **Manifest file:** provides URLs for different chunks

Client: Periodically measures server-to-client bandwidth. Consulting manifest, requests one chunk at a time. Chooses maximum coding rate sustainable given current bandwidth. Can choose different coding rates at different points in time (depending on available bandwidth at time)

"intelligence" at client: client determines

- **when** to request chunk (so that buffer starvation, or overflow does not occur)
- **what encoding rate** to request (higher quality when more bandwidth available)
- **where** to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

4.10.3 Content Distribution Networks (CDNs)

CDN stores copies of content at CDN nodes. Subscriber requests content from CDN, directed to nearby copy, retrieves content, may choose different copy if network path congested.

5 Chapter 3

5.1 Transport vs. Network Layer

Network Layer: logical communication between hosts

Transport Layer: logical communication between processes; relies on, enhances, network layer services

5.2 Multiplexing/demultiplexing

5.2.1 How demultiplexing works

- host receives IP datagrams
  - each datagram as source IP address, destination IP address
  - each datagram carries one transport-layer segment
  - each segment has source, destination port number
- host uses **IP addresses and port numbers** to direct segment to appropriate socket

Connectionless Demultiplexing

A UDP socket needs to have a local port number assigned to it (both client and server)

Connection-oriented demux

TCP socket identified by 4-tuple: **(source IP address, source port number, dest IP address, dest port number)**

5.3 UDP

Table 3: UDP Segment Header

32 bits	
source port #	dest port #
length	checksum
application data (payload)	

5.3.1 UDP Checksum

Sender:

- treat segment contents, including header fields, as sequence of 16-bit integers
- checksum: addition (one's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

Receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value

5.4 Pipelined Protocols

Pipelining: sender allows multiple, "in-flight", yet-to-be-acknowledged packets. Range of sequence numbers must be increased, buffering at sender and/or receiver.

5.4.1 Go-Back-N

- sender can have up to$N$ unacked packets in pipeline
- receiver only sends **cumulative ack**. Doesn't ack packet if there's a gap
- sender has timer for oldest unacked packet. When timer expires, retransmit all unacked packets

5.4.2 Selective Repeat

- sender can have up to N unacked packets in pipeline
- receiver sends **individual ack** for each packet
- sender maintains timer for each unacked packet. When timer expires, retransmit only that unacked packet

5.5 TCP Segment Structure

TCP contains a handshake to make sure both ends are willing to open a connection

Table 4: TCP Segment Structure

32 bits	
source port #	dest port #
sequence number	
acknowledgment number	
(head len, not used, UAPRSF)	receive window
checksum	urg data pointer
options (variable length)	
application data (variable length)	

sequence number, acknowledgment number: counting by bytes of data (not segments)
U: urgent data (generally not used)
A: ACK # valid
P: push data now (generally not used)
RSF: RST, SYN, FIN; connection established (setup, teardown commands)
checksum: Internet checksum (as in UDP)
receive window: # bytes receiver willing to accept

Sequence Numbers: byte stream "number" of first byte in segment's data
Acknowledgements: sequence # of next byte expected from other side, cumulative ACK

5.5.1 TCP Round Trip Time, Timeout

$$E = (1 - \alpha) \times E + \alpha \times \mathit{SampleRTT}$$

Where E is EstimatedRTT. Influence of past sample decreases exponentially fast. Typical value: $\alpha = 0.125$

$$\mathit{TimeoutInterval} = E + 4 \times \mathit{DevRTT}$$

Where *DevRTT* is the safety margin ($\mathit{DevRTT} = (1-\beta) \times \mathit{DevRTT} + \beta \times \mid \mathit{SampleRTT} - E \mid$(typically, $\beta = 0.25$))

5.5.2 TCP Flow Control

- receiver “advertises” free buffer space by including `rwnd` value in TCP header of receiver-to-sender segments
  - `RcvBuffer` size set via socket options (typical default is 4096 bytes)
  - many operating systems autoadjust `RcvBuffer`
- sender limits amount of unacked (“in-flight”) data to receiver’s `rwnd` value
- guarantees receive buffer will not overflow

5.5.3 Closing

- client, server each close their side of connection (send TCP segment with FIN bit 1)
- respond to received FIN with ACK (on receiving FIN, ACK can be combined with own FIN)
- simultaneous FIN exchanges can be handled

5.6 TCP Congestion Control

Approach: sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs

Additive Increase: increase `cwnd` by 1 MSS every RTT until loss detected

Multiplicative Decrease: cut `cwnd` in half after loss

5.7 Fairness

TCP is fair because:

- additive increase gives slope of 1, as throughput increases
- multiplicative decrease decreases throughput proportionally

UDP is not fair:

- do not want rate throttled by congestion control
- send audio/video at constant rate, tolerate packet loss

5.8 Explicit Congestion Notification

Network-assisted Congestion Control:

- two bits in IP header (ToS field) marked by **network router** to indicated congestion
- congestion indication carried to receiving host
- receiver (seeing congestion indication in IP datagram) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion

6 Chapter 4

6.1 Network Layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers segments to transport layer
- network layer protocols in **every** host, router
- router examines header fields in all IP datagrams passing through it

6.1.1 Network Layer Functions

Forwarding: move packets from router’s input to appropriate router output

Routing: determine route taken by packets from source to destination *(routing algorithms)*

6.1.2 Data Plane, Control Plane

Data Plane

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:

  Traditional Routing Algorithms: implemented in routers

  Software-defined networking (SDN): implemented in (remote) servers

6.2 Router Forwarding

Destination-based forwarding: forward based only on destination IP address (traditional)

Generalized forwarding: forward based on any set of header field values

6.2.1 Destination-based forwarding

A link interface is assigned to a range of destination address ranges

Note 9: Longest Prefix Matching

When looking for forwarding table entry for given destination address, use **longest** address prefix that matches destination address. Longest prefix matching: often performed using ternary content addressable memories (TCAMs). Cisco Catalyst can hold up ≈1M routing table entries in TCAM.

Content Addressable: present address to TCAM; retrieve address in one clock cycle, regardless of table size

6.2.2 Switching Fabrics

- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfered from inputs to outputs (often measured as multiple of input/output line rate, N inputs: switching rate N times line rate desirable)
- three types of switching fabrics

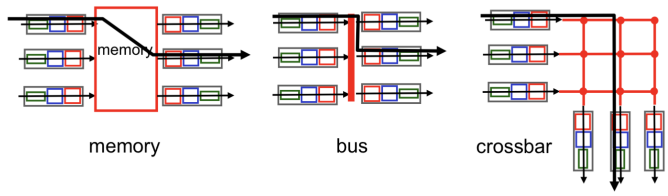

Figure 2: Different Types of Switching Fabrics

Switching via Memory
- traditional computes with switching under direct control of CPU
- packet copied to system's memory
- speed limited memory bandwidth (2 bus crossing per datagram)

Switching via a Bus
- datagram from input port memory to output port memory via a shared bus
- **bus contention:** switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

Switching via Interconnection Network
- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric
- Cisco 12000: switches 60 Gbps through the interconnection network

6.2.3 Input port queuing

- fabric slower than input ports combined → queuing may occur at input queues (queuing delay and loss due to input buffer overflow)
- **Head-of-the-Line (HOL) blocking:** queued datagram at front of queue prevents others in queue from moving forward

6.2.4 Output ports

- **Buffering** required from fabric faster rate (Datagram (packets) can be lost due to congestion, lack of buffers)
- **Scheduling** datagrams (Priority scheduling – who gets best performance, network neutrality)

Note 10: How much buffering?

RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C (e.g. C = 10 Gbps link, 2.5 Gbit buffer). Recent recommendation with N flows, buffering equal to

$$\frac{RTT \times C}{\sqrt{N}}$$

6.2.5 Scheduling Mechanisms

Scheduling: choose next packet to send on link

FIFO scheduling: send in order of arrival to queue

discard policy: if packet arrives to full queue, who to discard

tail drop: drop arriving packet

priority: drop/remove on priority basis

random: drop/remove randomly

priority scheduling: send highest priority queued packet. Multiple *classes*, with different priorities (class may depend on marking or other header info, e.g. IP source/dest, port number, etc)

RR scheduling: multiple classes. Cyclically scan class queues, sending one complete packet from each class (if available)

WFQ scheduling: generalized Round Robin. Each class gets weighted amount of service in each cycle

6.3 IP

6.3.1 IP Datagram Format

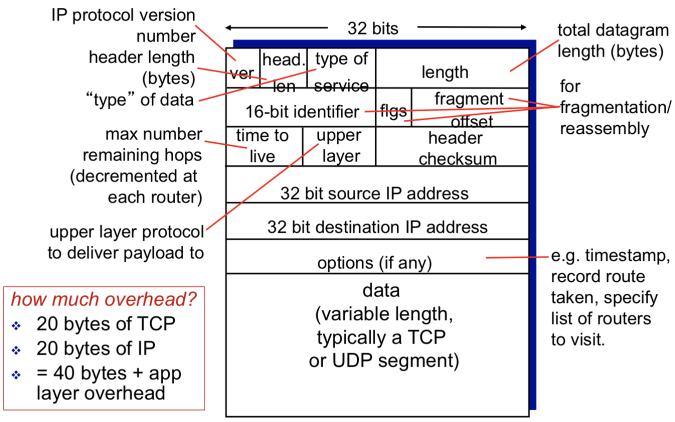

Figure 3: IP Datagram Format

6.3.2 IP Fragmentation, Reassembly

Large IP datagram divided ("fragmented") within net

- one datagram becomes several datagrams
- "reassembled" only at final datagrams
- IP header bits used to identify, order related fragments

6.3.3 IP Addressing

IP Address: 32-bit identifier for host, router interface

interface: connection between host/router and physical link. Router's typically have multiple interfaces

6.3.4 Subnets

Subnet part – high order bits. **Host part** – low order bits

- device interfaces with same subnet part of IP address
- can physically reach each other **without intervening router**
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- each isolated network is called a **subnet**

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: `a.b.c.d/x`, where `x` is # bits in subnet portion of address

6.3.5 DHCP: Dynamic Host Configuration Protocol

Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected)
- support for mobile users who want to join network (more shortly)

DHCP overview:

- host broadcasts "DHCP discover" msg *[optional]*
- DHCP server responds with "DHCP offer" msg *[optional]*
- host requests IP address: "DHCP request" msg
- DHCP server sends address: "DHCP ack" msg

DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS server
- network mask (indicating network versus host portion of address)

6.3.6 ICANN

- allocates addresses
- manages DNS
- assigns domain names, resolves disputes

6.3.7 NAT

All datagrams **leaving** local network have **same** single source NAT IP address

Motivation: local network uses just one IP address as far as outside world is concerned

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicity addressable, visible by outside world (a security plus)

Implementation: NAT router must

Outdoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #) ... remote clients/servers will respond using (NAT IP address, new port #) as destination address

Remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair

Incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

- 16-bit port-number field: 60,000 simultaneous connections with a single LAN-side address
- NAT is controversial:
  - routers should only process up to layer 3
  - address shortage should be solved by IPv6
  - violates end-to-end argument (NAT possibility must be taken into account by app designers, e.g. P2P applications)
  - NAT traversal: what if client wants to connect to server behind NAT?

6.3.8 IPv6

> 32-bit address space soon to be completely allocated

Additionally:

- header format helps speed processing/forwarding
- header changes to facilitate QoS

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

6.3.9 IPv6 Datagram Format

Priority: identify priority among datagrams in flow

Flow Label: identify datagrams in same "flow" (concept of "flow" not well defined)

Next Header: identify upper layer protocol for data

Table 5: IPv6 Format

32 bits		
version	pri	flow label
payload len	next hdr	hop limit
source address (128 bits)		
destination address (128 bits)		
data		

6.3.10 Other changes from IPv4

checksum: removed entirely to reduce processing time at each hop

options: allowed, but outside of header, indicated by "Next Header" field

ICMPv6: new version of ICMP (additional message types e.g. "Packet Too Big", multicast group management functions)

6.3.11 Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously (no "flag days", how will network operate with mixed IPv4 and IPv6 routers)
- **tunneling:** IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers

6.4 Generalized Forwarding and SDN

Each router contains a **flow table** that is computed and distributed by a logically centralized routing controller

6.4.1 OpenFlow data plane abstraction

Flow: defined by header fields

Generalized Forwarding: simple packet-handling rules

Pattern: match values in packet header fields

Actions: for matched packet: drop, forward, modify, matched packet and send matched packet to controller

Priority: disambiguate overlapping patterns

Counters: # bytes and # packets

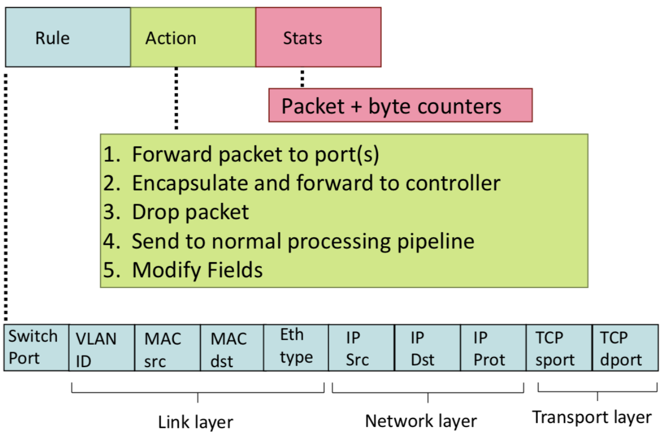

Figure 4: Flow Table Entries

6.4.2 OpenFlow Abstraction

- **Match+Action:** unifies different kinds of devices
- Router
  match: longest destination IP prefix

action: forward out a link

- Switch
  match: destination MAC address
  action: forward or flood
- Firewall
  match: IP addresses and TCP/UDP port numbers
  action: permit or deny
- NAT
  match: IP address and port
  action: rewrite address and port

7 Chapter 5

7.1 Control Plane

Two approaches to structuring network control plane:

7.1.1 Per-router control plane

> Individual routing algorithm components **in each and every router** interact with each other in control plane to compute forwarding tables

7.1.2 Logically centralized control plane

> A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

7.2 Routing Protocols

> **Routing protocol goal:** determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers
> **path:** sequence of routers packets will traverse in going from given initial source host to given final destination host
> **"good":** least "cost", "fastest", "least congested"

7.2.1 Global or Decentralized information

Global:

- all routers have complete topology, link cost info
- **"link state" algorithms**

Decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- **"distance vector" algorithms**

7.2.2 Static or Dynamic

Static:

- routes change slowly over time

Dynamic:

- routes change more quickly (periodic update, in response to link cost changes)

7.2.3 Link-State Routing Algorithm

Note 11: Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source') to all other nodes (gives **forwarding table** for that node)
- iterative: after k iterations, know least cost path to k destinations

Notation:

$c(x, y)$: link cost from node x to y; $= \infty$ if not direct neighbors
$D(v)$: current value of cost path from source to destination v
$p(v)$: predecessor node along path from source to v
N': set of nodes whose least cost path definitively known

Algorithm Complexity: n nodes

- each iteration: need to check all nodes, W, not in$N$
- $\frac{n(n+1)}{2}$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log n)$

7.2.4 Distance Vector Algorithm

Note 12: Bellman-Ford equation

(dynamic programming)

let $d_x(y) :=$ cost of least-cost path from x to y then

$$d_x(y) = \min\{c(x, v), d_v(y)\}$$

$c(x, v)$: cost to neighbor v
$d_v(y)$: cost from neighbor v to destination y

- $D_x(y) =$ estimate of least cost from x to y (x maintains distance vector $\mathbf{D}_x = [D_x(y) : y \in N]$)

- node x:
  - knows cost to each neighbor $v : c(x, v)$
  - maintains its neighbor's distance vectors. For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y) : y \in N]$

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$D_x(y) \rightarrow \min\{c(x, v)+D_v(y)\}$ for each node $y \in N$

7.2.5 Comparison of LS and DV algorithms

Message Complexity:
LS: with n nodes, E links, $O(nE)$ msgs sent
DV: exchange between neighbors only (convergence time varies)

Speed of Convergence:
LS: $O(n^2)$ algorithm requires $O(nE)$ msgs (may have oscillations)
DV: convergence time varies (may be routing loops, count-to-infinity problem)

Robustness: What happens if router malfunctions?
LS: Node can advertise incorrect *link* cost. Each node computes only its *own* table
DV: DV node can advertise incorrect *path* cost. Each node's table used by others, error propagate through network

7.3 Making Routing Scalable

At the moment, can't store all destinations in routing tables. Routing table exchange would swamp links. Solution: Aggregate routers into regions known as "autonomous systems" (AS) (a.k.a "domains")

intra-AS routing:

- routing among hosts, routers in same AS ("network")
- all routers in AS must run **same** intra-domain protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es

inter-AS routing

- routing among AS'es
- gateways perform inter-domain routing (as well as intra-domain routing)

7.3.1 Interconnected ASes

Forwarding table configured by both intra-AS and inter-AS routing algorithm

- intra-AS routing determine entries for destinations within AS
- inter-AS and intra-AS determine entries for external destinations

7.3.2 Intra-AS Routing

Also known as **Interior Gateway Protocols (IGP)**. Most common intra-AS routing protocols:

RIP: Routing Information Protocol
OSPF: Open Shortest Path First (IS-IS protocol essentially same as OSPF)
IGRP: Interior Gateway Routing Protocol (Cisco proprietary *for decades, until 2016*)

Note 13: OSPF (Open Shortest Path First)

- "open": publicly available
- uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- router floods OSPF link-state advertisements to all other routers in **entire** AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
  - link state: for each attached link
- **IS-IS routing** protocol: nearly identical to OSPF

"Advanced Features"

- **security:** all OSPF messages authenticated (to prevent malicious intrusion)
- **multiple** same-cost **paths** allowed (only one path in RIP)
- for each link, multiple cost metrics for different **TOS** (e.g. satellite link cost set low for best effort ToS; high for real-time ToS)
- integrated uni- and **multi-cast** support (Multicast OSPF (MOSPF) uses same topology data base as OSPF)
- **hierarchical** OSPF in large domains

7.3.3 Hierarchical OSPF

two-level hierarchy: local area, backbone

- link-state advertisements only in area

- each nodes has detailed area topology; only know direction (shortest path) to nets in other areas

area border routers: "summarize" distances to nets in own area, advertise to other Area Border routers

backbone routers: run OSPF routing limited to backbone

boundary routers: connect to other AS'es

7.3.4 Internet inter-AS routing

Note 14: BGP (Border Gateway Protocol)

- The de facto inter-domain routing protocol
- BGP provides each AS a means to:
  - **eBGP:** obtain subnet reachability information from neighboring ASes
  - **iBGP:** propagate reachability information to all AS-internal routers
  - determine "good" routes to other networks based on reachability information and **policy**
- allows subnet to advertise its existence to rest of Internet: "I am here"

7.4 Software Defined Networking (SDN)

Internet network layer: historically has been implemented via distributed, per-router approach

- **monolithic** router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g. Cisco IOS)
- different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ...

Why a logically centralized control plane?

- easier network management: avoid router misconfiguration, greater flexibility of traffic flows
- table-based forwarding (recall OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming" more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- open (non-proprietary) implementation of control plane

7.4.1 SDN perspective: Data Plane Switches

- fast, simple, commodity switches implementing generalized data-place forwarding in hardware
- switch flow table computed, installed by controller
- API for table-based switch control (e.g. OpenFlow) – defines what is controllable and what is not
- protocol for communicating with controller (e.g. OpenFlow)

7.4.2 SDN perspective: SDN controller

- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness

7.4.3 SDN perspective: Control Applications

- "brains" of control: implement control functions using lower-level services, API provided by SND controller
- unbundled: can be provided by 3^{rd} party: distinct from routing vendor, or SDN controller

Note 15: OpenFlow Protocol

- operates between controller, switch
- TCP used to exchange messages (optional encryption)
- thress classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

7.4.4 OpenFlow: controller-to-switch messages

Key controller-to-switch messages

features: controller queries switch features, switch replies

configure: controller queries/sets switch configuration parameters

modify-state: add, delete, modify flow entries in the OpenFlow tables

packet-out: controller can send this packet out of specific switch port

packet-in: transfer packet (and its control) to controller. See packet-out message from controller

flow-removed: flow table entry deleted at switch

port status: inform controller of a change on a port

> Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

7.5 OpenDaylight (ODL) controller

- ODL Lithium controller
- network apps may be contained within, or be external to SDN controller
- Service Abstraction Layer: interconnects internal, external applications and services

7.6 ONOS controller

- control apps separate from controller
- intent framework: high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

7.7 ICMP: Internet Control Message Protocol

Table 6: ICMP Types and Codes

Type	Code	Description
0	0	echo reply (ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control – not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

- used by hosts and routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP: ICMP messages carried in IP datagrams
- **ICMP message:** type, code, plus first 8 bytes of IP datagram causing error

7.8 Network Management and SNMP

> "Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

Managed devices contain **managed objects** whose data is gathered into a **Management Information Base (MIB)**

7.8.1 SNMP Protocol: Message Types

Table 7: SNMP Message Types

Message type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: "get me data" (data instance, next data in list, block of data)
InformRequest	manager-to-manager: here's MIB value
SetRequest	manager-to-agent: set MIB value
Response	agent-to-manager: value, response to request
Trap	agent-to-manager: inform manager of exceptional event

Table 8: SNMP Message Formats

			$\leftarrow$ SNMP PDU $\rightarrow$				
	$\leftarrow$	Get/set header	$\rightarrow$	$\leftarrow$	Variables to get/set		$\rightarrow$
PDU Type	Request ID	Error Status	Error Index	Name	Value	Name	Value
(0 - 3)		(0 - 5)					
	$\leftarrow$		Trap header		$\rightarrow$	$\leftarrow$ Trap info $\rightarrow$	
PDU Type	Enterprise	Agent Addr	Trap Type	Specific code	Timestamp	Name	Value
4		(0 - 7)					

8 Acronyms

IP: Internet Protocol
TCP:
UDP:
HTTP: Hypertext Transfer Protocol
SMTP: Simple Mail Transfer Protocol
RDP: Remote Desktop Protocol
VOIP: Voice over IP
RTT:
POP: Post Office Protocol
IMAP: Internet Mail Access Procotol
DNS: Domain Name System
SSN:
TLD: Top-level Domain
TTL: Time To Live
RR: Resource Records
DDoS:
CBR: Constant bit rate
VBR: Variable bit rate
ABR:
UBR:
DASH: Dynamic, Adaptive Streaming over HTTP
CDN: Content Distribution Networks
RDT: Reliable Data Transfer
MSS:
ECN: Explicit Congestion Notification
ECE:
SDN: Software-defined networking
TCAMs: Ternary Content Addressable Memories
HOL: Head-of-the-Line
FIFO: First in first out
RR: Round Robin
WFQ: Weighted Fair Queuing
CIDR: Classless InterDomain Routing
DHCP: Dynamic Host Configuration Protocol
ICANN: Internet Corporation for Assigned Names and Numbers
NAT: Network Address Translation
QoS:
OSPF:
ODL:
ONOS:
ICMP: Internet Control Message Protocol
SNMP:
CA: Control Agent
DV: Distance Vector
B-F: Bellman-Ford
LS: Link State
AS: Autonomous Systems
IGP: Interior Gateway Protocols
RIP: Routing Information Protocol
OSPF: Open Shortest Path First
IGRP: Interior Gateway Routing Protocol
ToS:
MOSPF: Multicast OSPF
BGP: Border Gateway Protocol
SDN: Software Defined Networking
SNMP:
ODL: OpenDaylight
SAL: Service Abstraction Layer
OVSDB:
MIB: Management Information Base