CS2105: Overview

Internet (In General)

- Network of connected devices known as **hosts** or **end systems**
- Hosts run **network applications** (such as browsers, instant messaging, video conferencing software)
- **Access network** – where hosts access the Internet

Physical Media

- **Guided media** – solid media (e.g. fiber)
- **Unguided media** – freely propagate (e.g. Wi-Fi, cellular)

Network Implementations

- **Circuit Switching** – dedicated circuit for every call
- **Packet Switching** – break messages to smaller chunks, packets, of length L
  - **Store and forward** – entire packet must arrive before it is transmitted

Delays

- **(Nodal) Processing Delay:** check for bit errors, determining output link ($<$ 1ms)
- **Queuing Delay:** Time waiting in queue for transmission. Dependent on congestion level of router
- **Transmission Delay:** time required to push bits to the link. $d_{trans} = \frac{L}{R}$, where L is the packet length (bits) and R is the link bandwidth (bps)
- **Propagation Delay:** time to send packet through the link. $d_{prop} = \frac{d}{s}$, where d is the length of physical link and s is the propogation speed in medium (approx. 2×10^8 m/s)
- **End to end Delay:** time taken for a packet to be transmitted across a network $d_{end-end} = N[d_{trans} + d_{prop} + d_{proc} + d_{queue}]$

Internet Protocol Stack

Layer	Example
Application	FTP, SMTP, HTTP
Transport	TCP, UDP
Network	IP, Routing
Link	Ethernet, 802.11, PPP
Physical	actual wire

CS2105: Application Layer

Basics

Architectures

Client-Server Architecture

- Two types of hosts – client & server
- **Server** waits for incoming requests & serve requested service to clients
- **Client** initiates contact with server & request for service from server (example: web browser)

P2P Architecture

- No always-on server
- Arbitrary end systems directly communicate with each other (peers request from other peers)

Properties of Transport Services

- Data integrity – some applications require 100% reliability (e.g. file transfer)
- Timing – low delay (e.g. games)
- Throughput – bandwidth (e.g. media streaming requires higher bandwidth)
- Security – encrypting, data integrity, authentication, etc...

Properties of App-Layer Protocols

- **Types** of messages exchanged (request, response)
- **Syntax** of messages (fields, delimiters)
- **Semantics** of messages
- **Rules** for when and how applications send & respond to messages
- **Open** vs **Proprietary** protocols

Hypertext Transfer Protocol

Basics

- Client/Server Model
- Uses TCP as transport service
- **Stateless** – no memory of previous transactions. Use cookies to carry over state

HTTP Request Message

- `<request-type> <resource> HTTP/<version>`
  `Host: www.example.com`
  `User-Agent: example`
  `\r \n`
  Host header is required in HTTP 1.1

HTTP Response Message

- `HTTP/<version> <status-code> <status-name>`
  `Date: Thu, 15 Jan 2015 13:02:41 GMT`
  `Server: Example`
  `\r \n`
  `<data>`
  Host header is required in HTTP 1.1

HTTP Response Status Codes

- Status Codes are categorised into five classes
  - 1xx - Informational
  - 2xx - Success
  - 3xx - Redirection
  - 4xx - Client Error
  - 5xx - Server Error
- Example (most common) status codes
  - 200 OK
  - 301 Moved Permanently – location specified in Location header
  - 304 Not Modified
  - 403 Forbidden
  - 404 Not Found

HTTP 1.0

- Non-persistent – need to establish connection for every request (TCP socket closes)

Response Time

- 2 * RTT + file transmission time
- First RTT to establish TCP connection
- Second RTT to perform actual HTTP request

HTTP 1.1

- Persistent – subsequent HTTP requests can be sent over same TCP connection

Conditional GET

- Specify `If-modified-since` header in request
- Server replies with status code `304 Not Modified` if cached resource is up to date

Domain Name System (DNS)

Basics

- Provides **translation** between hostname and IP address
- Stored in a distributed, hierarchical manner – query root DNS server first
- **Root Servers** – returns authoritative NS for appropriate top-level domains
- **Authoritative servers** – DNS servers that provide authoritative records for organisation hosts

Resource Records (RR)

- Host name records are stored as resource records
- **Format** – `(name, value, type, ttl)`

RR Types

- **A** – maps hostname with IP address
- **NS** – maps domain to authoritative name server for domain
- **CNAME** – maps name to a "canonical" name
- **MX** – maps name to a mail server

Example of App-Layer Protocols

Protocol	Transport Protocol	Standard Port #
DNS	TCP/**UDP**	53
SMTP	TCP	25/587/2525 (Secured: 465, 25025)
Telnet	TCP	23
HTTP	TCP	80/443 (SSL)
SIP	TCP/UDP	5060/5061
RTP	UDP	Varies

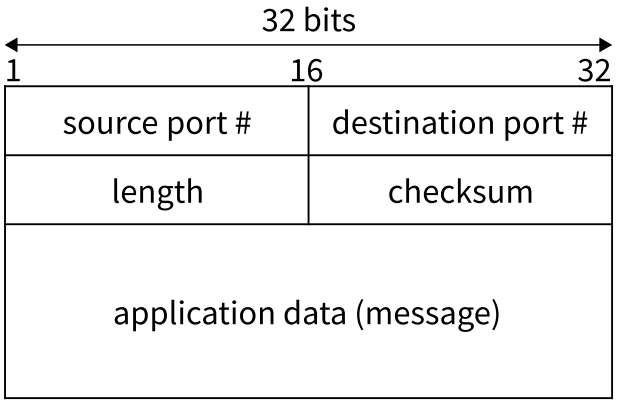
CS2105: Transport Layer

Basics

- Process-to-process communication
- **Example** of transport protocols
  Transmission Control Protocol (TCP) & User Datagram Protocol (UDP)
- Transport layer protocols run in **hosts**
  - **Sender** breaks application layer message into **segments** (if needed) & passes them to the network layer (e.g. IP)
  - **Receiver** reassemble segments and passes message to application layer
  - **Packet switches** (e.g. routers) check destination IP address to decide routing

User Datagram Protocol (UDP)

Datagram Format



Basics & Motivation

- **Unreliable** protocol, for loss tolerant & rate sensitive applications
- **Not much** service on top of IP
  - Connectionless multiplexing/demultiplexing
  - Checksum

Advantages of UDP

- No need to establish connection (minimal delay)
- Simple – no connection state at both sender and receiver
- Small header size
- **No congestion control**

UDP Checksum

- Detection of errors (flipped bits) in segment
- **Optional**, can be set to zeros

Checksum Calculation

- Treat UDP segment as sequence of **16-bit integers**
- **Binary add** every **16-bit integer**
- Carry from MSB is added to result
- Compute the **1s complement** of result

Reliable Data Transfer (rdt)

Basics

- Creating **reliable transport layer protocol** on top of **unreliable network layer link**

Reasons for Unreliability

- Corruption of packets
- Packets dropped (i.e. never reach receiver)
- Re-ordering of packets
- Packets delayed

rdt 1.0 – Perfectly Reliable

- No special implementation required – just send and receive!

rdt 2.0 – Bit Errors

- Channel may **flip bits** in packet
- Use **checksum** to detect bit errors
- **Recovering** from bit errors – receiver must send Acknowledgement (ACK) or Negative Acknowledgement (NAK) to notify sender if packet is received successfully
- Sender **retransmits** packet on NAK or when acknowledgement is corrupted
- **Issue** – receiver does not know if packet sent by sender is duplicate or next packet

rdt 2.1 – rdt 2.0 + Sequencing

- Handles **duplicate** packets by attaching **sequence number** to each packet
- Receiver **discards** duplicate packet (**MUST** still ACK)

rdt 2.2 – NAK-free version of 2.1

- **ACK** packet contains sequence number of the packet being ACKed
- Receiver sends **ACK** for the **last packet received OK**
- Sender **retransmits packet** if **duplicate ACK** is received

rdt 3.0 – Channel with Errors and Loss

Assumptions on Underlying Channel

- Bits may be **flipped**
- Packets may be **lost**
- **Long delay** in packet transmission

Implementation

- Sender waits for a "reasonable" amount of time for ACK – **retransmit** if no ACK received

Pipelining

- Allow **multiple**, in-flight, yet-to-be-acknowledged packets
- Same assumption as rdt 3.0

Go-back-N

- **Sender**
  - Can have N unACKed packets in pipeline
  - Maintain **sliding window** – keep track of unACKed packets
  - **Timer** for oldest unACKed packet
  - On **timeout**, retransmit all packets in window
- **Receiver**
  - ACK packets that arrive **in-order**
  - **Cumulative** ACK
  - **Discard** out of order packets and **ACK last in**

order packet

Selective Repeat

- **Receiver** sends **individual ACK** for correctly received packets (out-of-order packets are buffered)
- **Sender** maintains timer for each unACKed packet (on **timeout**, retransmit only **that packet**)

Transmission Control Protocol

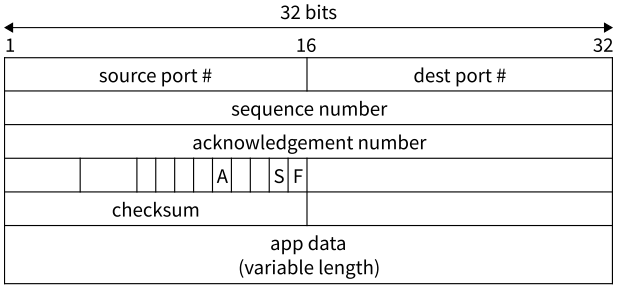
Basics

- **Connection (Socket)** is identified by 4-tuple – `(srcIP, srcPort, destIP, destPort)`
- **Maximum Segment Size (MSS)** – largest amount of data (no headers), specified in bytes

Properties of TCP

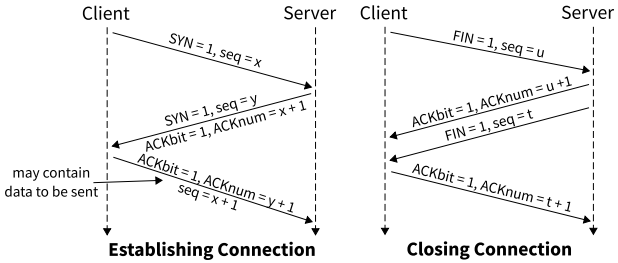
- **Connection-oriented** – handshake before sending application data
- **Reliable, in-order byte stream**
- **Flow & congestion control**

Header Format



- **TCP Sequence Number** – "byte number" of the first byte of data in segment (starting number chosen arbitrarily, client & sender may have different seq num)
- **TCP ACK Number** – seq num of next byte of data expected by receiver (**cumulative ACK**)
  - TCP specifications does not outline how receiver should handle out-of-order segments!

Establishing & Closing Connection



TCP Receiver Events

Event	Action
Received in-order segment (All data already ACKed)	Delayed ACK (500ms), wait for next segment
Received in-order segment (ACK pending for other segment)	Send single cumulative ACK, ACKing both segments
Received out-of-order segment (higher than expected)	Send duplicate ACK immediately (seq # of next expected byte)
Received segment that fills a gap	Send cumulative ACK immediately (if possible)

TCP Sender Events

Event	Action
New data from application	Start timer (if not running), pass segment to network layer & increment sequence #
Timer timeout	Retransmit oldest unACKed packet (and restart timer)
Received ACK	Start timer for oldest unACKed packet

Timeout

- Timeout interval must be longer than RTT!
- **Calculation** of Timeout

$$\text{EstRTT} = (1-\alpha) * \text{EstRTT} + \alpha * \text{SampleRTT}$$

$$\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstRTT}|$$

$$\text{TimeoutInterval} = \text{EstRTT} + 4 * \text{DevRTT}$$

Fast Retransmission

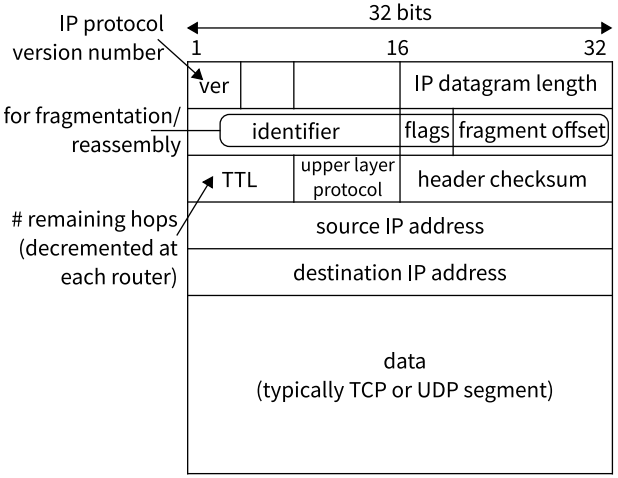
- Resend segment after receiving 3 duplicated ACKs

CS2105: Network Layer

- Network layer delivers packets to receiving hosts

Internet Protocol (IP)

Datagram Format



- **IP header** – 20 bytes

Fragmentation & Reassembly

- **Maximum Transfer Unit** (MTU) – maximum amount of data link-level frame can carry
- Links may have different MTUs – datagrams need to be fragmented
- Destination host will reassemble packet
- **Flag (frag flag)** is set to 1 if the next datagram is from the same segment, else 0. **Offset** is in unit of 8 bytes!

Internet Control Messaging Protocol

- Used by hosts & routers to communicate **network-level** information
- **Examples** – error reporting (unreachable host/network/port/protocol) & echo request/reply (ping)
- ICMP messages are carried **in** IP datagrams (with upper layer protocol number 1)
- **ICMP header** – type + code + checksum + etc

Type	Code	Description
8	0	echo request (ping)
0	0	echo reply (ping)
3	1	dest host unreachable
3	3	dest port unreachable
11	0	TTL expired
12	0	bad IP header

IPv4 Addressing

- Identifier for hosts (or router), associated with a network interface
- **32-bit** integer, expressed in binary or decimal
- Can be assigned **manually** or through DHCP
- **Special IP Addresses**
  - `0.0.0.0/8` – Current network (non-routable)
  - `127.0.0.0/8` – Loopback Address
  - `10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16` – Private Network Addresses
  - `255.255.255.255/32` – Broadcast (all hosts in subnet will receive datagram)
- **Subnet** – network formed by group of "directly" interconnected hosts
  - Hosts in same subnet has same network prefix
  - Can reach each other without router
  - Connected to outside world with router
- **Classless Inter-domain Routing (CIDR)** `a.b.c.d/x`, where x is the number of bits in subnet prefix of IP address
- **Subnet mask** – determines which subnet IP address belongs to (set subnet prefix bits to 1 and host bits to 0)
- **Longest Prefix Match** – match the longest prefix of subnet mask in forwarding table of router, to determine next hop

Dynamic Host Configuration Protocol (DHCP)

- Allows hosts to dynamically obtain IP address when joining network
- Runs on UDP (Port 68 (client), 67 (server))
- Renewable, reusable & support for mobile users
- **4-step process**
  - Host broadcasts "DHCP discover"
  - DHCP responds "DHCP offer" (with IP address)
  - Host broadcasts "DHCP request"
  - DHCP responds "DHCP ACK"
- DHCP may also provided additional network info – IP of first-hop router, local DNS & network mask

Network Address Translation (NAT)

- **Basic Methodology**
  - **Outgoing datagram** – replace (source IP, port #) with (NAT router IP, port #)
  - **Remembers** (source IP, port #) ↔ (NAT router IP, new port #) mapping in **NAT translation table**
  - **Incoming datagram** – replace (NAT router IP, new port #) with (source IP, port #) stored in NAT translation table
- **Motivations/Benefits**
  - Each client does not need public IP, only one public IP for NAT router
  - Clients in network can change private IP addresses without notifying outside world
  - Change ISP without changing addresses of clients in local network
  - Clients in private network are **not explicitly** addressable and visible to outside world

Internet Routing

- "Network-of-networks" – hierarchy of Autonomous Systems (AS) (e.g. ISP)
- **Routing** is done hierarchically
- **Intra-AS** Routing
  - Find path between two routers within AS
  - Under purview of single admin/entity – routing based on performance
  - **Common Protocols** – Routing Information Protocol (RIP), Open Shortest Path First (OSPF)
- **Inter-AS** Routing
  - Handles interfaces between AS

  - Admin often want to control who and how routing is done – policy may dominate performance
  - **De-facto Protocol** – Border Gateway Protocol (BGP)

Intra-AS Routing

Abstract View

- **Graphs** – where vertices are routers and edges are physical links between routers
- Associate a **cost** to each edge – can be 1, inversely related to bandwidth or related to congestion
- **Routing** – finding least cost path between two vertices
- **Graph Notations**
  - $c(x, y)$ – cost of link between routers x & y (∞ if x, y are not direct neighbours)
  - $d_x(y)$ – cost of **least-cost path** from x to y

Routing Algorithm Classifications

- **Link State Algorithms** – all routers have knowledge of network topology and link cost
- **Distant Vector Algorithms** – routers only know physically-connected neighbours and link costs. Requires exchanging of **local views** with neighbour

Bellman-Ford Equation

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

Routing Information Protocol

- Implements **distance vector** algorithm
- Cost metric – **hop count** (no consideration of network congestion)
- **Routing table** entries are aggregated subnet masks (routing to destination subnet)
- Exchange routing tables every 30 seconds over UDP port 520
- If no update from neighbouring router for 3 minutes, assume neighbour failed

CS2105: Link Layer

Properties

- **Link layer** sends datagram between adjacent nodes (hosts/routers) over a single link
- **Possible Link Layer Services**
  - **Framing** – encapsulate datagram into frame (adding header & trailer)
  - **Link access control** – coordinate multiple nodes sending frames into **single link**
  - **Reliable delivery** – more prominent in error-prone links (e.g. wireless)
  - **Error detection** – errors due to signal attenuation or noise. Detect presence of errors (either request for retransmission or drop frame)
  - **Error correction** – correcting bit error(s) without retransmission
- Link layer is implemented in **adapter** (semi-autonomous, both link & physical layers)

Error Detection & Correction

- **Error detection schemes** – checksum (used in TCP/UDP/IP), parity checking & CRC (common in link layer)
- **Parity Checking**
  - **Single bit parity** – detect **single bit** errors in data | 0 1 1 1 0 0 0 1 1 0 | 1 |
  - **two-dimensional bit parity** – detect and correct **single bit error** & detect any **two bit** error!

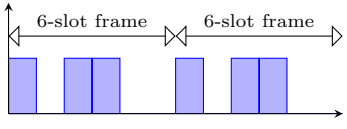
1 0 1 0 1	1
1 1 1 1 0	0
0 1 1 1 0	1
0 0 1 0 1	0

1 0 1 0 1	1
1 0 1 1 0	0
0 1 1 1 0	1
0 0 1 0 1	0

- **Cyclic Redundancy Check (CRC)** – bit-wise XOR operation without carry/borrow

Multiple Access Links & Protocols

- **Type of Network Links**
  - **Point-to-Point** – receiver and node connected by dedicated link (multiple access control not required)
    Examples: Point-to-Point Protocol (PPP), Serial Line Internet Protocol (SLIP)
  - **Broadcast Link** – multiple nodes connected to a shared link/broadcast channel.
    Example: 802.11, Satellite, Ethernet in Bus topology
- **Multiple Access Protocol** – distributed algorithm that determines when/which node can transmit.
  Note: Coordination on channel sharing must use channel itself (no out-of-band signalling)
- **Classes of Multiple Access Protocols**
  - **Channel partitioning** – divide channel into smaller "pieces" (time slot, frequency) and allocate to nodes exclusively
  - **"Taking Turns"** – nodes take turns to transmit
  - **Random Access** – channel not divided (collisions possible). But sender tries to "recover" from collisions
- **Channel Partitioning Protocols**
  - **Time Division Muliple Access (TDMA)** – partition channel by time (fixed length slots), each node gets fixed length slot. Unused slots remain idle (i.e. 6 nodes with 3 transmitting, 3 slots idle)



  - **Frequency Division Multiple Access (FDMA)** – channel partitioned into frequency bands
- **"Taking Turns" Protocols**
  - **Polling** – master node polls slave nodes to transmit
    Disadvantages: Polling overhead & master node is single point of failure
  - **Token Passing** – control token passed from one node to next, sequentially
    Disadvantages: Token overhead & token is single point of failure

Random Access Protocols

- No *a priori* coordination amongst nodes
- Protocols specify **detection & recovery** of collisions
- **Slotted ALOHA**
  - **Assumptions**
    - All frames have equal size
    - Time divided into slots of equal length (length = time to TX one frame)
    - Nodes transmit only at **beginning** of slot
  - **Operation**
    - Listen to channel while transmitting (collision detection)
    - If **collision**, retransmit frame in each subsequent slot with probability p, till success
- **Pure (unslotted) ALOHA**
  - No slot, no synchronization
  - **Increased** chance of collision – frame sent at t_0 collides with other frames sent in $(t_0 - 1, t_0 + 1)$
- **Carrier Sense Multiple Access (CSMA)**
  - Sense channel before transmission.
    - If **idle**, transmit frame
    - If **busy**, defer transmission
  - Collisions can still occur due to **propagation delay** (sensing before frame arrives node)
- **CSMA with Collision Detection (CSMA/CD)**
  - CSMA, but when collision is detected, transmission aborted
  - Retransmit after a **random amount** of time
  - **NOTE:** Collisions may not be detected if frame size is too small!
  - See CSMA/CD with binary backoff (Ethernet)
- **CSMA with Collision Avoidance (CSMA/CA)**
  - Receiver needs to transmit ACK if a frame is received OK

Switched Local Area Network

Link Layer Addressing & ARP

- **MAC address** – 48 bits permanent (burned in ROM) address for every adapter (NIC)
- **Example** – `12-34-56-78-9A-BC` (first three bytes identifies vendor)
- Basic handling of link layer frames by NIC – check if destination MAC address matches (or is broadcast)
  - If **yes**, extract enclosed datagram and passes to protocol stack
  - If **no**, discard frame without interruption
- **Address Resolution Protocol (ARP)** – resolution of Internet layer address to link layer address
- **ARP Table** – lookup table on every IP node that contains mapping of IP <–> MAC address for other nodes in same subnet
- Sending frames to another node – depends if in same subnet
  - If **same subnet**, check if sender's ARP table contains destination MAC address
    - If **yes**, create frame with destination IP and MAC address (as provided in ARP)
    - If **not**, broadcast (destination MAC: `FF-FF-FF-FF-FF-FF`) **ARP query** packet
  - If **different subnet**, transmit frame with IP address of sender and MAC address of router (i.e. send frame to router to be passed on)

Ethernet

- **Local Area Network (LAN)** – computer network connection computers within geographical areas
- **Various LAN technologies**
  - **IBM Token Ring** (802.5)
  - **Ethernet** (802.3)
  - **Wi-Fi** (802.11)
- **Physical Topologies**
  - **Bus** – all nodes can collide with each other
  - **Star** – nodes do not collide
- **Ethernet Frame Structure** – NIC encapsulates IP datagram in Ethernet frame (minimum 64 bytes)

8 bytes	6	6	2	46 – 1500	4
Preamble	Dest Addr	Src Addr	Type	Payload	CRC

  - **Preamble** – 7 bytes of `10101010` followed by `10101011`. Used to sync receiver & sender clock rates
  - **Source & Dest MAC** – to determine if NIC should discard or forward frame to network layer protocols.
  - **Type** – indicate higher layer protocol (mostly IP)
  - **CRC** – drop if frame is corrupted
- **Properties of Ethernet Delivery Service**
  - **Connectionless** – no handshakes
  - **Unreliable** – no ACK/NAK sent (need higher layer rdt, e.g. TCP)
- **Ethernet Multiple Access Protocol** – CSMA/CD with binary (exponential) backoff
- **Ethernet CSMA/CD Algorithm** – backoff adapts retransmission to estimated channel load
  1. NIC receives datagram from network layer, creates frame.
  2. If NIC senses **channel idle**, starts frame transmission. If NIC senses **channel busy**, waits until channel idle, then transmits.
  3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
  4. If NIC **detects** another transmission while transmitting, **aborts** and sends **jam signal**.
  5. After aborting, NIC **enters binary back-off:**
     - after m^{th} collision, NIC chooses random K from $\{0, 1, 2, ..., 2^m - 1\}$
     - NIC waits $K \times 512$ bit times, return to Step 2

Link-layer Switches

- **Ethernet Switch** – store/forward Ethernet frames using MAC address
  - **Transparent** to hosts – no need for IP addresses
- Switch in **star topology**, dedicated connection to node, thus, full duplex
- CSMA/CD protocol used, with no collision
- **Switch Forwarding Table** – maps MAC <–> Physical Interface (port)
- **Switch Table Format**
  <MAC address, interface, TTL>
- **Switch Forwarding Algorithm**
  - When frame is received, note down sender's location in table (if not yet inside)
  - If destination is in switch table, forward to interface
  - If unknown destination, broadcast frame to **all outgoing links**

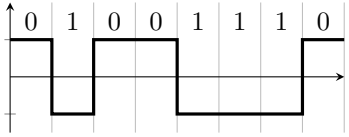
CS2105: Physical Layer

Signal Basics

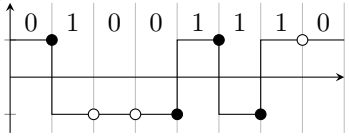
- **Analog signal** – continuous levels, no limit
- **Digital signal** – limited number of defined values (discrete)

Digital Transmission

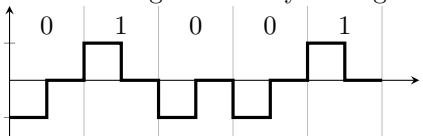
- **Non-Return-to-Zero (NRZ)** – uses two voltage level (non-zero)
  - **Level (NRZ-L)** – voltage level determines the bit (no coding)



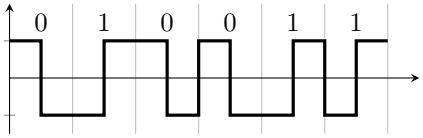
  - **Inverted (NRZ-I)** – inverts voltage level if bit is 1



- **Return-to-Zero (RZ)** – **three** voltage levels, returns to voltage 0 halfway through a bit interval



- **Manchester** – inversion of signal in middle of bit
  - **Pos to Neg (Downwards)** – represents bit 0 (ZERO)
  - **Neg to Pos (Upwards)** – represents bit 1 (ONE)

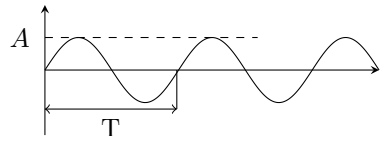


Analog Transmission

- Basic analog signal – **sine wave**
- Equation of **sine wave**:

$$f(t) = A\sin(2\pi ft + \phi)$$

where ϕ is phase and f is the frequency (Hz)

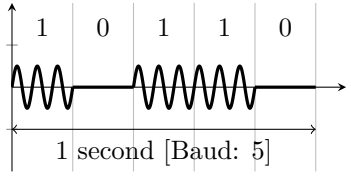


- **Bandwidth** – difference between the upper and lower frequencies
- **Channel Bandwidth** – represents the range of frequency that can pass through a channel
- **Signal to noise ratio (SNR)** – measures the strength of signal over noise
- **Shannon Capacity** – theoretical maximum bitrate of a noisy channel

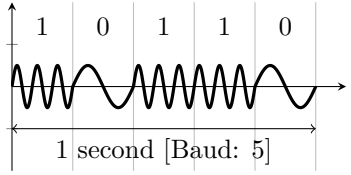
$$C = B * \log_2(1 + SNR)$$

B is the channel bandwidth
SNR is the signal to noise ratio of channel

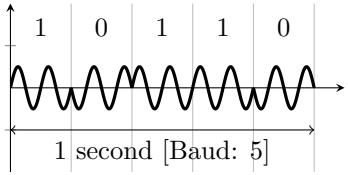
- A, f & ϕ in a sine wave can be varied
- **Amplitude Shift Keying (ASK)** – varying **peak amplitude** (A) to represent signal element (bits) [Susceptible to noise]



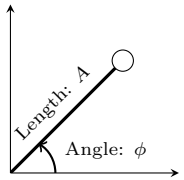
- **Frequency Shift Keying (FSK)** – vary **frequency** (f) [Limited by channel bandwidth]



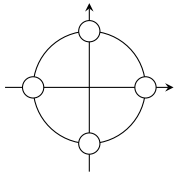
- **Phase Shift Keying (PSK)** – vary **phase** (ϕ)



- **Constellation Diagram** – representation of signal modulated by a digital modulation scheme



- **Quadrature Phase Shift Keying (QPSK)** – send signals with 4 phases (2 bits of data per segment)



- **8-PSK** – eight phases, carrying 3 bits of data!
- **Quadrature Amplitude Modulation (QAM)** – combines ASK and PSK
- **Baud rate** – signal units per second
- **Bit rate** – bits receiver receives per second
- n-[ASK/FSK/PSK/QAM] has n distinct signal elements that can carry $\log_2 n$ bits of data

CS2105: Network Security

Aspects of Security

- **Message confidentiality** – only the sender is allowed to view message
- **Message integrity** – message is not modified in transit
- **Message authenticity** – message is from the sender
- **Service availability** – link between sender and receiver is available

Terminology

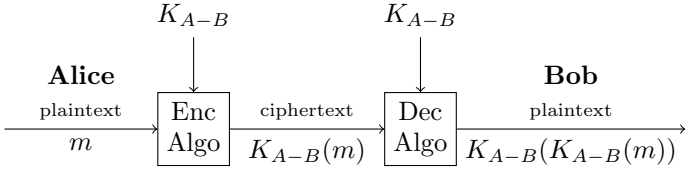
- m – plaintext message
- K_A – encryption/decryption key A

Message Confidentiality

Cryptography Systems

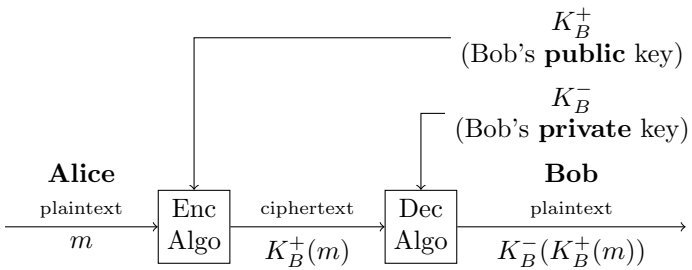
- **Symmetric key cryptography** – only ONE key (regardless of number of entities)
- **Public key cryptography** – uses a pair of keys per entity

Symmetric Key Cryptography



- Alice & Bob share & use same (symmetric) key – K_{A-B}
- **Examples**
  - DES (Data Encryption Standard)
  - AES (Advanced Encryption Standard)
  - Mono-alphabetic cipher (Caesar cipher)
- **Issues**
  - Requires both parties to **share** secret
  - Same key is used for both **encryption** & **decryption**

Public Key Cryptography



- Each party has a **pair of keys**
- **Public (encryption) key** – known to world
- **Private (decryption) key** – receiver only knows
- **Must** satisfy following property $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$
- **Example** – RSA

Utilizing Both

- **DES** (symmetric) is faster than **RSA** (public)
- Utilize RSA to establish connection and then, use symmetric key to encrypt data to be transmitted
- **Session key** (K_S) – key that is valid for one session
  - Alice & Bob use RSA to exchange symmetric key K_S for session
  - Utilize symmetric key crypto with K_S to communicate

Message Integrity & Authenticity

Cryptographic Hash Functions

- **Hash function** ($H(m)$) – given m, compute **fixed size** string known as **message digest/hash/fingerprint**
- **Properties**
  - **One-way function** – infeasible to find two messages with same hash
  - **Avalanche effect** – small change in message results in significant change in hash
- **Popular Examples** – MD5 (Message Digest) & SHA-1 (Secure Hash Algorithm)

Message Authentication Code (MAC)

- Affirms **both** integrity and authenticity
- **Basic idea** – hash concatenation of m and s (shared secret, **authentication key**)
- **Caveat** – unable to prove which party crafted message, as secret is shared!

Digital Signature

- Use public key cryptography to sign hash
- Ensures **verifiability** & **non-repudiation**
- **Cons** – speed

