Data-Link Layer

- Node-to-node communication
  - end systems and packet switches all consists of a [DL]

- Responsible in delivering datagrams to the next node in the path.
- Services
  - Framing: Encapsulate (@ source) and decapsulate (@ Dest) datagrams in Frames.
    Frames can Have a Header and a trail.
  - Media Access Control: For shared links, provide a link share mechsm.
  - Error Control: Detect and Some instances Correct bit errors
    - Hardware-based fast error detection

- Data link Control (DLC) : Handle issues common to all links.

- Framing happens @ each intermediate nodes.
  - Different links could using different framing formats
  - Link Layer addresses change per node.

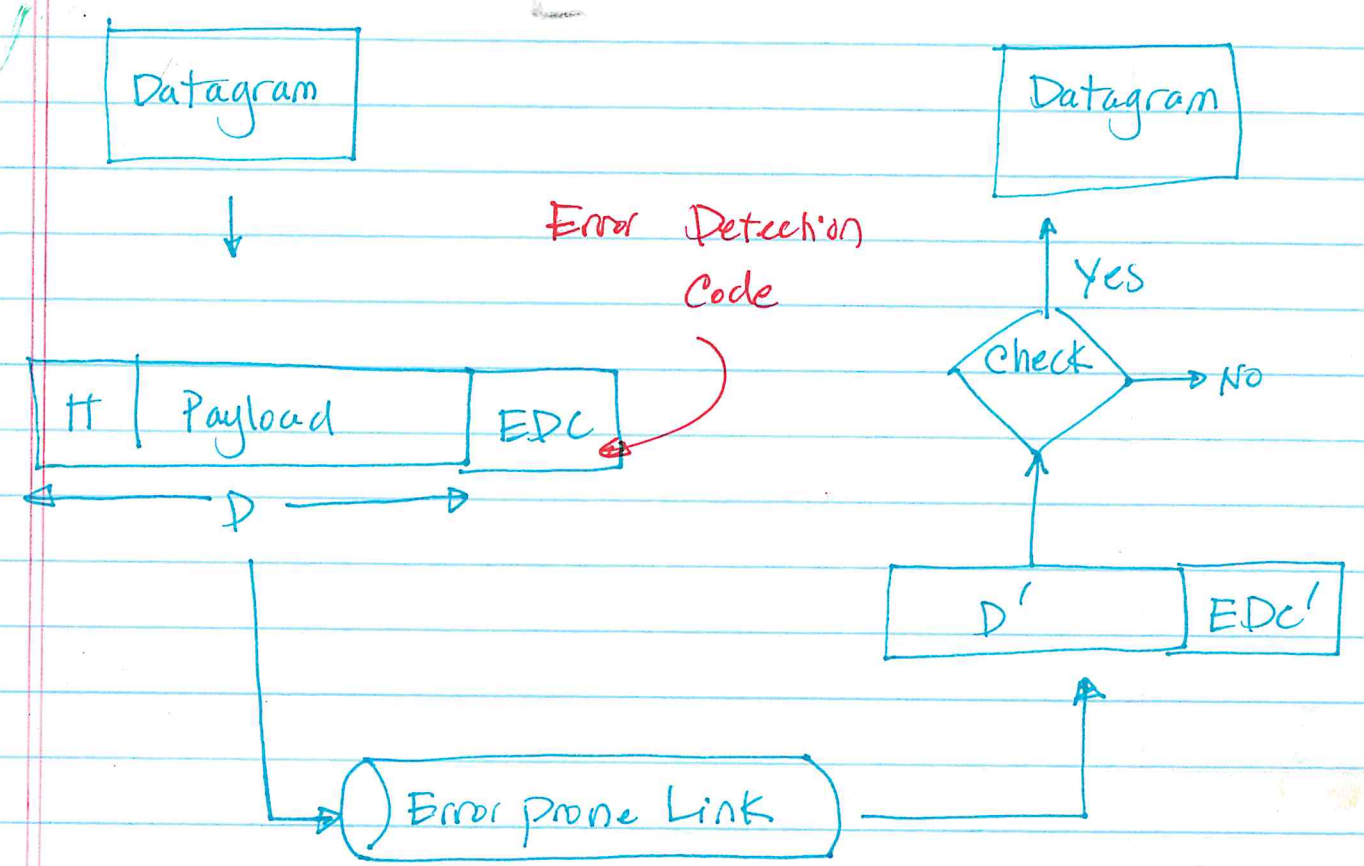
Data links can be of two types:

- Point-to-Point links : One sender and one receiver exclusively.
  - No Media Access Control.
- Broadcast links : More than one user pair sharing a link

[cell phone (thro air) vs. land phone analogy.]

many Seadecs.
many Receivers : Security Trick := Catch "packets" ~~Not~~ not bound to self.

Error Detection and Correction



Parity check: A parity bit (s) used to indicate bit errors.

parity

Even parity (total # of 1's in packet is even)

Odd parity (total # 1's in packet is odd)

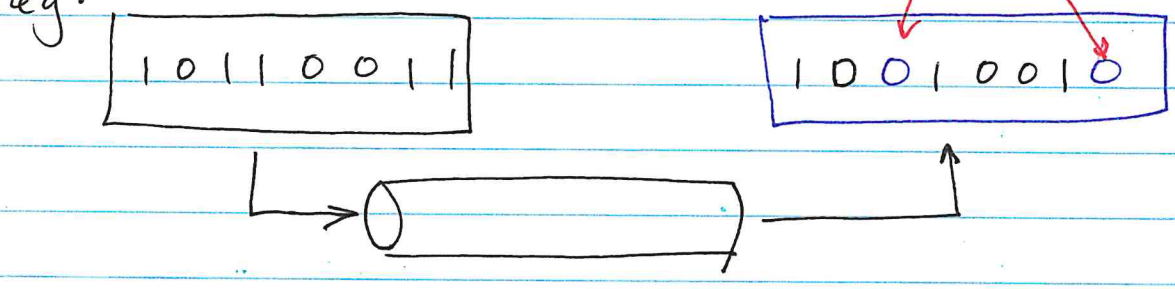
E.g. | 1 | 0 | 1 | 1 | 0 | 0 | 1 | |

odd parity [1]
even parity [0]

A single parity can only detect odd # errors

will go undetected.

eg.

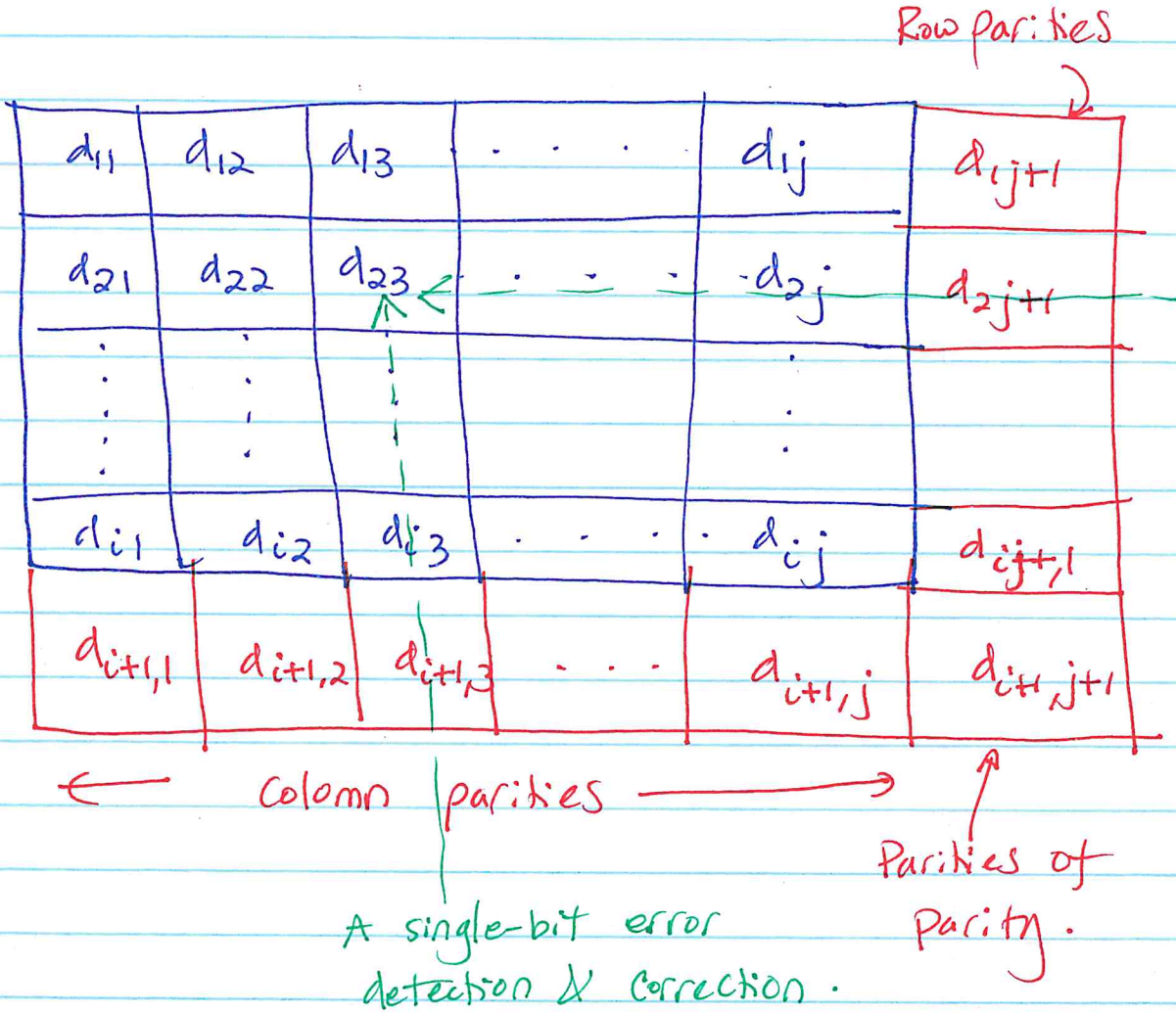


Two-Dimentional parity

- A row parity, a column parity and a parity of parity
- Can detect and correct single-bit errors w/o re-transmissions
  - Forward Error Correction

→ thecsemonk.com/two-dimentional-parity.

- can detect 2-bit errors but cannot make corrections. - Not all the time though.
  - Even # of bits can be only detected.
- Some error combinations still can go undetected.



- By extending the dimentions higher order bit errors can be corrected
  - But too cumbersome and costly.

Why 3 levels of Error checking?

- Transport Layer : Checksum
- Network Layer : Internet Checksum
- Link Layer : EDC

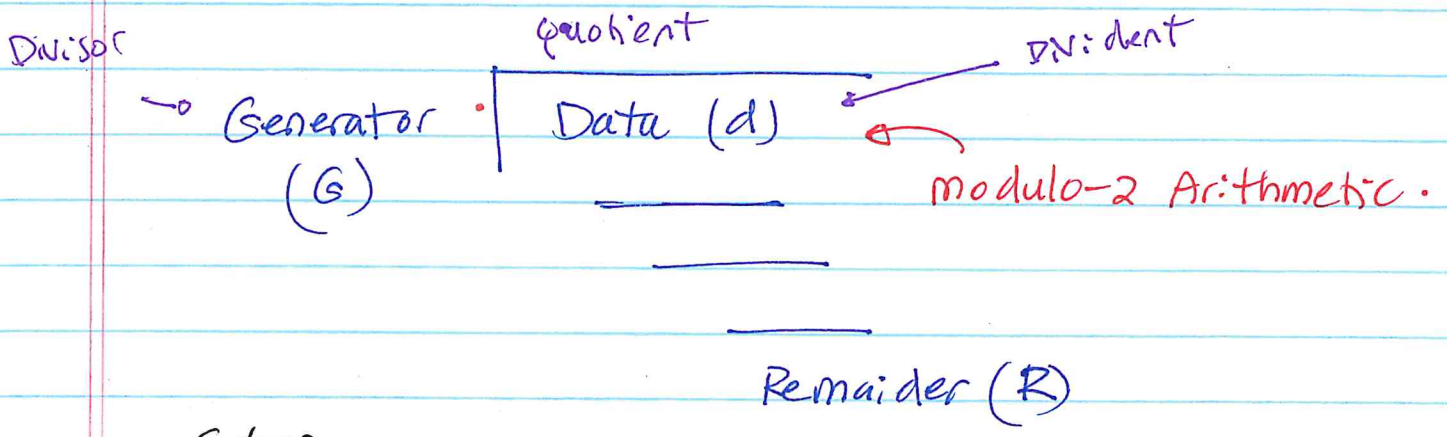
TL — software-based; simple, less complex methods for overall efficiency.

NL — only for headers; not worried about errors in data

DL — Hardware-based; Complex algorithms can be implemented efficiently through hardware.

Cyclic Redundacy check (CRC)

- A comprehensive detection Algorithm.



Setup

Data :	d-bits
Generator :	(r+1) bits
Remainder :	r-bits

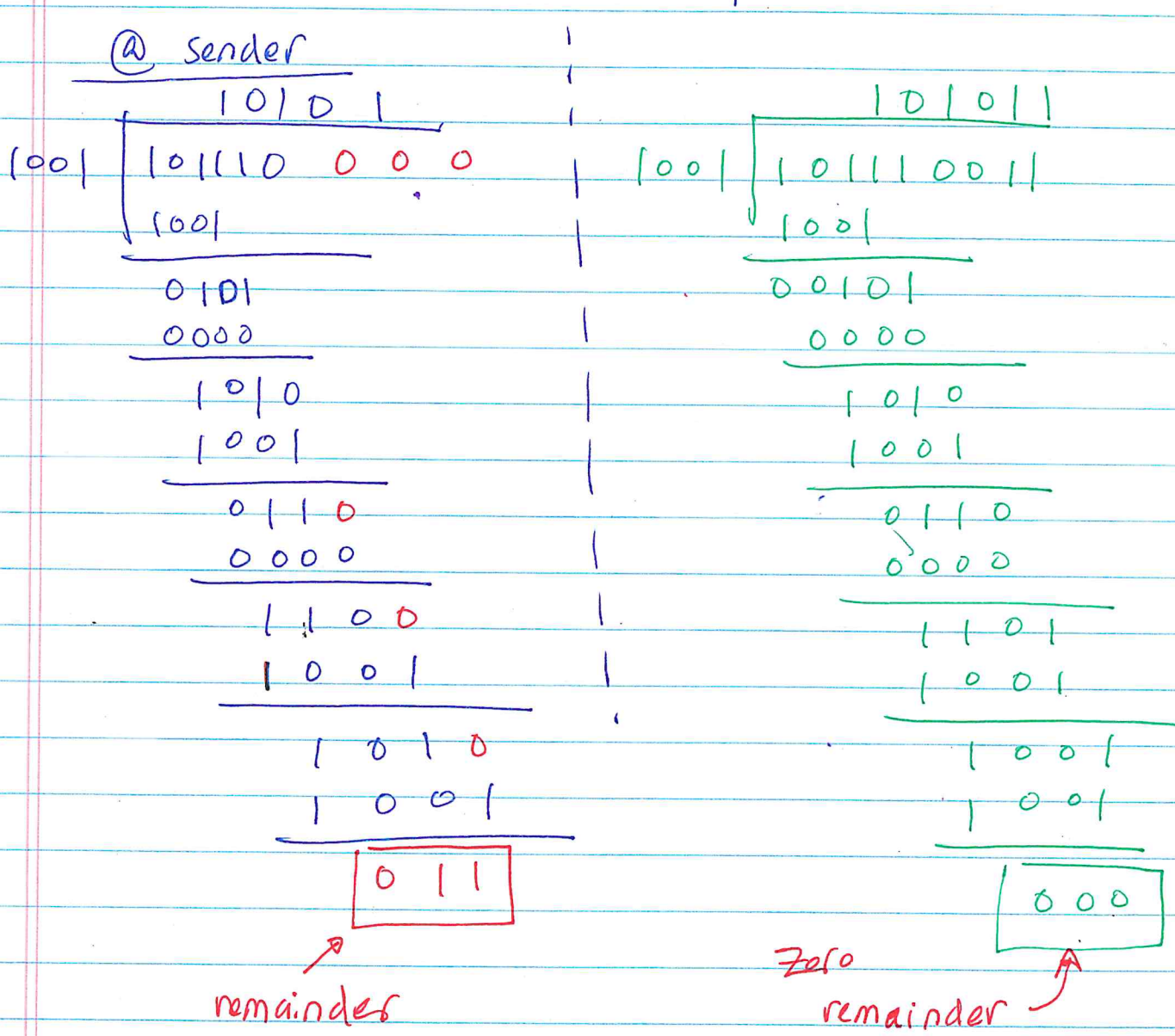
Thus per frame a r-bit overhead.

Detection @ receiver.

$$\frac{(D+R)}{G} = \text{Zero remainder}$$

(if No errors)

E.g. D = 10110 / G = 1001 / R = 3-bits.



Hamming Distance

- The difference between two equally sized (binary) strings.

e.g. 01101110 vs
11001011
---
x - x - - x - x

↳ Hamming Distance = 4.

It is possible to measure / estimate the maximum probability error on a noisy channel.

↳ The theorem is called Shannon's Noisy-Channel Coding theorem.

How is error probability related to hamming distance?

- Suppose a particular channel has 2-bit error probability.

↳ what this means is at max the sent message could be ~~of~~ two bits different from the received message

↳ Hamming Distance = 2.

- If all valid messages has a min. hamming distance greater than error probability, all error can be detected (and some even corrected).

Theory (Derivation beyond the scope of the course).

Let h define the min. Hamming Distance.

If so

- Then, $(h-1)$ errors can be detected

- $\frac{(h-1)}{2}$ errors can be corrected.

example.

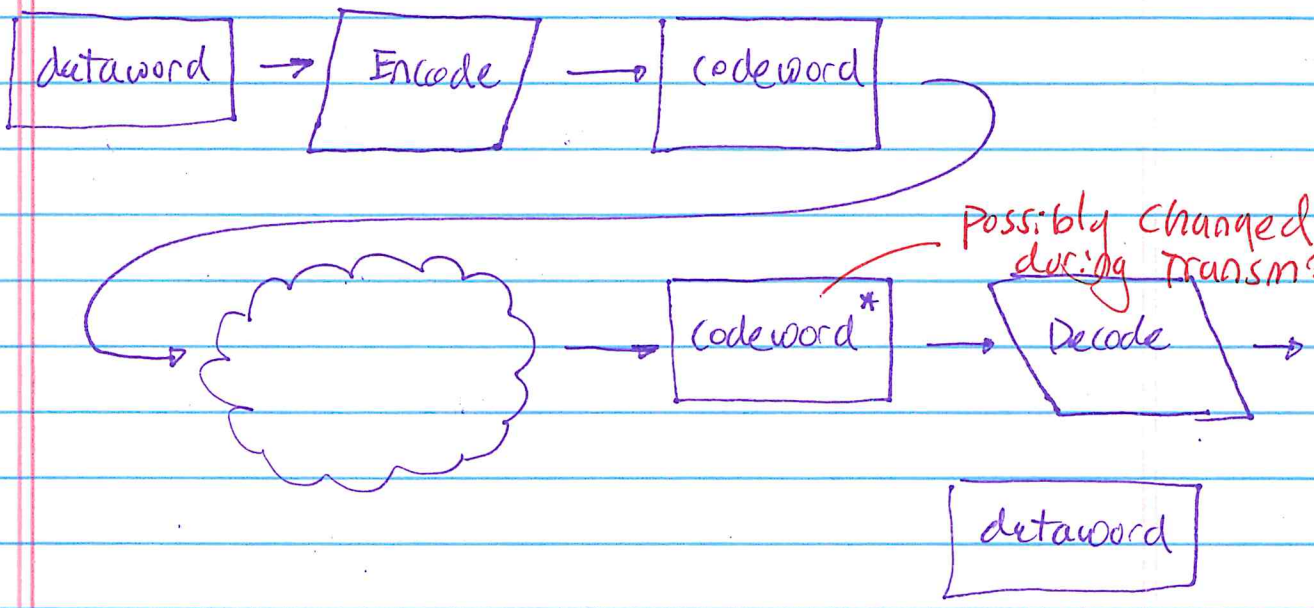
suppose two datawords (messages) 000 and 111.

→ min Hamming distance = 3.

Thus, 2 errors can be detected
1 error can be corrected.

Idea:

Change datawords (messages) into codewords with a min. Hamming distance.



- BCH-code
- Nordstrom-Robinson code

Self Research (11/14/22)

Simple Example:

- suppose we are interested in transmitting a single-bit message $m = \{0, 1\}$
  - consider a channel that is 2-bit error prone.
    - ↳ min. Hamming distance needs to be 3.

Encoding:

$0 \rightarrow 000$

$1 \rightarrow 111$

consider all possible decodings.

(000) ⟶ 0

All detectable because non are valid codewords:

001

010

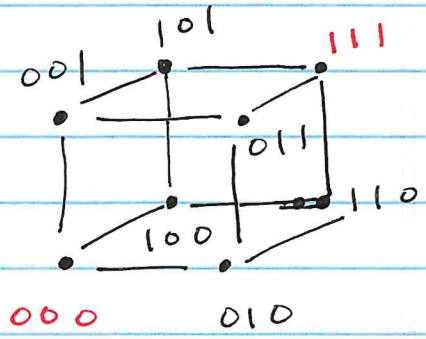
011

100

101

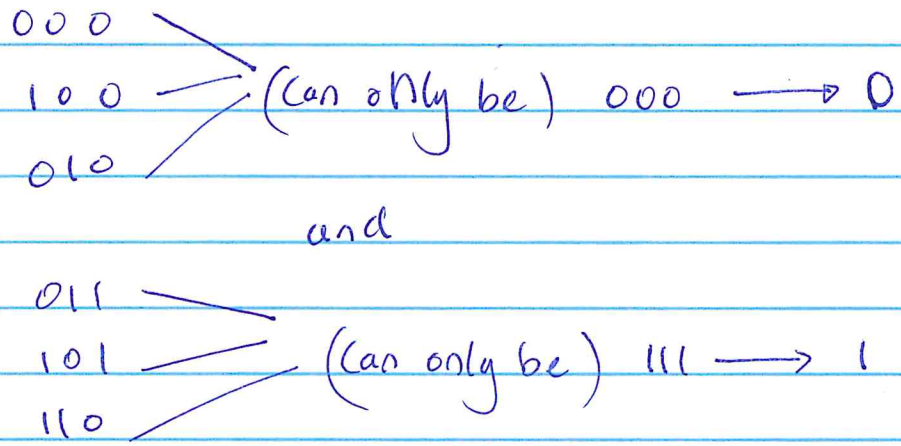
110

(111) ⟶ 1



For the same example, what if the max. error probability was just 1 bit?

Now note:



Meaning, a single error can be corrected.

↳ This is called Forward Error Correction (FEC).

However, it's not possible to distinguish between if a single-bit or double-bit errors happened.

Note: The overhead (as in the additional bits needed) becomes quite high if the channel is very noisy.

Need additional bit(s).

† Read about BCH-code (Bose-Chaudri-Hocquenghem)

Practical use of Hamming Distance for Error detection and Correction

Hamming Codes: → min. Hamming distance 3.

- Can correct a single-bit error and detect two bit errors.

- Notation Hamming (7, 4) — 4: Data bits; 7: total bits

e.g. Hamming (3,1)
Hamming (7,4)
Hamming (15,11)
⋮
Hamming (255, 247).

- all have a min. Hamming distance of 3.

very commonly used in RAM memory forward error correction.

Formula for Parity bits (P) (redundant bits).

$2^P \geq d + P + 1$

d: data bits.

Example. Assume we want to transmit 8-bits (A byte of data).

$2^p \geq 8 + p + 1$ which means

$p \geq 4$. → minimum transmitted bits = 12.

Algorithm.

(i) All bit locations that are powers of 2 become parity bit locations.

(ii) The data word is used to fill other bit locations.

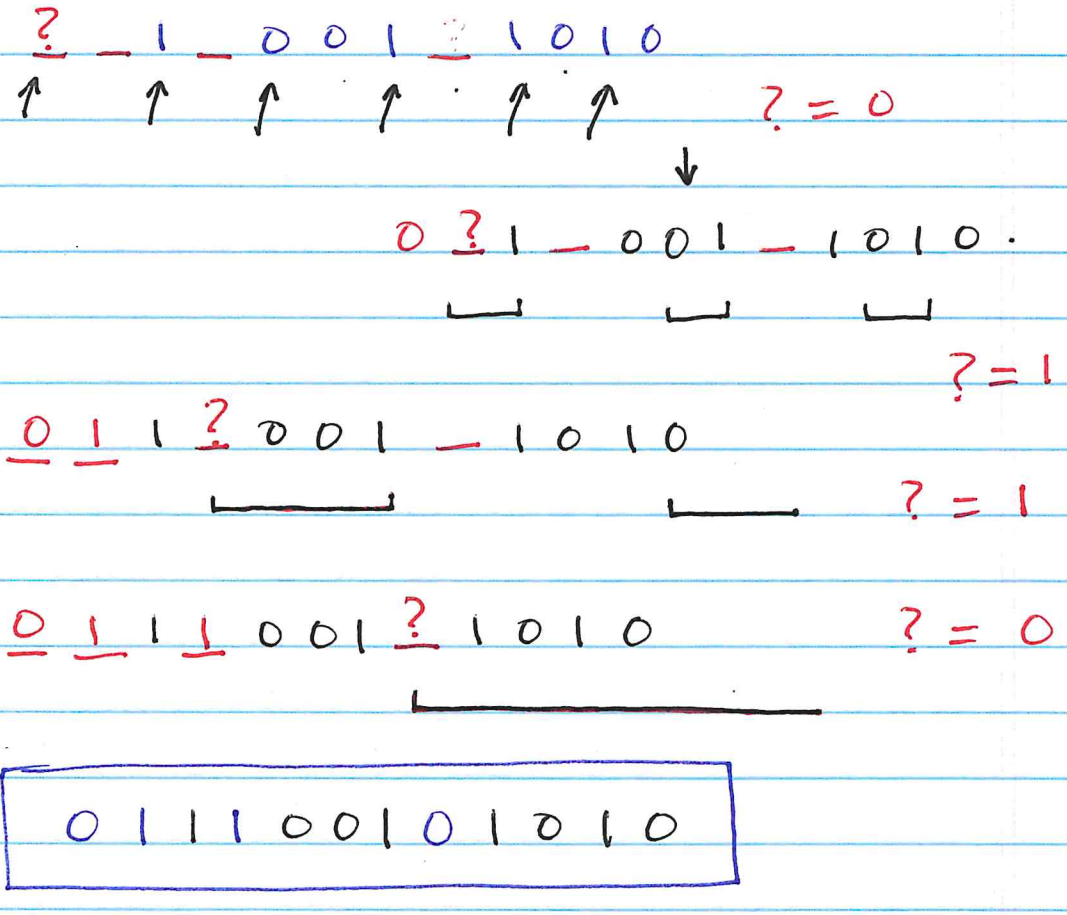
example: A byte of data 10011010

? ? 1 ? 0 0 1 ? 1 0 1 0

Parity bits

- parity bit location 1 is calculated by skipping every other bit.
- parity bit location 2 is calculated by skipping every other 2 bits

- parity bit location 4 is calculated by skipping every other 4 bits.
- parity bit location 8 is calculated by skipping every other 8 bits.



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

↳ Code word.

Suppose during transmission 011100101010 became 011100101110. How can the receiver forward error correct this?

- The method is to check every parity bit.

(i) 0 1 1 1 0 0 1 0 1 1 1 0 — parity bit 1 ✓.

(ii) 0 1 1 1 0 0 1 0 1 1 1 0 — parity bit X.

(iii) 0 1 1 1 0 0 1 0 1 1 1 0 — parity bit ✓

(iv) 0 1 1 1 0 0 1 0 1 1 1 0 — parity bit X.

parity bits 2 and 8 are erroneous.

$2 + 8 = 10 \rightarrow 10^{th}$ bit location needs corrected.

Calculating the Hamming Code

In practice, used to correct a single bit error and detect two bit errors.

The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

1. Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)
2. All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.
   Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
   Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
   Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
   Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)
   Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)
   Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...)
   etc.
4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Here is an example:

A byte of data: 10011010
Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0
Calculate the parity for each parity bit (a ? represents the bit position being set):

- Position 1 checks bits 1,3,5,7,9,11:
  ? _ 1 _ 0 0 1 _ 1 0 1 0. Even parity so set position 1 to a 0: 0 _ 1 _ 0 0 1 _ 1 0 1 0
- Position 2 checks bits 2,3,6,7,10,11:
  0 ? 1 _ 0 0 1 _ 1 0 1 0. Odd parity so set position 2 to a 1: 0 1 1 _ 0 0 1 _ 1 0 1 0
- Position 4 checks bits 4,5,6,7,12:
  0 1 1 ? 0 0 1 _ 1 0 1 0. Odd parity so set position 4 to a 1: 0 1 1 1 0 0 1 _ 1 0 1 0
- Position 8 checks bits 8,9,10,11,12:
  0 1 1 1 0 0 1 ? 1 0 1 0. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 0 1 0 1 0
- Code word: 011100101010.

Finding and fixing a bad bit

The above example created a code word of 011100101010. Suppose the word that was received was 011100101110 instead. Then the receiver could calculate which bit was wrong and correct it. The method is to verify each check bit. Write down all the incorrect parity bits. Doing so, you will discover that parity bits 2 and 8 are incorrect. It is not an accident that 2 + 8 = 10, and that bit position 10 is the location of the bad bit. In general, check each parity bit, and add the positions that are wrong, this will give you the location of the bad bit.

Try one yourself

Test if these code words are correct, assuming they were created using an even parity Hamming Code . If one is incorrect, indicate what the correct code word should have been. Also, indicate what the original data was.

- 010101100011
- 111110001100
- 000010001010

Linear Error-Correcting Codes

Lecturer: Michel Goemans

1 Linear Error Correcting Codes

We wish to transmit a message (represented in bits) to a destination through a channel reliably. Errors may occur during transmission and we hope to detect and correct these transmitted errors. In order to do so, we must introduce some redundancy in our transmitted message.

We saw that Shannon's Noisy Coding Theorem tells us that if the redundancy is above a certain level (constrained by the noise of the channel), then we can drive the probability of error (detecting the wrong message sent) to zero with increasing blocklength. The proof was based on a random code which is not very practical. For easy encoding and decoding, we need structure in our encoding/decoding functions. We will study a class of codes called *linear block codes*, because their structure offers several advantages. Linearity will allow an easier analysis of the error correcting ability of a code. Furthermore, the use of matrices to encode/decode messages is much easier than a random codebook, and the code can be concisely described.

Error detection means that the code can detect errors but we don't know where the errors are in a received sequence. *Error correction* means we know where the errors are and can correct the bit positions in error. If a code can correct up to t errors, then if the sequence contains more than t errors, the decoder may decode to the wrong information sequence.

All our messages, codewords, and received messages will be sequences with binary coefficients from the binary field. Let e_i be the vector with 1 in the i-th position and zero in all the other positions, so $e_3 = (00100\ldots00)$. For linear codes, encoding is a linear transformation c that maps a message m to a codeword $c(m)$. Block codes means all codewords $c(m)$ have the same length, which we denote as n. Let k be the length of information bits that we encode. Then there are $n-k$ redundancy bits in each codeword, called *parity check bits*, which are left for error correction. The (n, k) specify the parameters (codeword length and parity check bit length) for a block code. Let $\tilde{c}$ be the received vector or string of n bits.

The *Hamming distance* d_H between any two bit strings is the number of positions in which the 2 strings differ. For example, the Hamming distance d_H between the codewords $c_1 = (101101)$ and $c_2 = (100110)$ is 3. Denote d^* to be the minimum Hamming distance between any two distinct codewords of a code C as

$$d^* = d_{\min} = \min_{c_i \neq c_j} d_H(c_i, c_j). \tag{1}$$

A code with minimum distance d^* between codewords can detect $d^* - 1$ errors and can correct $\frac{d^*-1}{2}$ errors with nearest neighbor decoding. Thus, to correct one error, the minimum distance of the code must be at least 3.

A *linear* code means that the encoding function c is linear (over $\mathbb{Z}_2$). If messages m and m' have respective codewords $c(m)$ and $c(m')$, then $c(m + m') = c(m) + c(m')$ is the codeword for the

The generator matrix of the Hamming code has dimension $k \times n$ and is of the form

$$G = [I_{k\times k}\ S]\,.$$

For this Hamming code, S has one row for each possible m-bit string with weight at least 2. There are exactly $k = 2^m - m - 1$ such strings. Any Hamming code is a 1-error correcting code as the two conditions above are satisfied.

Example. The $(7, 4)$ Hamming code is given by the generating matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

We know how to encode with any linear code. The codeword generated by message m is simply $c = mG$. But how about decoding? If no errors were introduced, the first k bits constitute the message. But errors might have been introduced. When there is only one error, decoding is easy as we show now.

For every generator matrix $G = [I\ S]$, there is a parity-check matrix H defined by

$$H = \begin{bmatrix} S \\ I \end{bmatrix}.$$

Since S has size $k \times (n-k)$, we have that the identity matrix is of size $(n-k) \times (n-k)$, and H has size $n \times (n-k)$. The important observation is that $GH = S + S = 0$, and therefore $c = mG$ satisfies that $cH = mGH = 0$. This means that if there an error and we have received $\tilde{c} = c + e_i$, we can find the error e_i by computing the *syndrome* $\tilde{c}H$ as this is supposed to be $cH + e_iH = e_iH$. Thus, $\tilde{c}H$ must be one of the rows of H and the index of the row indicates which bit has been corrupted. Once we know which bit was corrupted, we can recover the message.

Example: A $(7, 4)$ Hamming Code with the following generator matrix (the last 3 columns of each row must be at least 2 and no two are identical)

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Suppose we have the received string $\tilde{c} = (1110111)$. Then the syndrome is

$$\tilde{c}H = (111)$$

Address Resolution Protocol (ARP).

- layer-2 devices do not speak IP.
  - Speak MAC (Media Access Control)
  - 6 bytes → 2^{48} address space.
- Must convert IP ⟶ MAC.
  - ARP
    - An IP layer Auxillary protocol.

Note: layer-2 devices switch frames; Do not employ RIP or OSPF as they do not route.
- layer-2 device = switch.

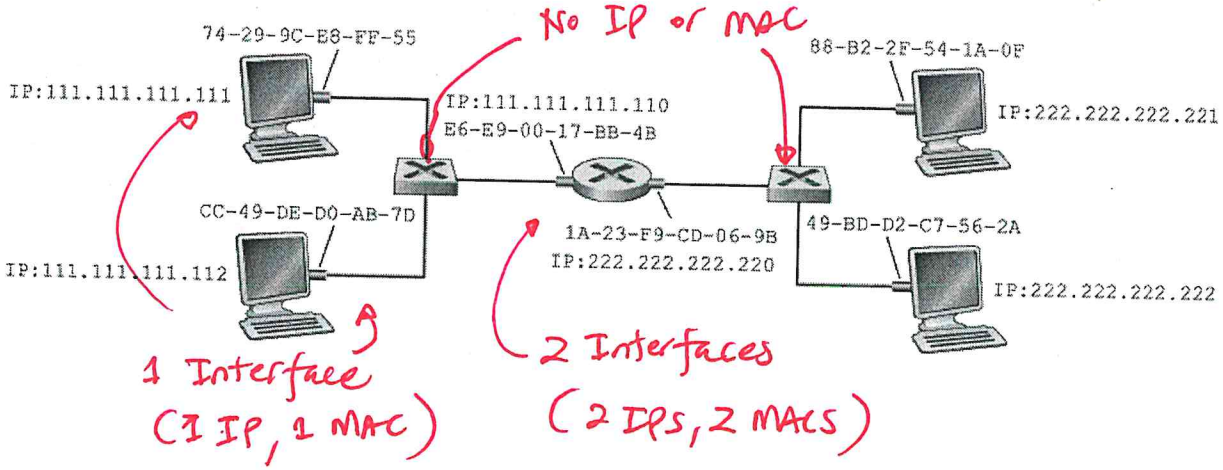
- MAC Addess is a Hardware Address.

> Analogy:
> MAC ≡ SSN number
> IP ≡ postal Address / Drivers licence
> Change state (network) new IP but not a new MAC.

- MAC addresses are truly unique.
- Companies get a chunk of address space (2^{24}) from IEEE.
- MAC broadcast : FF-FF-FF-FF-FF-FF

Important: Switches (layer-2 packet switches) have NO MAC addresses OR IP addresses.

↳ Not directly addressible



abovementioned.

- Routers and End systems keep a ARP table → e.g. on linux arp -n.

IP address	MAC address	TTL
X.X.X.X	A-A-A-A-A-A	YY:YY

only a limited # of entries.
each will expire after TTL
(typically ~20 mins after placing)

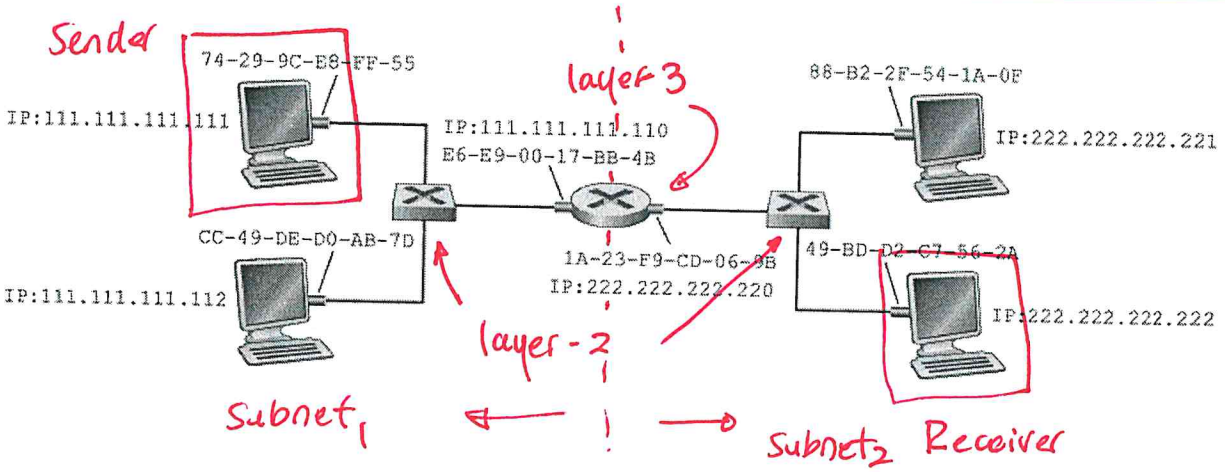
Important:

- ARP is similar in function to DNS but scoped to only within subnets

Note: Routers (layer-3) and End systems only know about neighboring (within subnet) MAC addresses.

↳ Security check: leak MACs outside subnets.

How to find MAC address? (If not in ARP table).



- Sender knows Receiver's IP → Need MAC.
  - Check local ARP table (No hit)
  - Broadcast a ARP Query
    - ~~Anamashan?~~ looking for who?
    - ** First hop router (111.111.111.110)
    - Get MAC using ARP

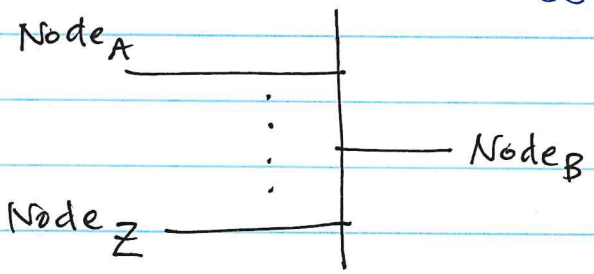
- Create a frame with Dest. MAC: Router and Dest. IP: Receiver.

- First hop router sees the frame is addressed to him
  - Accept frame → decapsulate → Pass to NL
  - Get Dest. Address → Consult Forwarding table → Determine forwarding interface (222.222.222.220 interface)
  - Need dest MAC (for framing)
    - broadcast ARP query looking for 222.222.222.222.
    - New frame. Dest. MAC: dest. Dest. IP: Dest.

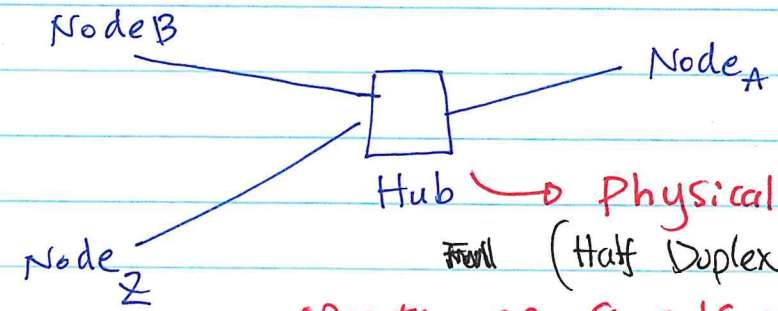
Ethernet

- Main & wired LAN technology
  - Alternatives: ring, FDDI, ATM
    - (Fiber Distibuted Data interface)
  - cheap, inexpensive, early adoptation (hence admins reluctant to change).

- Original Ethernet (early '70s) by Bob Metcalfe and David Boggs was a bus.



- Late '90's Bus → Star topology

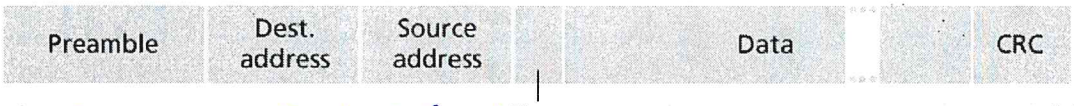


Hub → Physical layer device.
(Half Duplex Links)
operate on signals; receive → boost signal → broadcast on all interfaces.

- Hubs operate on bits (signals) NOT frames.
  - Duplicates every bit on every other interface
    - Collisions are plenty.

- Early 2000's : Hub → switch.
  - Collision-less
  - Store-and-Forward device @ L2
  - Acts on Frames.
  - Full Duplex.

Ethernet Frame



← 8 bytes → ← 6 → ← 6 → ← 2 → ← 46 - 1500 bytes (MTU) → ← 4 →

Preamble : First 7 bytes 10101010

last byte 10101011

synchronization → Get ready for important data/info.

- Type → Type of payload
  - e.g. ARP, IP, Novel IPX, AppleTalk.

- A frame can carry different types of payloads (NOT just IP)

- Current Ethernet standard (100 Mbps)
  - 100 Mbps ~~switched~~ twisted pair upto 100 meters and fiber optics for longer.

- Gigabit Ethernet : 40,000 Mbps.

Link Layer Switching

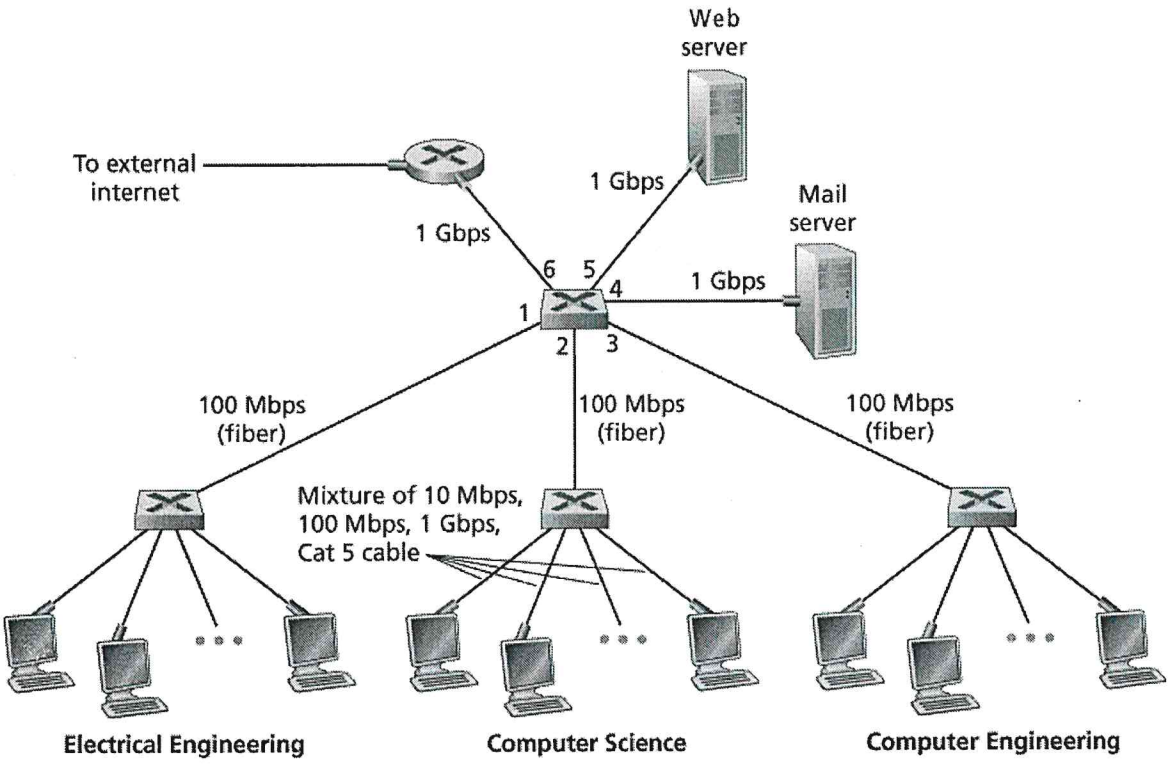
- Switches are transparent to hosts/nodes.
  - addressing per host-to-host basis
    NOT Host → switch → Host basis.

- Can Include buffers (to throttle) for flow control.
  - outgoing link rate could be smaller than incoming
- A store-and-forward device.

- Two main functions
  - Switching (Layer-2 Forwarding)
    - Match the outgoing interface using a switch table (self-learning)
  - Filtering
    - Decide whether or not to switch the frame
      - e.g. CRC failure.

- Three types of forwarding decisions
  - No entry for Dest. MAC in switch table
    - Broadcast on all interfaces
  - Dest. MAC is in the incoming interface
    - Drop Frame
  - Entry found in table - Forward (switch)

Example Switched LAN



Example switch table.

Address	Interface	Time
XX-XX-XX-XX-XX-XX	Y	ZZ:ZZ.

time @ which entry was placed.

- Fun^{tns} Similar to a forwarding table; looks similar to a ARP table

- Switches self-learn; no explicit configur^{tn}.
  - Initial switch table is empty.
    - Upon arrival of a new frame place sender's MAC & incoming interface on table
  - Prune "old entries" after a aging time

- Switches are plug-and-play.

- No collision; because of store-and-forward.
  only process and switch one frame @ a time

- Can Interconnect Hetegenous Links that operate @ different link rates.

- Enhanced Security as host will only receive frames explicitly addressed and broadcast frames.
  - Packet sniffing between two hosts seperted by a switch is difficult (not impossible)

- Provides fault isolation; can detect and isolate malfunctioning adapters (jabbering adapter) w/o human intervention.

Switch vs Router

Epic showdown ☺

Switch

- layer-2
- operate on Frames
- packet: Frame

Pros

- self learning
- Plug-and-play
- Relatively high filtering and forwarding rates
- collision-Free

Cons

- Active switched network scoped to a spanning tree → to prevent broadcast cycles.
- large switched network require large ARP tables
- Broadcast storms
  - malfunctioning node spamming network with broadcast traffic

Router

- layer-3
- operate on Datagrams
- packet: Datagram

Pros

- Intelligent → Can use the best path between source → destination
- Not spanning tree restricted
- multiple active paths possible between two hosts
- Firewall protection against layer-2 broadcast storms

Cons

- Not plug-and-play
- expensive
- larger per packet processing time.

Store-and-Forward

	Hub	Switch	Router.
Traffic isolation	No	yes	yes
Plug-and-play	yes	yes	No
optimal Routing	No	No	yes.

switch poisoning

- Send bogus frames to switch with bogus source MACs.
- Fill switch table with bogus entries
  - This causes switch to broadcast all received frames → allows sniffing on switched networks.

- switch table prevents broadcasts (except ARP requests, DHCP), but a switch Poisoning attack can exploit a switch.

Network Design Challenge: Multiple Access Problem

↳ Coordinate the Access of multiple sending and receiving nodes to a shared broadcast channel.

Human analogy: A cocktail party: many people gather in a large room to talk and listen.
e.g. classrooms.

→ Need protocols to dictate.
i. who gets to talk
ii. How to get permission to talk.
iii. How to let multiple people talk @ the same time w/o ~~con~~ corrupting interference

Multiple Access protocols

→ All nodes are capable of transmitting frames. More than one node might transmit @ the same time.

↳ Collision: Data does not make sense.

i. Channel partitioning protocols
ii. Random Access protocols
iii. Taking-turns protocols.

Ideal Model

A broadcast channel of rate R bps.

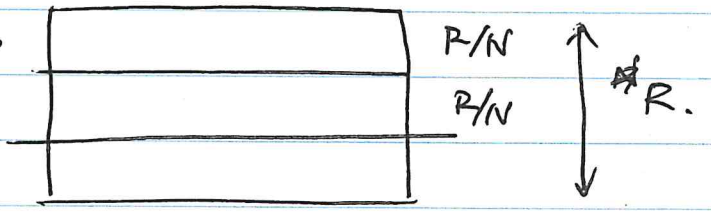
i. when only one node has data to send, that node has a throughput of R bps.

ii. when M nodes have data to send, each node has a throughput of R/M bps.
↳ Average throughput (not necessarily instantaneous).

iii. No master. The protocol is decentralized.

iv. Simple and inexpensive to implement.

FDM ⇒



Channel Partitioning:
- TDM : Time-division Multiplexing.
- FDM : Frequency-division Multiplexing.
- CDMA : Code-division multiple Access.

N nodes / Bandwidth R. (TDM and FDM)

↳ Max throughput R/N : Fair ??
↳ only allowed R/N even when only 1 node is transmitting.
↳ Have to wait to take the turn.

TDM ⇒

→ Code Division Multiple Access.

CDMA — use a code to encode data.

↳ only receivers who knows the sender's code can understand data.

↳ More Prominent in wireless Comm^{n}.

Random Access protocols.

i. Transmit @ full rate.

ii. If Collision retransmit until no Collision.

↳ But NOT imediately

wait a RANDOM delay.

Slotted ALOHA.

i. All frames consists of exactly L bits.

ii. Time slots are L/R : Amount of time to transmit one Frame.

iii. Transmit only @ the beginning of a Slot

iv. All nodes are synced. each node knows when the Slot begins.

v. If Collison all nodes detect it before the Slot ends.

probability p. The event tails corresponds to "skip the slot and toss the coin again in the next slot"; this occurs with probability $(1-p)$. All nodes involved in the collision toss their coins independently.

Slotted ALOHA would appear to have many advantages. Unlike channel partitioning, slotted ALOHA allows a node to transmit continuously at the full rate, R, when that node is the only active node. (A node is said to be active if it has frames to send.) Slotted ALOHA is also highly decentralized, because each node detects collisions and independently decides when to retransmit. (Slotted ALOHA does, however, require the slots to be synchronized in the nodes; shortly we'll discuss an unslotted version of the ALOHA protocol, as well as CSMA protocols, none of which require such synchronization.) Slotted ALOHA is also an extremely simple protocol.

Slotted ALOHA works well when there is only one active node, but how efficient is it when there are multiple active nodes? There are two possible efficiency concerns here. First, as shown in Figure 5.10, when there are multiple active nodes, a certain fraction of the slots will have collisions and will therefore be "wasted." The second concern is that another fraction of the slots will be *empty* because all active nodes refrain from transmitting as a result of the probabilistic transmission policy. The only "unwasted" slots will be those in which exactly one node transmits. A slot in which exactly one node transmits is said to be a **successful slot**. The **efficiency** of a slotted multiple access protocol is defined to be the long-run fraction of successful slots in the case when there are a large number of active nodes, each always having a large number of frames to send.

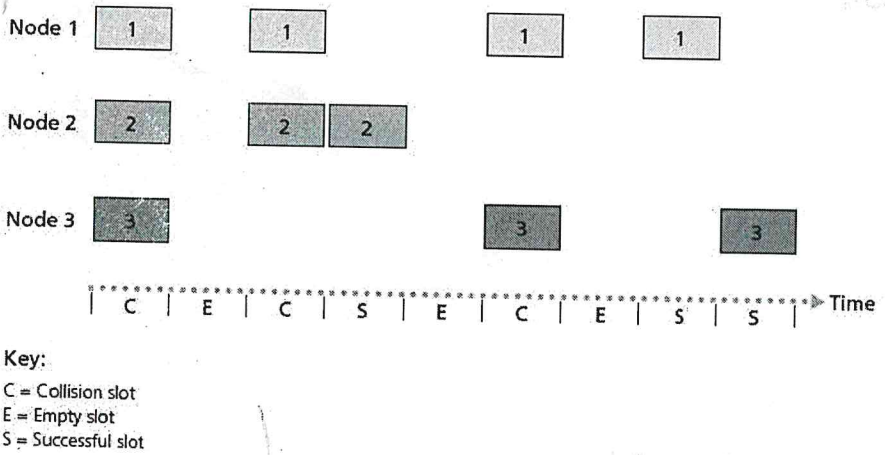


Figure 5.10 ♦ Nodes 1, 2, and 3 collide in the first slot. Node 2 finally succeeds in the fourth slot, node 1 in the eighth slot, and node 3 in the ninth slot

Slotted ALOHA efficiency.

Let p be the probability of a node transmitting after a collision.

$p \in \{0, 1\}$. 0 — No retransmit (skip the slot)
1 — retransmit

when multiple nodes attempt to transmit certain slots gets ~~unused~~ wasted.
- Collision
- After collision ⇒ everyone backs-up.; slot goes unused./empty.

what is the long-run efficiency?

assume N nodes.

a successful transmission means 1 node has $p=1$ and $(N-1)$ nodes have $p=0$. $(1-p)$
↳ mutually independent Coin-tosses

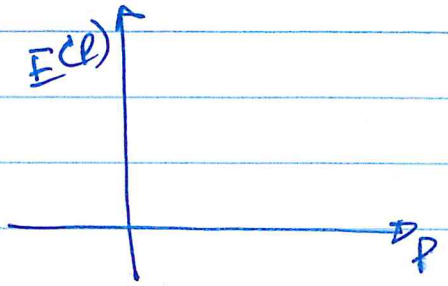
$$= p \times (1-p)^{N-1}$$

Since there are N nodes, the probability of any one of the nodes succeeding is,

$$N \cdot p(1-p)^{N-1}$$

max. efficiency is achieved when p^* maximizes the ~~prop~~ expression.

$$N p^* (1-p^*)^{N-1} \cdot$$



$$E(p) = N p^* (1-p)^{N-1}$$

$$E'(p) = N(1-p)^{N-1} - NP(N-1) \cdot (1-p)^{N-2}.$$

$$= N(1-p)^{N-2} \left[(1-p) - p(N-1) \right]$$

$$= N(1-p)^{N-2} \left[1 - \cancel{p} - pN + \cancel{p} \right]$$

$$= N(1-p)^{N-2} \left[1 - pN \right]$$

$$= N(1-p)^{N-2} \cdot \frac{1}{N} \left[\cancel{B} \frac{1}{N} \right].$$

when $E'(p) \Rightarrow 0$

$$1 - pN = 0$$

$$\boxed{p = \frac{1}{N}}$$

substitute this back,

$$E(p^*) = \cancel{N} \cdot \frac{1}{\cancel{N}} \cdot \left(1 - \frac{1}{N}\right)^{N-1} = \frac{\left(1 - \frac{1}{N}\right)^N}{\left(1 - \frac{1}{N}\right)}$$

$$\lim_{N \to \infty} \left(1 - \frac{1}{N}\right) \Rightarrow 1.$$

$$\lim_{N \to \infty} \left(1 - \frac{1}{N}\right) = 1$$

$$\lim_{N \to \infty} \left(1 - \frac{1}{N}\right)^N \Rightarrow \frac{1}{e} \qquad e \approx 2.71828$$

Thus,

$$\lim_{N \to \infty} E(P^*) = \frac{1}{e} = \underline{\underline{0.37}}$$

Thus, when a large # of nodes have many frames to send, the efficiency of slotted ALOHA drops to 37%.

ALOHA [Preceeds slotted ALOHA]

↳ As the name implies, no slotting, meaning, nodes are not synced. @

↳ Fully decentralized [@ what cost?]

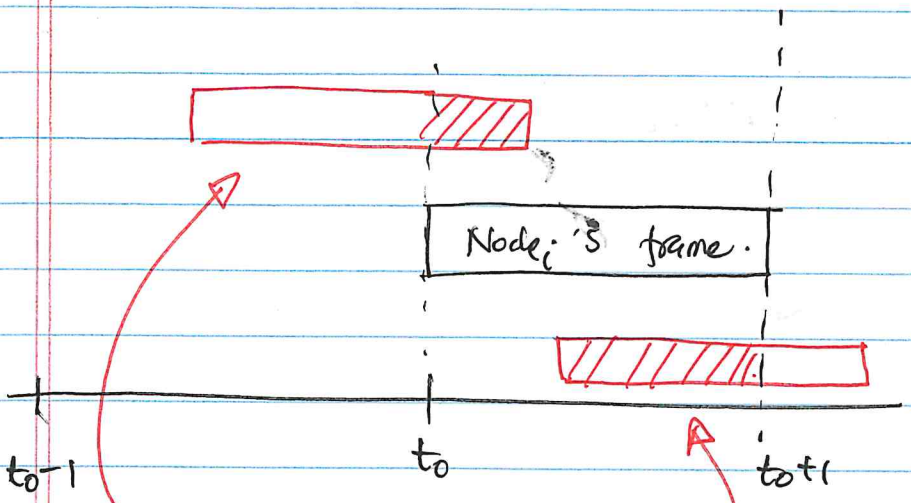
1. upon receiving the IP layer datagram, the node immediately begins transmission.

↳ If collision, immediatly retransmit with a probability P, after fully transmitting the "collided" frame.

or with.

wait a (1-P) probability

Efficiency of ALOHA



- No other node should have initiated transmission between $[t_{0-1} - t_0]$ → $(1-p)^{N-1}$

- No other node should initiate transmission between $[t_0 - t_{0+1}]$ → $(1-p)^{N-1}$

thus, a successful transmission by node$_i$ would be the probability of

$$E(p) = p(1-p)^{N-1} \times (1-p)^{N-1}$$

$$= p(1-p)^{2(N-1)}$$

$$E(p^*) = \frac{1}{2e}$$ which is half the efficiency of slotted ALOHA.

↑ Cost of fully decentralizing.

Taking-Turns protocol

ALOHA and CSMA CD — provide

(i) full throughput of R bps when only one active node

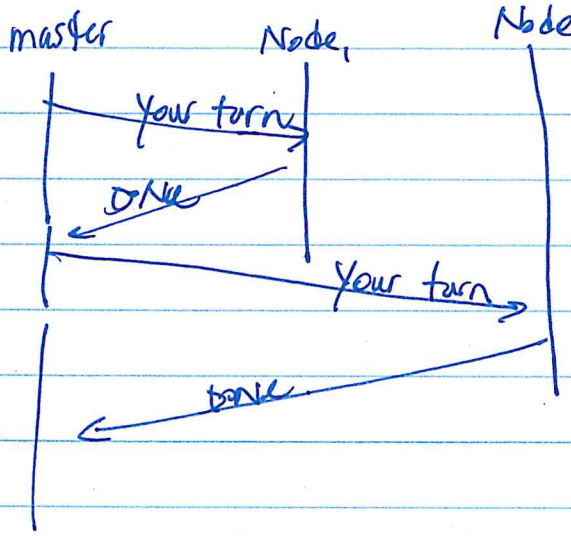
BUT NOT

(ii) avg. throughput of R/M bps if M nodes are active.

→ Alternatives

Polling Protocols

1. Designate a master who decides who gets the chance to transmit and for how long by polling



→ Master decides how long each node gets to transmit by listening to the channel; Lack of activity means done. (Signal Strength)

advantages: eliminates collision and empty slots
Higher efficiency

Disadvantages: Introduces polling delay.
Failure of master (SPoF)
unfair

token-passing Protocol

→ ~~whoever~~ who ever has the token gets to transmit.

e.g. token-ring : Token passed in a circle.

~~Soo~~

Link-Layer Addressing and ARP

→ switches operate @ link-layer. Does not understand ~~IP~~ IP datagrams; Does not employ RIP or OSPF like routing protocols.

→ ~~Swic~~ switches switch link-layer frames using (link-layer ~~IP~~ addresses. (MAC).

MAC address

Just like in IP addressing, each interface have a MAC (Media Access Control) access. This is the Hardware address.

Important: Not the switch but the interface(s) Connected to it has MAC address(es).

CSMA – Carrier Sense Multiple Access

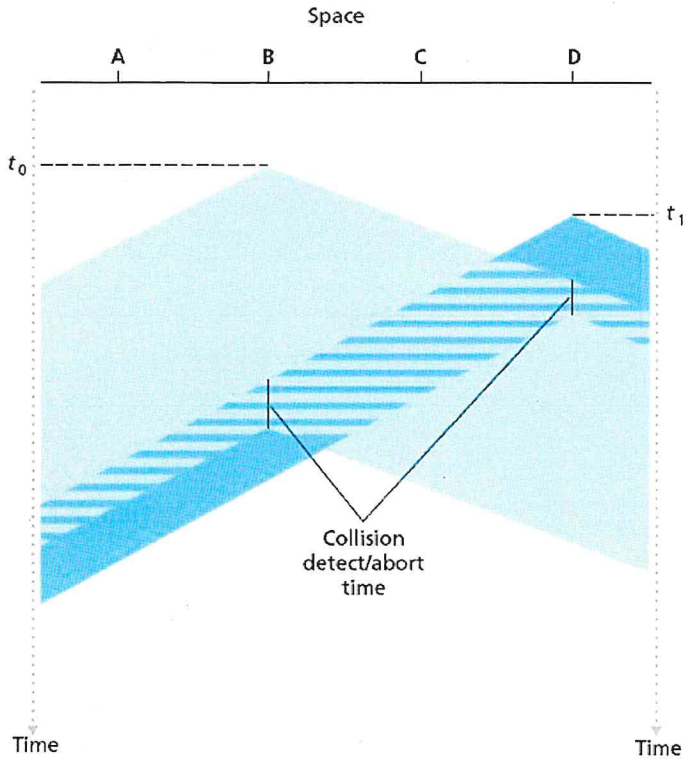
- Several variants
  - CSMA / CD (Collision detection) for ethernet
  - CSMA / CA (Collision Avoidance) for wireless.

General Idea:

- Listen before speaking → Carrier sense
- Stop talking if someone else is talking → collision detection

→ Retry or (transmit if no one is speaking) after a reasonable wait.

  - How long to wait depends on amount of contention (detected collisions).



CSMA Objective

- minimize collision
- maximize performance

- collisions can be minimized if a station "sense" the medium before using it.

Figure 5.13 ♦ CSMA with collision detection

The need to wait a random (rather than fixed) amount of time is hopefully clear—if two nodes transmitted frames at the same time and then both waited the same fixed amount of time, they'd continue colliding forever. But what is a good interval of time from which to choose the random backoff time? If the interval is large and the number of colliding nodes is small, nodes are likely to wait a large amount of time (with the channel remaining idle) before repeating the sense-and-transmit-when-idle step. On the other hand, if the interval is small and the number of colliding nodes is large, it's likely that the chosen random values will be nearly the same, and transmitting nodes will again collide. What we'd like is an interval that is short when the number of colliding nodes is small, and long when the number of colliding nodes is large.

The **binary exponential backoff** algorithm, used in Ethernet as well as in DOCSIS cable network multiple access protocols [DOCSIS 2011], elegantly solves this problem. Specifically, when transmitting a frame that has already experienced

CSMA/CD operation

Inter packet gap.

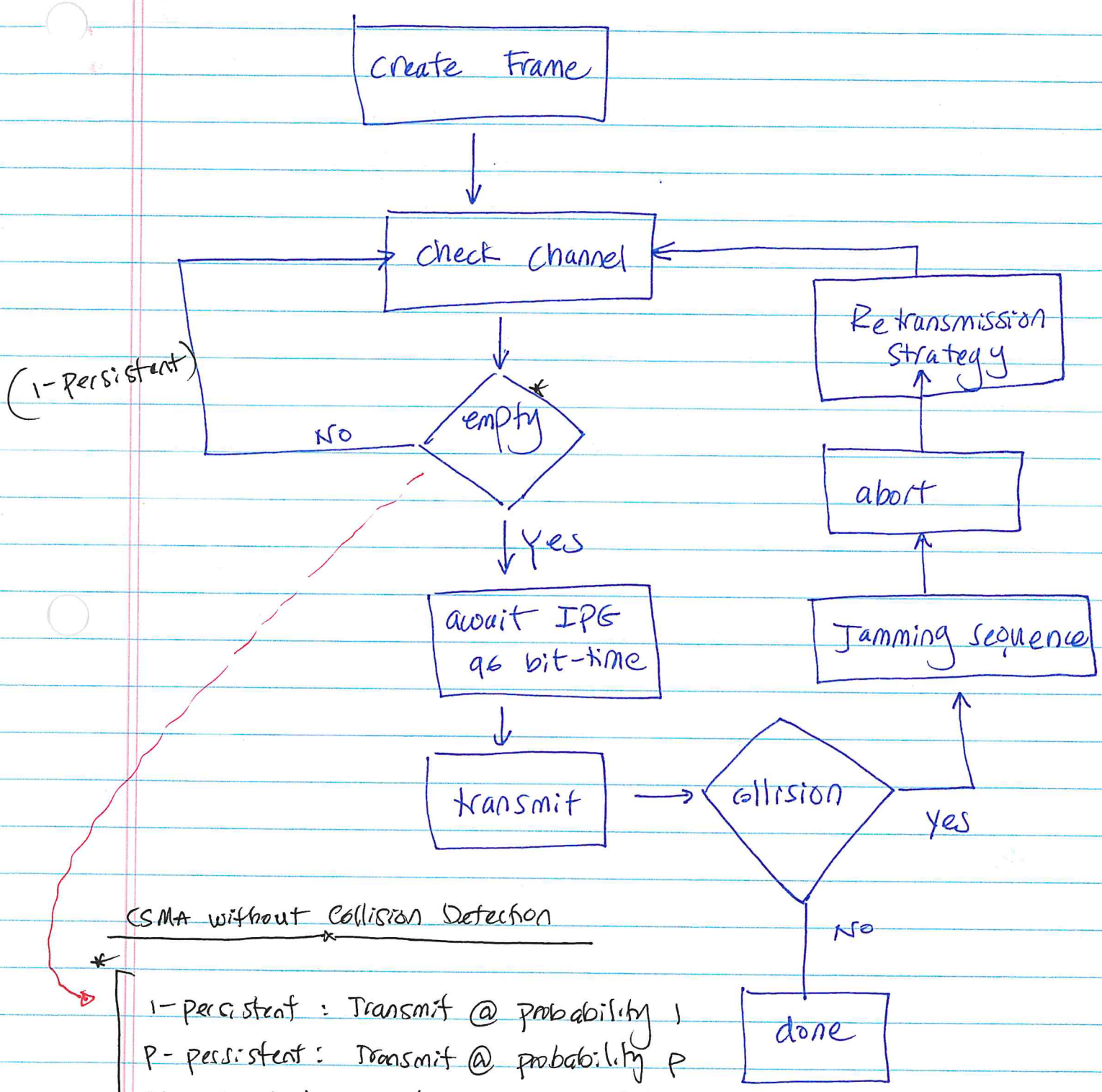
- when ready & line idle, await 96-bit times and send while listening

- If collision send a max 48-bit Jamming sequence and exponential backoff
  - Inform all stations that a collision has occurred.

- Retransmission strategy.
  For N collisions

if $(1 \leq N \leq 10)$
  chose k @ random on
$U(0 \ldots\ldots 2^{N-1})$ => Binary exponential Backoff.
if $(11 \leq N \leq 15)$
  chose k @ random on
$U(0 \ldots\ldots 1023)$
if $N > 16$
  drop frame

longer wait implies lower priority (Not a fair strategy).



CSMA without Collision Detection

1-persistent : Transmit @ probability 1

P-persistent : Transmit @ probability p

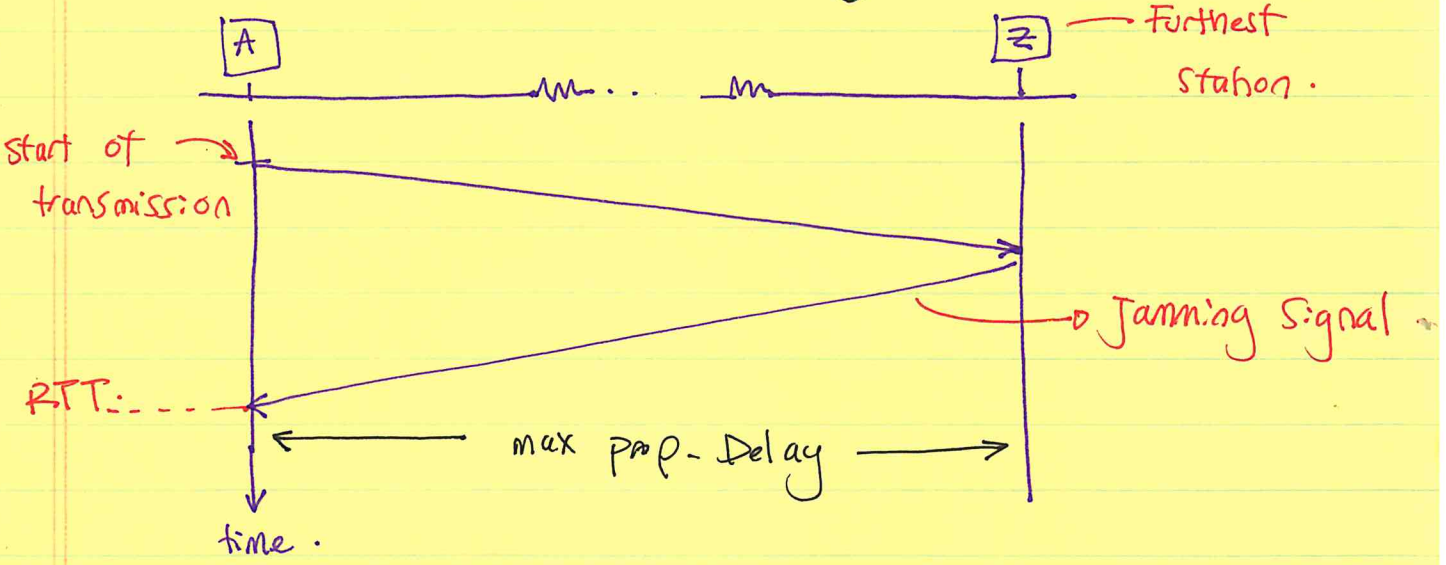
nonpersistent : wait a random delay if channel is busy before re-attempting [less greedy than 1-persistent]

Collision Detection

- Can observe channel for signal attenuation.
  e.g. - look for voltage signals exceeding transmitting voltage.

- Sender must not complete transmission until everyone has heard the frame.

  - Moreover, should ensure to detect collision before completing transmission.



- Sender must transmit beyond RTT.
  - Enforces a min. frame size.

Note: Stations do not hold on to 'successful' frames. If a collision goes undetected, frames could be lost for ever.

E.g. Assume a 10 Mbps LAN. A max. propagation delay 25.6 μs. ~~Assum~~ (Ignore time needed to send jamming sequence)

Min. frame transmission time = 25.6×2 μs
$= 51.2$ μs.

Amount of bits transmitted during time
time = $51.2 \times 10^{-6} \times 10 \times 10^{6}$
$= 512$ bits $= 64$ ~~kbps~~ kb.

which means, min frame size is 64 ~~kbps~~. kb

E.g. 2. A 1 Gbps LAN on a 1 km cable.
Prop. speed is 2×10^8 m/s.

- Assuming no other delays, $RTT = \frac{2 \times 10^3}{2 \times 10^8}$

$= 1 \times 10^{-5}$ s.

- Amount of bits transmitted $= 1 \times 10^{-5} \times 10^9$
$= 10^4$ bits.

Thus, min. Frame size is 10,000 bits = 1250 bytes.

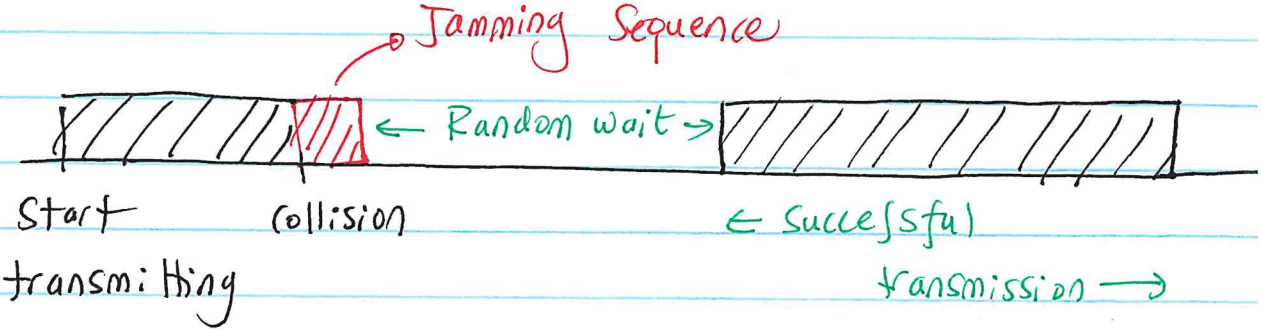
Captone Effect

- unfair strategy could cause one station to continuesly win over another.

  e.g. Assume A and B ready to send. If A wins and B backoff B will continue to wait as the probability of winning is halved every attempt.

CSMA/CD Efficiency

Typical Scenario.



Assume max. prop. time between two nodes is t_p.

- N independent nodes attempting to transmit

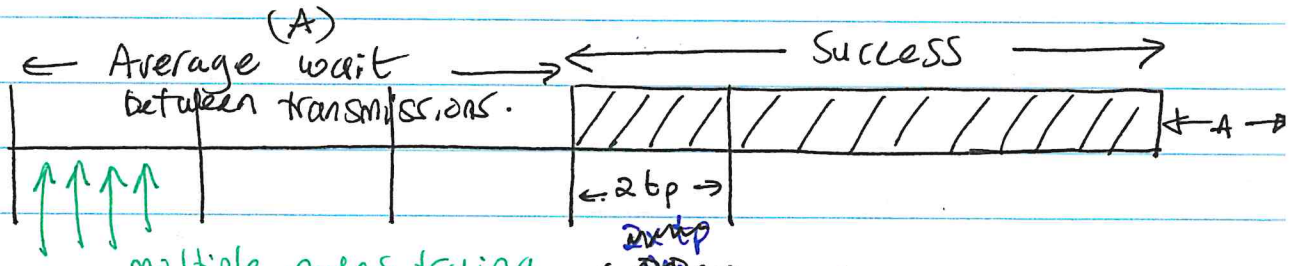
  - probability of successful transmission

$$N \times p \times (1-p)^{N-1}$$

taking the optimal probability $\left(p = \frac{1}{N}\right)$

from spotted ALOHA discussion,

$$p(\text{success}) = 0.36.$$



- each slot most be min 2x tp (for everyone to detect collision)

(no waiting when transmitting)

thus $A = 0.36 \times 0 + 0.64 (2T_p + A)$

$$A = \frac{1.28 T_p}{0.36} = 3.5 T_p$$

~~thus, overall wait for a successful transmission~~

~~$= 3.5 T_p + 2T_p = 5.5 T_p$~~

In practice

$A = 5 T_p$.

- detecting a collision will take $2t_p$
- detecting a successful transmission again will take $2t_p$
- On average the total time elapsed excluding transmission is $5t_p$ $5 * t_p$

not optimal in practice so $5t_p$ instead of $5.5t_p$.

includes backoff wait time.

$$\text{Efficiency} = \frac{L/R}{L/R + 5t_p}$$

$$= \frac{1}{1 + \frac{5t_p}{L/R}}$$

$5t_p$ → max. prop. time

L/R → transmission time

let $a = \frac{R \times t_p}{L}$

$$\text{CSMA/CD efficiency} = \frac{1}{1 + 5a}$$

Value of a impacts the overall efficiency.

→ If 'a' is small → early collision detection

Higher efficiency.

→ If 'a' is large → late detection

Inefficient

$\lim_{t_p \to 0}$ → efficiency → 1 (faster detection)

$\lim_{R \to \infty}$ → efficiency → 0 (larger frames are to be occupied → higher collision probability) (Not enought time to detect collision).

e.g. 100 Mbps LAN on a 1000 m Cable.
speed of light over copper 2.5×10^8 m/s.
Consider a 4000 ~~byte~~ bits Frame.

$$\text{Efficiency} = \frac{1}{1+5a} = \frac{1}{1+\frac{5 \times t_p \times R}{L}}$$

$$= \frac{1}{1+\frac{2}{5}\times\frac{\left(\frac{10^3}{2.5\times10^8}\right)\times 100\times10^6}{4\times10^3}} = \frac{1}{1+\frac{2\times10^3}{4\times10^3}}$$

$$= 66\%.$$

e.g. Assume cable upgraded to a 1 Gbps link.
and LAN shortened to 200 m.

$$\text{New Efficiency} = \frac{1}{1+\frac{2}{5}\times\frac{\left(\frac{200}{2.5\times10^8}\right)\times10^9}{4\times10^3}}$$

$$= \frac{1}{1+\frac{4\times10^3}{4\times10^3}} = 50\%.$$

Ethernet minimum packet size is 64 bytes for 10/100M but 512 bytes for 1000M. Minimum packet size is chosen on the basis that in case of half duplex, the sender should be able to detect collision before it finishes sending the frame.

At best the signals propagate (radiate) through free space at the speed of light (3*10^8 m/s). In contrast, the speed of propagation through twisted pair wire or coax is 2/3 of this value(2*10^8 m/s).

2*Maximum Distance between two hosts= speed of light in twisted pair * time

time=packet size/Bandwidth

For **10M, Bandwdith** =10^7bps

time=8*64bits/10^7bps=51.2us

So **Maximum Distance between two hosts**=(2*10^8)*(51.2*10^-6)/2=**5.12km**

For 10Mbps, this LAN length was more than sufficient.

For 100Mbps, LAN length comes out to be 512m which is also good enough.

But for 1Gbps, LAN length comes out to be only 51.2m. So the solution proposed was to increase the minimum ethernet packet size to 4096bits, i.e., 512bytes.

So max LAN length becomes 409.6m.

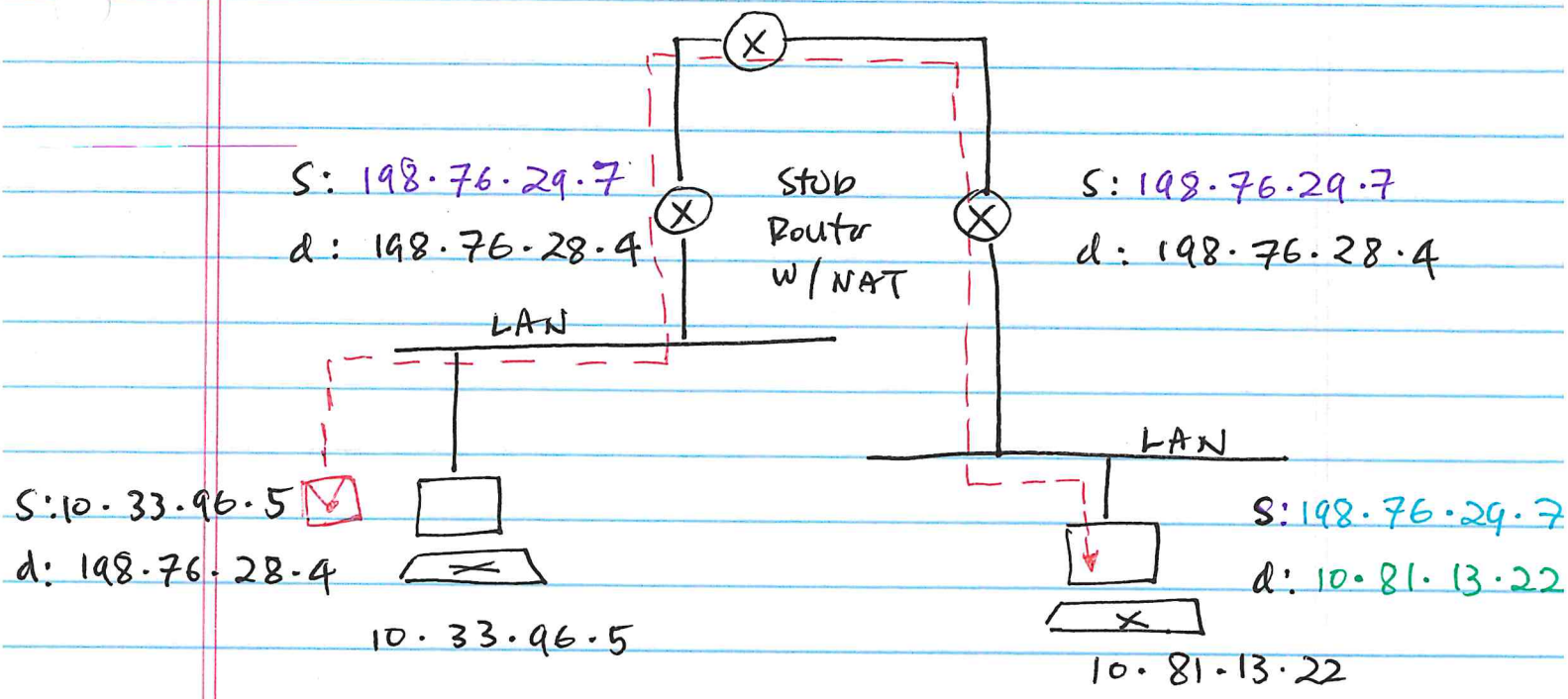
Network Address Translation (NAT)

- RFC # 3022 - A short RFC, worth reading as an assignment.

- Fundamental Idea: MAP IP addresses from one group to another transparent to the end users.

- Why?
  - Globally unique IP addresses are scarse and NOT cheap.
  - Lot more devices than available IP addresses
  - NOT really necessary to expose every device on public networks.
  - Internal IPs invalid outside the stub network
- A NAT enabled Router can "Translate" private internal IPs to public / External IPs.

NOTE: Conceptually two devices on "the internet" can't share the same IP address. This concept applies to any scope of a network.



Basic NAT Operation (from RFC 3022)

- NAT capitalizes on the CIDR addressing to allow more private IPs to "pseudo" exist as a public IP holder.

- NAT routers keeps an address translation table and other stateful information

- Common Types
  - Static NAT
  - Dynamic NAT
  - Port Address Translation (PAT) or Network Address Port Translation (NAPT)
    - Source NAT (SNAT)
    - Destination NAT (DNAT)

Static NAT

- One-to-one mapping between a private IP and a public IP address.
- Useful for servers within a private networ
- All requests from same internal IP+port always mapped to same external IP+port.

Dynamic NAT

- maps a group of private IPs to a pool of public IP addresses
- Addresses are translated FCFS basis
- Allows multiple devices to access the public networks using a limited #s of IP

Port Address Translation (PAT)

- Many-to-one translation
- a.k.a. NAT overload
- Most common in Home and small office networks
- port numbers used to distinguish private hosts

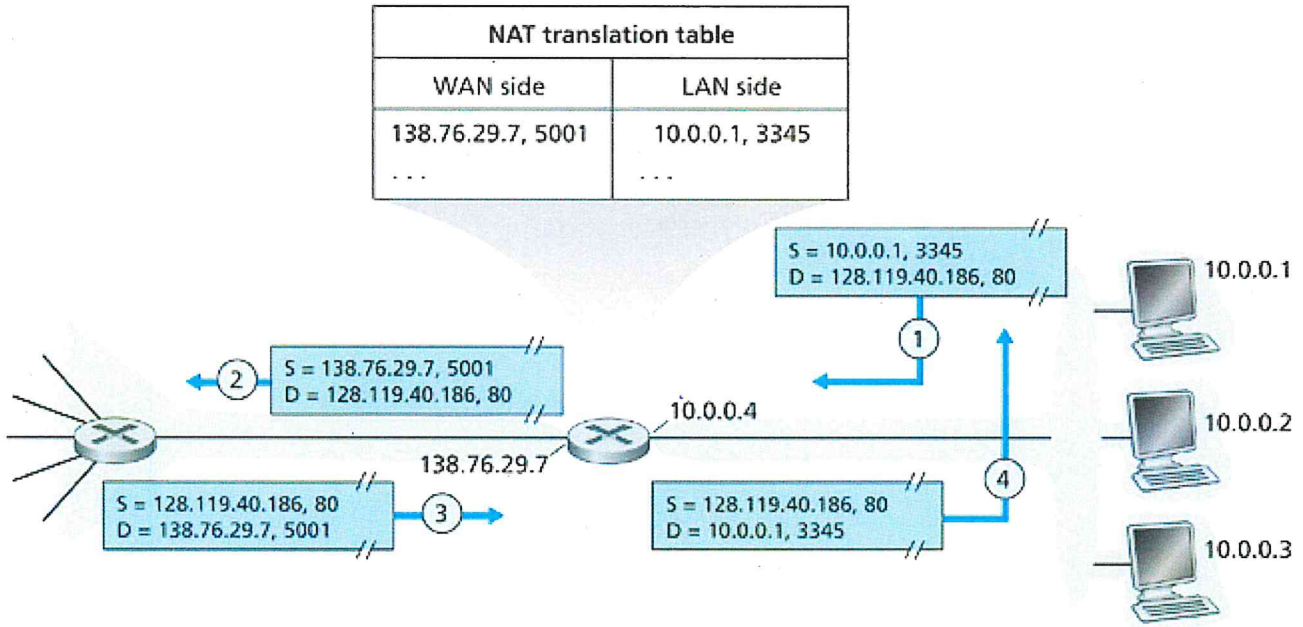
Source NAT — Translates the source IP of an (outbound) packet

Destination NAT - Translates destination IP of an (inbound) packet.
- useful for services like web servers or ftp servers.

Symmetric NAT

- All requests from the same internal IP + port to a specific external IP + port are mapped
  - Different mappings for different destinations.

PAT / NAPT Example



NAT traversal protocols

STUN (Session Traversal Utilities for NAT)

- originally designed for UDP. Now also supports TCP
- A stun server exists on the public network and report the public IP of a private host when queried.

TURN (Traversal Using Relays around NAT)

- provide a relay server acting as an intermediary for data transfer.
- useful for high reliability, low latency applications.

ICE (Interactive Connectivity Establishment)

- Combines STUN and TURN along with other techniques.

  e.g. WebRTC.

- NAT-PMP (NAT port mapping protocol) and PCP (port control protocol)

- Application-Level Gateways (ALG)

CONE NATS - Focus on outbound connections. (initiated from the private network).

- Full Cone NAT

- Restricted NAT

- Port Restricted NAT.

Analogy: Imagine a house with a single phone number (public IP).

- Full Cone: Once someone inside the house makes a call to a specific number (external host), anyone from outside can call home.

- Restricted: Only the external destination can call home (known numbers)
  - e.g. Game servers

- Port Restricted: Further restricted. The external host need to also use the same port.

NAT is a controversial Concept.

- Directly conflicts with the end-to-end principle
- Routers supporting NAT process more than Layer-3 traffic
- port # (TL construct) used to identify hosts in PAT
- IPv6 has a broad addressing space
- Running servers and P2P applications behind NAT need traversals.