Computer Networks Summary

Apr 10, 2021

Network Performance Metrics

Byte conversion

- bit = just one binary digit
- ASCII character = 7 bits, often stored in 8 bits
- $Byte = 8\ bits$
- The size of a `word` depends on the architecture you’re working in. In `x86 assembly language`, regardless of the machine word size:
    - `WORD` = 2 bytes
    - `long` (doubleworld: `DWORD`) = 4 bytes
    - `QUADWORD` (`QWORD`) = 8 bytes are used for 2, 4 and 8 byte sizes.
- char = 1 byte
- short = 2 bytes
- int = 32 bytes
- long = 64 bytes
- float = 32 bytes
- double = 64 bytes
- $KB = 2^{10}\ bytes$
- $MB = 2^{20}\ bytes$
- $GB = 2^{30}\ bytes$
- $TB = 2^{40}\ bytes$
- frame: m data bits + r redunancy check bits
    - codeword: n-bit unit containing 1 frame. where n = m + r
    - corate: fraction of the codeword that is not redundant: m/n

Speed

- Bandwidth: **Amount of total data that can maximally be sent via a connection per time unit**. (Max throughput) i.e. 10MB/s
- Latency: **bit travel time** from A to B. i.e. 20ms
- file transmission time (i.e. ms): **Time it takes to send all** bits of the message: (buffer) + file size / transmission rate
    - Transmission rate: **sent bits per unit of time** (usually the bandwidth or smaller)
- Packet Delivery Time (ms): file transmission time + latency
- Throughput (**data transmitted per unit** i.e. 5 MB/s): file size / packet delivery time
- Overhead percentage: supportive info/total data
    - Total data = goodput + supportive info
    - goodput: actual data
- Packet loss is associated with a probaiblity p
- Jitter: variance in transmission delay (in latency)
    - Data might have been lost and resent without it being apparent to the user in anything but jitter.

Bitwise operations

Display the integers as a binary string

- Java:

```
Long.toBinaryString(6) => "110"
```

- Python:

```
format(6, 'b') => '110'
```

Selecting

- To select parts of a binary sequence you can make use of bit masks. A bit mask is a binary sequence with 1s in the positions that are to be selected. This bit mask is then bitwise-AND'ed with the binary sequence. For example:

```
10110110 (input) AND 11110000 (mask) => 10110000
```

Moving

- To move the selected part left or right in the sequence, bitwise-shifts can be used. A shift moves the entire sequence left or right depending on the chosen operator. For example:

```
10110110 LEFT_SHIFT 1  => 101101100
10110110 LEFT_SHIFT 2  => 1011011000
10110110 RIGHT_SHIFT 1 => 01011011
10110110 RIGHT_SHIFT 4 => 00001011
```

- Be careful when shifting numbers by more than 32 steps. This will overflow if the number is not a 64-bit integer Combining To combine multiple bit sequences, a bitwise-OR can be used. For example:

```
10110000 OR 00000110 => 10110110
```

Length

- To calculate the length of the binary sequence (measured from the highest 1 bit), you can make use of:
- Generic:

```
floor(log2(10110110)) + 1 => 8
```

- Java:

```
long bitSequence = Long.parseLong("10110110", 2)
Math.floor(Math.log(bitSequence)/ Math.log(2)) + 1 => 8
```

- Python:

```
bit_sequence = int('10110110', 2)
bit_sequence.bit_length() => 8
```

Bitcount

- To calculate the amount of 1s in a binary sequence, you can make use of:
- Generic:

```
bit_count(integer)
count = 0
while int
count += (int AND 1)
int = int RIGHT_SHIFT 1
return count
```

- Java:

```
long bit_sequence = Long.parseLong("110", 2)
Long.bitCount(bit_sequence) => 2
```

- Python:

```
bit_sequence = int('110', 2)
bit_count(6) => 2
```

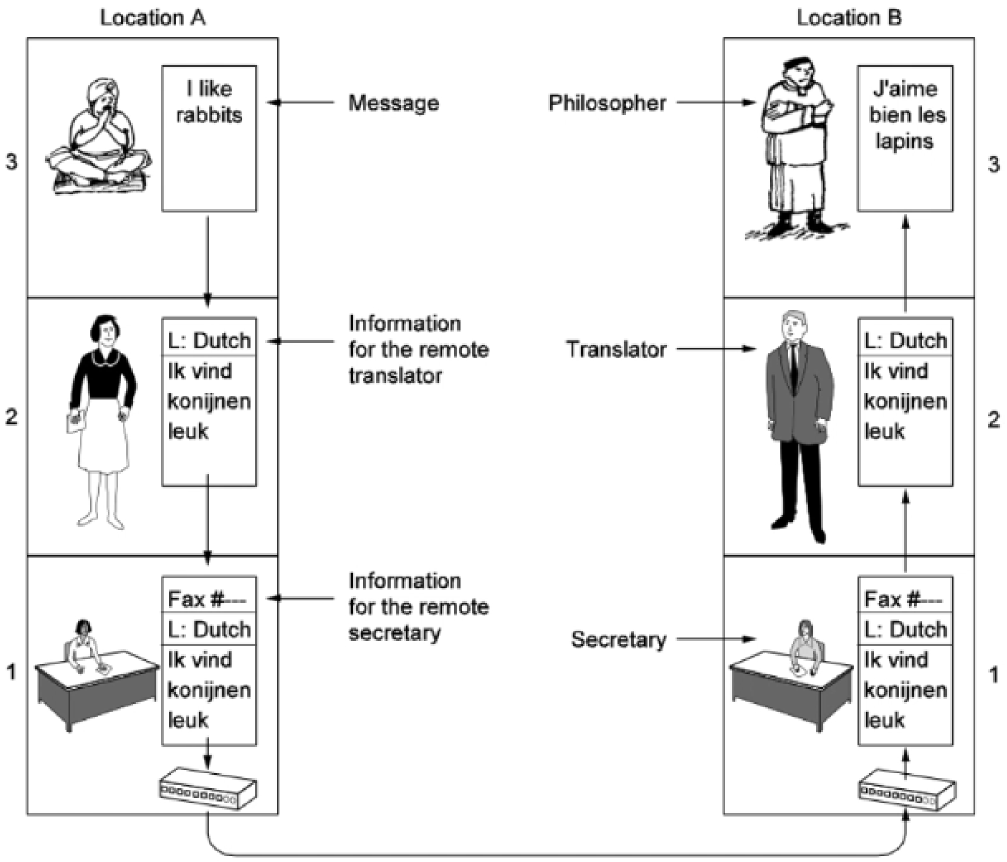
Java operators

- bitwise and: &
- bitwise or: |
- bitwise XOR*: ^ (it’s a binary sum without carry over)
- bitwise compliment: ~
- left shift: «
- right shift: »

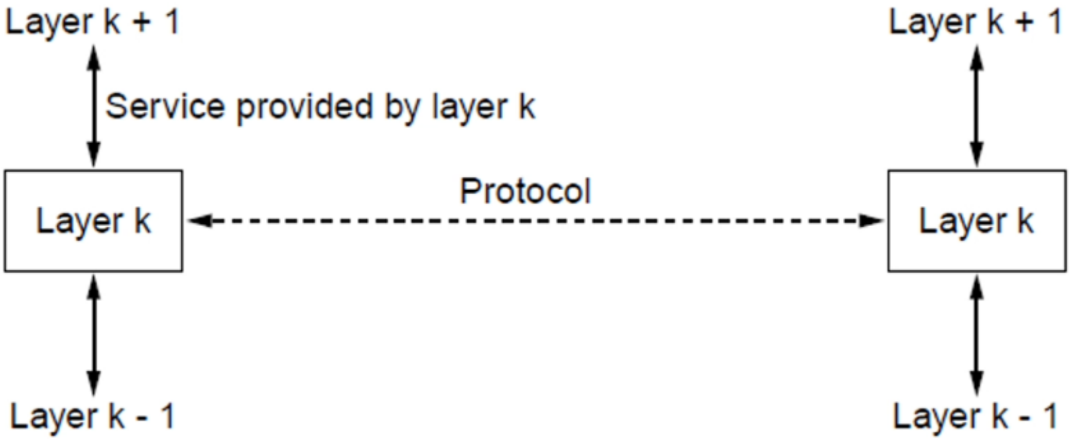
Java methods

```
private static void debugging() {
  // Showing binary representation of an integer value for debugging:
  long integerNotation = 6;
  String binaryStringNotation = Long.toBinaryString(integerNotation);
  System.out.println(integerNotation + " -> " + binaryStringNotation);
 }

 private static void binaryNotation() {
  // Creating integer from binary notation:
  long integer1 = Long.parseLong("110", 2);
  // Add L on the end for values larger than 32-bit
  long integer2 = 0b110;
  System.out.println(integer1);
  System.out.println(integer2);
 }

 private static void createBitSequence() {
  // Creating 010110001 (bit sequences) from scratch
  long res = 0;
  // or res = res | (1L << 7)
  res |= 1L << 7;
  res |= 1L << 5;
  res |= 1L << 4;
  res |= 1L;
  System.out.println(Long.toBinaryString(res));
 }

 private static void flippingBit() {
  // Flipping a bit in 10(1)1 to become 10(0)1
  long unflipped = 0b1011;
  // XOR original value with 10
  long flipped = unflipped ^ (1L << 1);
  System.out.println(Long.toBinaryString(unflipped) + " -> " + Long.toBinaryString(fli
 }

 private static void prepending() {
  // Prepending the bit sequence 1101110 with 101
  long original = 0b1101110;
  long prependee = 0b101;
  long prepended = original | (prependee << 7);
  System.out.println(Long.toBinaryString(original) + " -> " + Long.toBinaryString(prep
 }

 private static void appending() {
  // Appending the bit sequence 01110001 with 011
  long original2 = 0b01110001;
  long appendee = 0b011;
  long appended = original2 << 3 | appendee;
  System.out.println(Long.toBinaryString(original2) + " -> " + Long.toBinaryString(app
 }

 private static void changeBitSequence() {
  // Changing the bit sequence 11(010)11 to 11(100)11
  long unchanged = 0b1101011;
  long newValue = 0b100;
  // Create a mask that selects the parts that need to remain
  long mask = 0b1100011;
  // Clear part to be changed
  long changed = unchanged & mask;
  // Apply new value
  changed |= (newValue << 2);
  System.out.println(Long.toBinaryString(unchanged) + " -> " + Long.toBinaryString(cha
 }

 private static void inserting() {
  // Inserting a '1' in between bit sequence 110011010 to become 110(1)011010
  long uninserted = 0b110011010;
  // Mask to select first part
  long mask1 = 0b111000000;
  // Mask to select second part
  long mask2 = 0b000111111;
  // Create a gap
  long tmp1 = ((mask1 & uninserted) << 1) | (mask2 & uninserted);
  // Add the value 1000000
  long inserted = tmp1 | (1L << 6);
  System.out.println(Long.toBinaryString(uninserted) + " -> " + Long.toBinaryString(in
 }
```

Network Layers



- Each layer is a "level of abstraction"
  - Provides "services" to the layers above and below
  - Create the illusion of direct communication (horizontal line between same level layers), but in reality the information goes down from the origin up to the destination

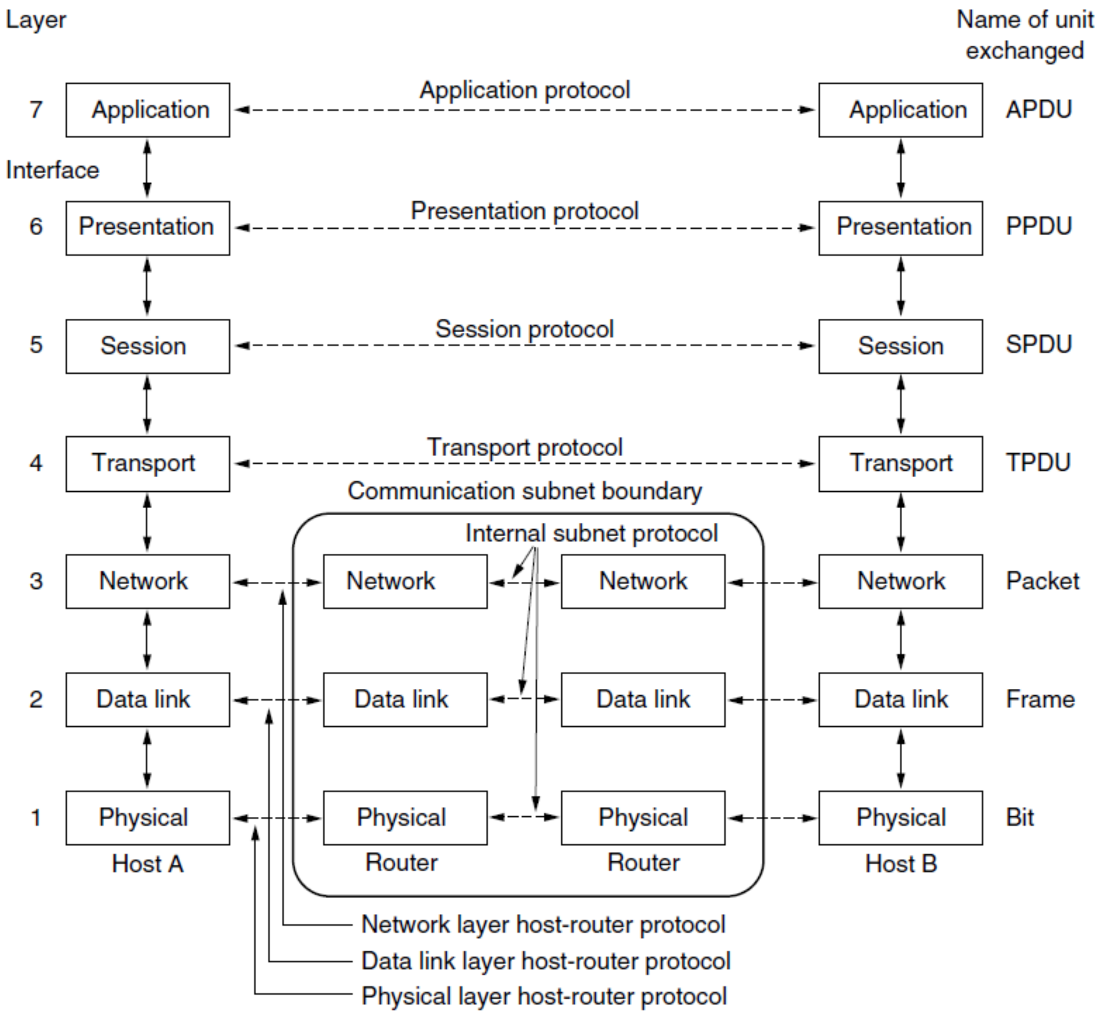


  - Only the lowest layer is the one that does the actual communication
  - Middle layers on different machines can have totally different service implementations, we dont care as long as the message is unchanged

Open Systems Interconnections (OSI) Model

- This "model" divides the internet to diverse protocols (illusory direct communication between layers)
  - Each protocol is assigned to each layer

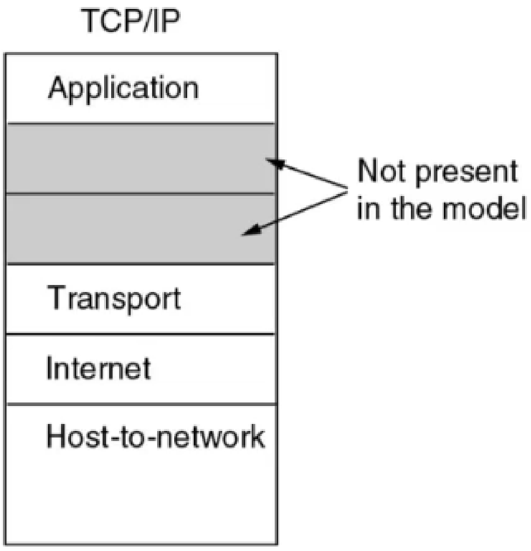
1. **Physical Layer** (lowest layer): Sends one bit via a physical medium, called a link
   - such physical medium can be an (ethernet) cable or a Wifi frequency
2. **Data link layer**: Sends a sequence of bits over a link *reliably*.
   - The pysical layer has a lot of errors (1 gets flipped into a 0 or viceversa or some data is lost during transmission). This layer deals with these errors so that the next layer doesnt even notice them.
   - **The MAC-Sublayer** regulates shared access to a (physical) medium, such as a wifi router. When multiple users are using a router, this sublayer manages how it is shared so that everybody can use it in a fair manner. MAC = medium access control
3. **Network layer**: Such as a LAN network or even the entire internet. This layer executes the sending of data from a source to a destination, which might be on the same network or on a different network.
4. **Transport layer**: The network layer has similar problems as the physical layer, where data is corrupted or lost, but at a higher level (i.e. sockets rather than single bits). This layer manages these problems and either decides to fix them or to ignore them (such as in a live videocall, where you do not want longer delays and rather have a potato but fast frames per second rather than HD frames per minute…). This is layer and upward ones only exist on the end hosts machines (i.e. the router has no longer nothing to do with). Therefore the source and destination only receive inderect feedback in the form of whether the packages have been received or not.
5. **Aplication layer** (highest level): These are protocols that deal with sending application specific information (email, web, instant messaging, games…)
   - The original OSI model actually has 2 additional layers between transport and application layer (5. Session and 6. Pressentation), but have no dedicated protocols and in practice the roles of these layers have been merged into either transport or application layers.



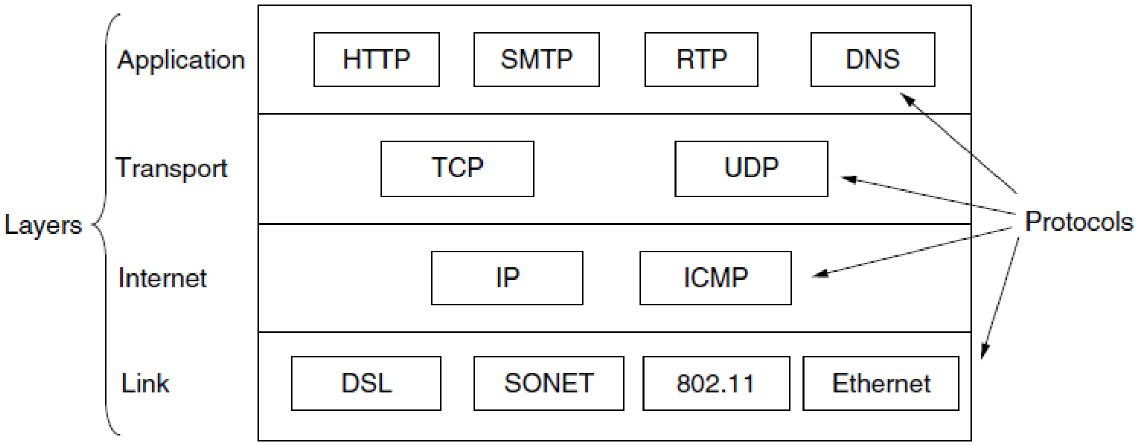
- Host stands for both source and destination parties
    - The data that a host sends is first forwarded to the lowest layers on its machine (network, data link, physical)
    - Each of these apply some “preparing” tasks if needed, then send to the router, which then sends the data to another router (who may send it to another one and so on until reaching the destination router), who will pump the information up to the higher layers who will forward it to the correct application layer. (i.e. whatsapp messages should be sent to the whatsapp app used by the phone with the corresponding local IP address)

TCP/IP model

- Whereas the OSI model (especially the original one that included Session and Presnetation) is a platonic model of what the ideal communication layers would look like, the TCP/IP model looked at the protocols that exist in practice and based on the ones being used it came up with:

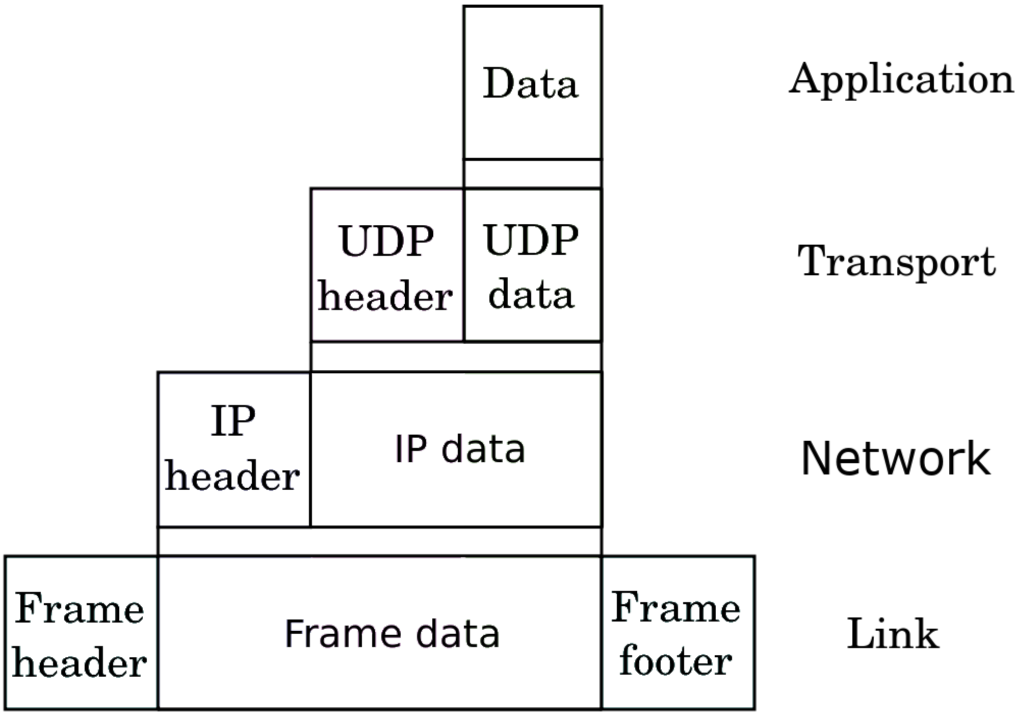


- It is practically the same but it officially removes the presentation and session layers, renames Network for Internet, and merges Data link and Pysical into Host-to-network (sometimes called the link layer). The reason of the merging is because even though OSI initially let each layer to be implemented independently, the wifi (pyhisical) was so irreliable that it was easier to enforce that all wifi layers are uniformly consistent with the data link layer so that errors are more intuitive and easier to handle.



- TCP is reliable and connection-oriented
- TCP handles flow control
- UDP is unreliable and connection-less oriented

Units of data



- The link layer wraps all the units of data, and the rest of the layers wraps those above, like a matrioshka:
  - Data link layer uses frames
  - Network layer uses packets
  - Transport layer uses segments
  - Application layer uses ???

Error Detection

Parity bits and checksums

- Checks for errors in the network. Possible causes are:
  - Errors between the client side wifi router and the machine
  - Errors in one of the multiple wifi routers on the server side
- Possible types of errors:
  - Bit could be flipped
    - Bit flip algorithms
  - Truncated/lost bits
    - Solution: Sender and reciver keep track of how much quantity they have sent and received

Error Detection

1. Receipient detects an error
2. Recipient tells sender that error has been found, requests resend
3. Sender has to resend the data

Error correction

1. Recepitent detects an error
2. Recipient is smart enough to fix it on its own

Both detection and correction are only possible by adding extra information to the messages which can allow to detect for errors.

Detection increasesy accuracy but increases latency and band-width usage whereas correction does the opposite.

- Just detection and request retransmission is preferred over cable connections as the the errors appear much less frequently than in wifi connections and the occasional double request is offset by having less overhead.
- For wireless connections errors are much more frequent and double-requests would be too expensive, therefore error-correction is prefrerred.

Parity bits

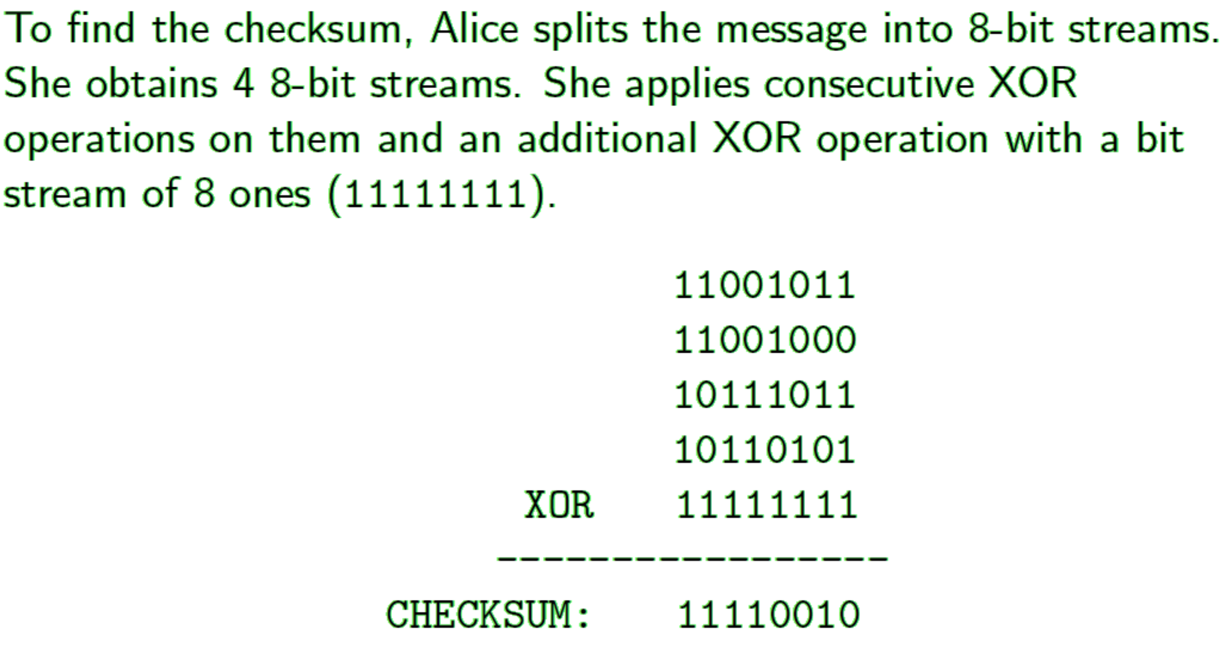
- A binary sequence has either even or odd parity (if it has an even/odd number of 1s)
- Parity bit algorithm: Sender and receiver agree on a fixed parity for all messages
  - Sender adds bit to message to get desired parity (appended to the right)
  - If parity is different then message must have been corrupted (however, keeping the same parity does not guarantee a successful message)
  - Algorithm can only detect odd number of bit flips

Checksum

- Divides bits into groups of k-bits. Assume k=4.
- Computes the binary sum (bitwise xor) of all the blocks (carried over bits are ignored)
- Appends the sum to the recepient
- The bitwise sum of the recepient which includes the appended sum should be all zeros (111 XOR 111 = 000), if not, then there was an error

Checksum in 1s complement

- It’s the same but in the XOR addition add a row of all 1’s.



To find the checksum, Alice splits the message into 8-bit streams. She obtains 4 8-bit streams. She applies consecutive XOR operations on them and an additional XOR operation with a bit stream of 8 ones (11111111).

```
              11001011
              11001000
              10111011
              10110101
       XOR    11111111
     -----------------
  CHECKSUM:   11110010
```

- The receiver should get all 1’s when adding the checksum

Hamming distance

To be able to detect or correct an error in a transmission, the data needs to have redundant information. This extra data can be used by the receiver to check if the original data has changed during the transmission and possibly correct the error. A piece of data together with its redundant information is called a codeword. Given an algorithm for computing the (redundant) check bits, it is possible to construct a set of all valid codewords, which is called a code.

The Hamming distance is used to count the number of bit positions that two codewords differ. To compute the Hamming distance of an entire code, you need to find **the smallest Hamming distance between all combinations of valid codewords in the code**.

For example:

```
Code = {0000, 0011, 1100, 1111}
distance(0000, 0011) = 2
distance(0000, 1100) = 2
distance(0000, 1111) = 4
distance(0011, 1100) = 4
distance(0011, 1111) = 2
distance(1100, 1111) = 2
```

Therefore Hamming Distance = 2

The error-detecting and error-correcting properties of a code depend on its Hamming distance. **To reliably detect d errors, you need a code with a minimal distance of `d + 1`** because with such a code there is no way that d single-bit errors can change a valid codeword into another valid codeword. When the receiver sees an illegal codeword, it can tell that a transmission error has occurred.

Similarly, **to correct d errors, you need a code with a minimal distance of `2d + 1`**. If all legal codewords differ in at least `2d + 1` bits, a codeword with at most d changes is still closer to the original codeword than any other codeword. This means the original codeword can be uniquely determined based on the assumption that d is in indeed the maximal number of errors.

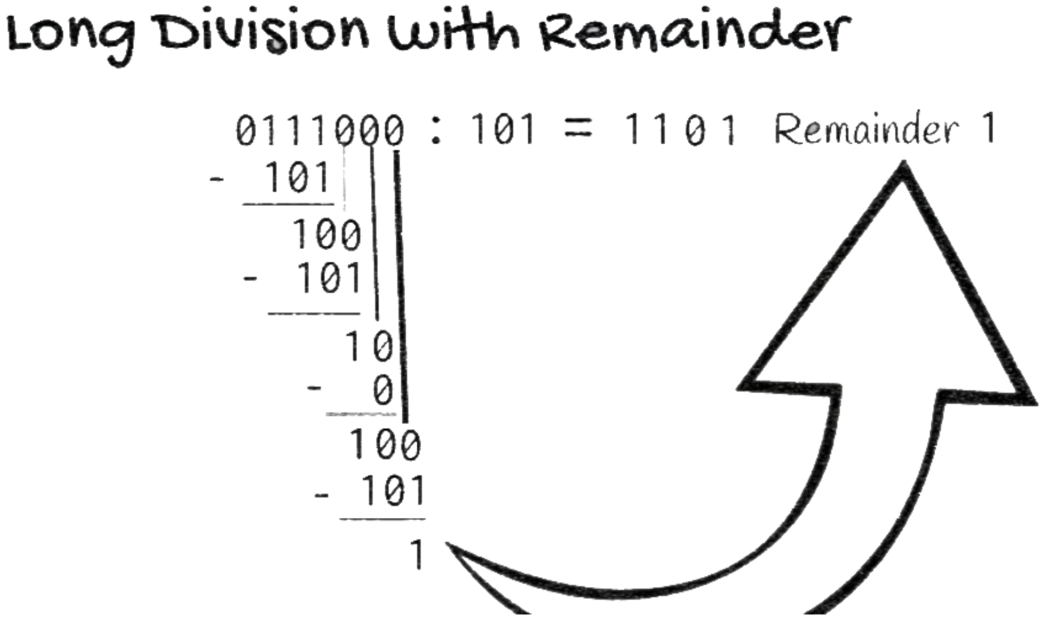
- The number of bits in a codeword is case specific and may or **may not** have the same number of bits as an assembly `WORD`.

```
class HammingDistance {

  /**
   * Calculates the hamming distance of the given code, or returns -1 if it cannot be ca
   *
   * @param code The code, consisting out of a list of codewords
   * @return The hamming distance of the given code , or -1 if it cannot be calculated
   */
  static long calculate(List<Long> code) {
    Long min = Long.MAX_VALUE;


  for (Long currentWord : code){
    for (Long otherWord : code){
        if (currentWord != otherWord){
                long xor = currentWord ^ otherWord;
                long count = Long.bitCount(xor);
          if (count < min){
            min=count;
          }
        }
    }
  }
  return min == Long.MAX_VALUE ? -1 : min;
  }
}
```

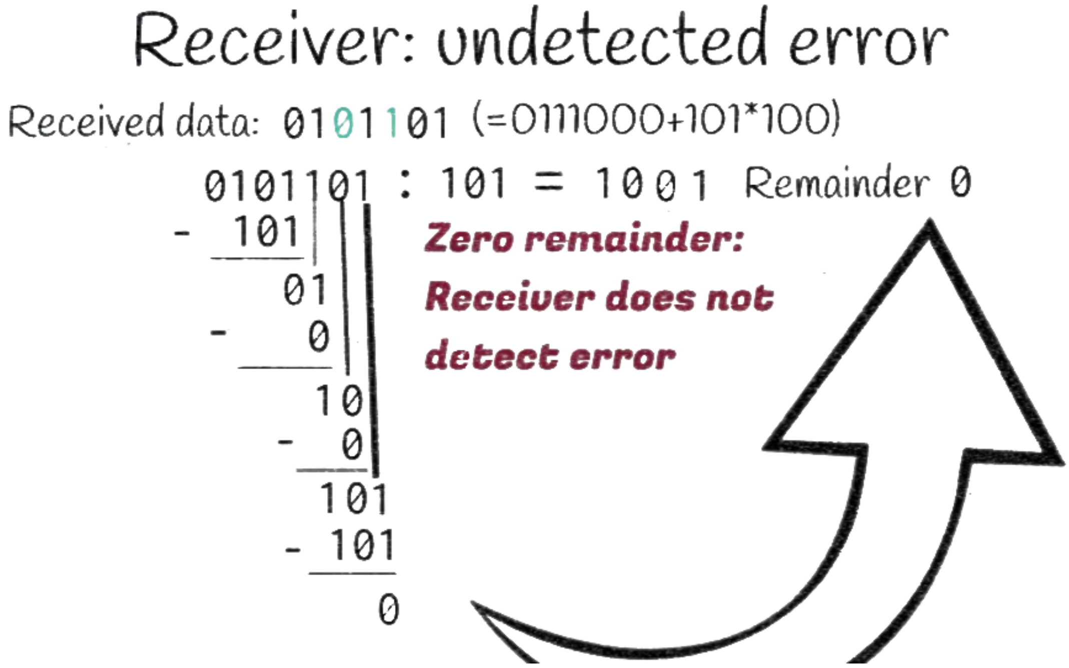
Cyclic Redundancy Check (CRC)

- Long division with remainder for binary numbers Uses a polynomial interpretation for each of the digits of a number, therefore there is like addition with no carryover (at all)

  - $x^3 + x^2 + x + 1$


Long Division with Remainder
0111000 : 101 = 1101 Remainder 1
- 101
100
- 101
10
- 0
100
- 101
1

- The data is the numerator and "error" is the denominator (different errors can find different amount of flipped bits)
  - CRC cannot detect any errors where the error is a multiple of generator
- The error denominator (aka generator) is chosen such that the remainder is 0
- When the receiver computes the division, if the remainder is not zero then it has detected an error
- There can be undetected errors if the changed bits come from the data + multiple of error


Receiver: undetected error
Received data: 0101101 (=0111000+101*100)
0101101 : 101 = 1001 Remainder 0
- 101
01
- 0
10
- 0
101
- 101
0
Zero remainder:
Receiver does not
detect error

- CRC is a very common error detection scheme, which is used in the Ethernet protocol
- it's not secure against deliberate attacks
- If we want to send our own sequence of bits, just divide it by the generator and append the result to the right of your sequence
  - Keep in mind that you have to append 0's always to the bit sequence before dividing it such that the appended block length is equal to the number of bits in the generator - 1. These appended bits are called "frame check sequence".

```
from library import bit_count

class CRC:

    @staticmethod
    def calculate(bit_sequence: int, input_length: int, generator_sequence: int) -> int:
        """
        Calculates the CRC check value.

        :param bit_sequence: The input bit sequence.
        :param input_length:  The length of the input bit sequence (including possible l
        :param generator_sequence: The generator bit sequence.
        :return: The CRC check value.
        """

        gen_length = len(format(generator_sequence, "b"))

        bit_sequence <<= gen_length - 1
        input_length += gen_length - 1

        for shift in range(input_length - 1, gen_length - 2, -1):
            print(format(bit_sequence, "b").zfill(input_length))
            if bit_sequence >> shift:
                bit_sequence ^= generator_sequence << (shift - gen_length + 1)

        return bit_sequence


    @staticmethod
    def check(bit_sequence: int, input_length: int, generator_sequence: int, check_seque
        """
        Checks the correctness of the bit sequence.

        :param bit_sequence: The CRC bit sequence excluding the CRC check value
        :param input_length: The length of the input bit sequence (including possible le
        :param generator_sequence: The generator bit sequence used
        :param check_sequence: The CRC check value to check against
        :return: True if the sequence is correct, False otherwise
        """

        return CRC.calculate(bit_sequence, input_length, generator_sequence) == check_se
```

Hamming code (n+k,n)

- Like the previous techniques were extra bits provide information that allows an algorithm to correct certain errors. Hamming Codes have two numbers:
    - n: bit length of the original data/message (d3 = data bit 1)
    - k: number of extra bits and their parity (the size of k is determined by n) (p1 = parity of slot 1)

Example: 10011010

a. Sender allocates n+k slots and enamurate the slots from the left starting at 1.

p1	p2	d3	p4	d5	d6	d7	p8	d9	d10	d11	d12

b. The sender copies the n-bit message into slots that are NOT powers of two.

p1	p2	d3	p4	d5	d6	d7	p8	d9	d10	d11	d12
		1		0	0	1		1	0	1	0

c. The other slots have the parity of a sequence of bits, depending on the position of the parity bit.

- p1 checks the parity of odd numbers, and converts itself to 1 or 0 to keep the XOR sum = 0
    - p1+1+0+1+1+1=p1+0, p1=0

p1	p2	d3	p4	d5	d6	d7	p8	d9	d10	d11	d12
0		1		0	0	1		1	0	1	0

- p2 checks 2 bits, skip 2 bits, formally checks the parity of binary numbers whose second bit is “ON”, i.e. 00**1**0,00**1**1,01**1**0,01**1**1,10**1**0,10**1**1,11**1**0,11**1**1=(2,3,6,7,10,11,14,15)
    - p2 = p2+1+0+1+0+1=p2+1,p2=1

p1	p2	d3	p4	d5	d6	d7	p8	d9	d10	d11	d12
0	1	1		0	0	1		1	0	1	0

- pn checks the parity of binary numbers whose k bit is “ON”…
- p4 is k-bit number 3, so 100,101… = 4,5,6,7,12,13,14,15
    - p4+0+0+1+0=p4+1,p4=1

p1	p2	d3	p4	d5	d6	d7	p8	d9	d10	d11	d12
0	1	1	1	0	0	1		1	0	1	0

- p8 is k-bit number 4 so 8-15, 24-31…

p1	p2	d3	p4	d5	d6	d7	p8	d9	d10	d11	d12
0	1	1	1	0	0	1	0	1	0	1	0

d. Receiver flips those bits whose parity doesnt match

- error at d10

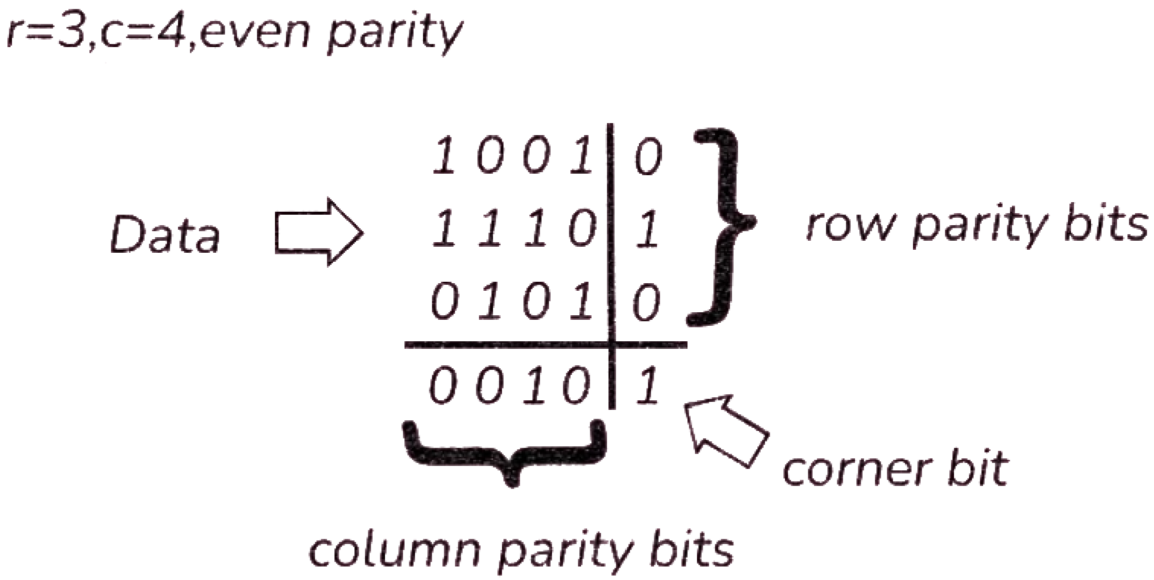
p1	p2	d3	p4	d5	d6	d7	p8	d9	d10	d11	d12
0	1	1	1	0	0	1	0	1	**1**	1	0

- Manually check the parity of all parity bits again
    - p2 is flipped
    - p8 is also fipped
    - 2+8=10, d10 is the bad bit

Parity block

Sender

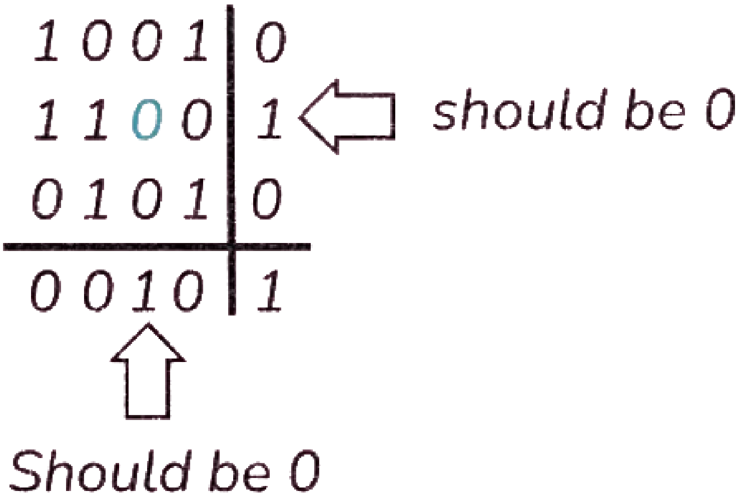
- Arrange the message in r by c bits matrix
- Compute the parity bits for all rows and columns respectively
- The corner bit is the parity of all bits, it has the same parity as the row and the column parity bits



```
1001 | 0
1110 | 1
0101 | 0
-----+--
0010 | 1
```

Receiver

- Assumes there was at most one bit flip
- Computes the parity bits for all rows and columns and finds the coordinates of the flipped bit and flips it

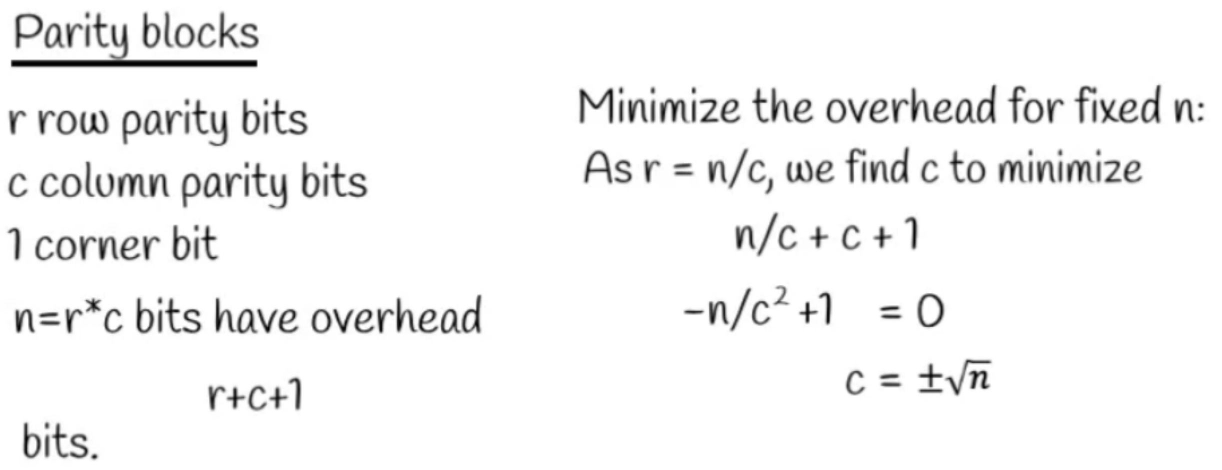


```
1001 | 0
1100 | 1
0101 | 0
-----+--
0010 | 1
```

- We can use the column/rows bit and corner bit to check if the flip was in the appended parity bits rather than in the data. These may be fixed depending on whether we forward these parity bits or not.

Overhead Analysis

- Overhead: amount of extra information required by protocol in comparision to actual data
  - If for 100byte of data you have 1byte of appended information then you have a 1% overhead with O(n)
- Parity blocks have an overhead of r+c+1 bits
- Use derivative to minimize overhead



Parity blocks

r row parity bits

c column parity bits

1 corner bit

n=r*c bits have overhead

$$r+c+1$$

bits.

Minimize the overhead for fixed n:

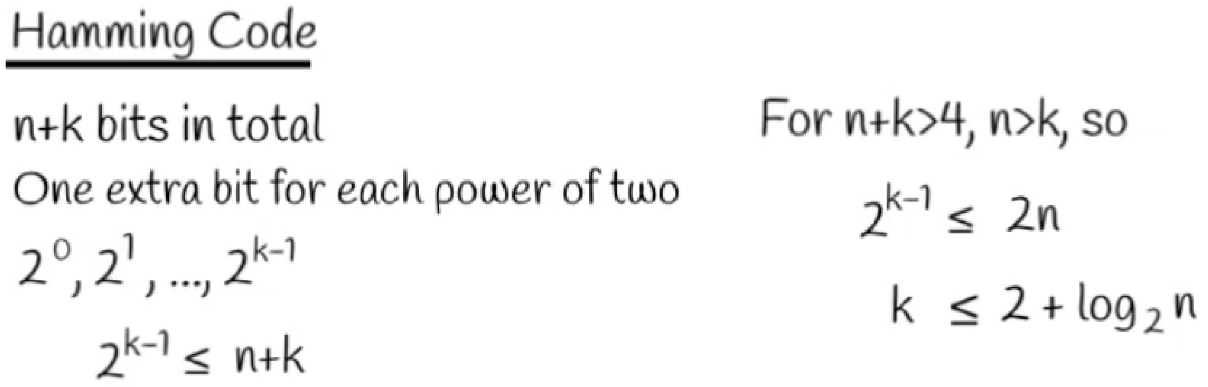
As r = n/c, we find c to minimize

$$n/c + c + 1$$

$$-n/c^2 + 1 \quad = 0$$

$$c = \pm\sqrt{n}$$

- This has asymptotic complexity of $O\sqrt{n}$
- Hamming Code has n+k bits overhead
- The asymptotic complexity is $O(log(n))$



Hamming Code

n+k bits in total

One extra bit for each power of two

$2^0, 2^1, ..., 2^{k-1}$

$$2^{k-1} \leq n+k$$

For n+k>4, n>k, so

$$2^{k-1} \leq 2n$$

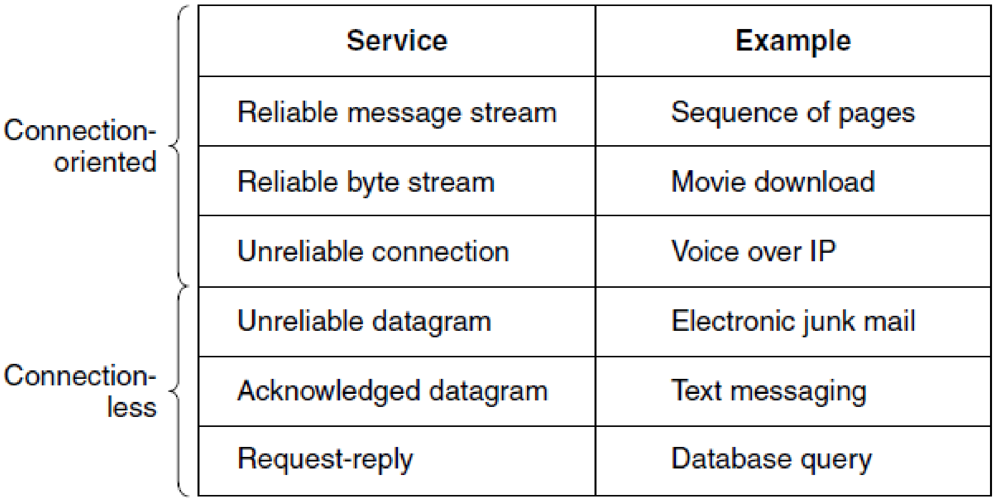
$$k \leq 2 + \log_2 n$$

- Both parityblocks and hamming code can only correct 1 bit flip. There are other algorithms that can flip more bits, but none of the standard network protocols use them.

Burst Errors

- Errors typically occur in bursts, so if one bit is flippied there is a high probability the next one is flipped as well
- We can mitigate error bursts by arranging the data in columns and rows and assigning a sepearate hamming code algorithm to teach column and row, which then only has to handle 1 flipped bit at a time

Services



	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Movie download
	Unreliable connection	Voice over IP
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Text messaging
	Request-reply	Database query

- Ethernet does not provide reliable communication (acknowledged).
    - It is up to higher protocol levels to recover from this problem.
- In real-life application unreliable communication is prefered because of speed and flow
- For file transfer reliable is imprescindible.

Connection-Oriented service

- Communication between layers based on the telephone system. To talk to someone, you pick up the phone, dial the number, talk, and then hang up.
- To use a connection-oriented network service, the service user first establishes a connection, uses the connection, and then releases the connection.
- The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out at the other end.
- In some cases when a connection is established, the sender, receiver, and subnet conduct a negotiation about the parameters to be used, such as maximum message size, quality of service required, and other issues.
- Typically, one side makes a proposal and the other side can accept it, reject it, or make a counterproposal.
- A circuit is another name for a connection with associated resources, such as a fixed bandwidth.

Connectionless service

- Communication between layers based on the postal system.
- Message carries full destination address
- each one is routed through the intermediate nodes inside the system independent of all the subsequent messages
    - a packet is a message at the network layer.
        - When the intermediate nodes receive a message in full before sending it on to the next node, this is called store-and-forward switching. The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called cut-through switching. Normally, when two messages are sent to the same destination, the first one sent will be the first one to arrive. However, it is possible that the first one sent can be delayed so that the second one arrives first.

Reliable service

- having the receiver acknowledge the receipt of each message so the sender is sure that it arrived. The acknowledgement process introduces overhead and delays, which are often worth it but are sometimes undesirable.

Connectionless application

- All that is needed is a way to send a single message that has high probability of arrival but no guarantee nor acknoweldge is returned to the sender. This is often called a datagram service.

Acknowledged datagram

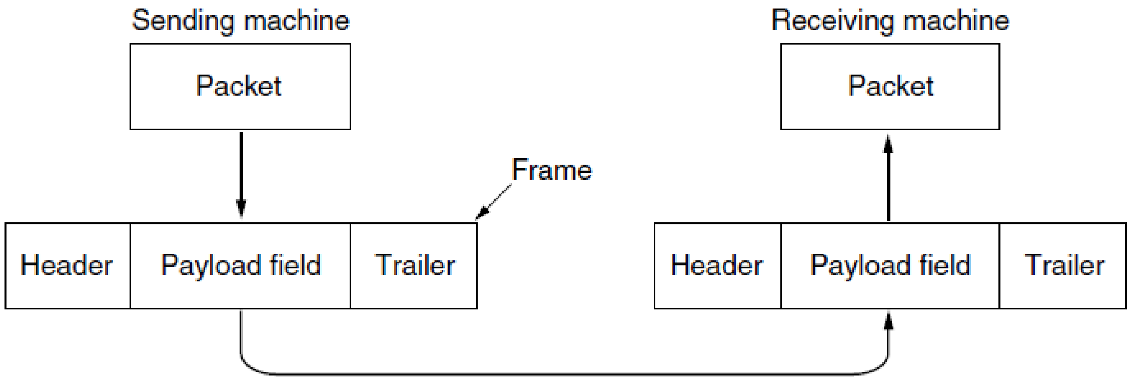
- acknowledged datagram service can be provided for these applications. It is like sending a registered letter and requesting a return receipt.
- When the receipt comes back, the sender is absolutely sure that the letter was delivered to the intended party and not lost along the way.
- Text messaging on mobile phones is an example.

Request-reply service

- In this service the sender transmits a single datagram containing a request; the reply contains the answer. Request-reply is commonly used to implement communication in the client-server model
    - the client issues a request and the server responds to it.

Data link layer

- The data link layer uses the services of the physical layer to send and receive bits over communication channels. It has a number of functions, including
    1. Providing a well-defined service interface to the network layer.
    2. Dealing with transmission errors.
    3. Regulating the flow of data so that slow receivers are not swamped by fast senders.
- To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer.



Relationship between packets and frames.

Services Provided to the Network layer

- The function of the data link layer is to provide services to the network layer.
    - transferring data from the network layer on the source machine to the network layer on the destination machine

Unacknowledged connectionless service

- having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. suchas Ethernet.
- No logical connection is established beforehand or released afterward.

acknowledged connectionless service

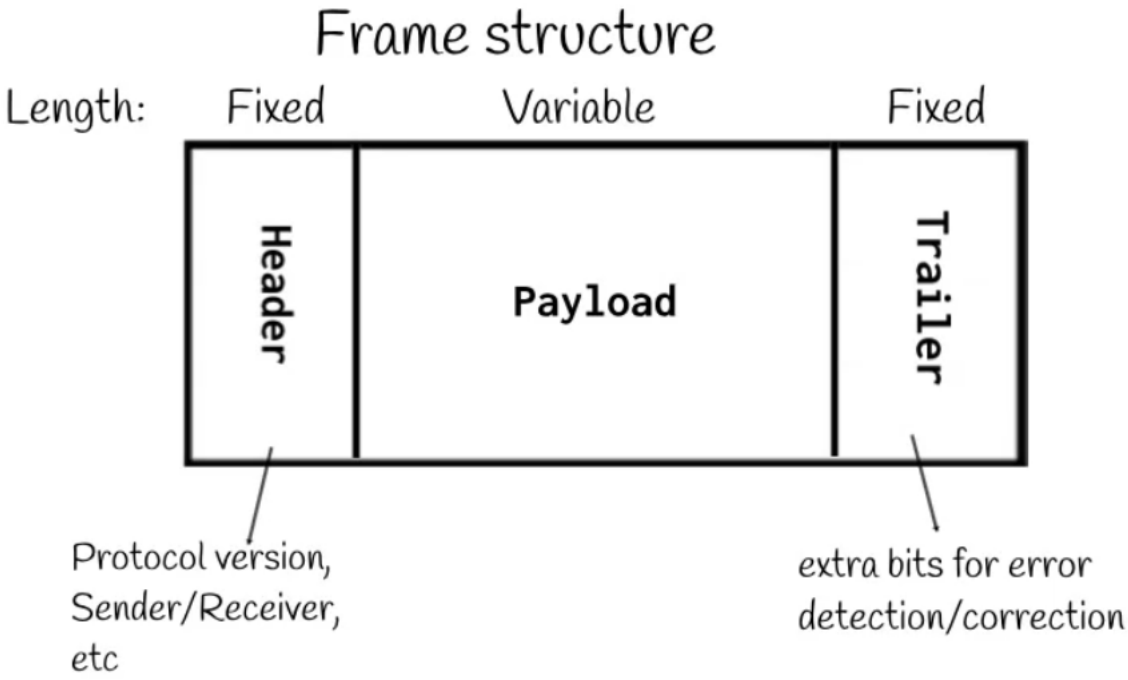
- there are still no logical connections used, but each frame sent is individually acknowledged.
- the sender knows whether a frame has arrived correctly or been lost. If it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems. 802.11 (WiFi) is a good example of this class of service.
- On reliable channels, such as fiber, the overhead of a heavyweight data link protocol may be unnecessary, but on (inherently unreliable) wireless channels it is well worth the cost.

connection-oriented service

- the source and destination machines establish a connection before any data are transferred.
- Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received.
- it guarantees that each frame is received exactly once and that all frames are received in the right order
- provides the network layer processes with the equivalent of a reliable bit stream.
- transfers go through three distinct phases:
  - In the first phase, the connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not.
  - In the second phase, one or more frames are actually transmitted.
  - In the third and final phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection.

Data link layer framing

- The main responsibility of the data link layer is to provide reliable transport of bits.
  - To achieve that it uses error detection or error correction
    - You dont apply that on the entire file, you do it in “frames”


Frame structure
Length:
Fixed
Variable
Fixed
Header
Payload
Trailer
Protocol version, Sender/Receiver, etc
extra bits for error detection/correction

- Ethernet damaged cables have occasional data loss just as wifi connected devices far from the wifi range.

How to determine start of a frame

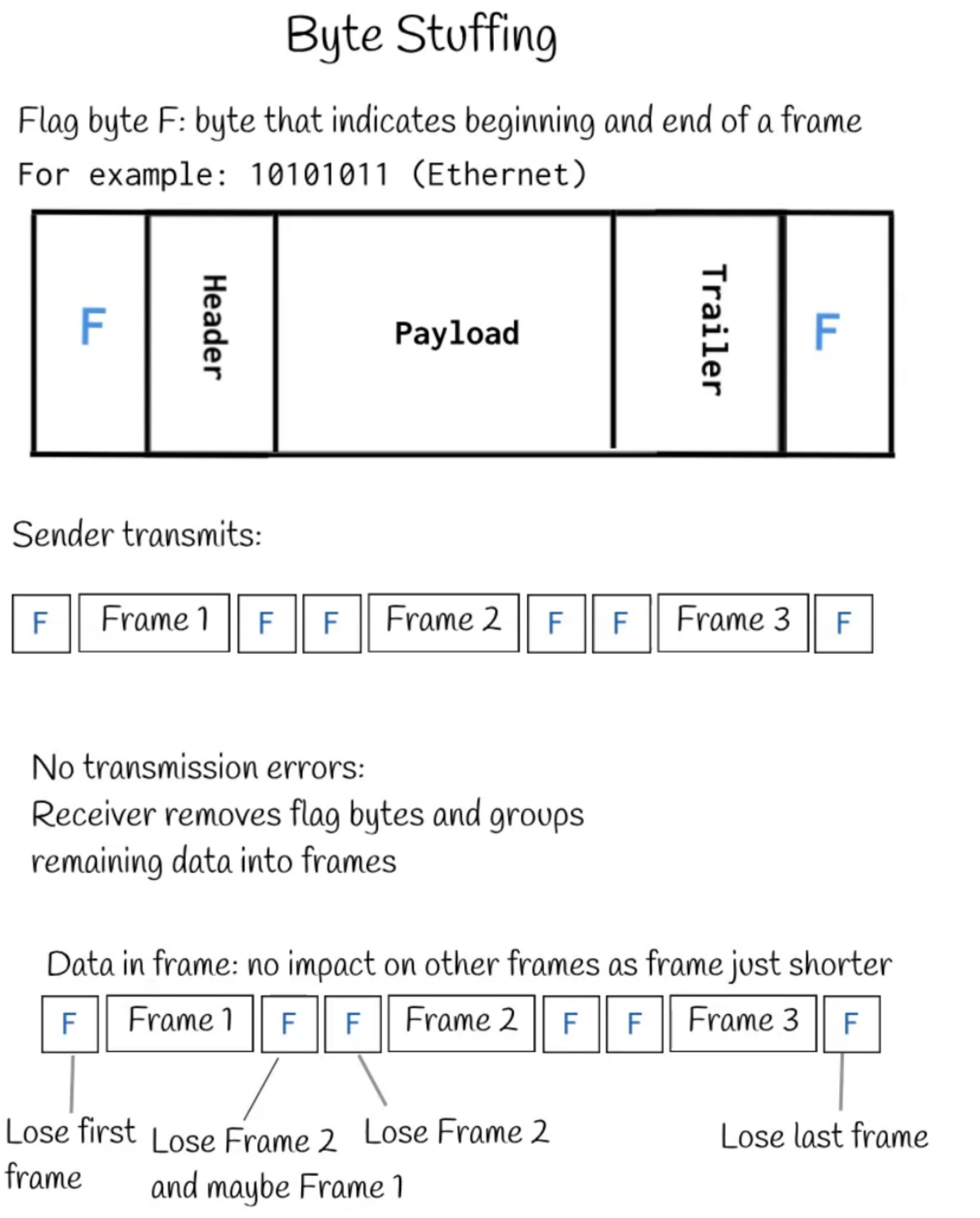
- Break between frames

  - But: telling breaks from lost bits is impossible & several parties using the medium can cause interference and mask breaks.
- Fixed length frames
  - But hard to tell which frame is damaged

Therefore:

1. Only rely on data rather than on assumptions about speific properties of the medium (such as ability to detect breaks)
2. Use an algorithm so that data loss only has consequences for the affected frames

Byte Stuffing: Flag determines beginning and end of the frame.



Byte Stuffing

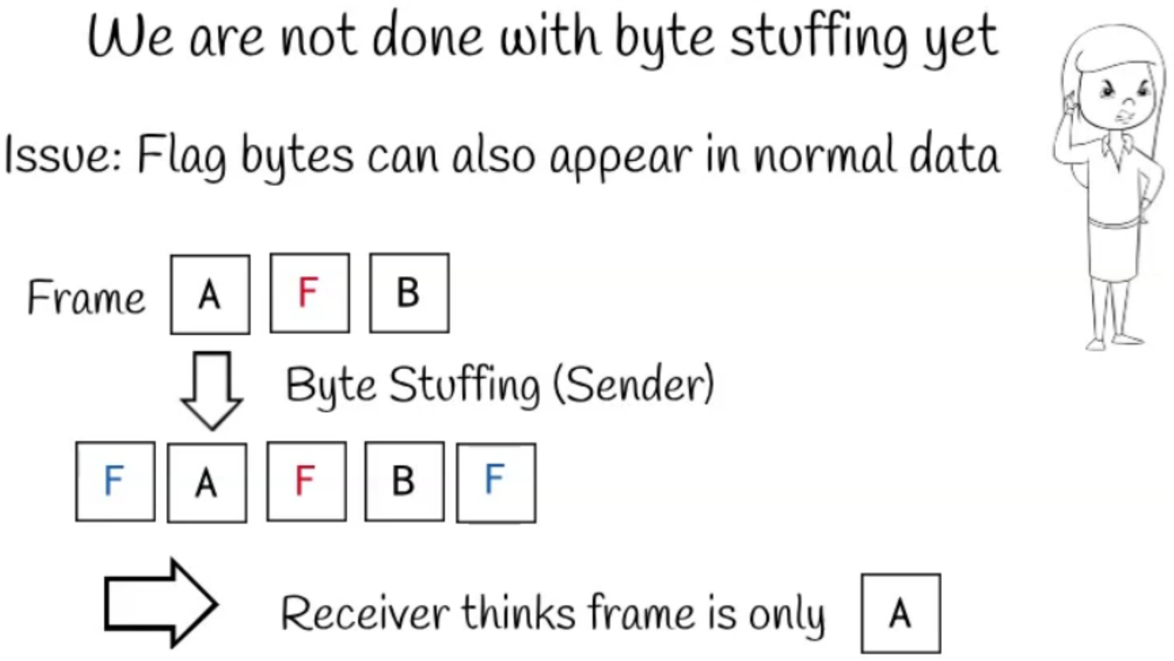
Flag byte F: byte that indicates beginning and end of a frame

For example: 10101011 (Ethernet)

Sender transmits:

No transmission errors:
Receiver removes flag bytes and groups remaining data into frames

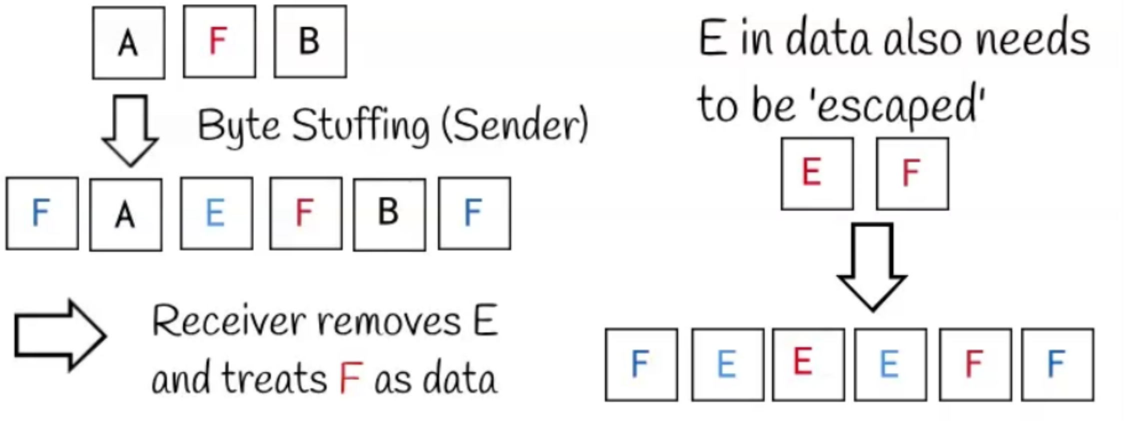
Data in frame: no impact on other frames as frame just shorter



We are not done with byte stuffing yet

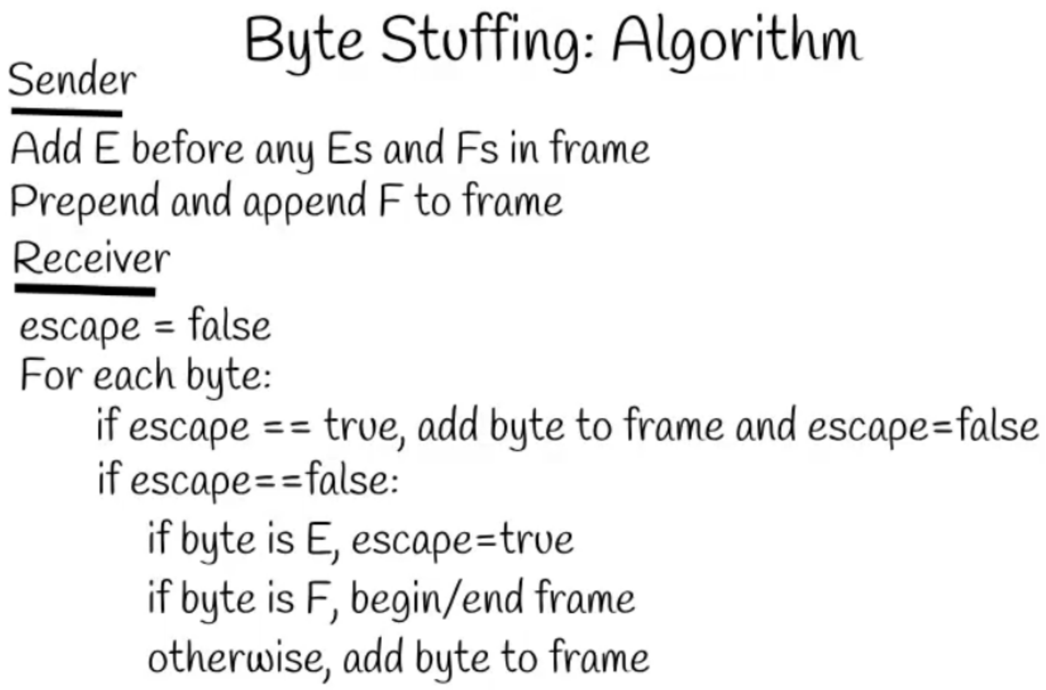
Issue: Flag bytes can also appear in normal data

- Escape byte is a byte that indicates that next byte should be treated like normal data and not as a special character.
  - The escape byte also needs to be escaped



E in data also needs to be 'escaped'

Receiver removes E and treats F as data

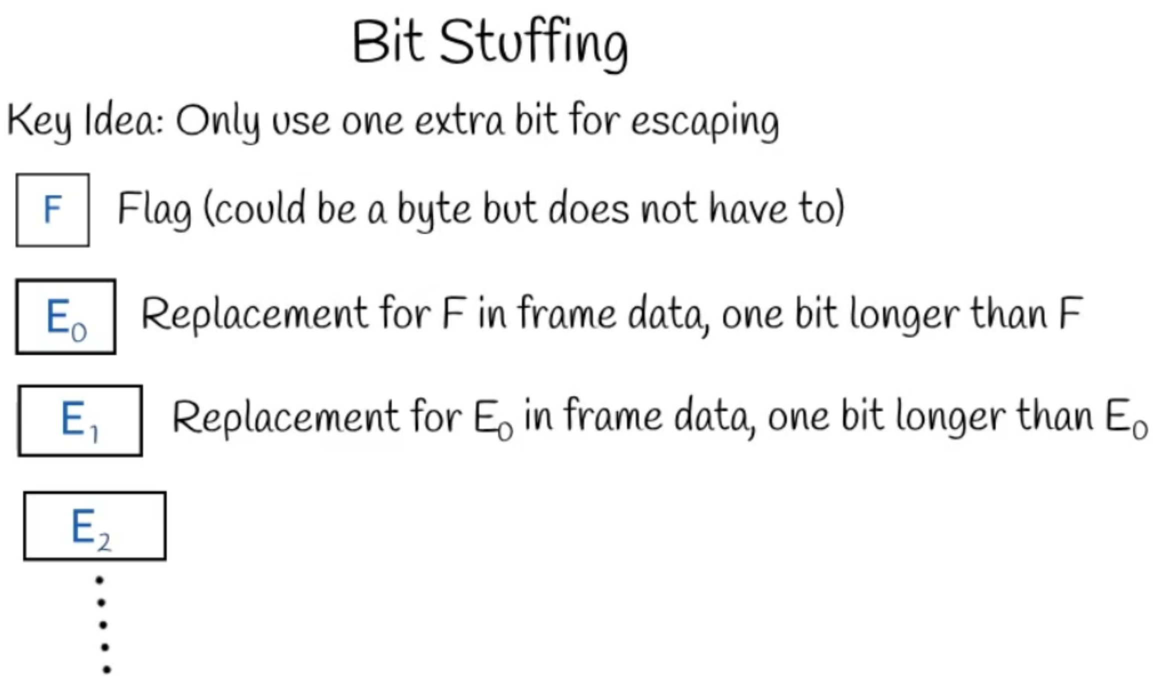


Byte Stuffing: Algorithm

```
Sender
Add E before any Es and Fs in frame
Prepend and append F to frame
Receiver
escape = false
For each byte:
    if escape == true, add byte to frame and escape=false
    if escape==false:
        if byte is E, escape=true
        if byte is F, begin/end frame
        otherwise, add byte to frame
```

- Disadvantage: a lot of overhead due to escape bytes

- Alternative, Bit stuffing:



Bit Stuffing

Key Idea: Only use one extra bit for escaping

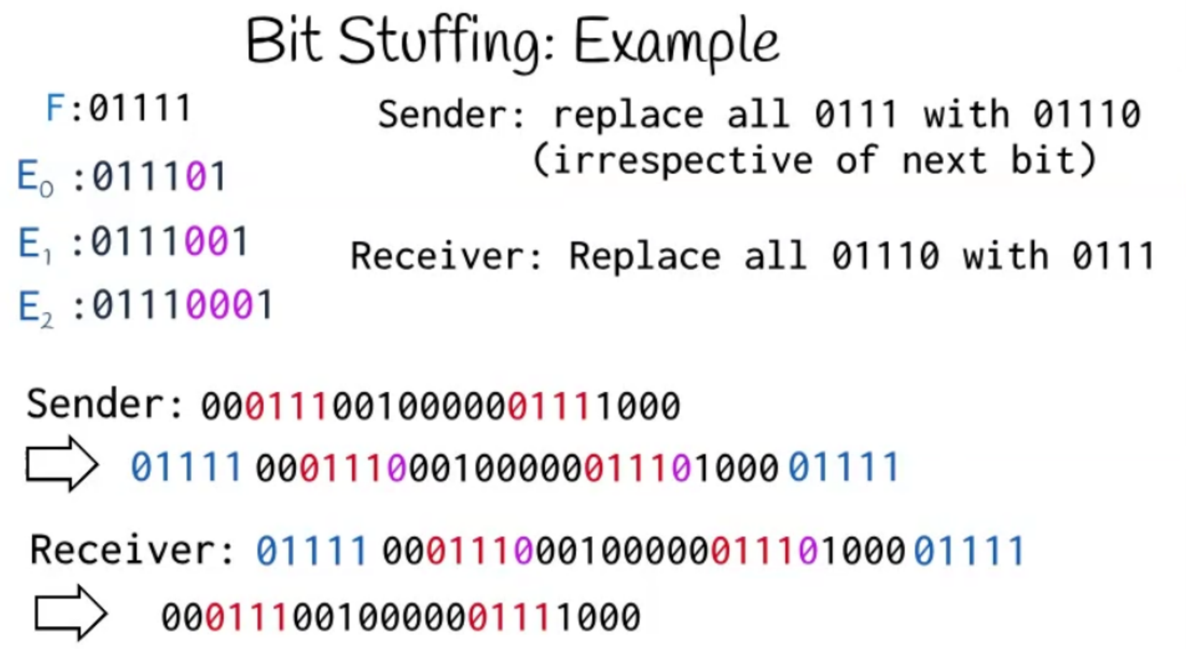
F Flag (could be a byte but does not have to)

E_0 Replacement for F in frame data, one bit longer than F

E_1 Replacement for E_0 in frame data, one bit longer than E_0

E_2

⋮



Bit Stuffing: Example

F:01111

E_0 :011101

E_1 :0111001

E_2 :01110001

Sender: replace all 0111 with 01110 (irrespective of next bit)

Receiver: Replace all 01110 with 0111

Sender: 0001110010000001111000

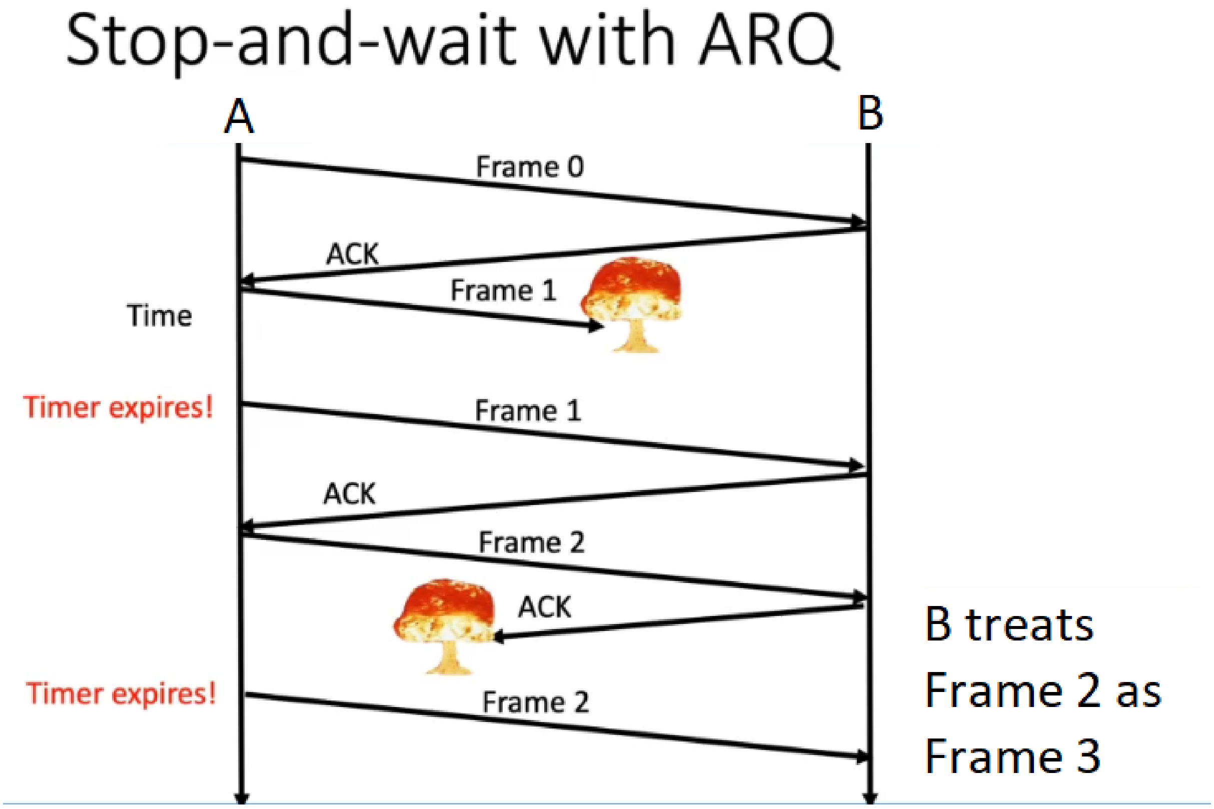
⇨ 01111 00011100010000001110 1000 01111

Receiver: 01111 00011100010000001110 1000 01111

⇨ 0001110010000001111000

Dealing with lost frames: Automatic Repeat ReQuest (ARQ) for noisy channel uses acknowledgement

- Receiver acknowledges frames that are correctly delivered
- Sender sets timer and resends frame if no ack
- If there is no acknowledgement (timeout) then sends the frame again
- It can be the rare case when there is no acknowledgement wihin the timout but it has indeed been sent (it's just slow) then you'd have a duplicated frame



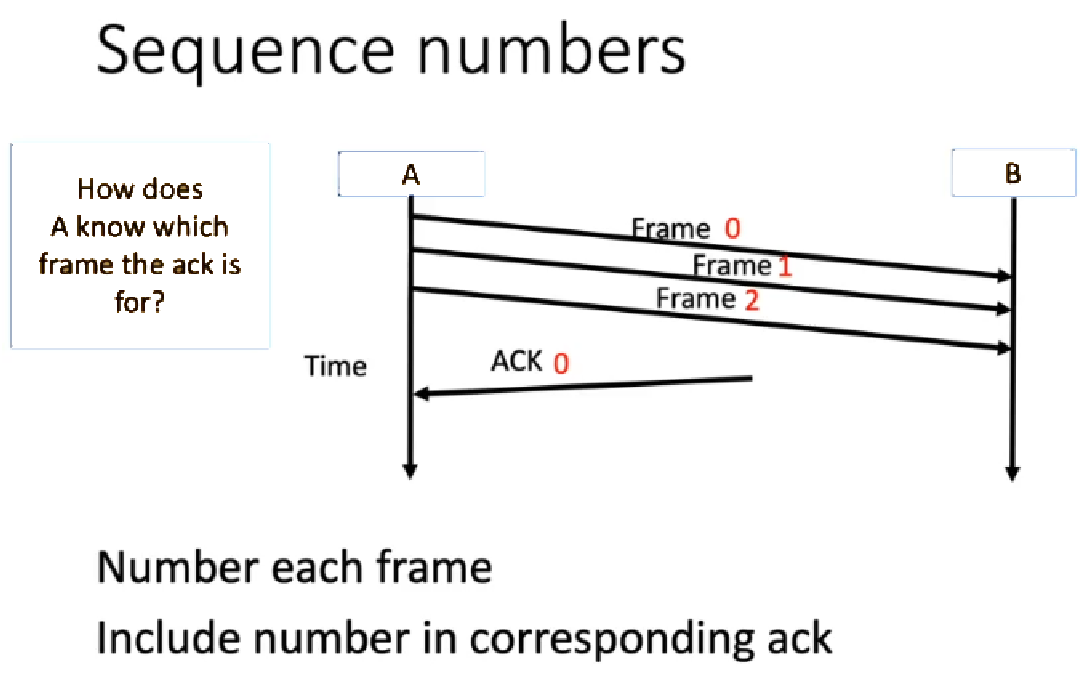
- Frames and acknowledgements must be numbered to fix the duplicated issue
- The problem of sop and wait is that it wastes a lot of bandwidth by waiting

Sliding windows

- The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as piggybacking. The principal advantage of using piggybacking over having distinct acknowledgement frames is a better use of the available channel bandwidth.
- This is something different to the images below, what it means is that in a 2-way communication where A and B both are poractively talking, instead of wasting one dummy frame for just the ack, add the ack into the header of the next frame containing real data.
- However If the data link layer waits longer than the sender's timeout period, the frame will be retransmitted, defeating the whole purpose of having acknowledgements.
- Therefore, If a new packet will be sent quickly, the acknowledgement is piggybacked onto it. Otherwise, if no new packet will be sent by the end of this time period, the data link layer just sends a separate acknowledgement frame.

Real sliding windows:

- The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send.
  - These frames are said to fall within the sending window.
  - Similarly, the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept.
- Since frames currently within the sender's window may ultimately be lost or damaged in transit, the sender must keep all of these frames in its memory for possible retransmission. Thus, if the maximum window size is n, the sender needs n buffers to hold the unacknowledged frames

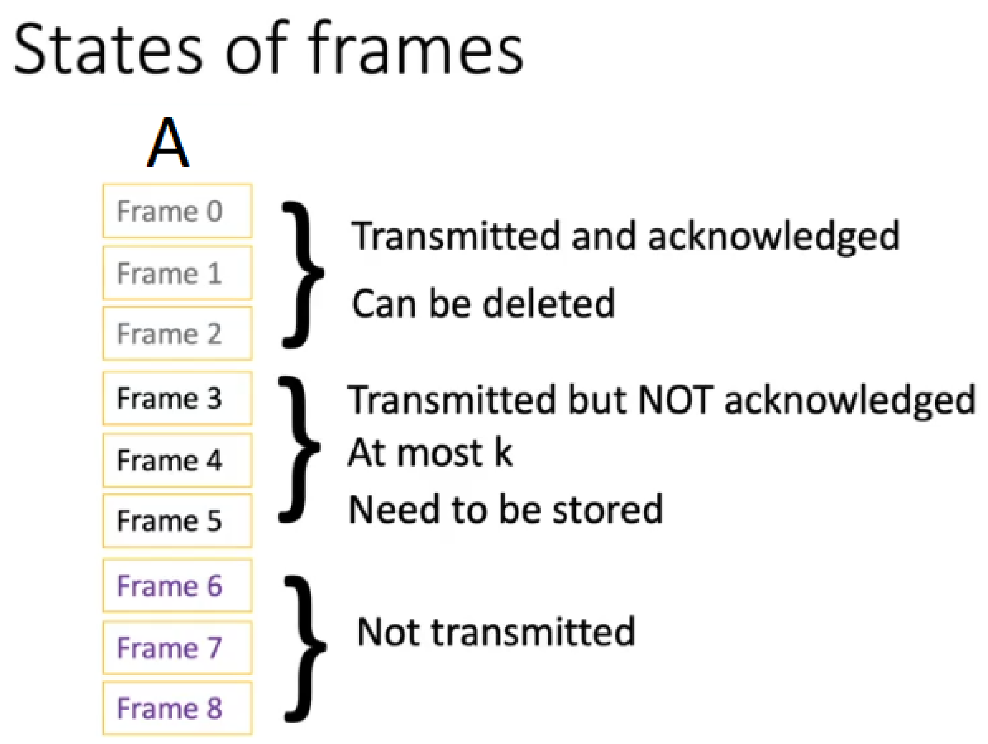


Sequence numbers

How does A know which frame the ack is for?

Number each frame

Include number in corresponding ack



States of frames

Transmitted and acknowledged

Can be deleted

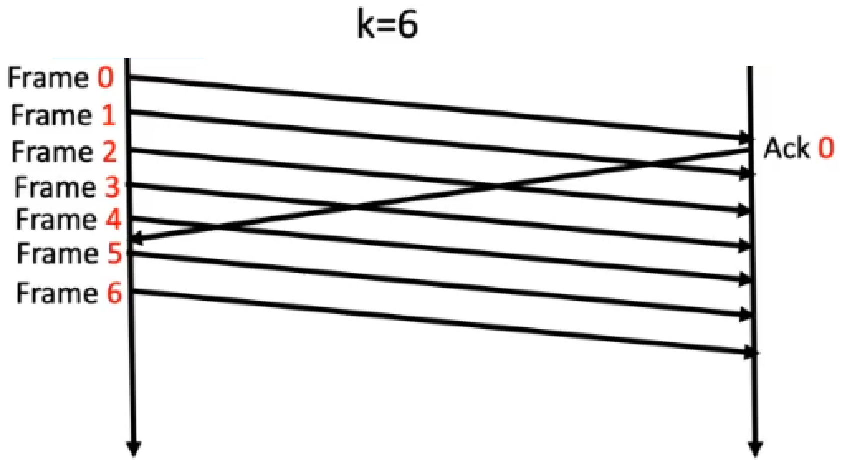
Transmitted but NOT acknowledged

At most k

Need to be stored

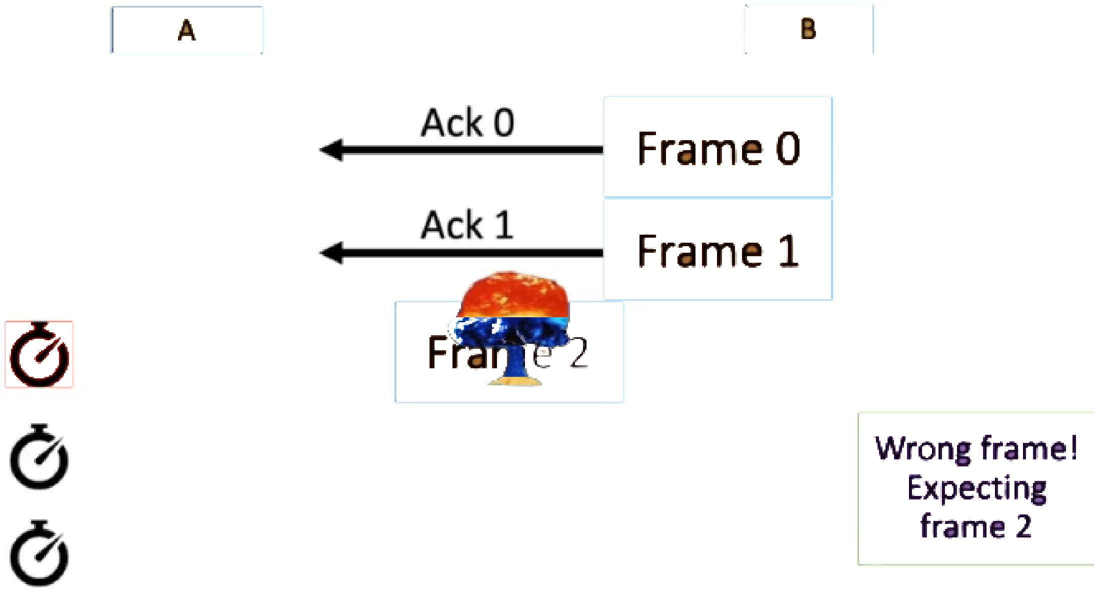
Not transmitted

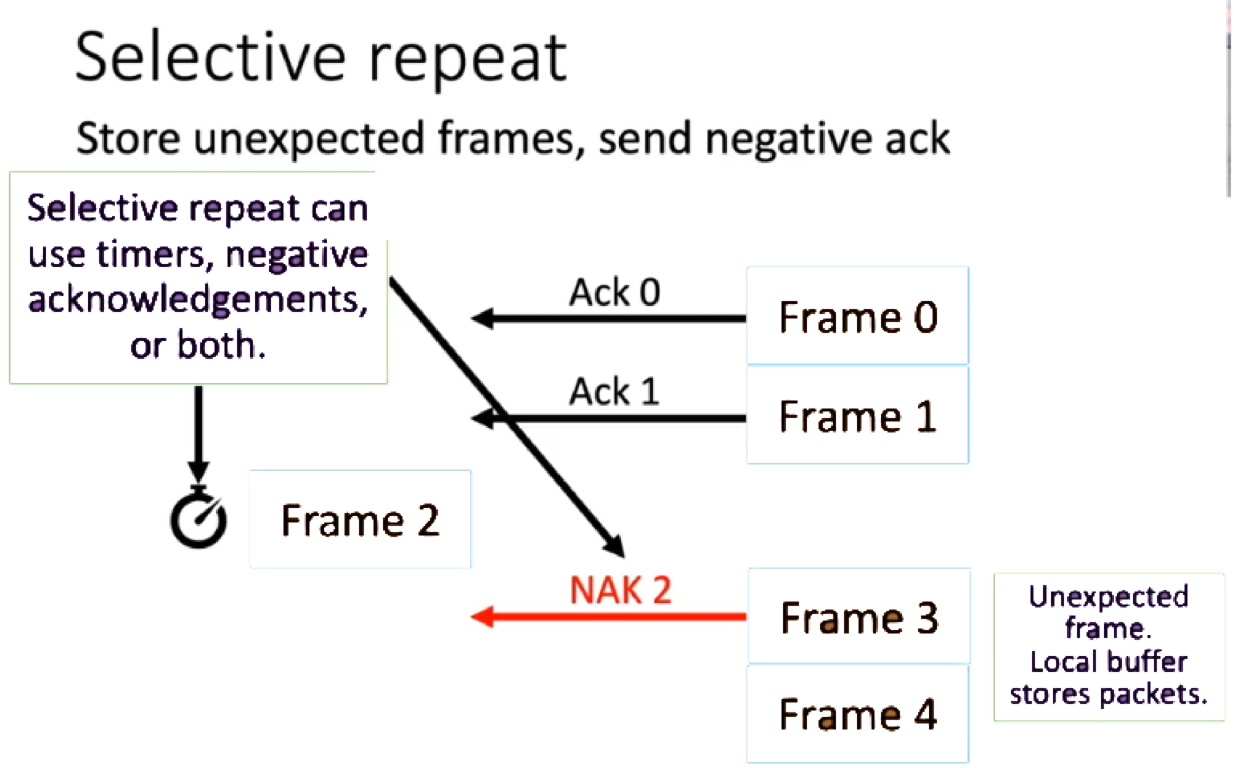
Impact of k without errors



Go-back-N

Discard frames with unexpected sequence numbers





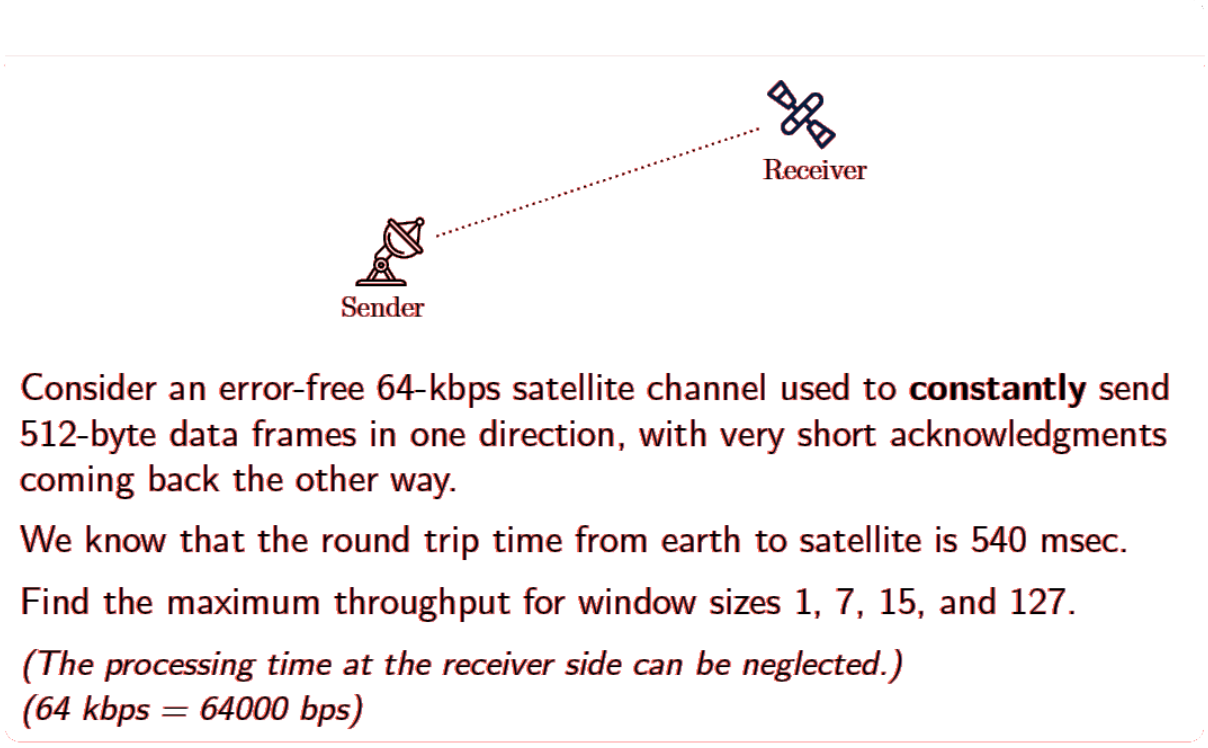
Selective repeat

Store unexpected frames, send negative ack

Selective repeat can use timers, negative acknowledgements, or both.

Unexpected frame. Local buffer stores packets.

Example



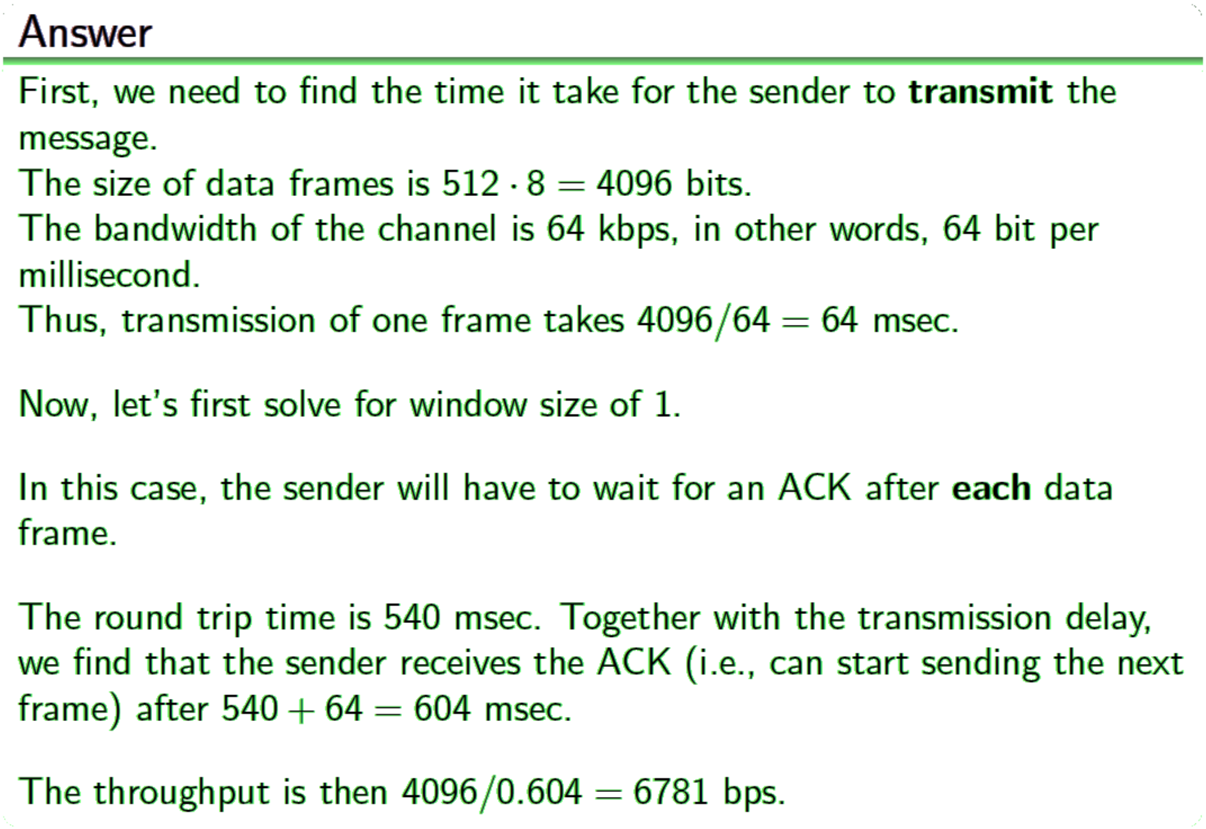
Receiver
Sender

Consider an error-free 64-kbps satellite channel used to **constantly** send 512-byte data frames in one direction, with very short acknowledgments coming back the other way.

We know that the round trip time from earth to satellite is 540 msec.

Find the maximum throughput for window sizes 1, 7, 15, and 127.

(The processing time at the receiver side can be neglected.)
(64 kbps = 64000 bps)



Answer

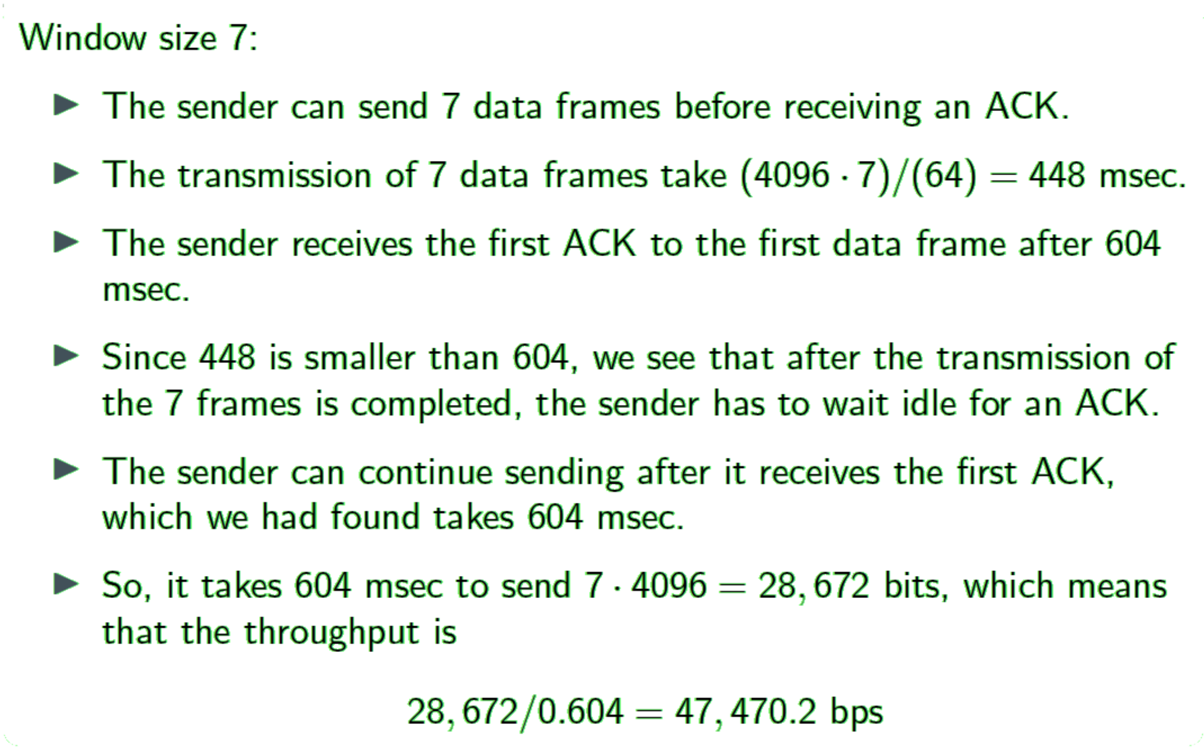
First, we need to find the time it take for the sender to **transmit** the message.
The size of data frames is $512 \cdot 8 = 4096$ bits.
The bandwidth of the channel is 64 kbps, in other words, 64 bit per millisecond.
Thus, transmission of one frame takes $4096/64 = 64$ msec.

Now, let's first solve for window size of 1.

In this case, the sender will have to wait for an ACK after **each** data frame.

The round trip time is 540 msec. Together with the transmission delay, we find that the sender receives the ACK (i.e., can start sending the next frame) after $540 + 64 = 604$ msec.

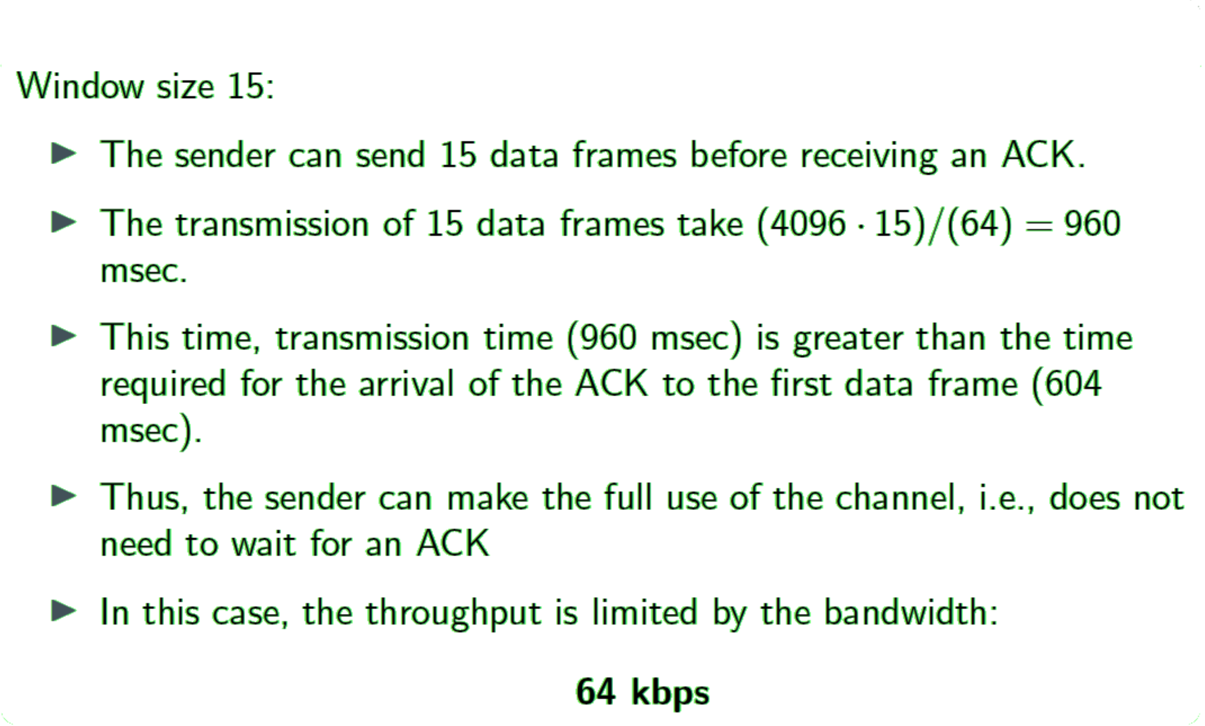
The throughput is then $4096/0.604 = 6781$ bps.



Window size 7:

- The sender can send 7 data frames before receiving an ACK.
- The transmission of 7 data frames take $(4096 \cdot 7)/(64) = 448$ msec.
- The sender receives the first ACK to the first data frame after 604 msec.
- Since 448 is smaller than 604, we see that after the transmission of the 7 frames is completed, the sender has to wait idle for an ACK.
- The sender can continue sending after it receives the first ACK, which we had found takes 604 msec.
- So, it takes 604 msec to send $7 \cdot 4096 = 28,672$ bits, which means that the throughput is

$$28,672/0.604 = 47,470.2 \text{ bps}$$



Window size 15:

- The sender can send 15 data frames before receiving an ACK.
- The transmission of 15 data frames take $(4096 \cdot 15)/(64) = 960$ msec.
- This time, transmission time (960 msec) is greater than the time required for the arrival of the ACK to the first data frame (604 msec).
- Thus, the sender can make the full use of the channel, i.e., does not need to wait for an ACK
- In this case, the throughput is limited by the bandwidth:

64 kbps

MAC Sublayer

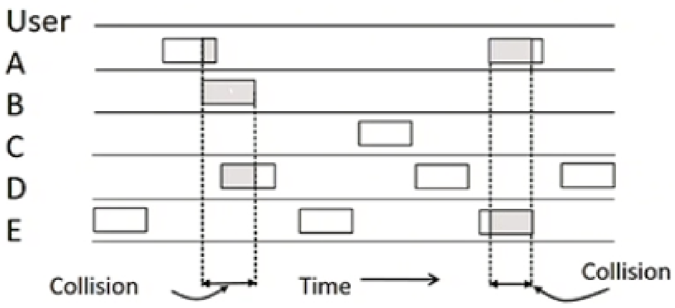
- The main purpose of the MAC Sublayer is to allow multiple parties to share a medium or a small local network.

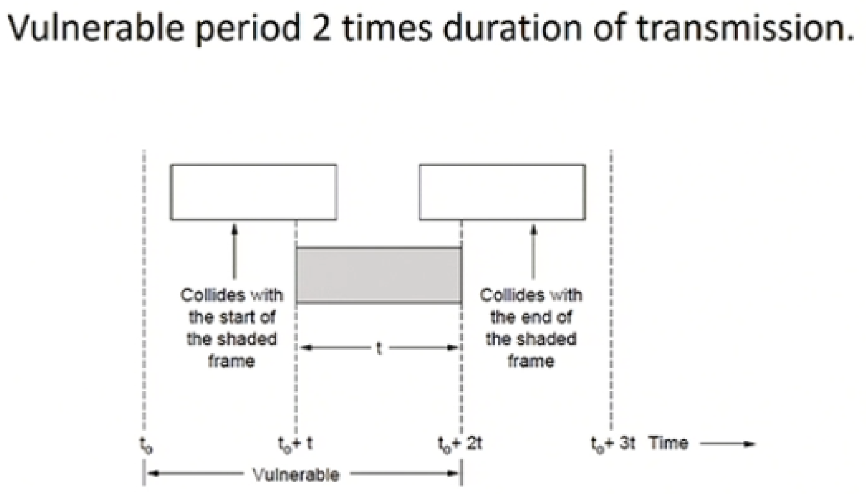
Channel allocation protocols

- protocols run on the sender side that determine when it’s the sender’s turn to use a shared medium.

ALOHA

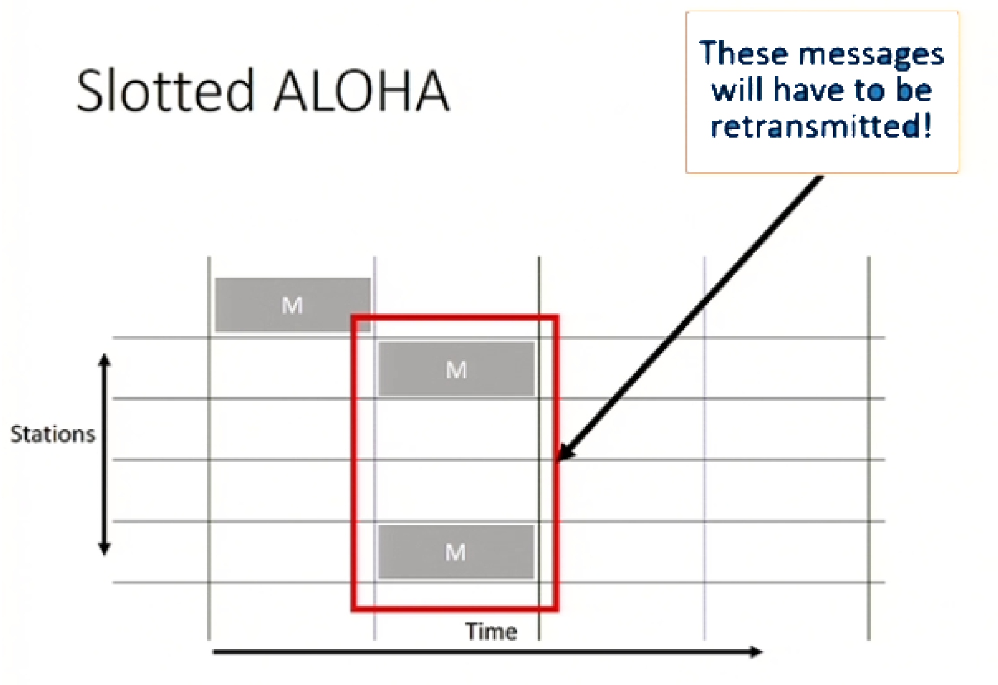
- Users transmit frames whenever they have data
- if collision occurs, retry after a random delay
- does not scale well





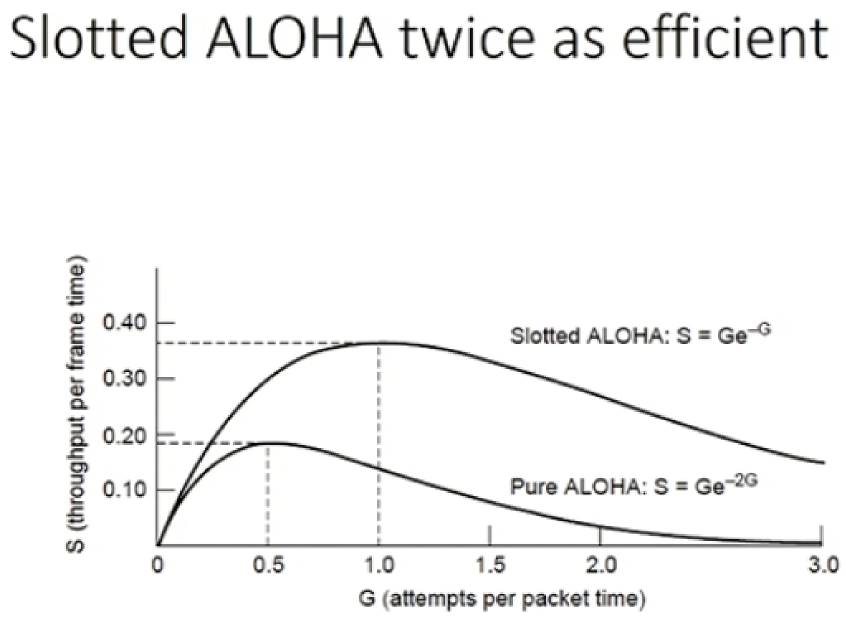
Vulnerable period 2 times duration of transmission.

Collides with the start of the shaded frame
Collides with the end of the shaded frame
t
t_0
t_0+ t
t_0+ 2t
t_0+ 3t
Time
Vulnerable



Slotted ALOHA

These messages will have to be retransmitted!



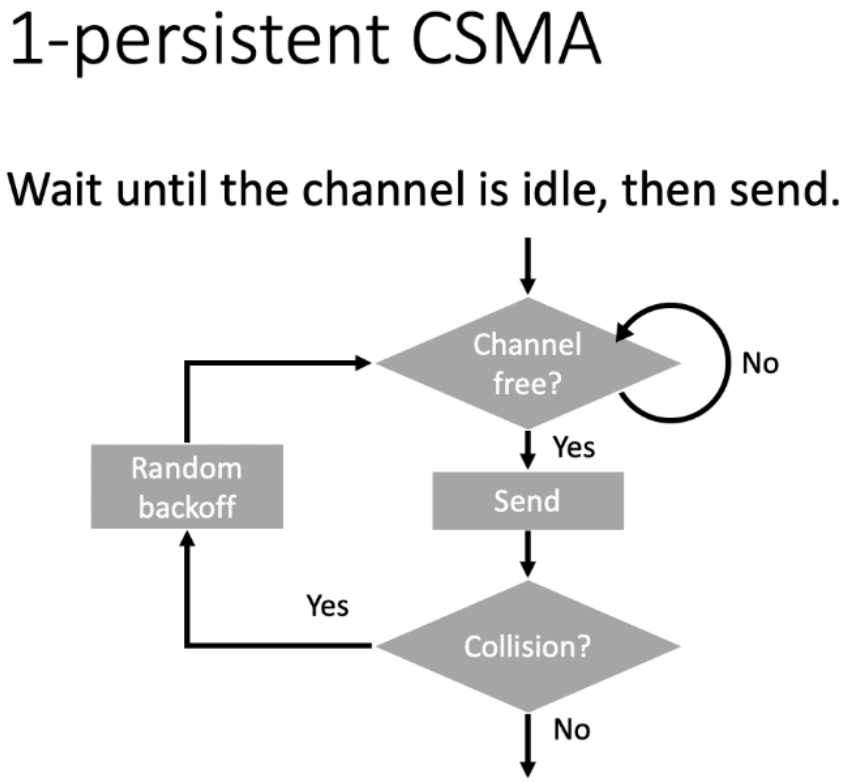
Slotted ALOHA twice as efficient

Carrier-Sense Multiple Access (CSMA)

- Senders can detect (sense) if the channel is in use
    - 1-persisten: wait for idle, then send

  - Nonpersistent: if busy, wait random amount of time and try again
  - p-persistent: if busy, wait for next slot, if idle, send with probability p

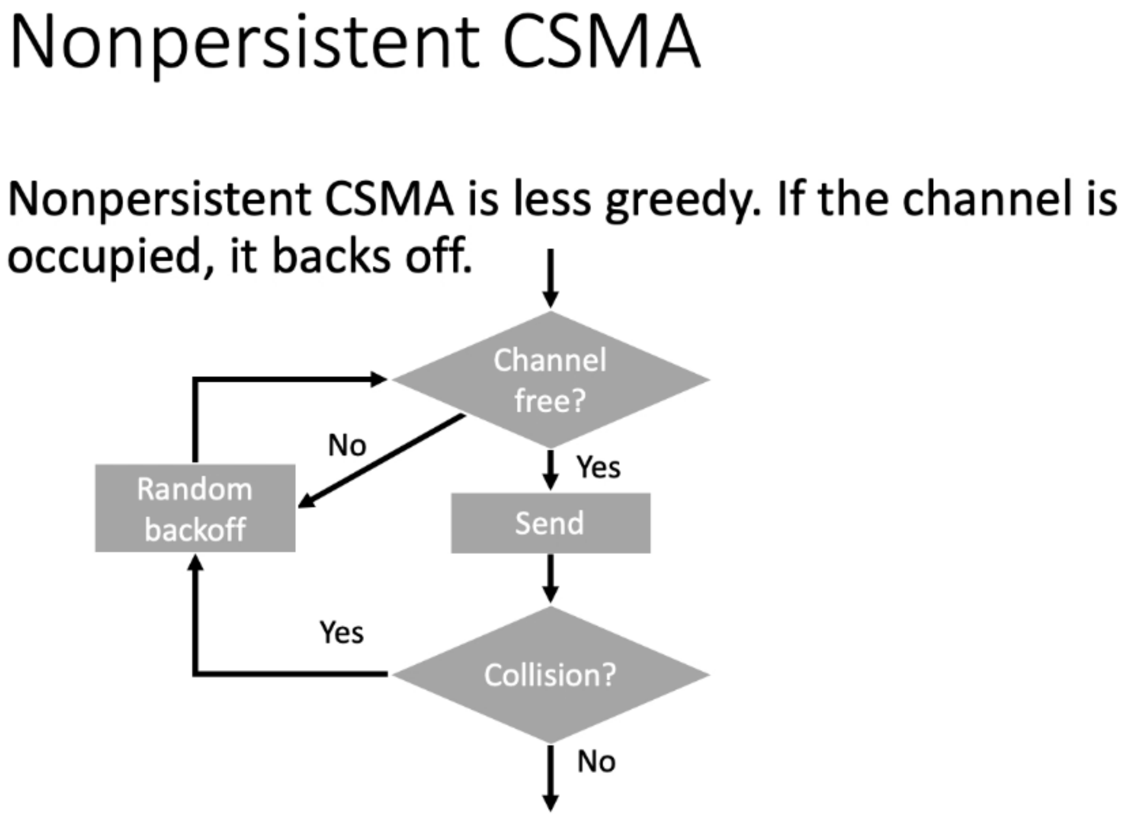
1-persistent


1-persistent CSMA

Wait until the channel is idle, then send.

- problem: if parties send it at the exact same time, they will resend at same time and have collision again.

Nonpersistent CSMA


Nonpersistent CSMA

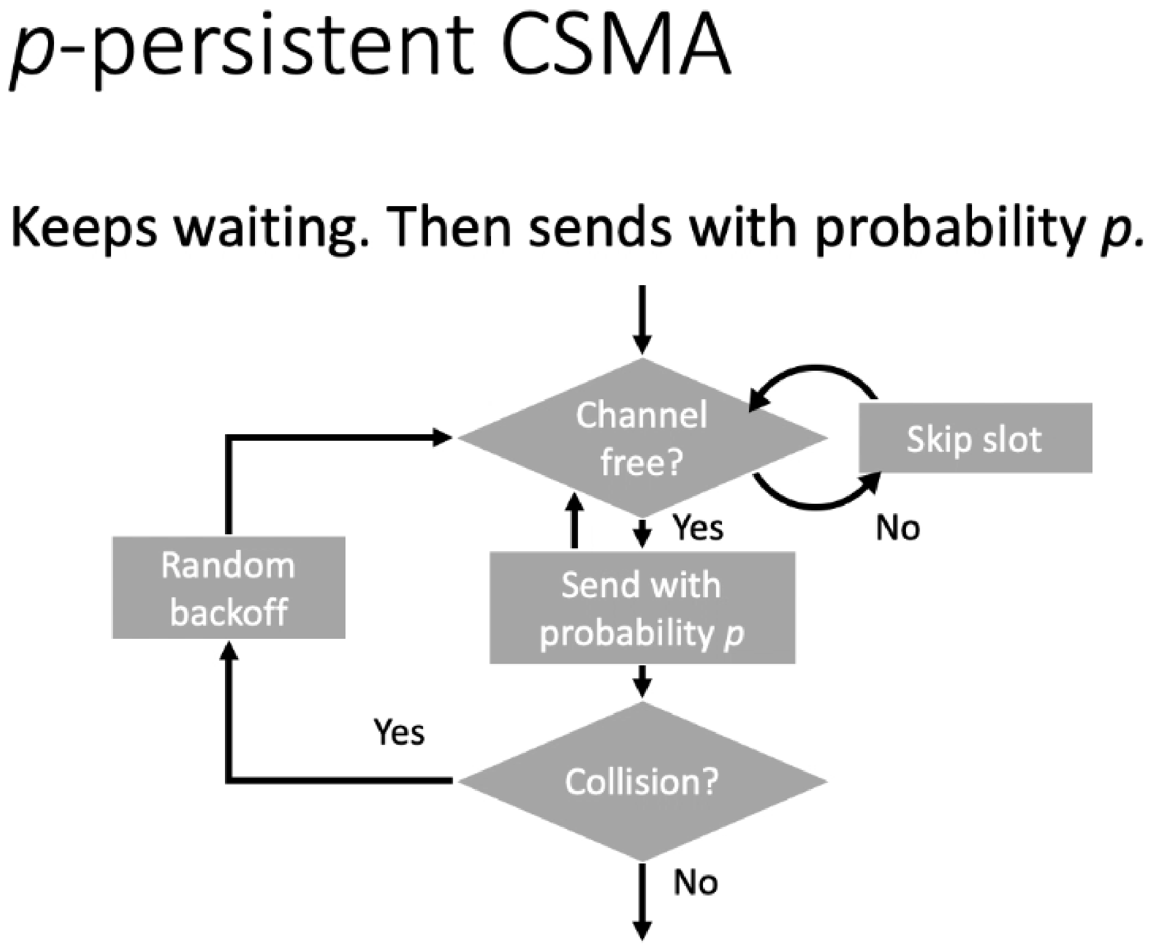
Nonpersistent CSMA is less greedy. If the channel is occupied, it backs off.

- instead of immedeatly resending, it randomly waits before sending again.

- problem: the channel is unitilized often

P-persistent

- compromise between 1-persistent and nonpersistent
- the higher the p, the closer it gets to 1-persistent (higher probability of collisions)

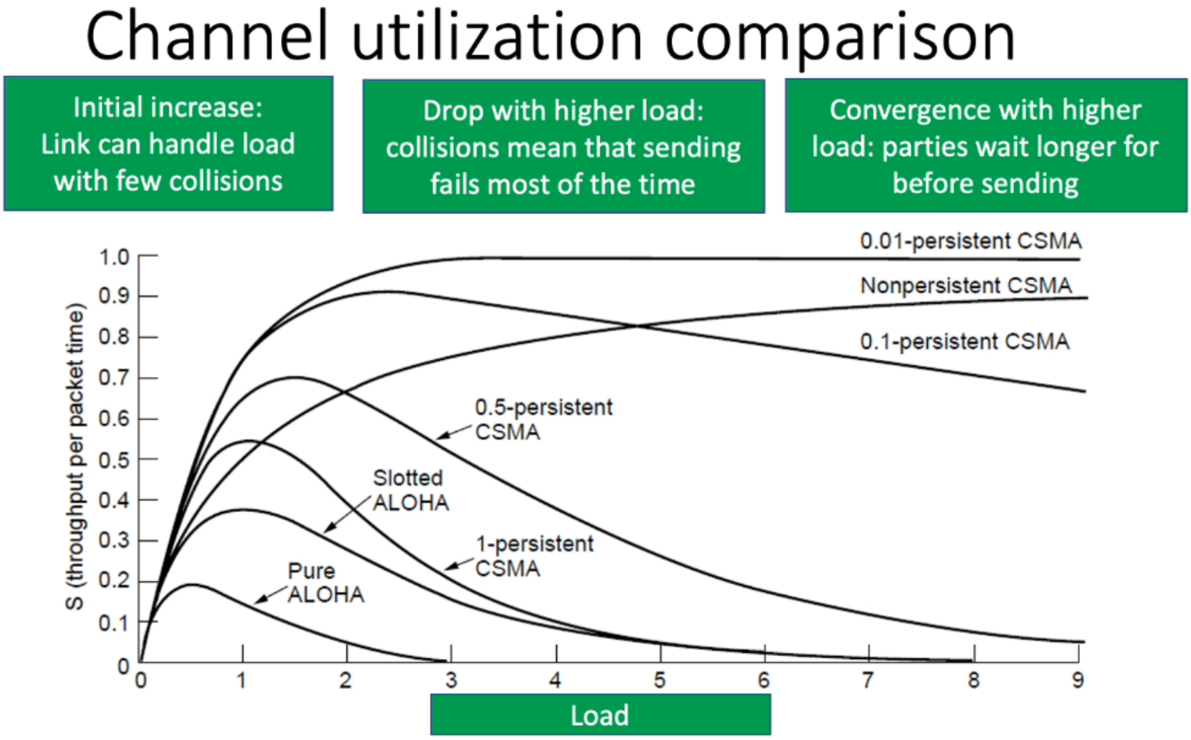


p-persistent CSMA

Keeps waiting. Then sends with probability *p*.

Channel utilization comparision

- As the load increases the utilization of the channel decreases due to an increased collision probability (poisson distributed)



Channel utilization comparison

Initial increase: Link can handle load with few collisions

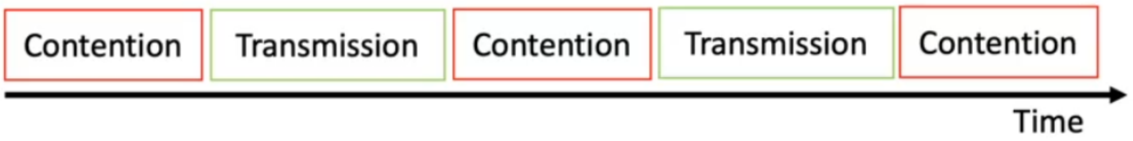
Drop with higher load: collisions mean that sending fails most of the time

Convergence with higher load: parties wait longer for before sending

- Being greedy gives good performance under low load
- Being generous gives good performance under high load

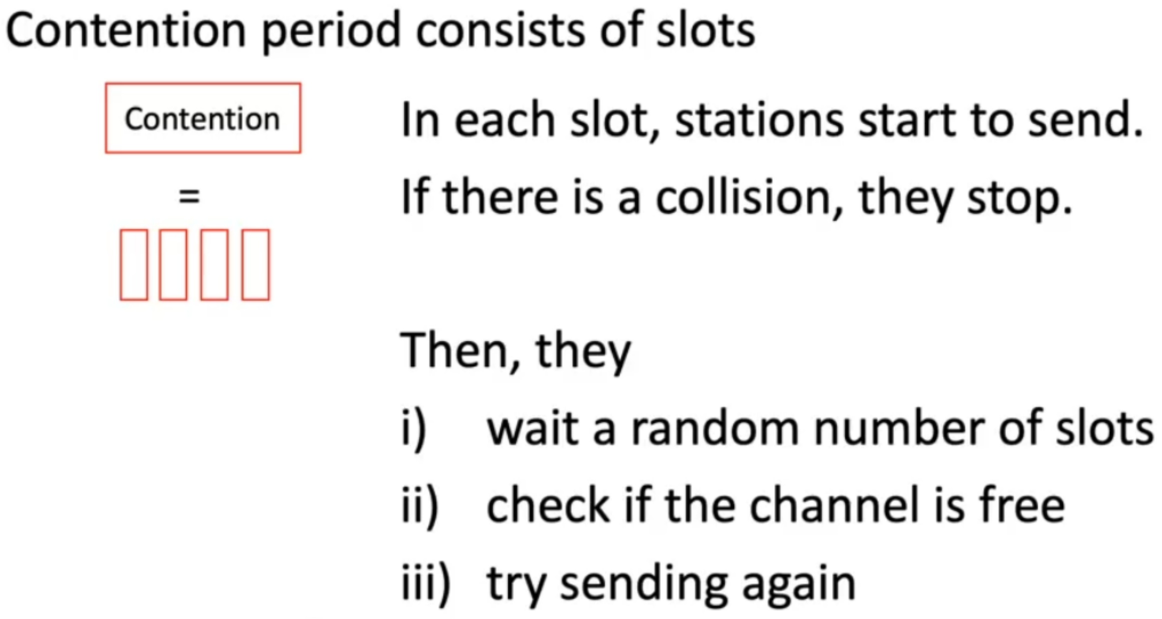
CSMA with Collision Detection (cable)

- The other CSMA do of course know when a frame has been collided (carrier sense is the ability to detect if a medium is being used).
- But Collision detection aims to detect a collision while the medium is being used, not just after an acknoweldge timeout, so that we can abort the transmission and saves time and bandwidth.
- To do so the devices must be able to send and listen at the same time. With Ethernet is possible but with Wifi it is not, as sending is 1M times stronger than listening frequencies, hence interfering with what it could potentially be heard.
- we can think of CSMA/CD contention as a slotted ALOHA system (for the listening part) with a slot width of 2 delta (delta being the propagation delay from one device to another).



Contention period: Decide who sends next

Transmission period: Party (called *station*) who has been selected in contention period sends frame



Contention period consists of slots

In each slot, stations start to send.
If there is a collision, they stop.

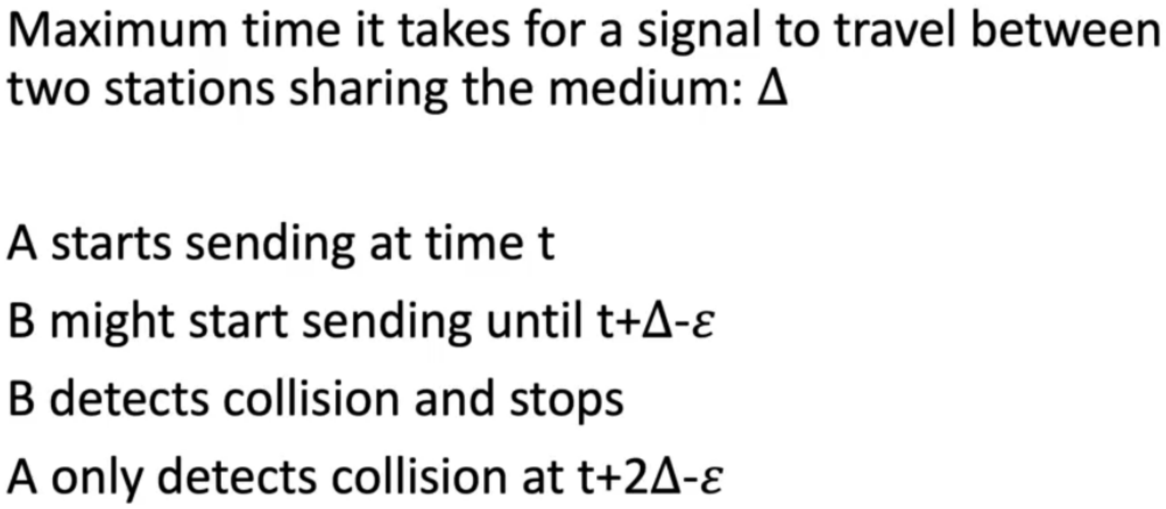
Then, they

i) wait a random number of slots

ii) check if the channel is free

iii) try sending again

- Transmission starts when there is only one party sending



Maximum time it takes for a signal to travel between two stations sharing the medium: Δ

A starts sending at time t

B might start sending until t+Δ-ε

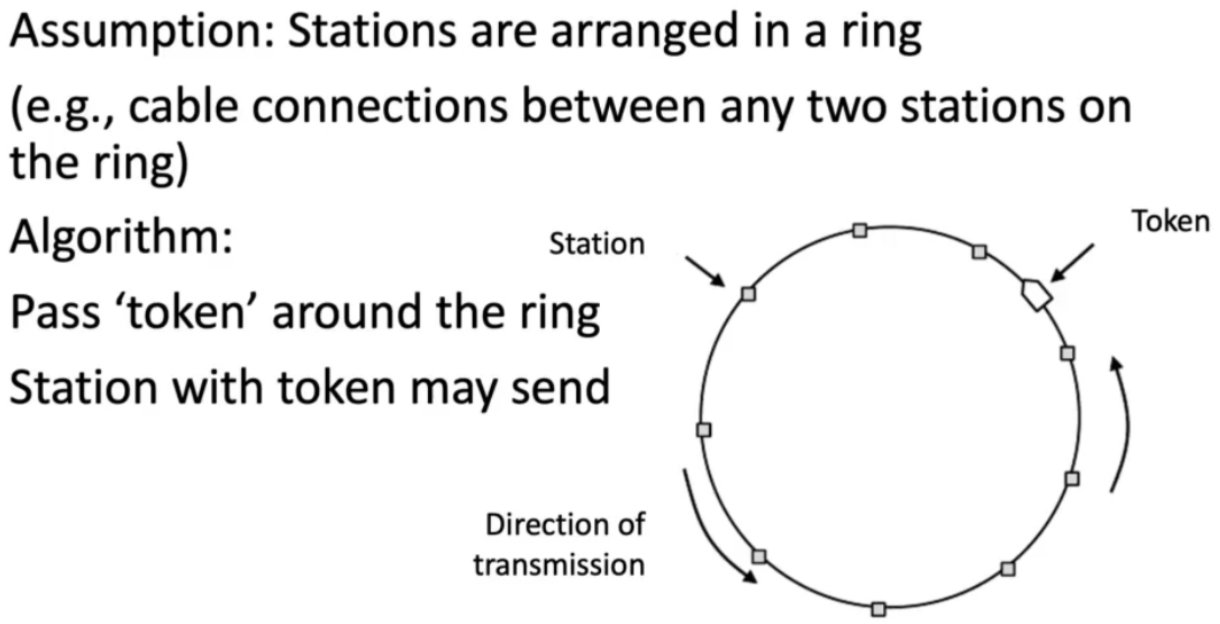
B detects collision and stops

A only detects collision at t+2Δ-ε

Collision-free protocols

- Usueful in very controlled environments, but not for general environments.

Token ring



Assumption: Stations are arranged in a ring

(e.g., cable connections between any two stations on the ring)

Algorithm:

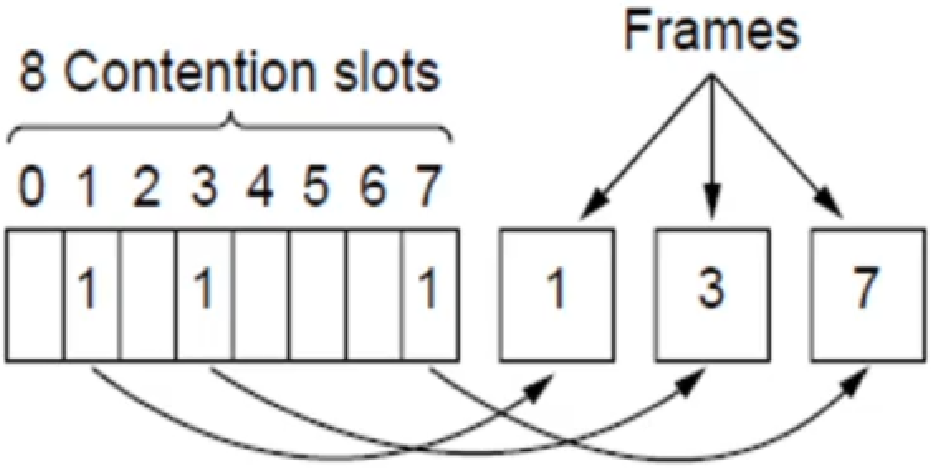
Pass 'token' around the ring

Station with token may send

- You can add new stations into the ring

Bit-map protocol

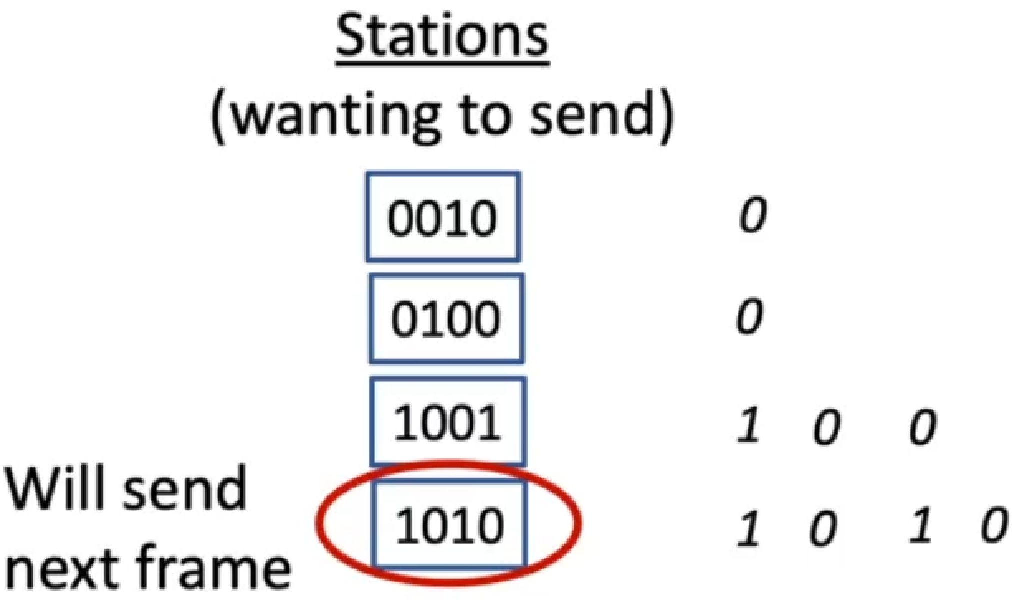
- Assumption: Static and known stations with common notion of slots
- Contention slots are N
- Algorighm: Enumerate stations
  - In contention period, each station is assigned a slot to transmite the bit 1 if they want to send something
  - In tramsission period, all stations that indicated they want to send, send one frame.
  - The order in which the frames are sent is already hardcoded in the algorithm for each station.



Binary countdown

- Assumption: Static and known stations with common notion of slots
- Contention slots are ceil(log N)
- Algorighm: Stations have distinct priorities
  - Collision results to binary OR of bits sent at the same time
  - Stations express their priority as binary numbers

  - When a station wants to send, it send bits of priority starting at the most significant bit (left)
  - If 1 is received when 0 was sent, then the station stops and waits for new round
  - The one with the highest priority will send the next frame



Stations (wanting to send)

0010 0

0100 0

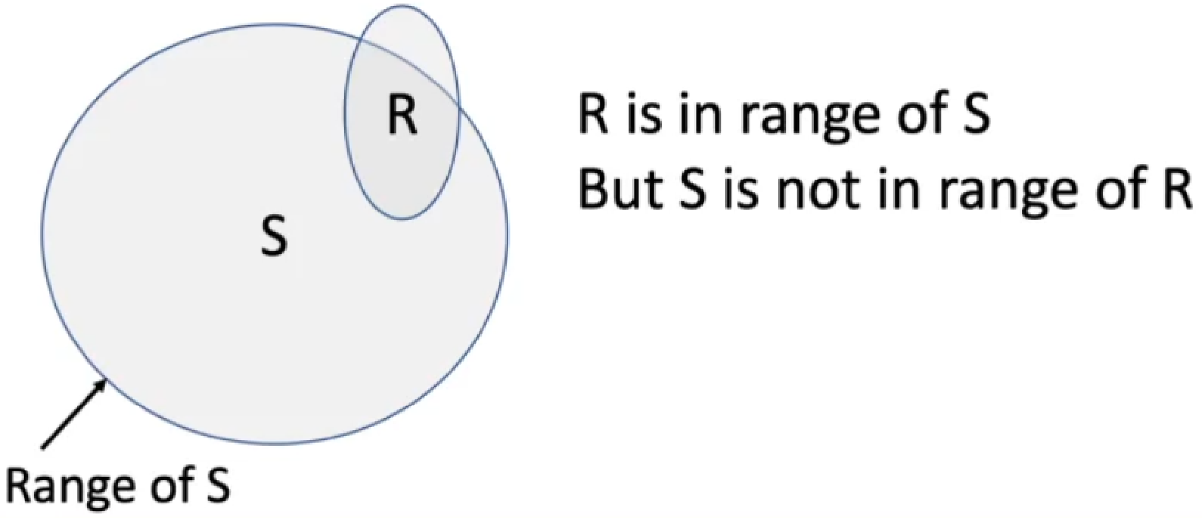
1001 1 0 0

1010 1 0 1 0

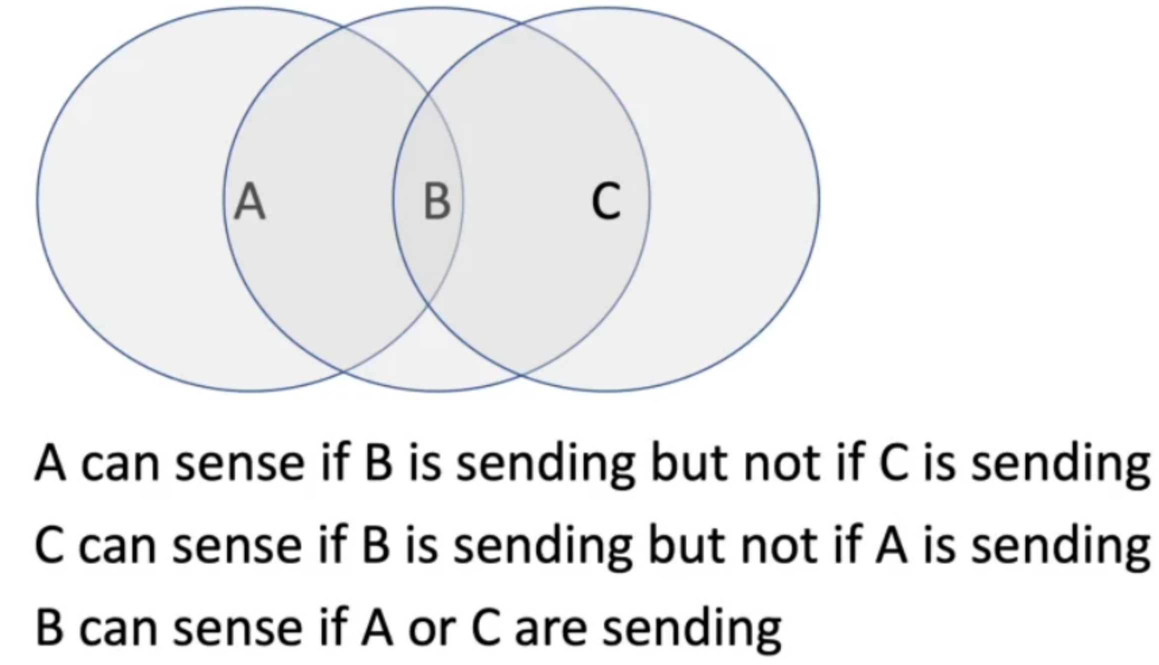
Will send next frame

Multiple access with collision avoidance (MACA) (wireless)

- Having wireless channels removes the need for wires (which is good), but it does make it harder to detect collisions (which is bad).
- MACA stands for multiple access with collision avoidance (as opposed to collision-free, here there are no guarantees)
- Terminals (stations for wireless networks) have a range. Their transmissions are only received by parties within the range.



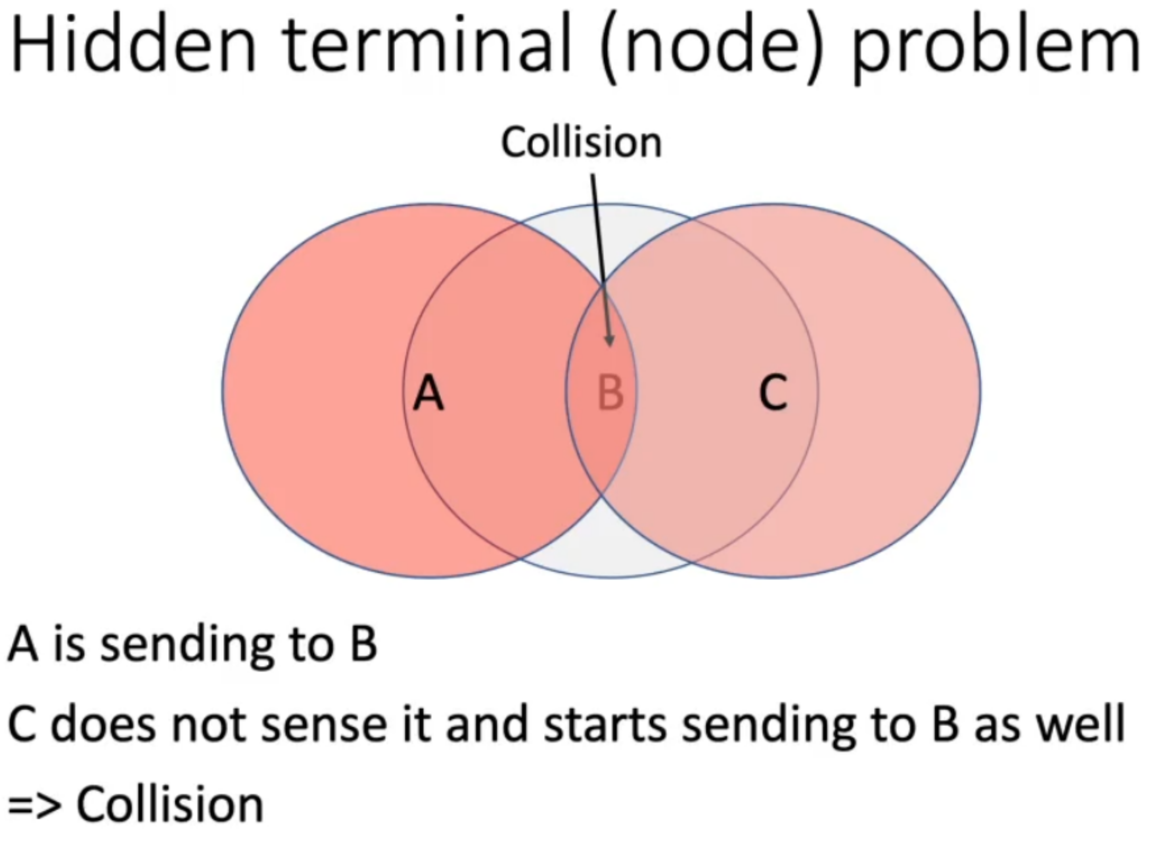
R is in range of S
But S is not in range of R



A can sense if B is sending but not if C is sending

C can sense if B is sending but not if A is sending

B can sense if A or C are sending

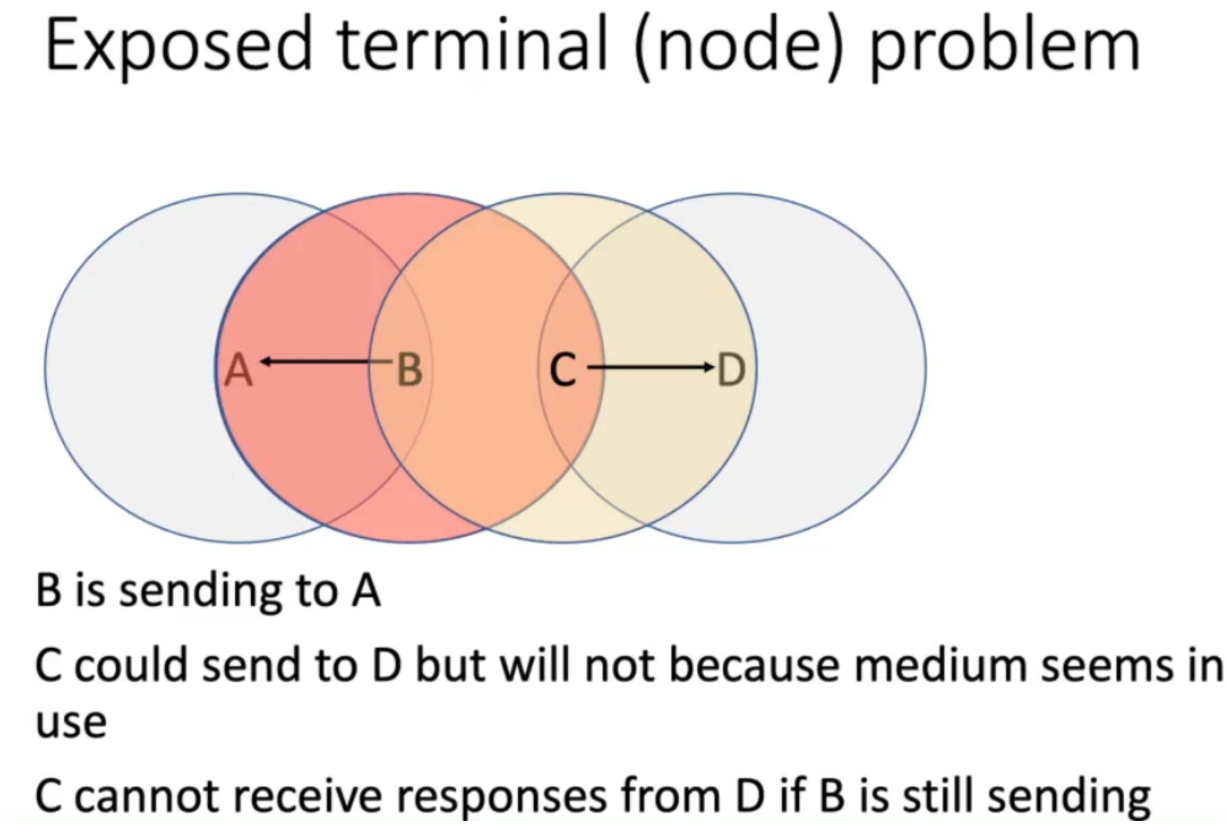


Hidden terminal (node) problem

A is sending to B

C does not sense it and starts sending to B as well

=> Collision



Exposed terminal (node) problem

B is sending to A

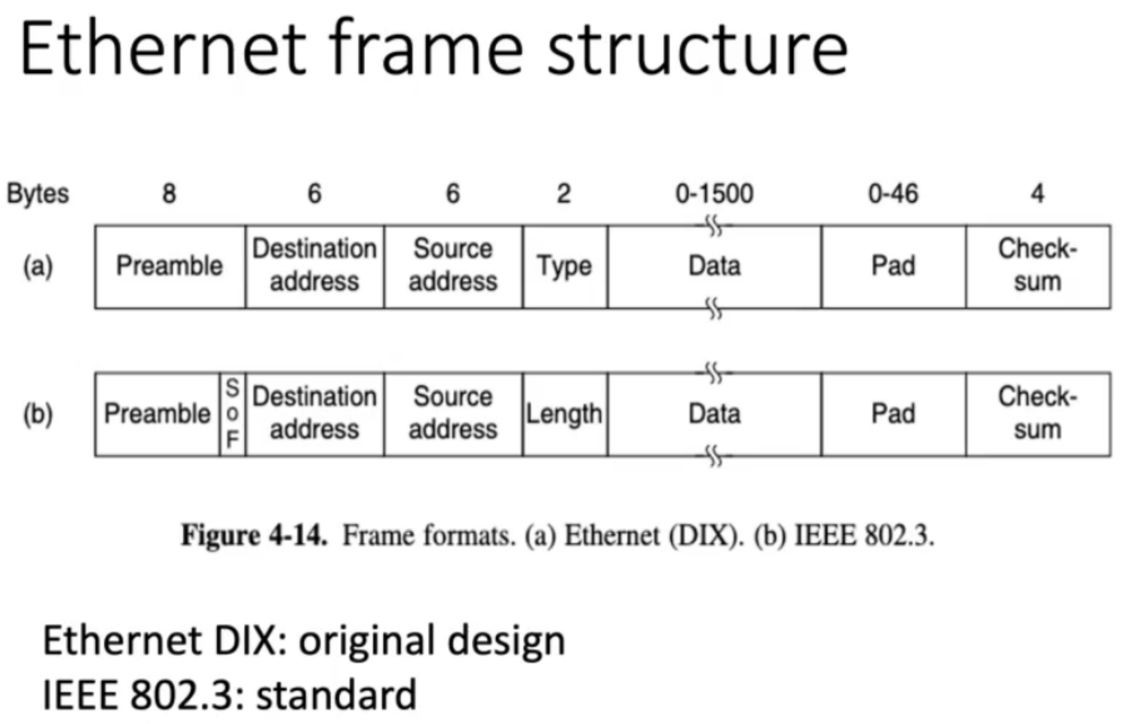
C could send to D but will not because medium seems in use

C cannot receive responses from D if B is still sending

- MACA mitigates the hidden node problem and exposed node problem through two extra messages (request to send and clear to send)
- Protcol
  - Request-to-send (RTS):
    - Indicate to terminal that you want to send and include sender (you) and receiver MAC address
  - Clear-to-send (CTS):
    - Indicate receiver that the terminal is free to receive frames (if it is not receiving anything else)
    - Include address of receiver
  - Sender sends frame to receiver
  - Receiver sends acknowledge back
  - Terminals in the range of the CTS will behave and stop sending anything until A sends an acknowledge or a timeout
  - A will back off before sending another RTS if it does not receive CTS
- In the event that 2 terminals RTS a third terminal at the same time, this one doesnt receive a proper RTS due to interference. Each of them will try again at randomly different times.
- The exposed terminal node problem is solved only when the RTS and CTS happen simulatenously, when they happen with one step offset the CTS of the second gets interfered with the frame from the first one.

Ethernet

- Used for wired internet
- Uses 1-persistent CSMA/CD because there is not too many parties using the medium.
  - The random backoff is binary: Choose number of slots uniformly at random in {0,1,3,7,...,2^i-1} where i is the number of failed tries to send the rame.



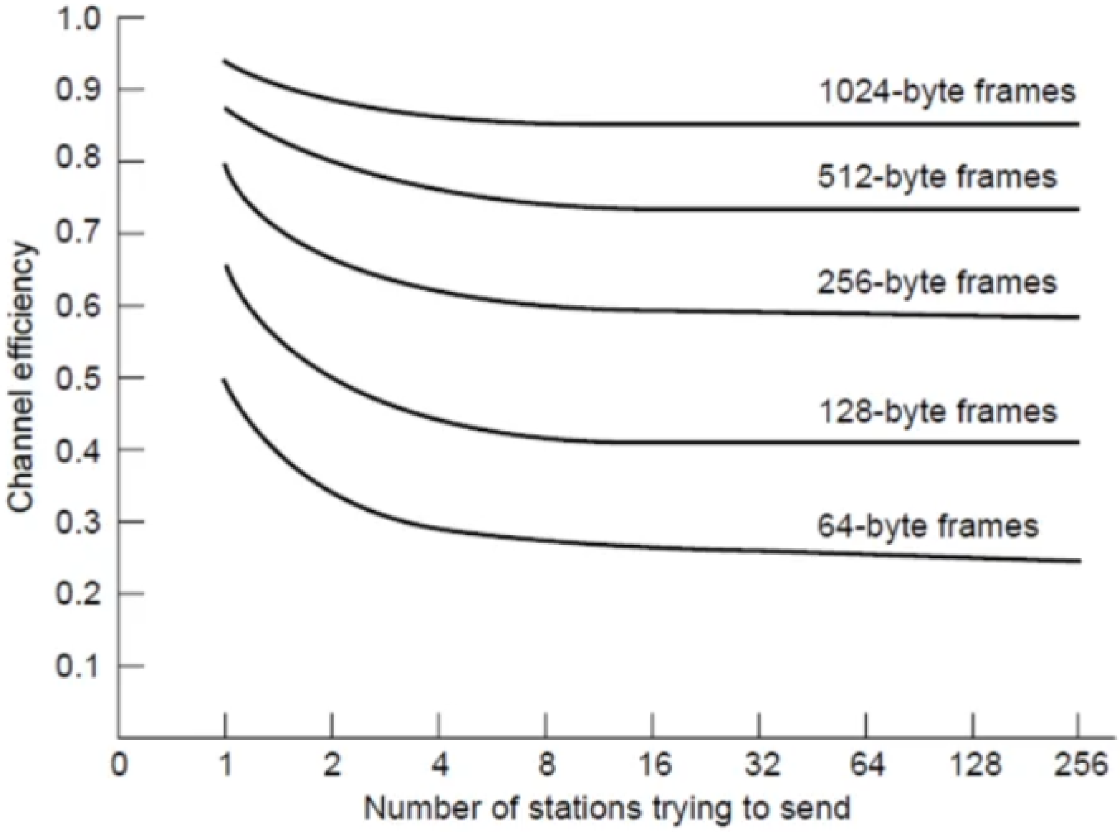
Ethernet frame structure

Bytes	8	6	6	2	0-1500	0-46	4
(a)	Preamble	Destination address	Source address	Type	Data	Pad	Check-sum
(b)	Preamble SOF	Destination address	Source address	Length	Data	Pad	Check-sum

Figure 4-14. Frame formats. (a) Ethernet (DIX). (b) IEEE 802.3.

Ethernet DIX: original design
IEEE 802.3: standard

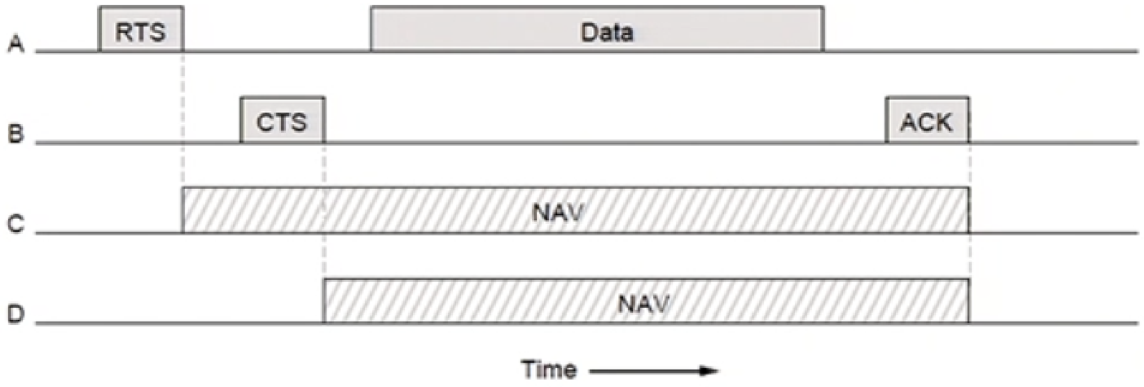
- Header:
  - The preamble (or preabmle + SoF) contains 7 bytes with 1010 1010 and the last one with 1010 1011
  - The destination address and source address are the MAC addresses (basically the hardware unique address of the receiver and the sender)
  - Type contains the upper layer protocol used. Turned out it was not effecient to just have that field.
  - Length contains length of frame in bytes, which allwos the receiver to split the frames without knowing the data
- Payload:
  - Data must have a minimum length of 72B, if not reached its padded automatically.
  - Padding: This is due to the propagation delay (easier when the length has a min size for collision detection calculations). When detecting a collision, station warns other by 48 bit noise burst.
- Trailer:
  - Checksum

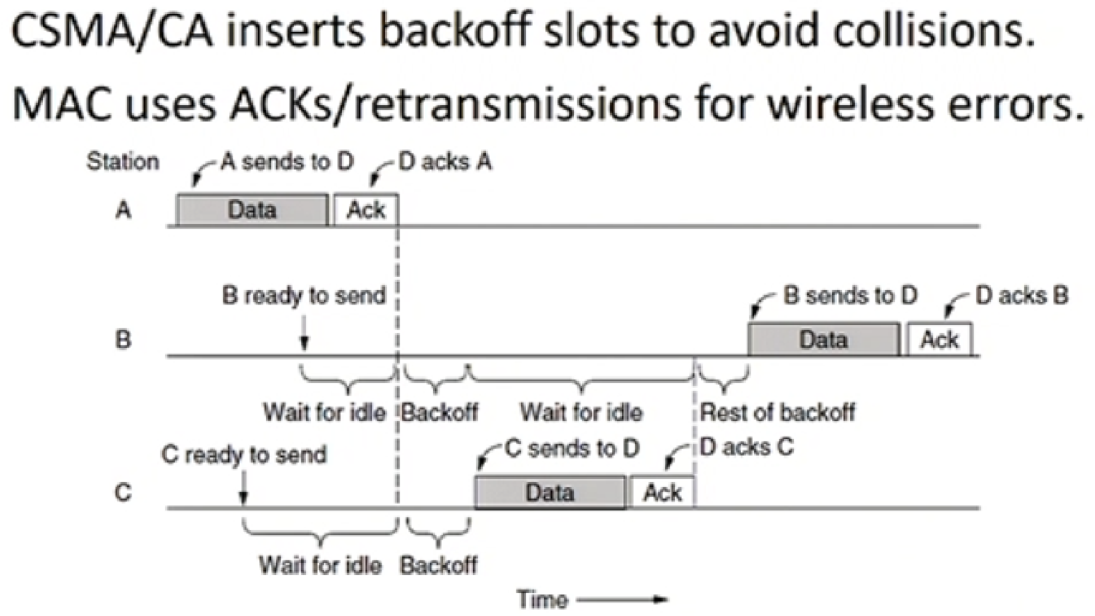


Wifi (802.11 standard)

- A device cannot send and listen to frequencies at the same time, as the act of sending will interfere with what could be listened.
- Devices can only rely on ACKs to determine if collision ocurred
- Wifi uses MACA with carrier sense avoidance: CSMA/CA

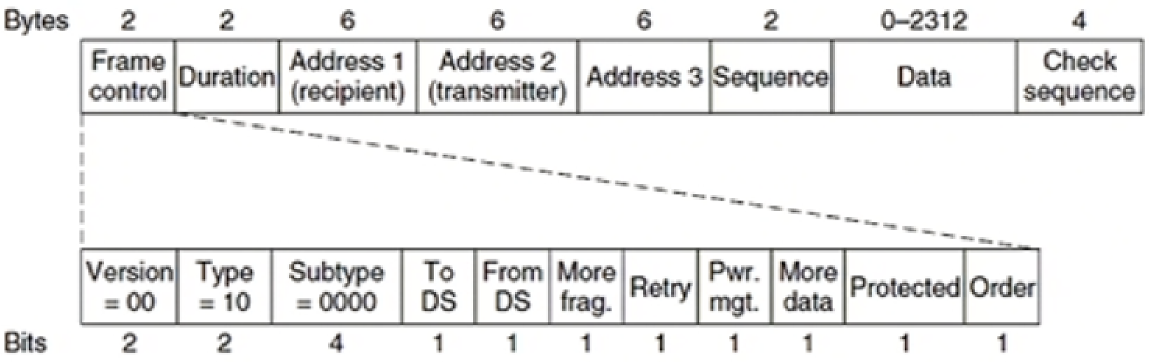
Instead: CSMA/CA, Collision Avoidance.





CSMA/CA inserts backoff slots to avoid collisions.

MAC uses ACKs/retransmissions for wireless errors.

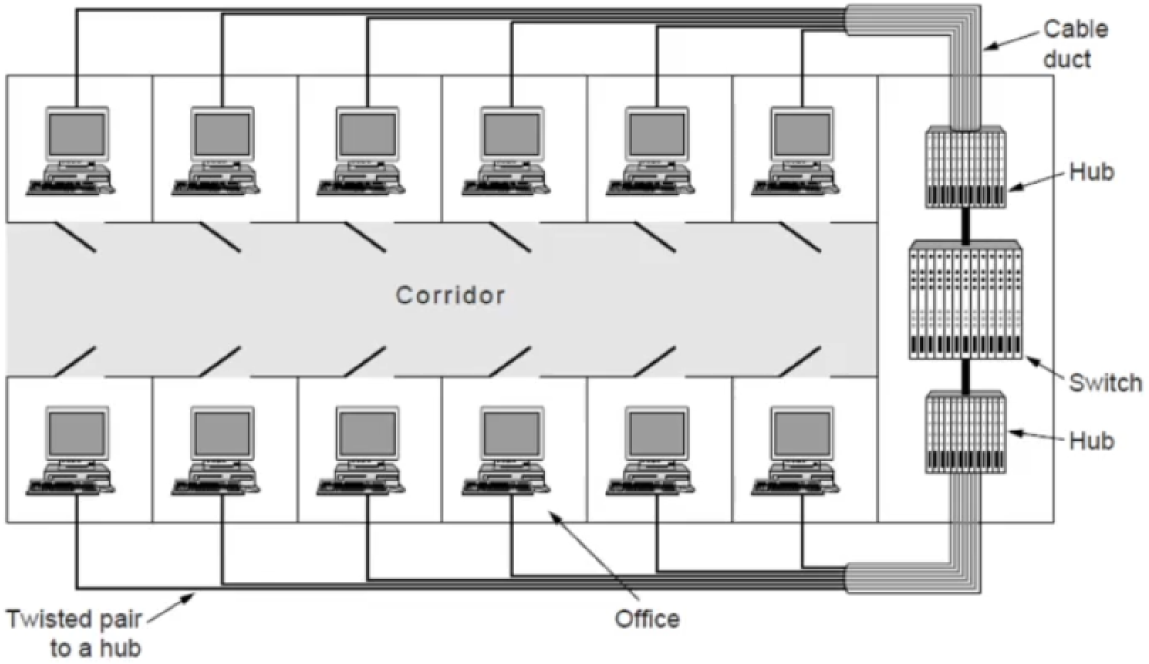


Bytes	Frame control	Duration	Address 1 (recipient)	Address 2 (transmitter)	Address 3	Sequence	Data	Check sequence
	2	2	6	6	6	2	0–2312	4

Bits	Version = 00	Type = 10	Subtype = 0000	To DS	From DS	More frag.	Retry	Pwr. mgt.	More data	Protected	Order
	2	2	4	1	1	1	1	1	1	1	1

LAN

- MAC addresses are used to identify the receiver

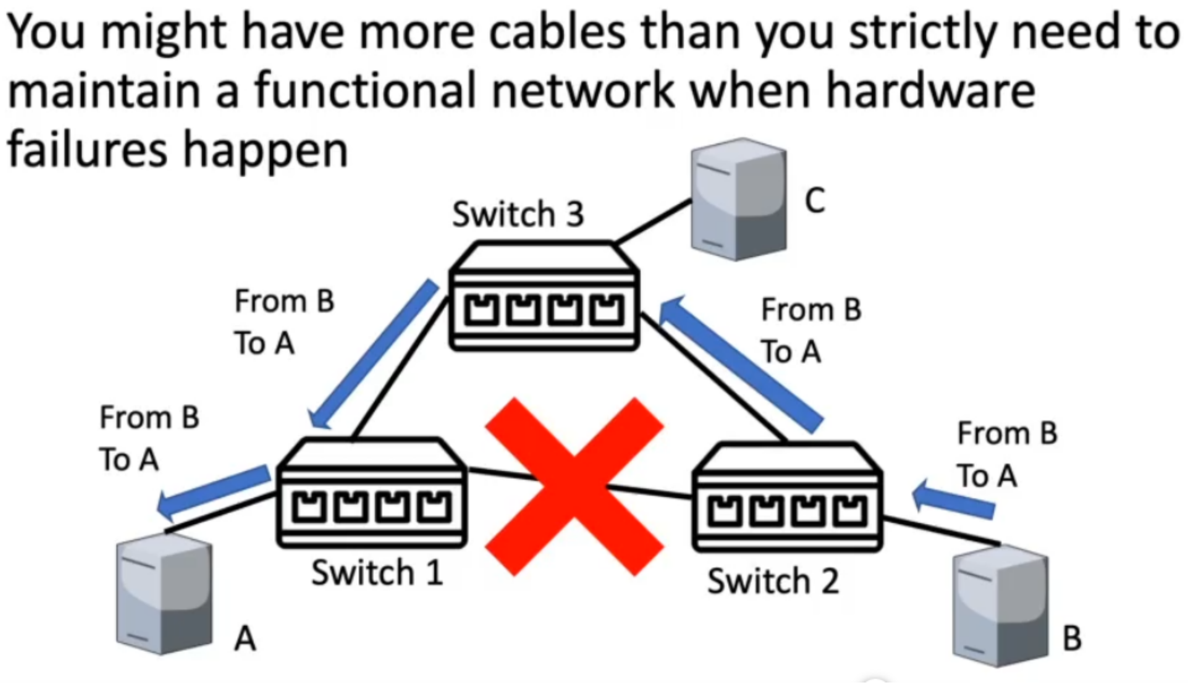


Hub

- Extension of the phyisical layer, its a device that connect multiple Ethernet devices.
- The signal from the input/output ports is extended to all other ports, making all the cables act as one single long cable

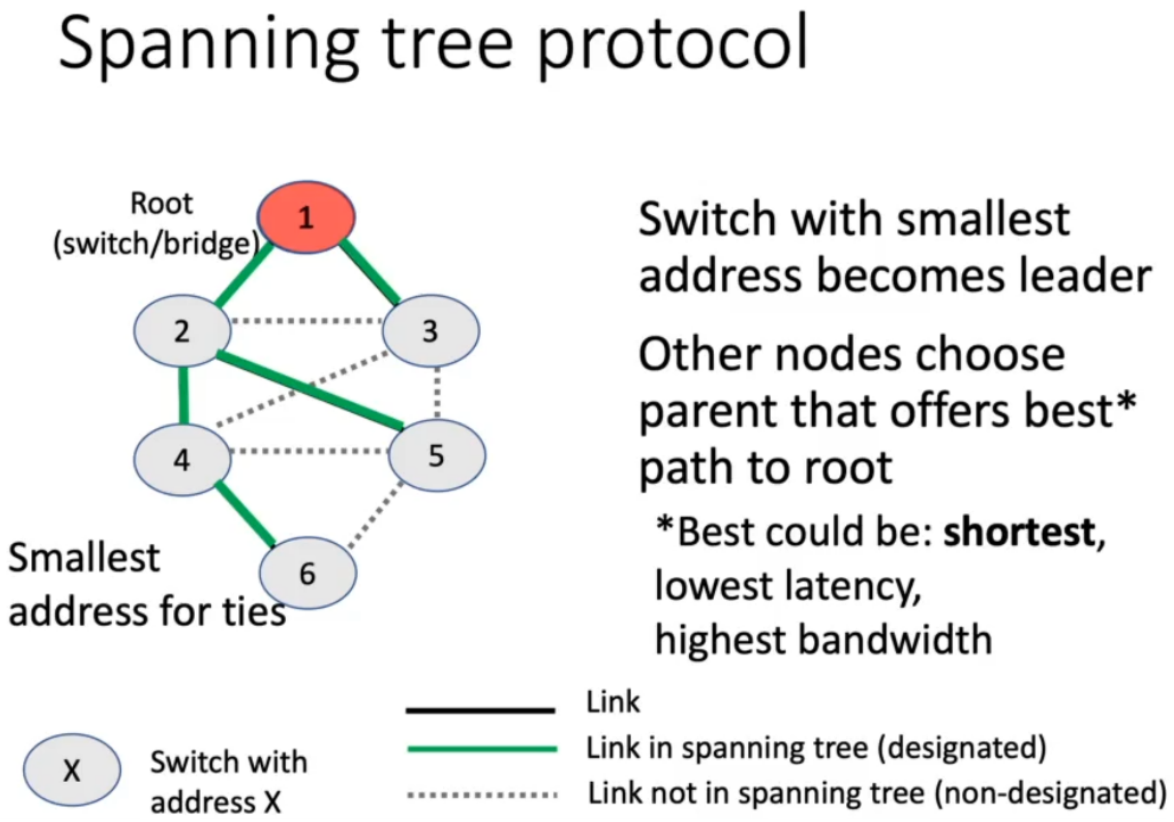
Switch

- As opposed to a hub where the entire signal is forwarded to all the devices, the switch uses mac addresses to forward frames only to the intended destination
  - Use backwards learning data structure
    - Frames have source and destination address
    - Switch keeps a hash table that maps addresses to port numbers
    - First time a device sends a frame the switcher can already assign a port to the sender device
    - Switch broadcasts frame to all other devices
    - When device replies back to the frame the switch can map the responding device
    - Devices move and hence change port. Switcher keeps track of “last seen” time for each MAC address. Periodically remove old addresses (a minute or so).
- This one acts on the datalink layer (rather than on the physical link layer like the hub)



You might have more cables than you strictly need to maintain a functional network when hardware failures happen

- However it can end up on an infinite loop if the receiver is never found. Which would bring the network down
- **Spanning tree**: To fix the infinite loop problem we only use a minimal subset of cables and leave the others as backups



Spanning tree protocol

Switch with smallest address becomes leader

Other nodes choose parent that offers best* path to root

*Best could be: **shortest**, lowest latency, highest bandwidth

━ Link

🟩 Link in spanning tree (designated)

┈ Link not in spanning tree (non-designated)

- If a cable fails, the nodes will look for a different parent, if the parent of a node fails, two things may happen, the child doesnt notice it fast enough, so it keeps the same parent (as the parent managed to fix the connection), or the child re-routes via a different parent that is connected to the route.

Network layer

- Remember that the data link layer operates only when you have a shared channel (i.e. a wave frequency, an ethernet cable)
  - You have to create frames from bits/bytes (head, load, trailer, checksum…)
  - You have to detect/correct transmission errors (CRC…)
  - You have to efficiently use the avaible bandwidth between machines (ALOHA, collision avoidance/detection etc)
- The network layer already assumes that the datalink layer works and members can talk to each other
  - The (routing, transprt) network layer will forward your messages everywhere it can
  - The network layer will manage network congestion
  - The NL is responsible for customer service
  - The NL connects multiple networks

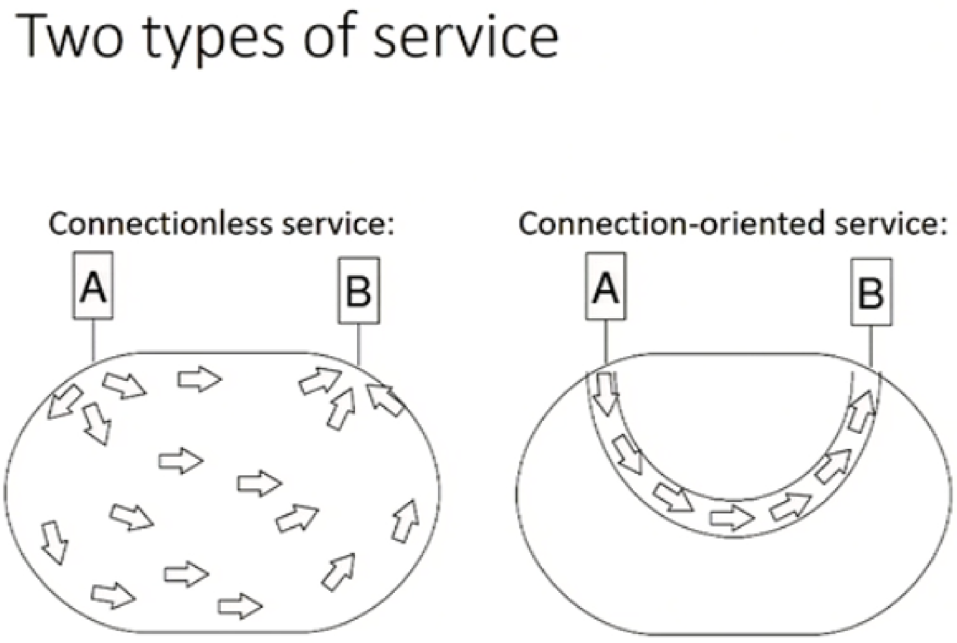
"It takes charge of how the network looks and how to get messages through"

- The network layer is the heart of the networking model, together with the transport layer
- OSI layers below (data link layer, physical layer) do not know about end-to-end delivery (they just put the message in a medium, where everybody is listening)
- OSI layers above do not know about the topology of the network
- Both layers above and below do not know about routing (in theory)

- Only ISP (internet service provider) companies, or companies that sell ISP services, can own an IP address, the rest of users can only rent fixed IP addresses. Dynamic addresses are included by default in any internet subscription.

- The network layer provides unique (IP) addresses to the transport layer
- Internet is linked to politics, i.e. Isreal doesnt exist in Dubai, China firewall google, etc. So there's not an universal routing.

- Nevertheless, the network layer stands for the global network



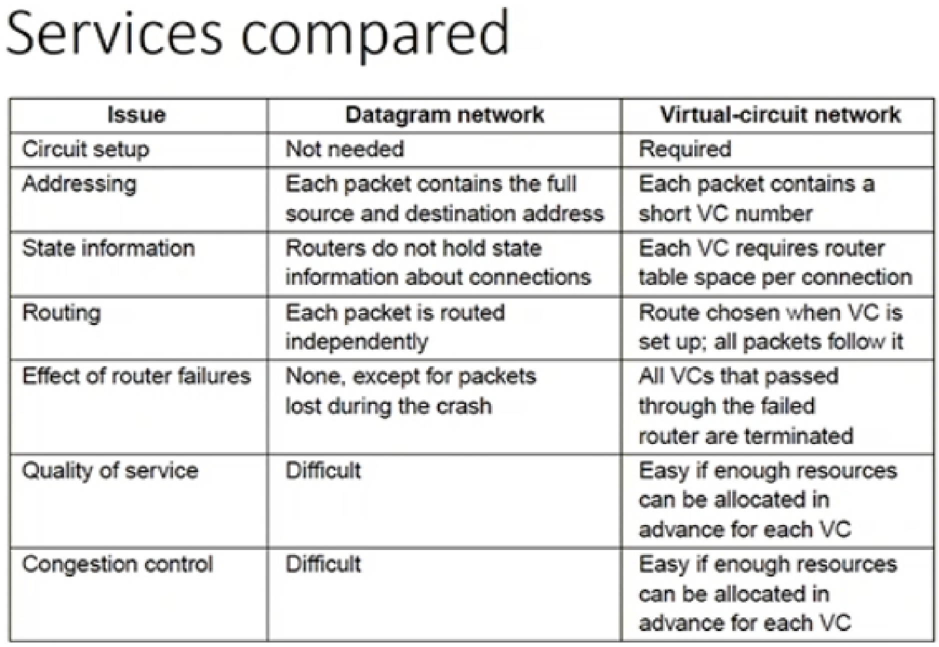
Two types of service

The connection-less model

- Dominant choice
- Routers use algorithms to decide where to send each packet individually every time
  - This model scales better than the telephone network model where 1 big lookup table had to be checked by all the users
  - This model routes each packet and gives it individual attention
- Used by the Internet Protocol (IP)
- Deals with congestion better than connection-oriented service

Connection-oriented service

- Makes virtual circuits
- When conenction is made, fixed route is decided
- All packets follow the same route
- ISPs sometimes use this on top of IP
- The route needs to be neogitated in advance, together with bandwidth and other paramaters



Services compared

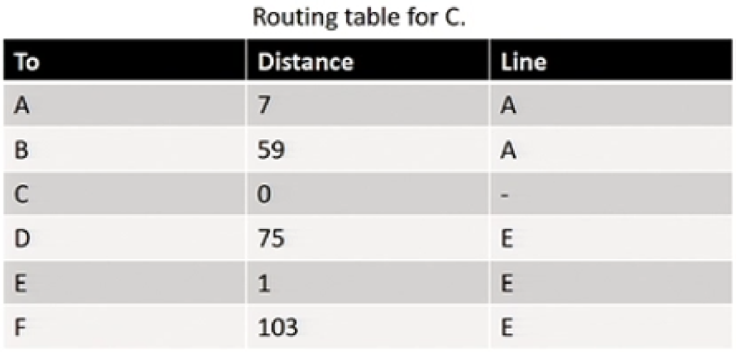
Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Routing

- How to move packets around the world
- Routing properties (they constitute trade-offs often):
  - Correctness
  - Simplicity
  - Robustness
  - Stability
  - Fairness
  - Efficeny
- Traffic engineers optimize routes between points. If the best route between A and C goes through B, then we know that the the route from B to A is preciesly a subset of such route.
- The collection of all best paths to a given destination is known as a tree

Routing tables

- You want to know for every address on which link to forward the packet (so cartesian product of all addresses…) look up table. The cost could be distance.



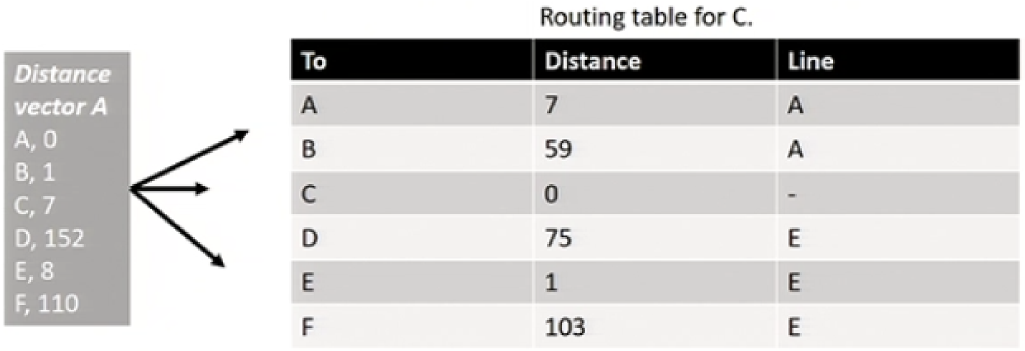
Routing table for C.

To	Distance	Line
A	7	A
B	59	A
C	0	-
D	75	E
E	1	E
F	103	E

- Routes get updated and oveloaded
- Need to keep it updated

Distance vector routing

1. Send your distance vector to your neighbors.
2. You use incoming distance vectors from your neighbor to construct a routing table.



Distance vector A
A, 0
B, 1
C, 7
D, 152
E, 8
F, 110

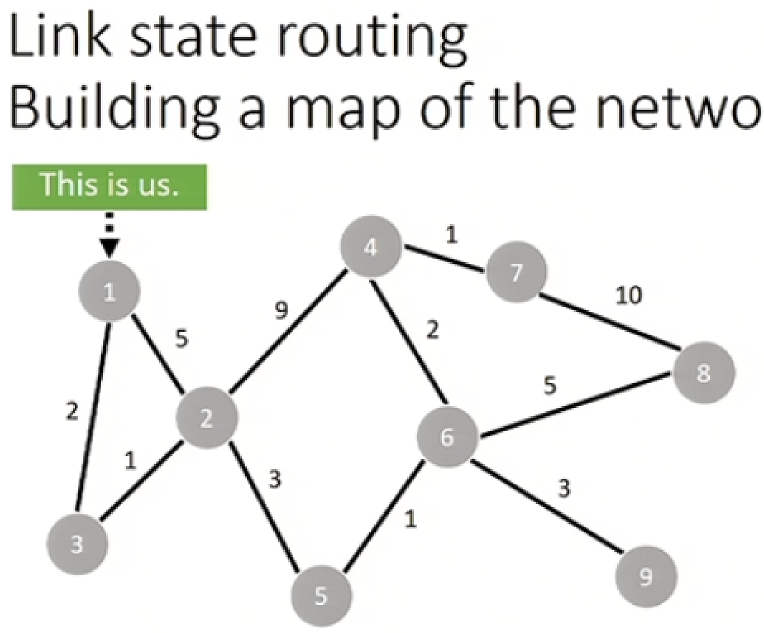
Routing table for C.

To	Distance	Line
A	7	A
B	59	A
C	0	-
D	75	E
E	1	E
F	103	E

3. The distance vector of A might be different than the old one existing in the routing table. Therefore A updates the routing table.
    1. If routers are faulty then you're screwed (i.e. by relying on outdaded wrong tables)
    2. Count to infinity problem when machine fails: A machine who previously had a cost of 1 to that machine, will get 9999…., then will just go through another node, but then the node at some point will realize that its previous work didnt work either, etc. then everything will be incremented by 1 continuously, until reaching 9999999….
    3. This happens because nobody has the overall picture

Link state routing

- Link state routing does not suffer from the count to infinity problem, but it is more complicated.
- Uses shortest path algorithm:
    - Routers only send packets with information about their direct neighbors (information that they confirmed is correct)
    - These packets are flooded over the network (flooding is robust)
        - Need to give sequence numbers to the packets to distinguish between old and new. But you never know what the latest sequence number is, so what you have might be outdated. Therefore validity age is used as well.
    - Routers built an overview of the network using these packets and run a shortest path algorithm

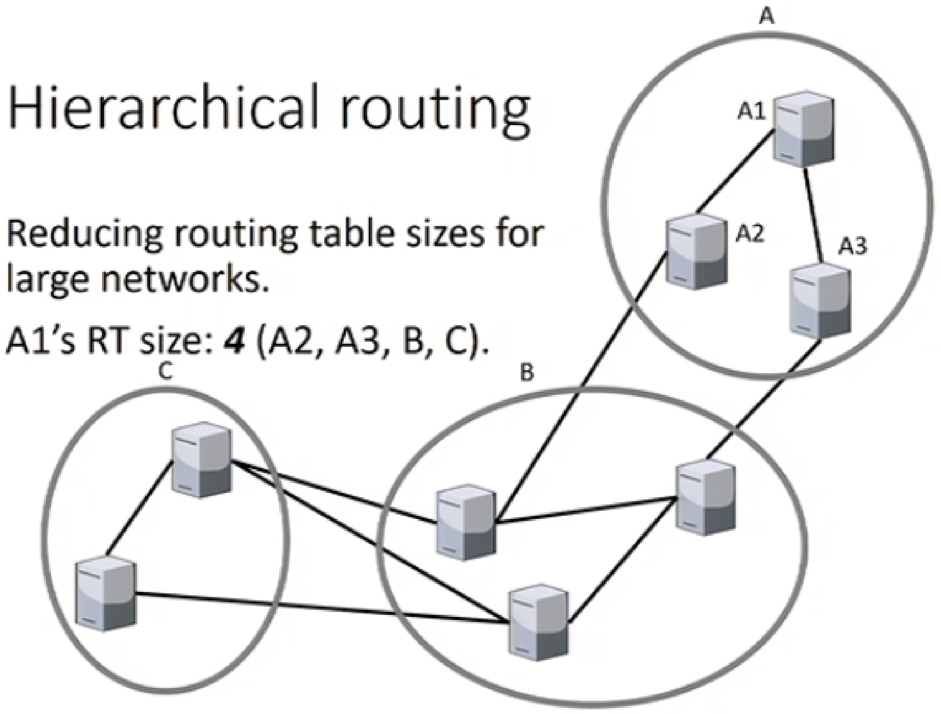


Link state routing
Building a map of the network

- Then we use Dijkstra

Hierarchical routing

- Reduce the size by grouping nodes



Hierarchical routing

Reducing routing table sizes for large networks.

A1’s RT size: **4** (A2, A3, B, C).

Congestion control

- Responsibility of both the network and transport layers

Types:

Prevent congestion by increasing available bandwidth

Self explanatory

Traffic aware routing

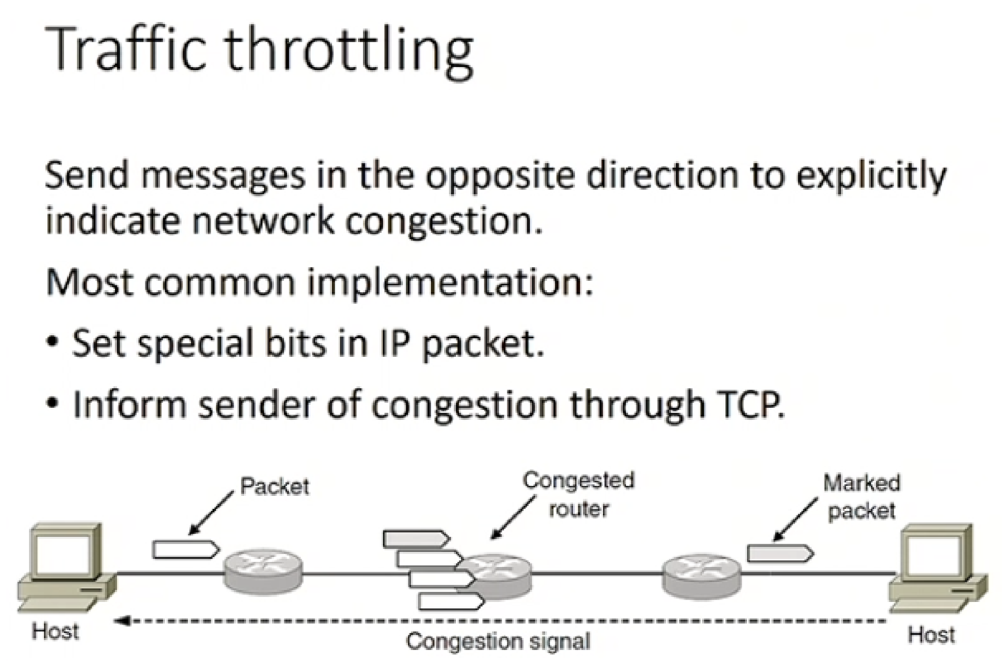
- Choose routes depending on traffic, not just topology

Admission control

- If theres’ congestion, new traffic has to wait
- Can be combined with traffic aware arouting

Traffic throttling

- ISP deliberately slows down your speed due to high traffic



Traffic throttling

Send messages in the opposite direction to explicitly indicate network congestion.

Most common implementation:

- Set special bits in IP packet.
- Inform sender of congestion through TCP.

Load shedding

- Sacrifice packets for overloaded routers
- Wired links are reliable
  - Wireless link need to solve transmission errors on the data link layer to support this
- Packet loss likely caused by congestion
- Random Early Detection (RED) drops packets randomly if buffer space is almost full
  - Sends implicit signal to the sender: slow down!
  - Sender interprets packet loss as conegestion signal

Internetworking

- Getting packets to their destination across multiple networks
- Networks may use different protocols
- Networks may offer different QoS guarantees
- Networks may have different maximum packet sizes

Tunneling

- If the source and destination networks use the same protcols, we can use tunneling
  - There’s an inbetween zone were both the entrance and exit routers are bilingual
  - These bilingual routers actually wrapp the packet into something their protocol understands and then unwraps it at the exit
- Cons: you cant debug the network as tools like ping dont work with the tunnel

Packet fragmentation

Transparent fragmentation

- Packet size can be limited by hardware, software, protocols, law
- Too big packets are split into smaller packets, and then group together

Nontransparent fragmentation

- The router splits the package but then the receiver doesnt get it grouped back, it has to fix it himself

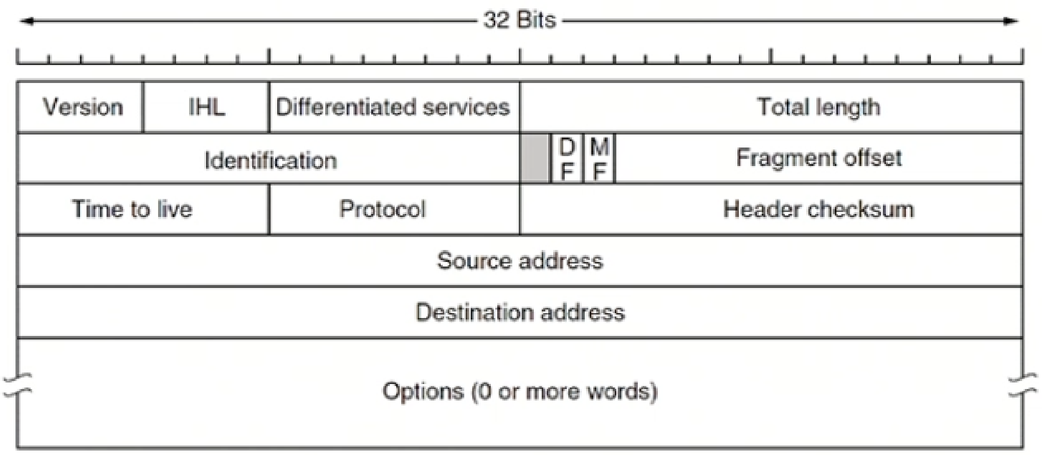
MTU discovery

- Router refuses to transmit the packet (so forces the source to split it) naturally the receiver will also have to group it back
- This used in IP protocol

Internet protocols

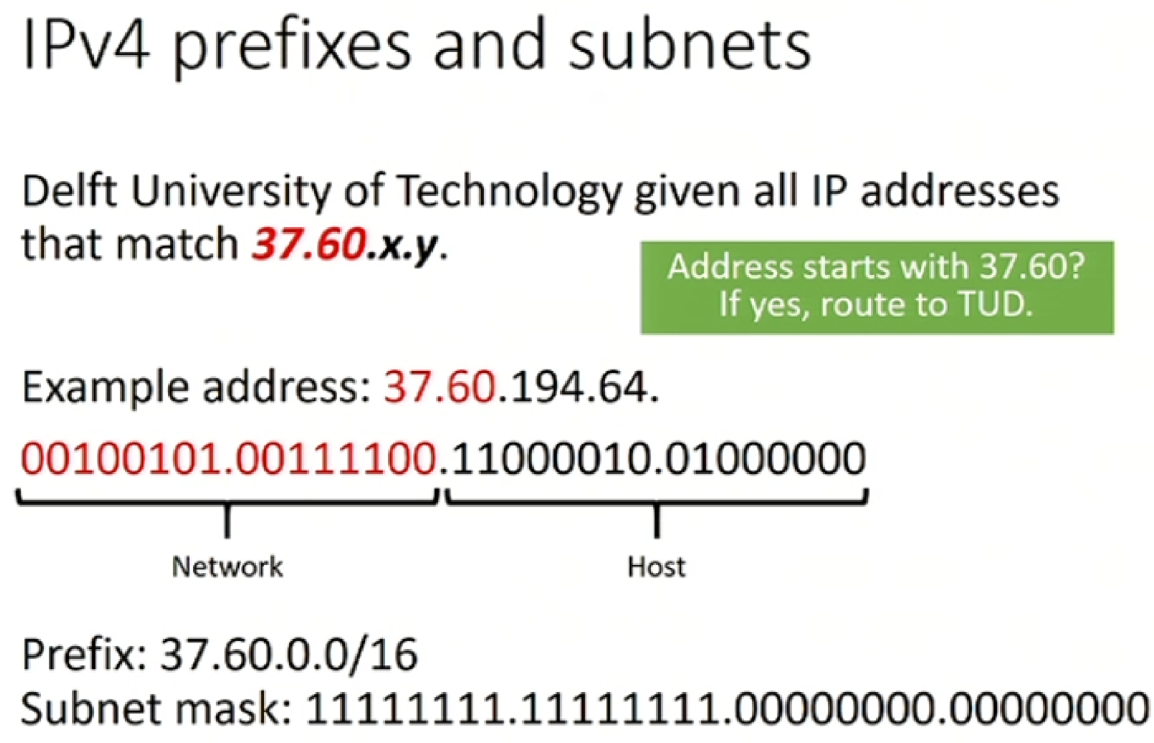
IPv4

The layout of a IPv4 packet header.



- One network to unify everything
- Time to live tells for how long to keep the packet floating in the internet, when its time expires and it gets deeleted, a message is sent back (to the sender). This could be used for DDOS attacks
- Identification is kept among split packets
- IPv4 uses 32-bit addresses
- Written in dotted decimal notation
- 2^32 is around 4 billion addresses
    - This would create a huge routing table
    - Instead the internet is divided into subnets so that we reduce the routing table sizes using hierarchical routing

For TU Delft any IP address that match 37.60.x.y will be routed to them. (TUD owns that subnet)



IPv4 prefixes and subnets

Delft University of Technology given all IP addresses that match ***37.60**.x.y*.

Address starts with 37.60?
If yes, route to TUD.

Example address: 37.60.194.64.

00100101.00111100.11000010.01000000

Network
Host

Prefix: 37.60.0.0/16
Subnet mask: 11111111.11111111.00000000.00000000

- IP addresses with the same prefixes are compressed into one entry of the routing table (this is called route aggregation).

Network Address Translation (NAT)

- Most people from nepal are behind 1 public IP address
- the NAT box gives internal IP addresses to those devices sharing 1 external IP address
- Since there is no more space in the IP header, to implement this we use the port number to distinguish among the machines sharing the external IP
- Now the problem is shifted to having 2^32 NAT boxes instead

IPv6

- Many more address
- Simplified header (improves bandwitdth/latency)
- Easier to add options in the header
- Improved security support
- 2^128 addresses
- Slow adoption rate

Internet control protocols

ICMP

- If something goes wrong, routers send these messages to senders
  - Destination unrecheable
  - Time exceeded - Used by the program traceroute
  - Echo and echo reply - used by the program ping
  - Router advertisment/solicitation

Address Resolution Protocol (ARP)

- ARP packet asks “who owns this packet?” with an ethernet desination address.
  - Allegedly only the owner replies saying “it’s mine”

Dynamic Host Configuration Protocol (DHCP)

- A machine without an IP address requests to get one assigned
- Somewhere a DHCP server received the request and offers an avaiable address. (IT INCLUDES SUBMASK, dns, GATWEAY ETC)

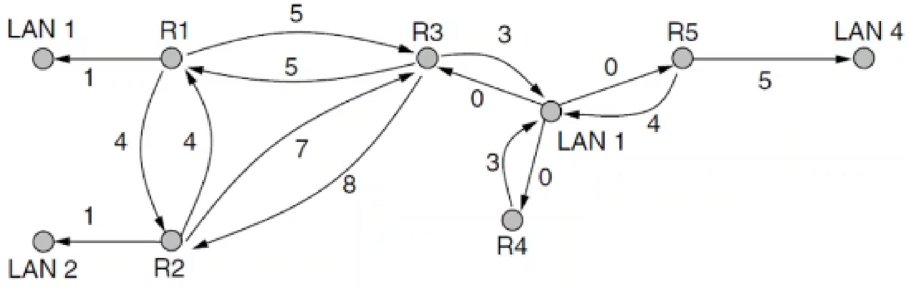
Gateway protocols

Routing in the internet

- Internet is made of Autonomous Systems (ASes), such as companies, universities, countries, ISPs
- Routing in the internet addresses:
  - Interoperability
  - political/economical policies of each AS
- Solution:
  - intradomain routing (routing packets inside AS) - interior gateway protocols
  - Interdomain routing (between AS) - exterior gateway protocols

Open ShortestPath First (OSPF)

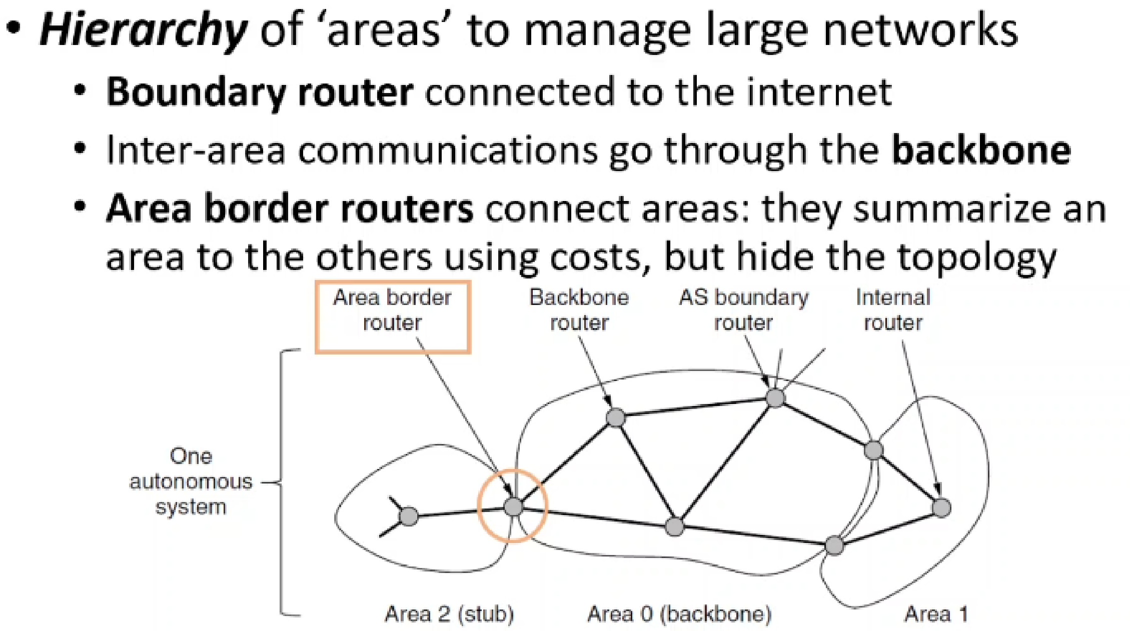
- Interior gateway protocol
- Routing within a (large) AS
- A form of link state routing
  - Builds a graph representation of the network



- Shortest path algorithm is applied

Properties:

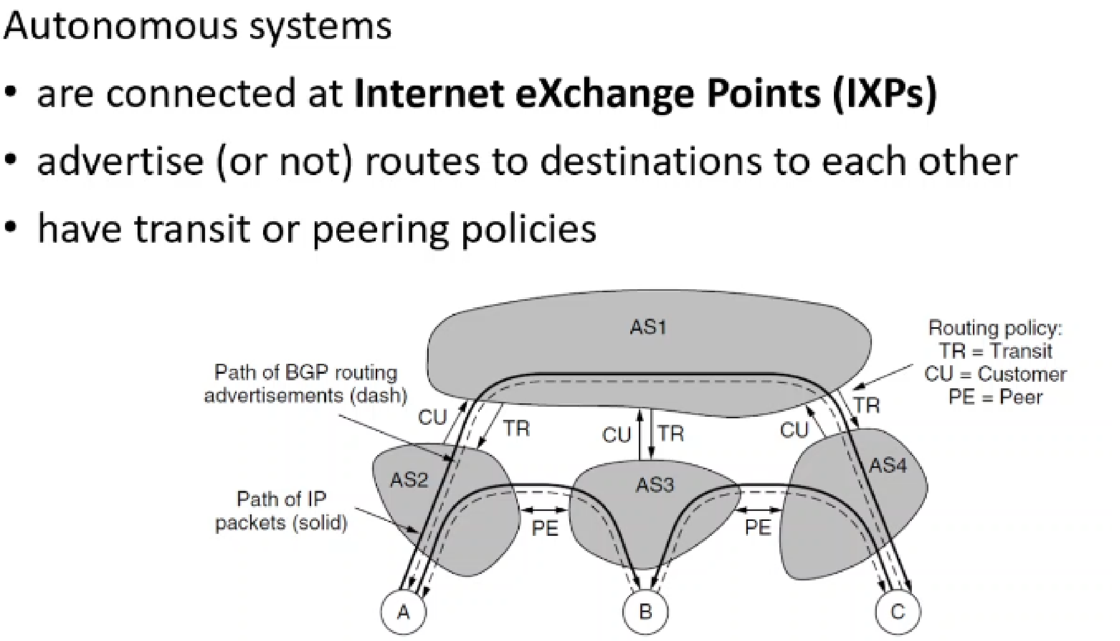
- Distance metrics may be measured as delay, physical distance, etc.
- Routers balance the load: communication load is split over routes of equal costs
- Dynamic: OSPF responds to routers crashing and changes the topology of the network accordingly
- Hierarchy: Uses areas to manage large networks



- ***Hierarchy*** of 'areas' to manage large networks
  - **Boundary router** connected to the internet
  - Inter-area communications go through the **backbone**
  - **Area border routers** connect areas: they summarize an area to the others using costs, but hide the topology

Border Gatweay Protocol (BGP)

- Exterior gatweay protocol
- Routing between (large) AS
- Supports various policies of ISPs, companies or countries



Autonomous systems

- are connected at **Internet eXchange Points (IXPs)**
- advertise (or not) routes to destinations to each other
- have transit or peering policies

- BGP uses distance vector routing
  - Combined with path vector protocol
- A route to a desination is a next hop router and a list of ASes
- Routing costs are not communicated
- An AS chooses a route using its own policies

Transport layer

Addressing and Connections

- Internet uses IP addresses for Network Service Access Points (ip forwards to transport)
- Internet uses ports for Transport Service Access Points (ports forward to application)
- The transport layer offers 5 primitives:
  1. Listen: waits for another process to contact him in the transport layer
  2. Connect: asks continuously if someone wants to connect at a specific port

3. Send: The connection has been established and so data is sent.
4. Receive: receive data over the established connection
5. Disconnect: release the conenction by leaving the port.

- Transport layer wraps stuff into segments, which contains transport layer specific information, such as ports, sequence numbers, etc.
- TCP service (transport layer API) is obtained by both the sender and the receiver creating endpoints, called sockets (each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a port). In order to use a socket at one machine a connection with the socket at the other machine must have been established.
- The socket contains the following berkely socket primitives (used by TCP):
    1. Socket: creates a new communication endpoint
    2. Bind: assigns a local address to the socket
    3. Listen
    4. Accept: passively accepts an incoming connection request
    5. Connect
    6. Send
    7. Receive
    8. Close

TCP (Transport layer) binds the sockets inside the segments to the relevant application. This means that every application using the internet must be connected to a port. Constantly running applications connected to the internet will have an open port connection and that is a lot of processing power, that's why smartphones run out of battery that quickly (each app running in the background constitutes a separate port connection)

process server (to the rescue): instead of constantly running application connections that you dont use that often, you have a process server that looks up the incomming sockets, then after checking the port it will open the port and forward it to the relevant application only when necessary.

- If the application uses random ports (instead of the default i.e. mail, web,…) the process server doesnt know what to do with the socket.
- If the application is "live" it doesnt make sense to shut the connection on and off constantly.

Segment Headers

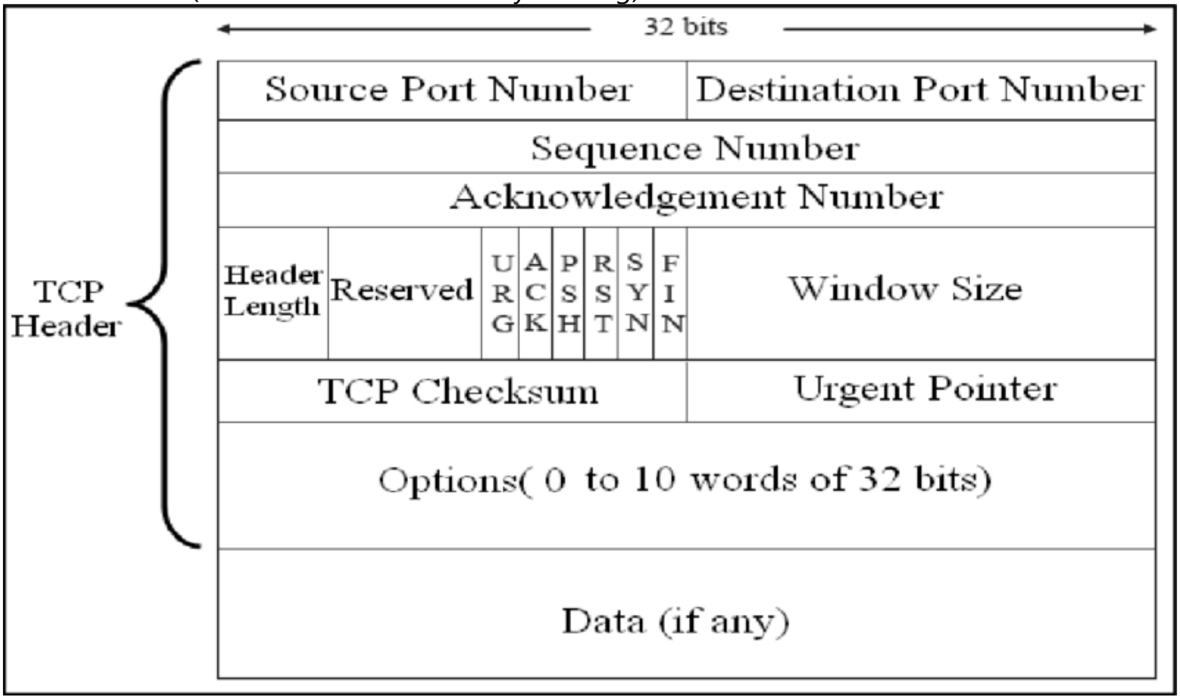
- Segment is the unit of transfirmation of the transport layer
- It consist of a header
    - Which has fields
- And a body
    - Actuall data

User Datagram Protocl (UDP)

- Connection-less protocol on the transport layer
- Very thin layer on top of the IP. The header just provides ports needed to connect to remote applications
    - Source port & Destination port
    - UDP length & UDP checksum
- Despite having a checksum, UDP itself does not automatically retransmit in case of error
- It is up to the application to use the checksum to trigger retransmission.
- UPD is a good choice for avoiding the complexity of TCP while having the possibility of doing retransmission.
- There is no IP address in the header as that information is known by context since it is already available in the network packet IP header

Transmission Control Protocol (TCP)

- Connection-oriented protocol
- Does have reliability by default (as opposed to UDP)
  - Error detection
  - Retransmission (automatic)
  - Flow control
  - Congestion control
- RFC = request for comment, published by the Internet Engineering Task Force (IETF)
- UDP: RFC 769 (a single one)
- TCP: RFC 4614 (this summarizes the many existing)



- The vertical letters are flags (only a bit indicating 0 or 1)
- ACK = 1 means the message is an acknwoledge
- Sequence number, acknowledge number anc TCP checksum are used for error detection
- Window size is for flow control
- Header length determines the length of the header because as opposed to UDP where its fixed, TCP can add Option arguments (i.e. timestamps) which increase the length of the header.

For each TCP connection, local state consists of:

- Next sqeunce nubmer to use
- Next acknowledgement number to use
- Congestion window
- Timers for retransmission

TCP/UDP reliability measures

Checksum

- Both use checksum
- Routers may be drop packets due to congestion or misfunction
- Routers may flip bits
- UDP/TCP checksum divides data into 16-bit words
  - compute bitwise XOR over all words
  - Take one’s complement of the result
  - For the receiver, if not all bits are 1, there was an error

Sequence numbers (of the TCP, not Data Link layer)

- Important in TCP. Not used in UDP

- Deals with lost data

Purpose of sequence sumbers:

- Detect missing segments
- Detect if segments arrive out of order
  - Due to routers with different latency
- Detect if parts of segments are missing
- Detect duplicates
  - It is up to the receiver to pick one
- Sequence numbers are 32 bits and stored in the TCP header
- SYN: connection establishment flag
- ACK: "this message is just an ack"
- FIN: connection termination flag
- These flags only need headers, so the body is often empty

- Before sending a SYN or after receiving a SYN a new squence number is established
  - initial own sequence number + number of previously sent bytes
- Acknowledgement number:
  - initial sequence number of communication partner + number of previously received bytes
- Use mod 32 bits when overflowing the numbers

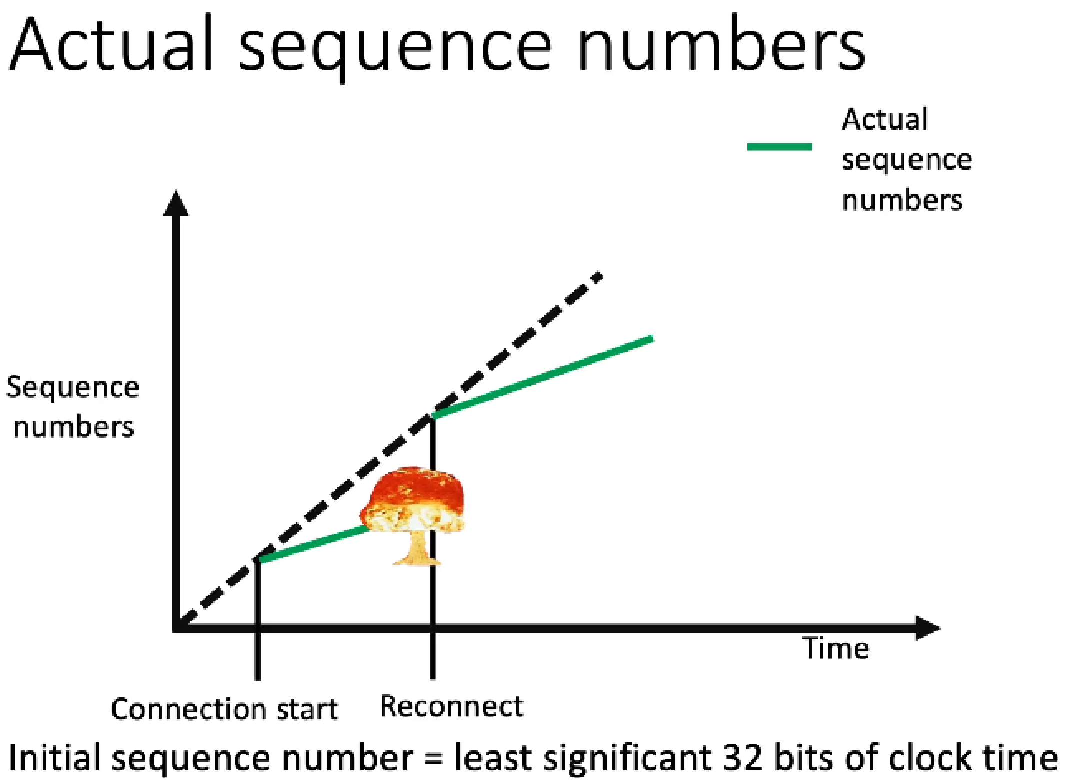
- Every data byte adds 1 to the squence number
- SYN and FIN flags also add 1 to the squence number
- ACK does not increase sequence numbers

Initial sequence number

- 0 (and generally any constant number) is a bad idea due to delays and crashes
- If you start at 0, and then the router crashes, then when it restarts it will use the sequence number 0 again, and the receiver has the same sequence number but a different ack. Since they dont match, they will never manage to send data.

Clock-based sequence numbers

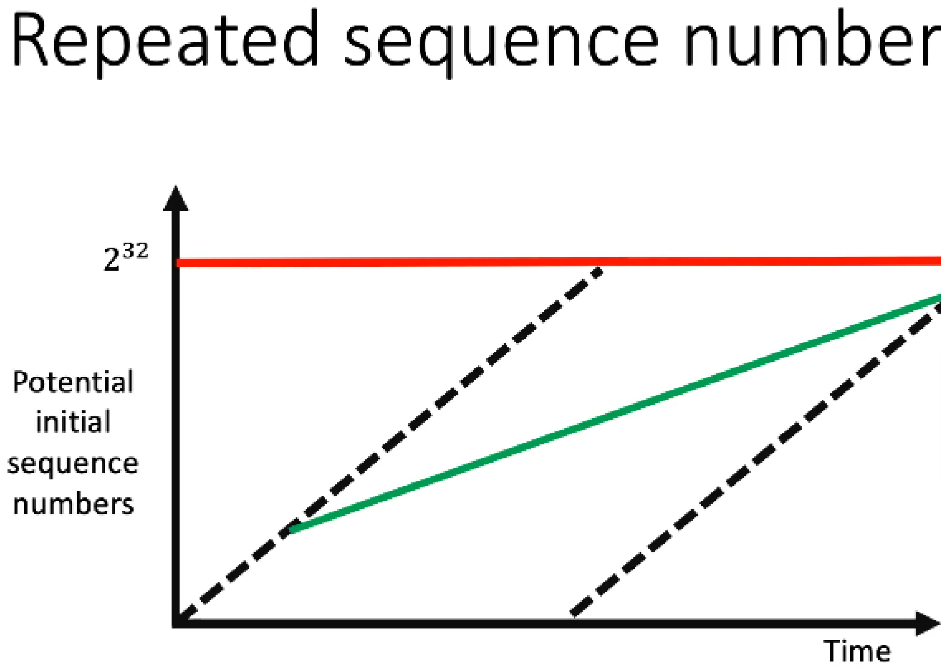
- Use the least significant 32 bits of clock time as initial sequence number
- TCP increases the sequence number at a slower rate than the clock speed



Actual sequence numbers

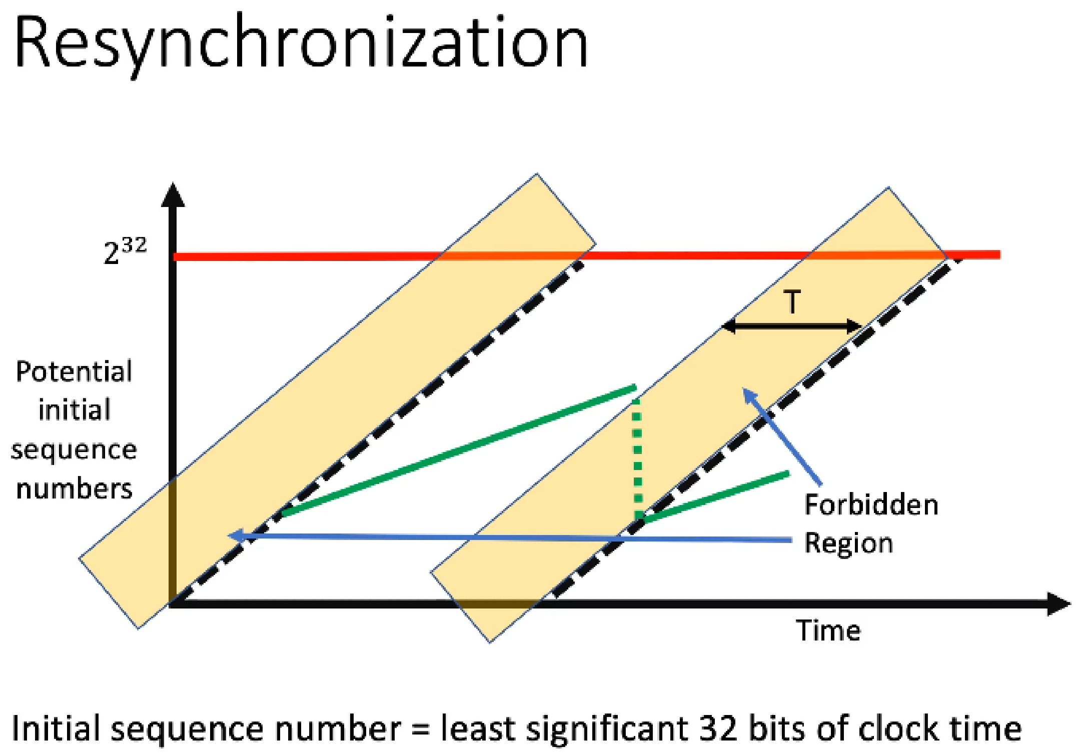
Initial sequence number = least significant 32 bits of clock time

Repeated squence numbers



Repeated sequence number

- Problem when there's a crash right before overlapping, this will confuse both sender and receiver just like in the 0 sequence number example. (incorrect duplicate detection)
- We use packet life time to deal with this together with forbidden regions (run a protocol that selects a new squence number for both parties)



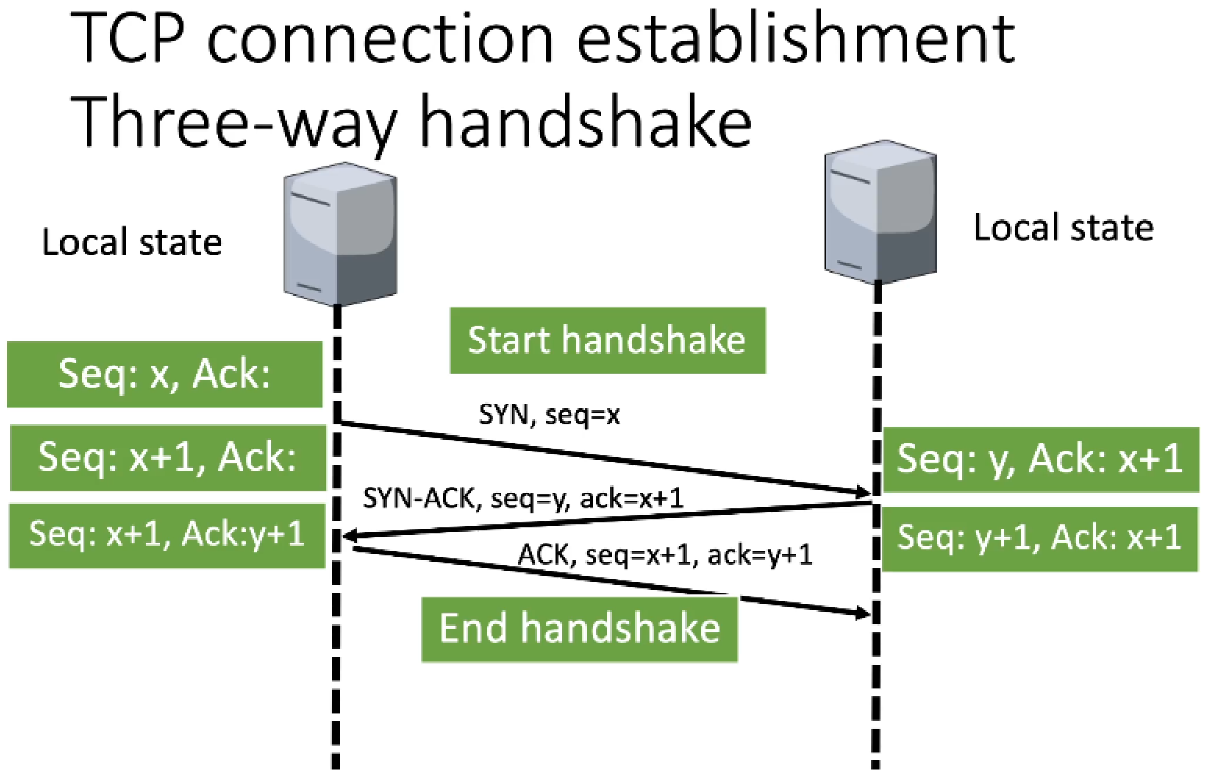
Resynchronization

Initial sequence number = least significant 32 bits of clock time

Modern TCP sequence number

- dont use clock-based sequence numbers because they are predictable and thus easy for an attacker to guess and impersonate a party in the communication
- modern TCP use randomized sequence numbers (you still need to resyncronize for the forbidden region problem)

Establishing a TCP connection (Three-way handshake)



TCP connection establishment
Three-way handshake

Local state

Local state

Start handshake

Seq: x, Ack:

SYN, seq=x

Seq: x+1, Ack:

Seq: y, Ack: x+1

SYN-ACK, seq=y, ack=x+1

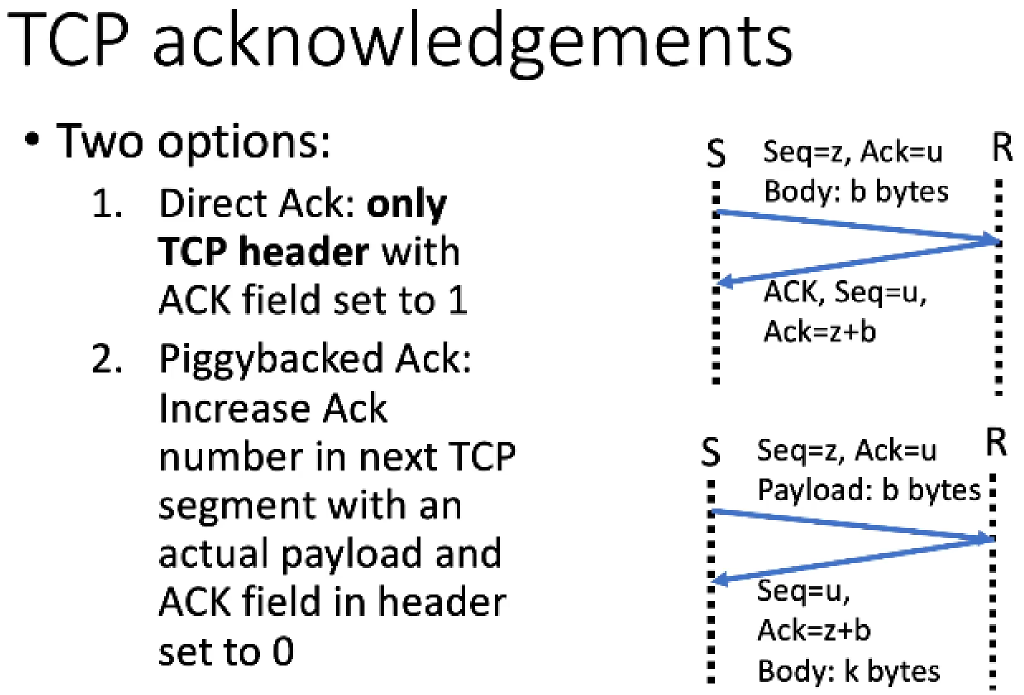
Seq: x+1, Ack:y+1

ACK, seq=x+1, ack=y+1

Seq: y+1, Ack: x+1

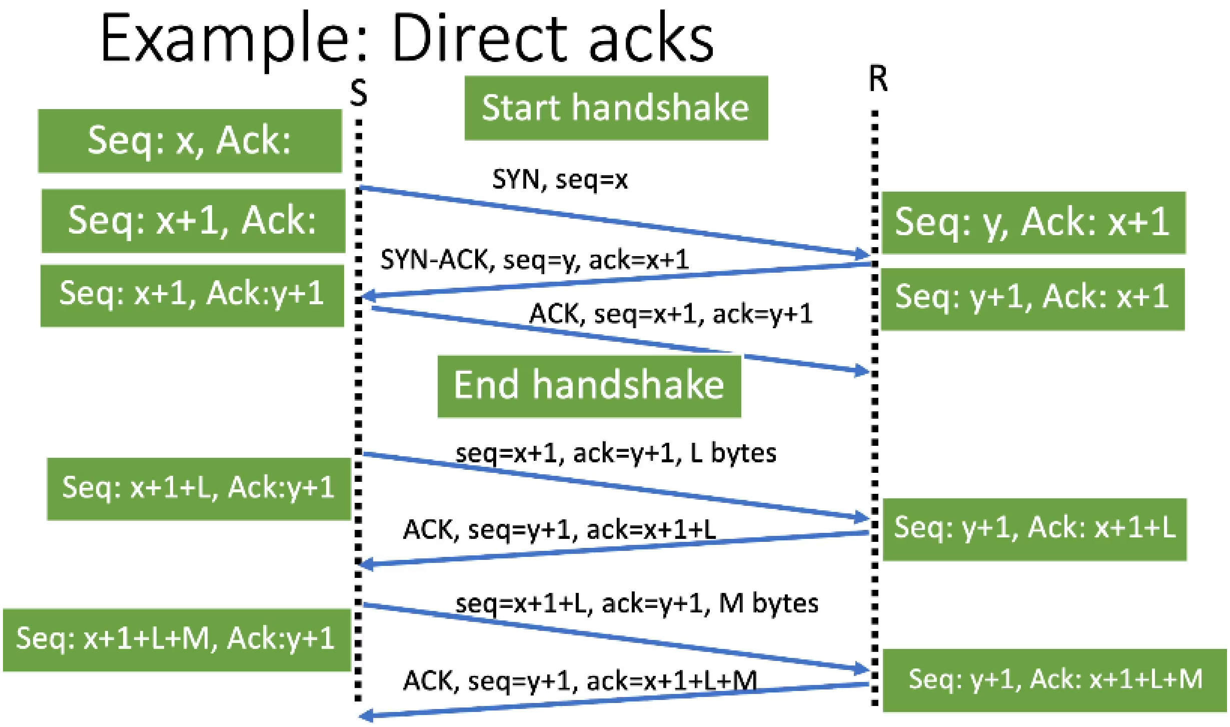
End handshake

TCP acknolwedgements



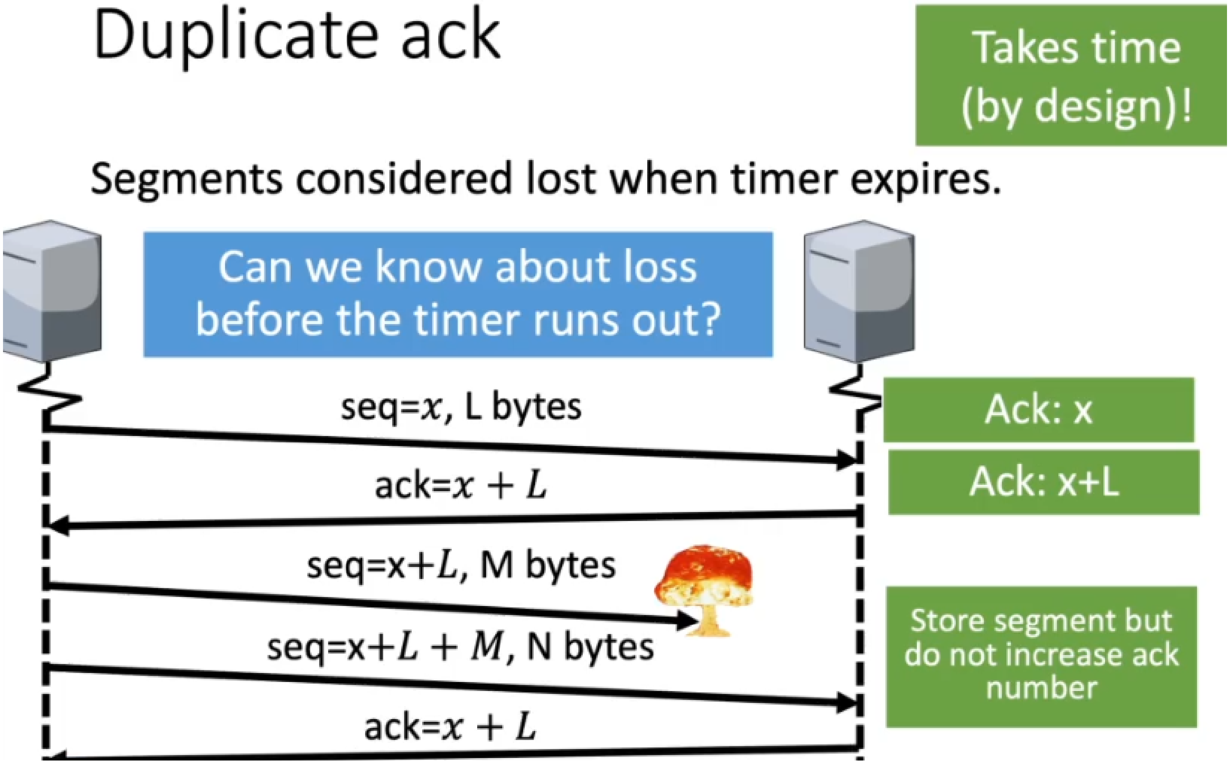
TCP acknowledgements

- Two options:
  1. Direct Ack: **only TCP header** with ACK field set to 1
  2. Piggybacked Ack: Increase Ack number in next TCP segment with an actual payload and ACK field in header set to 0



Example: Direct acks

Duplicate ACK

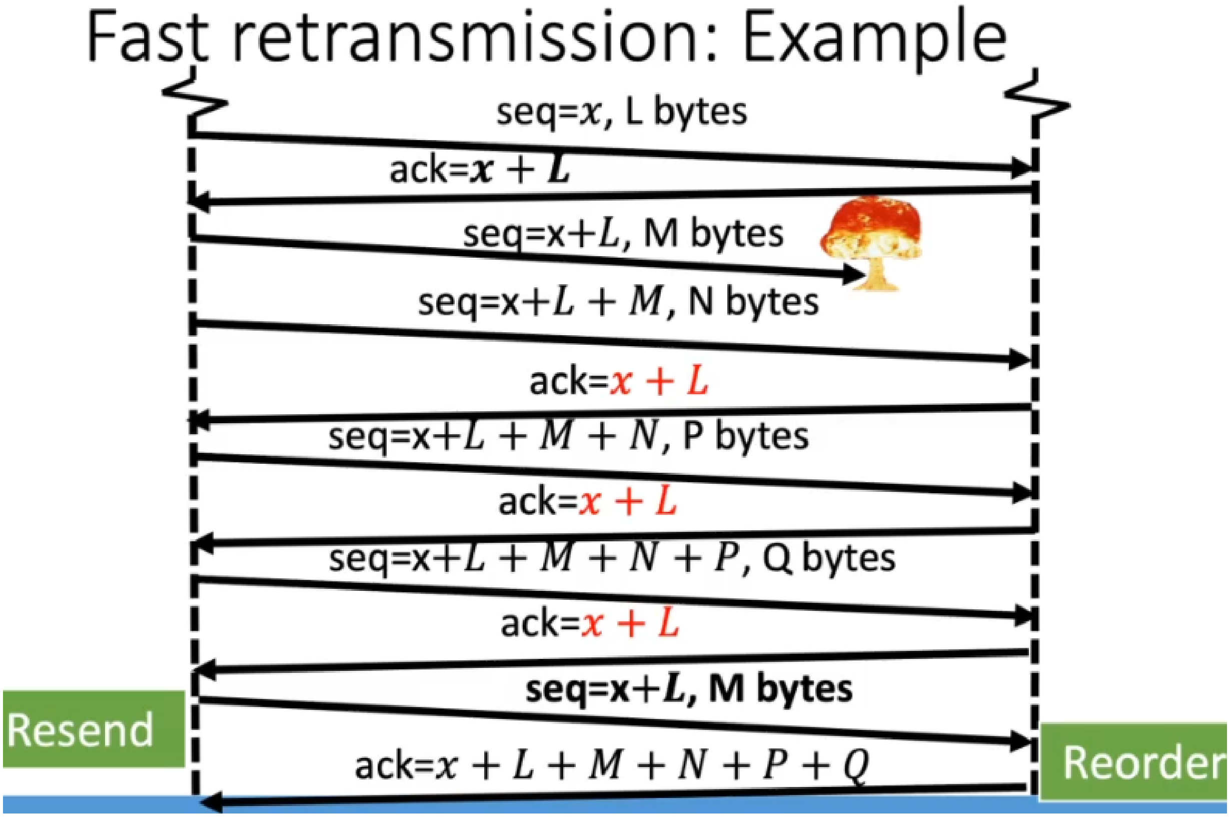


Duplicate ack

Takes time (by design)!

Segments considered lost when timer expires.

Can we know about loss before the timer runs out?



Fast retransmission: Example

- TCP as asliding window protocol uses selective repeat (3 duplicate acks = negative ack)
  - Instead of a number of k unack frames, TCP allows only for a lmited amount of data to be unack
  - Amount of data depends on current situation in network and on receiver's buffer size

Crash recovery

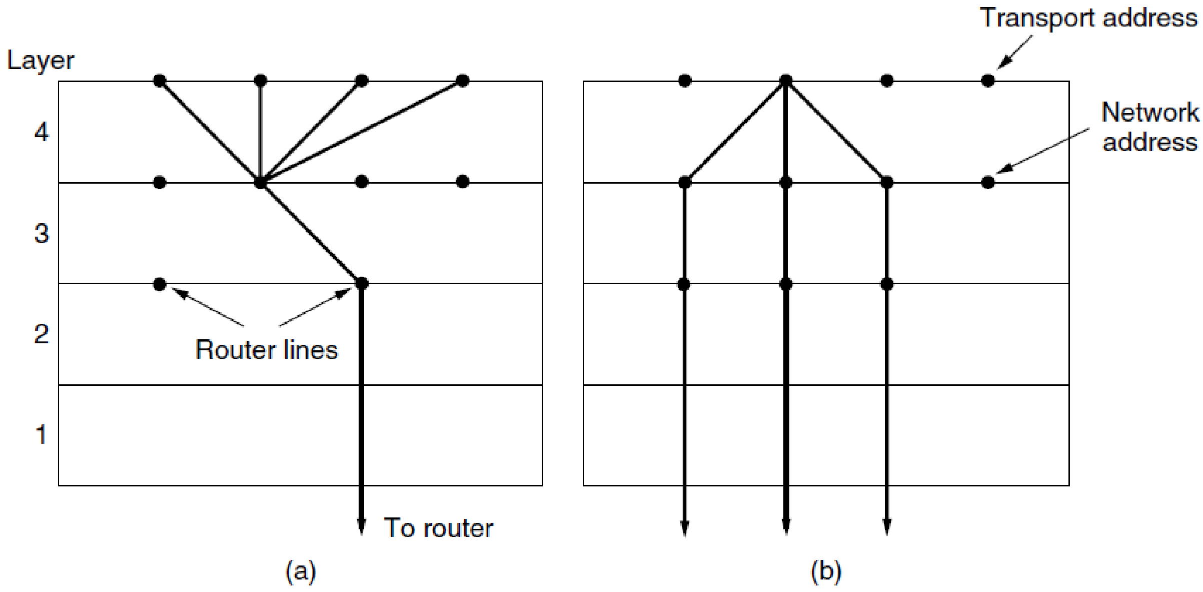
- We cannot create fool-proof crash recovery in layer k. Memory loss on layer k makes it impossible to know if there is inconsisentency between sent acknolwedgement and forwarded data

- The application layer has to deeal with it

Acknowledgements and error detection/correction differences between data link layer and transport layer

- We only error control at the data link layer and the transport layer.
- We need it on both datalink layer and transport layer because the two protect against different types of errors:
    - the data link layer deals with bitflips or lost data on a link (e.g., cable)
    - the transport layer deals with bitflips or lost data introduced by a device, most commonly a router
        - RED (Randopm Early Detection): a router randomly drops packets on the network layer to indicate congestion, which is something the error control of the data link layer cannot detect as it happens after the data link layer checks have been completed. That is why the transport layer also has error control (because of unreliable network routers)
    - Remember that altough the network layer (IPv4) packets have header checksums, it is seldom used by the ISP routers and it is officially discarded in IPv6 packets.

Multiplexing

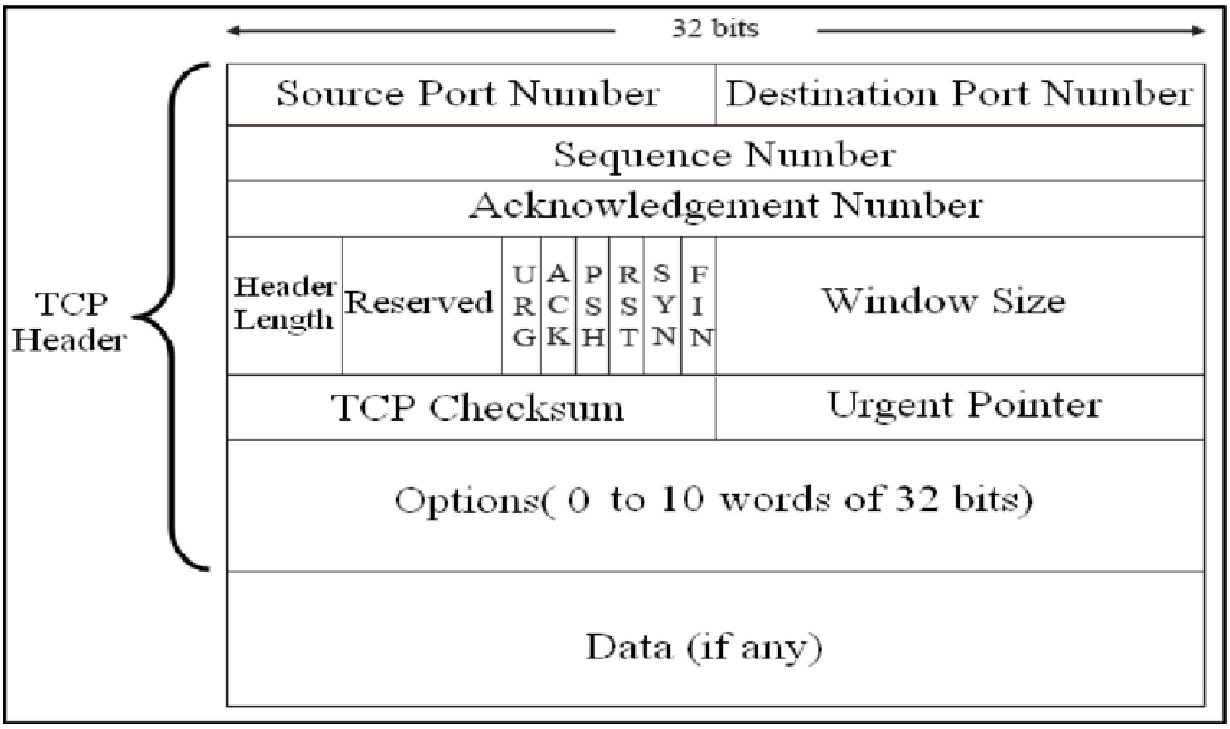


(a) Multiplexing. (b) Inverse multiplexing.

Difference between flow control and congestion control

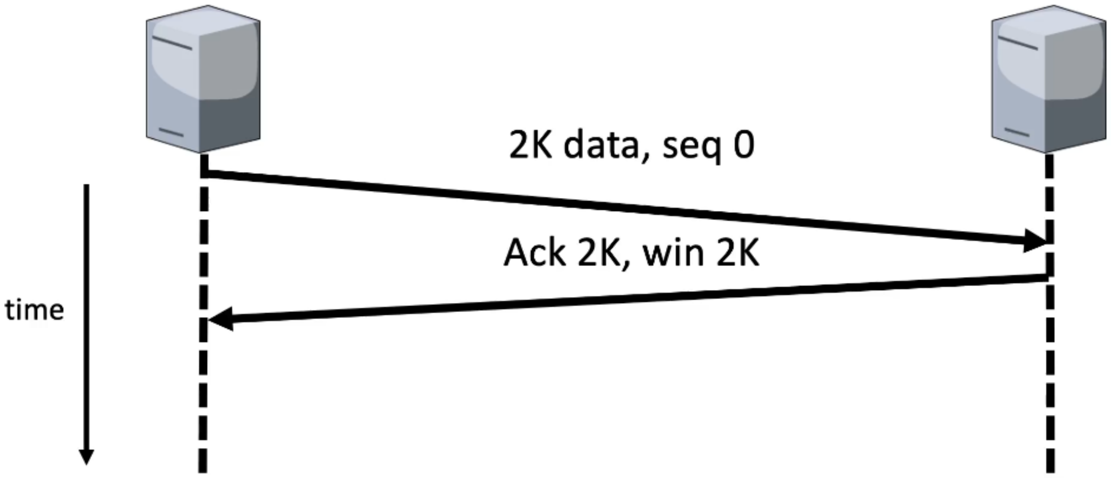
- Flow control: To say the sender to slow down the speed so that the receiver can process the data. Receiver is bottleneck.
- Congestion control: The same but here the network is the bottle neck whereas the receiver has not problem with the speed.

TCP Flow control

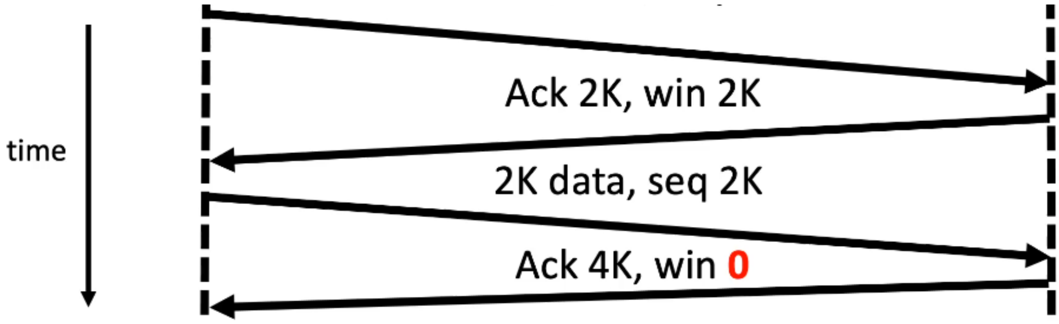


- Window size (buffer) tells the sender how much data the recepient can handle

TCP window size
Flow control



- If the sender keeps sending and the recepient hasnt had time to process the data then the window size decreases

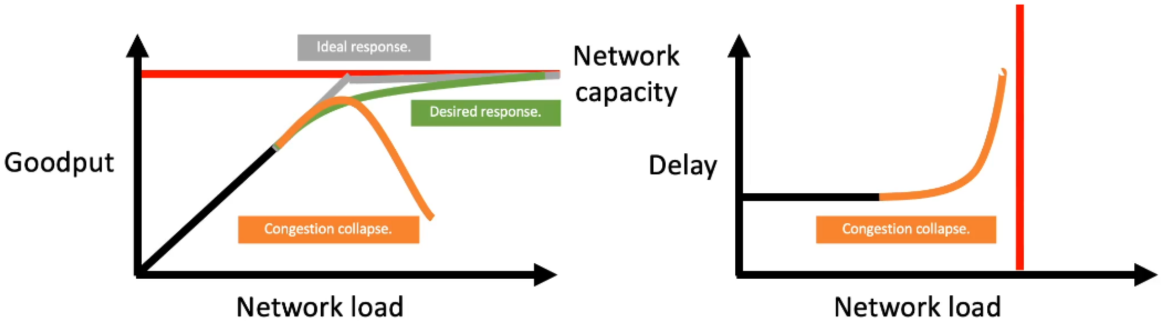


- Now sender will be quiet until the receiver has gained some buffer space for which the receiver will send another acknowledgement with the same number but with increased window size (it will not be interpreted as a duplicate acknowledgement as the window size is different)

- Very tiny window size will make the receiver to increase the overhead of the channel and waste bandwidth with a lot of acknowledgements for when the buffer has been restored. This is known as the silly-window syndrome.
    - Furthermore, by the time the ack has been sent, the actual buffer size might have increased, but the sender will send the old smaller size. This is inefficient as the receiver could have actually handle more and bandwidth is being wasted.
        - The slower the network the even more wasted the bandwidth is from the silly-window syndrome
- TCP in being conservative and making sure that the reciver can handle the data is sacrifying throughput. This is why some applications prefer to just use UDP and implement more efficient reliability algorithms on top of it.

TCP congestion control

- Both network and transport layers are responsible for congestion control
- The network layer sends signals if there is congestion by dropping packets or routing them by a longer/less congested path
- The transport layer can arrange the amount of load that is added onto the network



- While everything is going fine TCP will try to increase the bandwidth it uses
    - It will slow down as it receives a congestion signal

Congestion signals

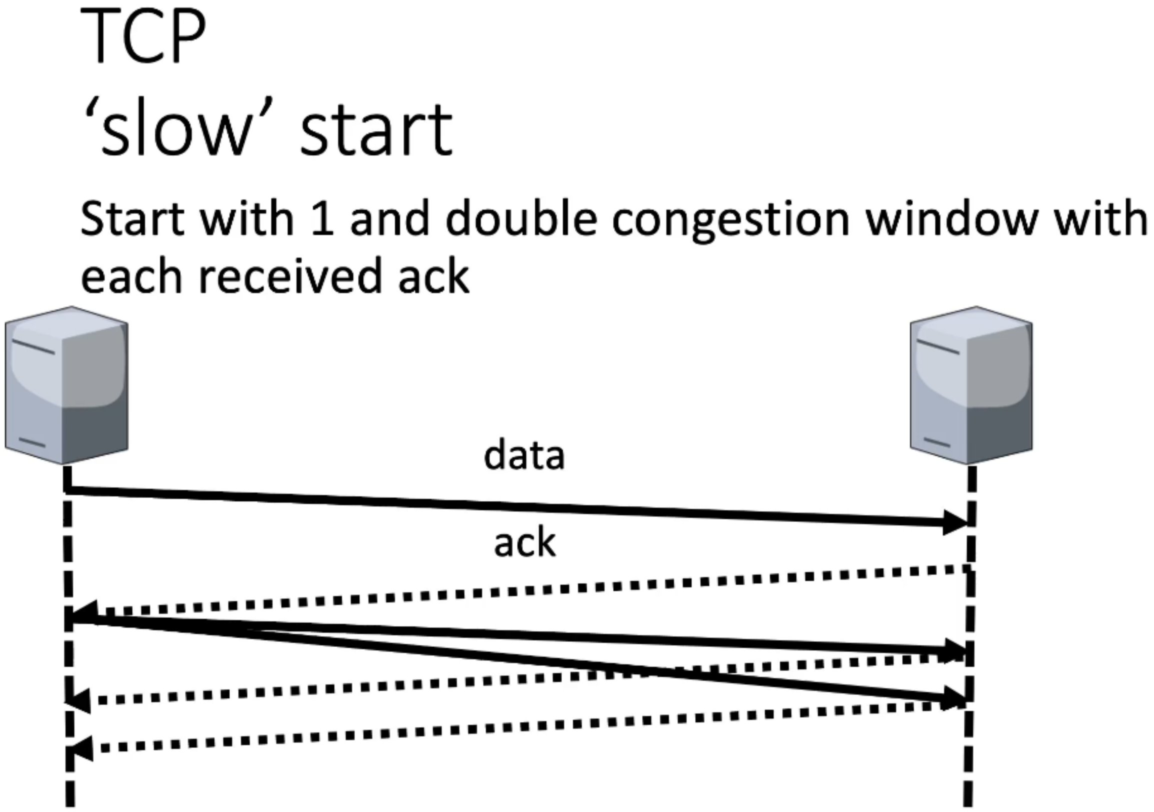
- Explicit feedback from routers (max amount, rare implementation)
- Loss (most common one)
    - Protocol RED (random early detection): drops packets as congestion happens and transport layer reacts accordingly
    - a delayed packet can be confused for a lost packet
- Latency (delays between received acks bigger than between sent segments)
    - Takes longer to calculate but gives a more accurate description of how much data the network can handle

Sharing bandwidth

- We want to approximate the optimal point, that is the point that uses 100% bandwidth efficency and 100% fairness (that is, 100% of the bandwidth is split evenly among all the users using it)
- In practice it is hard to achieve it as a user doesnt know accurately with how many other users it’s sharing the network
- TCP: increases as things go well and decreases when a a bad signal is received
- Increase/decrease approaches:
    - Additive: It’s linear, increases by +k and decrease by -k
    - Multiplicative: It’s geometric, increases by * k and decreases by * 1/k
    - Combination of both: (cartesian product of of additive and multiplicative)
    - Only additive or only multiplicative will not converge to the optimal solution
    - Additive increase with Multiplicative decreases (AIMD) does converge to optimum

TCP implmentation of congestion control

- Sender needs to know how much the network can handle.
- TCP implements congestion window. Congestion window is not the window size field of the TCP segment header. Congestion window regards the network.
- Congestion window is tracked on the sender. It specifies how many segments can be transmitted.
- TCP will use the minimum of min(congestion window, window size)
- TCP will not start with additive inc multuplicative dec but with exponential (doubles):



TCP
‘slow’ start

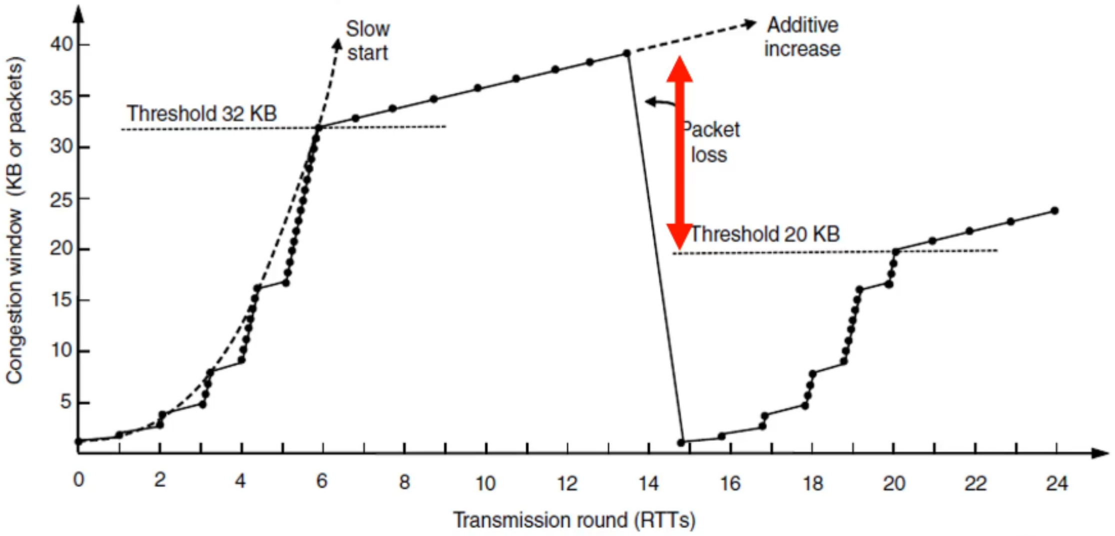
Start with 1 and double congestion window with each received ack

- Despite being exponential, the name “slow” start comes from comparing it to the previous algorithm that started the congestion window value = flow control window (window size). Slow start therefore starts with the smallest value as opposed to the maximum.ssssssssssssssssssssssss
- The exponential increase needs to stop before congesting the network:
  - Threshold congestion window size is reached
    - Then swith to increaseing by constant value (additive increase)
    - Threshold value is arbitrary and differs between TCP versions
  - Missing acks indicate a data loss and hence congestion
    - Behaviour depends between TCP versions. TCP Tahoe (first TCP version) would first drop to 1 and set the congestion window to half the previous one and from there switch to additive increase.

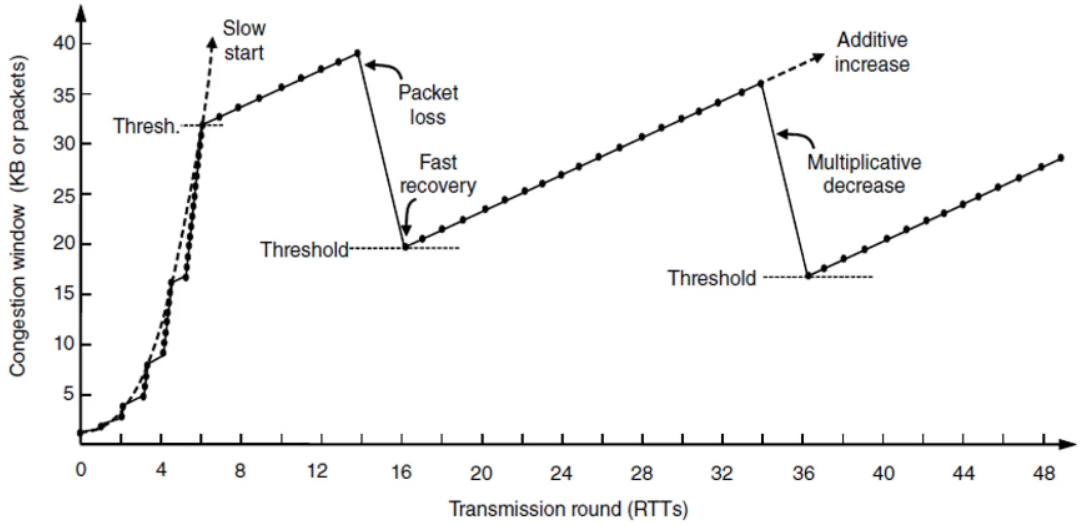
TCP Tahoe

Packet loss (indicated by missing acks) resets congestion window to 1.

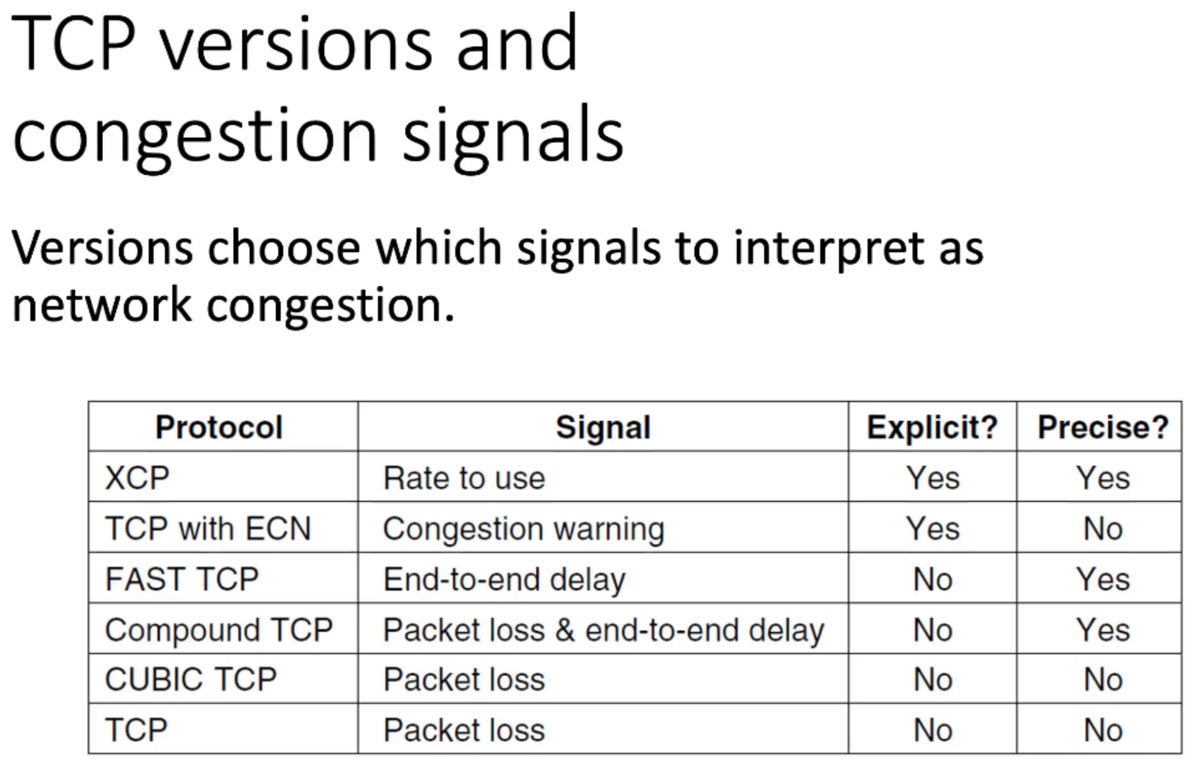
Threshold set to previous window/factor



TCP Reno
(= TCP Tahoe + fast *recovery*)



- Reno moves directly to the half threshold of the congestion window and continues with additive



TCP versions and congestion signals

Versions choose which signals to interpret as network congestion.

Protocol	Signal	Explicit?	Precise?
XCP	Rate to use	Yes	Yes
TCP with ECN	Congestion warning	Yes	No
FAST TCP	End-to-end delay	No	Yes
Compound TCP	Packet loss & end-to-end delay	No	Yes
CUBIC TCP	Packet loss	No	No
TCP	Packet loss	No	No

- Most TCP protocols rely on implicit congestion feedback based on how the network behaves (easier to implement)

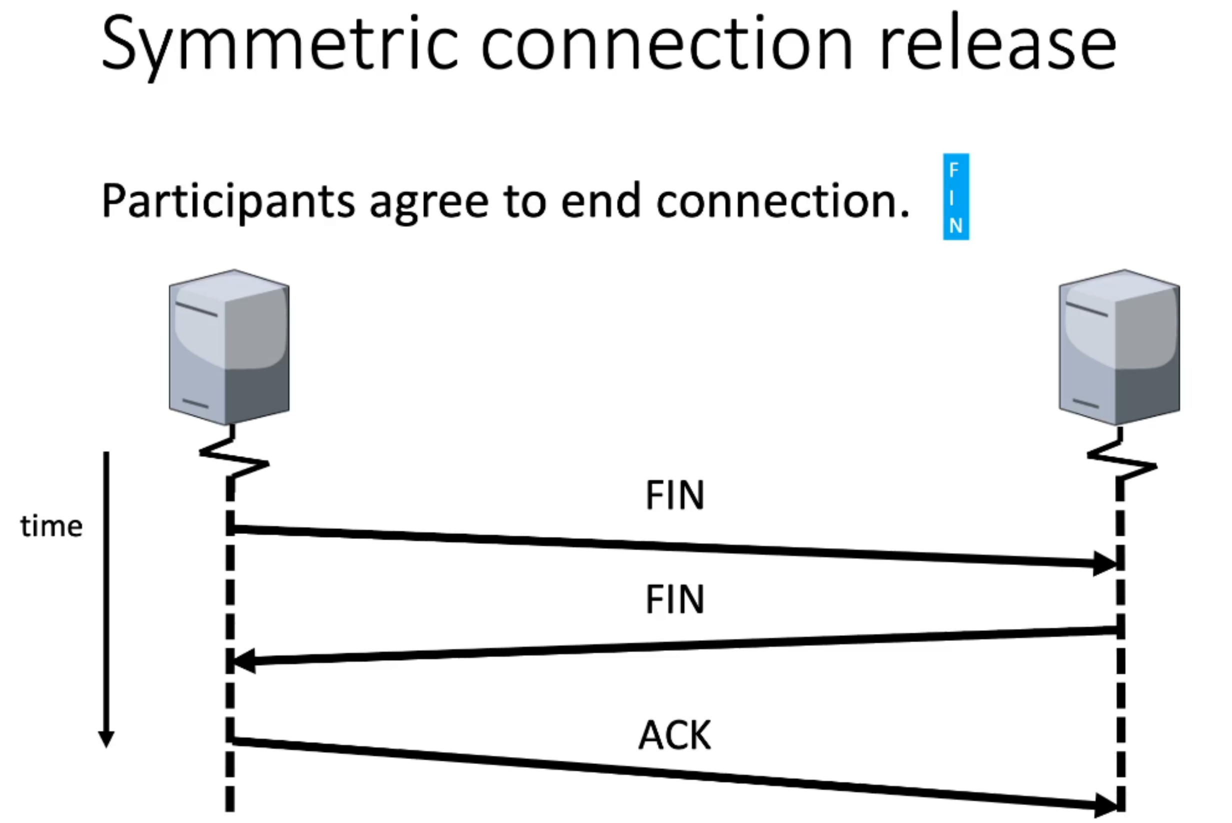
TCP connection release/termination

Asymmetric disconnect

- Just one guy does it without having to agree on it

Symmetric disconnect (used by TCP)

- Both parties agree on it
- Two armies problem



Symmetric connection release

Participants agree to end connection.

- If you dont get response to the FIN you make (or to the ack if you are the receiver), you make an asymmetric disconnect after some time

Aplication Layer

DNS

Domain Name System, an application that enables most of the other applications. When you want to visit a website, you enter a human-readable name in your browser. However, routing replies on IP addresses. DNS servers enable you to find the IP address that you are looking for.

- The top level domain is the .com, .gov, .edu, etc, it can be generic (as the examples given before) or country: .es, .nl, .jp, etc. Top level domains are controlled by Internet Corporation for Assigned Names and Numbers (ICANN). There are some politics involved. Originally top level domains end with a dot, (as it is a seperator that also indicates the end), but in practice it is skipped.
- Second-level domains can be requested by orgonizations from registrars
- Once you own a second-level domain, you are free to establish any third-level domains you want within your second-level domain.
- Inputting an address on telnet/firefox should directly send you to the location without going via the name server.

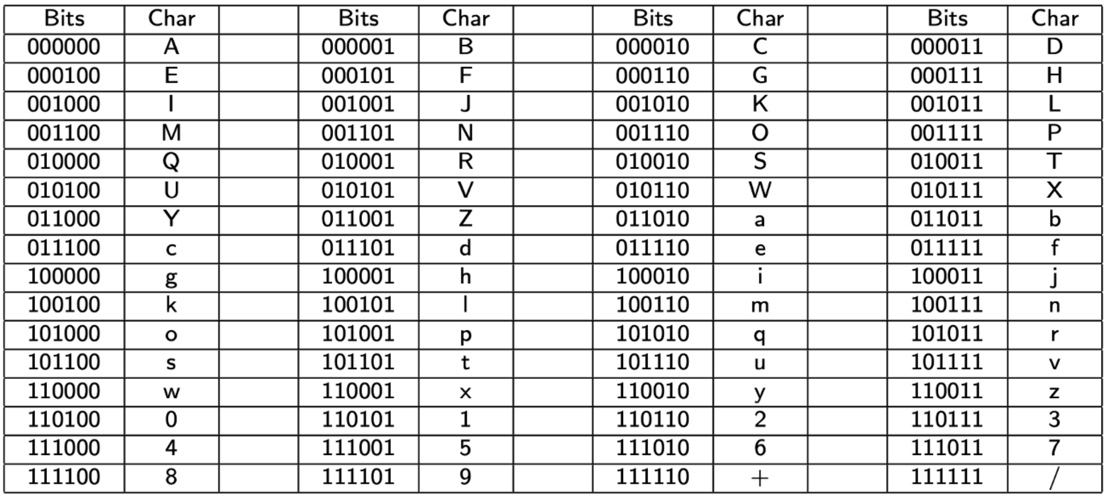
Name servers

- Name servers are dedicated machines that return the IP address of a human readible address.
- Our operative systems keep track of name servers and dynamically selects which one to use. The location of name servers is configured via DHCP (your internet service provider will hand it to you)
- A custom name server (instead of using the one via DHCP) is called “local name server” (but its not necessarily “local” as you can use a public one like google’s dns). The local name server probably doesnt have all the information that you need. The server will contact other servers:

  - **root server**: It has information about all the top level domains
  - **2nd level domain server**: i.e. a server that knows all the 3rd level domains within ".es"
  - **3rd level domain server**: same but within a 3rd level domain.
- The local name server might have already cashed the 2nd level domain server.
- The host (i.e. the person that wants to resolve a dns query to visit a website) runs a recursive query, that is, it will first ask the root, then 2nd then 3rd level domain server until obtaining the desired result (and will not just return a partial result), which is sent to the application.
- The name servers between themselves run iterative queries, these guys send the partial results they get (and call it a day) so that the host can move onto asking the subsequent name servers.
- The host machine can also cache name servers wich skips certain queries and speeds up the process. However, cached answers are not authoritative, since changes made at cs.washington.edu will not be propagated to all the caches in the world that may know about it. For this reason, cache entries should not live too long. This is the reason that the Time to live field is included in each resource record
- Name servers reply with domain resource records. It may contain:
  - IPv4 address
  - IPv6 address
  - Accepts email yes/no
  - Name server for this domain
- DNS messages are sent in UDP packets with a simple format for queries, answers, and name servers that can be used to continue the query resolution.

Email

- You can send and receive email on your own domain and give out addresses to your friends
- Or you can use an email service provided by an organization
- An email message contains an envelope, a header and a body
  - Envelope: Fromn: email@address.com, To: email@address.com and Encryption:type
  - Header: From: Name, To: Name, Subject: The subject
  - Body: Hi,...,kind regards, Sergio.
- Email uses multiple protocols
  - Users use POP3 or IMAP (Internet Message Access Protocol) to interact with their mailbox.
    - IMAP sends commands to mail server to manipulate mailboxes
      - LOGIN (into server)
      - FETCH (fetch messages from a folder)
      - CREATE/DELETE (a folder)
      - EXPUNGE (Remove messages marked for deletion)
        - POP 3 would delete all the messages from the server automatically as they are being fetched into a client. However since most people use multiple devices it is better to keep the messages in the server. That's why IMAP replaced POP3. Expunge will delete messages only after loging again.
  - To send messages users use SMTP (Simple Mail transfer protocol, which in itself is just the end to end protocol between the servers (back then the user agent were the servers too, so it was fully end-to-end)) + Extensions (e.g. AUTH)
  - If people want to send more than just text messages additional headers are introduced via Multipurpose Internet Mail Extensions (MIME) which contain the content-type such as Text (text/plain, text/html), Images (image/jpeg, image/gif), Video (video/mp4, video/mpeg) or multipart (a combination, for attachment).
    - When MIME was introduced, servers were not expecting non-ASCII data.
    - Binary data would be encoded to make it look like ASCII, then the receiver would have to do the decoding. Base 64 is used.



Bits	Char		Bits	Char		Bits	Char		Bits	Char
000000	A		000001	B		000010	C		000011	D
000100	E		000101	F		000110	G		000111	H
001000	I		001001	J		001010	K		001011	L
001100	M		001101	N		001110	O		001111	P
010000	Q		010001	R		010010	S		010011	T
010100	U		010101	V		010110	W		010111	X
011000	Y		011001	Z		011010	a		011011	b
011100	c		011101	d		011110	e		011111	f
100000	g		100001	h		100010	i		100011	j
100100	k		100101	l		100110	m		100111	n
101000	o		101001	p		101010	q		101011	r
101100	s		101101	t		101110	u		101111	v
110000	w		110001	x		110010	y		110011	z
110100	0		110101	1		110110	2		110111	3
111000	4		111001	5		111010	6		111011	7
111100	8		111101	9		111110	+		111111	/

- Sometimes you have to padd the data of the last 6-bit group. For 2 paddings you append an ‘=’ sign.

Instant messaging vs email

- Some messaging services use same concept as email (e.g Jabber/XMPP)
  - Choose or set up a server
  - Messages can be sent between parties using different servers
- More commonly, servers are controlled by one party (e.g. WhatsApp, Signal)
  - Often server pusshes messages to receiver rather than letting the reciver be the one that pulls them (eg email).

World Wide Web

- Webpage: one document on the web
- Website: set of webpages, usually linked under one main domain
- HTTP: Hypertext Transfer Protocol (for fetching and posting data)
- HTML: Hypertext Markup Language (for displaying content on the web browser)
- Web uses Universal Resource Locators (URLs) as an alternative to protocol:IP:port/path

HTTP requests

- GET: retrieve content, consists of header and body
- HEAD: like get but without body (just the header) (used for develop, to see whether response exsits, redirects, etc.)
- POST: create new resource, might just add to resource (e.g. comment)
- PUT: put resource, replace current resource
- PATCH: modify existing resource
- DELETE: delete a resource
- These names are case sensitive (domain part of urls are not case sensitive, paths can be)

- HTTP runs on top of TCP on port 80
- HTTP 1.0 would setup a TCP connection for each request, HTTP 1.1 would reuse the initial connection for requests within the somain domain to amortize TCP setup/disconnect costs and suspport pipelined requests
  - TCP uses a 3 way handshake to setup a connection
  - HTTP keeps the current TCP connection alive for requests of the same website (same domain and server)
  - Multiple parallel requests are possible with pipelining (HTTP 1.1)
    - disadvatnage: server has to send responses in the same order it received the requests (bad for performance and abused for attacks)
  - Multiplexing in HTTP2: Give each request a number andmatch based on the number

HTTP response codes

Responses are grouped in five classes:

1. Informational responses (100–199)
2. Successful responses (200–299)
3. Redirects (300–399)
4. Client errors (400–499)
5. Server errors (500–599)

MIME types in the Web

- Browser uses MIME type to decide what to do with data
- text/html is parsed
- Other data s passed to a plug-in or another app
- A source of conflicts is that multiple plug-ins and helper applications are available for some subtypes, such as video/mpeg. What happens is that the last one to register overwrites the existing association with the MIME type, capturing the type for itself.

Content delivery

- No single server is able to handle today's demand for content, YouTube and other large content providers build their own content distribution networks.
- These networks use data centers spread around the world to serve content to an extremely large number of clients with good performance and availability.
- They use load balancing and caching to make it scalable

Content distribution (or delivery) networks (CDN)

- a provider sets up a distributed collection of machines at locations inside the Internet and uses them to serve content to clients.
- Each of those machines contain a copy of the page that a potential cient may request

404

- The more potential clients the more nodes (machines) have to be set up in the network
- Each client gets good performance by fetching pages from a nearby server (assigned statitically based on the client network, i.e. the ISP) and the total load placed on the network is kept at a minimum
- Another way to support a distribution tree with one level is to use mirroring
  - The CDN nodes are called mirrors, this design lets the user manually select a nearby mirror to use for downloading content.
- The bad approach is to use proxies to redirect users to certain nodes (technically you feed the proxies with a specific node and the cache of the proxy is shared to the people connected to the proxy).
- The actual approach is DNS redirection. The name server is run by the CDN. Instead, of returning the same IP address for each request, it will look at the IP address of the client making the request and return different node addresses (the one closest to the client).
  - The CDN DNS server and CDN origin server (with the content) are different servers
  - "closest" might entail a combination of length, speed, and congestion.

404

- Akamai was the first major CDN and became the industry leader. Other ones are Cloudflare, AT&T or even Peer-to-Peer
  - Companies outsource the CDN to Akami
  - Akami places nodes inside ISPs networks to increase speed and to decrease bandwidth consumption (win-win)

  - Often HTML and low size files are not outsourced to the CDN but things like video, images, audio, etc.
- CDN can quickly scale up a site's serving capacity.

Peer to peer networks

- a collection of computers pool their resources to serve content to each other, without separately provisioned servers or any central point of control
- BitTorrent is the most popular P2P protocol
- there is no dedicated infrastructure, unlike in a CDN. Everyone participates in the task of distributing content, and there is often no central point of control.
- The aggregate upload capacity of the P2P network with N users is N Mbps, which is also the download capacity.
- The approach taken in BitTorrent is for every content provider to create a content description called a torrent and is used by a peer to verify the integrity of the data that it downloads from other peers.
  - The torrent is just a file in a specified format that contains two key kinds of information.
    - the name of a tracker, which is a server that leads peers to the content of the torrent.
    - a list of equal-sized pieces, or chunks, that make up the content.
  - The torrent file contains the name of each chunk, given as a 160-bit SHA-1 hash of the chunk.
- the torrent file is at least three orders of magnitude smaller than the content, so it can be transferred quickly
- To download the content described in a torrent, a peer first contacts the tracker for the torrent. The tracker is a server that maintains a list of all the other peers (swarm) that are actively downloading and uploading the content
  - The members of the swarm contact the tracker regularly to report that they are still active, as well as when they leave the swarm
  - Peers that have already downloaded the full file are called seeders. They only upload
  - People that have not fully downloaded the file do both, upload and download. But some nodes take resources from a system without contributing in kindk, called free-riders or leechers. BitTorrent clients in response reward peers who show good upload behavior.
  - The peers with which a node is currently exchanging chunks are said to be unchoked. this algorithm is intended to match peers with comparable upload and download rates with each other. The more a peer is contributing to the other peers, the more it can expect in return.
  - if a peer is not uploading chunks to other peers, or is doing so very slowly, it will be cut off, or choked

Streaming Audio and Video

- Compressing audio and video files is done in such a way to remove information that is not perceptible to humans, such as some color information and inaudabile audio frequencies for humans.
- By compressing we are reducing bandwidth
- For many applications, a multimedia document will only be encoded once (when it is stored on the multimedia server) but will be decoded thousands of times (when it is played back by customers).
  - It doesnt matter if the encoding algorithm is slow as long as the decoding algorithm is fast.
- On the other hand, for live audio and video, such as a voice-over-IP calls, slow encoding is unacceptable.
- encode/decode process need not be invertible. As opposed to files, with media it's okay if it's not the same as the original as long as it ***looks*** the same. It is said to be lossy.
- the most popular formats are MP3 (MPEG audio layer 3) and AAC (Advanced Audio Coding) as carried in MP4 (MPEG-4) files.

Digital audio

- Audio waves can be converted to digital form by an ADC (Analog-to-Digital Converter).
- An ADC takes an electrical voltage as input and generates a binary number as output
  - The sound frequency is compressed into samples of 4 bits The error introduced by the finite number of nits per sample is called the quantization noise. If it is too large, the ear detects it (unless it is a frequency inaudable for the human ears)
  - Audio compression can be done in two ways. In waveform coding, the signal is transformed mathematically by a Fourier transform into its frequency components. Theo other one is called perceputal coding in which we remove the inaudible parts for human ears. (due to frequency and to sounds masking each other)
- A DAC (Digital-to-Analog Converter) takes digital values and produces an analog electrical voltage.
- The audio compression is done by sampling the waveform at a rate from 8 to 96 kHz for AAC, often at 44.1 kHz, to mimic CD sound. Sampling can be done on one (mono) or two (stereo) channels. Next, the output bit rate is chosen. MP3 can compress a stereo rock 'n roll CD down to 96 kbps with little perceptible loss in quality, for a piano concert, AAC with at least 128 kbps is needed. The difference is because the signalto- noise ratio for rock 'n roll is much higher than for a piano concert (in an engineering sense, anyway). The samples are processed in small batches. Each batch is passed through a bank of digital filters to get frequency bands. The frequency information is fed into a psychoacoustic model to determine the masked frequencies. Then the available bit budget is divided among the bands, with more bits allocated to the bands with the most unmasked spectral power, fewer bits allocated to unmasked bands with less spectral power, and no bits allocated to masked bands. Finally, the bits are encoded using Huffman encoding, which assigns short codes to numbers that appear frequently and long codes to those that occur infrequently.

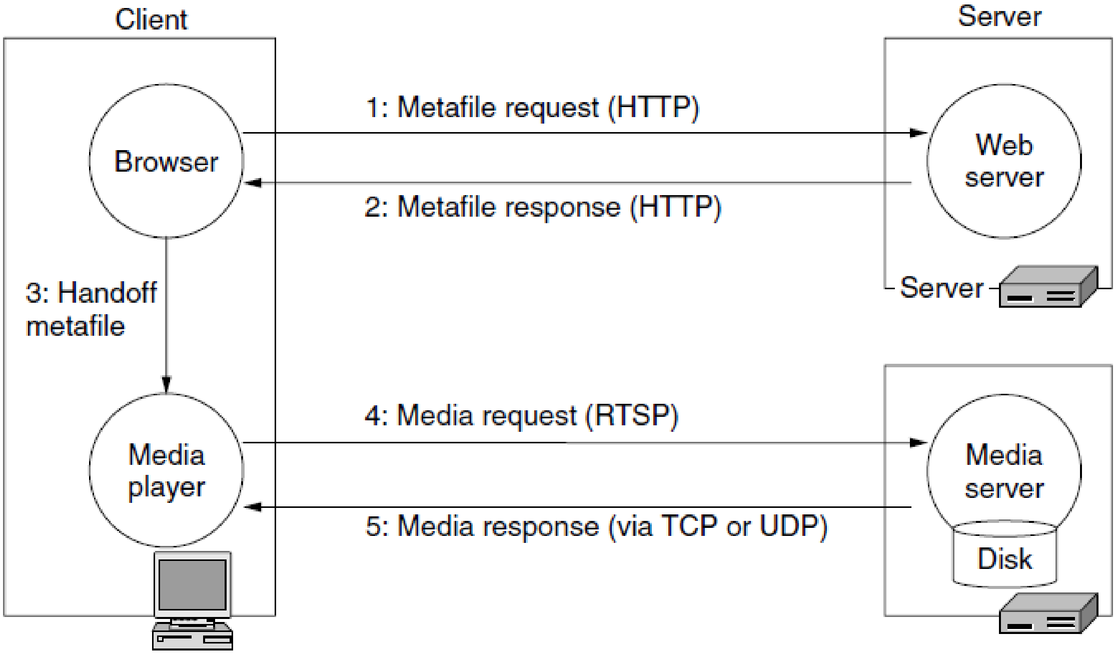
Digital Video

- For digital video many systems use 8 bits for each of the red, green and blue (RGB). With 24 bits per pixel there are 16 million colors, more than what the human eye can distinguish
- Different frame rates are used: 24 fps, 30 fps (NTSC), 25fps (PAL), the latter not good enough fro broadcast television requiriing the images to be split into two fields, one with the odd-numbered scan lines and one with the even-numbered scan lines. Giving rougly 60 fps NTSC or 50 PAL. This is knowno as interlacing.
- Computer video cards have buffers so interlacing is not necessary and they are progressive. They can place the images at a faster rate than the video
- MPEG compresses over a sequence of frames, further using motion tracking to remove temporal redunacancy
  - Intracoded frames are self-contained. (Each frame is a picture)
  - Predictive frames only store changes to previos frames (i.e. backgrounds are skipped)
  - Biderectional frames do prediction of the past and the future
    - Advantage: They can deal with lost frames and replace them with an estimation based on previous frames
- MPEG compresses up to 50 times
- RGB is not the best color model to use for compression. The eye is much more sensitive to the luminance, or brightness, of video signals than the chrominance, or color, of video signals. The following formulas are used for 8-bit values that range from 0 to 255:
  - $Y = 16 + 0.26R + 0.50G + 0.09B$
  - $Cb = 128 + 0.15R - 0.29G - 0.44B$
  - $Cr = 128 + 0.44R - 0.37G + 0.07B$
  - compresses the total amount of data by a factor of two.

Streaming Stored Media

- Back then a client would make an HTTP request to the Youtube server. The youtube server would retrieve the video from their disk, send it via an HTTP response, and then the client

would save the file on its local disk, and finally play it from their disk.
    - Cons: writing to disk is expensive and you might not need the whole video
- A metafile is a file that contains information about a file (i.e. what type of file it is, where it is, etc)
- Nowadays the client makes a Metafile request to the YT server, hands off the metafile to the media player, and then the media player interacts with the media server of youtube via Media rqusts (RTSP) which the YT media server replies either via TCP or UDP with information from the Media disk of YT.
    - RTSP is an extension of HTTP. TCP increases jitter due to transmission errors and starts slow, has overhead, etc.
    - Alternative is to have an error correction in the application layer and let UDP send stuff fast. It still has jitter (not as much as TCP) and also comes with decoding complexity and overhead.
    - Another alternative is interleave media ():
        - Multiple data sources (i.e. one for audio, one for video) can be interleaved (mixed evenly) so that when a burst error appears it affects just a few audio and just a few video frames instead of a whole chunk of just either. By spreading the damage we can still use biredictional compression to fix the lost packages and have a decent experience.



- When it comes to the media buffer you want to set a low-water mark large enough to prevent stalls in the playback (freeze frames) and you also want a large enough high-water mark to prevent the client from running out of buffer space (which also freezes frames as those after a full buffer will be lost)
    - When the low-water mark is hit the protocol will try to increase the download speed
    - When the high-water mark is hit the protocol will try to slow the download speed down

Streaming Live media

- Similar to stored media but you can't stream faster than the live rate to get ahead
    - Therefore you need a larger buffer to absorb jitter (and hence you're watching the football game 30 seconds later than it actually happens)
- Often it has many users consuming the same stream at the same time
    - UDP with multicast greatly improves efficency, but it is rearely available, so many TCP connections are used.
    - Peer-2-peer livestreaming is an alternative

Real-Time Conferencing

- Real-time conferencing requires a buffer of at most 3 seconds to let people feel that they are talking actually live.
  - Furthermore, the connection goes both ways (upload and download)
  - The goal is to decreace the latency by using “adapt compression loss rate”.
    - As your internet speed decreases, the compression is more lossy.

Peer-to-peer streaming (slides)

- Instead of relying on a central infrastructure, users can create their own infrastructure by connecting to each other
- Napster as opposed to BitTorrent, had a central server that redirected users to the IPs of the hosts containing the files
- BitTorrent maps a torrentfile to peers having chunks (it still needs a central authority)
- Newer verisons of BitTorrent use a decentralized tracker thanks to Distributed Hash Tables

Chord Distributed Hash Tables

- Small amount of information to store per node
- Quick look up
- Concurrent use by all users
- We use a ring of 2^m places. Every place can hold a user and to compute the location you use a hash function that uses the IP address as input.
- Nodes keep track of next node with succesor(location + 2^i) so that we can travel around the ring in log(n) time

Blockchains/Distributed Ledgers

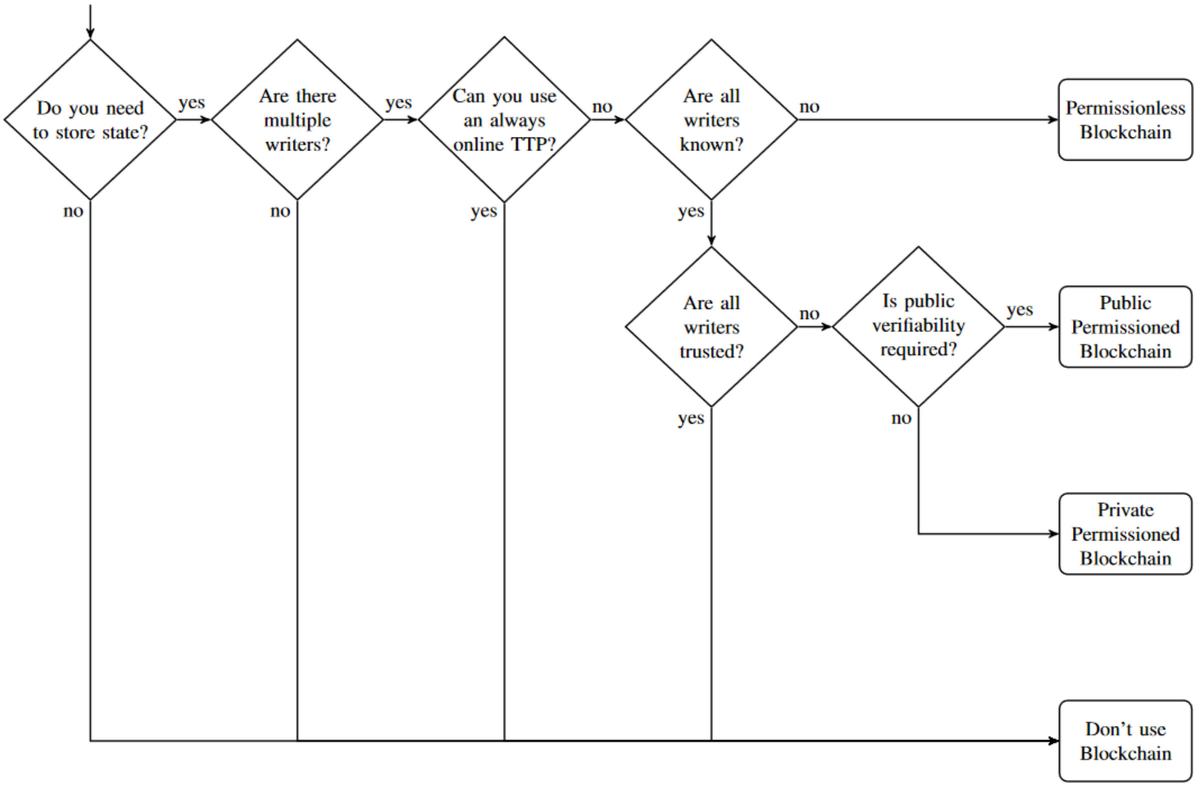
- Blockchains are usually built on top of a P2P network
- Ledger: is a record of transactions
- Distributed ledger: distributed record of transactions requires replication and consesus (and eventual) synchronization
- Only valid transactions are accepted
- Liveness: New valid transactions are eventually added
  - When it’s synchronised, you can specify a timeout in order for a transaction to happen
- As the transactions are added, we can see that there is a chronology, a sequence of total tranasctions
  - We have 1 transaction, then 2 transactions … then n transactions
  - All the nodes store the n transactions
  - In the event of a malicous node with fake transactions, that block is compared to other blocks in the network.
  - The larger the network that accepts a block version wins
  - Then when we have n+1 the node with the highest proof-of-work is chosen as the leader, where the new blockchain is appended.
- Blockchain just refers to the transaction ledgers and they are on top of the application layer (in this case the application would be P2P)
  - The blockchain layer is also known as the consensus layer. While the P2P layer (application) disseminates the information, the consensus layer agrees on a common state
- Some implementations include an Off-chain layer on top: it locally conducts transactions (with blockchain security guarantees), and then only send the final summary back to the consensus layer.

Proof-of-work (Bitcoin)

- Leader: Person who solves computational puzzle (and gets mining payoff)
- The cryptographic hash function is the way to measure amount of computational work
  - It is impossible to crack, only a valid transaction would constitute a reasonable amount of computational work (brute forcing would make you waste your electricity bill for

nothing)
    - The leader is the one who managed to be the first to append the solution to the hash function puzzle (verifying is very fast)
    - The proof of work is directly based on the transactions that appear on the ledger
- Longest chain rule: In case of a draw, eventually the draw will be broken and one blockchain will be longer than the other ones. The longest chain is the chosen one.
- It takes a lot of time to confirm the latest blockchain
- To improve scalability there are two main approaches:
    - Do off-chain as much as possible and then merge back to the blockchain
    - Change the consensus protocol with a more efficient algorithm

Applications of blockchain


Do you need to store state?
yes
no
Are there multiple writers?
yes
no
Can you use an always online TTP?
no
yes
Are all writers known?
no
yes
Permissionless Blockchain
Are all writers trusted?
no
yes
Is public verifiability required?
yes
no
Public Permissioned Blockchain
Private Permissioned Blockchain
Don't use Blockchain

- TTP = trusted third party
- Required for any blockchain: state changes, multiple parties change the state, not always a trusted third party available.
- Permissionless Blockchain: Everybody can join (Bitcoin, Ethereum). The original blockchain model.
- Public permission blockchain: Everybody can read, only authorized parties can write.
- Private premission blockchain: Only authorized parties can read/write.

Security Terminology

- Safety: (sometimes called reliability) is to protect from random failures
- Security: protect from intentional attacks
- Privacy: privacy means an individual should be on control over his data. Also known as the security related to protecting the privacy interests of the individual.
- Trust: depends on the context. Here: trust in the sense of cryptographic protocols:
    - A trusted party is a party that will not behave maliciously. As such it is not necessary to protect a network against misbehavior by such party.
- Anonimity: hiding the identity of an individual

Security goals (CIA)

- Confidentiality: Only authorized parties have access to information/system
- Integrity: It is guaranteed that data has not been manipulated. (includes authentication, nonrepudiation)
- Availability: The system/data is there when you need it.

Disclosing security vulnerabilities

- Non-disclosure: it is legal (unless you were appointed to find the vulnerability and not-disclosuring threatens someones life).
- Full disclosure: you anounce it publicly without consulting the developers first
- Ethical hacking: hacking allowed by the company that hires you to test their vulnerability
  - Pentesting: penetration testing. Testing if a system has vulnerabilities when asked by developer/owner
    - Planning/Reconnaissance
    - Scanning
    - Gaining access
    - Mantaining access
    - Improve system
- Responsible disclosure:
  - Inform company or involve CERT (Computer Emergency Response Team)
  - Set a time until when it has to be fixed
  - Publish after this time (including information fix)
- Coordinated disclosure: Often used synomyously with responsible disclosure. But it actually implies coordination about the deadline between the developers of the vulnerable code, the party discovering and a coordinating authority such as a CERT.

Attacker model

- We assume that both hosts are trusted parties, that the network is malicious, but that there exist a third party that is also trusted
- We try to avoid the reliance on trusted parties as much as possible because if they end up being malicious the system is broken.
- Passive attackers only observe, they do not manipulate
- Active attackers they observe and also modify information
- An external attacker is a third party observant of a connection between 2 hosts
- An internal attacker is one of the endpoints.
- Attackers are also call adversaries
- Local attackers are present in some links of the system
- Global attackers are present in every link of the system
- Static attackers have a constant behaviour
- Adaptive attackers change their behavor overtime (usually based on new information)

Secure protocols

- Information-theoretically secure protocols: Those strong enough to withstand attackers with unlimited computational power
  - Very inefficient
- Computationally secure protocols: Polynomially bounded computational power strong that still gets the job done in practice (and more efficient).
  - Most implemented protocols in practice
- Dolev-Yao model: Attacker is polynomially bounded and can
  - overhear
  - modify
  - drop
  - replay
  - delay
  - create

- The goal is to protect against Dolev-Yao attackers

Cryptography

- Cryptography = encryption = turning meaningful text into seemingly random characters
    - But also concerned with integrity -> intentional changes can be detected.
    - All this formally known as "creating cryptosystems"
- Cryptosystem: set of algorithms with guaranteed security properties.
    - Cipher: cryptosystem used for encryption
        - They make use of 3 algorithms
            - KeyGen: the key generation algorithm
            - Enc: the encryption algorithm
            - Dec: decryption algorithm
            - plaintext: original message M
            - ciphertext: encrypting message M to C
            - prior(M) is prior knowledge of an attacker before the message is sent
            - post(M) the attacker knowledge after sending the message (we want that observing the ciphertext does not increase post(M) knolwedge of the message with regards to prior(M) besides the length).
        - Enc(K1,M) = $Enc_{K1}(M)$ = encription of M using encryption key K1
            - It assumes a passive adersary (wont change the message) who aims to break confidentiality. The only info we cant do nothing about is about the attacker sensing the length of the message.
        - Dec(K2,M) = $Dec_{K2}(M)$ = decription back to M using decryption key K2
- Cryptanalysis: research into breaking cryptosystems (without knowing the key).
    - Breaking: just knowing parts of the message can be considered as breaking.
- Cryptology: research of cryptosystem + cyrptanalysis
- Cryptographic key: concrete algorithm to use for the key.
- Kerckhoffs principle: the security of a cryptosystem should not depdend on the secrecy of algorithms, rather on the keys. As changing keys is easy, accidentally revealing a key can be fixed by generating a new one quickly. To generate a new algorithm from scratch is time-consuming.

Caesar cipher (stream and substitution cipher)

- Replace each letter of the message by letter i + (k mod n) with n being the letters of the alphabet
- with high probability the most frequent letter of the encrypted text is the most frequent letter of the original language of the original message. Thus easy to crack the value of k from here.

Symmetric-key encryption

- also referred to as secret-key encryption, we distinguish among two dymensions: stream vs block ciphers, and substitution vs transportation (or permutation) ciphers.
- Only uses one key K = K1 = K2 for encryption and decryption
- All symmetric-key encryption algorithms share the need for a secure channel to exchange keys.
- Usually assymetric key encryption is applied for the exchange of keys whereas the actual communication then uses symmetric-key encryption with the previously exchanged

Stream ciphers

- Act on each character of the plaintext individually. Tipically the characters are bits rather than letters.
- Example: **One-time Pad** (OTP). The exchanged key has at least the same length as the message to be sent. The message is encrypted with C = M XOR K. That is, by "xoring" the key

onto the message. Which can be decrypted by doing the exact same operation.

- They rely on pseudo-random number generators (PRNG) by taking a seed that outputs always the same deterministic number for the same seed.
  - Newer versions agree on a key whose length correspondes to to the seed length of PRNG
  - A nonce is a random number or character string used only once for randomization. This allows us to mantain the same key for various messages (otherwise patterns can be discovered easily by comparing messages). The seed for the PRNG is S = Key XOR nonce
  - Then the encryption of the message is actually C = M XOR KS and the decruption is M = C XOR KS

Block ciphers

- Divide the plain text in blocks of equal length and then act on each block
- It operates on blocks of a fixed length l.
- The plaintext is divides into blocks such that the i_{th} block corresponds to bit $(i-1)\cdot l+1$ to $i \cdot l$ (starting to count at 0 from the left).
- Padding ensures that the plaintext length is a multiple of l.
- For **Electronic Codebook Mode** it encodes and decodes each block seperately, the final decoding is just the concatenation of the each block in the received order. The details of how the Block Encryption algorithm uses the key is simplified with just B-Enc(Key,Plain Text Block)
  - This leaves patterns in the ciphertext and hence leads to security issues. Blocks that are identical have the same ciphertext. Which means that an exposed block is exposed in future messages too.
- **Cipherblock chaining mode** is a mode that ensures that identical plaintext blocks have different encryptions. This is done by implimenting "xoring" a nonce to the first block, and then the "xoring" the ciphertext of the previous block for all subsequent blocks. Formally:
  - $$C_1 = \text{B-Enc}(K, M_1) XOR \text{ nonce}$$
  - $$\text{for n} > 1: \ C_n = \text{B-Enc}(K, M_1) XOR\, C_{n-1}$$
  - Decryption:
  - $$M_1 = \text{B-Dec}(K, C_1) XOR \text{ nonce}$$
  - $$\text{for n} > 1: \ M_n = \text{B-Dec}(K, C_n) XOR\, C_{n-1}$$
  - The inconvinence is that encryption cannot be done in parallel
- **Counter mode** is the secre and parallelizable version of cipherblock chahining.
  - Assumes block length l is even
  - Requires nonce of length l/2
  - the nonce is concatenated with l/2-bit binary representation of i-1
  - Then encrypts the concatenation with key K and "xors" M_i to derive C_i. Formally:
    - $$C_i = M_i \ XOR \, \text{B-Enc}(K, nonce, (i-1))$$
    - $$M_i = C_i \ XOR \, \text{B-Dec}(K, nonce, (i-1))$$
- B-Enc is a combination of P-boxes (transposition ciphers) and S-boxes (substitution ciphers).
- The first widely used block encryption algorithm was DES (Data Encryption Standard). 16 combinations of S and P boxes but with a 64 length key with 8 redundant so 56 length key which can be solved by bruteforce.
  - The succesor of DES is AES (Advanced Encryption Standard).

Substitution ciphers

- Replaces a character or block with another character or block

Transposition cipher

- Changes the order of characters
- Generally only used in combination with substitution. This is because transposition without substitution mantains the same frequency of letters of the plaintext, which means they can be broken using linguistic properties.
  - At the very least we can determine if a certain word was not part of the plaintext (if we cannot find all the letters for such word).

Asymmetric-key encryption

- It uses different keys for encryption and decryption.
- The encryption key is called public key and denoted K_pub.
- The decryption key is only known to its owner and denoted by $Kpri$ aka private key.
- To communicate, the user that wants to recive encrypted messages must first generate a key pair consisting of a public key and a private key.
- Messages encrypted with the public key can only be decrypted using the private key.
- The user must publish the public key on a key server dedicated to storing people's public keys.
- The sender can lookup the public key in the server and encrypt the message accordingly such that $C = Enc(K_{pub}, M)$.
- The receiver decrypts the message with $M = Dec(K_{pri}, C)$.
- The challeng is to make algorithm such that the attacker cannot derive the private key from the public key
- RSA is both the oldest and most used asymmetric-key algorithm
  - Relies on the difficulty of **factorizing large numbers**
  - $$d \cdot e = y \cdot \phi(n) + 1$$
    - with phi(n) the number of totatives of n (something related to primes)
    - d an integer related to primes too
    - e is the number we want to use
    - y is another integer
  - This is determenistic, so in real world implementations a nonce is used somewhere in the process
  - It is computationally secure under the assumption that it is computationally unfeasible to
    - factorize n
    - find the e-th root of an arbitrary number modulo n, and derive other things
- Assymetric key encryption is applied often just to then move to symetric keys because RSA requires long keys and computationally expensive operations such as exponentiation, which entail slow computation.

Hybrid encryption

Combines an asymmetric-key algorithm with a symmetric-key algorithm:

1. looks up the receiver public key
2. chooses a key for the symmetric key encryption
3. encrypts the symmetric key with the public key such that $C_1 = Enc^A(K_{pub}, K_{sym})$
4. encrypts the message M with K_{sym} such that $C_2 = Enc^S(K_{sym}, M)$
5. sender sends C which is a concatenation of C_1, C_2
6. The receiver decrypts the symmetric key using his private key (which fits the encryption with the public key) $K_{sym} = Dec^A(K_{pri}, C_1)$
7. With the decrypted symmetric key the reciever can decrypt the message: $M = Dec^S(K, C_2)$

Diffie-Hellman Key Exchange (symmetric)

- In contrast to hybrid encryption, DH is interactive and both parties need to be online
- Like the RSA, assumes that a mathematical problem cannot be solved with polynomial-time algorithms, namely the **Discrete Log Problem**
- n and g are publicly known
- a and b are chosen by the sender and the reciever respectively and they are both an element of the set $Z_n = \{g^k : k \in \{1 \dots n\}\}$
- The sender sends $A = g^a \bmod n$
- The receiver responses with $B = g^b \bmod n$
- The sender computes $K = B^a \bmod n = g^{ba}$
- The receiver computes $K = A^b \bmod n = g^{ab}$
- Now both have a shared secret key

Message Authentication Codes (MAC)

- Sometimes called message integrity code (MIC), it appends a tag with the following properties to a message:
  - Message integrity: The receiver must be able to detect any modification made to the message by a third party
  - Authentication: The reciver must be able to prove (for himself) that a specific sender sent the message
  - Repudiation: A third party cannot prove who sent the message as both sender a reciver share the same key
- MAC relies on the use of a key, symmetric or asymetric and the sender sends both the message M and MAC(K,M).
- The receiver then can check MAC(K,M) himself again to see if it matches the one sent by the sender. If so, the receiver can assume that the message has not been tampered and that it is from the given sender, else the receiver understand that the message has been tampered and discards it.
- Repudation is guaranteed if the assumption that only the receiver and the sender know the key and that it cannot be tell whether the message and tag was sent by either of them.
  - Because of this we cannot use them to prove that a communication (either from A to B or from B to A or from B to B or A to A) took place. (we can still use them to ensure that there was no tampering)
- MAC uses cryptographic hash functions to achieve integrity: It maps inputs of arbitrary length to fixed length outputs and it has the following properties
  - Collision-resistance (very hard to find an alternative input with the same hash output): It is computationally infeasible to find a $hash(x) = hash(y)$ for some x and y such that $x \neq y$
  - Preimage resistance (very hard to find the input of a hash function just from the output): Given z, it is computationally infeasible to find x with h(x) = z.
  - Second preimage resistance (even if we know one input and one output, it is still impossible to find out another input with the same output): Given x,z = h(x) it is computationally infeasible to find $y \neq x$ with h(y)=z.
  - Common cryptographic hash functions are MD5, SHA1, SHA2, and SHA3.
    - For MD5 and SHA1 collisions have been found, so they are discarded now.

Digital signatures

- Like MACs they append tags to messages however it also ensusures **non-repudation**: everyone can prove that a sender sent the message to a receiver.
- Digital signatures use asymmetric-key cryptography:
  - Sender generates a key pair: public verification key and a private signature key.
  - Sender generates the signature $s = Sign(K_{pri}, M)$
  - Sender sends M and s to the receiver
  - Everybody can compute $Verify(K_{pub}, M, s)$, which does not expose K_{pri}.

- To acheive integrity, authentication and non-repudation the digital signature algorithm must gurantee that it is computationally infeasible to compute a signature without knowledge of the private key. (RSA achieves it)

Hash then sign

- Reliance on asymetric-key encryption is slow. A faster alternative would be to hash the message with a publicly known cryptographic hash function and if the hash of the receive message does not match then we can discard the message and skip the signature (slow) process.

Public Key Infrastructure (PKI)

- Ensures that adversarial parties cannot easily replace the keys of other users.
- PKI typically use certificate authorities that follow a hierearchical structure.
- Root autorities act as trusted parties.
- Each root authority has a public key par consisting of a public verification key and a private signature key. These keys can be shipped directly onto the operating systems of users.
- A root authority can sign the public keys of other users
- **certificate**: The combination of the public key, the identifying information, and the signature.
- Users with certificates can act as certificate authorities as well (and thus sign the public keys of other users, who can again sign the public keys of more users)
- **First-level certificate authorities**: those whose certificates have been signed directly by a root authority.
    - To verify you check the correctnes of the authority's signature on their certificate (the root authorities are already shipped in your OS so just check if they match)
- **Second-level certificate authorities**: those whose certificate is signed by a first-level certificate authority.
    - It has to present both: its own certificate and the certificate of the first-level autority that signed them.
    - For any k-level certificate authority, we start verifying from the root down to the last levels. This is known as the **certificate chain**
    - If any veryification step fails then the certificate is rejected
- The most common certificate format is **X.509**. It specifies:
    - version number
    - used signature algorithm
    - expiry date
- **web-of-trust**: The alternative in the abscence of trusted parties that can act as roots.
    - People sign public keys of participants they know.
    - Users verify a new public key by checking if it has been signed by one or more public keys that they have already signed themselves or by a public key they consider legitimate.

Secure Wireless Communication

- Securing a wireless network entails:
    - Confidentiality: Nobody but client and access point should be able to access the content of the communication
    - Integrity: Nobody should be able to modify communication between the access point and the client
    - Access Control: Only authorized users should be able to use the network

WEP (Wired Equivalent Privacy)

- First protocol aiming to protect wireless channels. It was a failure.
- WEP uses challenge-response protocol for access control, a cryptographically weak stream cipher called RC4 and a Cyclic Redundancy Check for integrity.

- All the users of the wireless network receive the same password (a secret key of 40 bits).
- User sends connection request to access point
- Access point replies with a random bit string called the challenge
- User concatenates the 40-bit key with a nonce of 24 bits.
- Then the key and nonce are used as input for RC4 to generate a keystream of the same length as the challenge
- The user sends nonce XOR keystream and another nonce' to the access point.
- The access point uses that nonce' and the keystream to generate the keystream and corresponding XOR.
- If the access point agrees with the recieved XOR it grants the user with access to the network

- An attacker can just observe the access point sending the challenge and the user sending the challenge XOR keystream (and a nonce'), that is c XOR s. For which the attacker can just compute the keystream with c XOR (c XOR s) = s.
  - Then the attacker can contact the access point legitimately and return the same nonce for which it cracked the keystream before
- Moving onto another case, let a (not necessarily bad) authenticated user and access point talk to each other. In WEP:
  - User computes CRC(M) and appends it to the the original message
  - The user chooses a nonce and computes the RC4 with the key and the nonce appended to obtain a keystream of the same length as the Message with the CRC(M) appended.
  - The user will send to the access point M || CRC(M) XOR keystream and the nonce
- The protocol fails confidentaility and integrity for the reasons:
  - too short key length (40 bits) which can be bruteforced
  - the presudo-randmo number generator in RC4 is weak because enables the partial reconstruction of original messages and keys from the ciphertext due to:
    - predictable patterns in the keystream
    - correlations between keystream and key
  - The lack of integrity can be attributed from the use of a CRC:
    - The CRC is a linear operation where CRC(M XOR N) = CRC(M) XOR CRC (N)
    - We can change the message without being detected and determine the respective lengths of M and CRC(M) from C
- The fact that it uses 1 global password is also a practical problem
  - There's high chance of leak (and hard to find the leaker)
  - It's cumberstome to revoke access

Wi-Fi protected access (WPA)

- WPA was a short-term replacement for WEP
- It used a MAC instead of a CRC
- It uses encryption protocol with 128 bits
- WPA2 is the carefully researched successor of WPA
  - Relies on AES as an encryption protocol
  - Uses a different interactively generated key for each message after the initial 4-way handshake
  - Enterprise mode allows to assign users individual passowrds
  - Replay attacks during the handshake are possible, WPA3 patched that, but it is still prone to other attacks.

Virtual Private Netowrks

The private network should ensure:

- Encryption and authentication of the communication between two private networks in physically different places where its communication has to traverse untrusted networks
- Encryption and authentication of the communication with authorized parties outside of the organization's network

- Sending content via an untrusted network reveals some information even if the communication is encrypted and anonymized.
- VPNs focus on encryption and authentication, to achieve them most rely on IPSec.

IPSec

- Provides security for IP packets (both IPv4 and IPv6)
- Uses symmetric-key encryption for confidentiality
- Uses MAC (message authentication codes) for integrity (HMAC, with SHA1 or SHA2 or a block cipher with a mode that combines intregrity and confidentiality)
- The encryption algorithm is either a block cipher (AES, TripleDES), or a stream cipher (typically Chacha20).
- Gateways play a key role, as in a protected and private network, gateways process all traffic from and to the outside. They also act as a firewall and identify undesired traffic.
- IPSec has multiple depending on the role of teh communication partners:
    - Transport mode: handles communication between two non-gateways hosts that are not in any private network or one gateway and a hosts outside of the private network.
        - Only protects the payload of the packet
    - Tunnel mode: addresses the communication betwen two gateways in different private networks
- IPSec uses the following components:
    - Security Associations (SA): Specify the key exchange protocol and the paramaters used to achieve confidentiality and integrity
    - Authenticated Header (AH): provides integrity of the IP header and payload
    - Encapsulated Security Payload (ESP): provides confidentiality and integrity for an IP packet
        - To apply ESP we must have already shared keys and there are difference between the modes.
            - Travel mode can only encrypt the payload but not the header:
                - Sender inserts an ESP header with IPSec related header fields between the IP header and the payload.
                - Sender encrypts the payload and appends a message authentication code to the payload.
                - Sender modifies the header fields that are affected by the changes to the payload (packet size)
                - Any router on the path will consider the combination of ESP header, encrypted payload, and MAC as the payload of the modified packet
            - Tunnel mode can protect the complete IP packet (thus also the header): Sender creates new IP header and sends it to the receiver consisting of:
                - the new IP header
                - ESP header
                - encrypted old IP header
                - encrypted payload
                - MAC that guarantes integrity of both old IP header and payload

IPSec-based VPNs

- VPNs leverage IPSec to ensure confidentiality and integrity
- VPNs use both transport and tunnel mode
- VPNs use transport mode when a user outside the private network communicates with hosts inside the private network. Assuming a successful key exchange:
    - The private network is set up such that all traffic goes via the gateway.From outside user to inside host we have:
        - The gateway checks the MAC from the users outside the network and decrypts the payload
        - The gateway forwards the packet to the host in the network for which it looks like a normal IP packet

  - From inside host to outside users we have:
    - The host applies the IPsec transport mode steps
    - The user outside the network verifies and decrypts the data
- VPNs use tunnel mode when both users are inside the private network but often in physically different location such that the communication has to traverse the untrusted Internet
  - Each host uses a different gateway
  - When the first host wants to contact the second hone, it sends a normal IP packet with a header and a payload
  - His gatway inspects the packet and modifies it with using IPSec’s tunnel mode
    - IP address source of modified packet is the first gatway and the destination is the second gateway
    - This way the IP address of both hosts are hidden under the gatways
  - When the second gateway receives the packet, it will check the MAC, decrypt the packet back to the original IP packet and forward it to the second host

Transport Layer Security (TLS, SSL originally (Secure Socket Layer))

- provides confidentiality and integrity for communication between clients and web servers by authenticating servers
- can also authenticate clients but the option is barely used outside server-to-server communication
- runs on top of a reliable transport layer protocol, often TCP
- HTTPS is the combination of HTTP and TLS arguably the most widely used application layer protocol as it is seamlessly integrated into browsers
- Hirearchical public key infrastructure with multiple root certificate authorities enables server authentication
- Servers are required to buy a certificate to use TLS (which is assigned to the domain name)
- TLS establishes a connection between a user and a server by first executing a handshake protocol to estabish the identity of the server and agree upon cryptographic algorithms and keys
  - The client first sends a connection request, which includes its TLS version and the different ciphersuites (encryption algorithms and protocols supported)
  - The server replies with the chosen ciphersuite (the best available by both parties) and its certificcate chain
  - THe user validates tbe certificate chain and exchange symmetric keys for encryption and message authentication

Secure DNS

- Regular DNS is highly vulnerable to DNS spoofing and DNS poisoning attacks
- A DNS spoofing (claiming that an object is a different object) occurs when an attacker claims an IP address corresponds to a domain name without that being the case.
- With a poisoning attack (corrupting the records of an honest entity) on a DNS request by the victim, the attacker can modify the response and redirect the victims to his own malicious server IP. The attacker only needs to succeed once as the victims own cache will store the malicious address.
  - This could be done when the DNS server doesnt have the server on its cache and the attacker sends a fake address (because if it does it will not make a request to a DNS server that the attacker can intercept)
- If the actual web server uses TLS the web server (now the bad one) will not be able to provide the necessary certificate chain and the user will be warned by the browser
- DNSSEC uses a hirearchical PKI to enable authentication of domains. It mirrors tehe hierarchical sturcture of DNS:
  - Name servers for one domain sign the public keys of all servers within their domain
- To rettrieve DNS certificate DNSSEC adds new record types:
  - RRSIG: resource record signature (signature over a set of DNS records)

  - DNSKEY: public key that should be used to check signatures
- DNNSEC only provides authentication, but not confidentiality
- DNNSEC is not as widely adopted as TLS (one reason is backward compatibility)

Pretty Good Privacy (PGP)

- Is a program for email security that relies on
  - hybrid encryption for confidentiality
    - The user has a public encryption K_{ENC}^{U} and a private decryption key K_{DEC}^{U}
    - It makes use of a cryptographic hash function and a symmetric key encryption algorithm with for both the encryption and the deryption
  - digital signatures for integrity and authentication (hash-then-sign)
    - The user has the public verification key K_{VERIFY}^{U} and the private signature key K_{SIGN}^{U}
- It is not advised to use the same key for encryption and signatures, although it is possible.
  - Signature keys should be kept long term
  - Encryption keys should be renewed often
- Public verification key is linked to the identity
- If the secret signing key leaks, the user can revoke the key and all signatures created after revocation are invalid.
- But if the user leaks a long-term decryption key, the attacar can still utilize it by decrypting previous messages
  - The longer an encryption key is used the larger the potential for damage
- PGP steps after both parties know each others public keys:

The sender:

1. computes hash(M)
2. computes s - Sign(hash(M),K_{SIGN}^{U}) using the sender's private signature key
3. chooses a key and encrypts it such that $C_1 = Enc^A(K_{ENC}^{B}, K)$
4. computes $C_2 = Enc^S(K, M, s)$ which the encryption of the concatenation of M and the signature s under the symmetric key
5. sends both C_1, C_2

The receiver:

1. decrypts the secret key $K = Dec^A(K_{DEC}^{B}, C_1)$
2. decrypts the concatenation M,s = $Dec^S(K, C_2)$
3. computes hash(M)
4. verifies the message using A's public verification key $Verify(K_{VERIFY}^{U}, hash(M), s)$
5. If the verification succeeds, accepts the message

- In contrast to TLS, PGP relies on Web-of-trust as decentrlized PKI

User authentication

- Strong client authentication ouuld rely on multi-factor authentication, which should at least have 2 of the following:
  - Something you know: passwords, pins…
  - Something you have: phone number, credit card, physical keys, 2-step verification apps…
  - Something you are (biometrics): finger prints, iris scan, face recognition…
- Additional weaker factor is somewhere you are, which often uses the country associated with an IP address, however this is mostly used to detect irregularities such as unusual places the user logs in from