Tutorial 1 – Performance Metrics

- **Bandwidth** (Max throughput MB/s)
- **Throughput** (data transmitted per unit MB/s): file size / packet delivery time
    - File size (MB): entire file size
    - **Packet Delivery Time** (s): transmission time + Single Trip Time (depends per context)
        - Single Trip Time (s): called latency in this course
        - Sometimes they may ask the packet delivery time of the entire file, sometimes per frame.
        - **transmission time** (s): Time it takes to prepare the frames (sometimes they ask all the frames, sometimes per frame):
          (optional buffers) + file size / transmission rate
            - File size (MB): entire file size
            - **Transmission rate** (MB/s): maximum amount of bits that can be sent per unit of time on the datalink layer. (can be the bandwidth)

Formal definitions

- Bandwidth: amount of total data that can maximally be sent via a connection per time unit
- Throughput: Amount of total data actually transmitted per time unit
- Time to transmit a message depends on:
    - Message size
    - Latency
    - Bandwidth or throughput (whichever is used in the problem)
        - Bandwidth is the theoretical maximum load a connection can carry
        - Throughput is the actual load expressed as the average value (because in practice throughput is highly variable)
- Latency: Single trip time of one bit from A to B
- Roundtrip time: latency from A to B + latency from B to A + (optional processing time on B)
- Transmission time: amount of time required for the sender to push all bits of the message into the communication link
- Packet delivery time: amount of time from when the sender sends the first bit until when the receiver receives the last bit

Latency

- The latency from A to B is known to be 214 milliseconds.
- A sends a PING message to B.
- It takes 67 milliseconds for B to process the message and prepare a PONG message back to A.
- Successfully receiving the PONG message, A measures the RTT as 501 milliseconds.

What is the latency from B to A

- Round Trip Time = Latency A to B + Processing Time B + Latency B to A
- 501 = 214 + 67 + x
- x = 501 – 214 – 67 = 220

Throughput, packet delivery, transmission time

- The latency from A to B is 60 ms.
- A sends a file of size 128 MB to B.
- A and B use a communication protocol that sends data using 1MB chunks.
    - The file of 128 MB will be split into 128MB / 1MB = 128 chunks
- The sender process halts for 10 ms after sending each 1MB chunk in order to load the next chunk from the buffer.
    - The first chunk is always ready, so there are 128 -1 = 127 buffers of 10ms each totaling 1270ms of buffer time
- The communication channel has a bandwidth of 64 MB/s and it is dedicated to the communication between A and B (They use full bandwidth).
    - Transmission rate = 64 MB/s

What is the transmission time of the file?

- Transmission time = buffer time + file size / transmission rate = 1270ms + 128MB / 64MB/s = 3270ms

What is the packet delivery time?

- Packet delivery time = transmission time + latency (which is single trip time) = 3270ms + 60ms = 3330ms

What is the throughput between A and B?

- Throughput = file size / packet delivery time = 128MB / 3.300s = 38.44 MB/s

Overhead

- They use a communication protocol that sends the data in 4MB chunks and add the following information to each chunk:
    - Chunk number (2 kilobytes)
    - ID of the sender (4 kilobytes)
    - ID of the receiver (4 kilobytes)
- A wants to send a 128MB file to B
    - In total there are 128 MB / 4MB = 32 chunks

What is the communication overhead of the used protocol?

- Each chunk has 2 + 4 + 4 = 10 KB of Overhead
- Overhead could be expressed as 10 KB / 4MB = 10 KB / 4096 KB = 0.24%

What is the total amount of data transferred?

- The total amount of data transferred is 10 KB of overhead * 32 chunks + 128 MB = 320 KB + (128 * 1024) KB = 131392 KB
- Overhead could be expressed as 320 KB / 131392 KB = 0.24 %

Packet loss

- Assume that an extensive experiment showed that for every 1000 data packets sent from A to B
    - two packets (on average) fail to reach their destination.
        - That is, 2/1000 packet loss
    - A wants to send 100 data packets to B

What is the probability that all 100 data packets reach their destination successfully (assume packet transmission success is an independent event)

- Probability of failure is 2/1000 = 0.002
- Probability of success is 1 – 0.002 = 0.998
- Probability of 100 successes is 0.998^100 = 0.819

True/False

- Low jitter is more desirable in video conferencing than in web browsing.
  - True
- In case of low latency, throughput of a link can exceed its bandwidth.
  - False
- Data overhead may be greater than 100%.
  - True

Tutorial 2 – Error detection

Parity bits

Assuming an even parity, which of them are guaranteed to be wrong?

0010 1010 1010 -> 5 ones = odd = for sure error

0011 1111 1111 - > 10 ones = even = can't tell (might have flipped even amount of times)

0010 1000 1101 -> 5 ones = odd = for sure error

0001 0110 0010 -> 4 ones = even = can't tell

For the errors, can you tell the correct version of the messages?

- No, as we don't exactly know the number of flips nor where they occurred

Parity blocks

the protocol uses even parity. The receiver receives a bit sequence and constructs the parity block:

1 0 1 1 0 1 0 | 0 even
0 0 0 1 0 1 1 | 1 even
1 0 **0** 0 1 0 **1** | 1 even
0 1 1 1 0 1 1 | 1 even
0 1 0 0 1 0 1 | 1 even
1 1 **1** 1 1 0 **0** | 1 even

_ _ _ _ _ _ _ _
1 1 1 0 1 1 0 | 1
e e e e e e e even

Does the receiver find any error?

- As a receiver compute the parity of each row and each column
  - It has to match the outside numbers, alternatively take them into account too and then check that they have the desired parity (even in this case)

Did the receiver receive the bit sequency correctly?

- We cannot be sure, as there might have been odd flips in columns and rows

- They have to make sense, need to provide a counter example (bold with borders)

Cyclic redundancy checks

- Alice wants to send the data 110 100 011 to Bob.
- For error detection, they decided to use a CRC with the generator 1101.

Compute the message that Alice sends.

- Generator has 4 digits so we must append three 0s
- Now the data with the appended 0s will be "long divided" by the generator, and then we replace the appended 0s with the module:
  - No carries
  - Substraction and addiction are XOR
  - Only track remaidners (no need to track quotient at each division)
  - 1101 0001 1000 "%" 1101 = 10
  - Alice sends 110 100 011 010

Compute if there was an error

- You received a message 1100 1110 0111 1000 01.
- You know that CRC with a generator polynomial of 110010 was applied to the data before sending.
  - 1100 1110 0111 1000 01 % 110010 != 0

Tutorial 3 – Error detection, Correction and Control

Checksum Generation

- Alice and Bob decided to use 8-bit checksum **with 1's complement** for error detection in their communications.
- Alice wants to send the following message: 11001011 11001000 10111011 10110101

What is the checksum of the message?

- Apply consecutive XOR operations on the streams and an additional XOR operation with a bit stream of 8 ones (11111111).
  - Result: 11110010

Checksum error detection

- Using the same protocol as in the previous question, Bob receives the following message from Alice: 10011011 00011000 10111011 10110101 $01110010_{(2)}$

Does Bob detect an error?

- It's 1 complements: so we expect the XOR of all streams to be all 1's: which is the case
  - However we cannot be sure that it's error free, as there might have been 2 or more bit flips on the same column

Hamming codes

- Hamming codes inserts parity bits into the message on powers of 2 cells (1, 2, 4, 8, 16…)

- The number of parity bits is that one such that p1+p2+p4… for all parity bits should add up to the length of the message
- Parity 1 checks the parity of odd numbers (check, skip, check, skip, check…)
- Parity 2: (check, check, skip, skip, check, check, skip, skip…)
- Formally, parity n checks the parity of binary cell number whose k bit is ON
    - i.e. p4 is 4 (100) so it checks
    - 100 = 4
    - 101 = 5
    - 110 = 6
    - 111 = 7
    - 1 100 = 12
    - 1 101 = 13
    - 1 110 = 14
    - 1 111 = 15
    - 10 100 = 20
    - 10 101 = 21
    - 10 110 = 22
    - 10 111 = 23
- The bit that regards itself can be interpreted as x, and adjust it such that the XOR of all bits is even or odd depending on the protocol specification
- As a receiver manually check the parity of all bits again, and if px and py are flipped x+y = d(x+y) which is the bad bit

Compute an (11,7) Hamming Code for 1010 110 using even parity

- Here they already say the length of the data and the amount of parity bits + data (11)
- Result: 0 1 1 1 0 1 0 0 1 1 0

Using odd parity, we receive: 0001 1011 1010 001. What should the original message (including parity bits) be if there was at most one error?

- There is an error in p4 and p8 so 4+8 =12 it means d12 has to be flipped: 0001 1011 1011 001

Sliding windows protocol

- In a connection between two hosts, the sender maintains a connection with the receiver where the window size determines the amount of frames that are kept in memory so that they can be acknowledged at once and increase the (good) throughput of the connection.
- 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.
- round trip time from earth to satellite is 540 msec.

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

- Bandwidth 64kbps = 64000 bps = 64 bpms
- 512 byte data frames = 512 * 8 = 4096 bits
- RTT is 540 ms
- 0 buffer time

Window size 1

- Transmission time (of 1 way frame) =
  0 buffer time + frame size / transmission rate = 4096 bits / 64 bpmsec = 64 msec
- Window size 1, uses 1 ack per frame so transmission delay would be RTT
  - The ack itself is included in the RTT
  - Packet delivery time = transmission time + RTT = 64 msec + 540 msec = 604 msec = .604s
- Throughput = frame size / packet delivery time = 4096 bits / .604s = 6781 bps

Window size 7

- Transmission time (of 1 way 7 frames) = 7 * 64 msec = 448 msec
- Window size 7, uses 1 ack per 7 frames.
  - 7 frames take 448 msec
  - Until we receive the ACK we can't do nothing. The ack takes 604ms.
  - Calculate transmission time – packet delivery time window size 1
    - 448 – 604
    - If negative take the packet delivery time of window size 1
    - Else it means that the transmission time is greater than the time required for the arrival of the ACK to the first data frame
  - Packet delivery time = 604 msec
- Throughput = 7 * frame size/packet delivery time = 7*4096 bits/.604 = 47470 bps

Window size 15

- Transmission time (of 1 way 15 frames) = 15 * 64 msec = 960 msec
- Window size 15, uses 1 ack per 15 frames.
  - Calculate transmission time – packet delivery time window size 1
    - 960 – 604 = 356
    - The sender can make full use of the channel (does not need to wait for an ACK, the ACK has already arrived). Throughput = bandwidth = 64kbps

Window size 15+

- Throughput = bandwidth (as explained by window size 15)

Assignment 1 – Data Link Layer

Hamming distance

- Codeword is a piece of data (payload) together with redundant information (overhead)
- Hamming distance between two code words is the number of bit positions that differ
- A code is a set of codewords
- The hamming distance of an entire code is the smallest hamming distance between all combinations of codewords
- To detect k errors, you need a code with a minimal distance of k + 1
- To correct k errors, you need a code with a minimal distance of 2k + 1

Java implementation:

```
/**
```

```
 * Calculates the hamming distance of the given code, or returns -1 if it cannot be
calculated.
 *
 * @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; //dummy value to check if min has been overwritten
    for (Long currentWord : code) { // for each word
        for (Long otherWord : code) { // compare it with all the words
            if (currentWord != otherWord) { //excluding itself
                long xor = currentWord ^ otherWord; // compute the XOR
                long count = Long.bitCount(xor); // compute the number of bits flips
(hamming distance)
                if (count < min) { // update min when the hamming distance is smaller
than the prior smallest
                    min = count;
                }
            }
        }
    }
    return min == Long.MAX_VALUE ? -1 : min; //return min if at least once updated,
else couldn't calculate (-1)
}
```

CRC

The Cyclic Redundancy Check makes use of binary division/polynomial division to calculate a remainder, that is used as the redundant information for the error detection. The divisor is a special polynomial called a generator. This generator determines the properties of the CRC. The amount of single-bit flip errors that a CRC can reliably detect therefore depends on the order of the generator.
Both sender and receiver know the generator. The sender computes a check value using binary division. The receiver performs the binary division with the same generator and compares their result to the check value. Although CRC is widely used, it's not secure against deliberate attacks (i.e. multiples of the generator XORed)

Java Implementation

```
public class CRC {
```

```
    /**
     * Calculates the CRC check value
     *
     * @param bitSequence       The input bit sequence
     * @param inputLength       The length of the input bit sequence (including
possible leading zeros)
     * @param generatorSequence The generator bit sequence
     * @return The CRC check value
     */
    static long calculate(long bitSequence, int inputLength, long generatorSequence)
{
        //Get the generator length so we can then append that number -1 of 0s to the
bitSequence
        int generatorLength = (int) (Math.log(generatorSequence) / Math.log(2) + 1);


        //The sequence will be updated each iteration, called remainingSequence as it
is the reminder of each (sub)division
        long remainingSequence = bitSequence << generatorLength - 1;


        // Get the length of the remaining sequence so we now how much to left shift
the generator such that both
        // the remainingSequence and shiftedGenerator have the most significant bits
in the same place
        // so that they can be XORed as in a long bit division
        int remainingSequenceLength = (int) (Math.log(remainingSequence) /
Math.log(2) + 1);


        // Shift the generator to match the most significant bits.
        long shiftedGenerator = generatorSequence << remainingSequenceLength -
generatorLength;


        System.out.println(Long.toBinaryString(remainingSequence) + " " +
Long.toBinaryString(shiftedGenerator));
```

```
        //In long bit polynomial we can divide 10 / 11 even if the left side is
smaller
        while (remainingSequenceLength >= generatorLength) {


            //XOR numerator and denominator
            remainingSequence ^= shiftedGenerator;


            // Update numerator length
            remainingSequenceLength = (int) (Math.log(remainingSequence) /
Math.log(2) + 1);


            // Shift denominator accordingly
            shiftedGenerator = generatorSequence << remainingSequenceLength -
generatorLength;


            System.out.println(Long.toBinaryString(remainingSequence) + " " +
Long.toBinaryString(shiftedGenerator));
        }


        return remainingSequence;
    }


    /**
     * Checks the correctness of the bit sequence.
     *
     * @param bitSequence       The CRC bit sequence excluding the CRC check value
     * @param inputLength       The length of the input bit sequence (including
possible leading zeros)
     * @param generatorSequence The generator bit sequence used
     * @param checkSequence     The CRC check value to check against
     * @return true if the sequence is valid, false otherwise
     */
    static boolean check(long bitSequence, int inputLength, long generatorSequence,
long checkSequence) {
```

```
        return checkSequence == calculate(bitSequence,inputLength,generatorSequence);
    }
}
```

Hamming Code

Java implementation (with arrays)

```
public class HammingCode {


    public static int redundantBits(int inputLength) {
        // To determine how many redundant bits wee need
        // we can make use of the fact that the sum of all parity bit positions has
to be bigger or equal to the
        // redundant bits + original bits length (but not bigger than necessary, 1
iteration extra is enough).
        // That is, we should be able to track all the bits for errors.
        int r = 0;
        int sum = 0;
        //System.out.println("sum: " + sum + ", r: " + r);
        while (sum < inputLength + r) {
            sum += Math.pow(2, r);
            r++;
            //System.out.println("sum: " + sum + ", r: " + r);
        }

        return r;
    }


    public static boolean evenParity(long[] bitSequence) {
        int count = 0;
        for (int i = 0; i < bitSequence.length; i++) {
            if (bitSequence[i] == 1) { //it also skips number 2 which is the dummy
parity bit value
                count++;
            }
```

```
        }
        return count % 2 == 0;
    }

    public static long[] pad(long bitSequence, int inputLength) {
        int totalBits = redundantBits(inputLength) + inputLength;
        long[] hammingSequence = new long[totalBits];
        for (int i = 0; i < totalBits; i++) {
            //if power of 2 skip
            if (i + 1 > 0 && ((i + 1 & i) == 0)) {
                hammingSequence[i] = 2; //arbitrary choice for the parity bit that
will be overwritten by 0 or 1
            } else {
                //populate: shifting 1 such that we copy the 1000 100 10 etc the
value and scale it to a unit
                hammingSequence[i] = (long) ((bitSequence & (0b1) << inputLength - 1)
/ Math.pow(2, inputLength - 1));
                inputLength--;
            }
            System.out.print(hammingSequence[i]);
        }
        System.out.println();
        return hammingSequence;
    }

    public static long[] fixedPad(long[] paddedSequence, boolean evenParity) {

        for (int i = 0; i < paddedSequence.length; i++) {
            //System.out.print(paddedSequence[i]);
            // for all parity bits:
            if (i + 1 > 0 && ((i + 1 & i) == 0)) {
```

```
                //System.out.print("(");
                //System.out.print(i + 1 + ":" + Long.toBinaryString(i + 1) + ":");
                // create a dedicated parity array
                long[] parityArray = new long[paddedSequence.length + 1]; //not
entirely sure about the length of the array



                // that starts at the parity bit (i)
                for (int j = i + 1; j < parityArray.length; j++) {


                    long maskShifts = (long) (Math.log(i + 1) / Math.log(2) + 1);
//based on the position of i
                    long mask = 0b1 << maskShifts - 1; //-1 because 0b1 is already
shifted once


                    //System.out.print("[" + Long.toBinaryString(j) + "/" +
Long.toBinaryString(mask) + "]");


                    if ((j & mask) != 0) {
                        //System.out.print("[" + j + "]");
                        //System.out.print(paddedSequence[i]);
                        parityArray[j - 1] = paddedSequence[j - 1];
                        //it doesnt really mater
                        // the order or the parity array as the number of 1s is the
same
                    }


                }


               // System.out.println(")");


                // check the parity of the array and adjust the parity bit (i)
accordingly
```

```
                paddedSequence[i] = evenParity(parityArray) ? (evenParity ? 0 : 1) :
(evenParity ? 1 : 0);
            }
        }
        //System.out.println("");
        for (int i = 0; i < paddedSequence.length; i++) {
            System.out.print(paddedSequence[i]);
        }
        System.out.println("");
        return paddedSequence;
    }
}
```

Assignment 2 – MAC Sublayer

The Medium Access Control layer is responsible for deciding who transmits data on the channel and when it is transmitted. In classic Ethernet, this is a large responsibility, because machines share a single channel. If multiple machines send at the same time on the same channel, a collision occurs and the transmitted data is distorted.

With modern Ethernet, this task is often no longer required, because machines use point-to-point links (In wireless channels, however, this sublayer is still very important). Every channel only has two devices connected to it. These devices can be regular computers, but they can also be routers or switches. Switches are what make the point-to-point link architecture possible. Without it, a machine would need to have a separate cable for each other machine it is connected to. Instead, switches are used. These switches can be compared to a telephone switch board; it receives frames on one link, and then transmits it on another link, depending on where the frame needs to go. Because these links are not directly connected to each other, the throughput of the network improves significantly. Every machine can send frames to the switch at the same time, without the possibility of collisions.

Backwards learning

In the data-link layer, hubs, bridges, and switches are the main type of devices you are going to encounter. The function of these devices is to eliminate the need for shared access mediums on wired network connections. Before the introduction of these devices, clients had to wait their turn before they could send data on the shared channel. With the introduction of hubs, it was possible for each client to have a dedicated channel to the hub that didn't go down when other computers were offline. Although the hub was a step forward, it still had the disadvantage of blocking the network when a packet was sent by broadcasting it to every client connected to the hub.

Bridges and switches solved this problem by keeping track of where clients are located on the switch, in a so called 'forwarding table'. To learn the location of the clients, bridges and switches use an algorithm

called backward learning. This algorithm automatically starts learning about all the clients when they become active on the network. When a machine sends a message, we can store the port it is coming from in our table, along with the corresponding MAC address. The next time we want the device with this MAC address to receive a message, we know on which link we should forward it. When the location of a machine is unknown, the message is flooded.

Switch implementation (backwards learning) on java

```
import java.util.*;

import java.util.stream.Collectors;


/**

 * Switch that receives and forwards messages following the backward-learning
algorithm.

 * <p>

 * The switch should store the information on which machines are connected to which
ports.

 */

class Switch {


    Hashtable<Integer, Integer> router;

    ArrayList<Integer> ports = new ArrayList<>();


    /**

     * Switch constructor

     *

     * @param numberOfPorts The number of ports of the switch.

     */

    public Switch(int numberOfPorts) {

        router = new Hashtable<>(numberOfPorts);


        //populate all ports

        for (int i = 0; i < numberOfPorts; i++) {

            ports.add(i);

        }
```

```
    }

    /**
     * Gets called for every incoming message to the switch.
     * <p>
     * Should determine on which links the message should be forwarded
     * using the backward-learning algorithm.
     * <p>
     * The frame received is in the format described in the
     * assignment description.
     *
     * @param incomingPort The port where the message came in
     * @param frame        The message that came in
     * @return The ports where the message should be send out from
     */
    public List<Integer> receive(int incomingPort, String frame) {
        int source = Integer.parseInt(frame.substring(0, 4), 16);
        router.put(source, incomingPort);

        List<Integer> destinationPorts = new ArrayList<>();

        int destination = Integer.parseInt(frame.substring(4, 8), 16);
        if (router.get(destination) == null) {

            //send to all ports
            destinationPorts = ports.stream().collect(Collectors.toList());

            //besides this one, as it is now assigned to the previously unknown
destination
            destinationPorts.remove(incomingPort);
        } else {
            //assuming that 1 known address has 1 destination
```

```
            if (router.get(destination) != incomingPort) {
                destinationPorts.add(router.get(destination));
            }
        }
        return destinationPorts;
    }
}
```

Spanning tree protocol (STP)

Devices in the data-link layer are normally connected redundantly to make sure that physical failures will not cause a network outage. A good example of this is a switch. When there are two cables connecting two switches, if one fails the other one can still carry the traffic.

However, when there are multiple routes to a destination (also called loops), a broadcast storm can occur, which is characterized by a high number of packets being sent to the entire network in a short period of time. This happens because, when we have a loop, there is always a path for the traffic to go (that isn't the incoming link).

To solve this problem, higher-end switches make use of a protocol called Spanning Tree Protocol (STP). This protocol constructs a view of the network and creates a tree out of the network graph, without any loops. The extra redundant links that form a loop are temporarily disabled to accomplish this. Then, all the data packets are routed through the links that are enabled, but not through the links that were disabled. When one of the links in the network has failures, the STP protocol can automatically reconfigure the network to use the redundant links by enabling them again. Note that broadcast packets used in constructing the view of the network can still be switched through these disabled links, as these links are only disabled for data packets.

The regular STP has two main steps:

1. Determine the root switch of the network.

2. Calculate the best (least costly) path based on e.g. hops, latency, speed, or bandwidth from each switch to the root switch. This is accomplished by broadcasting packets from the root to all other switches so that all switches determine their best path to the root. The switches also determine which ports to block, and which port to use to send packets to the root.

3. After performing the steps above, sending data packets is carried out using the enabled links only.

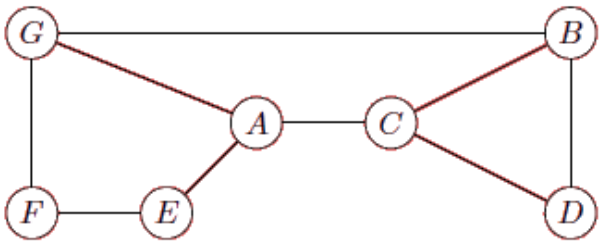
Tutorial 4 – The network Layer

Distance Vector Routing

- Routers send the [delay distance (distance vector) to other routers] to their direct neighbors (hence not flooding the network).

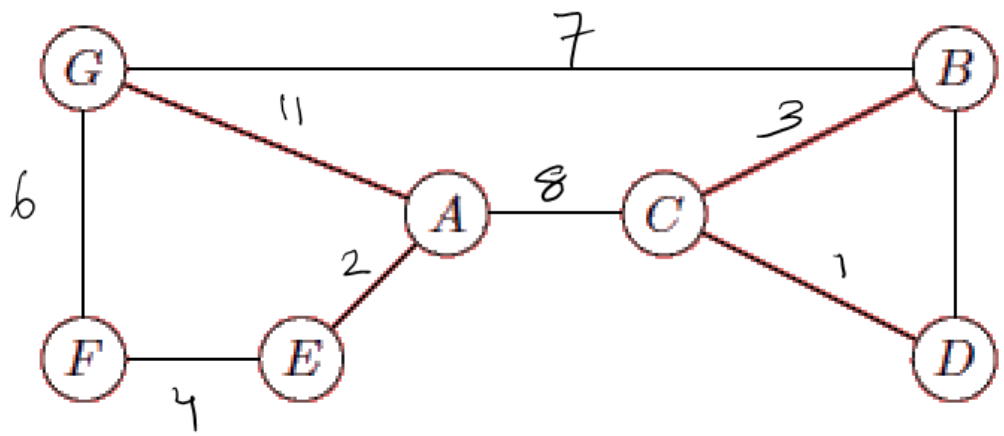
- With that information they update their own routing tables. Which contain the distance to a specific router, and the first router they hop onto to get there. This is how they get "the big picture" of the network.
- Failing routers might not get updated in "the big picture" on time which might lead to the count to infinity problem.
- The routing table always contains the least-cost path, even if the path is longer, including than the direct neighbor.

To	From		
	C	E	G
A	8	2	11
B	3	13	7
C	0	10	10
D	1	11	11
E	10	0	10
F	14	4	6
G	10	10	0
Delay	8	2	11



Assume A knows its distances to its neighbors. Then A collects distance vectors from C, E and G, and constructs the table on the left. For example, the number 13 on the table means that B's distance to E is 13 units.

What does the routing table of A look like?
(Give both the outgoing line and the cost.)



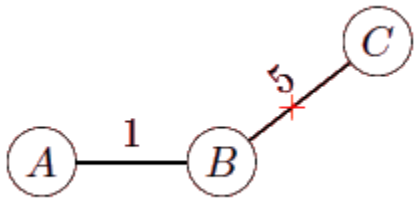
Routing table A for A

To	Distance	Line
A	0	-
B	11	C

C	8	C
D	9	C
E	2	E
F	6	E
G	11	G

How would the routing table change if the delay to G is 12 rather than 11?

- All entries except the entry for G would be same as before. For entry G:
- The value for the line could be E or G depending on the order of messages and the local policy (e.g., it could be the least recently received information or the most recently received
- information).

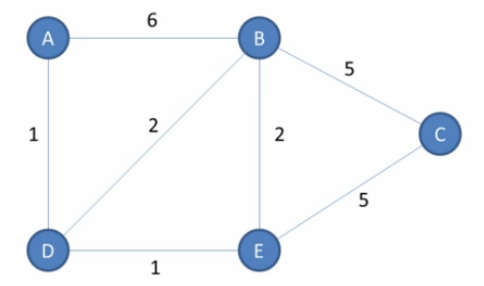


Consider the network on the left. What happens if C becomes unresponsive after nodes have constructed their routing tables? Can you think of changes to the algorithm that will prevent the issue?

- Before C becomes unresponsive, A will calculate its delay to C as 6.
- After C becomes unresponsive, B will be the first to detect this, and it will search for others ways of reaching C.
- B will learn from A that A can reach C with a delay of 6. Thus, B will update its routing table entry for C as link=A, delay=7. (It will also change its delay to A)
- B will learn from A that A can reach C with a delay of 6. Thus, B will update its routing table entry for C as link=A, delay=7. (It will also change its delay to A)
- A will learn from B that B's delay to C has increased to 7; so A will update its routing table entry for C as link=B, delay=8. (It will also change its delay to B)
- B will learn from A that … (Count to infinity problem)

Link State Routing

- No count to infinity problem
- Shortest path algorithm (Dijkstra)
- Only direct neighbor information (reliable) is sent over the entire network (flooding)
- From the direct neighbor information we can build a graph and apply Dijkstra



- Dijkstra calculates the shortest path from a specific starting node to all the other nodes in the graph
- For node A as starting node set up the table with:
  - All infinity except A itself with 0 shortest distance
- And keep track of two lists
  - visited: {}
  - univisited: {A,B,C,D,E}

- Start by visiting the unvisited nodes from the smallest known distance from the start vertex
    - It will always be the start vertex itself
    - Then examine the unvisited neighbors of the selected node
    - The distance of each neighbor from the start vertex will be the sum of the previous distances to reach the selected node + the distance shown by the edge number
- If the calculated distance of a node (from the start point) is less than the known distance (that appears in the table), then the table is updated with the shortest distance. The previous vertex value is updated accordingly

Node	Shortest distance from A	Previous vertex
A	0	-
B	0 + 6 = 6	A
C	∞	?
D	0 + 1 = 1	A
E	∞	?

- Visited: {A}
- Unvisited: {B, C, D, E}

Node	Shortest distance from A	Previous vertex
A	0	-
B	1 + 2 = 3 < 6, so 3	D
C	∞	?
D	1	A
E	1 + 1	D

- Visited: {A, D}
- Unvisited: {B, C, D}

Node	Shortest distance from A	Previous vertex
A	0	-
B	3 < 2 + 2, so 3 stays	D
C	2+5=7	E
D	1	A
E	2	D

- Visited: {A, D, E}
- Unvisited: {B, C}

Node	Shortest distance from A	Previous vertex
A	0	-
B	3	D
C	7 < 3 + 5, so 7 stays	E
D	1	A
E	2	D

- Visited: {A, D, E, B}
- Unvisited: {C}

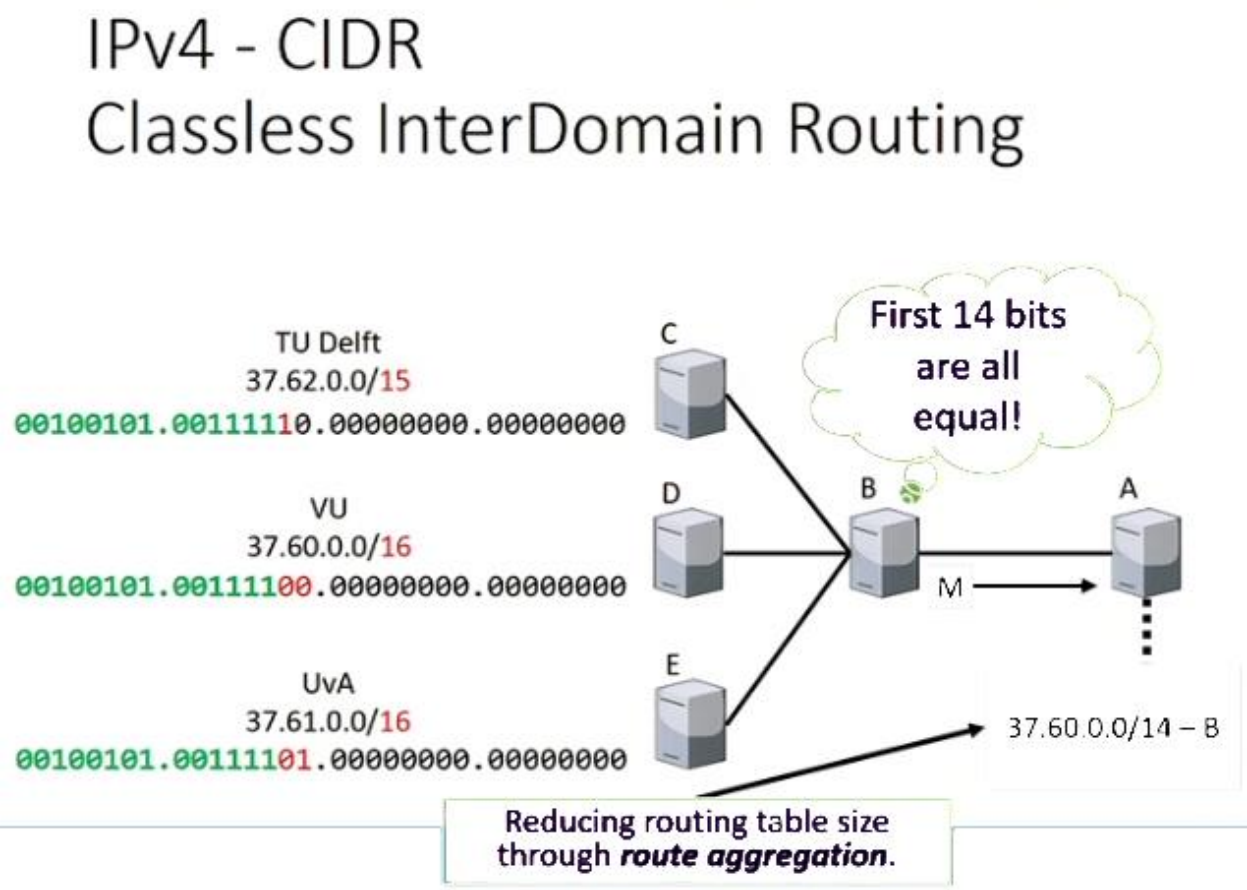
Node	Shortest distance from A	Previous vertex
A	0	-
B	3	D
C	7 (there's nothing left to do)	E
D	1	A
E	2	D

- Visited: {A, D, E, B, C}
- Unvisited: {}

Assume a node failed and became unresponsive (and neighbors have realized that is unresponsive). What messages are sent by the nodes now and what is the new routing table?

- If the failing node was on the shortest path of another node, then the routing table will change for that node as the failing node is removed completely from the table

Classless Inter-Domain Routing (CIDR)



Question:

Network Prefix/Subnet Mask	Gateway
0.0.0.0/0	Splinter
101.45.6.0/24	Leonardo
101.6.0.3/32	Michelangelo
166.1.0.0/16	Donatello
166.1.0.0/20	Raphael

Using the routing table above and assuming that CIDR is used; Find the addresses covered by each router, and decide which gateways the following IPs be routed to?

- the IP address is followed by the number of bits in the mask
- The bigger the subnet mask, the smaller the network it covers. Remember that you have to AND the IP and the subnet mask (which is based on having k bits).
- Gateway is a synonym for router in this context. The routers are responsible for the following:
    - Splinter: 0.0.0.0/0 is the default router for all non-local IP addresses (not covered by other routers)
    - Leonardo: 101.45.6.0/24 (24-bit prefix):
        - Byte.byte.byte AND 101.45.6.0 = 101.45.6.0 (to 101.45.6.255)
    - Michelangelo: is 101.6.03 (32-bit prefix):
        - A 32 bit prefix covers all the IP bytes. So Michelangelo only covers itself
    - Donatello: 166.1.0.0 (16-bit prefix):
        - Byte.byte. AND 166.1.0.0 = 166.1.0.0 (to 166.1.255.255)
    - Raphael: 166.1.0.0 (20-bit prefix):
        - Byte.Byte.11110000.0 AND 166.1.0.0:
        - 11111111.11111111.11110000.00000000
        - 10100110.00000001.00000000.00000000 (from 166.1.0.0)
        - 10100110.00000001.00001111.11111111 (to 166.1.15.255)
    - Notice that the responsibilities of Donatello and Raphael intersect
        - According to the longest matching prefix rule (aka more subnet mask bits aka smallest network), Raphael will be responsible for the overlapping interval
- 101.6.0.3 -> Michelangelo
- 101.6.0.4 -> Splinter
- 101.45.6.254 -> Leonardo
- 166.1.33.5 -> Donatello
- 166.1.0.7 -> Raphael

Assignment 3 – The network layer I

In computer networking, routers are responsible for forwarding data packets between computer networks. A router is connected, usually via cables, to at least two different networks. When it receives a new packet from one of the networks, it reads the packet's destination address (usually an IP address) and attempts to forward it to the appropriate network, so that it will (eventually, hopefully) reach its destination.

To successfully forward a data packet towards its destination, the router needs to know which destinations can be reached through the different networks to which it is connected. To determine which destinations can be reached a router can make use of three different kinds of algorithms:

- Static routing
- Dynamic routing - distance vector
- Dynamic routing - link state

Static Routing

Static routing is a type of routing algorithm that makes use of manually-configured routing table entries, in contrast to dynamic routing algorithms which use network messages to determine the best routes. When failures happen in the network, static routes will not get updated. They have to be manually updated to use a different route by the network administrator. This already highlights one of the drawbacks of static routing schemes. With that said, they also have their advantages and in reality, in order to maximize performance both static and dynamic routing can be used.

Advantages:

- Little resource usage
- Secure (Administrators have a lot of control over the routing)
- Great for small networks (less complexity)

Disadvantages:

- Human errors
- Fault tolerance

Routing tables

A router uses its routing table to decide where to forward incoming packets. A typical routing table may contain the following columns:

- Network destination: The range of destination IPs that can be reached on this network.
- Subnet mask: This separates the destination IP into its network address and host address components.
- Gateway: The IP address of the next router to which to send the packet.

When the router receives a packet, it checks which network destination(s) correspond(s) to the packet's destination address. If there are multiple network destinations possible, the one with the longest prefix match is preferred. The router then forwards the packet to the appropriate gateway.

Routing table implementation

```
import java.util.*;

import java.util.stream.Collectors;


/**

 * Routing table for static routing.

 * It holds a list of static routes that can be queried to find the gateway which can reach the destination.

 */
```

```
class RoutingTable {


    public static int ERROR_NO_ROUTE = -1;


    public static ArrayList<Network> routes = new ArrayList<>();


    // For testing
    public static int IPv4toInt(String ipString) {
        String[] ipParts = ipString.split("\\.");
        int ipInt = 0b0;
        for (int i = 0; i < ipParts.length; i++) {
            ipInt += (Integer.parseInt(ipParts[i]) << (8 * (3 - i)));
        }
        return ipInt;
    }


    /**
     * Adds a route to the routing table.
     *
     * @param networkPrefix Network prefix
     * @param subnetMask    Subnet mask
     * @param gateway       Gateway
     */
    public void addRoute(int networkPrefix, int subnetMask, int gateway) {
        Network route = new Network(networkPrefix,subnetMask, gateway);
        routes.add(route);
    }


    /**
     * Queries the routing table to determine which gateway can reach the desired
network the IP address belongs to.
     *
```

```
     * @param address The IP address we want to route to.
     * @return The gateway that can reach the desired network, or ERROR_NO_ROUTE when no route to that network can be found.
     */
    public int lookupRoute(int address) {
        List<Network> matches = new ArrayList<Network>();
        for (Network route : routes){
            if (route.contains(address)){
                matches.add(route);
            }
        }

        matches = matches.stream()
                .sorted(Comparator.comparingInt(Network::size).reversed())
                .collect(Collectors.toList());

        return matches.size() == 0 ? ERROR_NO_ROUTE : matches.get(0).getGateway();
    }
}

class Network {
    int gateway;
    int subnetMask;
    int smallestAddress;
    int largestAddress;

    public Network(int prefix, int subnetMask, int gateway) {
        this.gateway = gateway;
        this.subnetMask = subnetMask;
        this.smallestAddress = prefix & subnetMask;
        largestAddress = smallestAddress | ~subnetMask;
    }
```

```
    public boolean contains(int address) {
        return address >= smallestAddress && address <= largestAddress;
    }

    public int size() {
        return Long.bitCount(subnetMask);
    }

    public int getGateway() {
        return gateway;
    }
}
```

Dynamic routing

makes use of messages to learn where to route packets to get them to their destination. There are two different types of dynamic routing algorithms: *Distance Vector* and *Link State.* Distance vector routing algorithms only gather data from their neighbor (directly connected) routers, while link state gathers data from all routers in the network. Because of this difference both types of algorithms have different advantages and disadvantages.

Distance vector

Distance vector algorithms only know the state of their neighbors. They use propagation to spread knowledge about distant routers. Each router slowly learns more about the network by gossiping newly discovered routers.

Advantages compared to link state algorithms:

- Low bandwidth requirements due to local sharing
- Small packets
- No flooding

Disadvantages compared to link state algorithms:

- Based on local knowledge instead of global
- Converges slowly e.g. takes longer to propagate broken links

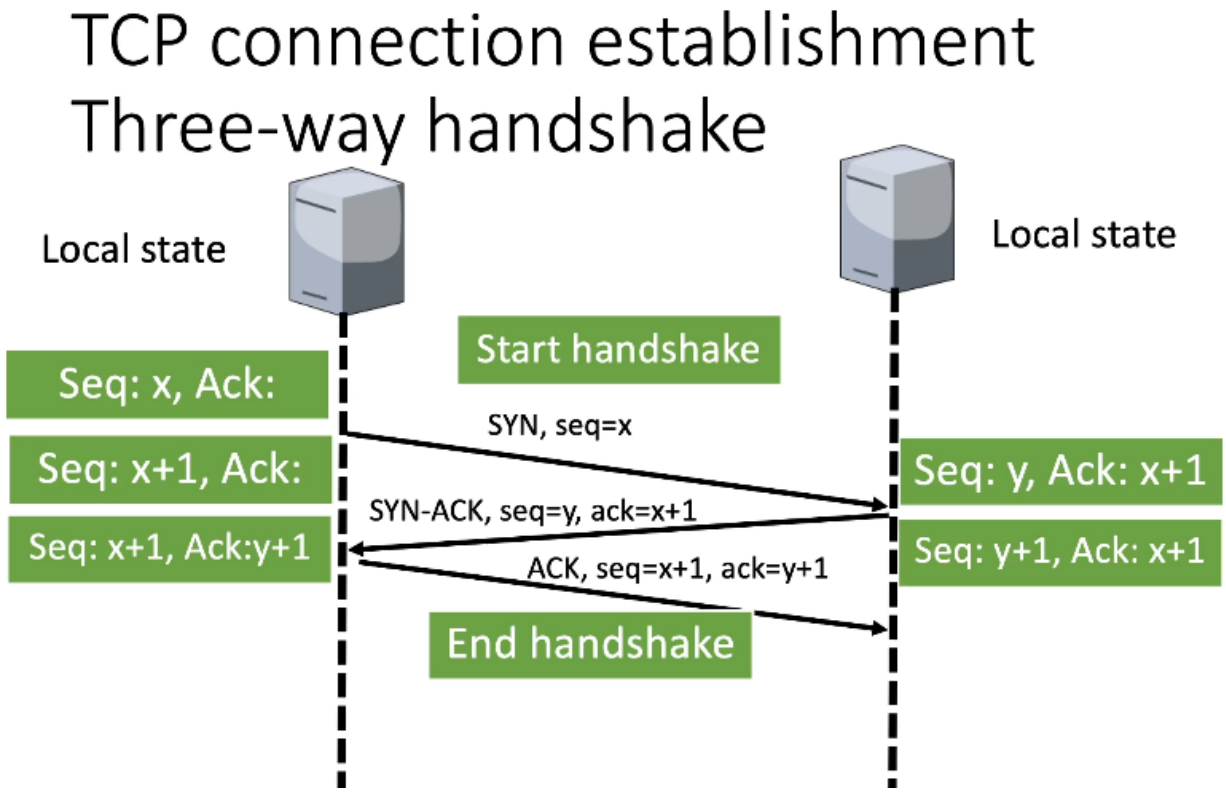
An example of a distance vector algorithm is RIP.

Link state

Link state algorithms work on global state. Every router gathers information about its neighbours and floods this through the network. This approach leads to a very fast converging time, but also creates network overhead.

An example of a distance vector algorithm is OSPF.

Tutorial 5 – The transport layer



TCP Sequence numbers

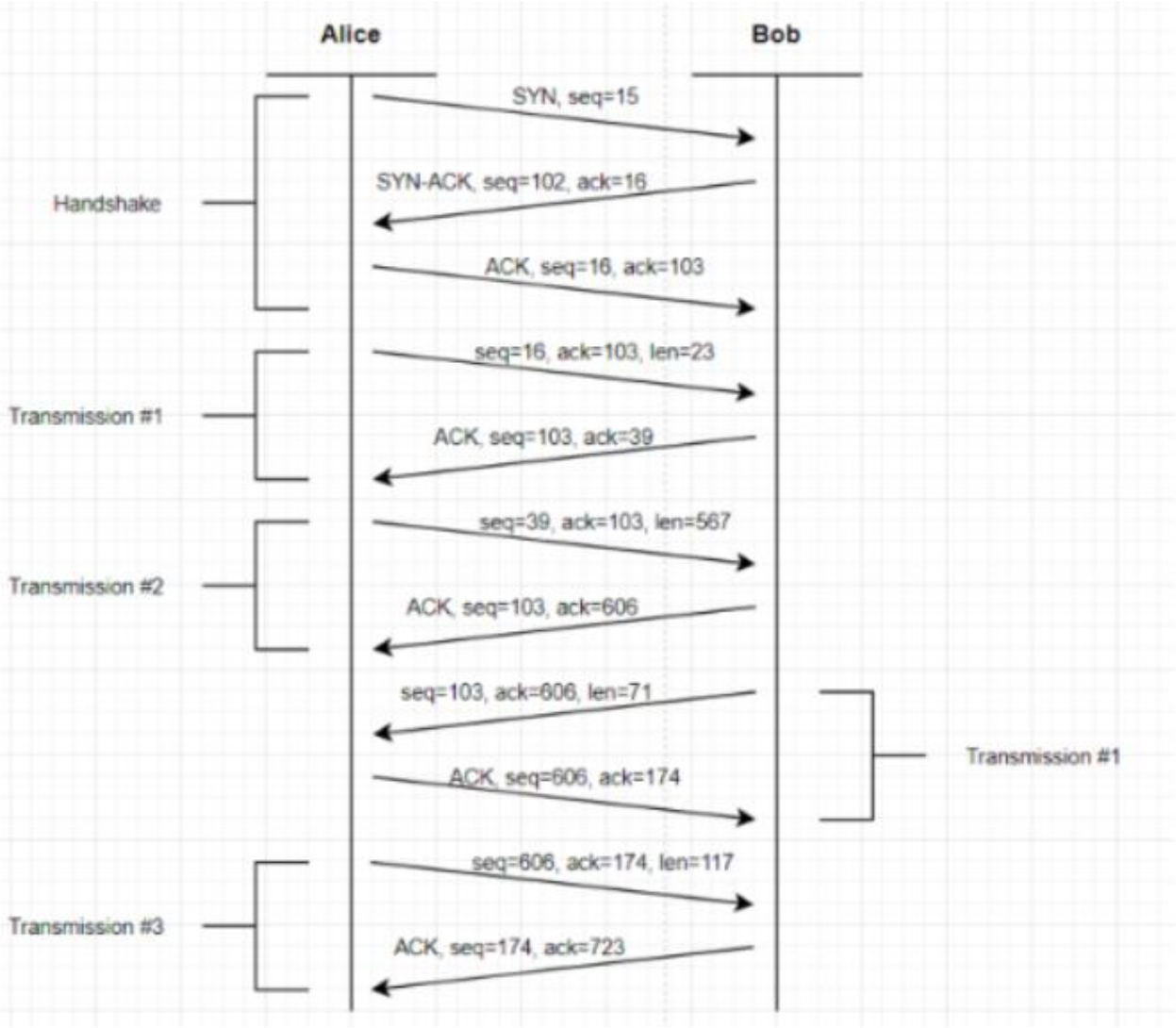
- Alice starts a TCP handshake with Bob using the initial sequence number 15.
- After receiving Bob's SYN-ACK and his choice of initial sequence number 102
- Alice sends three segments of size
    - 23
    - 567
    - 117 bytes.

- Remember that he acknowledgment number that Alice puts in its segment is the number of bytes Alice has received
    - This number must match Bob’s next segment sequence number

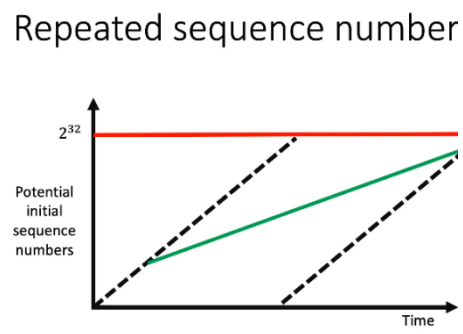
Which sequence number does the third segment have?

In the scenario from the previous question, Bob sends one segment of 71 bytes that arrives before Alice sends her third segment.

What is the acknowledgement number of Alice's third segment?



Forbidden region



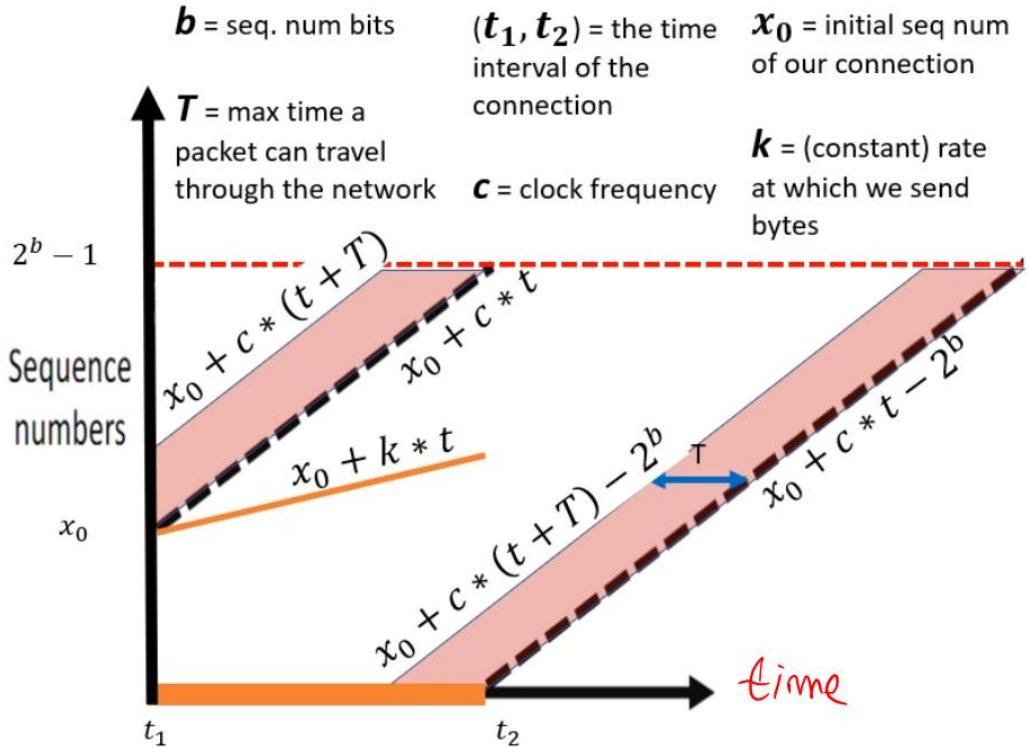
A non-standard transport layer protocol that uses 20-bit sequence numbers. They further use clocks that advance by 2^{10} ticks per second.

How many seconds does it take until sequence numbers are potentially re-used?

Assuming that the sequence numbers increase at the same rate of the clock, then it takes $2^{20}/2^{10} = 2^{10} = 1024$ seconds

- Now, assume that packets can remain in the network for up to 10 seconds.
- Tosh (continuously) sends data at 5 B/s.

How long does it take for her sequence numbers to enter the forbidden region?



Anything that happens before the connection we ignore it, therefore t1 = 0

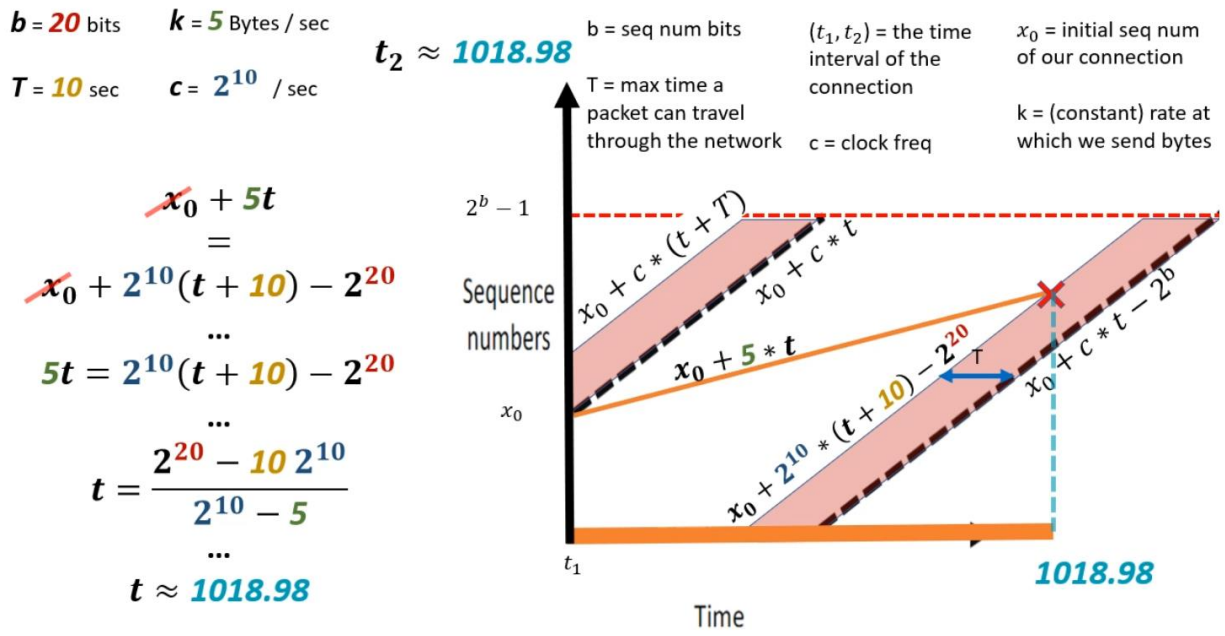
Question

The alien-hunting members of Torchwood rely on non-standard transport layer protocol that uses 20-bit sequence numbers. They further use clocks that advance by 2^{10} ticks per second. Assume that packets can remain in the network for up to 10 seconds. Tosh sends data at 5 B/s.

How long does it take for her sequence numbers to enter the forbidden region?

Variable	*Explanation*	*Value*
b	Sequence number bits, range 0 to $2^{20} - 1$	***20*** bits
c	Clock frequency	2^{10} / sec
T	Max delay	***10*** sec
k	Rate of sending bytes	***5*** B / sec
(t_1, t_2)	Time interval of the connection	(0, ***?***)

We just need to calculate the intersection of the sequence number increase rate line and the forbidden region line.



- This computation of when the sequence numbers enter the forbidden region is used by the TCP so that it can change sequence numbers to avoid entering the forbidden region.

Tutorial 5 MCQ

- UDP does not re-order segments
- Flow control handles destination bottlenecks (receiver)
- Congestion control handles network bottlenecks
- TCP Tahoe resets congestion window to 1

TCP retransmission triggered by three duplicate ACKs is considered less problematic than one triggered by a timeout. Why?

- The network is still (partially) available for duplicate ACKs

Process servers are used for applications that are used infrequently

Assignment 4 – Network Analysis Basics (Protocols)

In this assignment, you will inspect traces of network traffic.
These traces are stored in PCAP files, which you can open and analyze using the program *Wireshark*.

Useful filters:

- ip.addr == 172.20.8.33
  - eth.src == 14:f6:d8:d5:3d:24 (this one never changes and it's always this PC)
- tcp or dns
- tcp.port == 443

- tcp.analysis.flags, for tcp errors
- !(arp or icmp or dns), removes from tracefile (“pruning” the tracefile
- follow tcp stream (right click menu), focus on 1 single tcp connection
- tcp contains facebook, (good to look up for data in the stream)
- http.request (for get, put, etc)
- http.response.code == 200
- tcp.flags.syn == 1 (useful to check if a single server is being bombarded with new requests next to each other)

Data Link Layer

- The data link layer is responsible for achieving reliable and efficient communication of so called ‘frames’ between two machines.
- The MAC sublayer adds responsibility for shared networks. If multiple devices share a cable, they can’t all talk at the same time.

ARP (Address Resolution Protocol)

- ARP broadcasts a request to look for the owner of a specific local IP address
    - Allegedly only the owner, who matches the local IP address, replies by stating his MAC address such as the requester can update his ARP table with (IPs,MAC) tuples.
- A gratuitius ARP announcement is “gratuitus” as it does not reply to a specific request and it broadcast its to all devices.
  Sender MAC address is his actual MAC address
  It also sends his IP address (security threat, as it could lead to cache poisoning of the other devices’ ARP tables).
  The target IP address is his own IP address.
  Potential uses:
    - Help detect IP conflicts
    - They inform switches of the MAC address of the machine on a given switch port
    - They assist in the updating of other machines' ARP tables (security threat too)
    - The host just came online and wants to let everyone know where they can find this IP.

Network Layer

The **tracert** command is a Command Prompt command that's **used** to show several details about the path that a packet takes from the computer or device you're on to whatever destination you specify.

- Mostly the hops from a router to a destination that the packet is travelling, and network stats
- Request timed out for the in between steps are either due to a firewall configuration or ISP dropping ICMP traffic due to low priority and/or DOS security threats.

```
C:\Users\sergio>tracert -4 -w 1000 www.heyo.com.


Tracing route to www.heyo.com [104.130.102.169]
over a maximum of 30 hops:
```

```
 1    16 ms    17 ms    19 ms  172.20.8.201
 2     *        *        *     Request timed out.
 3     *        *        *     Request timed out.
 4     *        *        *     Request timed out.
 5     *        *        *     Request timed out.
 6     *        *        *     Request timed out.
 7     *        *        *     Request timed out.
 8     *        *        *     Request timed out.
 9     *        *        *     Request timed out.
10     *        *        *     Request timed out.
11     *        *        *     Request timed out.
12     *        *        *     Request timed out.
13     *        *        *     Request timed out.
14     *        *        *     Request timed out.
15   225 ms   201 ms   201 ms  104.130.102.169
```

- IPv4 vs IPv6 MCQ:
  - IPv4: 4294967296 addresses (2^32)
  - IPv6: 3.40282367e38 addresses (2^128)
  - The IPv6 header has less fields than the IPv4 header.

Transport layer

- Order of packets sent in the three way handshake of TCP:
  - SYN -> SYN/ACK -> ACK
- TCP is connection based, whereas UDP is connectionless.
- The UDP header does not contain a sequence number.
- UDP does not require a handshake to be done between hosts, whereas TCP does.

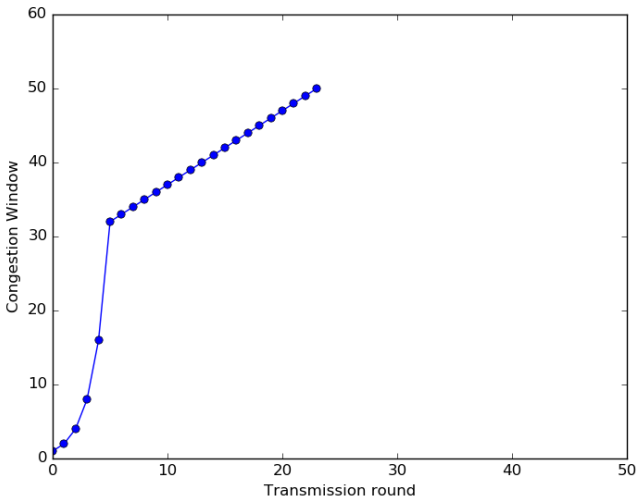
Application Layer

- DNS use transaction IDs to match the response to the query

Tutorial 6 – Transport & Application Layer

Transport layer – Congestion Control

- TCP congestion window is what is being sent
- TCP threshold is the border after which the increase goes additive
- While under the threshold the increase is times 2 times 2 times 2…



Assume the use of TCP Tahoe with a multiplicative factor of 2. In the figure, you see the size of congestion window at each transmission round.

What is the threshold at the start of this trace?

32, as we can see that after that it only increases linearly

Assume that the last transmission depicted here fails when the congestion window is 50. What is the new threshold?

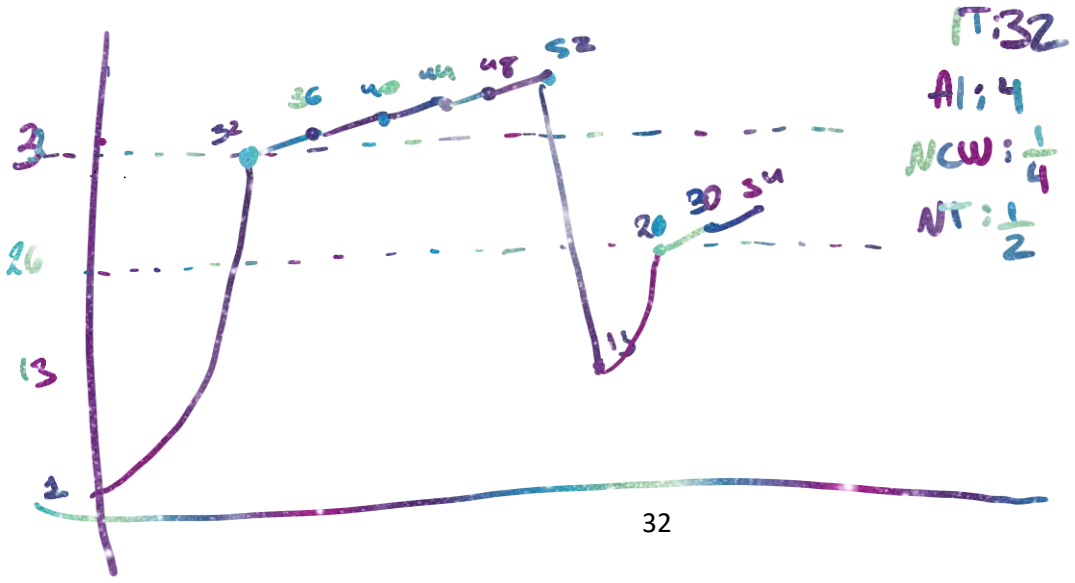
50/factor = 50/2 = 25.

Think of a modified version of TCP congestion control protocol. Assume it has following properties:

- Slow start with an initial threshold of 23 KB
- Additive increase of 4 KB
- If a lost packet is detected:
  - New congestion window is set to ¼ of the old congestion
  - New threshold is ½ of the old congestion window
  - Generally speaking, the new threshold is set in relation to the last window, and in Reno it would then jump to that threshold directly (fast recovery)
- Assume this happens:
  - Reach initial threshold of 32 KB
  - Then 5 ACKs without failure
  - Then 1 packet lost
  - Adjust window threshold and new congestion
  - 3 more ACKs
  - Then 1 packet lost

What is the value of the congestion window afterwards?

34



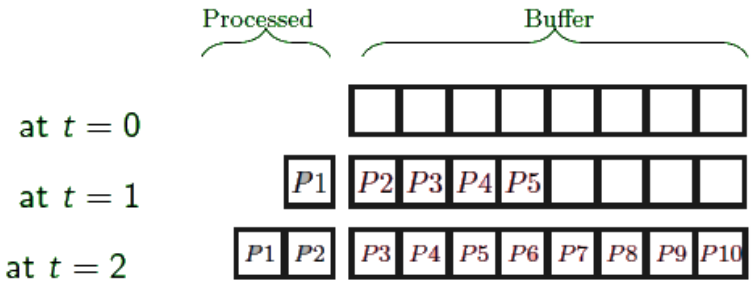
Transport layer – Flow Control

Assume there is an error-free connection between Batman and Superman.

- Batman wants to send data to Superman at 5 mbps.
- Superman's machine can only process the data at 1 mbps
- Superman's machine has a buffer of 8mb.
- After the buffer is full, the machine starts discarding data.

How much data can Batman maximally send without any of it being discarded?

- Data is written to the buffer at 5mbps and is read at 1mbps
- The available buffer at time t is:
  - Buffer – batmanspeed * t + superman speed * t
  - 8 – 5t + 1t = 8 – 4t
- Setting 8 – 4t = 0 (zero loss) gives t = 2. In 2s batman sends 10mb as given by:



Now, after the buffer is full, Batman adapts his sending rate to the maximal rate Superman can handle, and continues sending. Ignore processing times at Superman's side.

How long does it take for Batman to send 15 mb?

- As by the previous exercise, Batman sends 10 mb in 2 seconds.
- Afterwards, he adjusts his sending rate to 1 mbps.
- Hence, sending the remaining 5 mb takes 5 seconds.
  - It takes him a total of 7 s to send 15 mb.

Application layer – Base64 to ASCII

- Characters given by a table
  - From right to left or left to right given by exercise, for last block the padding is done on the right when reading from the left.
- Convention is that it has to be a multiple of 4
  - Pad with = or == if necessary

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	/

Decode 11101101 10010001 00111110 10110101 01100010 as ASCII characters using base64

- 111011: 7
- 011001: Z
- 000100: E
- 111110: +
- 101101: t
- 010110: W
- 0010(00): I

But we have 7 symbols, need multiple of 4. So we append =. Resulting in 7ZE+tWI=

Practice Exam: How many characters does the Base64 encoding of 3,688 bytes have?

- 3,688/3=1229.33, so there are 1229 groups of 3 bytes and 1 byte left over.
- Each of 3 bytes is encoded as 4 characters, so 4·1229=4916 characters.
- The last byte is encoded as 2 Base64 characters, and two = signs are appended for padding.
- Hence, the total number of characters is 4916+2+2=4920.

Or (maybe?)

3688 bytes = 3688 * 8 = 29504 bits = 29504/6 base64 chars = ceil(4917.33333333) = 4918 chars
However 4918 mod 4 = 2, so we add two characters, namely ==
So the total number of characters is 4920

Application layer – DHT Routing

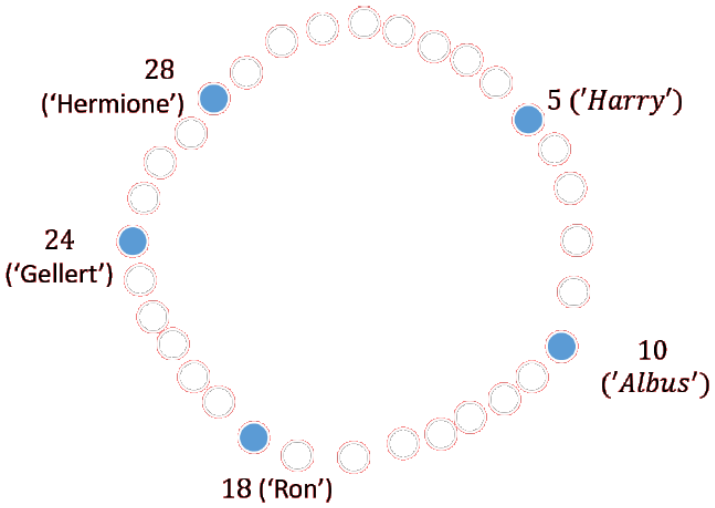
In order to face a massive increase of data P2P systems scalability is great for file sharing applications, which rely on DHT (Distributed Hash Table), which distributes the responsibility of storing data on a large set of nodes and access them efficiently (where machines act as both, clients and servers).

The goal is to route the message of any source to any destination.

With finger tables we only have the nodes to keep a subset of all the hops routes so that they don't need to keep a copy of the entire route. It takes a logarithmic complexity to reach the required hop and it takes logarithmic size to make the routing table for each node.

Question:

Consider a Chord DHT with m = 5. Possible node identifiers are illustrated above in a circle. Blue circles correspond to actual machines.



Compute the finger table of Hermione (machine 28).

- The finger table will have three columns: the entry number i, **start** and **address successor**, as below

i	Start	Address Successor

- Let's firstly find the **start** values in the table. For each entry i we will have

$$start_i = 28 + 2^i(\mod 32) \qquad (1)$$

- So, we fill the start values:

i	Start	Address Successor
0	29	
1	30	
2	0	
3	4	
4	12	

Successor is the is the closest "colored" node going only clockwise.

Tutorial 6 MCQ

Blockchain consensus algorithms:

- Proof of elapsed time
- Proof of work
- Proof of Stake

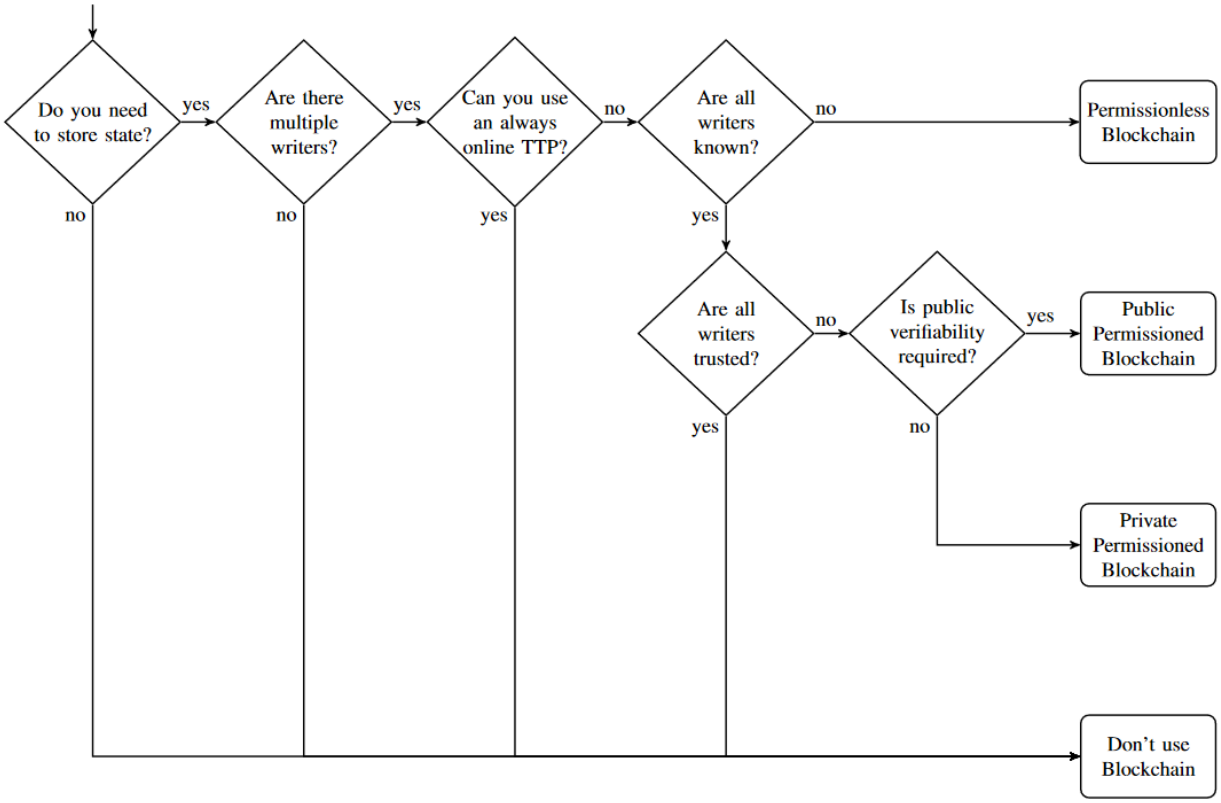
(Kahoot completion is NOT)

- Recursive DNS lookups are faster than iterative
- A local name server allowing recursive lookups is more vulnerable to an attack that aims to overload the server as the client can cause more load with a recursive lookup request than with an iterative one.

Let there be n nodes in a b-bit Chord Distributed Hash Table. Assume that 2^b is considerable larger than n. How many steps does it take to find the node that is responsible for a location in a b-?

- O(log n)

Players of **registered** online chess games both sign a document certifying the winner of the game. However, they also need to **store** the records of the games to derive the rank of players, which should be **available for fans and tournament organizers**. Which type of blockchain should they use?



- Public permissioned blockchain

Assignment 5 – Security

Man-in-the-Middle ARP spoofing

Networks operate on multiple different layers. The layers responsible for sending messages between different systems are the Data link layer (layer 2) and the Network layer (layer 3). Layer 2 is responsible for sending messages to systems on the same broadcast domain (network), while Layer 3 handles routing messages between different networks. Since a host does not know where a computer is, the layer 3 address (IP address) is used to address the target system. When a host determines, based on the IP address, that the target host is on the same network, it will use a protocol called Address Resolution Protocol (ARP) to find the layer 2 address (MAC address) to be able to send the data frame over the broadcast medium.

ARP works by maintaining a table (ARP table) that holds mappings of IP addresses to MAC addresses. This table is populated by broadcasting requests (ARP request) to the network looking for the MAC address of a specific target IP address when that host needs to be reached. The target host will upon receiving this request, send a response (ARP reply) to the requesting node with its MAC address. In bigger networks, a change of IP address could result in many requests. To optimize this process, a mechanism for unsolicited ARP replies, called Gratuitous ARP, was added. This makes it possible for a target node to broadcast its changed IP address to all hosts in the network who can then update their ARP tables.

The ARP protocol does not have any authentication or authenticity checks built-in. The protocol relies on everyone in the network to be truthful. As you might have seen in Assignment 4, ARP spoofing can be used relatively easily to attack the confidentiality of a network. In this assignment, we will use a simplified version of an ARP packet to illustrate a man-in-the-middle (MITM) attack. This involves secretly relaying and possibly altering the communication between two parties who think they are directly communicating.

This involves using the ARP protocol to convince the network (or a single node) to map a certain IP address to your computer. This causes all data meant for that IP address to be routed to your node. You can then view or alter this data and forward it to the intended recipient making it seem like it came directly from the source.

- You must intercept all traffic to the target IP.
- You must forward the frames to the target, making it seem like they were sent by the original source.
- Implementations of ARP have a timeout that decides how long the entries are kept in the ARP table, hence the spoof ARP must be sent at regular intervals to keep the traffic redirection in place. Since timed/scheduled functions would add unnecessary complexity, you do not need to implement this functionality in your ARP table. As for the MITM attack, you should resend the spoof ARP message every time you receive a data frame.

Message Modification

You conduct a targeted attack on one of the employees of the bank, right at the moment when the money is transferred. During this attack, you have peaked at their screen and you have seen to which account the money is being transferred.

Additionally, a secret source has provided you with the format of the packet, and the method of encryption. From this you have learned that the bank uses a stream cipher based on XOR.

The packet format is as follows (all fields are encoded in hexadecimal):

Opcode: 1 byte	Sender account number: 8 bytes	Target account number: 8 bytes	Amount: 4 bytes

- XOR stream ciphers are vulnerable to known-plaintext attacks (which we will implement in this exercise).
- XOR ciphers that use a constant repeating key can also be broken using frequency analysis, without using a plaintext attack. Hence the original message can be manipulated and encrypted back in the same fashion without knowing the secret key.

Message Authentication Code

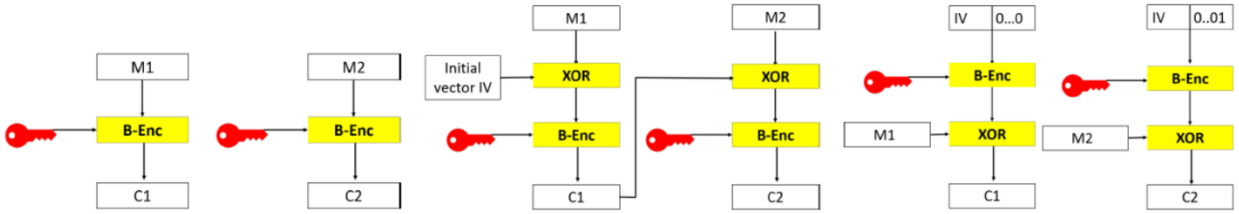
A MAC, sometimes known as a *tag*, is a short piece of information used to authenticate a message. More specifically, the receiver can confirm that the message came from the stated sender (its authenticity) and that the message data has not been changed (its integrity). A MAC system contains two important algorithms:

- A MAC generation algorithm, which returns the MAC given the secret key and message.
- A MAC verifying algorithm, which verifies a message given the secret key and the MAC.

The program needs to meet the following requirements to pass the exercise:

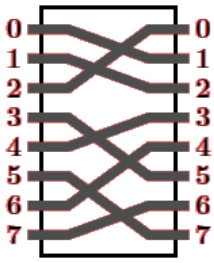
- A unique MAC should be generated each time the Add MAC function is called, i.e. generating a MAC for the same message twice should produce different results.
- Your implementation should be resistant to replay attacks, i.e. once the received data has been successfully verified, future attempts to verify the same received data should fail.

Tutorial 7 – Security



(a) Electronic Codebook (b) Cipherblock Chaining (c) Counter

P-Box (Cipherblock Chaining)



Pikachu and Charmander utilize the P-box on the left and Cipherblock Chaining Mode for encryption. Encrypt the binary input 1001 1100 1110 0100(2) with initial vector IV = 0100 1010(2). Enumerate blocks from the left.

- XOR of first 8 bits with initial vector "IV" (arbitrary name):
  - 1001 1100 XOR 0100 1010 = 1101 0110
- P-Boxes are transpositions where the 0th character (left column) regards the most significant B, which will be moved to the position determined by the right column: C1 = 0110 1101 (which we will use to XOR M2)
  - C2' (' denoting in progress) = 0110 1101 XOR 1110 0100 = 1000 1001
  - C2 (after applying P-box) = 01010010
- Final message is concatenation of C1,C2: 01101101 01010010

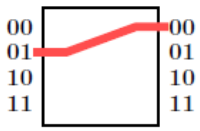
S-Box (Cipherblock chaining)

Eve observes Alice sending encrypted messages to Bob.

- She knows that Alice uses a 2-bit S-box with Cipher block chaining as the mode of encryption.
- plaintext of the first message is 0000
- Each message can have multiple blocks and each message has a different initial vector.
- The first day Eve observes Alice sends the ciphertext 0011 and the initial vector 01 to Bob as the first message of the day.
- the second day, Alice sends the ciphertext 1101 and the initial vector 00 to Bob.
- Finally, Alice sends the ciphertext C=0001 and initial vector 00 to Bob.
- Eve wants to figure out the plaintext corresponding to C.

Which plaintext is C the encryption of?

- 2-bit S-box means that the block size is 2 bits.
- If the first message is 0000, then we have M1:00, M2:00.
- If the first day the ciphertext is 0011, then we have C1:00, C2:11
- We also know IV:01
- Therefore C1 = B-Enc (M1 XOR IV) = B-Enc(00 XOR 01) = B-Enc(01) = C1 = 00, we save this in the S-Box:

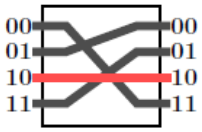


- C2 = B-Enc(00 XOR 00) = B-Enc(00) = 11

- N (as of next day C) N1 = 11, N2 = 01, NV = 00

- N1 = B-Enc (00 XOR 00) = B-Enc(00) = 11. Which matches our first day observation
- N2 = B-Enc(00 XOR 11) = B-Enc(11) = 01

- The last mapping can be inferred by elimination, B-Enc(10) = 10



- In reverse order we now have to decipher M1 and M2 from C1:00, C2:01 and IV:00
- C2 = 01 = B-Enc (M2 XOR C1, which is 00) = 01
  - M2 XOR 00= 11, so M2 =11
- C1 = 00 = B-Enc(M1 XOR IV, which is 00) = 00
  - M1 XOR 00 = 01, M1 = 01
- The message is 0111

Counter Mode

The IV gets appended with 0 or 1 to the right.

- Let an S-box be defined by (00,10), (01,11), (10,00), (11,01)
- wants to send the plaintext message 1010 and chooses the initial vector 1 (so 10, 11, 10, 11…).
- Furthermore, you apply the encryption to the vector, not to the message. Then the encrypted vector is XORed with the message

RSA Recap

Key Generation:

1. Select two large primes p and q
2. Compute n = pq, phi(n) = (p-1)(q-1)
3. Select integer d with gcd(phi(n),d) = 1
4. Find e such that ed = 1(mod phi(n))

Encrypt: C = M^e mod n

Decrypt = M = C^d mod n

What is the corresponding ciphertext?

C1 = B-Enc(10 XOR 10) = B-Enc(00) = 10

C2 = B-Enc(10 XOR 11) = B-Enc(01) = 11

C = 1011

RSA Encryption

- Fred Flintstone wants to use RSA to communicate with Barney.
- Fred chooses primes p = 5 and q = 7. He furthermore chooses e = 5 and d = 5.
- Fred and Barney decide to write only in upper-case letters and encode each letter as an integer corresponding to its position in the English alphabet starting at 1, e.g., 'A' is 1, 'B' is 2, etc.
- Barney uses Fred's public key to encrypt `BEER'.

Which numbers does Barney send to Fred?

1. Convert the message to numbers: B = 2, E = 5, R = 18. So BEER = 2 5 5 18
2. Encode each number M with the following formula: M^e mod n.
    a. It is given that e = 5, and n is pq, which is 5*7 =35
    b. M1 = 2^5 mod 35 = 32
    c. M2 = 5^5 mod 35 = 10
    d. M3 = M2 = 10
    e. M4 = 18^5 mod 35 = 23
3. Barney sends 32, 10, 10, 23

RSA decryption

Fred receives the ciphertext consisting of the numbers 20, 13, 23.

Decrypt and write in letters from the English alphabet

M = C^d mod n = C^5 mod 35. C1 = 20, C2 = 13, C3 = 23.

1. M1 = 20^5 mod 35 = 20 = T
2. M2 = 13^5 mod 35 = 13 = M
3. M3 = 23^5 mod 35 = 18 = R

Tutorial 7 – MCQ

- In a responsible disclosure process, if the developer has not fixed an issue until the deadline then you publish the vulnerability
- Counter mode is an encryption mode that allows parallelizable processing
    - If parallel is not an option then cipherblock chaining and counter mode are both as good
- Formally using blood (or any body part) is regarded as “something you are” factor.

Security Exam questions

P-box matrix (and Counter Mode)

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- An alternative representation of a P-box is a permutation matrix, i.e., a matrix whose rows and columns all have exactly one 1 and all other entries are 0.
- Applying the P-box then corresponds to multiplying the matrix with the binary input.
- Tom and Jerry use the following matrix as a P-box:
- Tom wants to encrypt the message `0101011000011010` using Counter Mode
- initial vector `1111`

What is the corresponding ciphertext?

IV = 1111, so we have 1111 || (0000 to 1111, || denoting concatenation)

So M:0101011000011010 is split into M1:01010110 and M2:00011010

Then IV1' (' denoting encrypted) = B-Enc(11110000) = Vec(11110000) * Matrix = 10010011.

So C1 = M1 COR IV1' = 01010110 XOR 10010011 = 11000101

Without loss of generality the same we do for C2 but using the values of M2 and IV2' = 100010111

So C2 = 1000 1101, such that C = 1100010110001101

Q&A1

Q: How do sender/receiver know which error (detection/correction) protocol and parameters (parity) for the protocol to use

A: There are standards and they are hardcoded into the implementation of a certain layer. I.e

- 802.3 (Ethernet): 4-byte checksums
- TCP/UDP: Checksums
- Wireless: Error correction

Q: 2-bit and 3 bit errors in Parity blocks can be detected?

A: Can be detected but not always corrected

Q: 4-bit errors in Parity blocks

A: Cannot always be detected

Q: Is the processing delay included in the RTT?

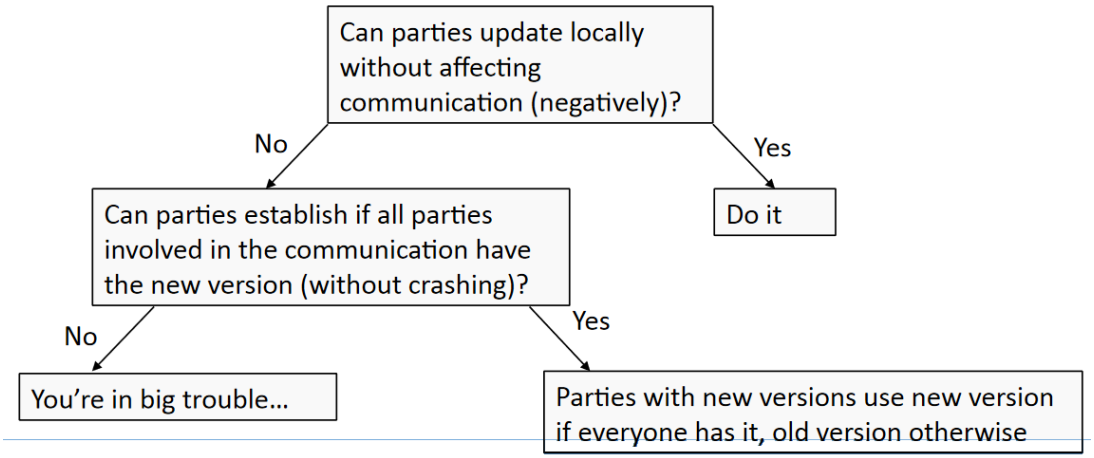
A: In theory no, in practice and in CSE1405 yes.

Q&A2

How to update to new protocol in networks (especially internet)?

Many internet protocols made for situation in the 70-90s

Some of them were developed quite ad-hoc



Q: How do switches know the root MAC address?

A: Solution 1 (common implementation): Switch sends lowest MAC address (initially its own) it

knows to all neighbors If it receives a message with lower MAC address, it updates its own record and forwards new lowest MAC address to neighbors.

Solution 2: (interleave with spanning tree const.): Nodes periodically send messages with

i) lowest MAC address they know, and
ii) cost of best path they know to that MAC address to

all neighbors, who can directly choose parent based on the information.

Deactivated links still need to transmit these periodic updates!

Q: What if a specific switch should be root (i.e., in center of network) that has not the lowest MAC address?

A: Manual configuration: Priority field allows administrator to manually assign different priorities to switches. Root is chosen from switches with highest priority.

Q: Good to know

A: In Ethernet, lost frames are (essentially always) due to collisions. As Ethernet has collision detection, the sender will automatically resend and there is no need for a sliding window protocol. The disadvantage is that stations need to wait longer on average until it's their turn.

Q: Benefits of longer frames:

A: Longer transmission period, higher fraction of time spent on actual transmission, higher effectiveness

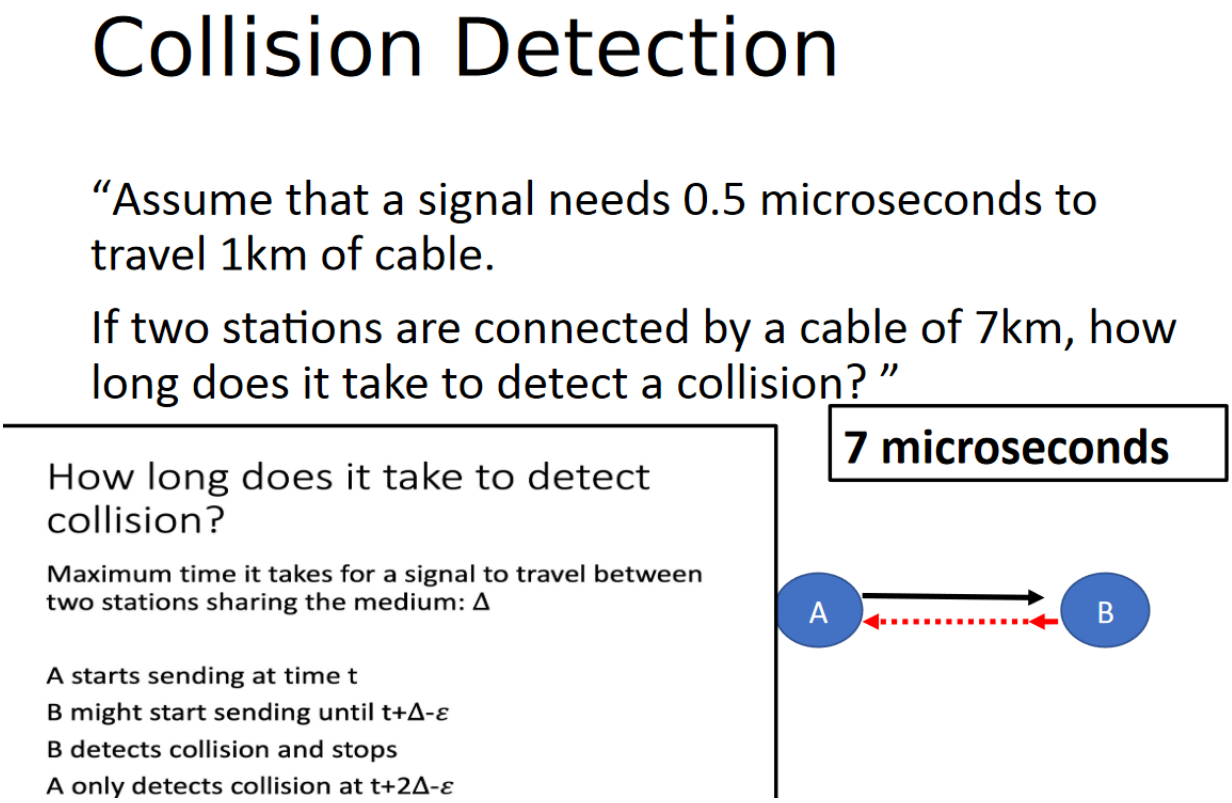


Figure: How long does it take to detect collision

Q: Wireless interference

A:

- X can send to Y and receive ACK without interference (either to their communication or others)
- X cannot send to Y without interference
- X can send to Y but not receive ACK without interference
- X and Y can not send/receive ACK because they are not in range

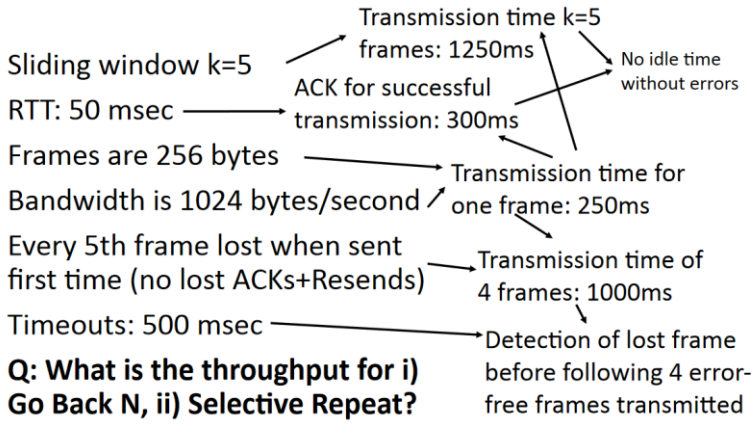
Q: Collision-free protocols - How do stations know that transmission ended? (e.g., are frames all of the same length?)

Short answer: Protocols are intended for carrier-sense, so parties can check if medium is free and won't send if transmission still on-going.

Q: Why are connection-oriented protocols reliable?

A: Connection-oriented protocols apply acks, resends, and re-ordering such that next layer receives complete data in correct order

An exercise



Exercise: Selective Repeat

Transmission time one frame: 250ms

Frames are 256 bytes

We just have one resend but we don't add idle time

⇒For every 5 correctly received frames, sender sent 6

⇒6 frames take 6*250ms = 1500ms = 1.5s

⇒5 frames are 5*256 bytes = 1280 bytes

⇒Throughput: 1280 bytes/1.5s = 853 1/3 bytes/s

Exercise: Go Back N

Transmission time one frame: 250ms

Frames are 256 bytes

Timeout: 500ms

⇒Two frames transmitted before timeout

⇒Three frames are resent (lost + two after)

Still, no idle time

⇒8 frames sent for each 5 correctly received ones

⇒5 frames are 1280 bytes, 8 frames take 2s to transmit

⇒Throughput: 1280 bytes/2s = 640 bytes/s

Q&A 3 (Network layer)

Q: Which layers apply error detection/correction?

A: Error detection/correction Implemented only by some layers:

• Data link layer deals with bitflips or lost data on a link (Weeks 2 - 3)

• Transport layer deals with bitflips or lost data introduced by a device (e.g., router) (Weeks 5 - 6)

Not a responsibility of the network layer, but some routers might do error detection on the header (checksum) to drop packets in advance.

Under congestion, the network layer might randomly drop packets (RED)

Loss indicated by either timeout or 3 duplicate acks

Q: Congestion – Traffic throttling

A:In the internet, the transport layer (TCP) takes care of congestion control (Weeks 5 – 6).

However, IP extensions exist:

• End-to-end: congested router sets a special bit to 1. The receiver notices the bit and sends a choke message back to the sender. Upon reception of the choke message the sender slows down.

• Link-by-link: The receiver also sends a choke message to the sender. Every router that relays the choke message slows down.

Q:Which one is used in practice?

A:By default, packets are dropped. Extensions might be negotiated by the transport layer, if all routers accept them.

Q: Why can't the routers send back a choke message themselves?

A: Some protocols do that (including IP extensions). However, it is not used by default because it adds traffic to an already congested network.

Q: Traceroute and ICMP. ICMP messages are typically used for diagnostic or control purposes. The lecture mentions that Traceroute uses ICMP. How?

A:

• Traceroute aims at discovering the routers that a packet crosses between the local host and a target host

• Sends multiple packets with increasing TTL

• When a router receives a packet with a TTL=1 field it returns an error message to the local host.

• The destination replies with a different error code

- Messages are returned with the original IP header + source ID set by the router

Q: Can you see multiple routes?

A:Yes, if a routing table had several entries of equal cost for the destination. You would notice several hosts at a given distance, not the actual routes.

Q: Role of a router

A: A router might process millions of flows of packets at the same time. Its tasks need to be as simple as possible. This lead to the following decisions:

- Connectionless
- Fixed-size header
- Best-effort service

Some servers implement IP extensions

Q: Quality of service

A:The network layer can also try to meet some quality of service (QoS) requirements.

Possible parameters (cf. Week 1):

- Bandwidth
- Delay
- Jitter
- Loss

Q&A 4 (Transport layer)

Router manipulating packet not detectable by data link layer as trailer/header already removed.

Receiver uses piggybacked ack if they are sending something anyway

Congestion control Signal types

Explicit: Router send messages

Implicit: Host infers from behavior (e.g., loss)

Precise: Derive (host/router) concrete transmission rate

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

Require IP extension

Negotiating IP extensions

Example: ECN

Reminder: ECN sends a choke signal to receiver who echos it back to sender

Step 1: End hosts agree whether to use ECN using TCP header

Step 2: Router marks packet using IP header

Step 3: End host receiving marked packet uses TCP header to inform sending end host

TCP flow control scenarios

Higher latency => Information more outdated, the more data that is actually already processed is considered as part of buffer

High bandwidth => Sender has capacity to send a lot and exhausts data amount granted by window fa

Q: Iterative DNS queries

A: Advantages: low load on server, client can cache intermediary results

Disadvantages: high load on client, higher latency (especially for badly connected clients).

Q: Recursive DNS queries

A: Advantages: Low load on client, Server can cache, Fast (assuming server well connected).

Disadvantages: High load on server, more vulnerable to denial-of-service attacks.

Q&A 5

Commercial VPNs used to hide your IP address

Different goals than company VPNs

Confidentiality and authentication vs. anonymity (location privacy)

Q: What are the private networks?

A: Private networks = one node (gateway/client)

Q:Difference VPN and Proxy

Depends on your definition of proxy.

Definition 1: Proxies do not encrypt.

⇒VPNs do

Definition 2: Proxies only forward web traffic (or other specific form of traffic).

⇒VPNs can be used for all types of traffic

Definition 3: Proxies include any type of relaying traffic in any form.

⇒VPNs act as a specific type of proxy (but they can do more)

Q: TLS vs DNNSEC

Both use hierarchical structure to establish trust using certificates

Diffferences: DNSSEC does not provide confidentiality

TLS relies on TCP and DNS on UDP

Q&A 6 Bitcoin

Bitcoin consensus
What to know for the exam

1. Participants solve puzzle that depends on latest block and transactions for next block
2. Puzzle takes time to solve but solution is fast to verify
3. Unlikely that two parties solve it at the same time
4. Party who solves it decides on next block
5. If there are multiple parties solving it at approximately the same time: valid block is decided later as the one most subsequent blocks were appended to (Longest Chain Rule)