Network and Cybersecurity
Assignment 1

Kristoffer Klokker

2022

Contents

1 Problem - Propagation and Transmission delay

Consider two hosts, A and B where

- They are connected by a single link and rate R bps
- They are separated by m meters
- With a propagation speed of s meter/sec
- A is sending B a packet of size L

1.1 Express the propagation delay d_{prop} in terms of m and s

$$d_{prop} = \frac{m}{s}$$

Propagation delay describe the time, it takes for a bit to propagate between two routers. This can therefore be described by the distance divided by the propagation speed.

1.2 Determine the transmission time of the packet, d_{trans}, in terms of L and R.

$$d_{trans} = \frac{L}{R}$$

Transmission delay describe the time it takes for the router to transfer the package to the link. Therefore it depends on the transmission rate and size of the package.

1.3 Ignoring processing and queuing delays, obtain an expression for the end-to-end delay.

$$d_{end-to-end} = d_{trans} + d_{prop} = \frac{m}{s} + \frac{L}{R}$$

The total end-to-end delay will then be equal to the sum of transmission delay and propagation delay.

1.4 Suppose Host A begins to transmit the packet at time $t = 0$. At time $t = d_{trans}$, where is the last bit of the packet?

The last bit will just have been transfered to the link between host A and B.

1.5 Suppose d_{prop} is greater than d_{trans}. At time $t = d_{trans}$, where is the first bit of the packet?

The first bit of the packet will always be in the link between the two hosts at $t = d_{trans}$. The first bit will not be able to arrive at host B since $d_{prop} > d_{trans}$.

1.6 Suppose d_{prop} is less than d_{trans}. At time $t = d_{trans}$, where is the first bit of the packet?

In the scenario of a theoretical 1 bit package, it would be in the link between the hosts.
In all other scenarios with a package size larger than 1 bit, the first bit will be at host B.

1.7 Suppose $s = 2.5 \cdot 10^8$ m/s,$L = 1200$ bytes, and $R = 5$Mbps. Find the distance m so that $d_{prop} = d_{trans}$

$$d_{prop} = d_{trans} \tag{1}$$
$$\frac{m}{s} = \frac{L}{R} \tag{2}$$
$$R = 5 \cdot 10^6 \text{bit/s} \tag{3}$$
$$s = 2.5 \cdot 10^8 \text{m/s} \tag{4}$$
$$L = 1200\text{byte} = (1200 \cdot 8)\text{bit} \tag{5}$$
$$\frac{m}{2.5 \cdot 10^8 \text{m/s}} = \frac{9600\text{bit}}{5 \cdot 10^6 \text{bit/s}} \tag{6}$$
$$\frac{m}{2.5 \cdot 10^8 \text{m/s}} = \frac{6}{3125}\text{s} \tag{7}$$
$$m = 2.5 \cdot 10^8 \text{meter/s} \cdot \frac{6}{3125}\text{s} \tag{8}$$
$$m = 480000\text{m} \tag{9}$$

First the equation is expanded (1) $\rightarrow$ (2). Then every variable is defined and converted to the same units without prefixes.
The variables are inserted (6) and reduced (7-9).
The distance of the cable between host A and B is must be $4.8 \cdot 10^8$ meter long for the propagation delay is equal to the tranmission delay.

2 Problem - HTTP and Sockets

2.1 Describe what each of these HTTP Request methods are used for

- GET - The HTTP GET request is used for getting data. The request should not include any data, other than the required header information for the HTTP request.
- POST - The HTTP POST request is used for sending data to a server.
- HEAD - Like the HTTP GET request it is used for getting data from the server. The difference is the HEAD request only request the header and not the body, therefore making it a faster request than GET. Mostly used by the browser to check for required updates in cached information.
- PUT - Like the POST this request is used for transferring data. It differ from POST by instead of creating new resources the PUT replaces or updates existing resources.
- PATCH - Like PUT this request is used for updating resources. It differs by instead of uploading a resources and updating from it, rather a patch document is used, containing updates which will be applied.

2.2 Create a simple program, which creates a TCP HTTP GET request and retrieves the HTML document from 'http://wireshark.grydeske.net/'

```
01 | import java.io.BufferedReader;
02 | import java.io.IOException;
03 | import java.io.InputStreamReader;
04 | import java.io.PrintStream;
05 | import java.net.InetAddress;
06 | import java.net.Socket;
07 | import java.net.URL;
08 |
09 | public class TCPRequest {
10 |     public static void main(String[] args) {
11 |         connectToTCPServer("http://wireshark.grydeske.net");
12 |     }
13 |
14 |     private static void connectToTCPServer(String url) {
15 |         Socket socket = null;
```

```
16 |         PrintStream out = null;
17 |         BufferedReader in = null;
18 |
19 |         try {
20 |             String host = new URL(url).getHost();
21 |             InetAddress ia = InetAddress.getByName(host);
22 |             String ip = ia.getHostAddress();
23 |             socket = new Socket(ip,80);
24 |             out = new PrintStream(socket.getOutputStream());
25 |             in = new BufferedReader(new InputStreamReader(
   socket.getInputStream()));
26 |
27 |             out.println("GET / HTTP/1.1");
28 |             out.println("Host: " + host);
29 |             out.println("Connection: close");
30 |             out.println(); //Empty line
31 |
32 |             String line = in.readLine();
33 |             while (line != null) {
34 |                 System.out.println(line);
35 |                 line = in.readLine();
36 |             }
37 |         }
38 |         catch(IOException e) {
39 |             e.printStackTrace();
40 |         }
41 |         finally {
42 |             try {
43 |                 if(socket != null) socket.close();
44 |                 if(out != null) out.close();
45 |                 if(in != null) in.close();
46 |             }
47 |             catch (IOException e) {
48 |                 e.printStackTrace();
49 |             }
50 |         }
51 |     }
52 |
53 | }
```

3 Problem - DNS

Use nslookup to obtain the authoritative answer for the ip address of nfl.com

```
~nslookup -type=soa nfl.com
Server:  UnKnown
Address:  103.86.96.100

Non-authoritative answer:
nfl.com
        primary name server = udns1.ultradns.net
        responsible mail addr = dns_admin.nfl.com
        serial  = 2017192735
        refresh = 60 (1 min)
        retry   = 60 (1 min)
        expire  = 3600000 (41 days 16 hours)
        default TTL = 300 (5 mins)

~nslookup nfl.com udns1.ultradns.net
Server:  udns1.ultradns.net
Address:  204.69.234.1

Name:    nfl.com
Addresses:  151.101.65.153
          151.101.1.153
          151.101.193.153
          151.101.129.153
```

Figure 1: Obtaining the authoritative answer for the ip address of nfl.com

4 Problem - RSA

For the following RSA the values are selected

- $p = 911$
- $q = 883$
- $e = 503$

4.1 Describe these criteria for e and d

To describe the criteria first the variable n can be calculated.

$$n = pq = 911 \cdot 883 = 804413$$

Here $e < n$ and the greatest common divider between e and n must be 1.
The criteria for d is $ed \mod z = 1$ where $z = (p-1)(q-1) = 910 \cdot 882 = 802620$

4.2 Argue e is a valid choice and calculate d using the extended euclidean algorithm

It can be seen that $e < n \rightarrow 503 < 804413$ holds true
To check if the greatest common divider between e and n is 1, the euclidean algorithm can be used:

$$\begin{aligned} gcd(804413, 503) \\ 804413 &= 503 \cdot 1599 + 116 \\ 503 &= 116 \cdot 3 + 5 \\ 116 &= 5 \cdot 23 + 1 \\ 5 &= 1 \cdot 5 + 0 \\ gcd(804413, 503) &= 1 \end{aligned}$$

It can therefore be seen that e is a valid choice for the algorithm since both criteria is fulfilled.

To find a value d the inverse euclidean algorithm can be performed

$$\begin{aligned}
503d \mod 802620 &= 1 \\
503d \equiv 1 \mod 802620 \\
d \equiv \frac{1}{503} \mod 802620 \\
d \equiv 503^{-1} \mod 802620 \\
\text{Using extended euclidean algorithm} \\
802620 &= 503 \cdot 1595 + 335 \\
503 &= 335 \cdot 1 + 168 \\
335 &= 168 \cdot 1 + 167 \\
168 &= 167 \cdot 1 + 1 \\
\\
\text{Begin substituting} \\
1 &= 168 - 167 \\
1 = 168 - (335 - 168) &= 168 - 335 + 168 \\
1 &= 168 \cdot 2 - 335 \\
1 &= (503 - 335) \cdot 2 - 335 \\
1 &= 503 \cdot 2 - 335 \cdot 3 \\
1 = 503 \cdot 2 - (802620 - 503 \cdot 1595) \cdot 3 \\
1 &= 503 \cdot 2 - 802620 \cdot 3 + 503 \cdot 4785 \\
1 &= 503 \cdot 4787 - 802620 \cdot 3 \\
1 &= 503 \cdot 4787 \\
\\
503^{-1} \equiv 4787 \mod 802620 \\
d &= 4787
\end{aligned}$$

It can be found that a valid value for d is 4787

4.3 Show how you can encrypt the number 3211. Show also how to decrypt your result

Encryption: $3211^e \mod n \rightarrow 3211^{503} \mod 804413 = 40516$
Decryption: $40516^d \mod n \rightarrow 40516^{4787} \mod 804413 = 3211$

5 Problem - Modes of operation for Block Ciphers

Explain how each of these 5 modes of operation works.
For the following explainations the following denotions are used:

- C_i - Cipher for block i
- p_i - Plaintext chunk i
- c_i - Ciphertext chunk i

The cipher has the index, in case of use of a indexed key.

5.1 ECB

Electronic codebook (ECB) mode works by dividing the input into appropriate chunk size for the block cipher.
Then each block is individually used as input for the block and the output will be the encrypted/decrypted text.

$$c_i = C_i(p_i)$$
$$p_i = C_i(c_i)$$

The problem with this mode, is that in the case of duplicate chunks a pattern can be recognized.

5.2 CBC

Cipher Block Chaining (CBC) mode works by first a random k-bit string is generated called initialization vector (IV) and shared in plain text.
Then the first block of input is XOR'ed with the IV. The following blocks is then XOR'ed with the last blocks output.

$$c_0 = C_0(p_0 \oplus IV)$$
$$c_i = C_i(p_i \oplus c_{i-1})$$

For the decryption the first block is decrypted an XOR'ed with the IV. Then the following blocks are decrypted and XOR'ed with the previous input block.

$$p_0 = C_0(c_0) \oplus IV$$

$$p_i = C_i(c_i) \oplus c_{i-1})$$

5.3 CFB

Cipher Feedback (CFB) mode works by first a random k-bit string is generated called initialization vector (IV) and shared in plain text.
The IV is then encrypted in the first block, and the output is XOR'ed with the first plain text chunk.
The XOR result is then the first ciphertext chunk. For the following cipher block the previous XOR result is used for input to the next block and the output is XOR'ed with the next plain text chunk.

$$c_0 = C_0(IV) \oplus p_0$$
$$c_i = C_i(c_{i-1}) \oplus p_i$$

For the decryption first the IV is used as input for the block and the result is XOR'ed with the first cipher text chunk, which gives the first plain text chunk.
For the following the ciphertext is used for input for the block and the result is XOR'ed with the next ciphertext chunk.

$$p_0 = C_0(IV) \oplus c_0$$
$$p_i = C_i(c_{i-1}) \oplus c_i$$

5.4 OFB

Output Feedback (OFB) mode works by first a random k-bit string is generated called initialization vector (IV) and shared in plain text.
This IV is then used as input for the first block cipher and the result is XOR'ed to create the cipher text.
For the following ciphers the block cipher output from the previous block is used for input and the plaintext chunk is XOR'ed with the output to create the ciphertext.

$$c_0 = C_0(IV) \oplus p_0$$
$$c_i = C_i(c_{i-1} \oplus p_{i-1}) \oplus p_i$$

For this the decryption the same steps are repeated the only difference is the output of the block cipher is XOR'ed with the cipher text to create the plain text.

$$p_0 = C_0(IV) \oplus c_0$$
$$p_i = C_i(p_{i-1} \oplus c_{i-1}) \oplus c_i$$

Note that for c_i and p_i the input to the cipher is the last ciphertext XOR'ed with the last plaintext which is equal to the last cipher output, but to simplify the equation and denotion the XOR cancellation is implicit.

5.5 CTR

Counter (CTR) mode works by first a random nonce is generated. Then a counter is initiated and synchronized between the encrypter and decrypter. The counter is then incremented for each block. The block then gets the sum of the nonce and counter as input. The output is the XOR'ed with the plaintext/ciphertext the encrypt/decrypt.

$$c_i = C_i(\text{Nonce} + \text{counter}) \oplus p_i$$

$$p_i = C_i(\text{Nonce} + \text{counter}) \oplus c_i$$

The advantage of CTR is the ability to run parallel unlike CBC, CFB and OFB, where the input is dependent on the result of another cipher.