Computer Networks Fall 2016
Problem Sheet #1

Tom Wiesing

September 14, 2016

1 Problem 1.1: optical fibre transmission

You are deploying an optical fiber across Germany with a length of $l = 790\text{km}$. The optical fiber has a data rate of $r = 10\text{Gbps} = 10^{10}\text{bps}$. The signal propagation speed in the optical fiber is $v = 2{\cdot}10^8 \frac{\text{m}}{\text{s}}$.

1.1 a) How many bits can be simultaneously in transit on the wire?

A bit exits transit after travelling at most the length of the entire fibre. Any bit takes at most $\tau = \frac{l}{v} = \frac{79{\cdot}10^4\text{m}}{2{\cdot}10^8 \frac{\text{m}}{\text{s}}} = 3.95{\cdot}10^{-3}\text{s}$ to do this. A situation where the maximal amount of bits are in transit simultaneously occurs if the fibre constantly transmits bits at the maximal data rate during this time. Thus at most $r{\cdot}\tau = 10^{10}\text{bps}{\cdot}3.95{\cdot}10^{-3}\text{s} = 3.95{\cdot}10^7\text{b}$ can be in transit at the same time.

1.2 b) What is the round-trip-time imposed by the propagation delay of the wire?

We have already computed the propagation delay τ above to be $3.95{\cdot}10^{-3}\text{s}$. The round-trip-time is $2\tau = 7.9{\cdot}10^{-3}\text{s}$.

1.3 c) Two former Jacobs Students are now located in Berlin and Seattle. They both measure the round-trip-time to Google's DNS server (IPv4 address 8.8.8.8). The person located in Berlin measures a minimum round-trip-time $rtt_B = 6\text{ms}$ while the person in Seatlle measures a minimum round-trip-time of $rtt_S = 12\text{ms}$. Are they sending packets to the same server?

The available information is insufficient to determine if the are sending packets to the same server, although it is likely that they are not.

If the persons are sending packets to the same server, then packet travelling from Seattle to Berlin would take an average of $\frac{1}{2}(rtt_B + rtt_S) = 9\text{ms}$ to arrive. Since Seattle and Berlin have a distance of 8141km, so if there was an optical fiber of the same specification as above connecting the two locations this distance

would need at minimum $\frac{v}{8141\text{km}} = 40.71\text{ms}$. That means the cable is not fast enough to deliver packages at that speed.

Usually a single server is responding to each IPv4 adress, but according to the Google DNS FAQ[1] Google uses AnyCast to route to the geograpically nearest server. Since Berlin and Seattle are seperated by a significant distance, the persons will likely send packages to different servers.

2 Problem 1.2: crc checksums

A transmission link uses a cyclic redundancy check (CRC) code to detect transmission errors. The generator polynomial is $G(x) = x^4 + x^2 + 1$. The bit sequence 1101011001011010 has been received by a network interface card. Does the message pass the CRC check? Show why or why not.

In binary, the generator corresponds to 10101. We perform a CRC check below.

```
G =  10101
M =  1101011001011010
      11010
      10101
      _____
      011111
       10101
       _____
       010101
        10101
        _____
        000000
         10101
         _____
         10101
         10101
         _____
         000000
          10101
          _____
          10101
          10101
          _____
          000001
           10101
           _____
           10100
           10101
           _____
           000010
            10101
            _____
```

[1] https://developers.google.com/speed/public-dns/faq#anycast

```
10111
10101
------
000101
 10101
 ------
 10000
 10101
 ------
 001011
  10101
  ------
  11110
  10101
  ------
  010110
   10101
   ------
   000111
    10101
    ------
    10010
    10101
    ------
    001110
     10101
     ------
     11011
     10101
     ------
     01110
```

As we get the non-zero remainder 1110, the bit sequence does not pass the CRC check.

3 Problem 1.3: ARQ protocols

Automatic Repeat reQuest (ARQ) is an error-control method for data transmission that uses acknowledgements and timeouts to achieve reliable data transmission over an unreliable service. If the sender does not receive an acknowledgment before the timeout, it usually re-transmits the frame until the sender receives an acknowledgment or exceeds a predefined number of retransmissions.

Well-known types of ARQ protocols are

- Stop-and-wait ARQ,
- Go-Back-N ARQ, and
- Selective Repeat ARQ.

Read about these three well-known ARQ protocols and briefly summarize how they work.

3.1 Stop-and-wait ARQ

In this protocol data frames are sent from sender to receiver one by one. Each frame has an internal CRC check. Upon receiving a frame, the receiver sends back an ACK (acknowledgement) packet if the CRC check passes. If the CRC check does not pass, or the frame is lost along the way, no such package is sent. The sender expects to receive an ACK package for each frame it sends. If it does not receive such a confirmation after a fixed timeout, it re-sends the data frame. The name of this protocol originates from this behaviour, the sender stops and waits for the acknoledgement.

3.2 Go-Back-N ARQ

Contrary to the stop-and-wait ARQ above, packages are not send one-by-one.

In this protcol, each frame has a continuous sequence number. This is used by the receiver to keep track of the number of frames received. Upon receiving a packet with the expected sequence number, the server sends an acknowledgement for it. If it receives a broken, or out-of-order packet, it instead sends an acknowledgement for the last correctly sent packet. The sender sends N sequential packages. After sending all of them, it can then detect the last successfull ACK it received and start sending again from that point onwards.

3.3 Selective Repeat ARQ

Compared to the Go-Back-N protocol above, in this protocol the sender only resends those frames for which is has not received an acknolegement. The receiver has to store all packages, even if they are out-of-order, and re-assemble them in the correct order afterwards.

4 Problem 1.4: stop-and-wait ARQ efficiency

Given a fixed frame size of n_f bits with an overhead (header, checksums) of n_o bits, an acknowledgment size of n_a bits, the propagation delay t_p, the frame transmission delay t_f , the acknowledgment transmission delay t_a and the processing delay t_c (for both frames and acknowledgments), and the data rate r, what is the time t that passes from the start of the frame transmission until the acknowledgment has been processed? Assume that no transmission errors occur.

The effective data rate r_e is the amount of information bits transmitted per time unit, i.e., the useful information transmitted without any overhead bits or acknowledgment bits. Derive an expression for the effective data rate r_e and the efficiency e, which is obtained by normalizing the effective data rate r_e by the data rate r, i.e. $e = \frac{r_e}{r}$.

In stop-and-go-ARQ, we first need to send the data-frame with overhead, i.e. a total of $n = n_o + n_f$ bits. This first has a frame transmission delay of t_f and then at a data rate of r, needs $\frac{n}{r} = \frac{n_o+n_f}{r}$ seconds to be fully transmitted and a time of t_p to reach the destination. We then have a a processing delay of t_c. Thus to successfully send the original frame from sender to receiver we need $t_f + \frac{n_o+n_f}{r} + t_p + t_c$. Next we send back the acknowledgement (size n_a) with transmission delay t_a for $\frac{n_a}{r}$ with a propagnation delay of t_p. We then again

have a processing delay of t_c. For sending the ACK we thus need $t_a + \frac{n_a}{r} + t_p + t_c$. Overall this gives a time of

$$t = t_f + t_a + 2t_p + 2t_c + \frac{n_o + n_f + n_a}{r}$$

from the start of the frame transmission until the acknowledgment has been processed.

To compute the effective data rate r_e, we notice that only n_f pieces of data have been transmitted in time t. Hence

$$r_e = \frac{n_f}{t} = \frac{n_f}{t_f + t_a + 2t_p + 2t_c + \frac{n_o + n_f + n_a}{r}}$$

. The efficiency e is given by

$$e = \frac{r_e}{r} = \frac{\frac{n_f}{t_f + t_a + 2t_p + 2t_c + \frac{n_o + n_f + n_a}{r}}}{r} = \frac{n_f}{r\left(t_f + t_a + 2t_p + 2t_c\right) + n_o + n_f + n_a}$$