CS 536 Fall 2016: Hw2 / Lab2

Maria L. Pacheco

September 28, 2016

1 Hw2

1.1 Problem 1

1.1.1 Stop-and-wait's average/expected reliable throughput in a slightly more realistic scenario.

- ACK frames do not suffer errors
- Data frames are not received with $p = 0.01$
- When transmission fails, re-transmission always succeeds.
- RTT estimation is perfect

$$\text{reliable\_throughput} = \text{data\_bits}/\text{total\_time}$$

(Peterson and Davies, 2015)

In data bits we need to consider: the size of the frame (bits). In the total time, we need to consider: the round trip delay time RTT (in sec) for a frame and the possibility of having to resend that frame with probability $p = 0.01$. ACK traffic is not counted and data bits do not count re-transmissions.

$$\text{expected\_reliable\_throughput} = \frac{\text{frame\_size}}{(RTT + p * RTT)}$$

$$\text{expected\_reliable\_throughput} = \frac{\text{frame\_size}}{RTT(1 + p)}$$

1.1.2 Generalization

- Transmission of data frames fail with independent (across successive transmission attempts) probability p.

  In this case, each successive attempt can be seen as a Bernoulli trial with independent probability of success $q = 1-p$. Then we need to estimate the expected number of failures **before** the first success, which is the expected value of the Geometric Distribution $E(Y) = (1 - q)/q$.

$$\text{expected\_reliable\_throughput} = \frac{\text{frame\_size}}{RTT + RTT\ \frac{1-q}{q}}$$

$$\text{expected\_reliable\_throughput} = \frac{\text{frame\_size}}{RTT\ (1 + \frac{1-q}{q})}$$

$$\text{expected\_reliable\_throughput} = \frac{\text{frame\_size}}{RTT\ (1 + \frac{p}{1-p})}$$

1.2 Problem 2

To support 4 users and their simultaneous bit transmissions using the CDMA-like approach we could use 4 orthogonal vectors in 4D to serve as code vectors.

- v_0: (1, -1, -1, -1)
- v_1: (-1, 1, -1, -1)
- v_2: (-1, -1, 1, -1)
- v_3: (-1, -1, -1, 1)

Since v_i is the code vector of receiver r_i. Taking advantage of the orthogonality of the code vectors, we can encode the 4 bits in a single message vector in a way that each of our receivers can perform a dot product between the message vector and the code vector. If the result is positive, the sent bit is a 1, if the result is negative, the sent bit is a 0.

This encoding can be done by multiplying each code vector by 1 in case of a 1 bit and by -1 in case of a 0 bit and sum them up. For example, to send 1 to r_1 and 0 to the rest:

$$(+1)v_0 + (-1)v_1 + (-1)v_2 + (-1)v_3 = (4, 0, 0, 0) = m$$

Then the receiver can decode his/her message by performing $m \cdot v_i$. Taking the above example where $m = (4, 0, 0, 0)$, we have:

- $m \cdot v_0 > 0$
- $m \cdot v_1 < 0$
- $m \cdot v_2 < 0$
- $m \cdot v_3 < 0$

By using this method to decode messages on the receiver end, **orthogonality** will cause all terms to vanish except for the one regarding the encoded bit for a specific receiver. If the vectors were not orthogonal, this would not be possible and information for other receivers might interfere.

The issue encountered without orthogonality is related to sending bits using amplitude modulation (AM) of electromagnetic waves modeled as complex sinusoids, where Fourier Transforms are used to perform the "dot product". If the signal bandwidth of different frequencies overlap and the amplitude detected by receiver is distorted, it will cause weight from different signal spectrum to be added. This is analogous to having terms that don't vanish when performing the dot product. This is prevented by:

- Putting neighboring carrier frequencies far apart
- Using orthogonal sinusoids (same solution!)

(Park 2016)

1.3 Problem 3

1.3.1 AM frequencies and its quality

AM radio carrier frequencies are in the frequency range 535-1605 kHz. Carrier frequencies of 540 to 1600 kHz are assigned at 10 kHz intervals (Nave 2012). However, most AM radio receivers will reproduce about to 4.5kHz, which is even lower. The human range is commonly given as 20 Hz-20 kHz, this might explain the low quality perception. Additionally, AM radio is susceptible to interference from the physical world.

1.3.2 AM radio vs. FDMA

FDMA is used to allocate channels to users, assigning them a specific frequency band. FDMA is used in AM radio to allow different stations to broadcast their signal.

1.3.3 AM radio becomes digital

The key difference is that AM radio deals with analog signals, which are continuous and digital signals are represented with binary numbers. Current AM receivers would need to be replaced with receivers that were built to interpret digital signals instead. Digital radio might increase the quality by providing more control and error correction capacities. However, the quality would still be susceptible to the bit rate used for transmission. On the other hand, the digital processing might cause delays.

2 Lab2

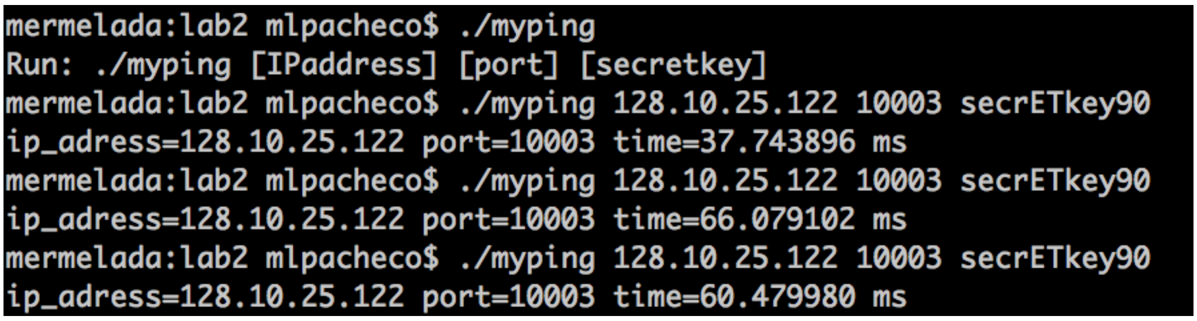
2.1 Problem 2

Each client-server pair will use a dedicated socket, for this reason, the interleaving problem encountered in the named pipe solution does not remain an issue here.

2.2 Bonus Problem

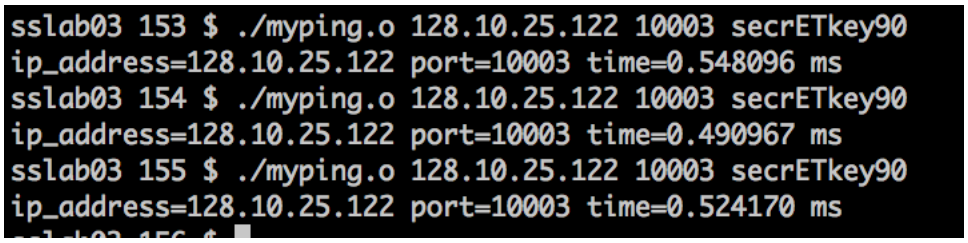
From a Macbook Air, Early 2015 running OSX El Capitan. I am able to perform ping, from an internet connection in my apartment to the cs lab machines without altering my code. The results can be observed in the figure below.

Figure 2.2 has a screen shot of the ping from my laptop in my appartment to `sslab22.cs.purdue.edu`. Figure 2.2 has a screen shot of the ping from `sslab03.cs.purdue.edu` to `sslab03.cs.purdue.edu`. We can observe a big difference in the transmission time.



```
mermelada:lab2 mlpacheco$ ./myping
Run: ./myping [IPaddress] [port] [secretkey]
mermelada:lab2 mlpacheco$ ./myping 128.10.25.122 10003 secrETkey90
ip_adress=128.10.25.122 port=10003 time=37.743896 ms
mermelada:lab2 mlpacheco$ ./myping 128.10.25.122 10003 secrETkey90
ip_adress=128.10.25.122 port=10003 time=66.079102 ms
mermelada:lab2 mlpacheco$ ./myping 128.10.25.122 10003 secrETkey90
ip_adress=128.10.25.122 port=10003 time=60.479980 ms
```

Figure 1: Performing ping from my laptop in my apartment to the sslab machines



```
sslab03 153 $ ./myping.o 128.10.25.122 10003 secrETkey90
ip_address=128.10.25.122 port=10003 time=0.548096 ms
sslab03 154 $ ./myping.o 128.10.25.122 10003 secrETkey90
ip_address=128.10.25.122 port=10003 time=0.490967 ms
sslab03 155 $ ./myping.o 128.10.25.122 10003 secrETkey90
ip_address=128.10.25.122 port=10003 time=0.524170 ms
```

Figure 2: Performing ping between two machines in sslab

3 References

- Fall16 CS536 Lecture notes, Purdue University. Kihong Park, 2016
- Computer Networks: a Systems approach. Peterson and Davies, 2015
- HypherPhysics, Georgia State University. Carl Nave, 2012