EE249: Homework 1

Fall 2010

General Comment: An answer to a problem is neither right nor wrong (but there are exceptions of course). I would rather consider your homework as solutions to design problems. I could argue that your solution is inefficient and you could argue the same about mine. Grade is established mostly on the reasoning that you follow to answer the questions. Hence, it is in your interest to justify your claims. You can use any kind of sources as long as you include references.

1 A Digital Modulator

We assume that there is a source that generates encoded words $w \in V$ where V is a vocabulary of words. For instance, the vocabulary can be composed of strings of 5 bits. Each word is mapped to a pair of numbers by a mapper which is an injective function $f : V \to X \times Y$ where $X, Y \subset Z$. Each pair $(x, y) \in X \times Y$ is a point in the Euclidean plane. The range of the function Im(f) is called a constellation. The two signals generated by the mapper are called the in-phase and quadrature components of the input signal and are denoted respectively by i and q. It is possible to send this complex signal on the air using a modulator. A modulator multiplies the in-phase component by a cosine and the quadrature component by a sine and then adds the two signals together. The functionality of this system can be written in the following way:

$$y(n) = f(w(n))|_x cos(\frac{2\pi n}{T}) - f(w(n))|_y sin(\frac{2\pi n}{T})$$

where $f|_x$ is the projection of f on the first component and $f|_y$ is the projection of f on the second component. An architecture to implement this function is shown in Figure 1.

QUESTION 1: Describe the system using the CFSM model of computation. In particular write each block in the diagram as a CFSM. You may assume to have two functions fx and fy that can be called on transitions of the mapper CSFM.

QUESTION 2: Describe the same system using the dataflow model of computation.

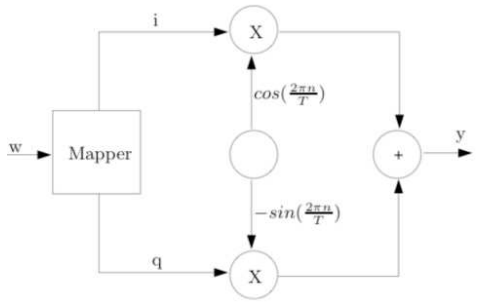

Figure 1: Block diagram of the modulator.

2 Sender and Receiver

The block diagram of the sender and receiver system is shown in Figure 2. We want to design two protocol interfaces. One scenario of this simple protocol is described in Figure 3. This scenario is triggered by an input of the TXInterface which is an array of n words. When the interface receives the array, it stores the array in an internal state variable and starts sending it, word by word, to the destination interface. A transmission is initiated by the TXInterface with a TXRequest signal. The receiver answers with an acknowledge. Then the TXInterface sends a series of data (depending on n) waiting for an acknowledge each time a data is sent. To close the session, the TXInterface sends and end of transmission signal and, before going to idle again, TXInterface waits for the last acknowledgment.

QUESTION 1: Describe the two interfaces using the CFSM model of computation.

QUESTION 2: Describe the two interfaces using the dataflow model of computation. For this part you may want to consider the two interfaces as two dataflow graph of finer grain actors.

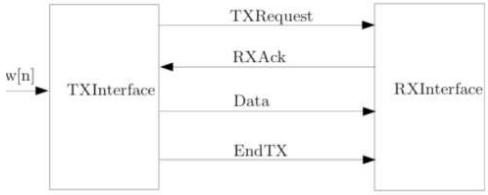

Figure 2: Block diagram of the sender-receiver system.

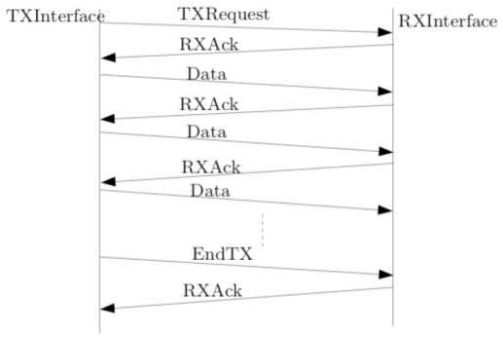

Figure 3: Activity diagram of the protocol.

3 Modeling a Packet Processing Subsystem

There are many ways of transferring data through a network. The network that you are most familiar with, the Internet, is a packet switched connectionless network. It is connectionless in the sense that given a source host and a target host, there is no need to establish a connection through the network in order to send data from one to the other. In other networks, like the one based on ATM (Asynchronous Transfer Mode), a connection has to be established first. During the connection setup, all the network resources needed to transfer packets from source to destination are reserved for that connection. When the transmission ends, all the resources are released and made available for other connections. The Petri Net in Figure 4 shows the model of a connection-oriented packet processing.

Question 1 : Give an intuition of how the system works. Now that you have an intuition of what the system is supposed to do, you should be able to do some formal analysis on it. Let the transitions be ordered as follows: (in,start,do,done,outt,release,connect) and the place as follows: (buffer,proc,conn,unconn,count,outp)

Question 2 : Prove that it is not possible to process a packet if the connection has not been established. Formally, marking (0,57,0,1,0,0) cannot be reached from initial marking (0,0,0,1,0,0).

Question 3: Is this a Free Choice Petri Net? Now it is time to actually generate code. Find a periodic schedule for the network using the method that you like and then answer the last question.

Question 4: Write a C program that simulate the Petri Net model. Assume that each transition is a task that only consumes and produces tokens.

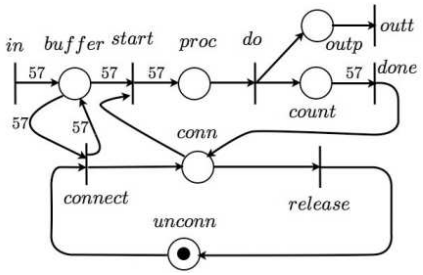

Figure 4: Petri-Net model of the connection-based packet processor.