Mandatory Assignment 3

INF 2140, Spring 2012

Due: 27.04.2012

Published Date: 13.04.2012

COMMUNICATION PROTOCOLS

1. **Unreliable channels:**

   Problem and description: In order to transmit data between two distant points, communication protocols and channels between them are needed. Some communication protocols are intended to work for unreliable channels. Assume that there are two Java processes called sender S and receiver R; S wants to transmit one package containing data to R, and R wants to send a confirmation acknowledgment to S after it has received the data, additionally assume that S and R communicate via two unreliable channels K (to send data) and L (to send acknowledgments), as it is shown in Figure 1.

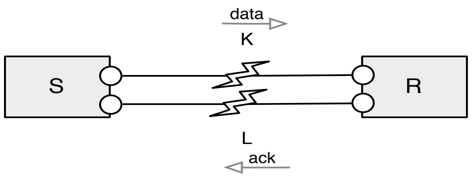

Figure 1

   In this mandatory assignment you are provided with a class implementing an unreliable channel with the following interface:

```
public interface ChannelInterface<T> {
   public void send(T v) throws InterruptedException;
   public T receive() throws InterruptedException;
}
```

   This means that K and L in Figure 1 are two channel objects of the class that implements the ChannelInterface. When S wants to transmit a message containing data, it calls the send method of channel K and when it wants to get a message containing an acknowledgement, it calls the method receive of channel L. In the same way, when R wants to get a message containing data, it calls the method receive of K and when it wants to transmit a message containing an acknowledgment it calls the method send of L.

IMPORTANT NOTE: In the description of the exercises we have used one character name for the processes (i.e., K, L, S, R, etc). We recommend to use longer names for processes in your solution to avoid errors in the LTSA tool.

Exercise 1: In order to analyze any communication protocol for unreliable channels in FSP, you first need to model the unreliable communication environment.

Figure 2 gives the structure diagram for an unreliable communication environment. Model the processes K and L as two unreliable channels, K manipulates data and L manipulates acknowledgments. K and L should satisfy the following safety/progress properties:

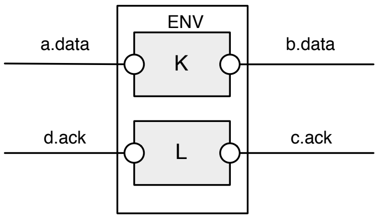

Figure 2

- A message from channel a to channel b can be lost (K receives one message from a, but K will not send that message through b), can be duplicated (K receives one message from a, and K will send the message more than once through b) and can be sent properly (K receives one message from a, and after that K sends that message through b).
- In order to avoid deadlock due to unlimited message loss and unlimited message duplication; a message from a to b will eventually be sent properly (without loss and duplication). This means that the receiver should eventually get a message containing data even under unfair conditions where the properly sending branch has low priority.
- A message from channel c to channel d can be lost (L receives one message from c, but L will not send that message through d), can be duplicated (L receives one message from c, and L will send the message more than once through d) and can be sent properly (L receives one message from c, and after that L sends that message through d).
- In order to avoid deadlock due to unlimited message loss and unlimited message duplication; a message from c to d will be eventually sent properly (without loss and duplication). This means that the sender should eventually get a message containing ack even under unfair conditions where the properly sending branch has low priority.

2. Alternating bit protocol:

Description: To overcome the shortages of unreliable channels (message loss and message duplication), the sender and the receiver processes will implement the *Alternating Bit Protocol* (ABP).

In this scenario, the sender S wants to send a sequence of messages to the receiver R assuming an unreliable environment (with message loss and message duplication). Each message from S to R contains a data part and a one-bit sequence number (i.e., the value 0 or 1). R sends an acknowledgment that is a one-bit sequence number (i.e., the value 0 or 1).

When S sends a message, it resends it repeatedly, with the same sequence number, until it receives an acknowledgment from R that contains the same sequence number. When that happens, S complements (flips) the sequence number (from 0 to 1 or from 1 to 0) and starts transmitting the next message.

When R receives a message with a sequence number, it starts sending the acknowledgment with that sequence number and keeps doing so until it

receives a message with a different sequence number. Then it complements (flips) the sequence number and starts sending the acknowledgment with the new sequence number, etc. This means that for example S may still receive acknowledgments with sequence number 0 when it is already transmitting messages with sequence number 1 (and vice-versa.). When that happens, S ignores these messages and continues transmitting. In the same way R may receive many messages with the same sequence number, so R will consider only the first one and ignore the others and will continue transmitting the acknowledgment until it receives a message with a different sequence number.
The sender and receiver should be initialized with the sequence number 0.

Exercise 2: In order to analyze the ABP protocol for unreliable channels, you need to model the ABP protocol in FSP. Figure 3 gives the structure diagram of the ABP protocol where:

- S receives data from the channel in_msg, sends the data through a (following the ABP protocol), receives an acknowledgment from d (following the ABP protocol) and finally sends a confirmation acknowledgment through the channel out_ack.
- R receives data from b (following the ABP protocol), sends these data through the channel out_msg (avoiding duplications), receives a confirmation acknowledgment from in_ack and finally sends the acknowledgment through c (following the ABP protocol).

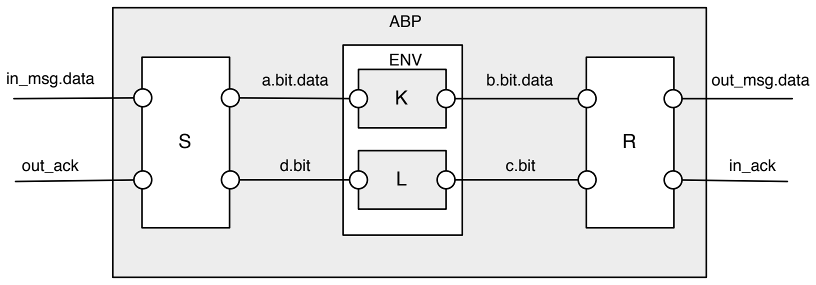

Figure 3

The ABP system has to satisfy the following safety/progress properties:

- The ABP system should behave as the process P in Figure 4, where:
  P = (in_msg[data:D] ->out_msg[data] ->
  in_ack -> out_ack -> P) .

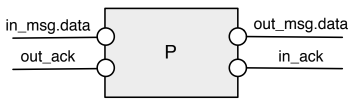

Figure 4

- Consider the unfair condition where R has low priority on the event that sends an acknowledgment through c, the ABP systems should satisfy the progress property that an acknowledgment through c should eventually happen.

Hint: you may use as a flipping function 1-bit

Exercise 3: Provide a Java implementation of S and R (using the provided unreliable channel in Java).

Hint: if convenient, you may use the Select class to select between channels as in Chapter 10 in the book, since the channel class is a subclass of Selectable.

3. **The Multiplexed Buffer:**

Description To allow multiple senders and receivers (in this case we restrict to two senders S1, S2 and two receivers R1, R2 where S1 sends messages with data to R1 and S2 to R2 correspondingly, and R1 sends messages with acknowledgments to S1 and R2 to S2 correspondingly), a multiplexed buffer is used with the ABP system. In this scenario, there are still two channels (channel K for sending messages and channel L for sending acknowledgments) and two processes S and R that implement the ABP protocol but in this case a message will also contain the identity of the sender/receiver in addition to the data and/or one-bit sequence number.

Exercise 4: In order to analyze the Multiplexed buffer system using the ABP protocol, you need to model it in FSP. Figure 5 gives the structure diagram of the full system where:

- The ABP system has been straightforwardly extended with one more piece of information i that is the identity of the sender/receiver (i.e., i can be 1 or 2).
- S1 and S2 receive data from i.input.data, send these data through i.in_msg.data and receive acknowledgments from i.out_ack.
- R1 and R2 receive data from i.out_msg.data, send these data through i.output.data and send acknowledgments through i.in_ack.

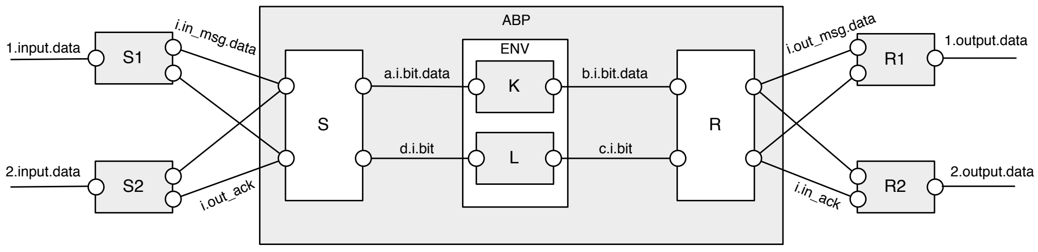

Figure 5

The Multiplexed buffer model has to satisfy the following safety property:

- The Multiplexed buffer model should behave as two one slot buffer processes COPY in parallel as shown in Figure 6, where:
  COPY = (input[data:D] ->
              output[data] -> COPY) .

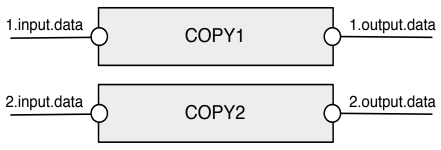

Figure 6

Exercise 5: Provide a Java implementation of your model (as in the previous exercise, use the provided unreliable channel in Java).

Exercise 6 (optional): Consider the unfair condition where S2 has low priority on sending a message trough 2.in_msg.data. Does your model satisfy the progress property that S2 eventually sends a message? Explain. If your system does not satisfy this property, you may as an *optional task implement* an extra process that guarantees fairness between S1 and S2, which means that S2 is guaranteed to eventually send a message.

Deliver to

The assignment should be carried out individually or with one or two other students (Note that groups of more than three students are not allowed) and delivered to the teaching assistant responsible through https://devilry.ifi.uio.no/.
Only one answer is delivered if two or three students work together. The answer must be marked with full name(s) and username(s) of the contributing students.

How to deliver

- You have to implement the FSP part of the exercises using the LTSA Tool (the tool is installed on the lab machines, or you can download it from: http://www.doc.ic.ac.uk/~jnm/book/ltsa/download.html). Your implementation should compile and generate a state machine isomorphic to the one above.
- For the Java part of the exercises you are provided with a project that contains the implementation of the unreliable channel.
  1.- Unzip the file "inf2140assignment3.zip"
  2.- Import the project "inf2140assignment3" (into Eclipse).
  3.- The project runs with the file "Assignment3.java".
  4.-Implement the Java part of the exercises.
  Note: Your final implementation must run when your project is imported into Eclipse, otherwise you must clearly state in the README file the exact commands to run your implementations.
- The delivery must only be in one file: either a .tgz or .zip archive. First you copy the contents of your assignment to a folder with the same name as your username (for groups with two or three students, choose one username of the two or three usernames), then delete all compiled files from the Java project and pack the folder in a .tgz or .zip archive.
- Your archive should contain
  - The source file of your FSP implementations.
  - The source files of your Java implementations (the whole project without the compiled files).
  - A report, named either report.txt or report.pdf , in which you should include the full name(s) and username(s) of the contributing students together with a clear explanation of your FSP and Java implementations.
  - A file named README in which you should describe any peculiarities of your solutions, for instance, things that may be missing, or assumptions made, or questions for the group teacher. If everything runs fine, state it in the README file.

Evaluation: This assignment is graded pass or fail. You must pass this assignment in order to take the final exam. ***Good luck!***