Assignment 2: Modelling with Promela Channels

Exercise 2.1: Model a client server network. Transmitted data should not be visible to clients/servers outside that connection.

The model contains two kinds of processes – clients and servers. The clients create a local channel, and sends it to a server using a global request channel. Using a local channel results in a private connection between the server and client. The following snippet shows the local channel being created and sent to the server:

```
chan replych = [0] of { byte, byte }; //create local channel
request ! replych, _pid;
```

The code is attached in clientserver.pml.

Exercise 2.2: Model a buffered channel by means of rendezvous channels.

The model contains a reader and a writer communicating by means of a buffered channel. The buffered channel is implemented using two rendezvous channels, and two processes handling the buffer. Each pair of process + rendezvous channel handles reading and writing respectively.

The code is attached in buffered.pml.

Implementation

The process handling input continuously receives data on the input channel, waits for space in the buffer and places data in the buffer. The code is shown below.

```
bufferIn ? data;                //wait for data from writer
fillCount < BUFFERSIZE;         //wait for space in buffer
buffer[bufferEnd] = data;       //place data in next empty space
```

The process handling output(reader receiving) waits for data to be available in the buffer (`fillCount > 0`), and attempts to send it.

```
fillCount > 0;                  //wait for data to be buffered
bufferOut ! buffer[bufferStart]; //send oldest data
```

Test

The writer sends data in the form of a byte, and increments the data by 1 continuously, so that the sent data is 0,1,2,…,255,0,1. The client asserts that the received data is incremented by 1 for each data item.

Alternative solution

It is possible to implement a buffered channel using only rendezvous channels and a *single* process handling the buffer. This solution requires extra synchronization, which must be implemented in the reader and writer.