– Lab2 –
Sliding Windows

1 General Instructions

In this lab you will simulate an existing protocol stack, draw diagrams, and make conclusions. The goal is that you should get an understanding for *sliding windows*. The lab can also be seen as a preparation for the next lab where you (among other things) will implement *sliding windows*.

The simulation environment requires JAVA 1.3 and *cygwin* with TinyOS packages. It is possible to use Linux, but that requires installation of `TinyOS`, `AVR cross-compiler`, and `nesc compiler`. Furthermore, several non-trivial modifications of the lab files are required.

You can solve this lab individually or together with another student. If you work in a group of two, both students must be active and be prepared to **motivate** the answers to the lab assistant. **Note that you will not be approved by just showing the simulation results, you should be able to clearly motivate and draw conclusions from your simulation results.**

It is recommended that you work on this lab at the second lab occasion. Read through the entire document *before* you begin with the lab.

Deliverables

- Hand in the lab specification with your answers to the lab assistant (in connection to the oral demonstration).
- Upload a simple text document (just containing your names) to Blackboard to simplify grading.
- The lab should be completed no later than March 4/7, 2014. There will **not** be any extra occasions for you to demonstrate your solutions after this deadline. It will be possible to demonstrate your solution at the next course instance.

Name (student 1)	Name (student 2)
User ID (e.g., abc01234)	User ID (e.g., abc01234)
Date	

2 Cygwin instructions

If you get compiler errors while compiling the simulator, try to install some packages in Cygwin:

1. **Start** -> **All Programs** -> **Cygwin Setup**
2. Run the program as MDH\xxxxx
3. Next
4. Choose **Install from Local Directory**
5. Next
6. Next
7. Next
8. Change package **Devel** from **Default** to **Install**
9. Next
10. Next

3 Introduction

In this lab you will simulate the sliding window protocol. All program code for the protocol stack is available so you do not have to write a single code line in this lab.

By performing simulations, make note of the results and draw conclusions, we hope that you will get a deeper understanding for sliding windows (specifically) and data communication (in general).

Figure 1 shows an overview of the protocol stack. You do not need to understand what all modules do.

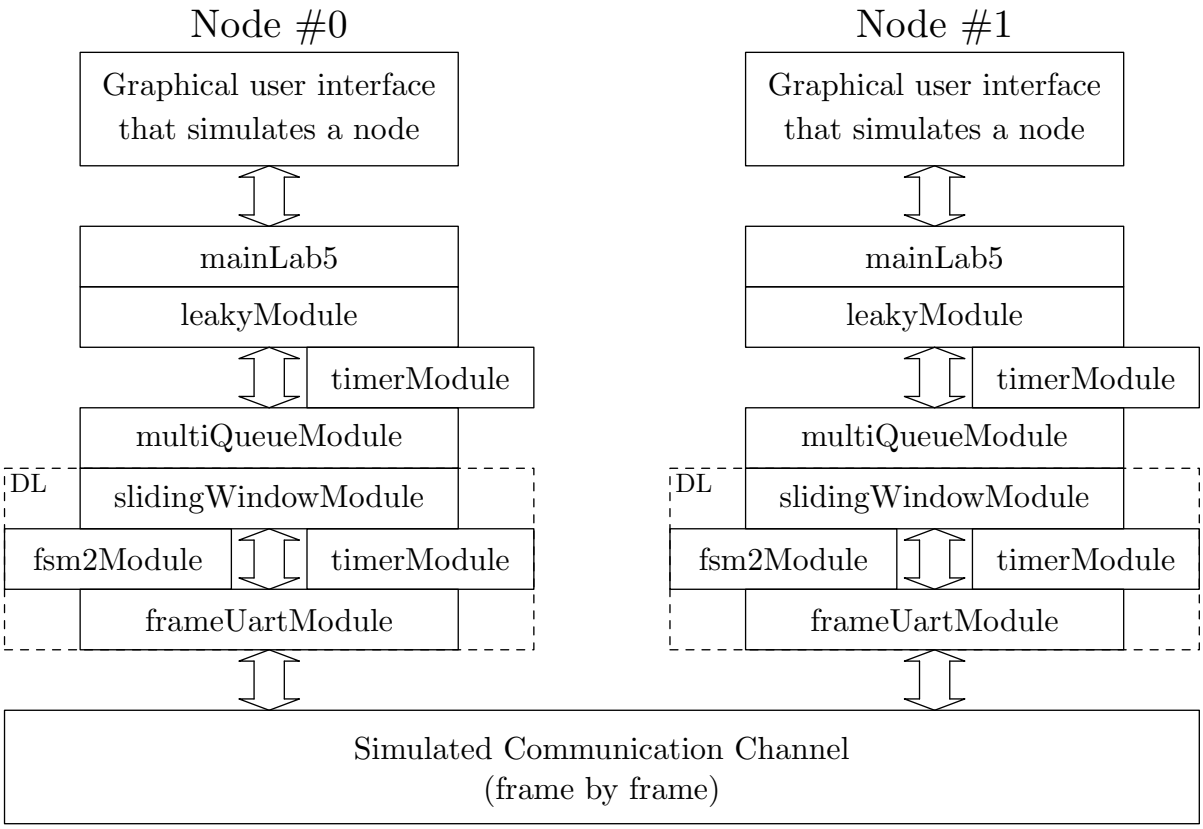

Figure 1: Protocol stack in lab2

The *slidingWindowModule* component is an implementation of the "Go-Back-N" protocol with blockAck.

Furthermore, there is a component named *multiQueueModule*. It is a queue with room for up to 100 frames. This queue is fixed to 100 frames in this lab.

NB: It is very important that you always press the "reset" buttons on *both* nodes when you modify any setting using the scrollbars (for error probability, window size, etc.).

Hint: You can use the arrow keys to fine-tune the values if it is difficult to set the desired value using the scrollbar.

4 Sliding Windows

Extract the lab files (`dllab2.zip`) and change directory to `lab5a`. NB: **lab5a** (this lab comes from an earlier lab series). Compile the program by writing `make pc`, and start the simulator by writing `./begin.bat` in a *cygwin* window.

You should start by measuring the effectiveness for a sliding window. Use the following basic settings: *block size* = 20 bytes, *propagation delay* = 10 ms, *bit speed* = 100 kbps, *error probability* = 0%, and *window size* = 1 (which is the same as a "stop-and-wait" protocol).

Measure the time to send 100 frames from node #0 to node #1 with different window sizes and write down the results in Table 1. Press "send 00-99" to send 100 frames. The time it takes to send 100 frames is shown in the lower left part in the graphical interface for a node. Make sure that all frames has arrived at the other node.

NB: It is important that you press the "reset" button after each time you modify the *window size*. This ensures that both sender and receiver are synchronized with each other.

Window size Error probability=0%, Bit speed=100kbps, Propagation delay=**10ms**	**Time to send 100 frames (ms)**
1	
2	
4	
8	
16	
32	
64	

Table 1:

Next, plot your measured values in the diagram (Figure 2) below.

Also, mark in the diagram in which part that the transfer speed is limited by the window size.

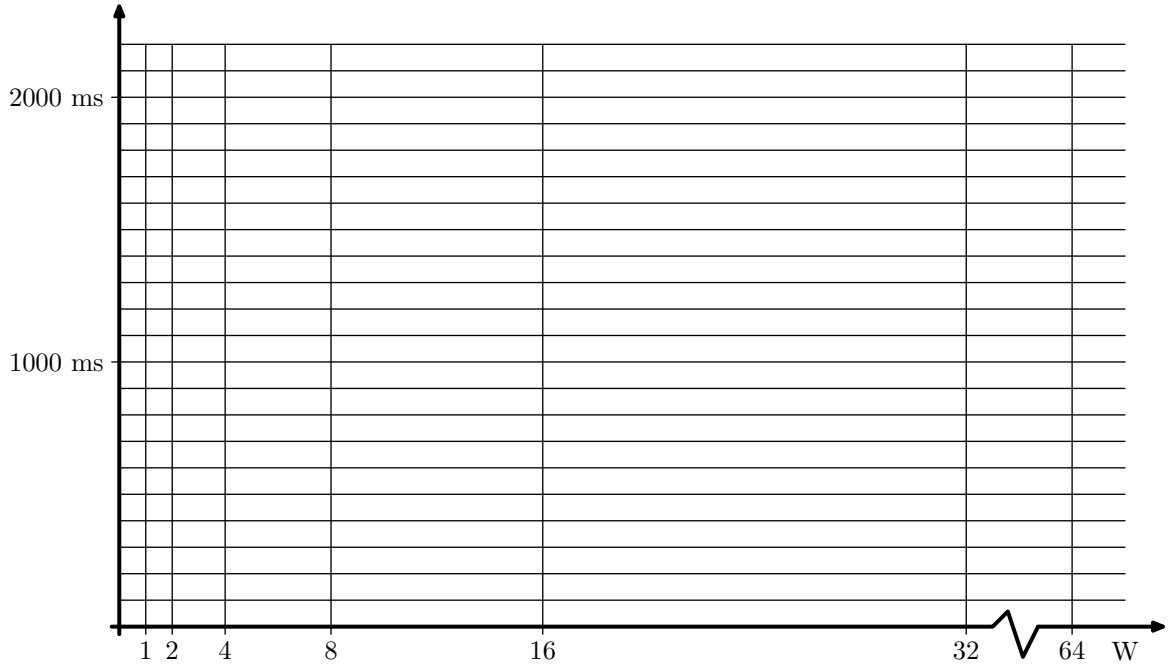

Figure 2: Diagram (Propagation delay=**10ms**)

Are the measured values what we could expect?

It is always suitable to make an estimation about the expected result. We start by looking at a "stop-and-wait" protocol, i.e., when W=1. 100 frames, where each consists of 20 bytes, is 2000 bytes data. The bit speed is 100kbps, which means that each frame takes less than 2ms to send. Note that it will take some time to send each packet (from node to medium), e.g. packet creation, queue-transfer, medium-encoding etc. This time (from node to medium) does **not** include the time it takes for a packet to reach its destination through the medium. This delay (from node to node) is known as the *propagation delay* and it depends on the length and speed of the medium (*bit-speed* and *propagation delay* is illustrated in Figure 3 below). However, we can approximate this time (it takes to send each packet) with zero (when comparing to propagation delay T_P). A frame can be sent in $2 * T_P = 20ms$. It should take approximately $100 * 2 * T_P$ to send all frames, which is 2000 ms. This value should (hopefully) be similar to your measured value.

Let us now look at how fast a transfer can be done, i.e., when the window size is so big that maximum speed can be achieved. 2000 bytes takes little less than 200 ms to send if the bit speed is 100kbps. This value should also (hopefully) be similar to the measured value.

Increase the propagation delay T_P to the double, i.e., 20ms. Redo all measurements with the new setting and write the results in Table 2 below.

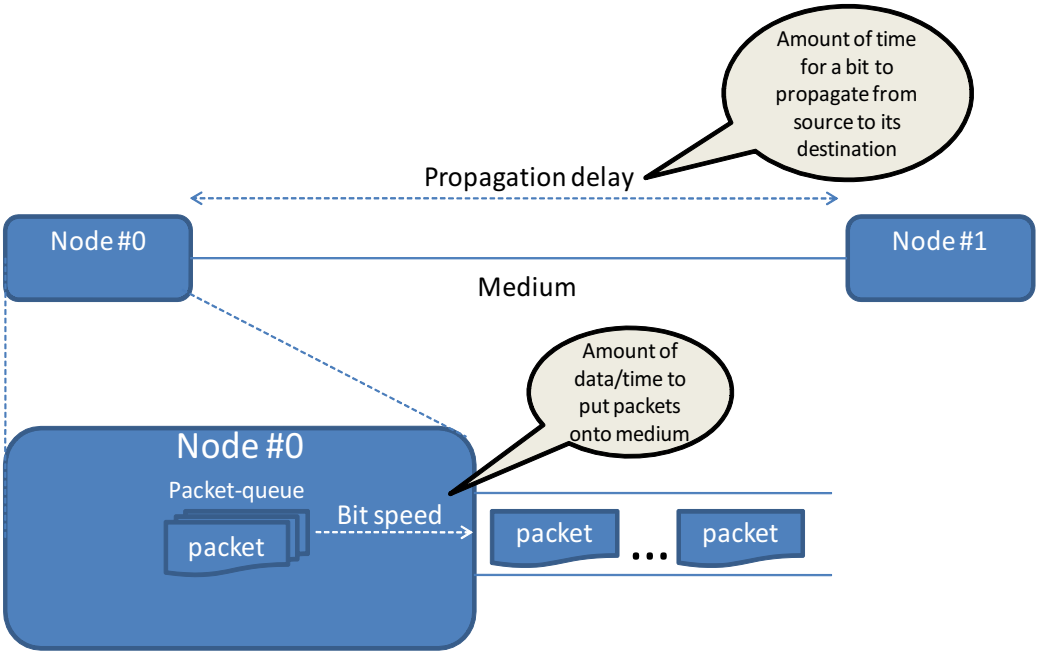

Figure 3: Bit speed and propagation delay

Window size Error probability=0%, Bit speed=100kbps, Propagation delay=**20ms**	**Time to send 100 frames (ms)**
1	
2	
4	
8	
16	
32	
64	

Table 2:

Before you plot the measured values in the diagram below (Figure 4), try to guess how the diagram should look like, and why!

Start by drawing the curve from the previous measurements (where the propagation delay was 10ms) so that you have something to compare the new curve with. Next, add the new values (where the propagation delay was 20ms).

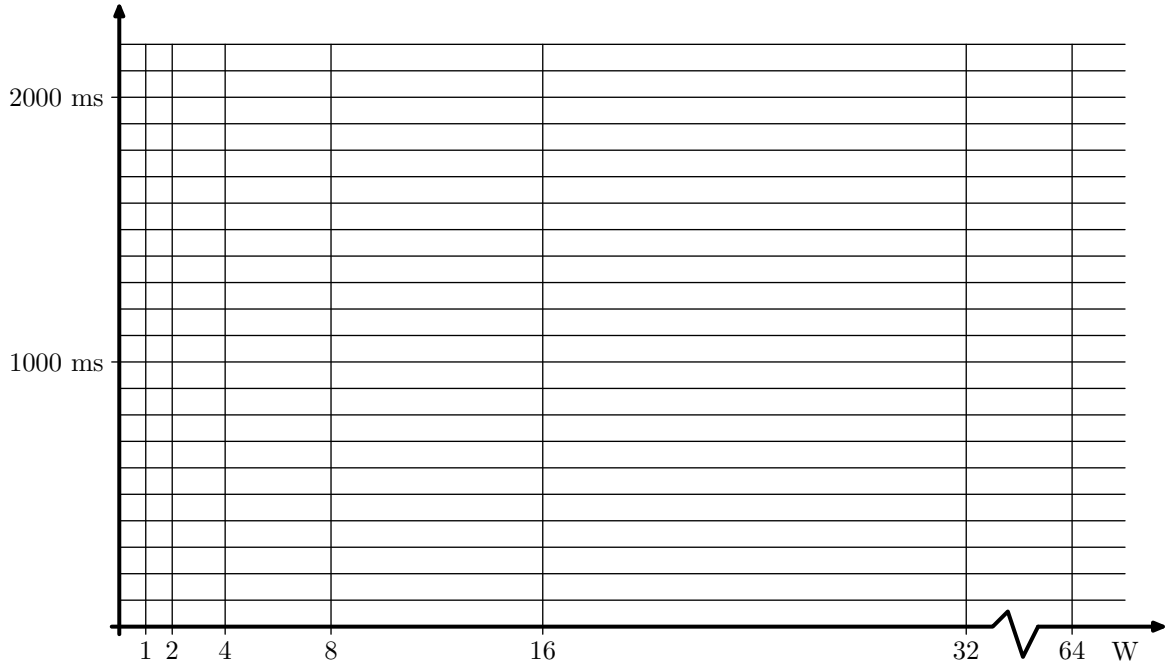

Figure 4: Diagram (Propagation delay=**20ms**)

Explain (in terms of *bit speed*, *propagation delay* and *window size*) the appearance of your plotted curves. Why does the curve suddenly change shape, i.e. why does the transfer time decrease and why does it stabilize suddenly (and not decrease anymore)? Motivate why the curve stabilizes at that particular window size.

Setup the following values: *window size*=16, *bit speed*=100kbps, *propagation delay*=10ms, and *error probability*=0%.

How much can the *propagation delay* be increased before the transfer speed is limited by the window size? It can be difficult to determine the exact point since it a gradually change, but you can use the point where the transfer time has increased by 10%. At some point, the transfer time should increase dramatically, explain why it happens at this point (i.e. at this propagation delay)? The answer should be motivated in terms of *window size*, *bit speed* and *propagation delay*.

How much can the *bit speed* be increased before the transfer speed is limited by the window size? Use a fixed propagation delay at 10ms. The total transfer time will off course be decreased when you increase the bit speed, but the transfer time will, at some point, (almost) not be decreased anymore. Why does the transfer time stabilize at this bit speed? The answer should be motivated in terms of *window size*, *bit speed* and *propagation delay*.

Setup the following values: *window size*=16, *bit speed*=50 kbps, and *error probability*=0%. Measure the time it takes to send 100 frames for different values of T_P (see Table 3 below). Also calculate the measured time in effective transfer speed and plot the values in the diagram (Figure 5).

Propagation delay Error probability=**0%**, Bit speed=50kbps, Window size=16	**Time to send 100 frames** (in milliseconds)	**Effective transfer speed** (2000 bytes data = 16000 bits)
1 ms		
5 ms		
10 ms		
15 ms		
20 ms		
25 ms		
30 ms		
35 ms		
40 ms		

Table 3:

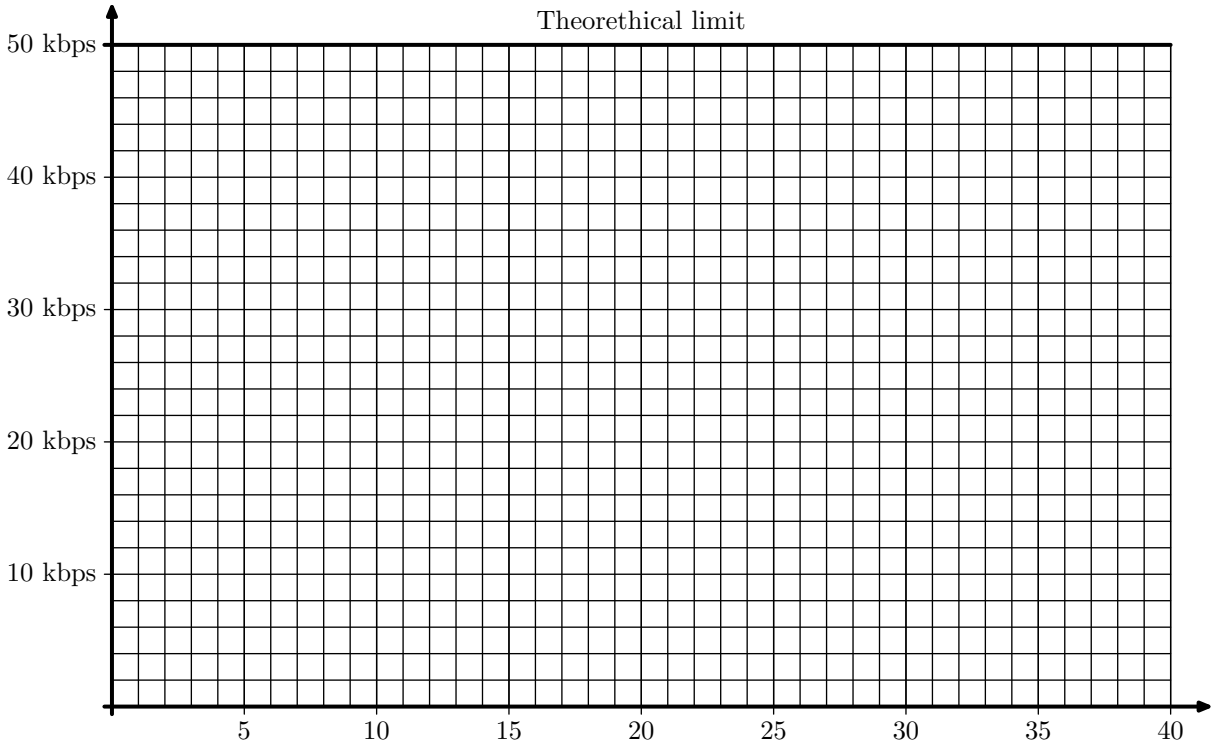

Figure 5: Diagram (Effective transfer speed)

If everything is correct you will clearly see when the window size is insufficient to maintain maximum transfer speed. At approximately which propagation delay does this occur? Motivate your answer in terms of *window size*, *bit speed* and *propagation delay*.