MAKE-UP ASSIGNMENT

Subject : Queue implementation with linked-list representation
Due Date : 12.09.2019 — 23:59:59

Pipelining Protocol Simulator

1 Introduction

Reliable data transfer and pipelining protocols

The Internet network layer provides only best effort service with no guarantee that packets arrive at their destination. Also, since each packet is routed individually it is possible that packets are received out of order. For connection-oriented service provided by TCP, it is necessary to have a reliable data transfer (RDT) protocol to ensure delivery of all packets and to enable the receiver to deliver the packets in order to its application layer.

There has been several approaches developed so far to ensure the RDT. They are: RDT 1.0, RDT 2.0, RDT 2.1, RDT 2.2, and RDT 3.0. At each development stage, some modifications are included. Since these approaches are out of the scope, they are not explained here in detail. These approaches are also known as *stop-and-wait* protocols. The Stop-and-wait RDT protocols have poor performance in a long-distance connection. The sender can only transmit one packet per round-trip time, or RTT. To improve transmission rates, a realistic RDT protocol must use pipelining. This allows the sender to have a large number of packets "in the pipeline". This phrase refers to packets that have been sent but whose receipt has not yet verified by the receiver.

There are two basic forms of pipelining protocols: *Go-Back-N* and *Selective Repeat*. They both allow multiple "in-flight" yet to be acknowledged packets. Selective Repeat differs Go-Back-N in some aspects such as timer, acknowledge (ACK), and the like. The main advantage of Selective repeat over Go-Back-N is that it avoids unnecessary retransmissions by having the sender retransmit only those packets dropped.

Since Go-Back-N is not within the context of this assignment, only selective repeat pipelining approach will be briefly explained here. For more details, you might refer to the slide notes by Jim Kurose. [1].

In this assignment, you are expected to implement a C program that simulates Selective Repeat pipelining protocol using **queues**.

[1]http://www-net.cs.umass.edu/kurose-ross-ppt-6e **(Chapter 3: The Transport Layer)**

2 Selective Repeat Approach

- N unACKed packets are allowed to take place in the pipeline, also known as `WINDOW`. Here, N is determined by the maximum window size, or WS.

  In Figure 1, only 5 packets are allowed to take place in the `WINDOW`. Initially, the packets 0 to 4 are sent in a successive manner and the following packets (5, 6, 7, 8, and so on) wait until they find room in the `WINDOW`.

- Receiver individually ACKs all correctly received packets.

  For every packet received, a packet containing ACK message is sent from the receiver to the sender. From the demonstration provided in Fig. 1, it is seen that receiver ACKs the packet 0 at $t = 5$, which is arrived to the sender at $t = 10$ in Fig. 1.

- Each packet leaves the `WINDOW`, thus makes a room, if and only if there is no packet in front of it waiting to be ACKed by the receiver.

  The packets 0 to 4 are removed from the `WINDOW` at time 10 to 14, respectively (see Fig. 1). However, the packets 4 to 7 have to wait the packet 3 being ACKed even if they are transferred (see Fig. 2).

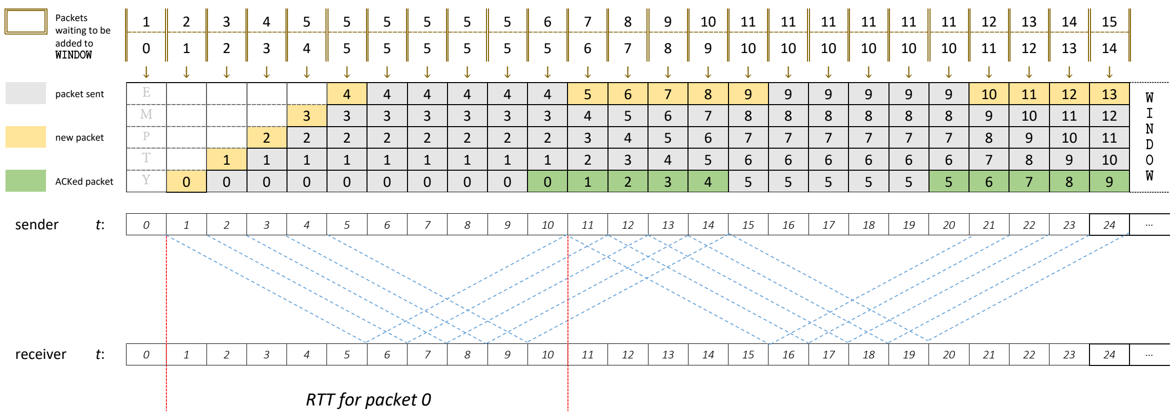

Figure 1: A demonstration of selective repeat approach

- Sender maintains a timer for each unACKed packet in the `WINDOW`.

  Particularly due to the congestion that occurs in the network, not every packet (data packet or ACK packet) is able to reach to other side. To detect a drop-case, sender waits ACK packet just after it sends the data packet. In Fig. 2, a drop-case is demonstrated. Here, sender detects packet 3 dropped at $t = 18$.

- Sender resends packets for which ACK not received.

  After a drop-case detected by the sender, it resents the data packet in question and restarts the timer for that packet to detect a potential drop. In Fig. 2, sender resends packet 3 at $t = 19$, just after drop-case is detected.

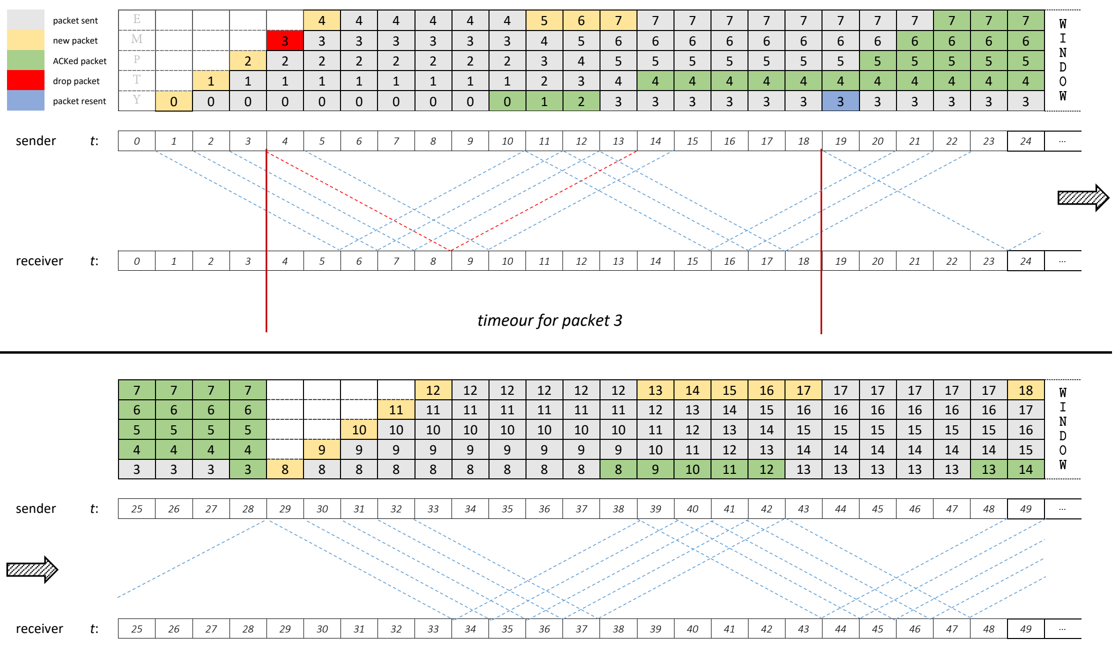

Figure 2: A drop-case demonstration in selective repeat approach

To gain insight on how selective repeat works, you can make use of an online simulator[2].

3 Experiment

Using queues, you are to develop a C program that simulates the selective repeat pipelining protocol, which is briefly introduced in the previous section. Within the context of this assignment, some features of selective repeat approach are discarded to keep the model as simple as possible. First, there are two `WINDOW`s in this approach, one is for the sender, the other is for the receiver. You will discard the `WINDOW` that receiver maintains. Second, you do not need to calculate transmission delay (d_{trans}) which is the amount of time required to push all data packet's bit into the wire. In this experiment, take d_{trans} as 1 sec.

Considering these assumptions/simplifications, you should design your implementation (simulator) such that it takes necessary parameters then simply runs selective repeat protocol. As you all could see (if not, I suggest you to read the previous section from scratch), the transfer process of the data to be sent through selective repeat approach depends on some parameters and their values, which will be provided as command line arguments:

- Window Size (WS): It is the number of packet allowed to take place in `WINDOW`. It must be dynamic and no upper bound must be defined for WS.
- Data Size (DS): It is the amount of the data (in Byte) that will be sent from sender. There must not be an upper bound for DS.

[2]http://www.ccs-labs.org/teaching/rn/animations/gbn_sr/

- Packet Size (PS): It is the amount of the data size (in Byte) that can be sent through a single packet and header size. Within the context of this assignment, you can discard the header size of a packet. So, PS refers to the amount of data enclosed in a single packet.
- Round Trip Time (RTT): It is the amount of time (in Sec.) expected to send a packet and to receive its ACK.
- Timeout (TOUT): It is the amount of time (in Sec.) required to resend a packet dropped. It should be much greater than RTT.
- Packet Loss (DROPPCK): The packet IDs that will be dropped during the simulation will be provided in a separate file whose path and name are dynamic. While designing your implementation, take the scenarios in consideration where **100 packets** (no matter it is data or ACK) might drop at most and one particular packet might drop **more than one times**.

The order of the arguments should be:

<WS><DS><PS><TOUT><RTT><DROPPCK>

Your implementation/simulator should both print out a **log** during the transmission of the whole data and display **transfer report** at the end of the simulation. The **log** should keep these information: Simulation time, event, and `WINDOW` status. The **simulation report**, on the other hand, should contain simple statistical results as well as simulation parameters. The display format of the log and the report will be explained in the following subsections.

3.1 Log content

The smallest unit metric of the time will be *second* and no smaller unit values will be considered such as millisecond, microsecond, nanosecond, and the like. Thus, at every seconds during the simulation, your implementation/simulator should print out the simulation time, events, and window state.

SIMULATION TIME:
Simulation time should be displayed as (HH:MM:SS). The output format of simulation time should be as follows.

```
<SIMULATION TIME> (00:05:09)
```

EVENT:
After simulation time is displayed, your implementation should print out the events that could possibly occur at every time. The events should be displayed line by line after <EVENTS> caption. There might happen three events at most at a particular time:

i) Receive an ACK packet from the receiver.

```
- Data packet (id:0) has now been ACK'ed by receiver!
```

ii) Send a new packet

```
- A new data packet (id:5) has now been sent!
```

iii) Packet-drop detection

```
- It is timeout for data packet (id:3), so it has been resent now!
```

At a particular time; none, one, or more than one events might happen. You should print out the appropriate event(s). The last line of events should always inform the user about the data size (in Byte) transferred so far in the following format:

```
- Data size sent so far is 4902.00 Byte
```

WINDOW STATE:
The last information that should be displayed at every second is the `WINDOW` state as below:

```
<WINDOW STATE>: 86 -> 10 | 85 -> 10 | 84 -> 10 | 83 -> 10 | 82 -> 9
```

where the numbers (82 to 86) at the left side of "–>" indicate the packet ids that should be displayed in order of the position at the `WINDOW`; whereas those at the right side (9 and 10) indicate the time elapsed for these packets. Note every packet inside the `WINDOW` is separated by the (*pipe*) | symbol. You should print out "empty" when there is no data packet waiting to be sent in the `WINDOW`.

To show the displaying format in detail, you can see an example below containing a log recorded at a particular simulation time:

Note: It is actually the log content that should be displayed at $t = 29$ *in a simulation environment demonstrated in Fig. 2.*

```
<SIMULATION TIME> (00:00:29)
<EVENTS>
- Data packet (id:3) has now been ACK'ed by receiver!
- A new data packet (id:8) has now been sent!
- Data size sent so far is 400.00 Byte
<WINDOW STATE>: 8 -> 1
```

3.2 Transfer report content

Once the whole data is sent, simulator should be terminated and the transfer report should be displayed just after the log. Transfer report should consist of two type information: Parameter setting and simulation results. An example regarding the transfer report is provided below to show the report formatting.

```
**************************************************
*                 TRANSFER REPORT                *
**************************************************
Parameter Setting:
--------------------------------------------------
Window Size                         :          05
Timeout                             :    0015 Sec.
RTT                                 :    0010 Sec.
Data Size                           : 5002.00 Byte
Packet Size                         :   50.00 Byte
--------------------------------------------------
Results:
--------------------------------------------------
Number of packet to send the data    :        0101
Number of packet dropped             :        0011
Average time to send a single packet :  11.634 Sec.
**************************************************
```

Although it is quite clear, you might get trouble on calculating the average time to send a single packet, though. It should obviously be equal to RTT unless a packet loss happens. However, this average time value increases with an increase in number of packet loss.

Note1: You should display the log and the report on the console, ***not in a file.***

Note2: You will be given two simulation results – one is at the end of this paper; the other will be provided with a separate file named `output` *– so that you can check your implementation if it works as expected.*

Notes

- Do not miss the deadline.
- Save all your work **until the assignment is graded**.
- The assignment must be **original**, **individual work**. Duplicate or very similar assignments are both going to be considered as cheating.
- Regardless of the length, use **understandable** names to your variables, functions, and etc.
- Should you have a question, feel free to ask on Piazza. Be aware that the question has not been asked/discussed before.
- Do not submit your works via e-mail.
- You will submit your work from https://submit.cs.hacettepe.edu.tr/index.php with the file hierarchy as below:

This file hierarchy must be zipped[3] before submission (Not .rar, only .zip files are supported by the system)

```
→ <student id>.zip
    → Source Code
         | *.c
         | *.h
         | Makefile[4]
```

[3]do not include any folder in your zip file, but use the files themselves to zip.
[4]should also include **clean** command — name executable file as **SRS**

Simulation Results:

```
<SIMULATION TIME> (00:00:00)
<EVENTS>
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: empty

<SIMULATION TIME> (00:00:01)
<EVENTS>
- A new data packet (id:0) has now been sent!
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 0 -> 1

<SIMULATION TIME> (00:00:02)
<EVENTS>
- A new data packet (id:1) has now been sent!
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 1 -> 1 | 0 -> 2

<SIMULATION TIME> (00:00:03)
<EVENTS>
- A new data packet (id:2) has now been sent!
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 2 -> 1 | 1 -> 2 | 0 -> 3

<SIMULATION TIME> (00:00:04)
<EVENTS>
- A new data packet (id:3) has now been sent!
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 3 -> 1 | 2 -> 2 | 1 -> 3 | 0 -> 4

<SIMULATION TIME> (00:00:05)
<EVENTS>
- A new data packet (id:4) has now been sent!
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 4 -> 1 | 3 -> 2 | 2 -> 3 | 1 -> 4 | 0 -> 5

<SIMULATION TIME> (00:00:06)
<EVENTS>
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 4 -> 2 | 3 -> 3 | 2 -> 4 | 1 -> 5 | 0 -> 6

<SIMULATION TIME> (00:00:07)
<EVENTS>
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 4 -> 3 | 3 -> 4 | 2 -> 5 | 1 -> 6 | 0 -> 7

<SIMULATION TIME> (00:00:08)
<EVENTS>
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 4 -> 4 | 3 -> 5 | 2 -> 6 | 1 -> 7 | 0 -> 8

<SIMULATION TIME> (00:00:09)
<EVENTS>
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 4 -> 5 | 3 -> 6 | 2 -> 7 | 1 -> 8 | 0 -> 9

<SIMULATION TIME> (00:00:10)
<EVENTS>
- Data size sent so far is 0.00 Byte
<WINDOW STATE>: 4 -> 6 | 3 -> 7 | 2 -> 8 | 1 -> 9 | 0 -> 10

<SIMULATION TIME> (00:00:11)
<EVENTS>
- Data packet (id:0) has now been ACK'ed by receiver!
- A new data packet (id:5) has now been sent!
- Data size sent so far is 10.00 Byte
<WINDOW STATE>: 5 -> 1 | 4 -> 7 | 3 -> 8 | 2 -> 9 | 1 -> 10

<SIMULATION TIME> (00:00:12)
<EVENTS>
- Data packet (id:1) has now been ACK'ed by receiver!
- A new data packet (id:6) has now been sent!
- Data size sent so far is 20.00 Byte
<WINDOW STATE>: 6 -> 1 | 5 -> 2 | 4 -> 8 | 3 -> 9 | 2 -> 10

<SIMULATION TIME> (00:00:13)
<EVENTS>
- Data packet (id:2) has now been ACK'ed by receiver!
- A new data packet (id:7) has now been sent!
- Data size sent so far is 30.00 Byte
<WINDOW STATE>: 7 -> 1 | 6 -> 2 | 5 -> 3 | 4 -> 9 | 3 -> 10

<SIMULATION TIME> (00:00:14)
<EVENTS>
- Data packet (id:3) has now been ACK'ed by receiver!
- A new data packet (id:8) has now been sent!
- Data size sent so far is 40.00 Byte
<WINDOW STATE>: 8 -> 1 | 7 -> 2 | 6 -> 3 | 5 -> 4 | 4 -> 10

<SIMULATION TIME> (00:00:15)
<EVENTS>
- Data packet (id:4) has now been ACK'ed by receiver!
- A new data packet (id:9) has now been sent!
- Data size sent so far is 50.00 Byte
<WINDOW STATE>: 9 -> 1 | 8 -> 2 | 7 -> 3 | 6 -> 4 | 5 -> 5

<SIMULATION TIME> (00:00:16)
<EVENTS>
- Data size sent so far is 50.00 Byte
<WINDOW STATE>: 9 -> 2 | 8 -> 3 | 7 -> 4 | 6 -> 5 | 5 -> 6

<SIMULATION TIME> (00:00:17)
<EVENTS>
- Data size sent so far is 50.00 Byte
<WINDOW STATE>: 9 -> 3 | 8 -> 4 | 7 -> 5 | 6 -> 6 | 5 -> 7

<SIMULATION TIME> (00:00:18)
<EVENTS>
- Data size sent so far is 50.00 Byte
<WINDOW STATE>: 9 -> 4 | 8 -> 5 | 7 -> 6 | 6 -> 7 | 5 -> 8

<SIMULATION TIME> (00:00:19)
<EVENTS>
- Data size sent so far is 50.00 Byte
<WINDOW STATE>: 9 -> 5 | 8 -> 6 | 7 -> 7 | 6 -> 8 | 5 -> 9

<SIMULATION TIME> (00:00:20)
<EVENTS>
- Data size sent so far is 50.00 Byte
<WINDOW STATE>: 9 -> 6 | 8 -> 7 | 7 -> 8 | 6 -> 9 | 5 -> 10

<SIMULATION TIME> (00:00:21)
<EVENTS>
- Data size sent so far is 50.00 Byte
<WINDOW STATE>: 9 -> 7 | 8 -> 8 | 7 -> 9 | 6 -> 10 | 5 -> 11

<SIMULATION TIME> (00:00:22)
<EVENTS>
- Data packet (id:6) has now been ACK'ed by receiver!
- Data size sent so far is 60.00 Byte
<WINDOW STATE>: 9 -> 8 | 8 -> 9 | 7 -> 10 | 6 -> 10 | 5 -> 12

<SIMULATION TIME> (00:00:23)
<EVENTS>
- Data packet (id:7) has now been ACK'ed by receiver!
- Data size sent so far is 70.00 Byte
<WINDOW STATE>: 9 -> 9 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 13

<SIMULATION TIME> (00:00:24)
<EVENTS>
- Data packet (id:8) has now been ACK'ed by receiver!
- Data size sent so far is 80.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 14

<SIMULATION TIME> (00:00:25)
<EVENTS>
- Data packet (id:9) has now been ACK'ed by receiver!
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 15

<SIMULATION TIME> (00:00:26)
<EVENTS>
- It is timeout for data packet (id:5), so it has been resent now!
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 1

<SIMULATION TIME> (00:00:27)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 2

<SIMULATION TIME> (00:00:28)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 3

<SIMULATION TIME> (00:00:29)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 4

<SIMULATION TIME> (00:00:30)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 5
```

```
<SIMULATION TIME> (00:00:31)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 6

<SIMULATION TIME> (00:00:32)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 7

<SIMULATION TIME> (00:00:33)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 8

<SIMULATION TIME> (00:00:34)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 9

<SIMULATION TIME> (00:00:35)
<EVENTS>
- Data size sent so far is 90.00 Byte
<WINDOW STATE>: 9 -> 10 | 8 -> 10 | 7 -> 10 | 6 -> 10 | 5 -> 10

<SIMULATION TIME> (00:00:36)
<EVENTS>
- Data packet (id:5) has now been ACK'ed by receiver!
- A new data packet (id:10) has now been sent!
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 1

<SIMULATION TIME> (00:00:37)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 2

<SIMULATION TIME> (00:00:38)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 3

<SIMULATION TIME> (00:00:39)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 4

<SIMULATION TIME> (00:00:40)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 5

<SIMULATION TIME> (00:00:41)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 6

<SIMULATION TIME> (00:00:42)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 7

<SIMULATION TIME> (00:00:43)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 8

<SIMULATION TIME> (00:00:44)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 9

<SIMULATION TIME> (00:00:45)
<EVENTS>
- Data size sent so far is 100.00 Byte
<WINDOW STATE>: 10 -> 10

<SIMULATION TIME> (00:00:46)
<EVENTS>
- Data packet (id:10) has now been ACK'ed by receiver!
- Data size sent so far is 102.00 Byte
<WINDOW STATE>: empty
```

```
***************************************************
*                 TRANSFER REPORT                 *
***************************************************
Parameter Setting:
---------------------------------------------------
Window Size                         :          05
Timeout                             :    0015 Sec.
RTT                                 :    0010 Sec.
Data Size                           :  102.00 Byte
Packet Size                         :   10.00 Byte
---------------------------------------------------
Results:
---------------------------------------------------
Number of packet to send the data    :        0011
Number of packet dropped             :        0001
Average time to send a single packet :  11.364 Sec.
***************************************************
```