Technische Universität Braunschweig

Institut für Betriebssysteme und Rechnerverbund

Kommunikation und Multimedia

Prof. Dr. L. Wolf

Praktikum Computernetze im WS0910

- Task Description -

Betreuer: Felix Büsching, Tobias Pögel

Aufgabe: Network Simulator Programming-ARQ Protocols

1. Objective

The purpose of this task/assignment is to code/implement the ARQ protocols: Stop-and-Wait, Go-Back-N, and Selective Repeat. Students are required to code/implement any two of these protocols. They have to implement these protocols inside an existing simulator. This simulator provides the basic infrastructure to share messages between a source node and a sink node.

We have eliminated those functions from the simulator that generate messages and perform necessary actions before the transmission of the messages and after their reception. Students are required to re-write these functions according to the above mentioned ARQ protocols, run simulations with different input values and compare their results with the theoretical performance of the protocols.

2. Features supported/not-supported by the simulator

The simulator provided in this assignment supports the necessary features to share messages between two nodes: a source and a sink. Following is an overview of these features:

- Reading the simulation settings from a file. This operation is performed in response to "***load filename***" command.
- Verification of simulation settings. This process is performed as a first step in response to "***run***" command so that there shouldn't be any complications in the later operations.
- Message sharing. This is the next step in response to "***run***" command. Simulator looks for any messages ready to transmit at the source or the sink and transfer them to the appropriate destination. It also invokes the relevant functions in these nodes so that they can take necessary actions such as message generation, response generation etc.
- The simulator also provides a central clock mechanism which keeps track of simulation time and events. One tick of this clock is equivalent to one nanosecond; therefore, all time calculations in the simulator are performed in nanoseconds.

3. Simulator Structure

The ARQP simulator is designed solely in C++ under Linux environment. It contains classes for different components to simulate ARQ functionality.

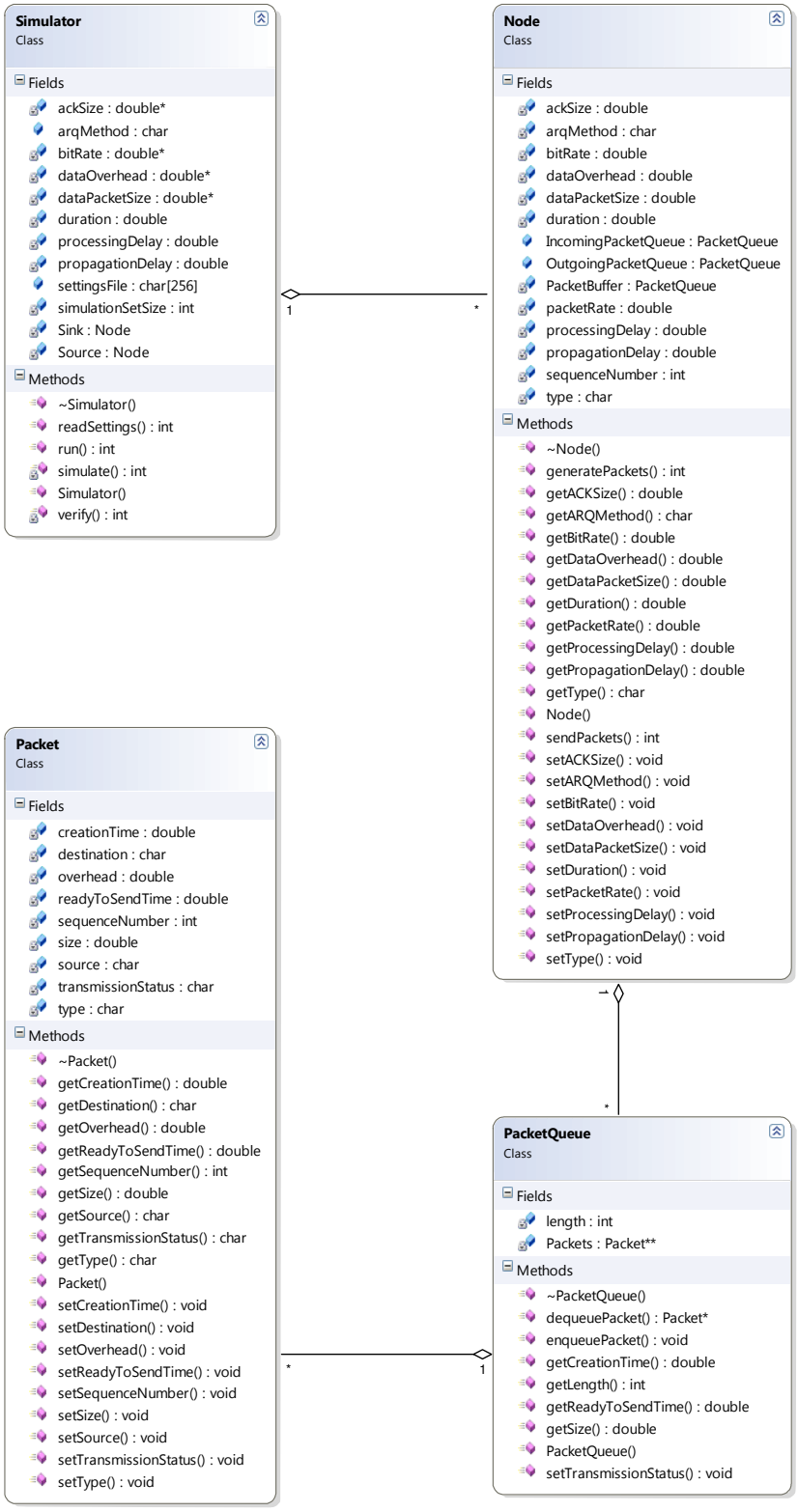

Fig 1: Simulator Structure: class diagram

These classes are described in different header files (.h) provided in the simulator source code. The class diagram (Fig. 1) shows how these classes are interrelated in the simulator. All these classes are briefly described here while a detailed description of these classes is independently given in the document "PCn-ARQP-class-description.pdf":

main (main.h, main.cpp)
These two files contain the ***main*** function and some other global functions and variables. None of these are really important in this assignment. In the ***main*** function, an instance of ***Simulator*** is created and a command prompt is displayed to accept input from the user. The relevant functions of ***Simulator*** are called to fulfill user commands. A complete list of available commands is provided in "PCn-ARQP-class-description.pdf".

Simulator (Simulator.{h/cpp})
This class represents the simulator instance. A simulator instance contains two ***Node*** instances: ***Source*** and ***Sink***. These two represent the source and the sink node.

Node (Node.{h/cpp})
This class represents a node either ***Source*** or ***Sink***. It contains all necessary components of a node to implement an ARQ protocol.

Packet (Packet.{h/cpp})
This class describes a message or packet that can be shared between the source and the sink.

PacketQueue (Packet.{h/cpp})
The purpose of this class is to provide a packet queue structure and related functionality wherever there is a need to store one or more messages or packets. For example, both ***Source*** and ***Sink*** contain three ***PacketQueue*** instances: ***PacketBuffer***, ***OutgoingPacketQueue***, and ***IncomingPacketQueue***. The ***IncomingPacketQueue*** contains packets sent by other nodes, the ***OutgoingPacketQueue*** contains packets being sent by this node (in transmission), while the ***PacketBuffer*** contains packet generated by this node but are not yet sent. Whenever, a node generates a message/packet, it is first placed in ***PacketBuffer***. Then, after ***processing delay*** it is moved to the ***OutgoingPacketQueue***. Simulator iteratively looks for the messages in the ***OutgoingPacketQueue*** of the two nodes and transfers them to the ***IncomingPacketQueue*** of the destination node after ***propagation delay + transmission time***.

Time (Time.{h/cpp})
This is a utility class which provides necessary features to keep track of simulation time and execute different events accordingly. The simulation clock tick is one nanosecond; hence, all time measurements are in terms of nanoseconds.

4. How it works

This section will describe the sequence of operations inside the simulator to provide an understanding about its working. A sequence diagram about the main operations of the simulator is also given below to enhance the understanding.

The simulator executable is named ***arqp.o*** and when invoked it displays a command prompt as following:

```
ARQ Protocols Simulator Command Line>>
```

An instance of ***Simulator*** named ***ARQ_PS*** is created to perform further operations. Inside ***ARQ_PS***, two instances of ***Node*** named ***Source*** and ***Sink*** are created as well. A complete list of available simulator commands and their output is given in the document "PCn-ARQP-class-description.pdf". The two commands "***load***" and "***run***" are important as these provide

the core functionality of this simulator. The sequence of operations performed by simulator in response to these two commands is as follows:

load filename

This command directs the simulator to load the simulation settings from the file "***filenam***e". In response, the name of this file is assigned to the ***settingsFile*** member of the ***Simulator*** instance and ***readSettings*** method is invoked. This method reads the settings from the file and set the relevant ***Simulator*** and ***Node*** members.

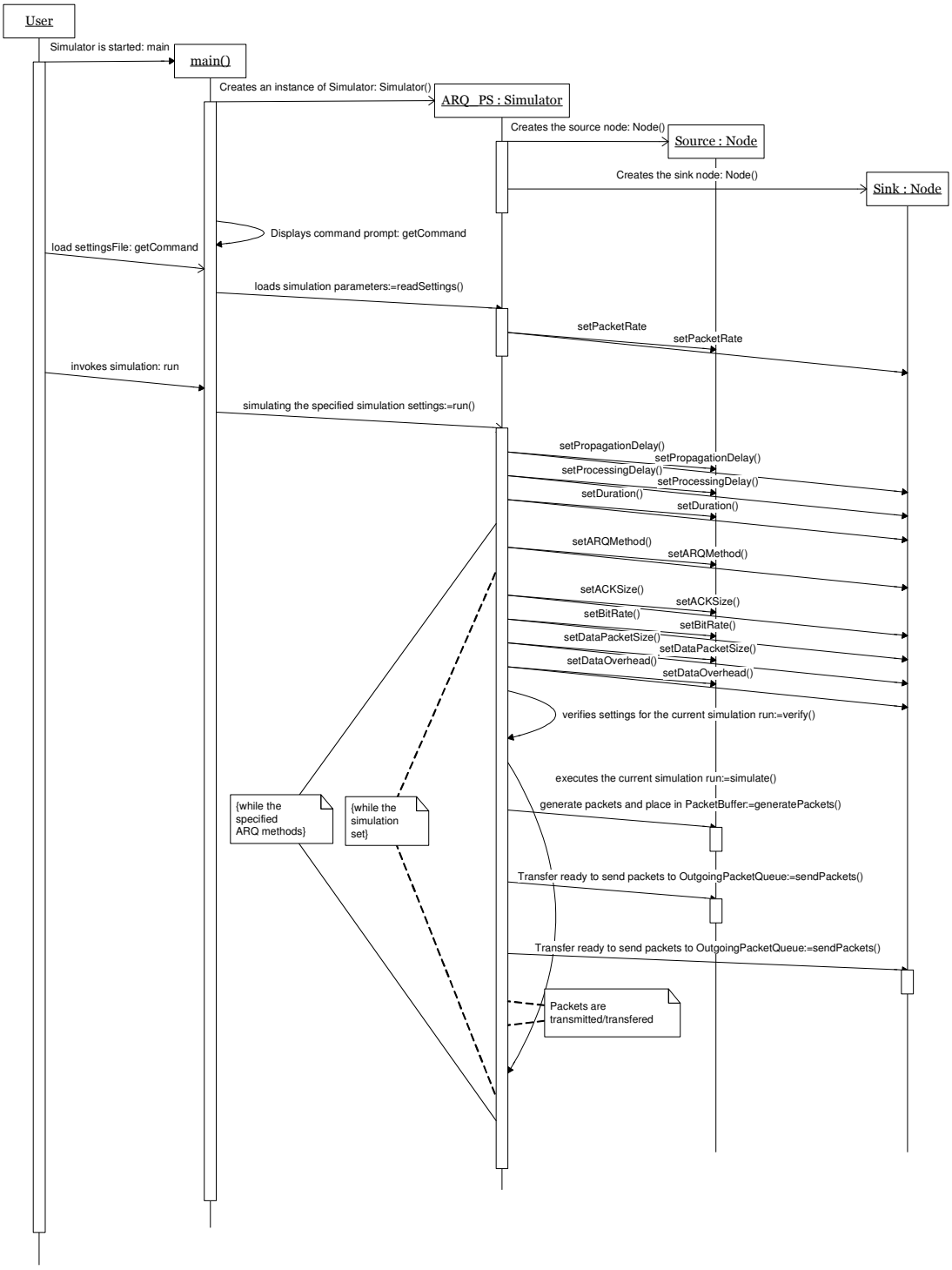

Fig 2: Sequence diagram showing sequence of operations inside "load" and "run" command

run
This command invokes the sequence of operations to simulate ARQ protocol behavior as per the settings loaded previously.

First, the ***verify*** method of ***Simulator*** instance is called to verify any ambiguities in the settings. It mainly verifies that none of the simulation parameters should be zero or less.

Then, the ***simulate*** method, which simulates the ARQ events, is invoked. In this method, first the simulation clock is started by calling ***Time::start()***. Then, a loop is started which continues until the simulation duration is over. Inside this loop, the ***generatePackets*** method of ***Source***, the ***sendPackets*** method of ***Source***, and the ***sendPackets*** method of ***Sink*** are called in sequence. Afterwards, the simulator checks the ***OutgoingPacketQueue*** of Source and Sink, and start transmitting packets if there are any.

5. Your task

The simulator provided for this assignment is fully functional. However, we have eliminated the code from ***generatePackets*** and ***sendPackets*** methods of ***Node*** class. Students have to implement these methods according to the specifications of the ARQ protocols. As mentioned earlier, students are required to implement any two of the three ARQ protocols. These two implemented protocols should be submitted independently. During this implementation, students also have to take care for the following rules:

- Following header files are provided only:

main.h, Simulator.h, Node.h, Time.h, Packet.h

The respective ***cpp*** files containing the implementation are provided in compiled ***(.o)*** form. However, ***Node.o*** does not have the implementation of ***generatePackets*** and ***sendPackets***. The blank definitions of these functions will be provided in an independent file ***arqp.cpp***.

- Students have to write the code for these two methods in the ***arqp.cpp*** file. They can add any functions or data members to any of the classes if required.
- The purpose of ***generatePackets*** method is to generate packets and place these packets in ***PacketBuffer***. This method is invoked for ***Source*** only. Hence, in this method, students have to write the code to generate data packets as per the data rate mentioned in the settings file.
- The packets in ***PacketBuffer*** are required to be moved to ***OutgoingPacketQueue*** after the processing delay and set the ***transmissionStatus*** flag. This job should be performed in ***sendPackets*** (or in ***generatePackets*** as per student's approach). However, while shifting these packets they have to take care of the underlying ARQ protocol. Furthermore, as this method works for both ***Source*** and ***Sink*** (***generatePackets*** is also common but it is not invoked for ***Sink***), they have to differentiate the functionality of ***Source*** and ***Sink***. Once a packet is in ***OutgoingPacketQueue***, simulator will start transmitting it to the destination and they need not to do any further operations on it.
- Students have to perform packet log operations as per their implementation logic, simulator does not provide any log of packet generation, reception, or transmission etc.
- For compiling the simulator code with their additions, a script file ***compile-arqp*** is also provided. They have to simply run the command ***./compile-arqp*** from OS/X command prompt. This will generate an executable with name ***arqp***.
- Once the code is complete and working, they have to run simulations with different ***data packet size (n_f)***, ***data overhead size (n_o)***, and ***bit rate (R)*** values given in the table below:

Table 1: Simulation Parameters				
Test	***Data Overhead Size (n_o)***	***Packet Size (n_f)***	***Bit Rate (R)***	***ACK Size (n_a)***
A	26 Bytes	Use any 5 different values from 72 Bytes to 1526 Bytes	Use any 5 different values from 1Mpbs to 1Gbps	14 Bytes
B	34 Bytes	Use any 5 different values from 34 Bytes to 2346 Bytes	Use any 5 different values from 1Mpbs to 1Gbps	14 Bytes
C	Use any 5 different values between 20 Bytes and 60 Bytes	For every value of $\boldsymbol{n_o}$, use any 5 different values between $\boldsymbol{n_o}$ Bytes and 65535 Bytes	Use any 5 different values from 1Mpbs to 1Gbps	40 Bytes
D	Use any 5 different values between 20 Bytes and 60 Bytes	For every value of $\boldsymbol{n_o}$, use any 5 different values between $\boldsymbol{n_o}$ Bytes and 65535 Bytes	Use any 5 different values from 1Mpbs to 1Gbps	40 Bytes

- The propagation delay ($\boldsymbol{t_{prop}}$) and the processing delay ($\boldsymbol{t_{proc}}$) both are fixed for these simulations and their value is one microsecond each.
- The features such as transmission error rate, channel error, timeout and channel busy are not considered/implemented in this assignment. This implies that every packet reaches the destination successfully and their can be more than one packets in transit.
- The above mentioned parameter values are provided to the simulator through simulation settings file. The format of this file is a given in the document "PCn-ARQP-class-description.pdf", while a sample settings file is provided with the code. Multiple values for these simulation parameters can be provide in one single file and the simulator performs all the simulations at a time. Students have to take care that how they can have independent log for each simulation run.
- The value of simulation duration and packet rate should be selected in such a way that sufficient number of messages should be exchanged between the source and the sink.
- Students have to submit their version of ***arqp.cpp*** and any other changed header files.
- Furthermore, they have to present their simulation results in the form of graphs and compare their results with the efficiency equations given in Table 5.2 (of Communication Networks-Fundamentals, Concepts and Key Architecture by Alberto Leon-Garcia and Indra Widjaja).
- Warning: Please perform the memory handling operations carefully.

6. Reading Material helpful in this assignment

- The main reading source for this assignment is the section 5.2 of Communication Networks-Fundamentals, Concepts and Key Architecture by Alberto Leon-Garcia and Indra Widjaja.