Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики»

(СибГУТИ)

Е.В. Кокорева

МОДЕЛИРОВАНИЕ ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ В СЕТЕВОМ СИМУЛЯТОРЕ OMNET++

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Новосибирск 2025

УДК 004.94

Утверждено редакционно-издательским советом СибГУТИ

Рецензент: доцент Моренкова О.И.

Кокорева Е.В. Моделирование телекоммуникационных систем в сетевом симуляторе OMNeT++: учебно-методическое пособие / Сибирский государственный университет телекоммуникаций и информатики; кафедра систем мобильной связи. – Новосибирск, 2025. – 32 с.

Учебно-методическое пособие предназначено для студентов магистратуры направлений 11.04.01 «Радиотехника», 11.04.02 «Инфокоммуникационные технологии и системы связи». Пособие содержит три лабораторные работы, к каждой из которых приведено краткое теоретическое описание и задание для практического выполнения. Для выполнения лабораторных работ применяется свободно распространяемая интегрированная среда моделирования OMNeT++ и язык программирования C++.

Содержание

Выполнение и оформление лабораторных работ

Варианты задания приведены в таблице последнего раздела описания каждой лабораторной работы. Ячейки таблицы содержат номер варианта (всего 80 вариантов), а заголовки строки и столбца – исходные данные к симуляции.

Номер варианта выбирается по двум последним цифрам пароля. Если число, образованное двумя последними цифрами пароля, превышает 80, то в качестве номера варианта принимается сумма этих цифр.

После выполнения каждой лабораторной работы преподаватель должен получить от студента на проверку архив, содержащий:

1. Отчёт в виде текстового документа (.doc, .docx или .pdf).
2. Проект OmNET++ .

Оформление отчёта к лабораторной работе.

Отчёт по выполнению лабораторной работы должен быть оформлен в соответствии с ГОСТ 7.32-2017, ГОСТ Р 2.105-2019 и содержать:

1. Титульный лист.
2. Содержание (с нумерацией страниц).
3. Задание в соответствии с вариантом.
4. Описание выполнения всех пунктов задания со скриншотами, программными кодами и другими необходимыми для понимания результатов компонентами.

Примечание 1: в содержательной части документа допустимо наличие иллюстраций, таблиц, формул и пр.

Примечание 2: основные структурные элементы отчёта приведены в описании к каждой лабораторной работе.

5. Вывод (описание полученных в ходе выполнения знаний, освоенных умений и навыков).
6. Список литературы (по ГОСТ Р 7.0.100-2018).

Рисунки (графики, схемы, диаграммы и пр.), таблицы, формулы и другие объекты должны быть пронумерованы и подписаны в соответствии с ГОСТ Р 2.105-2019.

Общие сведения о симуляторе OMNeT++

Симулятор OMNeT++ можно скачать по адресу:

https://omnetpp.org/

Там же находится документация по пакету, а также различные плагины, фреймворки и примеры моделей.

На сегодняшний день сайт недоступен для российских пользователей, поэтому либо получаем доступ к нему через VPN, либо скачиваем ПО по моей ссылке с Яндекс диска: https://disk.yandex.ru/d/60rXnoAgqomyKg (версия 4.5 достаточная для выполнения лабораторных работ) или https://disk.yandex.ru/d/TRhNVCv1y2Q7AQ (версия 6.1.0 – последняя на текущий момент).

OMNeT++ имеет гибкую модульную архитектуру на основе компонентных библиотек C++ и различных фреймворков и служит для моделирования (симуляции) поведения различных телекоммуникационных систем, включая проводные (Ethernet, TCP/IP, MPLS и пр.) и беспроводные сети (ad hoc, mesh, сети мобильной связи 3GPP и пр.).

Симулятор работает на платформах: Windows, Linux, Mac OS X и других Unix-подобных системах.

Компонентами OMNeT++ являются:

- библиотека ядра симуляции;
- язык описания топологии NED;
- интегрированная среда разработки OMNeT++ IDE;
- графический интерфейс пользователя GUI (англ. *Graphical User Interface*) для запуска симуляции (Tkenv);
- командный интерфейс пользователя для запуска симуляции (Cmdenv);
- утилиты;
- документация и примеры.

Интегрированная среда разработки IDE (англ. *Integrated Development Environment*) OMNeT++, основанная на Eclipse, предоставляет графическую оболочку для запуска и различные расширения, например, симуляция в реальном времени, подключение баз данных, интеграция с SystemC и некоторые другие.

Внешний вид OMNeT++ IDE представлен на рисунке 1.

Модель представляет собой иерархию простых (англ. *Simple Module*) и составных (англ. *Compound Module*) модулей и соединений (англ. *Connections*) между ними (рисунок 2).

Для описания структуры модели используется язык NED (см. лаб. работу №6), а для описания поведения модели функции из библиотеки компонентов

языка C++ фреймворков (лаб. работа №2). Топология сети сохраняется в файле с расширением `.ned`, а алгоритм симуляции – в файлах `.cc` и `.h`.

Пример ned-файла:

```
simple ChainNode extends Node
{
    parameters:
        @display("i=misc/square_vs");
    gates:
        inout left;
        inout right;
}

network Chain
{
    parameters:
        int n @prompt("Number of modules") = default(5);
    submodules:
        node[n]: ChainNode;
    connections allowunconnected:
        for i=0..n-2 {
            node[i].right <--> node[i+1].left;
        }
}
```

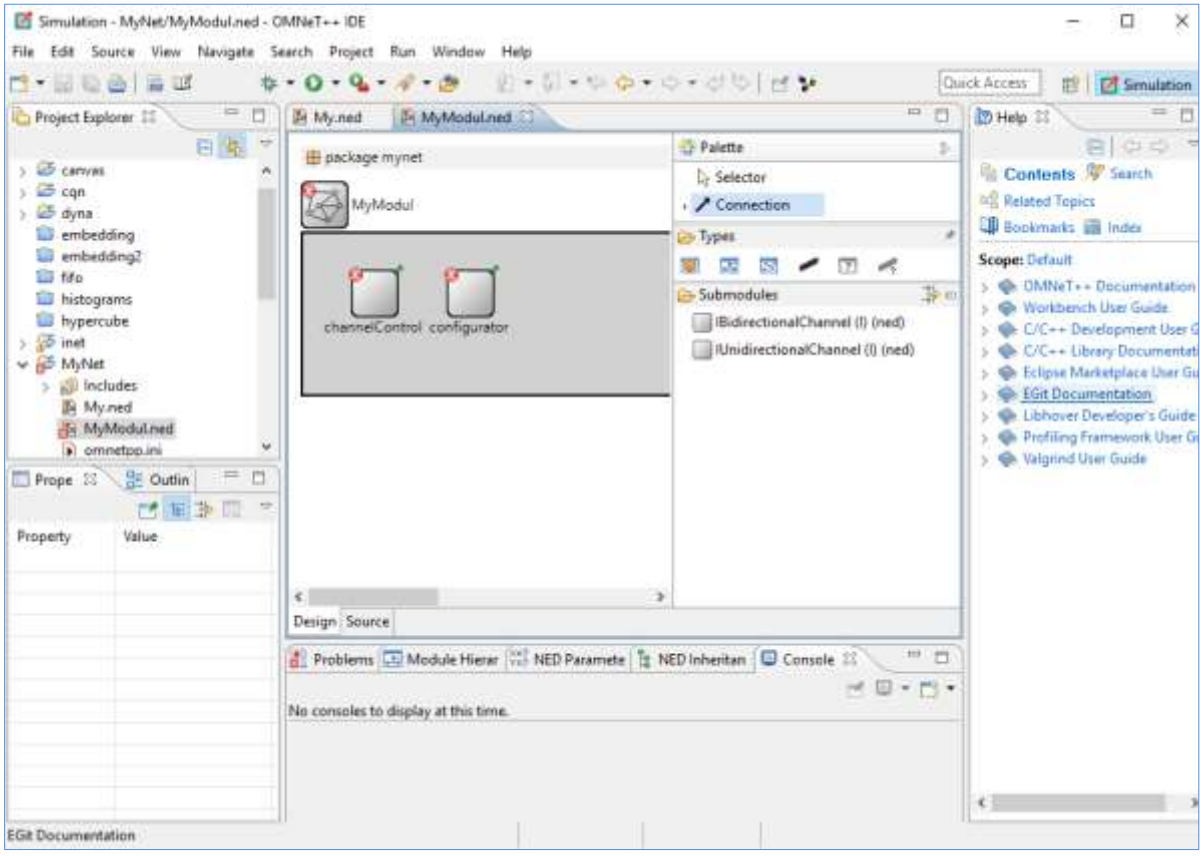

Рисунок 1 – Интегрированная среда разработки OMNeT++

Конфигурация модели (время симуляции, значения параметров и свойств и пр.) задаётся файлом `omnetpp.ini`, который OMNeT++ создаёт автоматически при первом запуске симуляции. Разработчик может создавать и редактировать этот файл самостоятельно.

Пример файла конфигурации:

```
[General]

[Config Grid]
network = Grid
Mesh.height = 4
Mesh.width = 6

[Config TriangleGrid]
network = TriangleGrid
TriangleGrid.rows = 12
TriangleGrid.cols = 5

[Config HexGrid]
network = HexGrid

[Config Chain]
network = Chain
Chain.n = 6

[Config Star]
network = Star
Star.n = 15
```

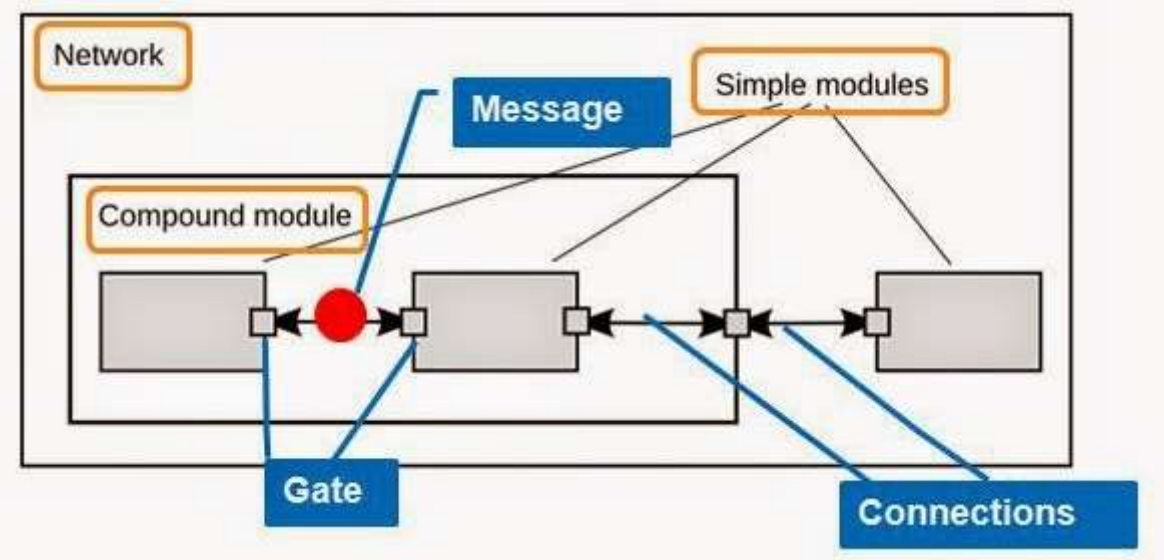

Рисунок 2 – Иерархия модулей OMNeT++

Новый проект в OMNeT++ IDE создаётся с помощью команд меню **File => New => Project OMNeT++**.

В открывшемся окне (рисунок 3) задаём имя проекта и если нужно папку, содержащую проект и поддержку C++, нажимаем кнопку **Next**.

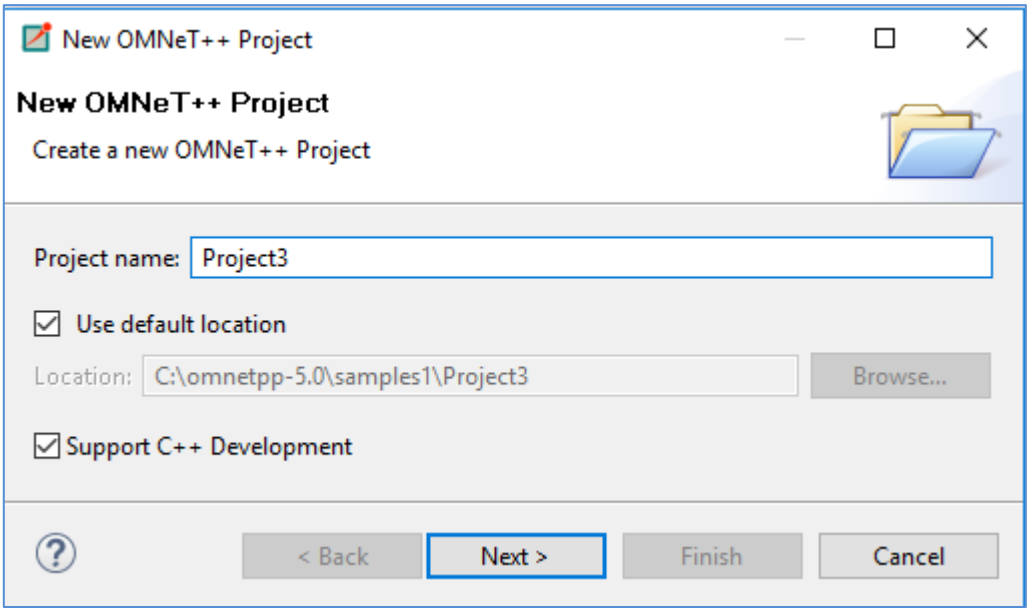

Рисунок 3 – Создание проекта OMNeT++

В следующем окне выбираем **Empty Project** и нажимаем кнопку **Finish**.

Лабораторная работа №1

Построение сети с примитивной топологией в симуляторе OMNeT++

Цель работы: Овладеть основными принципами моделирования в среде OMNeT++. Научиться работать с NED-редактором в графическом и текстовом режиме для создания топологии телекоммуникационной сети. Освоить компиляцию и запуск симуляции

Теоретические сведения:

1.1 Общие сведения о редакторе NED

NED (англ. *Network EDitor*) используется для описания структуры модели (топологии сети).

Редактор NED работает в двух режимах:

- графическом (англ. *Design*);
- текстовом (англ. *Source*).

Пример графического режима приведён на рисунке 4, овалом обведены вкладки переключения режимов.

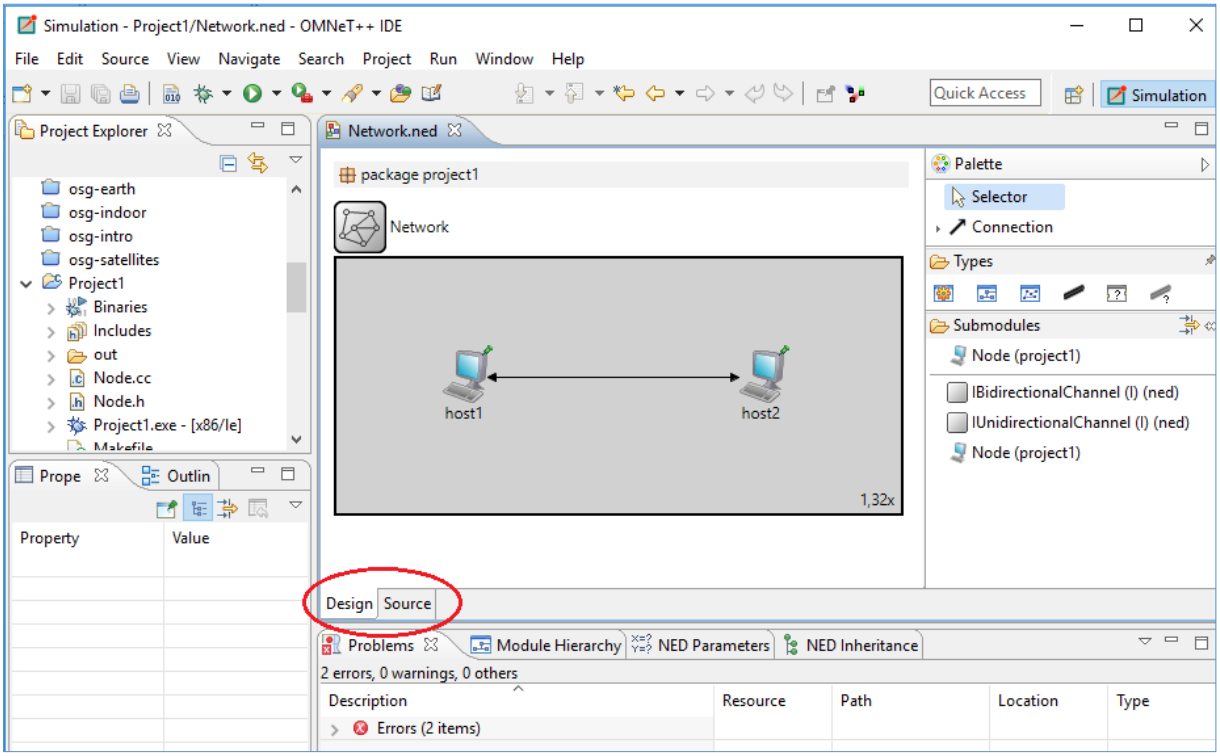

Рисунок 4 – Графический режим редактора NED

Справа отображена панель инструментов для добавления модулей и соединений в рабочую область (англ. *workspace*) модели.

1.2 Пример построения сети из двух узлов

Рассмотрим построение топологии сети, состоящей из двух узлов, соединённых простой двунаправленной линией без параметров (*Connection*).

Создадим новый пустой проект (англ. *Empty Project*) с названием MyNet.

1.2.1 Создание простого модуля *Node*

Определим в этом проекте простой модуль – узел (англ. *Node*). Для этого применим команды меню: **File => New => Simple Module**. В окне (рисунок 5) зададим имя NED-файла, нажмём кнопку **Next**.

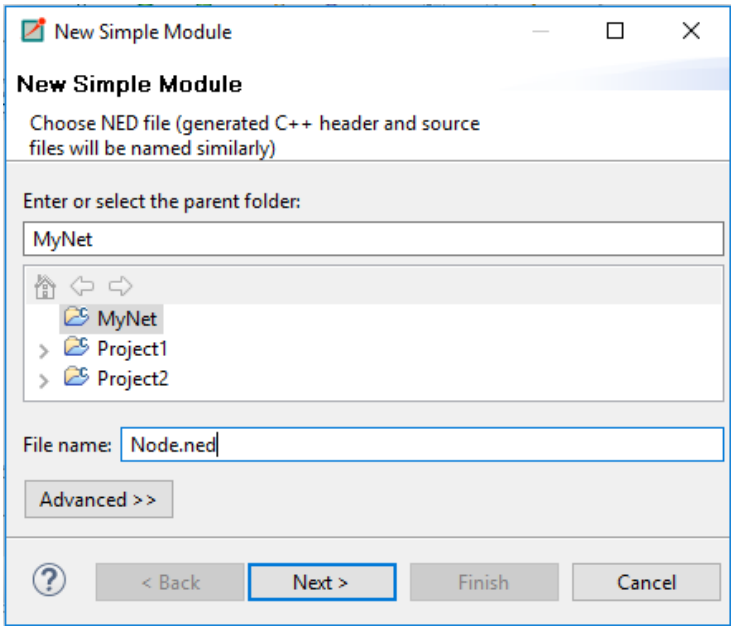

Рисунок 5 – Создание узла

В следующем окне выберем **The simple module** и нажмём кнопку **Finish**.

Результат в графическом режиме представлен на рисунке 6.

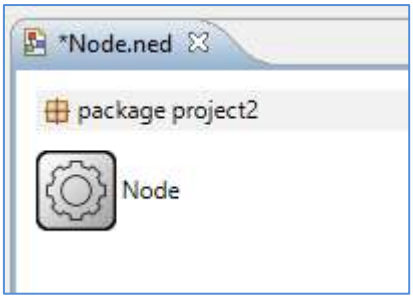

Рисунок 6 – Узел

Можно сменить иконку узла, чтобы она соответствовала его содержанию. Для этого необходимо щёлкнуть правой кнопкой мыши на значке узла и выбрать в контекстном меню пункт **Properties** (свойства) см. рисунок 7.

Перейдя в окне **Properties of Node** на вкладку **Appearance**, получаем доступ к разделу **Image** (рисунок 8) и выбираем нужную иконку из всплывающего окна **Select Image**.

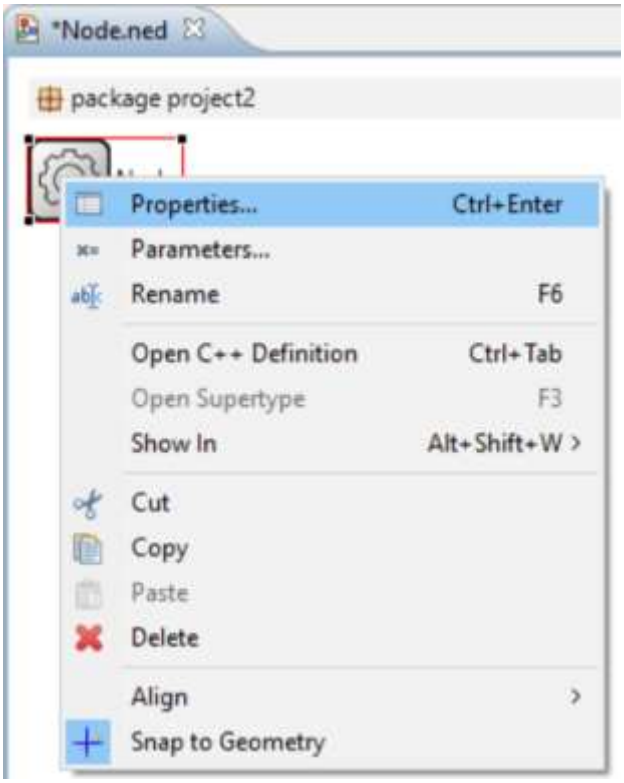

Рисунок 7 – Свойства объекта

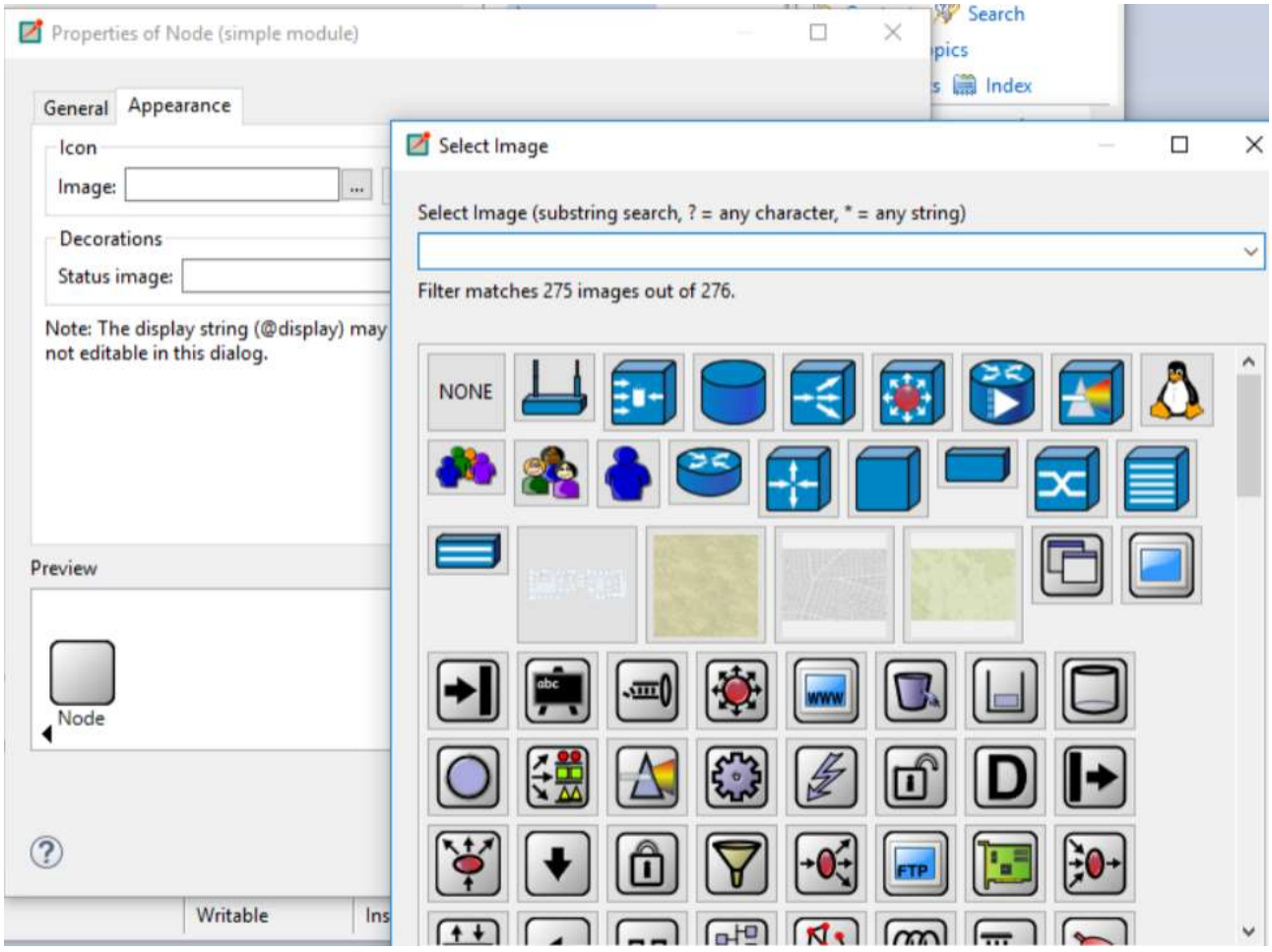

Рисунок 8 – Окно **Select Image**

Выберем из предложенных иконок изображение персонального компьютера (device/pc)  , которое далее будем использовать в примерах.

Перейдя на вкладку **Source** редактора NED можно увидеть автоматически созданный файл простого модуля (Simple module) Node.ned.

```
package mynet

simple Node
{
@display("i=device/pc);
}
```

Строка `@display("i=device/pc);` представляет собой метаданные и описывает выбранную иконку для графического обозначения узла Node.

Для того чтобы можно было соединить узлы в сеть им необходимы порты ввода-вывода. В OMNeT++ они названы *шлюзами* (англ. *gates*). Добавим к приведённому выше коду раздел `gates`:

```
package mynet;

simple Node
{
@display("i=device/pc);

gates:
     input in;
     output out;
}
```

Или, если мы ходим использовать двунаправленные линии, код описания будет следующим:

```
…
gates:
     inout port;
…
```

Причём, если узлов сети будет больше двух, и они будут иметь разное количество соединений друг с другом, необходимо определить вектор портов (шлюзов) неопределённой размерности.

```
…
gates:
     inout port[];
…
```

Автоматически сгенерированный C++ код созданного узла Node сохраняется в файле Node.cc и выглядит следующим образом:

```
#include "Node.h"

Define_Module(Node);
```

```
void Node::initialize()
{
    // TODO - Generated method body
}

void Node::handleMessage(cMessage *msg)
{
    // TODO - Generated method body
}
```

1.2.2 Построение сети

File => New => Network. В окне (рисунок 9) зададим имя NED-файла (Network.ned), нажмём кнопку **Next**.

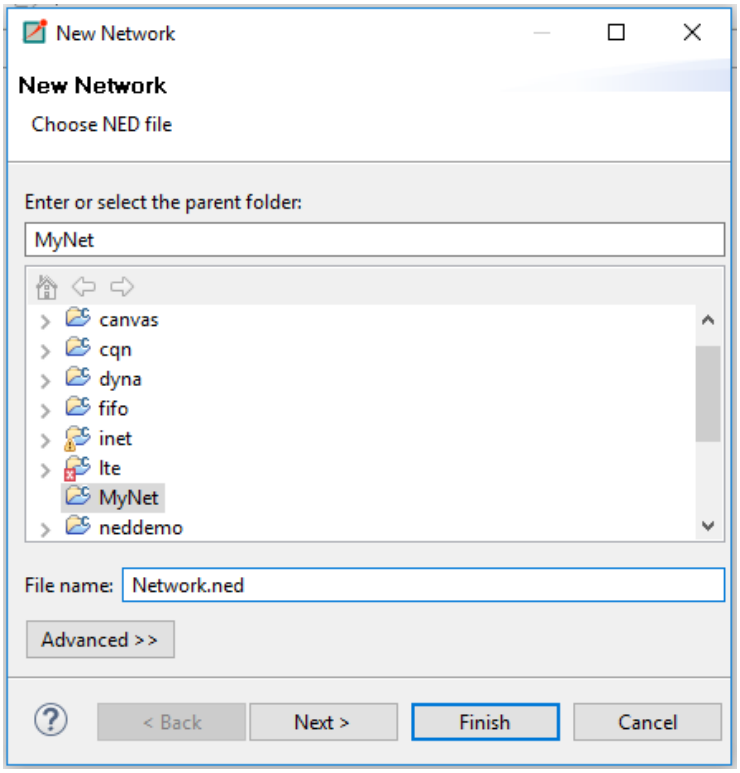

Рисунок 9 – Создание сети

В следующем окне выберем **An empty network** и нажмём кнопку **Finish**. В графической среде редактора NED пустая сеть выглядит так, как показано на рисунке 10.

Затем из панели инструментов (**Palette**) справа от рабочей области добавляем в область Network два узла Node. Для этого необходимо щёлкнуть левой кнопкой мыши на обозначении узла в **Palette**, а затем в том месте workspace, где будет располагаться узел (рисунок 11). Затем увеличить рабочую область сети (почему-то она уменьшается с добавлением узла до размеров узла) и добавить второй узел. Можно изменить название узла, выделив его щелчком левой кнопки мыши.

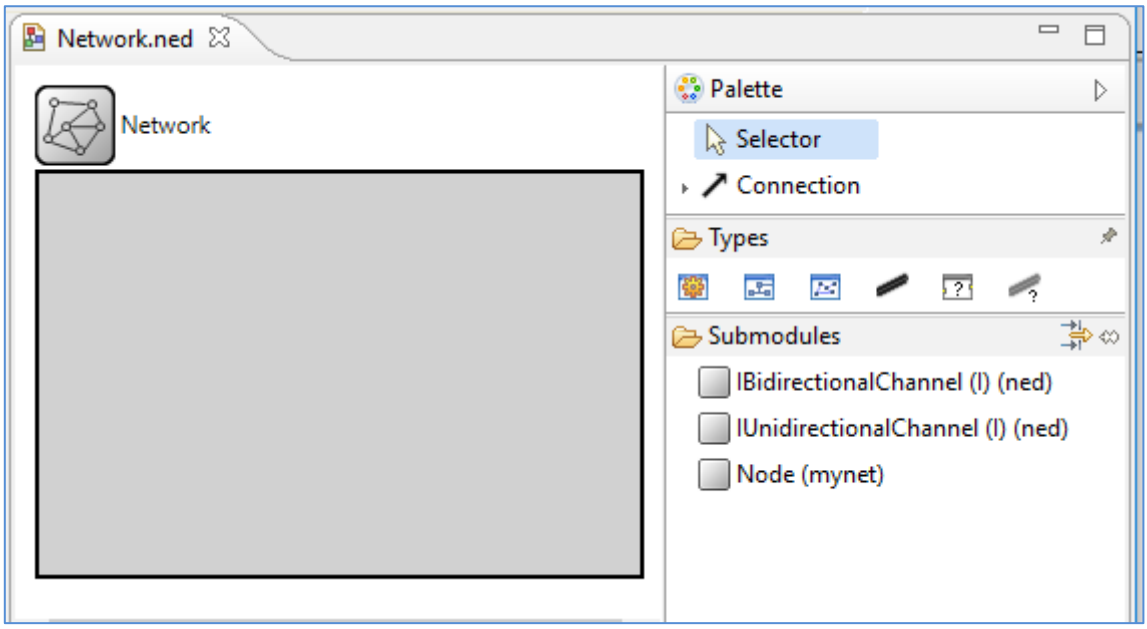

Рисунок 10 – Сеть Network в режиме Design

Таким же образом можно добавлять необходимое количество узлов к сети Network.

В примере узлы обозначены host1 и host2.

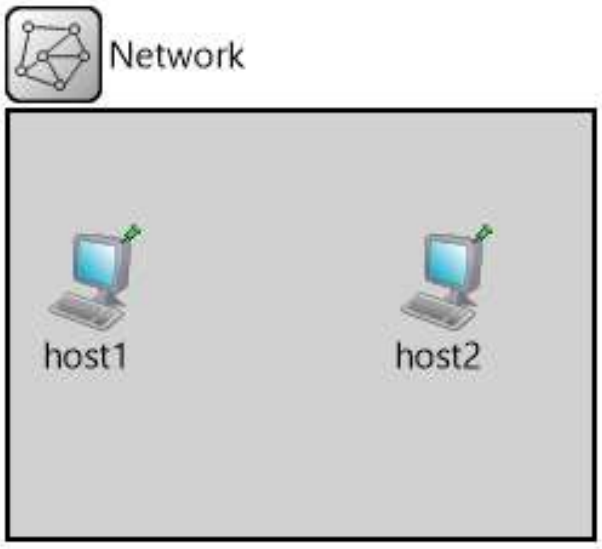

Рисунок 11 – Добавление узлов к сети

Продолжение создания сети заключается в соединении узлов линией. OMNeT++ содержит три предустановленных типа соединений

- идеальный канал (англ. IdealChannel);
- канал с задержкой (англ. DelayChannel);
- канал с заданной скоростью (англ. DatarateChannel).

Выбрать тип позволит щелчок левой кнопкой мыши на стрелке вниз слева от палитры соединений (Connection), чтобы увидеть выпадающее меню (рисунок 12).

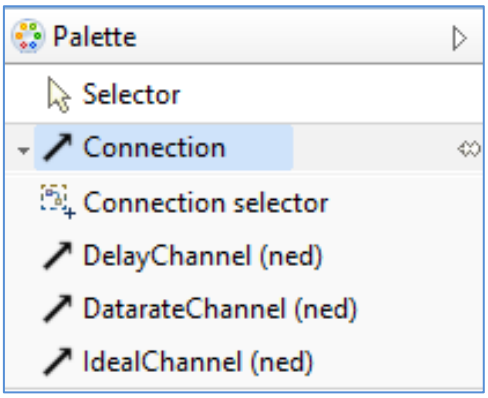

Рисунок 12 – Виды соединений

Теперь щёлкаем левой кнопкой мыши на обозначении соединения выбранного типа (**Connection**, **IdealChannel**, **DelayChannel**, **DatarateChannel**) в палитре инструментов, а затем поочерёдно на иконках узлов в рабочей области. Во всплывающем окне выбираем соединяемые порты узлов (рисунок 13).

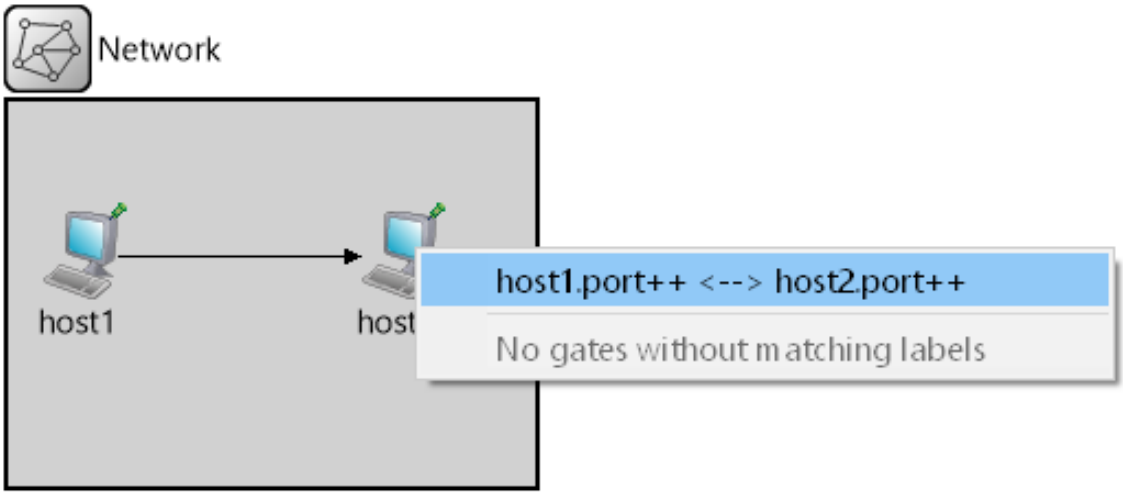

Рисунок 13 – Соединение узлов сети

На рисунке 13 видно, что если в ned-описании узла мы задали безразмерный вектор портов, то после создания очередного соединения номер порта увеличивается на единицу: `node.port++`.

Файл `Network.ned` выглядит следующим образом:

```
package mynet;

network Network
{
    @display("bgb=325,147");
    submodules:
        host1: Node {
            @display("p=74,68");
        }
        host2: Node {
            @display("p=245,68");
```

```
        }
    connections:
        host1.port++ <--> host2.port++;
}
```

В примере выбран универсальный тип соединения (**Connection**). Если бы мы выбрали, например, тип **DatarateChannel**, то раздел кода «Соединения» выглядел бы следующим образом:

```
…
    connections:
        host1.port++ <--> DatarateChannel <--> host2.port++;
…
```

Существует возможность также задавать параметры соединения (**Connection**). Например, задать канал с задержкой распространения, равной 10 мс можно следующим образом:

```
host1.port++ <--> {delay = 10ms;} <--> host2.port++;
```

1.3 Компиляция и запуск модели

Для компиляции модели используем команды меню: **Project => Build Project**. Первая компиляция, как правило, занимает довольно много времени.

Запуск модели можно осуществить через меню: **Run => Run as => OMNeT++ Simulation** или кнопкой **Run** панели инструментов Eclipse (рисунок 14).

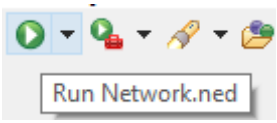

Рисунок 14 – Запуск

Запуск модели отображается в новом окне (рисунок 15), где можно просмотреть состав сети, анимацию событий, последовательность событий в порядке наступления, распределение временных интервалов и пр.

Симуляцию можно запустить кнопкой **Run**  , запись трейс-файла – кнопкой **REC**  , а остановить – кнопкой **Stop**  .

Поскольку созданная нами модель сети Network содержит только топологию, но не содержит событий, то при корректно заданных параметрах симуляция завершится сразу сообщением на рисунке 16.

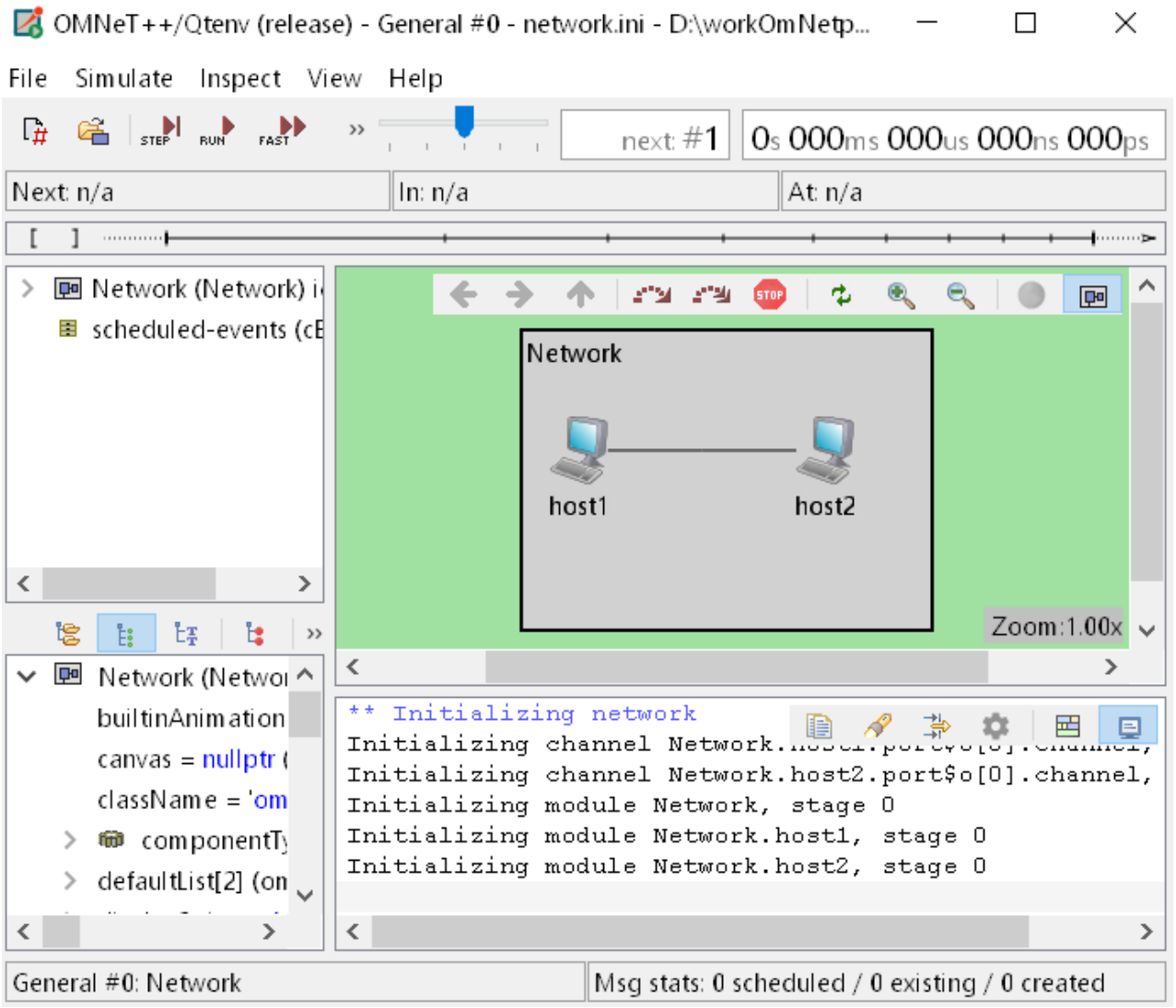

Рисунок 15 – Окно симуляции

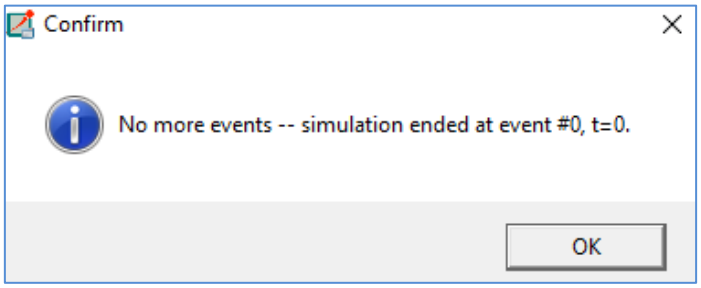

Рисунок 16 – Завершение примера симуляции

Задание:

1. Создать простой модуль Node для описания хоста сети, содержащий переменное количество шлюзов типа `inout` (ввод-вывод). Сохранить описание простого модуля в файле .ned.
2. В графическом редакторе NED создать топологию сети из четырёх узлов (хостов) в соответствии с вариантом. Топология сети (рисунок 17) и виды соединений заданы в таблице 1 по варианту (последние две цифры пароля). Сохранить топологию в файле `.ned`.
3. Просмотреть NED- и C++-коды узла.

4. Скомпилировать и запустить задачу на выполнение, посмотреть и зафиксировать результат.

Виды топологий представлены на рисунке 17 (выбираете ту, что указана в таблице 1 для вашего варианта).

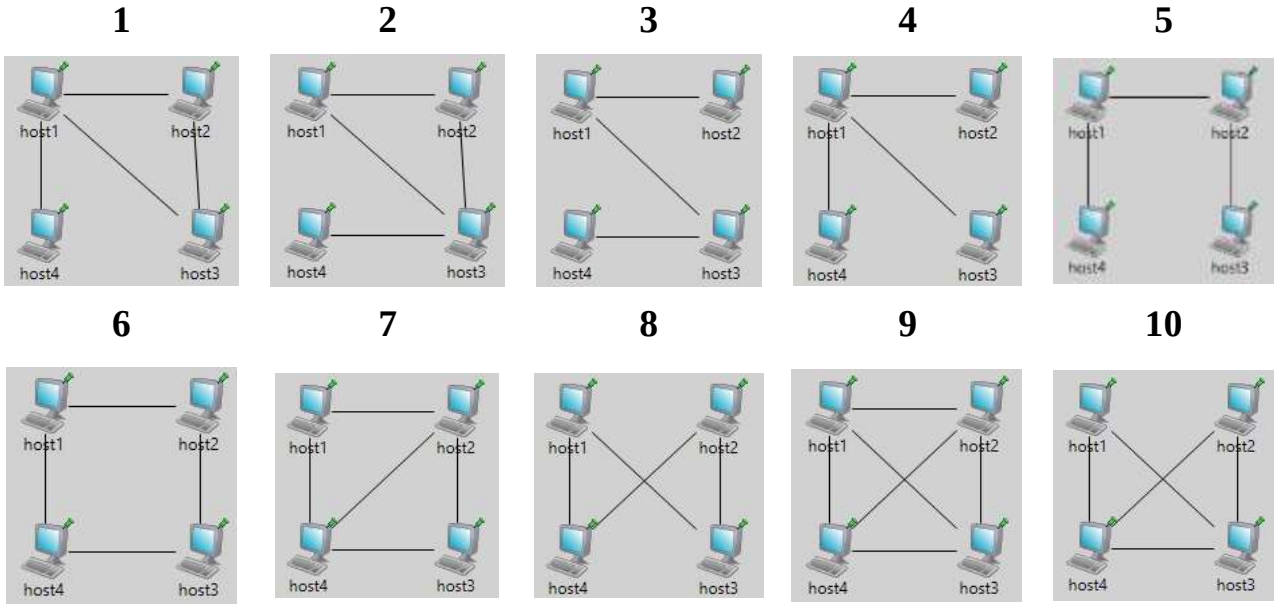

Рисунок 17 – Топологии сетей

Порядок выполнения:

1. Запустить симулятор OMNeT++.
2. Создать новый проект.
3. Создать в проекте простой модуль узел Node, содержащий переменное число шлюзов типа `inout`.
4. Изменить иконку узла.
5. Сохранить `Node.ned`.
6. Создать новую сеть Network.
7. Добавить к сети четыре узла Node.
8. Соединить шлюзы узлов линиями в соответствии с видом топологии и канала по варианту (таблица 1).
9. Сохранить `Network.ned`.
10. Просмотреть коды получившихся программ (`.ned, .cc, .h`).
11. Скомпилировать проект.
12. Исправить ошибки, если это необходимо.
13. Запустить программу на выполнение.
14. Если понадобится, повторить пп.12-11-13.
15. Оформить отчёт.
16. Сдать и защитить лабораторную работу.

Содержание отчёта:

1. Номер, название и цель лабораторной работы.
2. ФИО и группа студента, выполнившего лабораторную работу.
3. Номер варианта и задание к лабораторной работе.
4. Топология моделируемой сети.
5. Коды получившихся модулей и соответствующих им программ (`.ned`, `.cc`, `.h`).
6. Скриншоты NED-файлов в графическом режиме.
7. Результаты выполнения лабораторной работы (визуализация) со скриншотами.
8. Содержимое `elog`-файла.
9. Описание выполнения лабораторной работы и полученных результатов.
10. Выводы по проделанной работе.

Контрольные вопросы:

1. Чем отличаются друг от друга простой и составной модуль в симуляторе OMNeT++?
2. Для чего используется редактор NED? Как расшифровывается его название?
3. Поясните в двух словах процесс создания топологии сети.
4. Что содержит раздел `gates` в коде, описывающем простой модуль?
5. Какие виды соединений по умолчанию используются в симуляторе OMNeT++?

Варианты заданий:

Таблица 1 – Структура сети*

№ топологии / **Тип канала**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
IdealChannel	6, 50	25, 72	15, 41	18, 48	35, 74	40, 75	12, 58	20, 52	38, 65	4, 45
DatarateChannel	21, 66	8, 57	29, 67	39, 77	14, 55	1, 64	31, 61	5, 46	24, 80	27, 70
DelayChannel	10, 44	33, 49	2, 59	7, 69	23, 42	17, 62	28, 79	37, 76	13, 53	16, 60
Connection	36, 73	22, 51	11, 43	34, 63	3, 68	30, 47	9, 56	26, 78	19, 71	32, 54

* – № варианта в ячейке таблицы (*последние две цифры пароля*), заголовки строки и столбца содержат значения, соответствующие варианту. Каждая ячейка содержит два номера вариантов. **К примеру, если у вас вариант 21** (число 21 расположено в таблице на пересечение столбца 1 и строки DatarateChannel)**, то берете топологию сети 1 на рис.17 и тип канала DatarateChannel.**

Лабораторная работа №2

Симуляция в OMNeT++

Цель работы: Овладеть основными принципами моделирования в среде OMNeT++. Научиться описывать поведение OMNeT++ модели с помощью C++ кода. Разобраться с конфигурацией модели и записью трассировки симуляции.

Краткая теория:

Симулятор OMNeT++ осуществляет дискретно-событийное моделирование. Это значит, что состояние сети рассматривается только в определённые моменты времени и не рассматривается в промежутках между ними. В OMNeT++ такими моментами являются моменты поступления сообщений (англ. *Message*).

NED-файлы, рассмотренные в лабораторной работе №1, описывают топологию сети, но не её динамику. Для алгоритмизации функционирования сети применяются библиотеки компонентов C++.

2.1 Редактирование C++ файлов для описания поведения модели

Симулятор автоматически генерирует код программы при создании модулей на языке NED. Код представляет собой набор пустых шаблонов методов, которые необходимо заполнить содержимым, для симуляции работы сети.

В примере из лабораторной работы №1 автоматически сгенерированный код Simple Module узла Node, сохранённый в файле Node.cc выглядит следующим образом:

```
#include "Node.h"

Define_Module(Node);

void Node::initialize()
{
    // TODO - Generated method body
}

void Node::handleMessage(cMessage *msg)
{
    // TODO - Generated method body
}
```

В приведённом коде `Define_Module(Node)` – макрос, который связывает файлы `Node.cc` и `Node.ned`.

Основные методы симуляции:

- `initialize()` – инициализация. Метод вызывается OMNeT++ после установки сети;

- `finish()` – завершение. Метод вызывается автоматически при успешном завершении симуляции и записи статистики. Его можно использовать для записи статистики;
- `handleMessage(cMessage *msg)` – обработка сообщения. Метод вызывается с параметром `msg`, когда модуль принимает сообщение.

`cMessage` – C++ класс для описания сообщений в сети.

Основные методы для работы с сообщениями:

- `send()` – посылка сообщений другим модулям;
- `scheduleAt()` – планирование событий;
- `cancelEvent()` – отмена запланированного события.

Предположим, что каждый из двух узлов сети Network отправляет сообщение через определённые промежутки времен, заданные в NED-файле.

Для задания размера интервала (`Interval`) внесём изменения в файл `Node.ned` (заметим, что входной и выходной порты здесь разделены):

```
package mynet;

simple Node
{
    @display("i=device/pc");
    double Interval @unit(s) = default(0.1s);
    gates:
          input in;
          output out;
}
```

Мы добавили в NED файл строку с описанием переменной вещественного типа `double Interval`, содержащей значение промежутка времени между последовательными передачами сообщений – 0.1 сек:

```
double Interval @unit(s) = default(0.1s);
```

В примере `@unit(measurement_unit)` определяет единицу измерения параметра (в данном случае – секунды).

Генерирование случайных промежутков времени предполагает использование датчиков случайных чисел симулятора вместо константного значения. При этом для описания переменной необходимо кроме её типа указывать модификатор **`volatile`**, который означает, что значение переменной будет пересчитываться каждый раз при считывании этого значения:

```
volatile double value;
```

Файл `Node.cc` с изменениями будет следующим:

```
#include "Node.h"

Define_Module(Node);
```

```
 void Node::initialize()
 {
// время первого события (сообщения)
      cMessage *event = new cMessage("Event");
// значение интервала времени, взятое из NED файла
      interval = par("Interval");
// планирование первого события через промежуток interval
      scheduleAt(simTime() + interval, event);
    }
 void Node::handleMessage(cMessage *msg)
 {
 // если событие внутреннее (взятое из планировщика)
      if(msg -> isSelfMessage()) {
      // генерируем сообщение Data
        cMessage *data = new cMessage("Data");
      // отправляем через порт out
        send(data, "out");

      // планируем время следующего события
        cMessage *event = new cMessage("Event");
        interval = par("Interval");
        scheduleAt(simTime() + interval, event);
    }
      else {
           delete(msg);
    }
 }
```

Можно использовать макрос `WATCH(variable)` для просмотра значений заданной переменной (например, `interval`).

2.2 Изменение h файла

Для того, чтобы методы симуляции могли обмениваться данными, необходимо внести изменения в заголовочный файл Node.h, который содержит глобальные переменные и прототипы методов (приводится без директив препроцессора):

```
class Node : public cSimpleModule
      {
       double interval;
       protected:
          virtual void initialize();
          virtual void handleMessage(cMessage *msg);
      };

      #endif
```

2.3 Адаптация модели к топологии сети Network

Обратите внимание на то, что в примере соединены два узла и выбора, кому отправлять сообщение, не предоставляется. Топология сети в лабораторной работе может соединять каждый узел с двумя или тремя другими, поэтому необходим выбор, кому отправлять сообщение (или всем сразу).

Для этого можно использовать датчик дискретных случайных чисел следующим образом:

```
int i = intrand(this -> gateSize("port$o"));
send(data, "port$o", i);
```

Или перебор всех подключенных узлов в цикле следующим образом:

```
for(int i = 0; i < this -> gateSize("port$o"); i++)
{
    cMessage *data = new cMessage("Data");
    send(data, "port$o", i);
}
```

В первом случае текущий узел отправит сообщение случайно выбранному узлу, во втором случае сообщения будут отправлены всем подключенным к текущему узлу узлам.

2.4 Окно симуляции

При запуске симуляции мы увидим окно, представленное на рисунке 18.

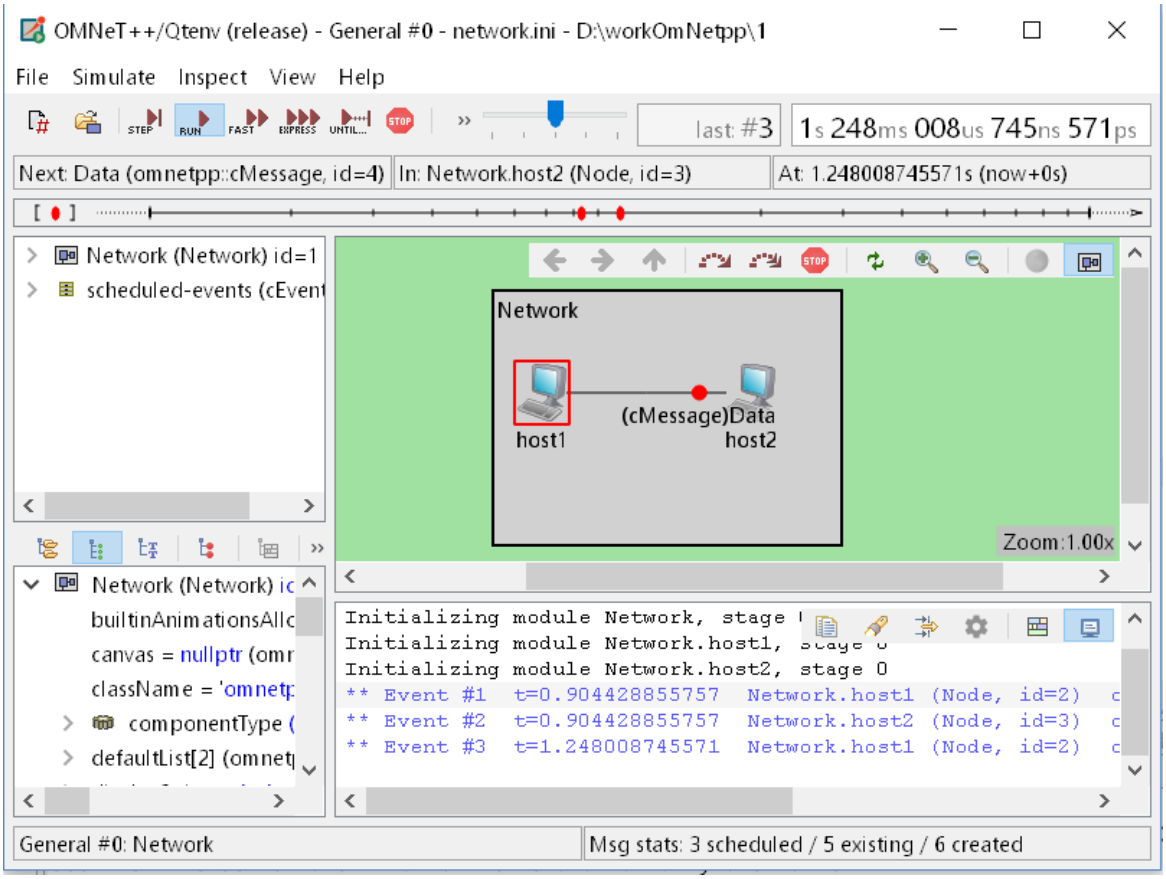

Рисунок 18 – Симуляция событий

Верхняя строка окна содержит меню для управления параметрами симуляции. Ниже размещена панель инструментов, позволяющая осуществить запуск симуляции, её останов, ускорить или замедлить процесс симуляции, а также произвести запись результатов для их последующей обработки.

Под ними располагается строка, в которой отображается тип и время следующего события симуляции, а расположенная под ней временная ось показывает временные интервалы между событиями.

В следующем ряду слева расположена область с описанием сети, её модулей и сообщений, а справа – рабочее пространство анимации сетевых процессов.

Ниже слева находится область с описанием объектов и переменных симуляции. Справа выводится список событий симуляции по порядку.

Заканчивается окно симуляции строкой статуса.

Как видно на рисунке 18, сообщения, которыми обмениваются узлы, обозначены в анимации перемещающимися красными шариками с подписью (cMessage)Data, а узел-передатчик обведён красной рамкой.

Задание:

Осуществить симуляцию работы сети Network из лабораторной работы №1. Для этого внести изменения в файлы `.ned`, `.cc`, `.h` и отредактировать файл конфигурации. Скомпилировать и запустить симуляцию. Просмотреть результаты.

Порядок выполнения:

1. Запустить симулятор OMNeT++.
2. Открыть проект из лабораторной работы №1.
3. Отредактировать коды программ, описывающих передачу сообщений между узлами (.ned, .cc, .h), в соответствии с вариантом задания (по двум последним цифрам пароля).
4. Задать время симуляции 20 сек. в файле OMNeTpp.ini.
5. Скомпилировать проект.
6. Исправить ошибки, если это необходимо, повторить п.5.
7. Включить запись трейс-файла.
8. Запустить программу на выполнение.
9. Просмотреть анимацию, сделать скриншоты для отчёта.
10. После окончания симуляции просмотреть в рабочей области проекта содержимое файла `General-0.elog` из папки `results`.
11. Оформить отчёт по лабораторной работе.

Содержание отчёта:

1. Номер, название и цель лабораторной работы.
2. ФИО и группа студента, выполнившего лабораторную работу.
3. Номер варианта и задание к лабораторной работе.

4. Структура моделируемой сети.
5. Содержимое файлов `.ned`, `.cc`, `.h`.
6. Результаты выполнения лабораторной работы в графическом виде и в виде скриншотов.
7. Содержимое файла выполнения (`.elog`).
8. Описание выполнения лабораторной работы и полученных результатов.
9. Выводы по проделанной работе.

Контрольные вопросы:

1. Какие методы отвечают в симуляторе OMNeT++ за работу с сообщениями?
2. Для чего в программе нужен метод `finish()`?
3. Каким образом задать в коде симуляции случайную величину?
4. За что отвечает файл с расширением .h?
5. Куда OMNeT++ записывает результаты симуляции?

Варианты заданий:

Топология сети и тип соединения соответствуют варианту из лабораторной работы №1.

Вариант лабораторной работы №2 (таблица 2) определяет вид распределения интервалов времени между последовательными сообщениями узлов.

Распределение интервалов времени между сообщениями (табл.2)

1. Усечённое нормальное;
2. Экспоненциальное;
3. Равномерное;
4. Гамма;
5. Логарифмически нормальное;
6. Бета;
7. Эрланга;
8. Стьюдента;
9. Вейбулла;
10. Парето.

Таблица 2 – Варианты заданий

Топология из лаб. раб. №1										
Варианты (в скобках рядом с номером варианта указан номер распределения)	6(2), 50(4)	25(9), 72(1)	15(3), 41(9)	18(7), 48(3)	35(1), 74(7)	40(4), 75(6)	12(6), 58(8)	20(5), 52(10)	38(8), 65(2)	4(10), 45(5)
	21(1), 66(8)	8(10), 57(3)	29(5), 67(4)	39(9), 77(5)	14(6), 55(10)	1(8), 64(9)	31(3), 61(7)	5(7), 46(2)	24(4), 80(1)	27(2), 70(6)
	10(3), 44(6)	33(7), 49(2)	2(8), 59(1)	7(10), 69(8)	23(2), 42(5)	17(5), 62(10)	28(4), 79(9)	37(3), 76(4)	13(7), 53(3)	16(9), 60(7)
	36(5), 73(7)	22(8), 51(5)	11(7), 43(2)	34(2), 63(6)	3(4), 68(9)	30(1), 47(3)	9(10), 56(1)	26(6), 78(8)	19(9), 71(10)	32(3), 54(4)

К примеру, последние цифры пароля 21, распределение интервалов времени будет 1 – усечённое нормальное.

Лабораторная работа №3

Запись и обработка результатов моделирования в OMNeT++

Цель работы: Освоить типы данных и методы для записи результатов симуляции, а также графический вывод результатов в окне OMNeT++.

Краткая теория:

В OMNeT++ возможны три типа результатов симуляции:

- вектор;
- скаляр;
- гистограмма.

Для записи результатов можно включить в файл `omnetpp.ini` строку `record-eventlog = true`.

Проследить изменение значений вектора можно, используя консольный вывод языка C++:

```
EV << "I'm: " << this -> getName() << ". My counter = " << counter << endl;
```

3.1 Запись вектора

Для записи результатов в виде вектора необходимо определить (например, в файле `.h`) объект класса `cOutVector` и использовать методы этого класса в методе `handleMessage(cMessage *msg)`.

Например:

```
        …
cOutVector VectorX; // определим объект класса cOutVector
int counter = 0;       // инициализируем счётчик
        …
VectorX.setName("ValueX"); // зададим имя вектора
        …
VectorX.record(counter++); // запишем текущее значение
// счётчика в вектор и увеличим значение на 1
        …
```

3.2 Запись скаляра

Скалярное (`Scalar`) значение определяется с помощью примитивного типа данных (`int`, `double`, `char` и т.д) и запись его осуществляется в методе `finish()` модуля. Для записи используется метод `recordScalar()`:

Например:

```
void Node::finish()
{
     …
   recordScalar("My counter", counter);
     …
 }
```

3.3 Запись гистограммы

Гистограмма показывает распределение значений результирующего параметра. По оси абсцисс откладываются диапазоны значений параметра, по оси ординат – количество значений параметра, попадающих в данные диапазоны за время симуляции. OMNeT++ автоматически вычисляет статистические характеристики гистограммы (среднее, дисперсию, минимальное, максимальное значения).

Запись гистограммы производится с помощью объекта класса `cHistogram` методом `recordAs("Histogram")` вызываемого в методе `finish()`, где в кавычках указывается имя гистограммы.

Пример:

```
      …
cHistogram HistogramX;    // объект гистограмма
      …
HistogramX.setName("Histogram"); // имя гистограммы
      …
```

Метод `collect(parameter)` служит для сбора статистики параметра `parameter` в методе `handleMessage(cMessage *msg)`.

Пример:

```
     …
HistogramX.collect(x); // распределение параметра x
     …
```

Пример записи гистограммы:

```
void Node::finish()
{
     …
   HistogramX.recordAs("Histogram");
     …
 }
```

Для вычисления статистики можно использовать методы: `getCount()`, `getMin()`, `getMax()`, `getMean()`, `getStddev()`, `getVariance()`, `getSum()`, `getSqrSum()`.

3.4 Графический вывод результатов симуляции

Результаты симуляции будут записаны в папку `results` рабочей области OMNeT++ (если вы не забыли нажать кнопку `Rec` или не добавили команду записи в файл конфигурации).

В папке `results` будет создано несколько файлов (рисунок 19) в зависимости от того запись результатов какого типа вы поместили в модель (вектор, скаляр, гистограмма).

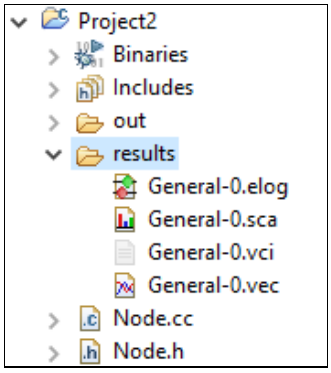

Рисунок 19 – Файлы с результатами симуляции в дереве проекта

Щёлкнув левой кнопкой мыши на названии файла в папке results, мы откроем окно обработки статистических данных и графического вывода результатов `General.anf` (рисунок 20).

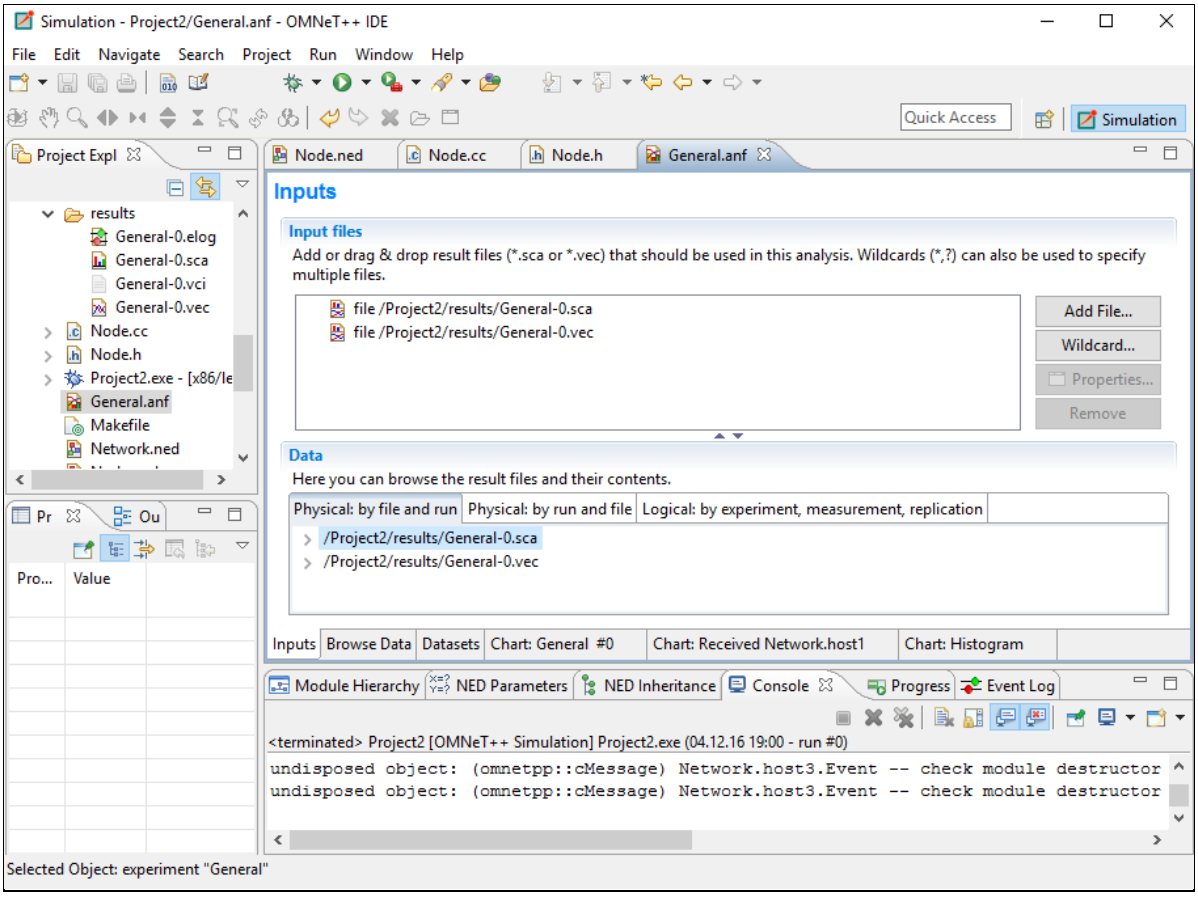

Рисунок 20 – Вывод результатов симуляции

Вкладка `Inputs` позволяет добавить данные в окно `General.anf`. Вкладка Browse Data служит для просмотра и выбора данных для вывода на график (рисунки 21-22).

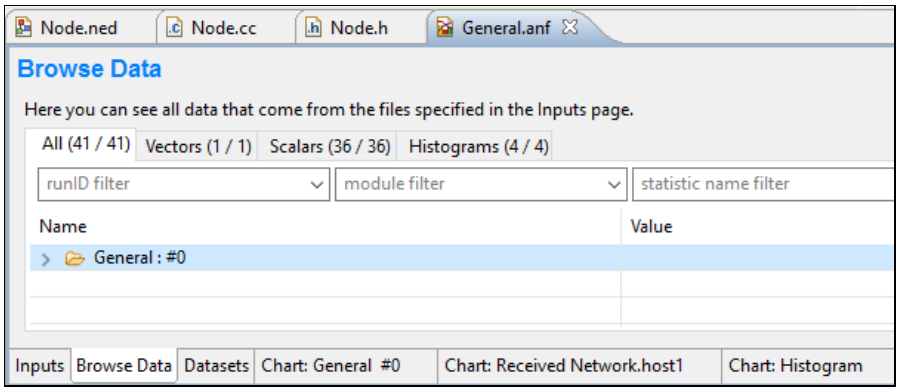

Рисунок 21 – Вкладка Browse Data – All

Щёлкнув на имени папки `General : #0`, мы можем вывести все графики и гистограммы в окне `General.anf`.

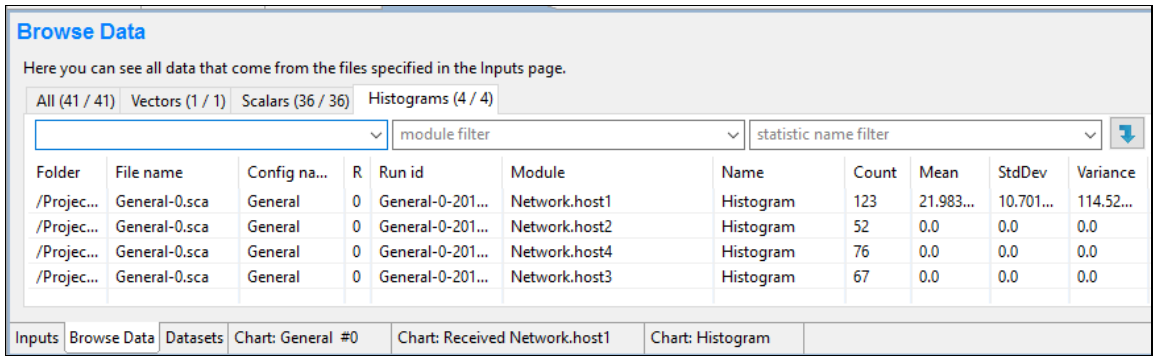

Рисунок 22 – Вкладка Browse Data – Histograms

Вкладка Datasets позволяет отредактировать графики (цвета, подписи осей и т.д.).

Примеры графика и гистограммы приведены на рисунках 23-24.

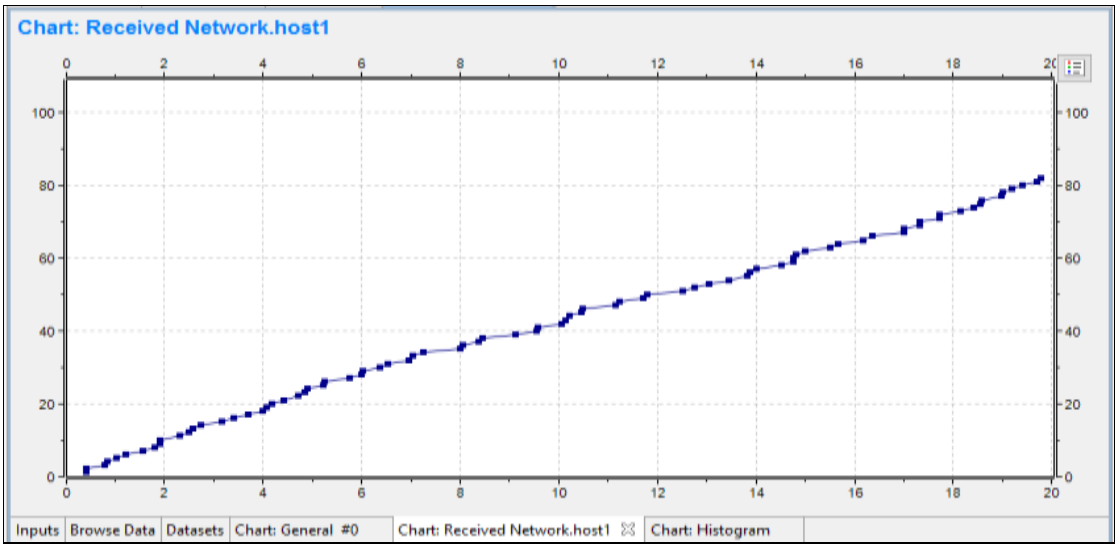

Рисунок 23 – Пример графика зависимости количества принятых сообщений от времени

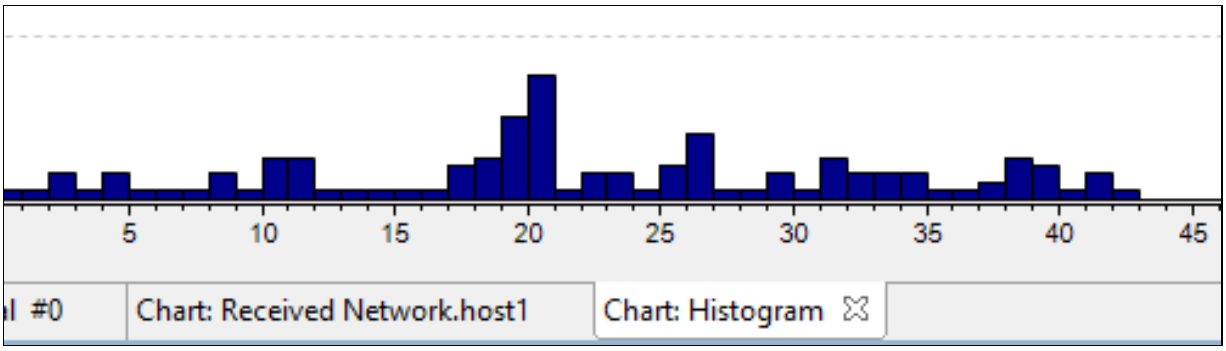

Рисунок 24 – Распределение разности между принятыми и переданными сообщениями

Задание:

Добавить в проект из лабораторных работ №№1-2 средства для записи и вывода результатов симуляции в соответствии с вариантом для заданного хоста. Запустить симуляцию с записью результатов. Вывести графические и скалярные результаты, отредактировать их. Просмотреть и сохранить результаты.

Порядок выполнения:

1. Запустить симулятор OMNeT++.
2. Открыть проект из лабораторных работ №№1-2.
3. Добавить в коды программ средства записи результатов симуляции в соответствии с вариантом задания.
4. Скомпилировать проект.
5. Исправить ошибки, если это необходимо, повторить п.5.
6. Включить запись трейс-файла.
7. Запустить программу на выполнение.
8. Просмотреть анимацию, сделать скриншоты для отчёта.
9. Вывести в окне General.anf результаты выполнения в графическом виде и скалярные значения.
10. Оформить графики титульными надписями и подписями осей.
11. Сделать скриншоты графических и скалярных результатов.
12. Оформить отчёт по лабораторной работе.
13. Сдать и защитить работу.

Содержание отчёта:

1. Номер, название и цель лабораторной работы.
2. ФИО и группа студента, выполнившего лабораторную работу.
3. Номер варианта и задание к лабораторной работе.
4. Структура моделируемой сети.
5. Результаты выполнения лабораторной работы в графическом виде и в виде скриншотов.

6. Полученные трейс-файлы.
7. Описание выполнения лабораторной работы и полученных результатов.
8. Выводы по проделанной работе.

Контрольные вопросы:

1. Поясните вкратце, как можно сохранить результат симуляции OMNeT++ в виде скалярного значения.
2. Как определить объект-вектор.
3. Объясните, какой смысл имеет результат симуляции гистограмма.
4. Какие средства статистической обработки результатов симуляции имеет пакет OMNeT++?
5. Что понимается в симуляторе под *трассировкой событий*?

Варианты заданий:

Топологии сетей и распределения промежутков времени соответствуют вариантам из лабораторных работ 1-2.

Результаты:

1. Количество принятых сообщений; 2. Количество отправленных сообщений; 3. Длина интервала; 4. Общее количество событий (типа Event и Data); 5. Сумма отправленных и принятых сообщений;	6. Количество событий типа Event; 7. Текущие моменты времени при получении сообщений; 8. Текущие моменты времени при отправлении сообщений; 9. Средняя длина интервала; 10. Минимальная длина интервала; 11. Максимальная длина интервала;

Таблица 3 – Варианты заданий

Гистограмма	**5**	**2**	**3**	**5**	**1**	**3**	**4**	**6**	**3**	**5**
Скаляр	**1**	**10**	**9**	**2**	**11**	**6**	**5**	**10**	**4**	**9**
Вектор	**8**	**6**	**7**	**8**	**3**	**2**	**1**	**5**	**6**	**4**
№ Хоста										
1	6, 50	25, 72	15, 41	18, 48	35, 74	40, 75	12, 58	20, 52	38, 65	4, 45
2	21, 66	8, 57	29, 67	39, 77	14, 55	1, 64	31, 61	5, 46	24, 80	27, 70
3	10, 44	33, 49	2, 59	7, 69	23, 42	17, 62	28, 79	37, 76	13, 53	16, 60
4	36, 73	22, 51	11, 43	34, 63	3, 68	30, 47	9, 56	26, 78	19, 71	32, 54

Например, вариант 21, тогда хост 2, гистограмма 5, скаляр 1, вектор 8

Литература

1. OMNeT++ Simulation Manual [Электронный ресурс]. URL : https://doc.omnetpp.org/omnetpp/manual/

2. OMNeT++ User Guide [Электронный ресурс]. URL : https://doc.omnetpp.org/omnetpp/ UserGuide.pdf/

3. Chamberlain T. Learning OMNeT++ // Packt Publishing. 2013. 102 p.

Учебное издание

Елена Викторовна Кокорева

МОДЕЛИРОВАНИЕ ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ В СЕТЕВОМ СИМУЛЯТОРЕ OMNET++

Редактор: К.И. Шурыгина

Подписано в печать
Формат бумаги 62×84/16, отпечатано на ризографе, шрифт №10,
п. л. 2,0, заказ №, тираж 5.

Редакционно-издательский отдел СибГУТИ
СибГУТИ 630102, Новосибирск, ул. Кирова, 86