

# Презентация по лабораторной работе №1

Дисциплина: Имитационное моделирование

---

Лобанова П.И.

11 февраля 2025

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Лобанова Полина Иннокентьевна
- Учащаяся на направлении “Фундаментальная информатика и информационные технологии”
- Студентка группы НФИбд-02-22
- [polla-2004@mail.ru](mailto:polla-2004@mail.ru)

## Цель работы

---

Приобретение навыков моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также анализ полученных результатов моделирования.

## Задание

---

Создать шаблон сценария для NS-2.

## Выполнение

---

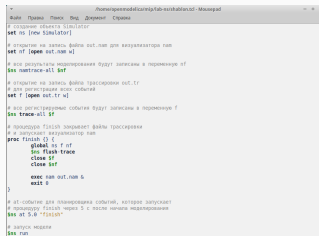


В своём рабочем каталоге создала директорию `mip`, к которой будут выполняться лабораторные работы. Внутри `mip` создала директорию `lab-ns`, а в ней файл `shablon.tcl`.

```
openmodelica@openmodelica-VirtualBox:~$ mkdir -p mip/lab-ns
openmodelica@openmodelica-VirtualBox:~$ cd mip/lab-ns
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ touch shablon.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ls
shablon.tcl
```

Рис. 1: Создание каталогов и файла

Сначала создала объект типа Simulator. Затем создала переменную nf и указала, что требуется открыть на запись nam-файл для регистрации выходных результатов моделирования. Далее создала переменную f и открыла на запись файл трассировки для регистрации всех событий модели. После этого добавила процедуру finish, которая закрывает файлы трассировки и запускает nam. Наконец, с помощью команды at указала планировщику событий, что процедуру finish следует запустить через 5 с после начала моделирования, после чего запустить симулятор ns.



```
#!/usr/bin/perl

set ns [new Simulator]

# открытие на запись файла out.nam для визуализатора nam
set nf [open out.nam w]

# все результаты моделирования будут записаны в переменную nf
$ns namtrace-all $nf

# открытие на запись файла трассировки out.tr
# все регистрации всех событий
set f [open out.tr w]

# все зарегистрированные события будут записаны в переменную f
$ns trace-all $f

# процедура finish закрывает файлы трассировки
# и запускает визуализатор nam
proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

# at-событие для планировщика событий, которое запускает
# процедуру finish через 5 с после начала моделирования
$ns at 5.0 "finish"

# запуск модели
$ns run
```

Рис. 2: Заполнение шаблона

Сохранив изменения в отредактированном файле `shablon.tcl` и закрыв его, запустила симулятор.

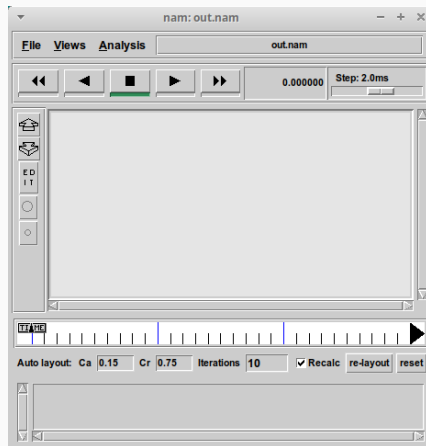


Рис. 3: Аниматора *nam*

## Задание

---

Требуется смоделировать сеть передачи данных, состоящую из двух узлов, соединённых дуплексной линией связи с полосой пропускания 2 Мб/с и задержкой 10 мс, очередью с обслуживанием типа DropTail. От одного узла к другому по протоколу UDP осуществляется передача пакетов, размером 500 байт, с постоянной скоростью 200 пакетов в секунду.

## Выполнение

---

Скопировала содержимое созданного шаблона в новый файл.

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example1.tcl
```

Рис. 4: Копирование шаблона

Добавила описание топологии сети. Создала агенты для генерации и приёма трафика. Далее создала Null-агент, который работает как приёмник трафика, и прикрепила его к узлу n1. Соединила агенты между собой. Для запуска и остановки приложения CBR добавляются at-события в планировщик событий.

A screenshot of a text editor window titled "C:\ProgramData\minipab-nu\example1.txt - Notepad". The editor contains a script for a network simulation. The script starts with a function definition for "close", followed by comments in Russian describing the setup of two nodes, a duplex link, and various agents (UDP, CBR, Null). It includes commands for setting packet size, interval, and starting/stopping the CBR application at specific times. The script ends with an "at" event for a "finish" procedure.

```
close br
close snf
exec nam out.nam &
exit 0
}

# создание 2-х узлов:
set N 2
for (set i 0) ($i < $N) {incr i} {
    set n($i) [$ns node]
}

# соединение 2-х узлов дуплексным соединением
# с полосой пропускания 2 Mb/s и задержкой 10 мс,
# очередь с обслуживанием типа DropTail
$ns duplex-link $n(0) $n(1) 2Mb 10ms DropTail

# создание агента UDP и присоединение его к узлу n0
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

# создание источника трафика CBR (constant bit rate)
set cbr0 [new Application/Traffic/CBR]

# устанавливаем размер пакета в 500 байт
$cbro set packetSize 500

# задаем интервал между пакетами равным 0.005 секунды,
# т.е. 200 пакетов в секунду
$cbro set interval_ 0.005

# присоединение источника трафика CBR к агенту udp0
$cbro attach-agent $udp0

# Создание агента-приёмника и присоединение его к узлу n(1)
set null0 [new Agent/Null]
$ns attach-agent $n(1) $null0

# Соединение агентов между собой
$ns connect $udp0 $null0

# запуск приложения через 0,5 с
$ns at 0.5 "$cbro start"

# остановка приложения через 4,5 с
$ns at 4.5 "$cbro stop"

# at-событие для планировщика событий, которое запускает
# процедуру finish через 5 с после начала моделирования
$ns at 5.0 "finish"
```

Рис. 5: Заполнение файла



Сохранила изменения в отредактированном файле и запустила симулятор.

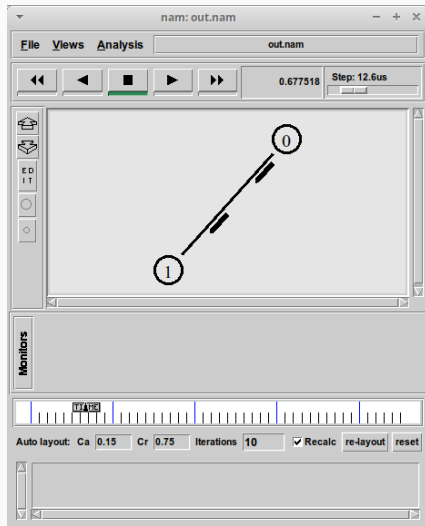


Рис. 6: Аниматора *nam*

## Задание

---

- Описание моделируемой сети: – сеть состоит из 4 узлов ( $n_0$ ,  $n_1$ ,  $n_2$ ,  $n_3$ );
- между узлами  $n_0$  и  $n_2$ ,  $n_1$  и  $n_2$  установлено дуплексное соединение с пропускной способностью 2 Мбит/с и задержкой 10 мс;
  - между узлами  $n_2$  и  $n_3$  установлено дуплексное соединение с пропускной способностью 1,7 Мбит/с и задержкой 20 мс;
  - каждый узел использует очередь с дисциплиной DropTail для накопления пакетов, максимальный размер которой составляет 10;
  - TCP-источник на узле  $n_0$  подключается к TCP-приёмнику на узле  $n_3$
  - TCP-приёмник генерирует и отправляет ACK пакеты отправителю и откидывает полученные пакеты;

- UDP-агент, который подсоединён к узлу n1, подключён к null-агенту на узле n3 (null-агент просто откидывает пакеты);
- генераторы трафика ftp и cbr прикреплены к TCP и UDP агентам соответственно;
- генератор cbr генерирует пакеты размером 1 Кбайт со скоростью 1 Мбит/с;
- работа cbr начинается в 0,1 секунду и прекращается в 4,5 секунды, а ftp начинает работать в 1,0 секунду и прекращает в 4,0 секунды.

## Выполнение

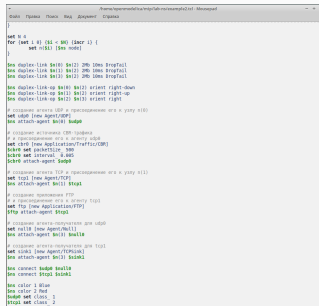
---

Скопировала содержимое созданного шаблона в новый файл.

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example2.tcl
```

*Рис. 7: Копирование шаблона*

Создала 4 узла и 3 дуплексных соединения с указанием направления. Создала агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP. Создала агенты-получатели. Соединила агенты udr0 и tcp1 и их получателей. Задала описание цвета каждого потока. Добавила отслеживание событий в очереди, наложение ограничения на размер очереди и at-события.



```
set n 4
for {set i 0} {N1 < N0} {incr i} {
    set n($i) {Sim node}

    #s duplex-link $n($i) $n($i+1) 2Mb 10ms DropTail
    #s duplex-link $n($i+1) $n($i) 2Mb 10ms DropTail
    #s duplex-link $n($i) $n($i+2) 2Mb 10ms DropTail

    #s duplex-link-ep $n($i) $n($i+2) orient right-ohb
    #s duplex-link-ep $n($i+2) $n($i) orient right-ohb
    #s duplex-link-ep $n($i+2) $n($i+3) orient right
    #s duplex-link-ep $n($i+3) $n($i+2) orient right

    # создание агента UDP и присоединение его к узлу n($i)
    set udr0 {new Agent/UDP}
    #s attach-agent $n($i) $udr0

    # создание источника CBR-трафика
    #s в присоединение его к агенту udr0
    set cbr0 {new Application/Traffic/CBR}
    #s set packetSize 1000
    #s set interval 0.005
    #s attach-agent $udr0

    # создание агента TCP и присоединение его к узлу n($i)
    set tcp1 {new Agent/TCP}
    #s attach-agent $n($i) $tcp1

    # создание приложения FTP
    #s присоединение его к агенту tcp1
    set ftp {new Application/FTP}
    #s attach-agent $tcp1

    # создание агента-получателя для udr0
    set r0 {new Agent/Null}
    #s attach-agent $n($i+1) $r0

    # создание агента-получателя для tcp1
    set r1 {new Agent/RCPSink}
    #s attach-agent $n($i+3) $r1

    #s connect $udr0 $r0
    #s connect $tcp1 $r1

    #s color 1 Blue
    #s color 2 Red
    #s set class 1
    #s set class 2
}
```

Рис. 8: Заполнение файла

Сохранив изменения в отредактированном файле и запустив симулятор, получила анимированный результат моделирования.

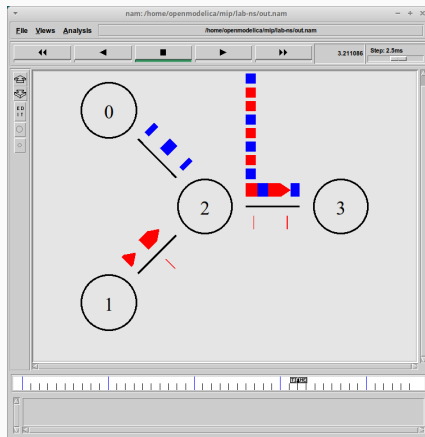


Рис. 9: Аниматора *nam*



## Задание

---

- Требуется построить модель передачи данных по сети с кольцевой топологией и динамической маршрутизацией пакетов: – сеть состоит из 7 узлов, соединённых в кольцо;
- данные передаются от узла  $n(0)$  к узлу  $n(3)$  по кратчайшему пути;
  - с 1 по 2 секунду модельного времени происходит разрыв соединения между узлами  $n(1)$  и  $n(2)$ ;
  - при разрыве соединения маршрут передачи данных должен измениться на резервный.

## Выполнение

---

Скопировала содержимое созданного шаблона в новый файл.

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example3.tcl
```

Рис. 10: *Копирование шаблона*



Сохранив изменения в отредактированном файле и запустив симулятор, получила анимированный результат моделирования.

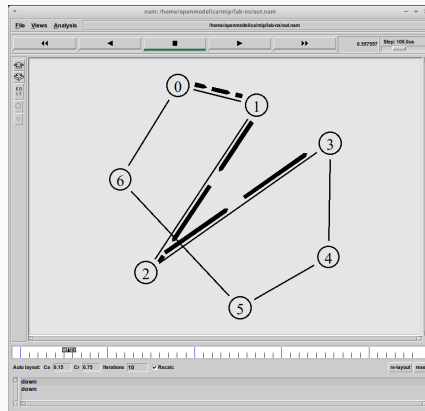


Рис. 12: Передача данных по кратчайшему пути



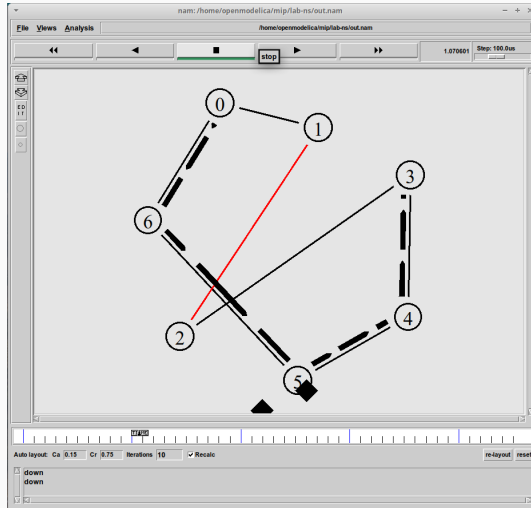


Рис. 14: Передача данных по резервному маршруту



## Задание

---

Внесите следующие изменения в реализацию примера с кольцевой топологией сети:

- передача данных должна осуществляться от узла  $n(0)$  до узла  $n(5)$  по кратчайшему пути в течение 5 секунд модельного времени;
- передача данных должна идти по протоколу TCP (тип Newreno), на принимающей стороне используется TCPSink-объект типа DelAck; поверх TCP работает протокол FTP с 0,5 до 4,5 секунд модельного времени;
- с 1 по 2 секунду модельного времени происходит разрыв соединения между узлами  $n(0)$  и  $n(1)$ ;
- при разрыве соединения маршрут передачи данных должен измениться на резервный, после восстановления соединения пакеты снова должны пойти по кратчайшему пути.

## Выполнение



Скопировала содержимое созданного шаблона в новый файл.

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl ex1.tcl
```

Рис. 15: *Копирование шаблона*



Сохранив изменения в отредактированном файле и запустив симулятор, получила анимированный результат моделирования.

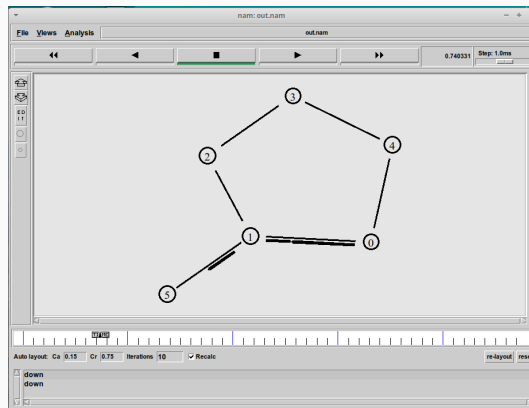


Рис. 17: Передача данных по кратчайшему пути

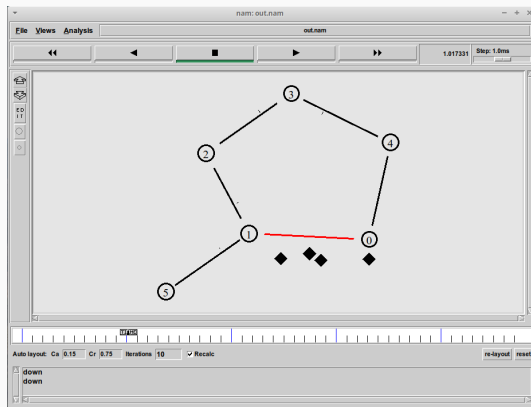


Рис. 18: Прерывание соединения

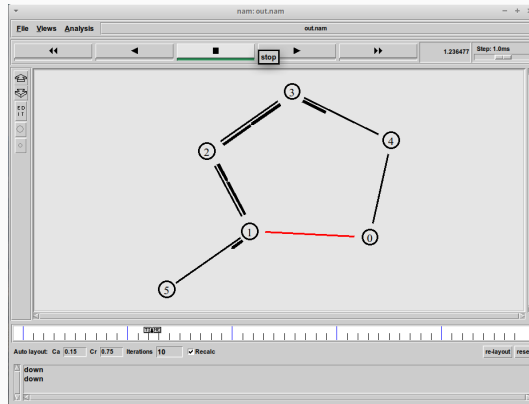


Рис. 19: Передача данных по резервному маршруту



## Вывод

---

Я приобрела навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также анализ полученных результатов моделирования.