

# **Отчёт по лабораторной работе №1**

**Дисциплина: Имитационное моделирование**

Ганина Таисия Сергеевна, НФИбд-01-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
3.0.1	Процесс создания модели сети для NS-2 состоит из нескольких этапов: . . . . .	8
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
<b>5</b>	<b>Выводы</b>	<b>31</b>
	<b>Список литературы</b>	<b>32</b>

## Список иллюстраций

4.1	Создание файлов . . . . .	10
4.2	Заполнение файла шаблона . . . . .	11
4.3	Результат выполнения файла шаблона, приложение . . . . .	11
4.4	Копирование шаблона . . . . .	12
4.5	Код программы . . . . .	12
4.6	Аниматор <code>nam</code> . . . . .	13
4.7	Визуализация простой модели сети с помощью <code>nam</code> . . . . .	14
4.8	Можно осуществлять наблюдение за отдельным пакетом, щёлкнув по нему в окне <code>nam</code> . . . . .	15
4.9	Кадр с неточностью . . . . .	16
4.10	Код программы . . . . .	16
4.11	Продолжение кода . . . . .	17
4.12	До запуска анимации, вид на топологию . . . . .	18
4.13	Выполнение моделирование, анимация . . . . .	19
4.14	Текст скрипта, измененный шаблон . . . . .	20
4.15	Кратчайший маршрут до разрыва соединения . . . . .	21
4.16	Момент разрыва соединения . . . . .	22
4.17	Резервный маршрут через узлы <code>n(6)</code> , <code>n(5)</code> и <code>n(4)</code> . . . . .	23
4.18	Сообщения в терминале о разрыве и восстановлении . . . . .	23
4.19	Топология сети . . . . .	24
4.20	Передача данных до разрыва соединения между 0 и 1 узлами . . . . .	25
4.21	Передача после разрыва соединения и обновления маршрута . . . . .	26
4.22	Передача данных после восстановления соединения между 0 и 1 узлами . . . . .	26
4.23	Сообщения в терминале . . . . .	27
4.24	Вид скрипта упражнения . . . . .	27

## Список таблиц

# 1 Цель работы

Целью данной работы является приобретение навыков моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также анализ полученных результатов моделирования.

## 2 Задание

1. Записать шаблон сценария для NS-2.
2. Создать простой пример описания топологии сети, состоящей из двух узлов и одного соединения.
3. Создать пример с усложнённой топологией сети.
4. Создать пример с кольцевой топологией сети.
5. Выполнить упражнение.

### 3 Теоретическое введение

Network Simulator (NS-2) — один из программных симуляторов моделирования процессов в компьютерных сетях. NS-2 позволяет описать топологию сети, конфигурацию источников и приёмников трафика, параметры соединений (полосу пропускания, задержку, вероятность потерь пакетов и т.д.) и множество других параметров моделируемой системы. Данные о динамике трафика, состоянии соединений и объектов сети, а также информация о работе протоколов фиксируются в генерируемом trace-файле.

NS-2 является объектно-ориентированным программным обеспечением. Его ядро реализовано на языке C++. В качестве интерпретатора используется язык скриптов (сценариев) OTcl (Object oriented Tool Command Language). NS-2 полностью поддерживает иерархию классов C++ и подобную иерархию классов интерпретатора OTcl.

Обе иерархии обладают идентичной структурой, т.е. существует однозначное соответствие между классом одной иерархии и таким же классом другой. Объединение для совместного функционирования C++ и OTcl производится при помощи TclCl (Classes Tcl). В случае, если необходимо реализовать какую-либо специфическую функцию, не реализованную в NS-2 на уровне ядра, для этого используется код на C++.

### 3.0.1 Процесс создания модели сети для NS-2 состоит из нескольких этапов:

1. Создание нового объекта класса Simulator, в котором содержатся методы, необходимые для дальнейшего описания модели (например, методы new и delete используются для создания и уничтожения объектов соответственно);
2. Описание топологии моделируемой сети с помощью трёх основных функциональных блоков: узлов (nodes), соединений (links) и агентов (agents);
3. Задание различных действий, характеризующих работу сети.

Для создания узла используется метод node. При этом каждому узлу автоматически присваивается уникальный адрес. Для построения однонаправленных и двунаправленных линий соединения узлов используют методы simplex-link и duplex-link соответственно. Важным объектом NS-2 являются агенты, которые могут рассматриваться как процессы и/или как транспортные единицы, работающие на узлах моделируемой сети. Агенты могут выступать в качестве источников трафика или приёмников, а также как динамические маршрутизирующие и протокольные модули. Агенты создаются с помощью методов общего класса Agent и являются объектами его подкласса, т.е. Agent/type, где type определяет тип конкретного объекта. Например, TCP-агент может быть создан с помощью команды: `set tcp [ new Agent/TCP ]` Для закрепления агента за конкретным узлом используется метод attach-agent. Каждому агенту присваивается уникальный адрес порта для заданного узла (аналогично портам tcp и udp). Чтобы за конкретным агентом закрепить источник, используют методы attach-source и attach-traffic. Например, можно прикрепить ftp или telnet источники к TCP-агенту. Есть агенты, которые генерируют свои собственные данные, например, CBR-агент (Constant Bit-Rate) — источник трафика с постоянной интенсивностью. Действия разных агентов могут быть назначены планировщиком событий (Event Scheduler) в определённые моменты времени (также в определённые моменты времени могут быть задействованы или отключены те или иные источники данных, запись ста-



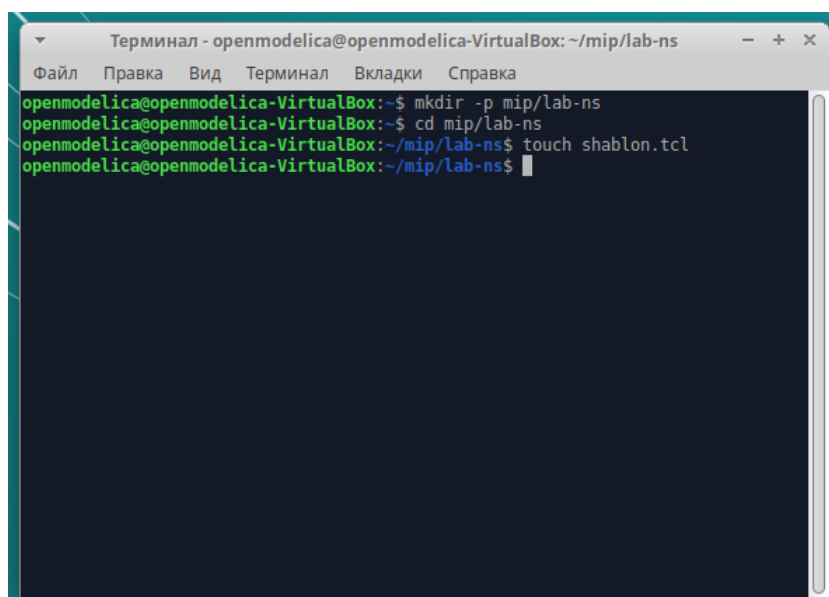
тики, разрыв, либо восстановление соединений, реконфигурация топологии и т.д.). Для этого может использоваться метод `at`. Моделирование начинается при помощи метода `run`. В качестве дополнения к NS-2 часто используют средство визуализации `nam` (network animator) для графического отображения свойств моделируемой системы и проходящего через неё трафика и пакет `Xgraph` для графического представления результатов моделирования. Запуск сценария NS-2 осуществляется в командной строке с помощью команды: `ns [tclscript]`

Здесь `[tclscript]` — имя файла скрипта Tcl, который определяет сценарий моделирования (т.е. топологию и различные события). `Nam` можно запустить с помощью команды `nam [nam-file]`

Здесь `[nam-file]` — имя `nam trace`-файла, сгенерированного с помощью `ns`.

## 4 Выполнение лабораторной работы

1. Я создала в рабочем каталоге директорию `mip`, к которой будут выполняться лабораторные работы. Внутри `mip` создала директорию `lab-ns`, а в ней файл `shablon.tcl`. Открыла файл шаблона на редактирование и заполнила его согласно коду из задания лабораторной. После этого запустила шаблон и посмотрела на результат. Пустой экран приложения, где, редактируя шаблон, в будущем мы будем добавлять сети определенных топологий. (рис. fig. 4.1, fig. 4.2, fig. 4.3).



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
Файл  Правка  Вид  Терминал  Вкладки  Справка
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$
```

Рис. 4.1: Создание файлов

```

/home/openmodelica/mip/lab-ns/shablon.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

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

# все регистрируемые 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 # объявление глобальных переменных
    # запуск nam в фоновом режиме
    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. 4.2: Заполнение файла шаблона

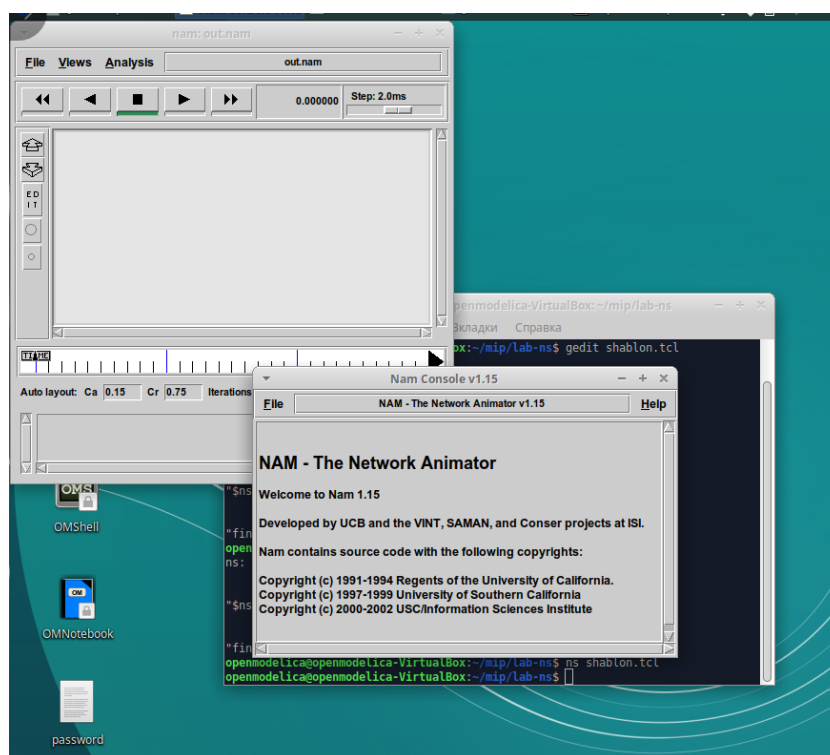
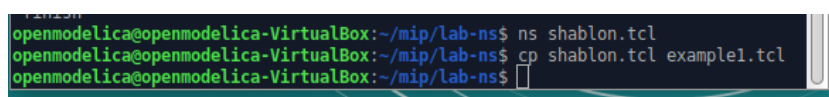


Рис. 4.3: Результат выполнения файла шаблона, приложение

2. Я выполнила по образцу из задания лабораторной простой пример описа-

ния топологии сети, состоящей из двух узлов и одного соединения. В начале скопировала содержимое шаблона в новый файл, после отредактировала его. Добавила описание топологии сети, создала агенты для генерации и приема трафика, создала агент UDP и присоединила к узлу n0. В узле агент сам не может генерировать трафик, он лишь реализует протоколы и алгоритмы транспортного уровня. Поэтому к агенту присоединяется приложение. В данном случае — это источник с постоянной скоростью (Constant Bit Rate, CBR), который каждые 5 мс посылает пакет  $R = 500$  байт. (рис. fig. 4.4, fig. 4.5, fig. 4.6, fig. 4.7, fig. 4.8).

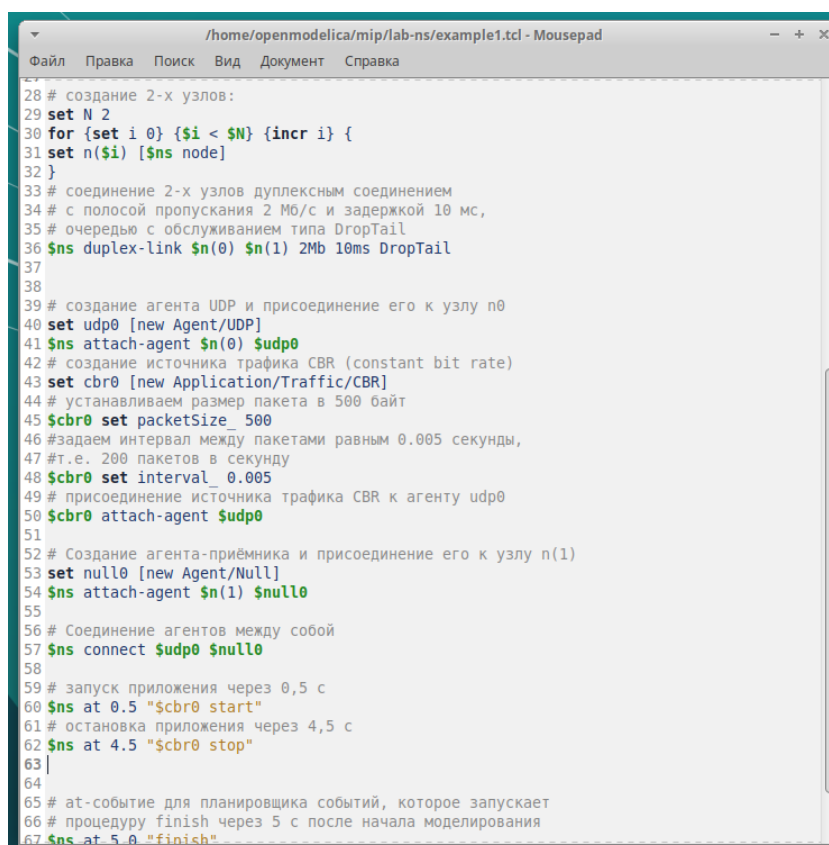


```

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

```

Рис. 4.4: Копирование шаблона



```

28 # создание 2-х узлов:
29 set N 2
30 for {set i 0} {$i < $N} {incr i} {
31   set n($i) [$ns node]
32 }
33 # соединение 2-х узлов дуплексным соединением
34 # с полосой пропускания 2 Мб/с и задержкой 10 мс,
35 # очередь с обслуживанием типа DropTail
36 $ns duplex-link $n(0) $n(1) 2Mb 10ms DropTail
37
38
39 # создание агента UDP и присоединение его к узлу n0
40 set udp0 [new Agent/UDP]
41 $ns attach-agent $n(0) $udp0
42 # создание источника трафика CBR (constant bit rate)
43 set cbr0 [new Application/Traffic/CBR]
44 # устанавливаем размер пакета в 500 байт
45 $cbr0 set packetSize_ 500
46 # задаем интервал между пакетами равным 0.005 секунды,
47 # т.е. 200 пакетов в секунду
48 $cbr0 set interval_ 0.005
49 # присоединение источника трафика CBR к агенту udp0
50 $cbr0 attach-agent $udp0
51
52 # Создание агента-приёмника и присоединение его к узлу n(1)
53 set null0 [new Agent/Null]
54 $ns attach-agent $n(1) $null0
55
56 # Соединение агентов между собой
57 $ns connect $udp0 $null0
58
59 # запуск приложения через 0,5 с
60 $ns at 0.5 "$cbr0 start"
61 # остановка приложения через 4,5 с
62 $ns at 4.5 "$cbr0 stop"
63 |
64
65 # at-событие для планировщика событий, которое запускает
66 # процедуру finish через 5 с после начала моделирования
67 $ns at 5.0 "finish"

```

Рис. 4.5: Код программы

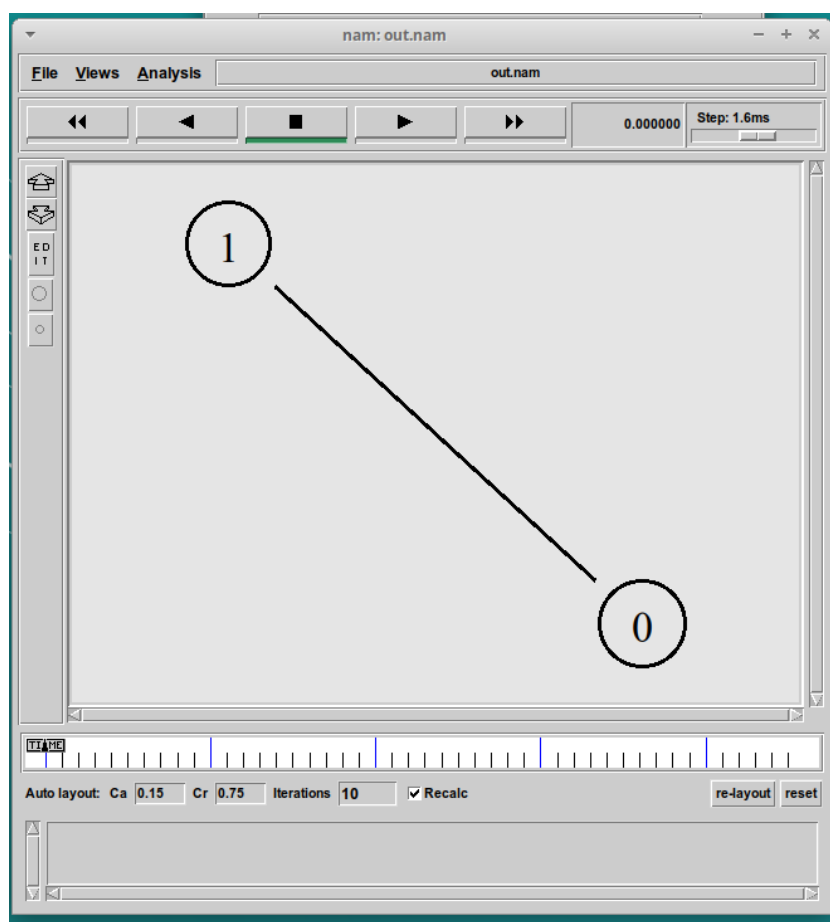


Рис. 4.6: Аниматор nam

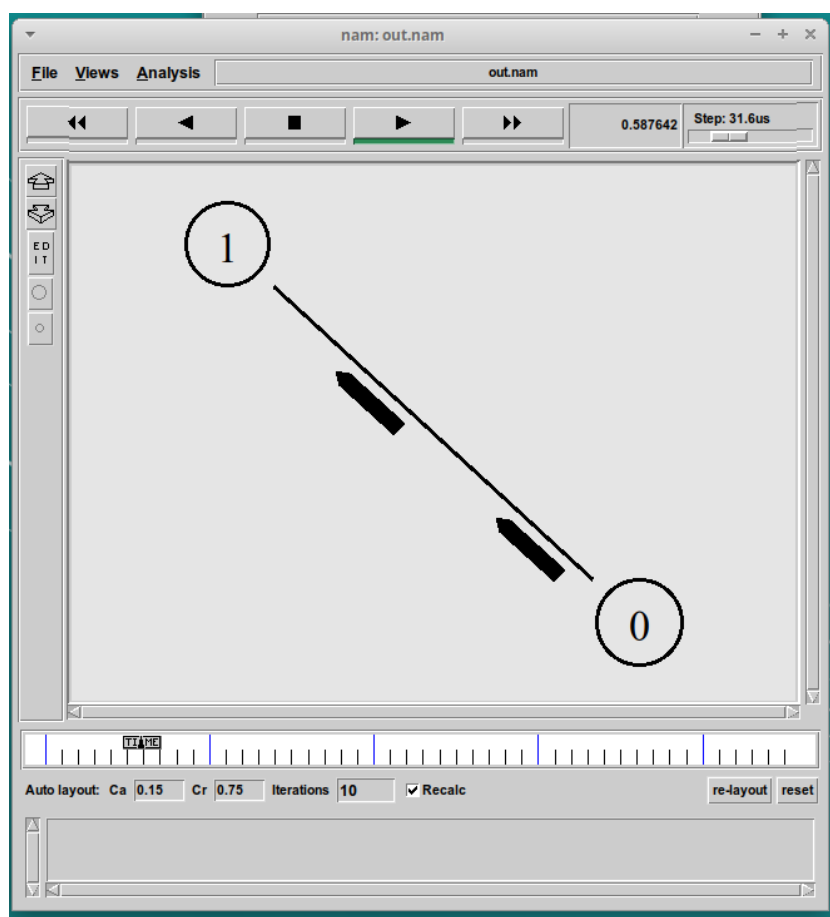


Рис. 4.7: Визуализация простой модели сети с помощью nam

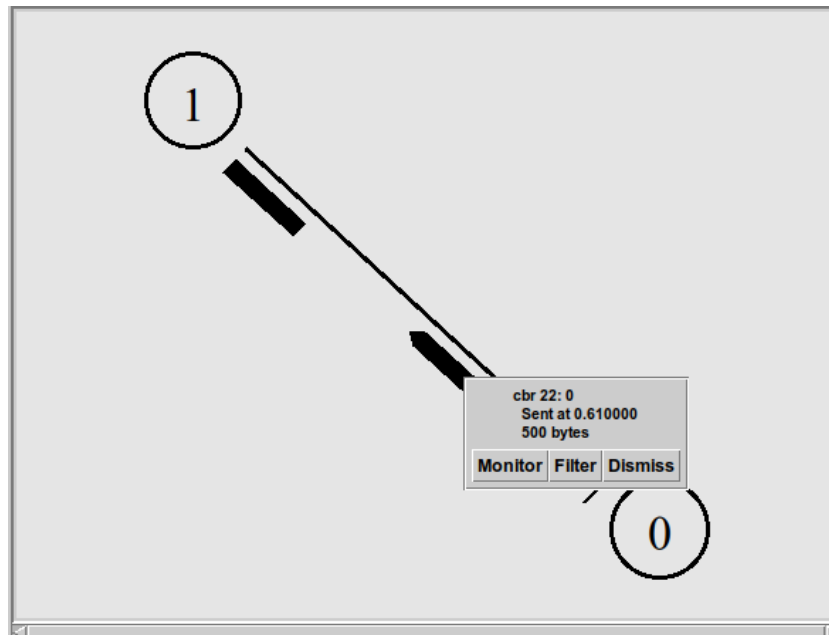


Рис. 4.8: Можно осуществлять наблюдение за отдельным пакетом, щёлкнув по нему в окне `pan`

3. Пример с усложнённой топологией сети. Точно также скопировала шаблон и заполнила его, прочитав постановку задачи. Обнаружила неточности в коде, который предлагается в качестве примера в лабораторной работе, исправила их.

- Задание: “– между узлами `n2` и `n3` установлено дуплексное соединение с пропускной способностью 1,7 Мбит/с и задержкой 20 мс;”
- Код: `$ns duplex-link $n(3) $n(2) 2Mb 10ms DropTail.`
- Заменяла 2 на 1.7 и 10 на 20.

Далее: - Задание: “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.” - Код: `$ns at 0.5 "$cbr0 start".` - Заменяла 0.5 на 0.1.

Сохранив изменения в отредактированном файле и запустив симулятор, получила анимированный результат моделирования. (рис. fig. 4.9, fig. 4.10, fig. 4.11, fig. 4.12, fig. 4.13).

Создадим 4 узла и 3 дуплексных соединения с указанием направления:

```
set N 4
for {set i 0} {$i < $N} {incr i} {
    set n($i) [$ns node]
}

$ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(3) $n(2) 2Mb 10ms DropTail

$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right
```

Рис. 4.9: Кадр с неточностью

```

/home/openmodelica/mip/lab-ns/example2.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка
27
28 set N 4
29 for {set i 0} {$i < $N} {incr i} {
30     set n($i) [$ns node]
31 }
32
33 $ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
34 $ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
35 $ns duplex-link $n(3) $n(2) 1.7Mb 20ms DropTail
36
37 $ns duplex-link-op $n(0) $n(2) orient right-down
38 $ns duplex-link-op $n(1) $n(2) orient right-up
39 $ns duplex-link-op $n(2) $n(3) orient right
40
41 # создание агента UDP и присоединение его к узлу n(0)
42 set udp0 [new Agent/UDP]
43 $ns attach-agent $n(0) $udp0
44
45 # создание источника CBR-трафика
46 # и присоединение его к агенту udp0
47 set cbr0 [new Application/Traffic/CBR]
48 $cbr0 set packetSize_ 500
49 $cbr0 set interval_ 0.005
50 $cbr0 attach-agent $udp0
51
52 # создание агента TCP и присоединение его к узлу n(1)
53 set tcp1 [new Agent/TCP]
54 $ns attach-agent $n(1) $tcp1
55
56 # создание приложения FTP
57 # и присоединение его к агенту tcp1
58 set ftp [new Application/FTP]
59 $ftp attach-agent $tcp1
60
61 # создание агента-получателя для udp0
62 set null0 [new Agent/Null]
63 $ns attach-agent $n(3) $null0
64 # создание агента-получателя для tcp1
65 set sink1 [new Agent/TCPSink]
66 $ns attach-agent $n(3) $sink1

```

Рис. 4.10: Код программы



```
/home/openmodelica/mip/lab-ns/example2.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

51
52 # создание агента TCP и присоединение его к узлу n(1)
53 set tcp1 [new Agent/TCP]
54 $ns attach-agent $n(1) $tcp1
55
56 # создание приложения FTP
57 # и присоединение его к агенту tcp1
58 set ftp [new Application/FTP]
59 $ftp attach-agent $tcp1
60
61 # создание агента-получателя для udp0
62 set null0 [new Agent/Null]
63 $ns attach-agent $n(3) $null0
64 # создание агента-получателя для tcp1
65 set sink1 [new Agent/TCPSink]
66 $ns attach-agent $n(3) $sink1
67
68 $ns connect $udp0 $null0
69 $ns connect $tcp1 $sink1
70
71 $ns color 1 Blue
72 $ns color 2 Red
73 $udp0 set class_ 1
74 $tcp1 set class_ 2
75
76 $ns duplex-link-op $n(2) $n(3) queuePos 0.5
77 $ns queue-limit $n(2) $n(3) 20
78
79 $ns at 0.1 "$cbr0 start"
80 $ns at 1.0 "$ftp start"
81 $ns at 4.0 "$ftp stop"
82 $ns at 4.5 "$cbr0 stop"
83
84 # at-событие для планировщика событий, которое запускает
85 # процедуру finish через 5 с после начала моделирования
86 $ns at 5.0 "finish"
87
88 # запуск модели
89 $ns run
90
```

Рис. 4.11: Продолжение кода

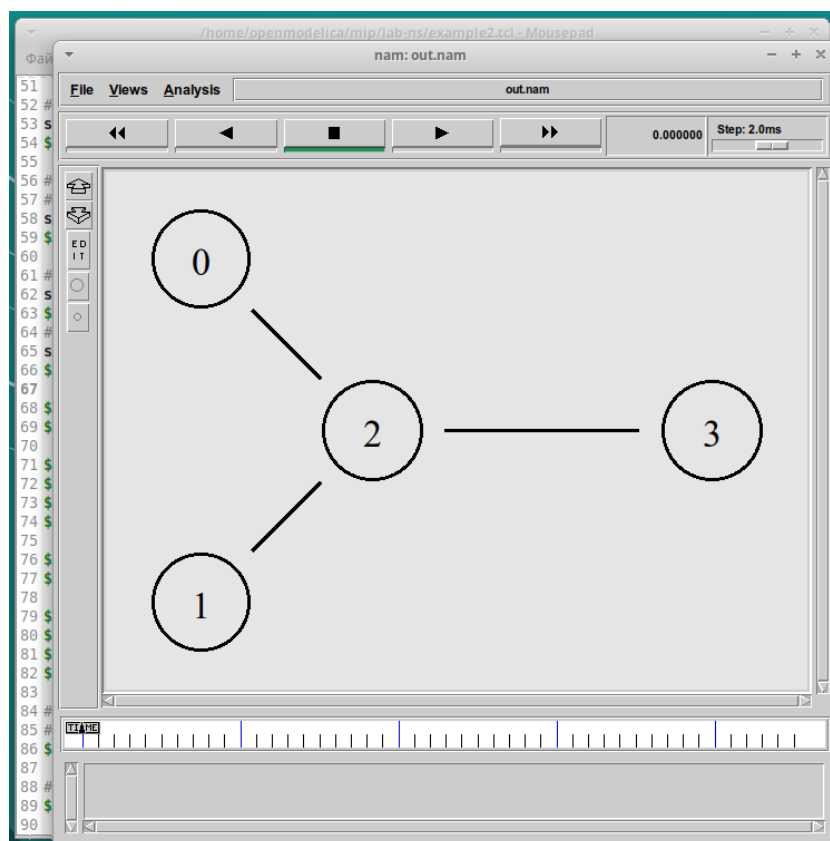


Рис. 4.12: До запуска анимации, вид на топологию

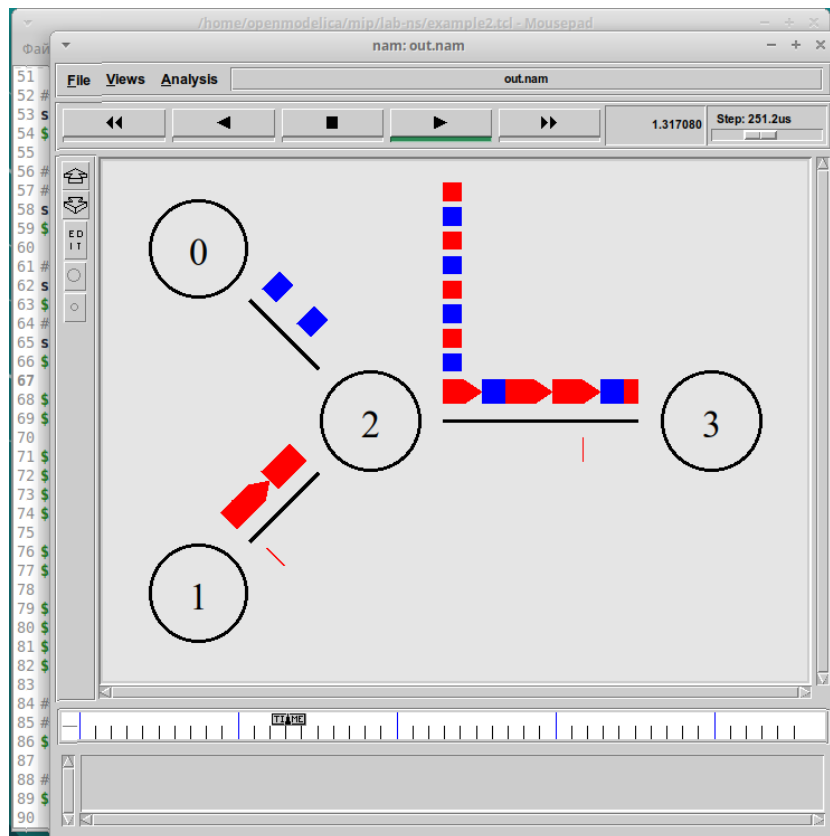


Рис. 4.13: Выполнение моделирование, анимация

4. Пример с кольцевой топологией сети. Я опять переписала код, в данном примере данные должны передаваться по кратчайшему маршруту, а при разрыве соединения информация о топологии должна обновляться и пакеты отсылаться по новому маршруту. После восстановления соединения снова будет выбран самый короткий маршрут. (рис. fig. 4.14, fig. 4.15, fig. 4.16, fig. 4.17, fig. 4.18).

```

/home/openmodelica/mip/lab-ns/example3.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка
29
30 set N 7
31 for {set i 0} {$i < $N} {incr i} {
32   set n($i) [$ns node]
33 }
34
35 for {set i 0} {$i < $N} {incr i} {
36   $ns duplex-link $n($i) $n([expr ($i+1)%$N]) 1Mb 10ms DropTail
37 }
38
39 set udp0 [new Agent/UDP]
40 $ns attach-agent $n(0) $udp0
41 set cbr0 [new Agent/CBR]
42 $ns attach-agent $n(0) $cbr0
43 $cbr0 set packetSize_ 500
44 $cbr0 set interval_ 0.005
45
46 set null0 [new Agent/Null]
47 $ns attach-agent $n(3) $null0
48
49 $ns connect $cbr0 $null0
50
51
52 $ns at 0.5 "$cbr0 start"
53 $ns rtmodel-at 1.0 down $n(1) $n(2)
54 $ns rtmodel-at 2.0 up $n(1) $n(2)
55 $ns at 4.5 "$cbr0 stop"
56 |
57
58 # at-событие для планировщика событий, которое запускает
59 # процедуру finish через 5 с после начала моделирования
60 $ns at 5.0 "finish"
61
62 # запуск модели
63 $ns run
64

```

Рис. 4.14: Текст скрипта, измененный шаблон

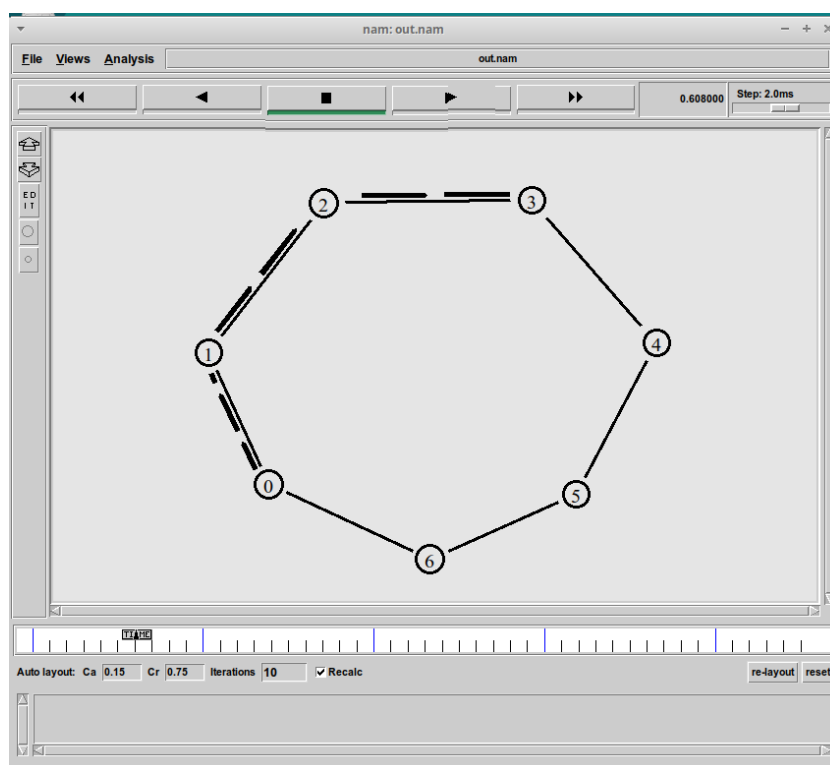


Рис. 4.15: Кратчайший маршрут до разрыва соединения

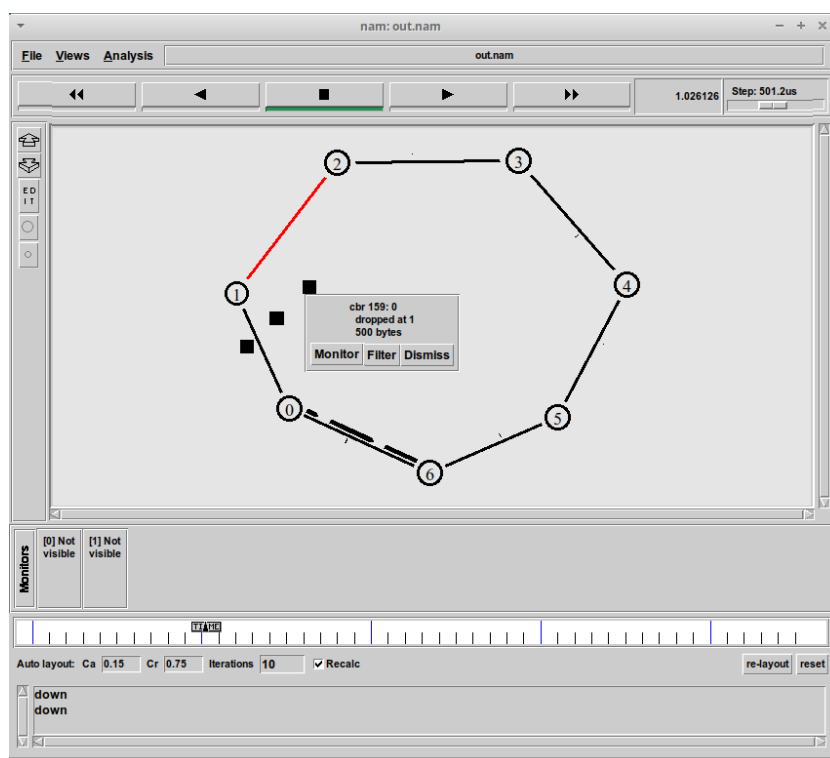


Рис. 4.16: Момент разрыва соединения

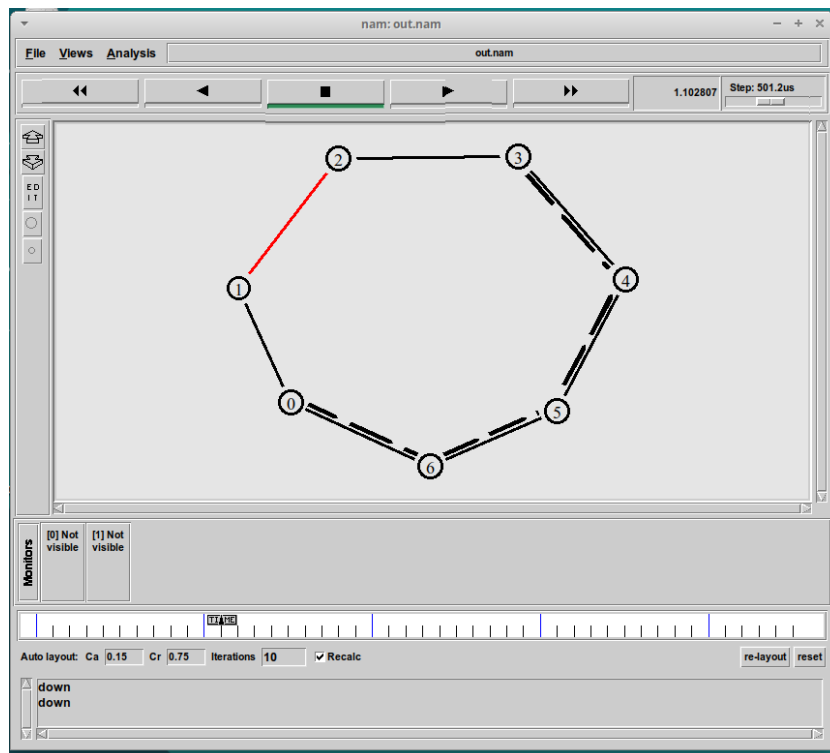


Рис. 4.17: Резервный маршрут через узлы n(6), n(5) и n(4)

```

Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
Файл  Правка  Вид  Терминал  Вкладки  Справка
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 2 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 1 2
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 1 2
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 2 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 2 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 1 2
Please use this format in the future.

```

Рис. 4.18: Сообщения в терминале о разрыве и восстановлении

5. Упражнение. Я внесла следующие изменения в реализацию примера с кольцевой топологией сети:

- топология сети должна соответствовать представленной в лабораторной.
- 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. (рис. fig. 4.19, fig. 4.20, fig. 4.21, fig. 4.22, fig. 4.23, fig. 4.24).

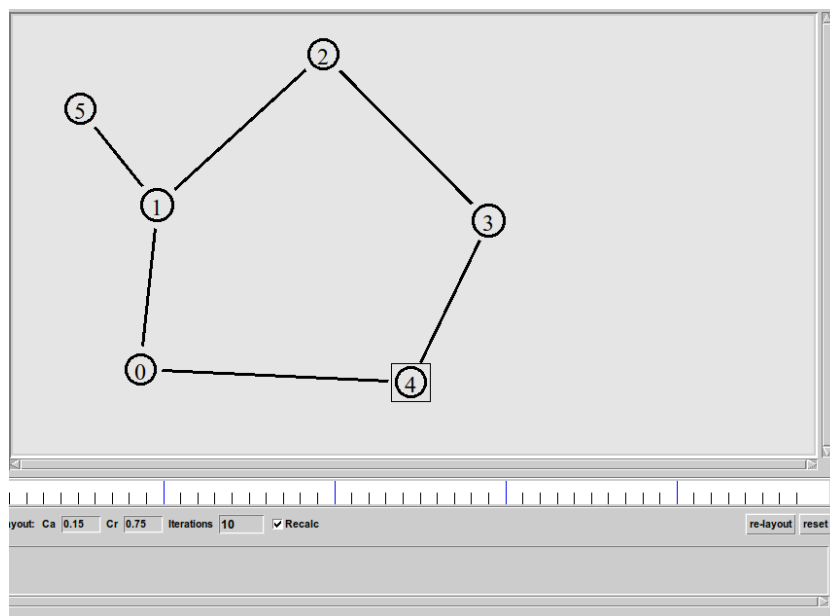


Рис. 4.19: Топология сети



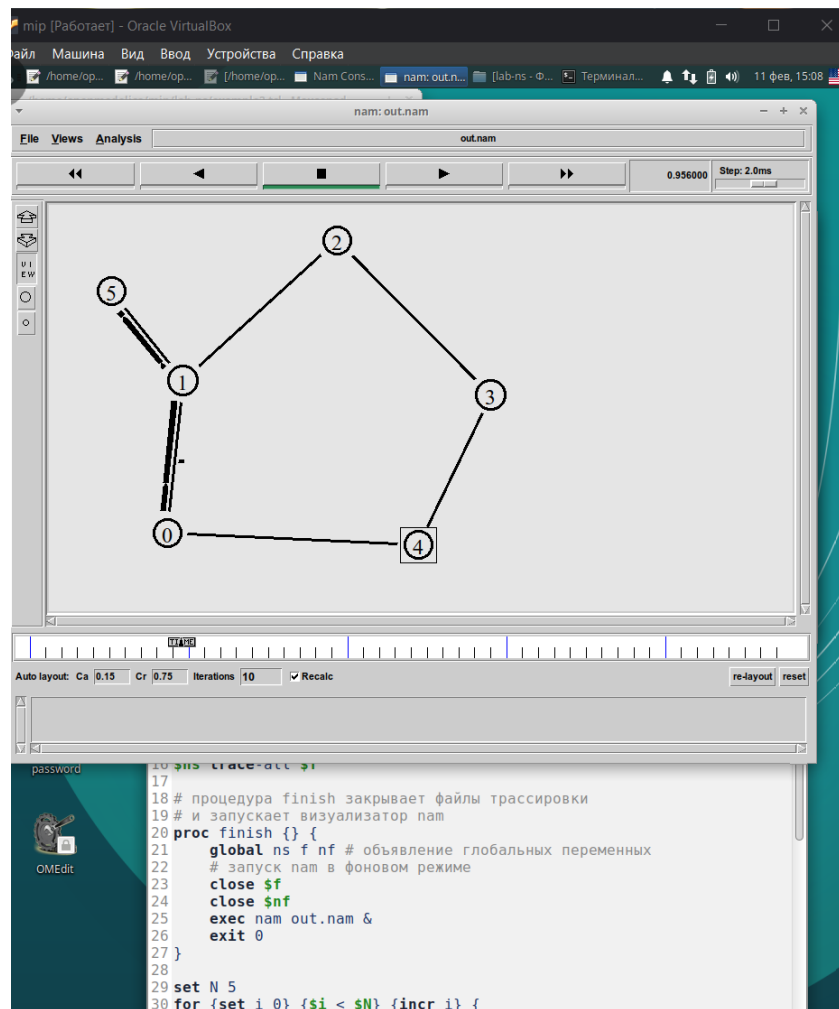


Рис. 4.20: Передача данных до разрыва соединения между 0 и 1 узлами

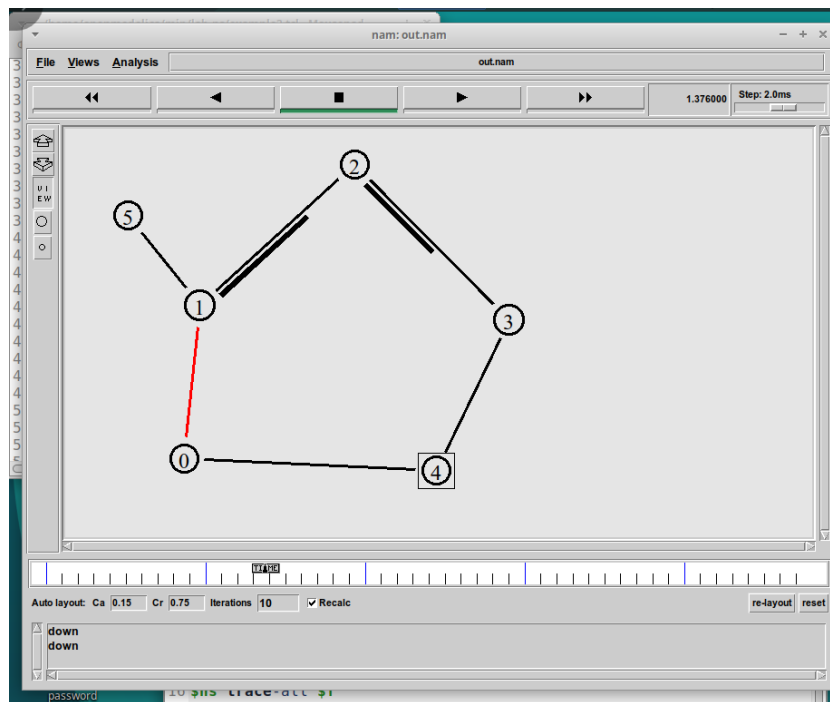


Рис. 4.21: Передача после разрыва соединения и обновления маршрута

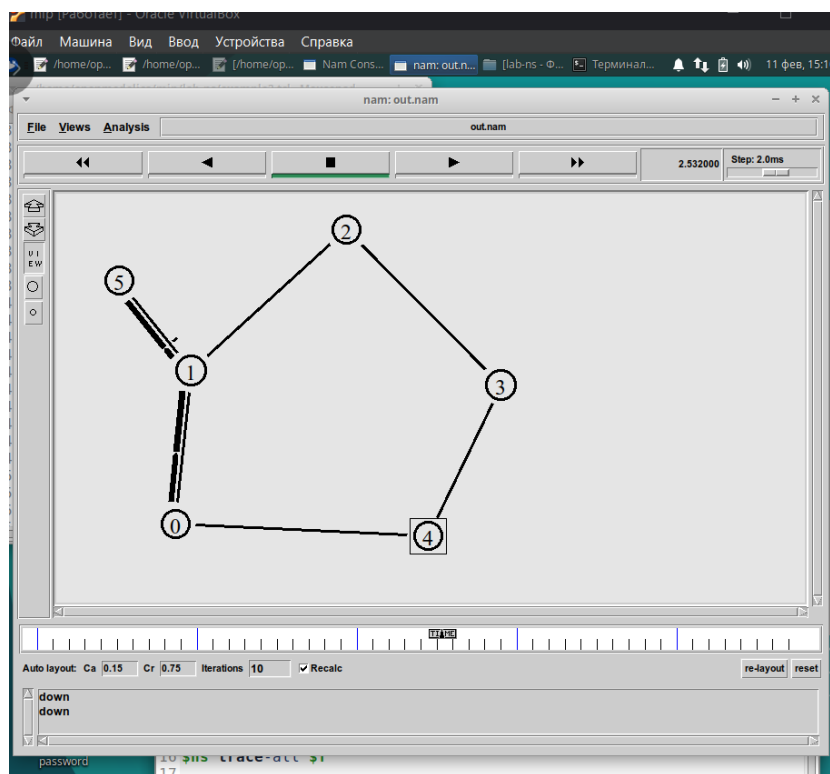


Рис. 4.22: Передача данных после восстановления соединения между 0 и 1 узлами

```

Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 0 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 1 link-down 1 0 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 1 0
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 1 0
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 0 1
Please use this format in the future.
v -t <time> -e <tcl expression>

Nam syntax has changed: v -t 2 link-up 2 0 1
Please use this format in the future.
v -t <time> -e <tcl expression>

```

Рис. 4.23: Сообщения в терминале

```

/home/openmodelica/mip/lab-ns/exerlab1.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка
20 exit 0
21
27 }
28
29 set N 5
30 for {set i 0} {$i < $N} {incr i} {
31 set n($i) [$ns node]
32 }
33
34 for {set i 0} {$i < $N} {incr i} {
35 $ns duplex-link $n($i) $n([expr ($i+1)%$N]) 1Mb 10ms DropTail
36 }
37
38 set n(5) [$ns node]
39 $ns duplex-link $n(5) $n(1) 1Mb 10ms DropTail
40
41 set tcp1 [new Agent/TCP]
42 $ns attach-agent $n(0) $tcp1
43
44 set ftp [new Application/FTP]
45 $ftp attach-agent $tcp1
46
47 set sink1 [new Agent/TCPSink/DelAck]
48 $ns attach-agent $n(5) $sink1
49
50 $ns connect $tcp1 $sink1
51
52
53 $ns at 0.5 "$ftp start"
54 $ns rtmodel-at 1.0 down $n(0) $n(1)
55 $ns rtmodel-at 2.0 up $n(0) $n(1)
56 $ns at 4.5 "$ftp stop"
57
58 # at-событие для планировщика событий, которое запускает
59 # процедуру finish через 5 с после начала моделирования
60 $ns at 5.0 "finish"
61
62

```

Рис. 4.24: Вид скрипта упражнения

## 6. Листинг упражнения:

```

# создание объекта Simulator
set ns [new Simulator]
$ns rtproto DV

# 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
# для регистрации всех событий
set f [open out.tr w]

# все регистрируемые 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 # объявление глобальных переменных
    # запуск nam в фоновом режиме
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

set N 5

```

```

for {set i 0} {$i < $N} {incr i} {
set n($i) [$ns node]
}

for {set i 0} {$i < $N} {incr i} {
$ns duplex-link $n($i) $n([expr ($i+1)%$N]) 1Mb 10ms DropTail
}

set n(5) [$ns node]
$ns duplex-link $n(5) $n(1) 1Mb 10ms DropTail

set tcp1 [new Agent/TCP]
$ns attach-agent $n(0) $tcp1

set ftp [new Application/FTP]
$ftp attach-agent $tcp1

set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(5) $sink1

$ns connect $tcp1 $sink1

$ns at 0.5 "$ftp start"
$ns rtmodel-at 1.0 down $n(0) $n(1)
$ns rtmodel-at 2.0 up $n(0) $n(1)
$ns at 4.5 "$ftp stop"

# at-событие для планировщика событий, которое запускает

```

```
# процедуру finish через 5 с после начала моделирования  
$ns at 5.0 "finish"
```

```
# запуск модели  
$ns run
```

## **5 Выводы**

В ходе данной работы я приобрела практические навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также выполнила анализ полученных результатов моделирования.

# Список литературы

1. Руководство к лабораторной работе