

# **Лабораторная работа №1**

**Простые модели компьютерной сети**

Оширова Юлия Николаевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.0.1	Шаблон сценария для NS-2 . . . . .	7
3.0.2	Простой пример описания топологии сети, состоящей из двух узлов и одного соединения . . . . .	9
3.0.3	Пример с усложнённой топологией сети . . . . .	11
3.0.4	Пример с кольцевой топологией сети . . . . .	14
3.0.5	Упражнение . . . . .	17
<b>4</b>	<b>Выводы</b>	<b>22</b>

# Список иллюстраций

3.1	Шаблон для сценария . . . . .	7
3.2	Добавление кода . . . . .	8
3.3	Область моделирования . . . . .	9
3.4	Добавление кода . . . . .	10
3.5	Область моделирования . . . . .	11
3.6	Добавление кода . . . . .	13
3.7	Область моделирования . . . . .	14
3.8	Добавление кода . . . . .	15
3.9	Область моделирования . . . . .	16
3.10	Область моделирования . . . . .	16
3.11	Область моделирования . . . . .	17
3.12	Добавление кода . . . . .	18
3.13	Область моделирования . . . . .	19
3.14	Область моделирования . . . . .	20
3.15	Область моделирования . . . . .	21

## **Список таблиц**

# 1 Цель работы

Приобрести навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также проанализировать полученные результаты моделирования.

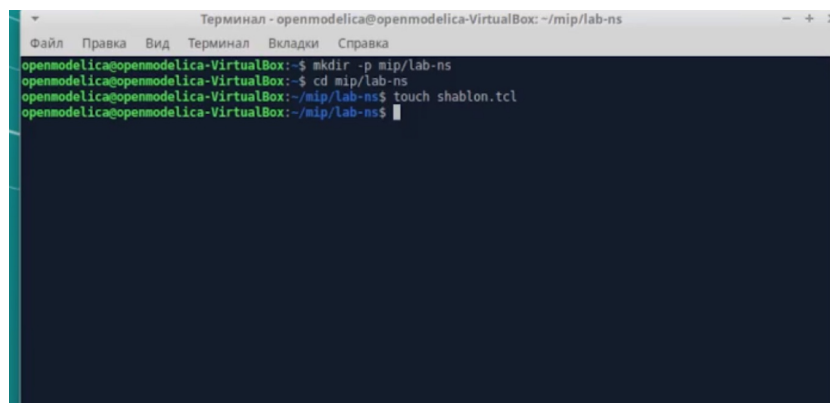
## 2 Задание

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

## 3 Выполнение лабораторной работы

### 3.0.1 Шаблон сценария для NS-2

В своём рабочем каталоге создадим директорию `mip`, в которой будут выполняться лабораторные работы. Внутри `mip` создадим директорию `lab-ns`, а в ней файл `shablon.tcl` (рис. 3.1).



```
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. 3.1: Шаблон для сценария

Откроем на редактирование файл `shablon.tcl` (рис. 3.2).

Сначала создадим объект типа `Simulator`. Затем создадим переменную `nf` и укажем, что требуется открыть на запись `nam`-файл для регистрации выходных результатов моделирования. Вторая строка даёт команду симулятору записывать все данные о динамике модели в файл `out.nam`. Далее создадим переменную `f` и откроем на запись файл трассировки для регистрации всех событий модели. После этого добавим процедуру `finish`, которая закрывает файлы трассировки и запускает `nam`. С помощью команды `at` указываем планировщику событий, что

процедуру `finish` запустим через 5 с после начала моделирования, после чего запустим симулятор `ns`.

```
# создание объекта Simulator
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
# 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
    $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. 3.2: Добавление кода

Сохранив изменения в отредактированном файле `shablon.tcl` и закрыв его, запустим симулятор командой `ns shablon.tcl`. Увидим пустую область моделирования, поскольку ещё не определены никакие объекты и действия (рис. 3.3).



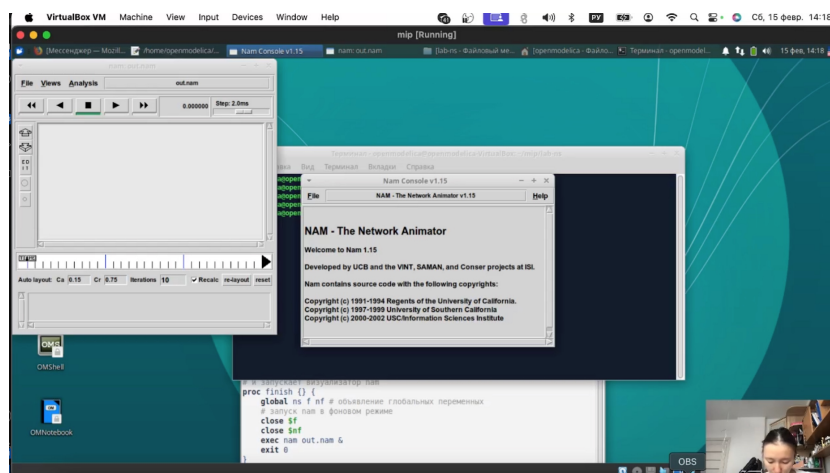


Рис. 3.3: Область моделирования

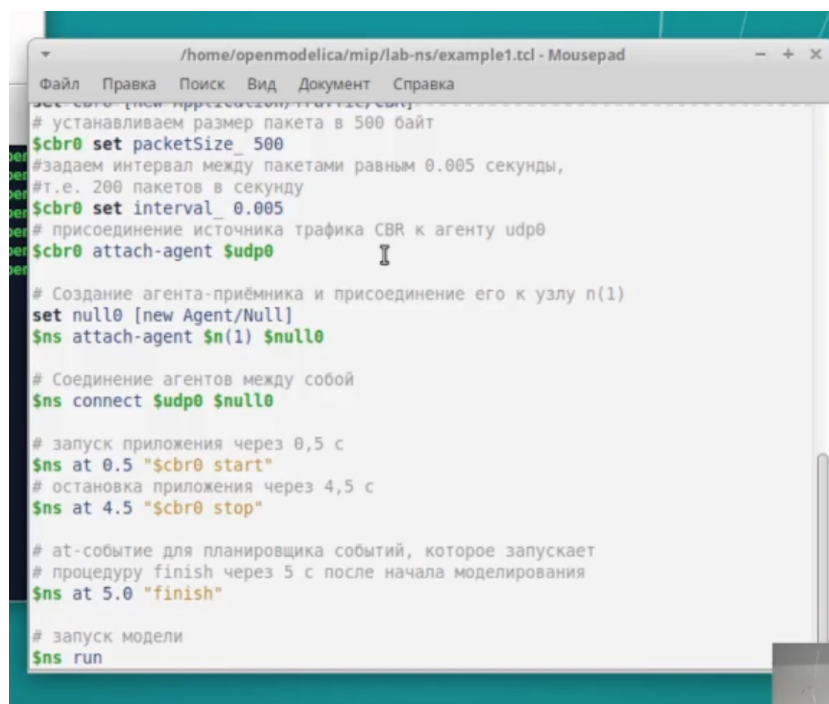
### 3.0.2 Простой пример описания топологии сети, состоящей из двух узлов и одного соединения

Требуется смоделировать сеть передачи данных, состоящую из двух узлов, соединённых дуплексной линией связи с полосой пропускания 2 Мб/с и задержкой 10 мс, очередь 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.

Скопируем содержимое созданного шаблона в новый файл: `cp shablon.tcl example1.tcl` и откроем `example1.tcl` на редактирование. Добавим в него до строки `$ns at 5.0 "finish"` описание топологии сети. Создадим агенты для генерации и приёма трафика. Создается агент UDP и присоединяется 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. Таким образом, скорость источника: 80000 бит/с.

Далее создадим Null-агент, который работает как приёмник трафика, и прикрепим его к узлу `n1`. Соединим агенты между собой. Для запуска и остановки

приложения CBR добавляются at-события в планировщик событий (перед командой \$ns at 5.0 "finish") (рис. 3.4).



```

/home/openmodelica/mip/lab-ns/example1.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка
# устанавливаем размер пакета в 500 байт
$scbr0 set packetSize 500
# задаем интервал между пакетами равным 0.005 секунды,
# т.е. 200 пакетов в секунду
$scbr0 set interval 0.005
# присоединение источника трафика CBR к агенту udp0
$scbr0 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
$ns at 0.5 "$scbr0 start"
# остановка приложения через 4,5 с
$ns at 4.5 "$scbr0 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

```

Рис. 3.4: Добавление кода

Сохранив изменения в отредактированном файле и запустив симулятор, получим в качестве результата запуск аниматора nam в фоновом режиме (рис. 3.5).

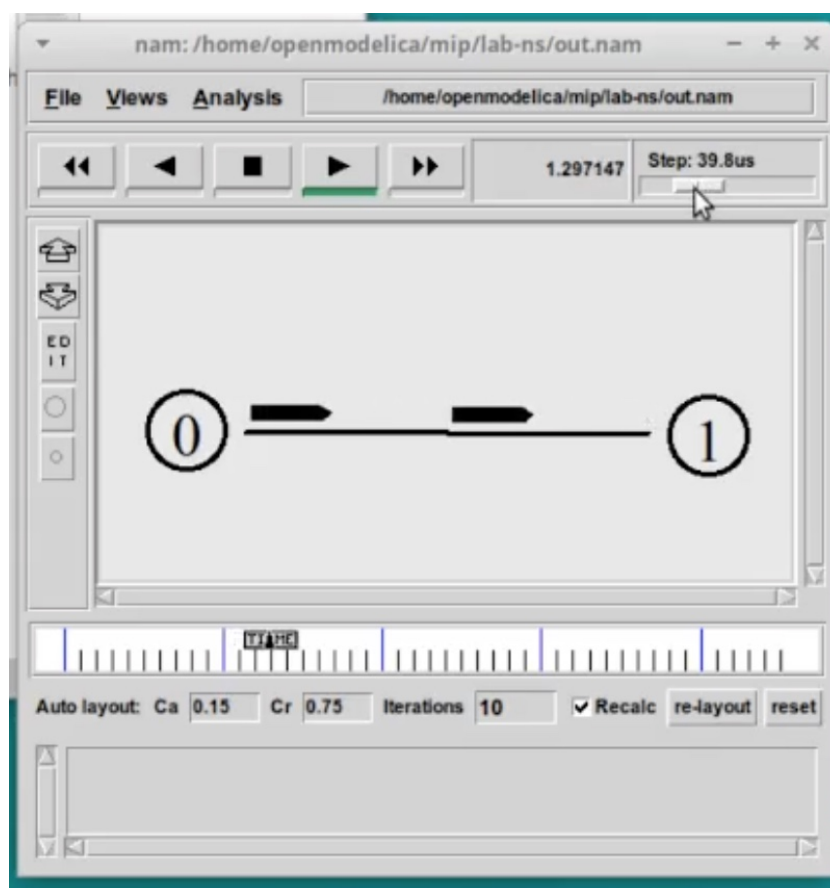


Рис. 3.5: Область моделирования

При нажатии на кнопку play в окне nam через 0.5 секунды из узла 0 данные начнут поступать к узлу 1.

### 3.0.3 Пример с усложнённой топологией сети

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 n3 (по-умолчанию, максимальный размер пакета, который TCP-агент может генерировать, равняется 1KByte) 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.

Скопируем содержимое созданного шаблона в новый файл: cp shablon.tcl example2.tcl и откроем example2.tcl на редактирование. Далее создадим 4 узла и 3 дуплексных соединения с указанием направления. Создадим агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP. Создадим агенты-получатели. Соединим агенты udp0 и tcp1 и их получателей. Зададим описание цвета каждого потока. Выполним отслеживание событий в очереди и наложение ограничения на размер очереди. Добавим at-события(рис. 3.6).

```

# создание приложения FTP
# и присоединение его к агенту tcp1
set ftp [new Application/FTP]
$ftp attach-agent $tcp1

# создание агента-получателя для udp0
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
# создание агента-получателя для tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n(3) $sink1

$ns connect $udp0 $null0
$ns connect $tcp1 $sink1

$ns color 1 Blue
$ns color 2 Red
$udp0 set class_ 1
$tcp1 set class_ 2

$ns duplex-link-op $n(2) $n(3) queuePos 0.5

$ns queue-limit $n(2) $n(3) 20

$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr0 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

```

Рис. 3.6: Добавление кода

Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования (рис. 3.7).

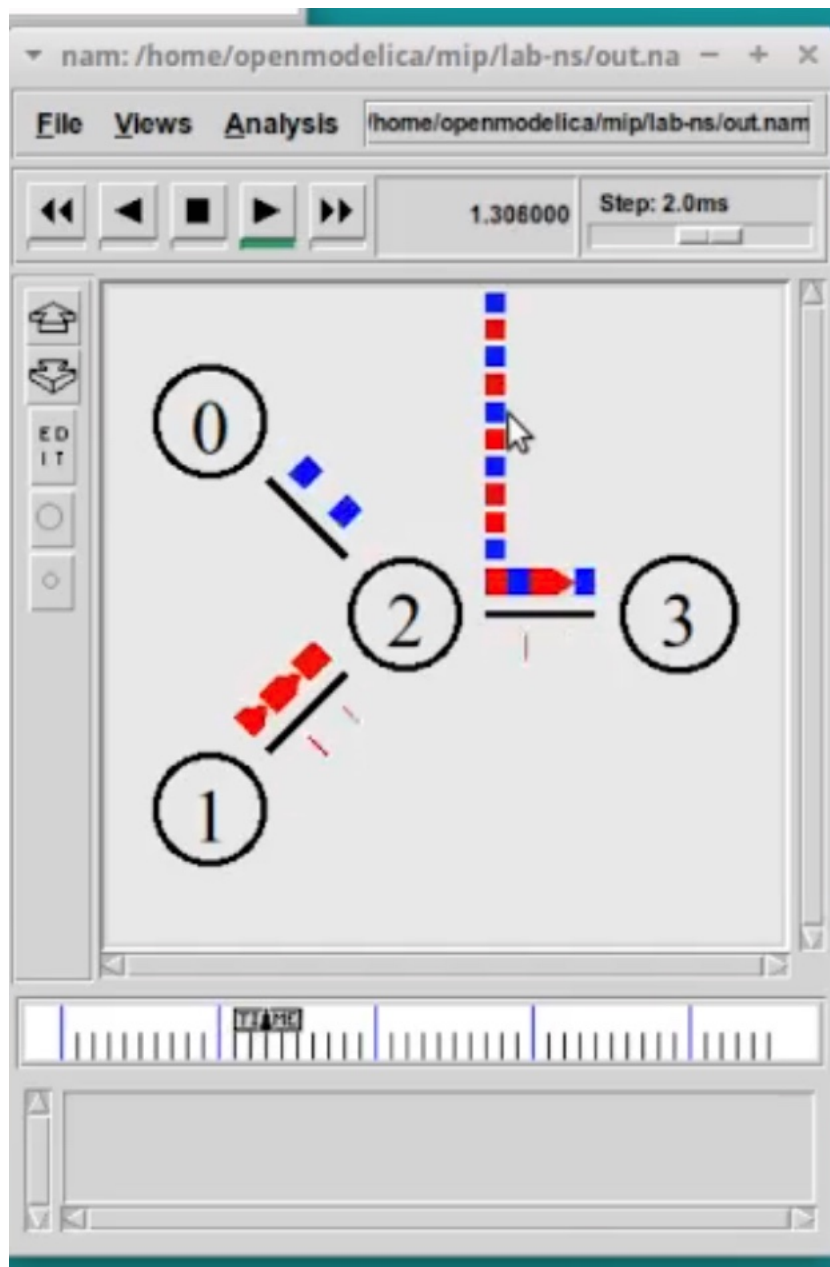


Рис. 3.7: Область моделирования

### 3.0.4 Пример с кольцевой топологией сети

Описание модели 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. Скопируем содержимое созданного шаблона в новый файл: `cp shablon.tcl example3.tcl` и откроем `example3.tcl` на редактирование. Опишем топологию моделируемой сети (рис. 3.8). Далее соединим узлы так, чтобы создать круговую топологию. Каждый узел, за исключением последнего, соединяется со следующим, последний соединяется с первым. Для этого в цикле использован оператор `%`, означающий остаток от деления нацело. Зададим передачу данных от узла  $n(0)$  к узлу  $n(3)$ . Данные передаются по кратчайшему маршруту от узла  $n(0)$  к узлу  $n(3)$ , через узлы  $n(1)$  и  $n(2)$  (рис. 3.9). Добавим команду разрыва соединения между узлами  $n(1)$  и  $n(2)$  на время в одну секунду, а также время начала и окончания передачи данных.

```

/home/openmodelica/mip/lab-ns/example3.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

exit 0
}

set N 7
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 udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Agent/CBR]
$ns attach-agent $n(0) $cbr0
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0
$ns connect $cbr0 $null0

$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"

# 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. 3.8: Добавление кода

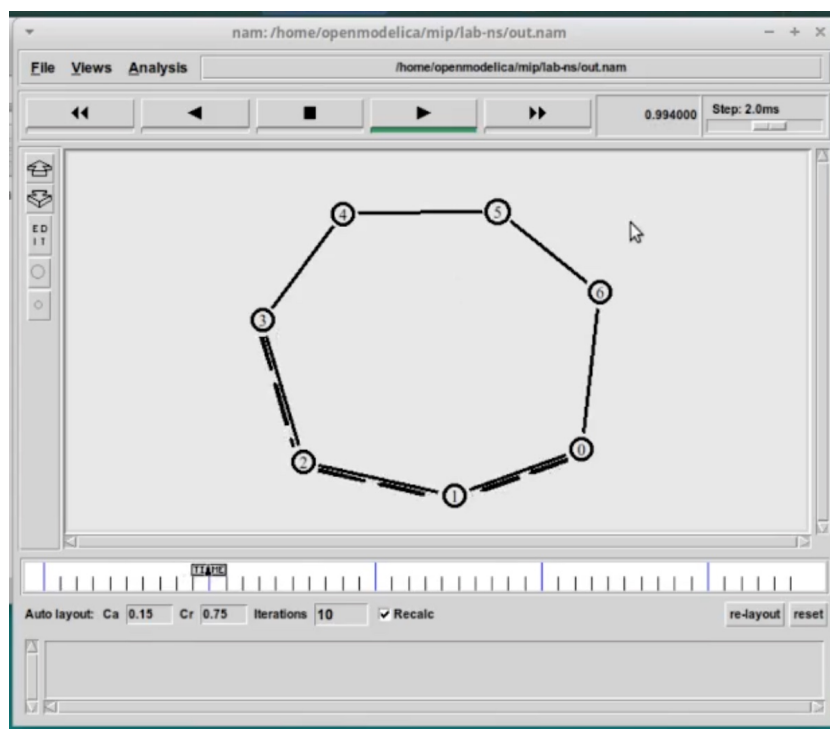


Рис. 3.9: Область моделирования

Передача данных при кольцевой топологии сети в случае разрыва соединения представлена на рис. 3.10.

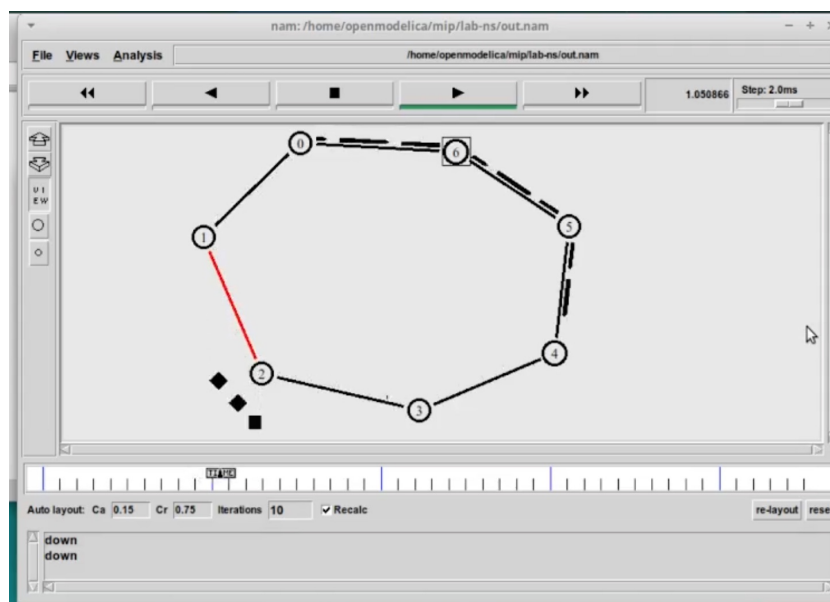


Рис. 3.10: Область моделирования



Добавив в начало скрипта после команды создания объекта Simulator:

```
$ns rtproto DV
```

увидим, что сразу после запуска в сети отправляется небольшое количество маленьких пакетов, используемых для обмена информацией, необходимой для маршрутизации между узлами (рис. 3.11). Когда соединение будет разорвано, информация о топологии будет обновлена, и пакеты будут отсылаться по новому маршруту через узлы  $n(6)$ ,  $n(5)$  и  $n(4)$ .

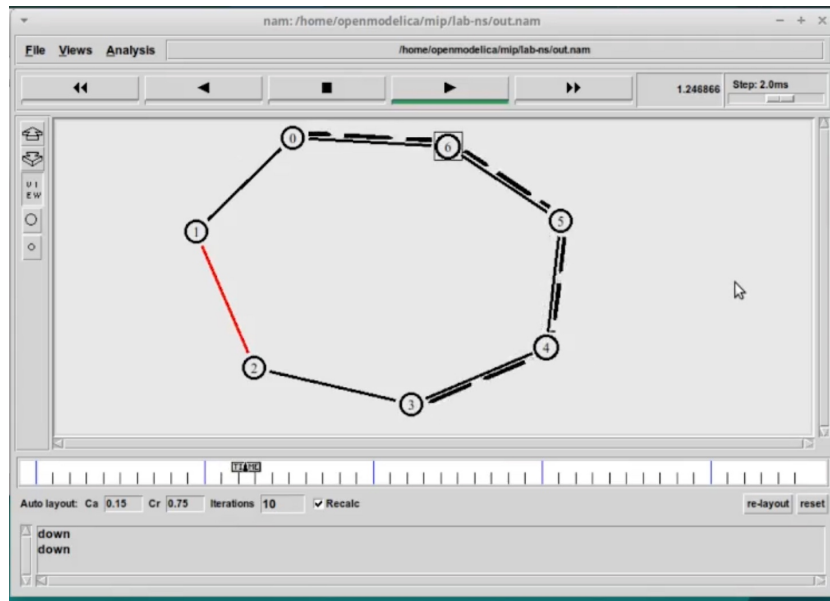


Рис. 3.11: Область моделирования

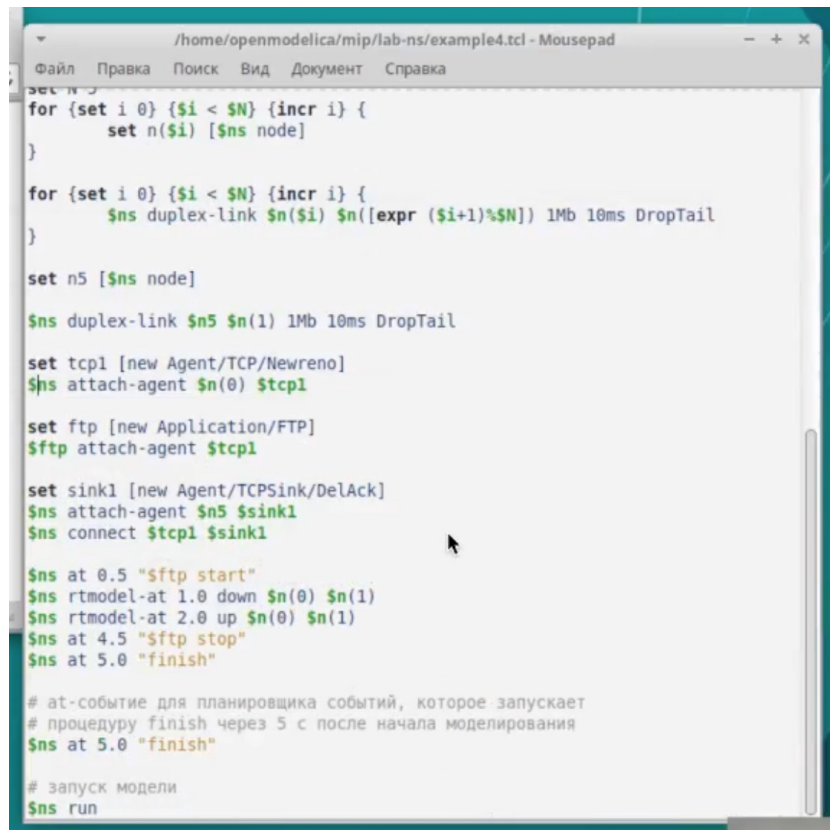
### 3.0.5 Упражнение

Внесем 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 переда-

чи данных должен измениться на резервный, после восстановления соединения пакеты снова должны пойти по кратчайшему пути.

Изменим количество узлов в кольце на 5, а 6 узел n(5) отдельно присоединим к узлу n(1). Вместо агента UDP создадим агента TCP (типа Newreno), а на принимающей стороне используем TCPSink-объект типа DelAck; поверх TCP работает протокол FTP с 0,5 до 4,5 секунд модельного времени Также зададим с 1 по 2 секунду модельного времени разрыв соединения между узлами n(0) и n(1)(3.12).

A screenshot of a text editor window titled "/home/openmodelica/mip/lab-ns/example4.tcl - Mousepad". The editor contains a TCL script for a network simulation. The script defines a ring network of 5 nodes, connects node 5 to node 1, sets up a TCP agent (Newreno) on node 0, and a TCPSink (DelAck) on node 5. It also sets up an FTP application on node 0. The script includes timing events for starting and stopping the FTP transfer, and a final 'finish' event. Comments in Russian explain the purpose of the 'finish' event and the model launch.

```
set n5 [new Agent/UDP]
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 n5 [$ns node]

$ns duplex-link $n5 $n(1) 1Mb 10ms DropTail

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

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

set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n5 $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"
$ns at 5.0 "finish"

# 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. 3.12: Добавление кода

Запустим программу и увидим, что пакеты идут по кратчайшему пути через узел n(1) (3.13).

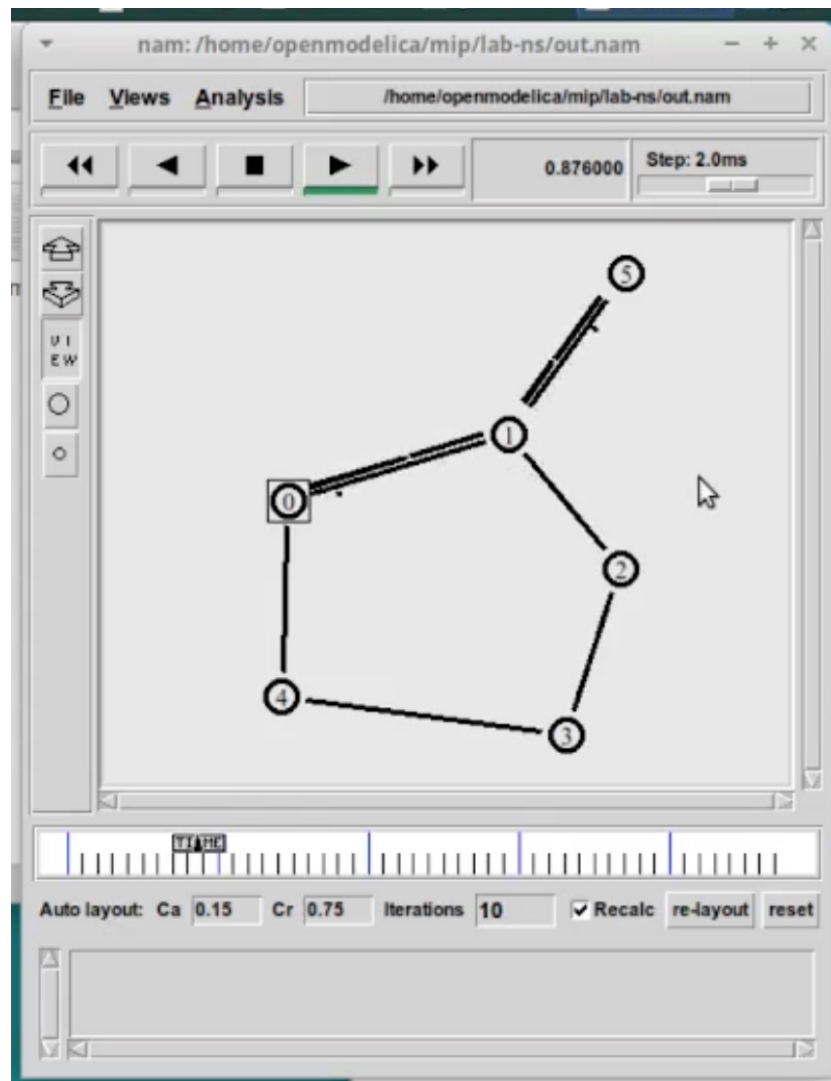


Рис. 3.13: Область моделирования

При разрыве соединения часть пакетов теряется, но поскольку данные обновляются пакеты начинают идти по другому пути (3.14).

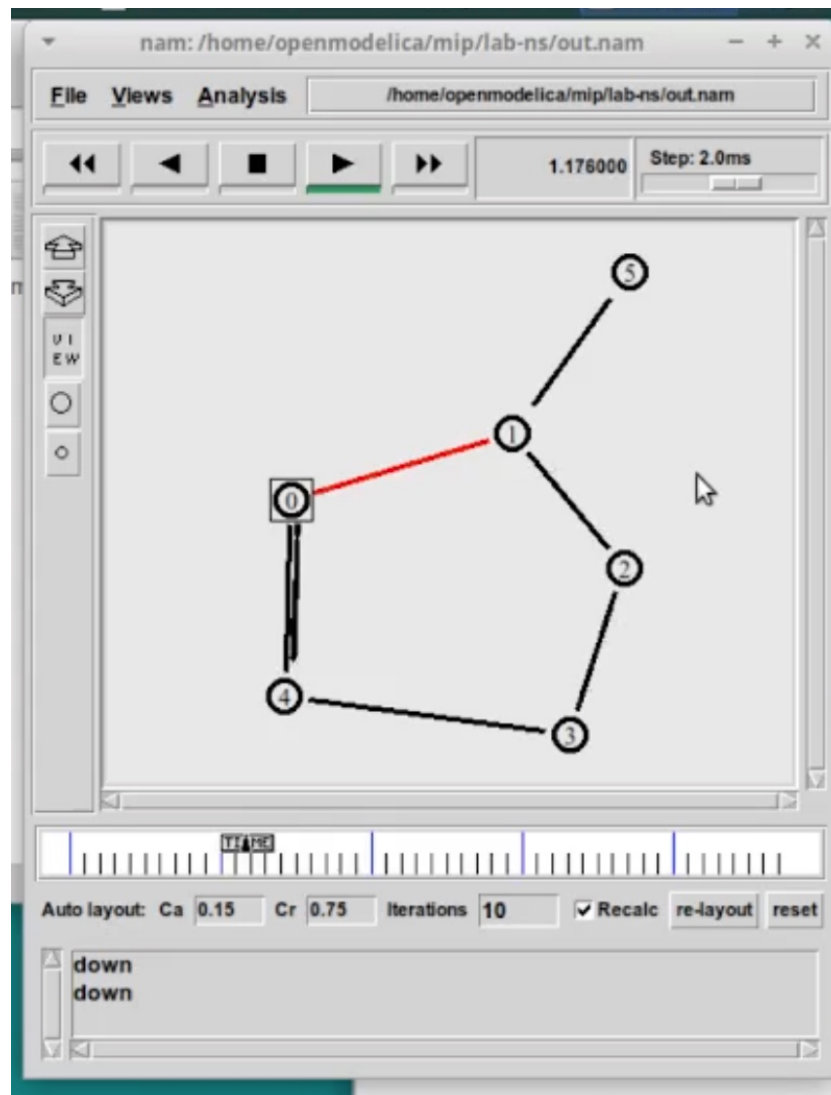


Рис. 3.14: Область моделирования

После восстановления соединения пакеты снова идут по кратчайшему пути (3.15).

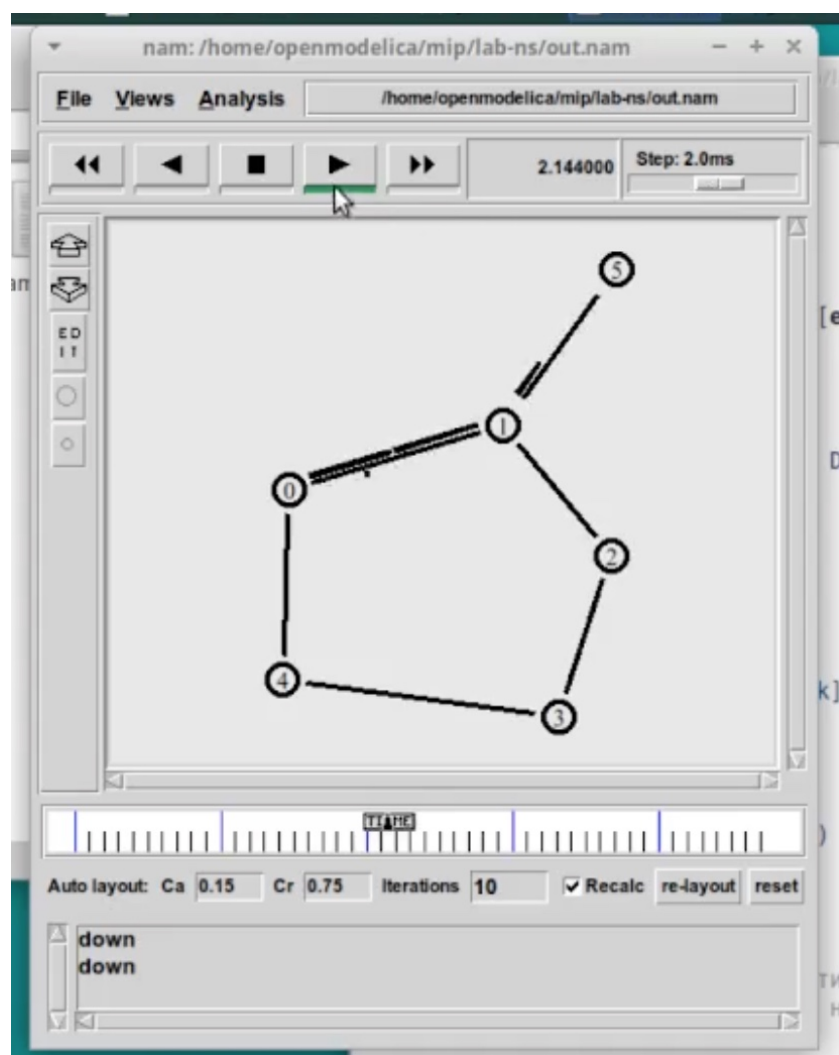


Рис. 3.15: Область моделирования

## **4 Выводы**

В процессе выполнения данной лабораторной работы я приобрела навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также проанализировала полученные результаты моделирования.