

# **Лабораторная работа №1**

**Имитационное моделирование**

Серёгина Ирина Андреевна

# 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>
<b>4</b>	<b>Выводы</b>	<b>21</b>

# Список иллюстраций

3.1	Создаю директорию и файл . . . . .	7
3.2	Содержание файла shablon.tcl . . . . .	8
3.3	Пустая область моделирования . . . . .	9
3.4	Создание нового файла . . . . .	10
3.5	Содержимое файла example1.tcl . . . . .	11
3.6	Визуализация простой модели сети с помощью nam . . . . .	12
3.7	Создание нового файла . . . . .	13
3.8	Содержимое файла example2.tcl . . . . .	13
3.9	Содержимое файла example2.tcl . . . . .	14
3.10	Содержимое файла example2.tcl . . . . .	15
3.11	Визуализация усложненной топологии сети . . . . .	16
3.12	Создание нового файла . . . . .	16
3.13	Содержимое файла example3.tcl . . . . .	17
3.14	Визуализация кольцевой топологии сети . . . . .	18
3.15	Создание нового файла . . . . .	18
3.16	Содержимое файла example4.tcl . . . . .	19
3.17	Визуализация модели, построенной по заданным требованиям . .	20

## **Список таблиц**

# 1 Цель работы

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

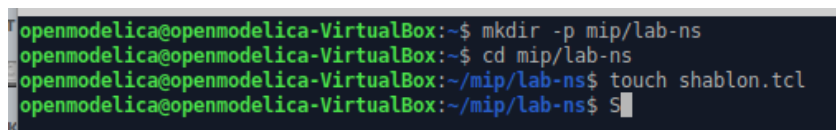
## 2 Задание

1. Создание шаблона сценария для NS-2.
2. Пример топологии сети, состоящей из двух узлов и одного соединения.
3. Пример усложненной топологии сети.
4. Пример кольцевой топологии сети.
5. Упражнение.

### 3 Выполнение лабораторной работы

#### 1. Создание шаблона сценария для NS-2.

Создаю директорию `mip`, где буду выполнять лабораторные работы. В ней создаю директорию `lab-ns`, а в ней файл `shablon.tcl` (рис. 3.1).



```
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$ S
```

Рис. 3.1: Создаю директорию и файл

После этого открываю файл `shablon.tcl` на редактирование и создаю симулятор. Затем создадим переменную `nf` и укажем, что требуется открыть на запись 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` следует запустить через 5 с после начала моделирования, после чего запустить симулятор `ns`. (рис. 3.2).

```
set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

set f [open out.tr w]

$ns trace-all $f

proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

$ns at 5.0 "finish"

$ns run
```

Рис. 3.2: Содержание файла shablon.tcl

Сохранив изменения в отредактированном файле shablon.tcl и закрыв его, можно запустить симулятор командой `ns shablon.tcl`, при этом на экране появится



сообщение типа `nam: empty trace file out.nam` поскольку ещё не определены никакие объекты и действия. По этой же причине наша область моделирования будет пустой (рис. 3.3).

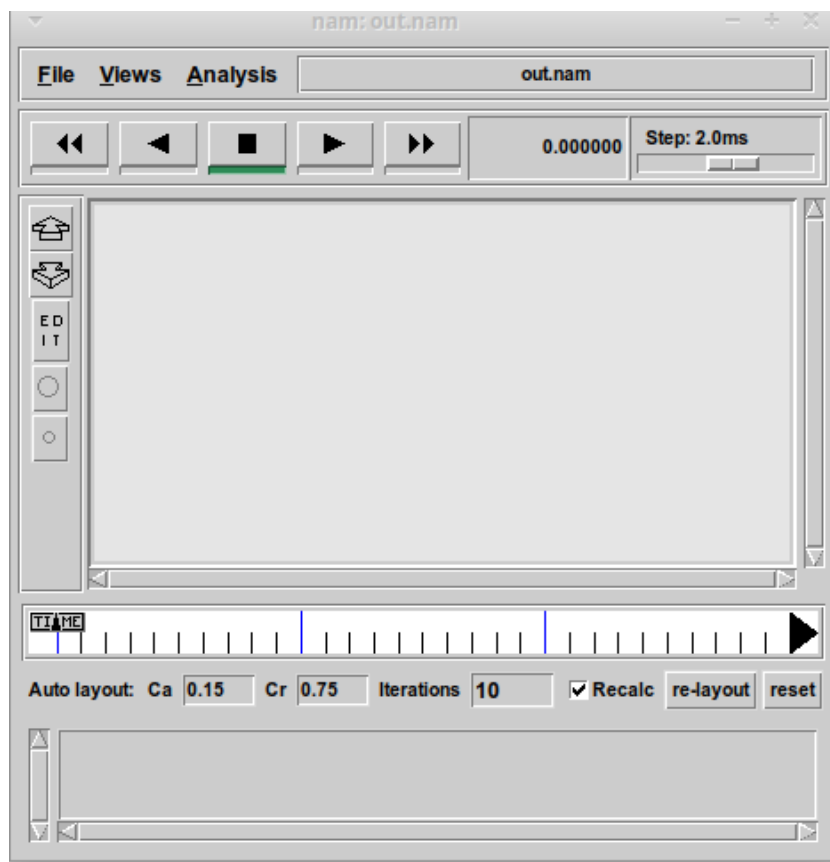


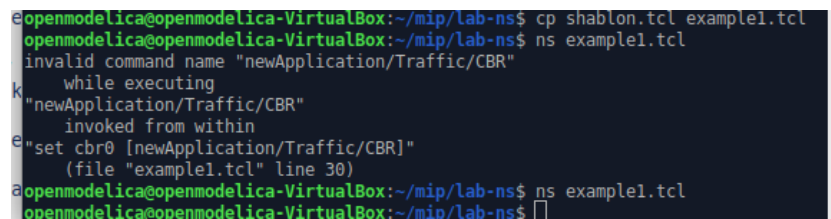
Рис. 3.3: Пустая область моделирования

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

Постановка задачи. 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.

Для начала я копирую содержимое файла `shablon.tcl` в файл `example1.tcl` и в дальнейшем все изменения буду вставлять перед строчкой `$ns at 5.0 "finish"` (рис.

3.4).



```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example1.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example1.tcl
invalid command name "newApplication/Traffic/CBR"
while executing
"newApplication/Traffic/CBR"
invoked from within
"set cbr0 [newApplication/Traffic/CBR]"
(file "example1.tcl" line 30)
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example1.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$
```

Рис. 3.4: Создание нового файла

Создадим агенты для генерации и приёма трафика. Создаётся 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:  $R = 500 \cdot 8 / 0,005 = 8000000$  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. 3.5).

```

set N 2
for {set i 0} {$i < $N} {incr i} {
    set n($i) [$ns node]
}

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

set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(1) $null0

$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run

```

Рис. 3.5: Содержимое файла example1.tcl

Сохранив изменения в отредактированном файле и запустив симулятор получим в качестве результата запуск аниматора nam в фоновом режиме. При нажатии на кнопку play в окне nam через 0.5 секунды из узла 0 данные начнут поступать к узлу 1. Это процесс можно замедлить, выбирая шаг отображения в nam. Можно осуществлять наблюдение за отдельным пакетом, щёлкнув по нему в окне nam, а щёлкнув по соединению, можно получить о нем некоторую информацию. (рис. 3.6).

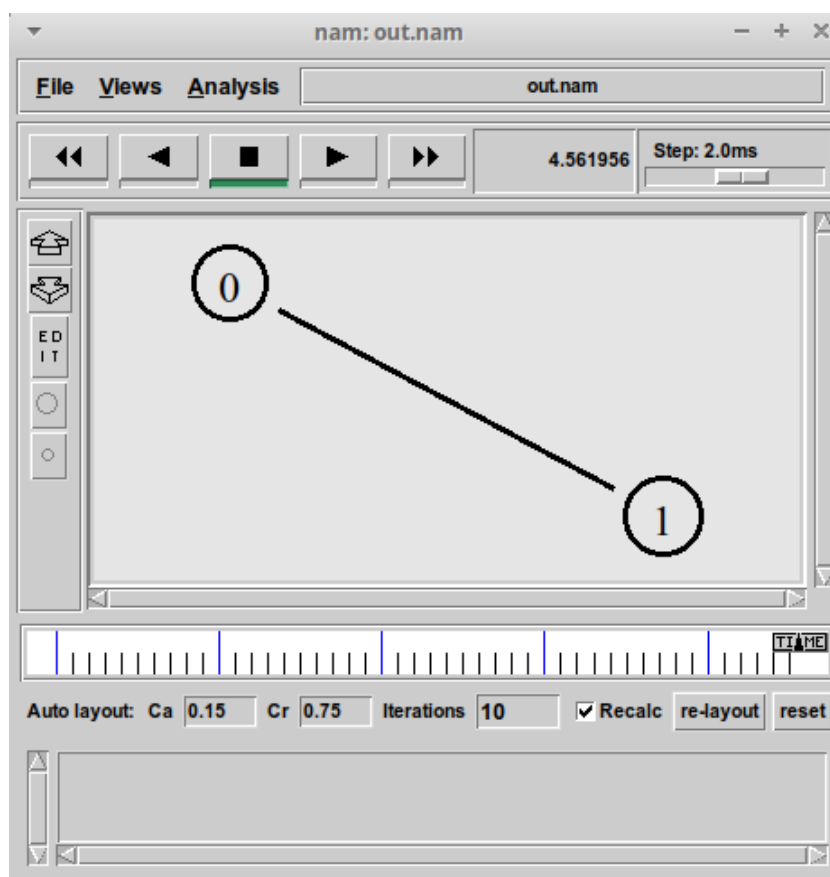


Рис. 3.6: Визуализация простой модели сети с помощью nam

### 3. Пример с усложненной 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  $n_3$  (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  $n_1$ , подключён к null-агенту на узле  $n_3$  (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 (рис. 3.7).

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example1.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example2.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example2.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$
```

Рис. 3.7: Создание нового файла

Создадим 4 узла и 3 дуплексных соединения с указанием направления (рис. 3.8).

```
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. 3.8: Содержимое файла `example2.tcl`

Создадим агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP (рис. 3.9).

```

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

# создание источника CBR-трафика
# и присоединение его к агенту udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize 500
$cbr0 set interval 0.005
$cbr0 attach-agent $udp0

# создание агента TCP и присоединение его к узлу n(1)
set tcp1 [new Agent/TCP]
$ns attach-agent $n(1) $tcp1

# 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

```

Рис. 3.9: Содержимое файла example2.tcl

Создадим агенты-получатели, соединим агенты udp0 и tcp1 и их получателей, зададим описание цвета каждого потока, отслеживание событий в очереди, наложим ограничения на размер очереди и добавим at-события (рис. 3.10).

```

# 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"

$ns at 5.0 "finish"

```

Рис. 3.10: Содержимое файла example2.tcl

Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования. При запуске скрипта можно заметить, что по соединениям между узлами  $n(0)$ – $n(2)$  и  $n(1)$ – $n(2)$  к узлу  $n(2)$  передаётся данных больше, чем способно передаваться по соединению от узла  $n(2)$  к узлу  $n(3)$ . Действительно, мы передаём 200 пакетов в секунду от каждого источника данных в узлах  $n(0)$  и  $n(1)$ , а каждый пакет имеет размер 500 байт. Таким образом, полоса каждого соединения 0, 8 Mb, а суммарная — 1, 6 Mb. Но соединение  $n(2)$ – $n(3)$  имеет полосу лишь 1 Mb. Следовательно, часть пакетов должна теряться. В окне аниматора можно видеть пакеты в очереди, а также те пакеты, которые отбрасываются при переполнении (рис. 3.11).

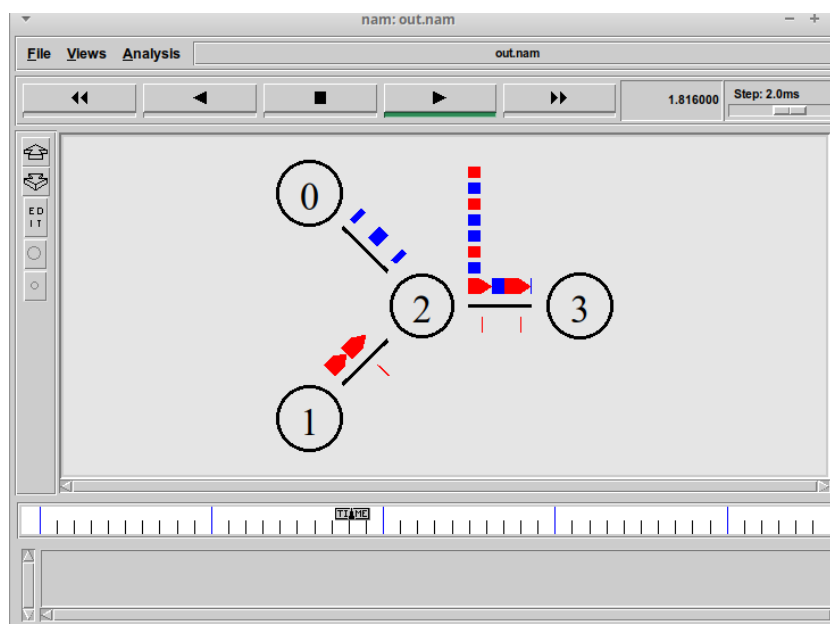


Рис. 3.11: Визуализация усложненной топологии сети

#### 4. Пример с кольцевой топологией сети

Постановка задачи. 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.

Скопируем содержимое созданного шаблона в новый файл и откроем `example3.tcl` на редактирование (рис. 3.12).

```
openmodelica@openmodelica-VirtualBox:~$ cd mip/lab-ns
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ nc example3.tcl
```

Рис. 3.12: Создание нового файла

Опишем топологию моделируемой сети, 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  $n(2)$  (Добавим команду разрыва соединения между узлами  $n(1)$  и  $n(2)$  на время в одну секунду, а также время начала и окончания передачи данных. (рис. 3.13).

```
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"

$ns rtproto DV

$ns at 5.0 "finish"
```

Рис. 3.13: Содержимое файла example3.tcl

Передача данных при кольцевой топологии сети в случае разрыва соединения представлена на рисунке ниже (рис. 3.14).

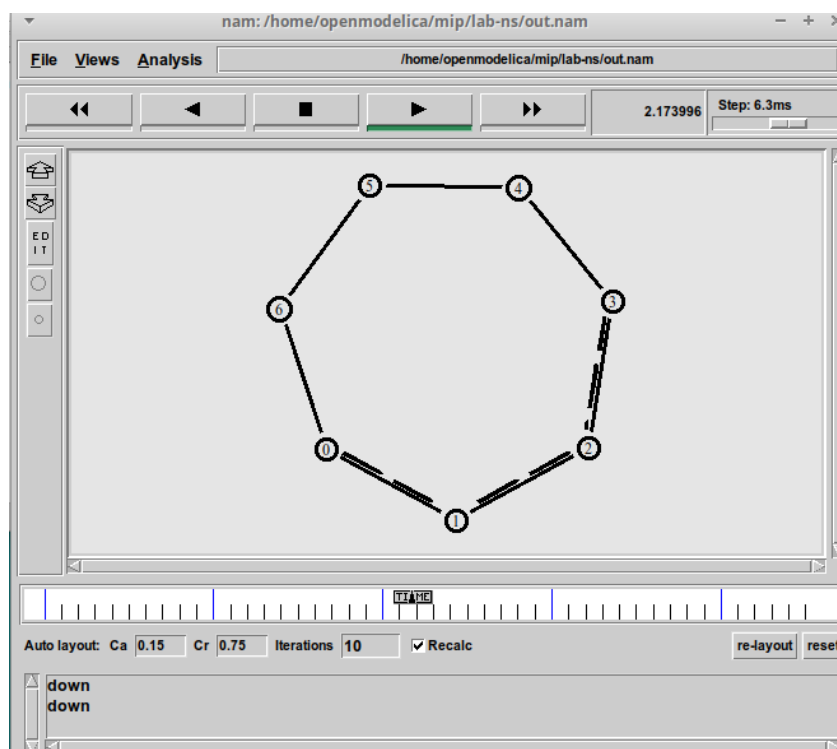


Рис. 3.14: Визуализация кольцевой топологии сети

## 5. Выполнение упражнения

Необходимо внести следующие изменения в реализацию кольцевой топологии 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.

Для начала я скопировал содержимое файла `example3.tcl` в новый файл `example4.tcl` и открыла его для редактирования (рис. 3.15).

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp example3.tcl example4.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example4.tcl
```

Рис. 3.15: Создание нового файла

После этого поменяла файл таким образом, что данная модели соответствовала всем требованиям, указанным выше (рис. 3.16).

```
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 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"

$ns run S
```

Рис. 3.16: Содержимое файла example4.tcl

Построенная модель соответствует требованиям и выглядит так же, как и требовалось в инструкции (рис. 3.17).

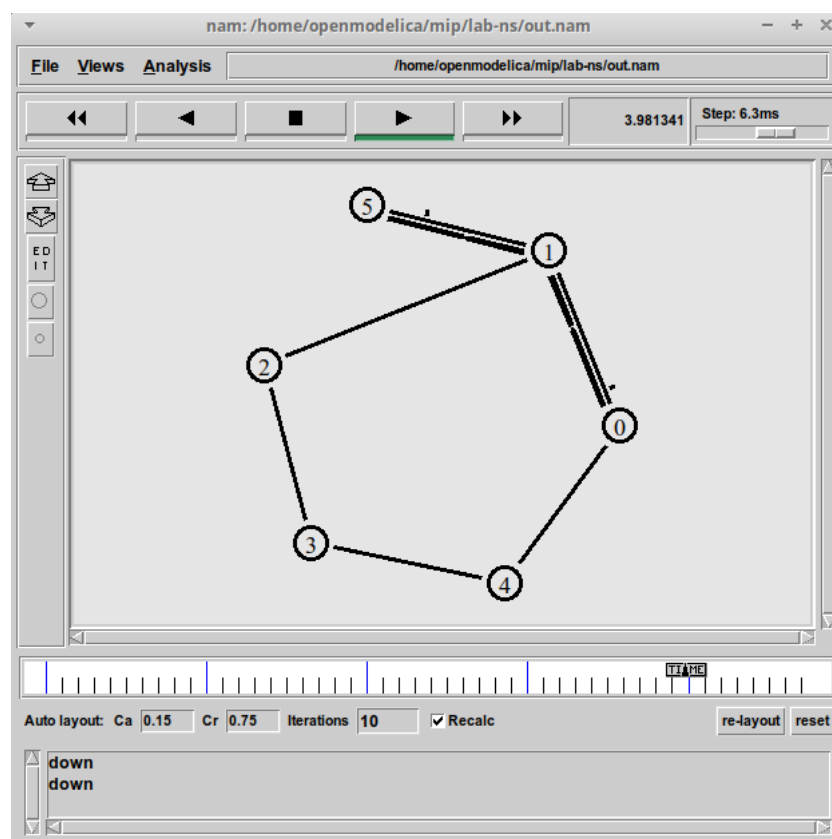


Рис. 3.17: Визуализация модели, построенной по заданным требованиям

## **4 Выводы**

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