

# **Лабораторная работа №1**

**Простые модели компьютерной сети**

Кадров Виктор Максимович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Шаблон сценария для NS-2 . . . . .	6
3.2	Простой пример описания топологии сети, состоящей из двух узлов и одного соединения . . . . .	8
3.3	Пример с усложненной топологией сети . . . . .	10
3.4	Пример с кольцевой топологией сети . . . . .	13
3.5	Выполнение упражнения . . . . .	15
<b>4</b>	<b>Выводы</b>	<b>19</b>
	<b>Список литературы</b>	<b>20</b>

# Список иллюстраций

3.1	Создание директорий и файла . . . . .	6
3.2	Симуляция шаблона . . . . .	7
3.3	Скрипт шаблона . . . . .	8
3.4	Визуализация простой модели сети с помощью <code>nam</code> . . . . .	9
3.5	Скрипт сети из двух узлов и одного соединения . . . . .	10
3.6	Модель с усложненной топологией сети без симуляции . . . . .	11
3.7	Модель с усложненной топологией сети. Симуляция . . . . .	12
3.8	Скрипт модели с усложненной топологией сети . . . . .	13
3.9	Передача данных по кратчайшему пути сети с кольцевой топологией	14
3.10	Передача данных по сети с кольцевой топологией в случае разрыва соединения . . . . .	14
3.11	Скрипт сети из двух узлов и одного соединения . . . . .	15
3.12	Скрипт модели из упражнения . . . . .	16
3.13	Топология сети из упражнения . . . . .	16
3.14	Движение по кратчайшему пути . . . . .	17
3.15	Разрыв связи между узлами . . . . .	17
3.16	Движение по длинному пути . . . . .	18

# 1 Цель работы

Приобретение навыков моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также анализ полученных результатов моделирования[1].


## 2 Задание

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

## 3 Выполнение лабораторной работы

### 3.1 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).



```
142 mkdir -p mip/lab-ns
143 cd mip/lab-ns/
144 ls
145 touch shablon.tcl
146 nano shablon.tcl
147 ns shablon.tcl
```

Рис. 3.1: Создание директорий и файла

Откроем на редактирование файл `shablon.tcl`. 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`. Наконец, с помощью команды `at` указываем планировщику событий, что процедуру `finish` следует запустить через 5 с после начала моделирования, после чего запустить симулятор `ns`. Сохранив изменения в отредактированном файле `shablon.tcl` и закрыв его, можно запустить симулятор. При этом на экране появится сообщение типа `nam: empty trace file out.nam` поскольку ещё не

определены никакие объекты и действия. (рис. 3.2).

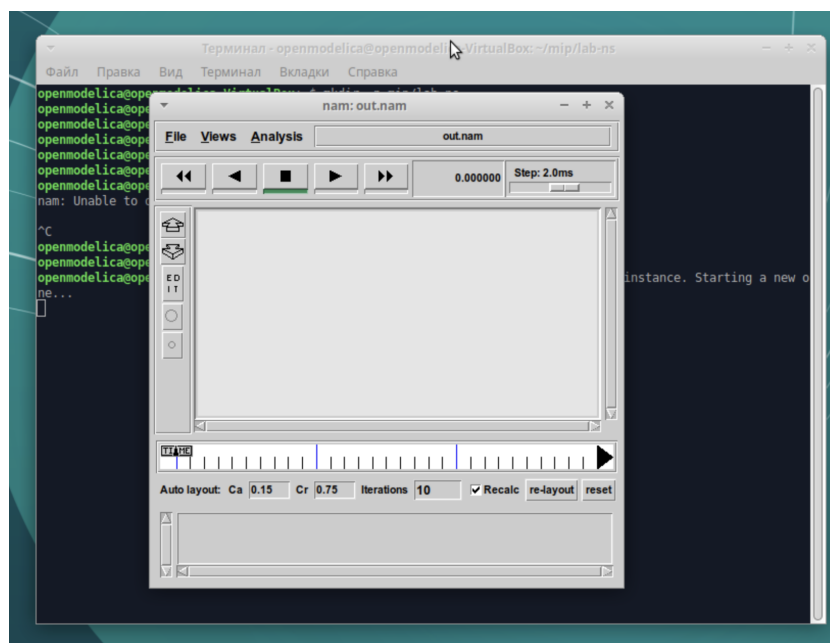
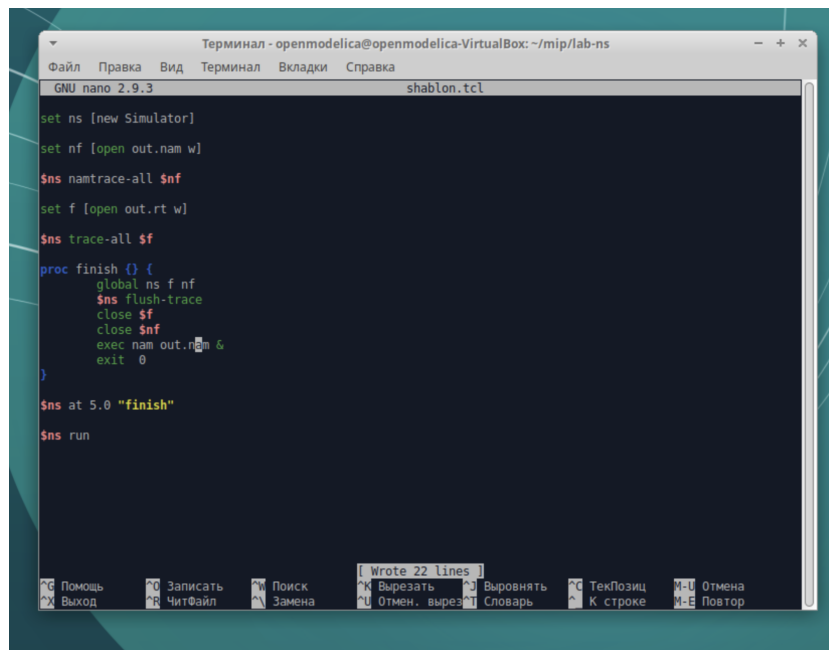


Рис. 3.2: Симуляция шаблона

Получившийся шаблон можно использовать в дальнейшем в большинстве разрабатываемых скриптов NS-2, добавляя в него до строки \$ns at 5.0 “finish” описание объектов и действий моделируемой системы. (рис. 3.3).

A screenshot of a terminal window titled "Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns". The terminal shows the GNU nano 2.9.3 editor editing a file named "shablon.tcl". The script content is as follows:

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set f [open out.rt 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
```

The bottom of the terminal shows the nano editor's status bar with various menu options like "Помощь", "Записать", "Поиск", etc.

Рис. 3.3: Скрипт шаблона

## 3.2 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.

Реализация модели. Скопируем содержимое созданного шаблона в новый файл `example1.tcl` и откроем на редактирование. Создадим агенты для генерации и приёма трафика. Создадим агент UDP и присоединим 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. 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. Сохранив изменения в отредактированном файле и запустив симулятор, получим в качестве результата запуск аниматора nam в фоновом режиме. (рис. 3.4).

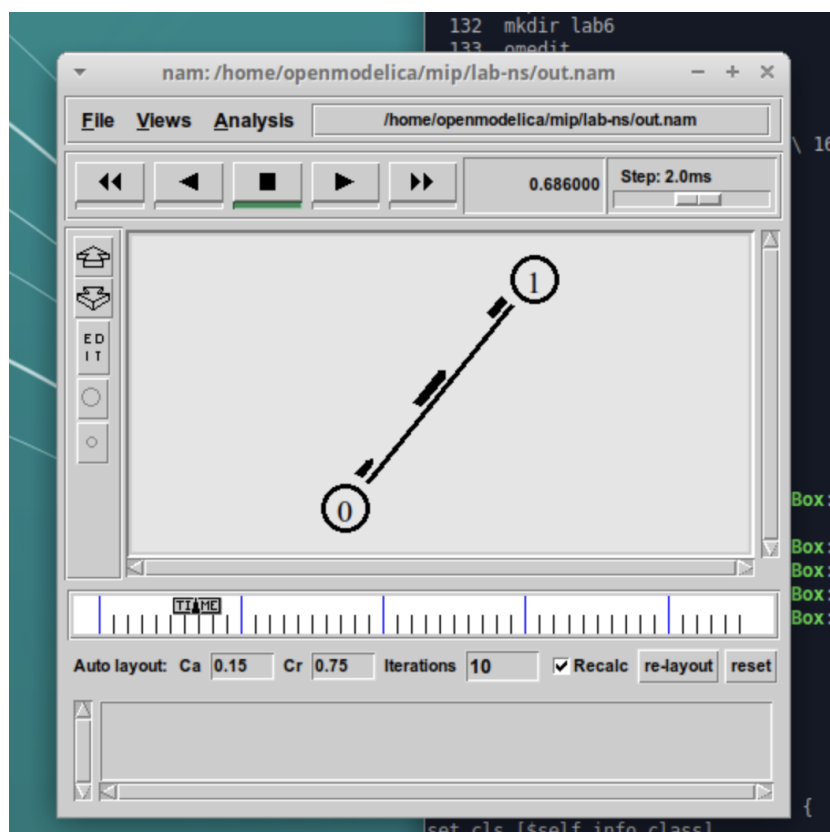
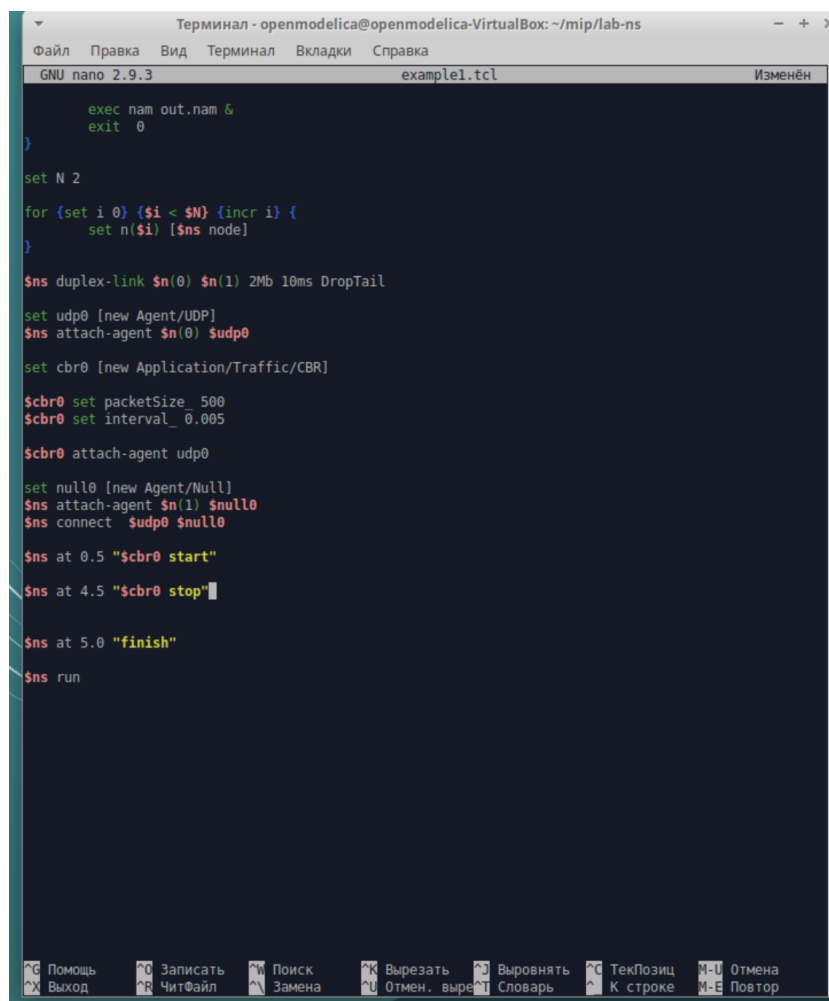


Рис. 3.4: Визуализация простой модели сети с помощью nam

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. 3.5).

A screenshot of a terminal window titled "Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns". The terminal shows the GNU nano 2.9.3 editor with a file named "example1.tcl". The script contains the following code:

```
exec nam out.nam &
exit 0

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
```

The terminal window has a menu bar with options: Файл, Правка, Вид, Терминал, Вкладки, Справка. The bottom status bar shows various keyboard shortcuts like ^G Помощь, ^O Записать, ^W Поиск, etc.

Рис. 3.5: Скрипт сети из двух узлов и одного соединения

### 3.3 Пример с усложненной топологией сети

Постановка задачи. Описание моделируемой сети: – сеть состоит из 4 узлов (n0, n1, n2, n3); – между узлами n0 и n2, n1 и n2 установлено дуплексное соединение с пропускной способностью 2 Мбит/с и задержкой 10 мс; – между узлами n2 и n3 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; – ТСП-источник на узле n0 подключается к ТСП-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.

Реализация модели. Скопируем содержимое созданного шаблона в новый файл и откроем example2.tcl на редактирование. Создадим 4 узла и 3 дуплексных соединения с указанием направления. (рис. 3.6).

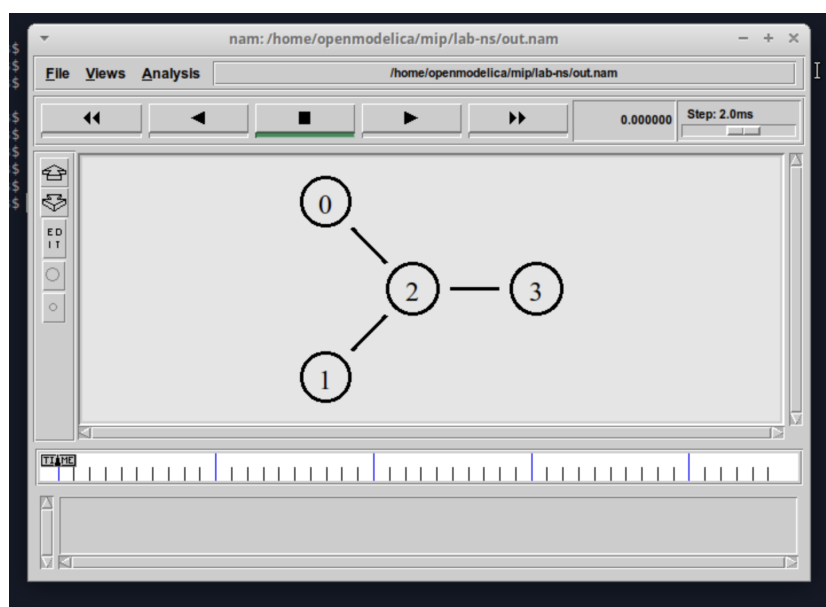


Рис. 3.6: Модель с усложненной топологией сети без симуляции

Создадим агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP. Создадим агенты-получатели. Соединим агенты udr0 и tcr1 и их получателей. Зададим описание цвета каждого потока. Отслеживание событий в очереди. Наложим ограничения на размер очереди. Добавим at-события. Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования (рис.

3.7).

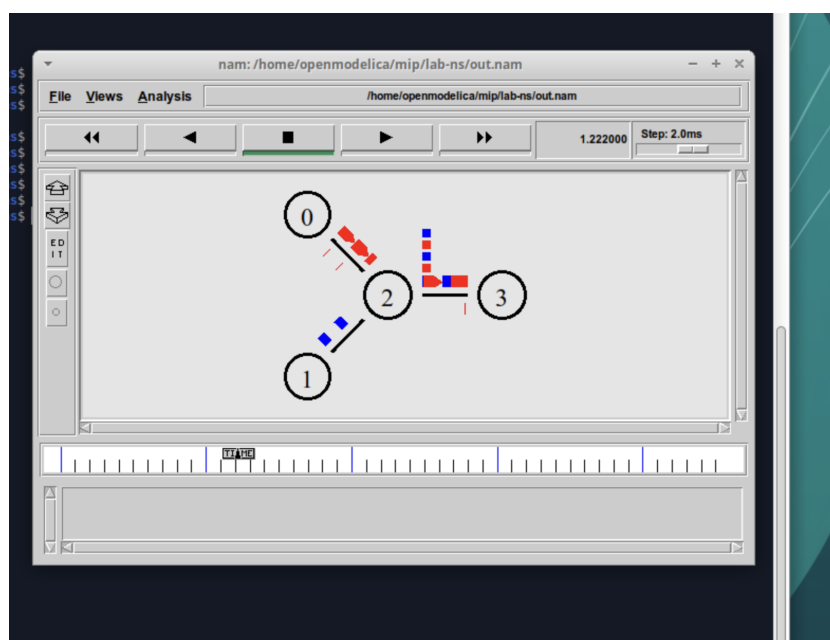
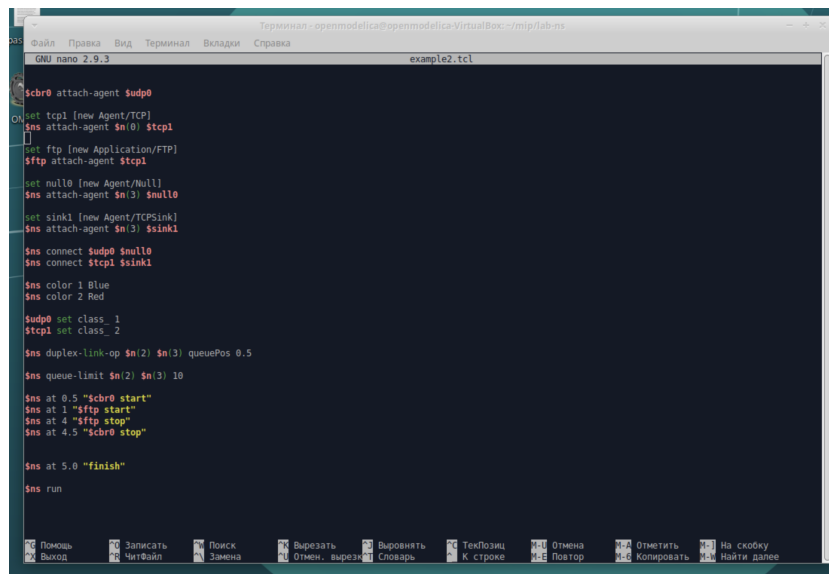


Рис. 3.7: Модель с усложненной топологией сети. Симуляция

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. 3.8).



```
#!/usr/bin/tcl
# example2.tcl

set ns [new AgentNS]
set tcp1 [new Agent/TCP]
set ftp [new Agent/FTP]
set null0 [new Agent/Null]
set sink1 [new Agent/TCPSink]

$ns attach-agent $n(0) $tcp1
$ns attach-agent $n(3) $null0
$ns attach-agent $n(3) $sink1

$ns connect $sudp0 $null0
$ns connect $tcp1 $sink1

$ns color 1 Blue
$ns color 2 Red

$ns duplex-link-op $n(2) $n(3) queuePos 0.5
$ns queue-limit $n(2) $n(3) 10

$ns at 0.5 "$tcp1 start"
$ns at 1 "$ftp start"
$ns at 4 "$ftp stop"
$ns at 4.5 "$tcp1 stop"

$ns at 5.0 "finish"

$ns run
```

Рис. 3.8: Скрипт модели с усложненной топологией сети

### 3.4 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.

Реализация модели. Скопируем содержимое созданного шаблона в новый файл и откроем `example3.tcl` на редактирование. 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  $n(2)$  (рис. 3.9).

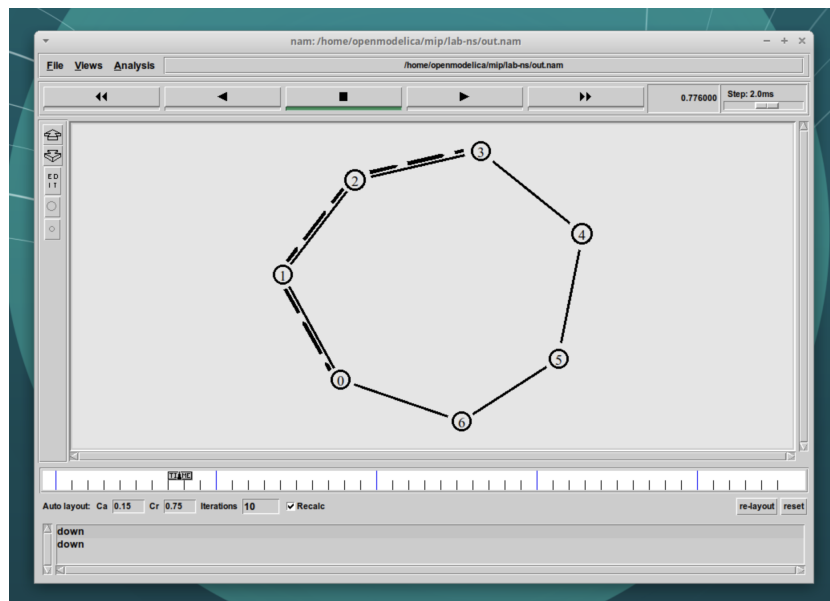


Рис. 3.9: Передача данных по кратчайшему пути сети с кольцевой топологией

Добавим команду разрыва соединения между узлами  $n(1)$  и  $n(2)$  на время в одну секунду, а также время начала и окончания передачи данных. Передача данных при кольцевой топологии сети в случае разрыва соединения представлена на (рис. 3.10).

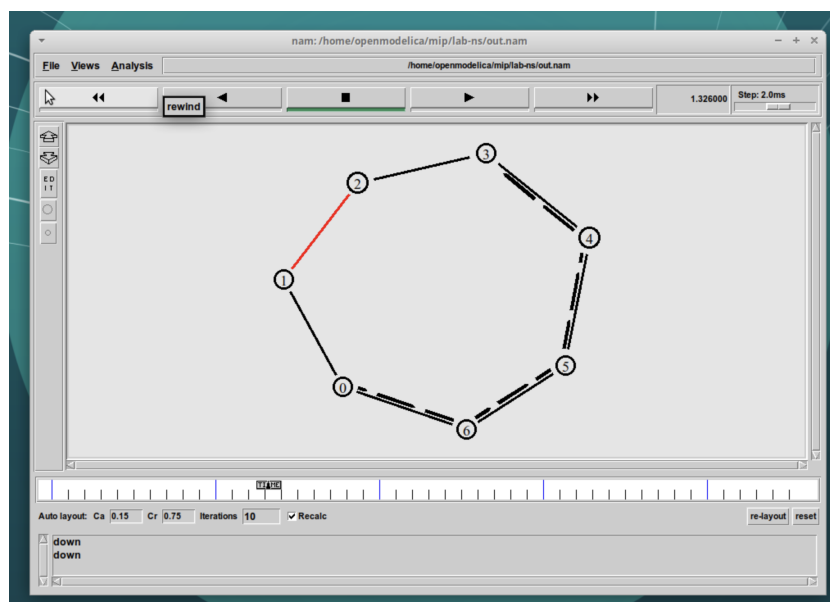


Рис. 3.10: Передача данных по сети с кольцевой топологией в случае разрыва соединения

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

```

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]) 10Mb 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 1.8 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 run
  
```

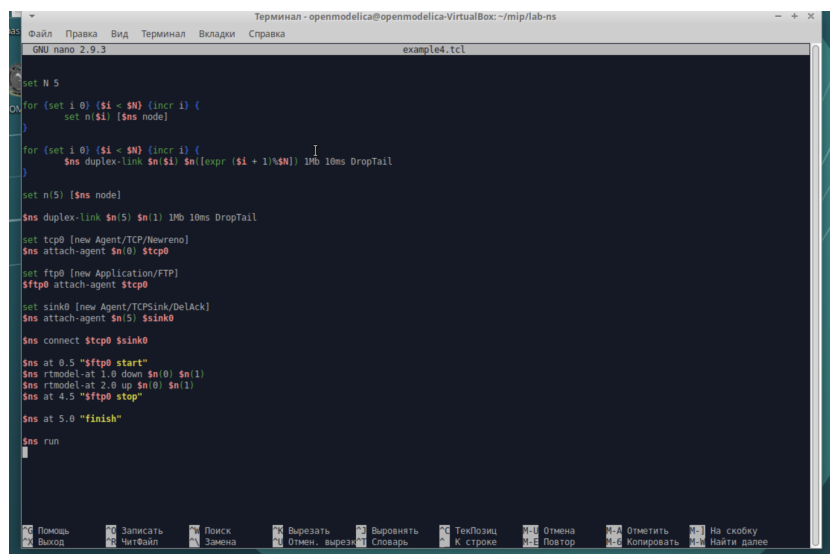
Рис. 3.11: Скрипт сети из двух узлов и одного соединения

## 3.5 Выполнение упражнения

Упражнение. Внесите следующие изменения в реализацию примера с кольцевой топологией сети: – повторить топологию сети, предоставленную в файле с заданиями к лабораторной работе; – 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. 3.12).



```
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}] 1000 10ms DropTail
}

set n(5) [$ns node]
$ns duplex-link $n(5) $n(1) 1000 10ms DropTail

set tcp0 [new Agent/TCP/TCPNewreno]
$ns attach-agent $n(0) $tcp0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

set sink0 [new Agent/TCP/link/DelAck]
$ns attach-agent $n(5) $sink0

$ns connect $tcp0 $sink0

$ns at 0.5 "$ftp0 start"
$ns rtmodel-at 1.0 down $n(0) $n(1)
$ns rtmodel-at 2.0 up $n(0) $n(1)
$ns at 4.5 "$ftp0 stop"

$ns at 5.0 "finish"

$ns run
```

Рис. 3.12: Скрипт модели из упражнения

Топология сети. (рис. 3.13).

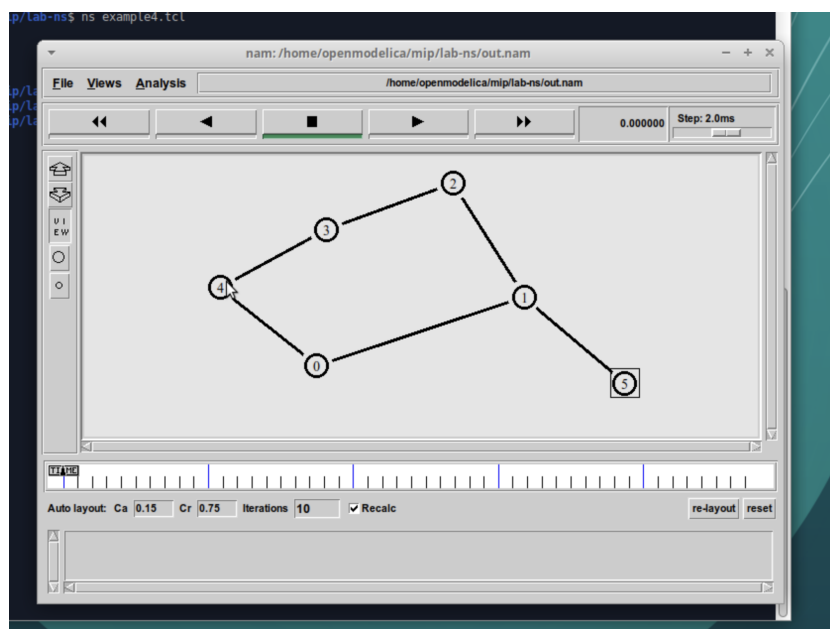


Рис. 3.13: Топология сети из упражнения



Движение по кратчайшему пути. (рис. 3.14).

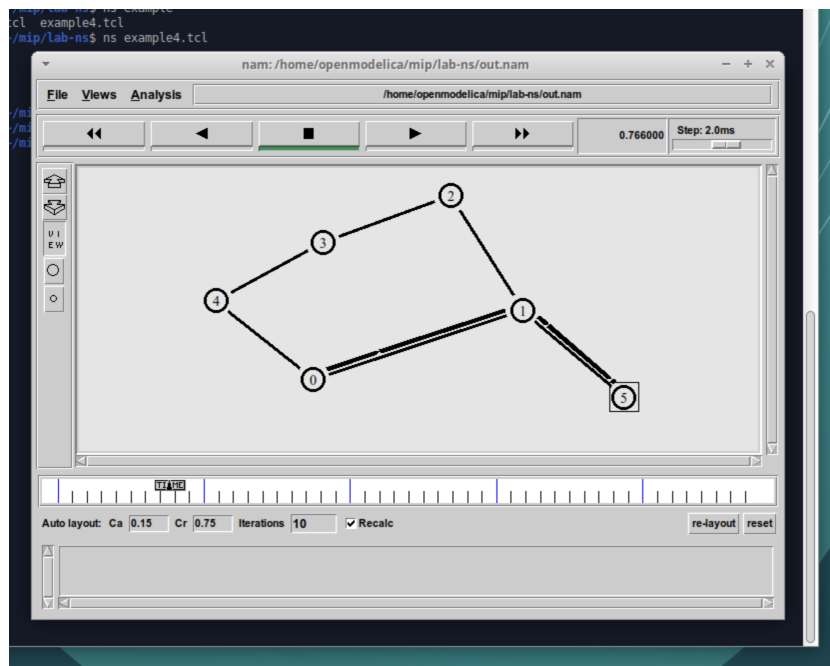


Рис. 3.14: Движение по кратчайшему пути

Разрыв связи между узлами.(рис. 3.15).

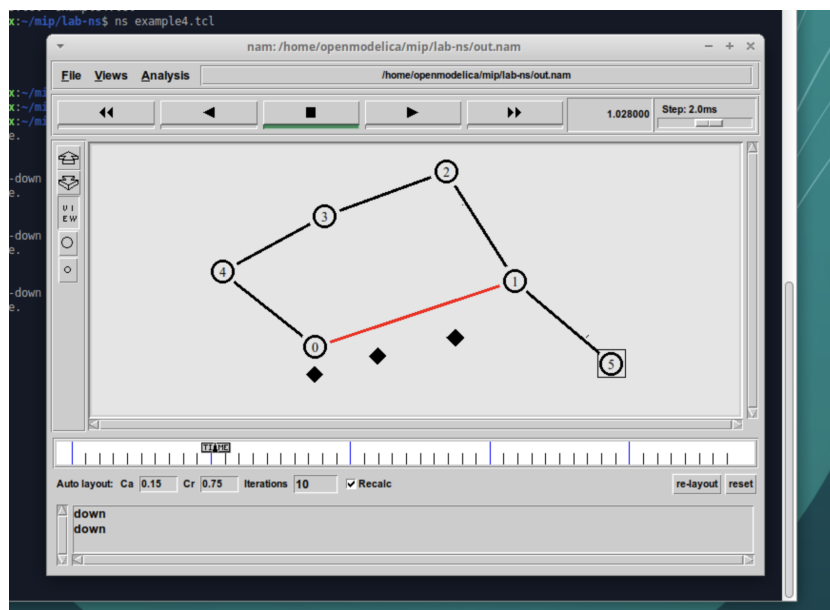


Рис. 3.15: Разрыв связи между узлами

Движение по длинному пути. (рис. 3.16).

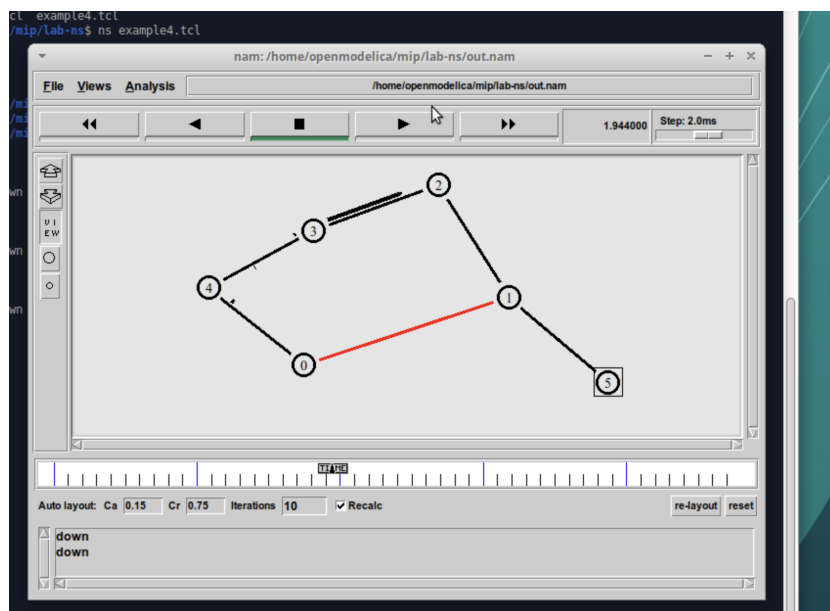


Рис. 3.16: Движение по длинному пути

## **4 Выводы**

Мы приобрели навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также провели анализ полученных результатов моделирования.

## **Список литературы**

1. Королькова А.В., Кулябов Д.С. Лабораторная работа 1. Простые модели компьютерной сети [Электронный ресурс].