

Лабораторная работа №1

Простые модели компьютерной сети

Алиева Милена Арифовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	20

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Редактирование shablon.tcl	8
3.3	Пустая область моделирования	9
3.4	Отредактированный файл для второго примера	10
3.5	Второй пример	11
3.6	Третий пример	13
3.7	Результат третьего примера	14
3.8	Четвёртый пример	15
3.9	Результат четвёртого примера	16
3.10	Результат четвёртого примера	17
3.11	Добавление строки \$ns rtp proto DV	18
3.12	Результат четвёртого примера после изменений	18
3.13	Упражнение	19

Список таблиц

1 Цель работы

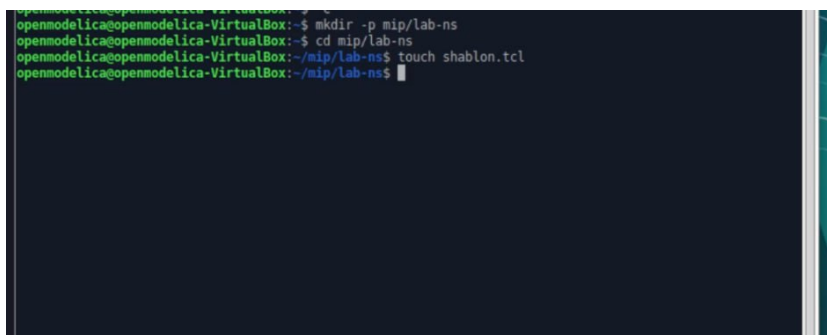
Приобретение навыков моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также анализ полученных результатов моделирования.

2 Задание

1. 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

1. В своём рабочем каталоге создаём директорию `mip`, в которой будут выполняться лабораторные работы. Внутри `mip` создаём 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$
```

Рис. 3.1: Создание каталога и файла

2. Открываем на редактирование файл `shablon.tcl`. Сначала создадим объект типа `Simulator`, создадим переменную `nf` и укажем, что требуется открыть на запись `nam`-файл для регистрации выходных результатов моделирования, вторая строка даёт команду 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`. (рис. 3.2).

```

# 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 # описание глобальных переменных
    $ns flush-trace # прекращение трассировки
    close $f # закрытие файлов трассировки
    close $nf # закрытие файлов трассировки nam
    # запуск nam в фоновом режиме
    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: Редактирование shablon.tcl

3. 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).

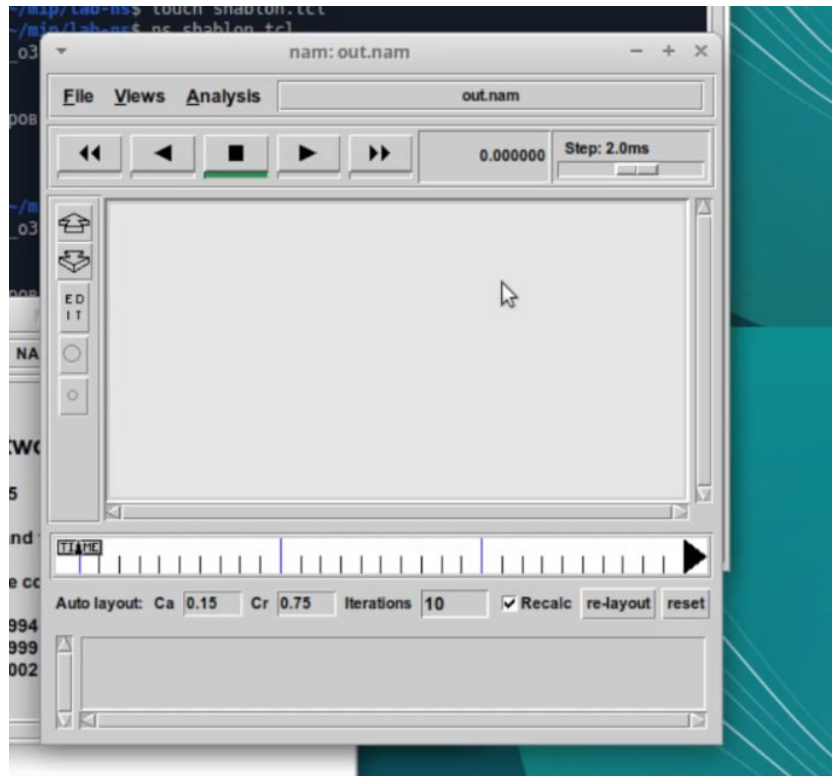


Рис. 3.3: Пустая область моделирования

4. Выполним второй пример, который посвящён описанию топологии сети, состоящей из двух узлов и одного соединения. Нам 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. (рис. 3.4).

```

* /home/openmodelica/mip/lab-ns/example1.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка

# 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
}

# 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 Мб/с 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"
# запуск модели
$ns run

```

Рис. 3.4: Отредактированный файл для второго примера

5. Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования (рис. 3.5).

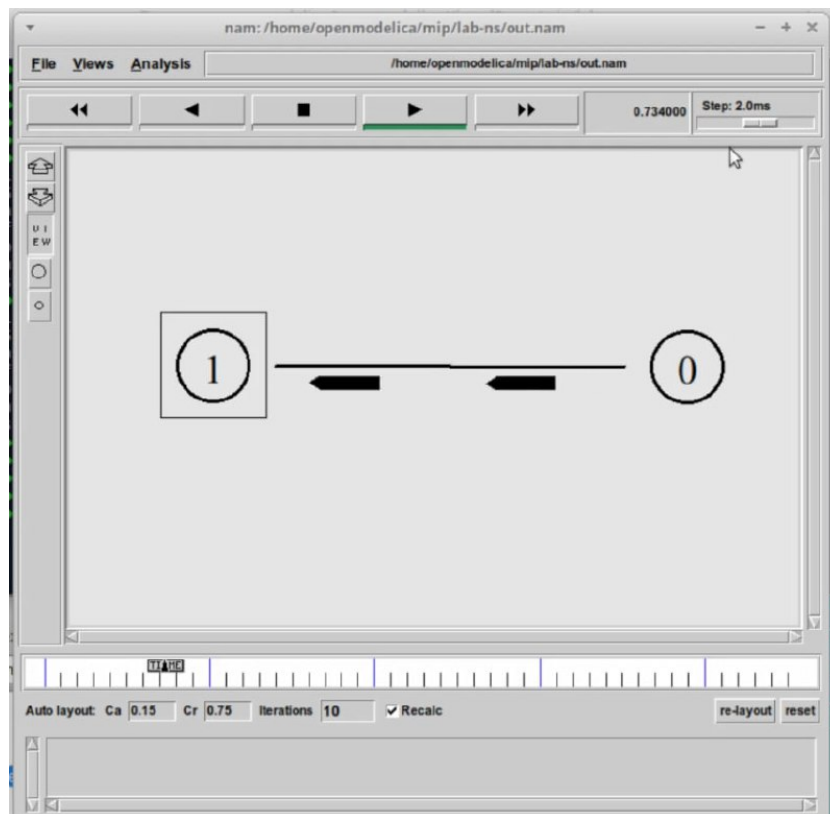


Рис. 3.5: Второй пример

6. Выполним третий пример. 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

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.

Откроем example2.tcl на редактирование, 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 udr0 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).

```

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

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

# 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

$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

```

Рис. 3.6: Третий пример

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

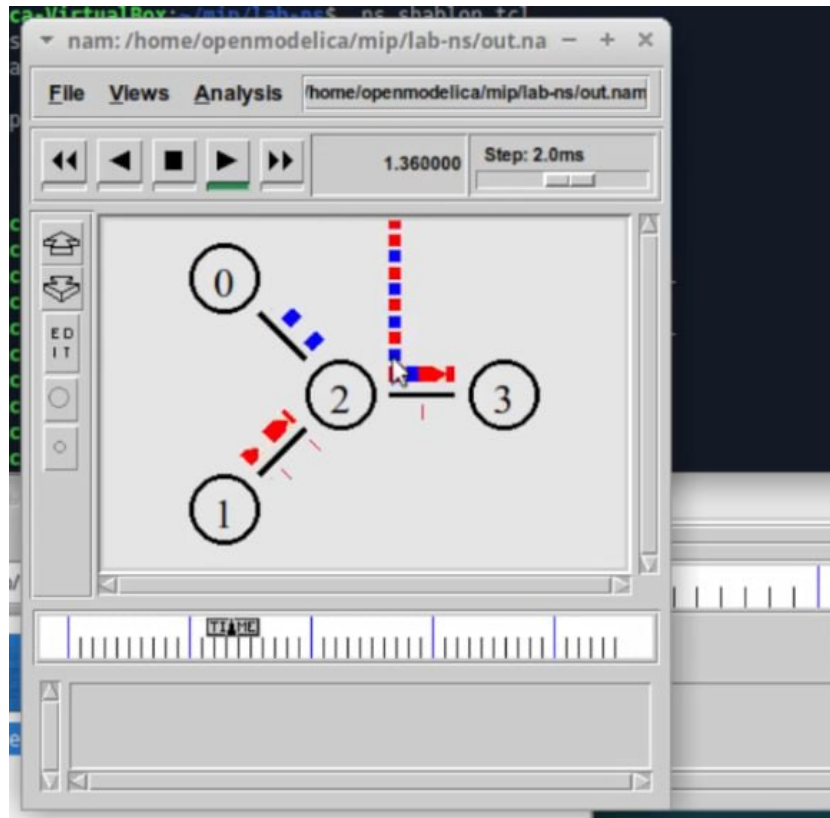


Рис. 3.7: Результат третьего примера

8. 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.

Откроем `example3.tcl` на редактирование. Опишем топологию моделируемой сети, 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. 3.8).

```

}
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: Четвёртый пример

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

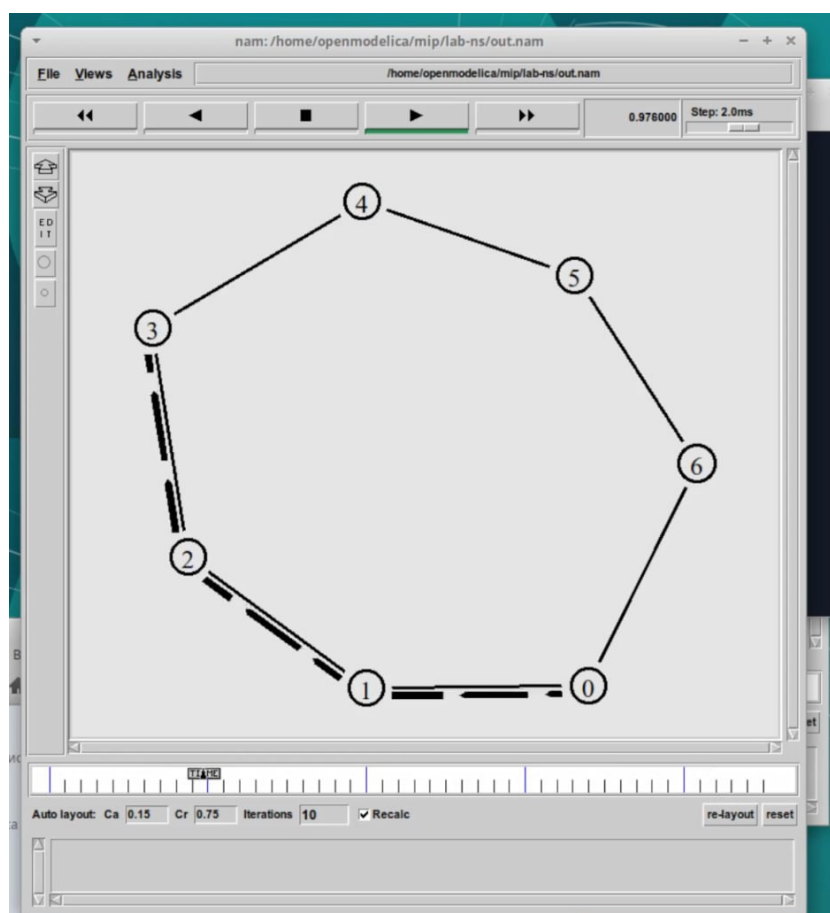


Рис. 3.9: Результат четвёртого примера

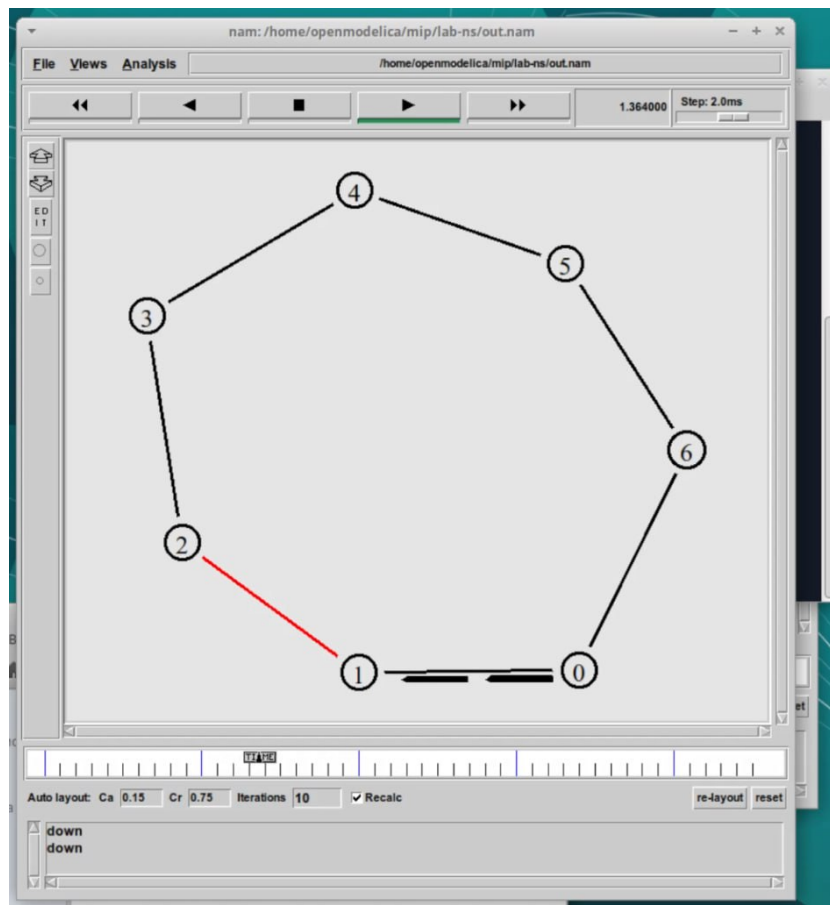


Рис. 3.10: Результат четвёртого примера

10. Добавив в начало скрипта `$ns rtproto DV` и увидим, что сразу после запуска в сети отправляется небольшое количество маленьких пакетов, используемых для обмена информацией, необходимой для маршрутизации между узлами, а когда соединение будет разорвано, информация о топологии будет обновлена, пакеты будут отсылаться по новому маршруту через узлы $n(6)$, $n(5)$ и $n(4)$.

```

# 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

```

Рис. 3.11: Добавление строки \$ns rtproto DV

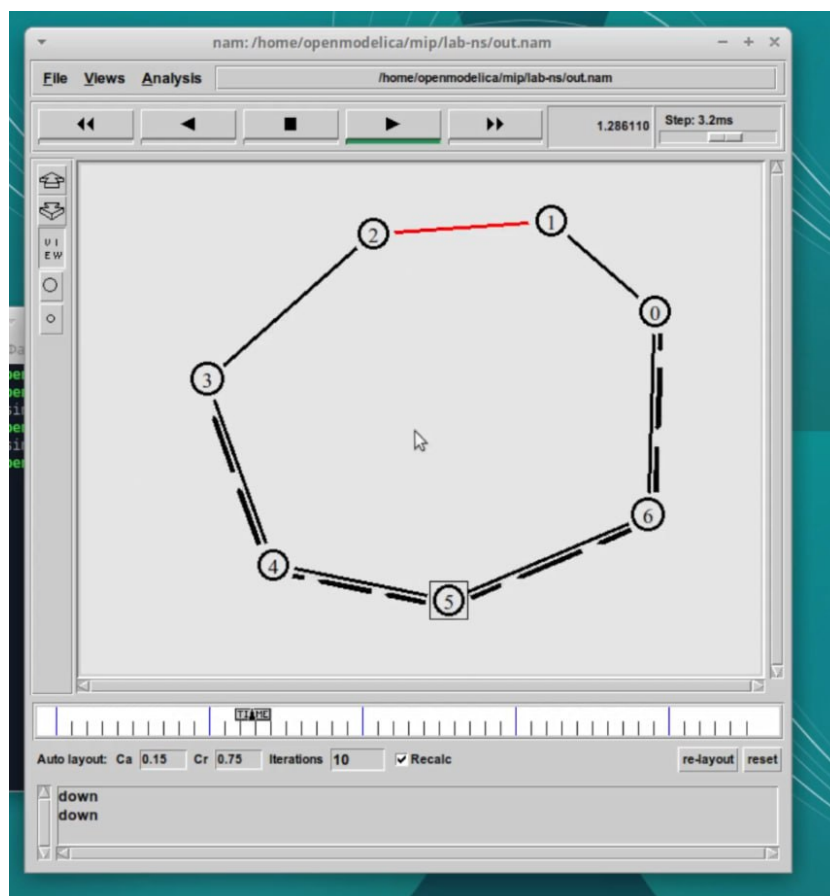


Рис. 3.12: Результат четвёртого примера после изменений

11. Выполним упражнение, для этого внесем следующие изменения в реализацию примера с кольцевой топологией сети:

- передача данных должна осуществляться от узла $n(0)$ до узла $n(5)$ по крат-

- чайшему 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. 3.13).

```

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]
$ns 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

```

Рис. 3.13: Упражнение

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.