

# **Отчёт по лабораторной работе №1**

**Простые модели компьютерной сети**

**Надежда Александровна Рогожина**

# 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>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

4.1	Рис. 1. Шаблон . . . . .	9
4.2	Рис. 2. Код . . . . .	10
4.3	Рис. 3. ns run . . . . .	10
4.4	Рис. 4. Сеть из 2х узлов . . . . .	11
4.5	Рис. 5. Сеть из 2х узлов . . . . .	12
4.6	Рис. 6. Первая программа . . . . .	13
4.7	Рис. 7. Реализация сети . . . . .	14
4.8	Рис. 8. Реализация сети . . . . .	15
4.9	Рис. 9. Маршрутизация пакетов . . . . .	16
4.10	Рис. 10. Сеть с кольцевой топологией . . . . .	17
4.11	Рис. 11. Изменение маршрутизации пакетов . . . . .	18
4.12	Рис. 12. Код упражнения . . . . .	19
4.13	Рис. 13. Код упражнения . . . . .	19
4.14	Рис. 14. Сеть из упражнения . . . . .	20
4.15	Рис. 15. Потеря пакетов . . . . .	20
4.16	Рис. 16. Изменение маршрутизации . . . . .	21

## **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 на основе кода, который дан в тексте лабораторной работы.
2. 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, основываясь на шаблоне, созданном в предыдущем пункте. От одного узла к другому по протоколу UDP осуществляется передача пакетов, размером 500 байт, с постоянной скоростью 200 пакетов в секунду.

### 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++.

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 функциональ-

ных 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 статистики, разрыв, либо восстановление соединений, реконфигурация топологии и т.д.). Для этого может использоваться метод `at`. Моделирование начинается при помощи метода `run`.



## 4 Выполнение лабораторной работы

В своём рабочем каталоге создайте директорию `mip` , к которой будут выполняться лабораторные работы. Внутри `mip` создайте директорию `lab-ns` , а в ней файл `shablon.tcl` :

```
mkdir -p mip/lab-ns  
cd mip/lab-ns  
touch shablon.tcl
```

Создание директории и файла шаблона (рис. 4.1).

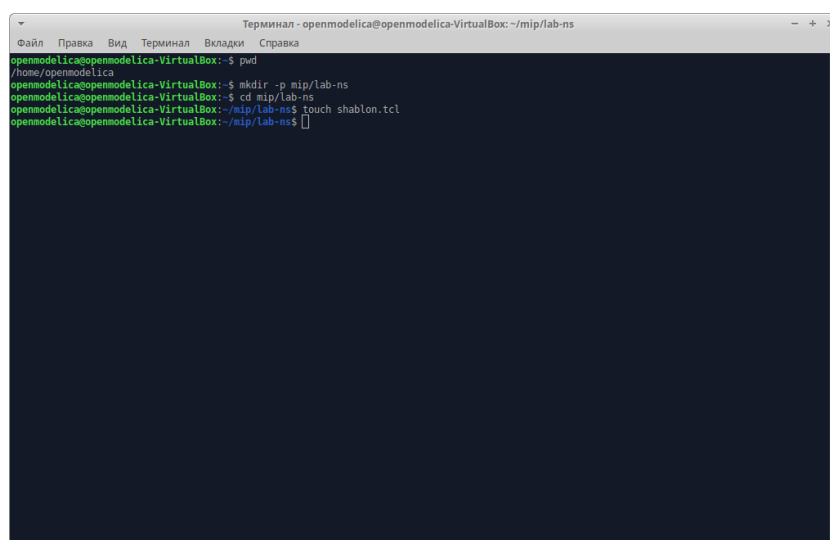


Рис. 4.1: Рис. 1. Шаблон

Далее, вводим сам код шаблона (рис. 4.2).

```

# 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. 4.2: Рис. 2. Код

Впервые запускаем nam (рис. 4.3).

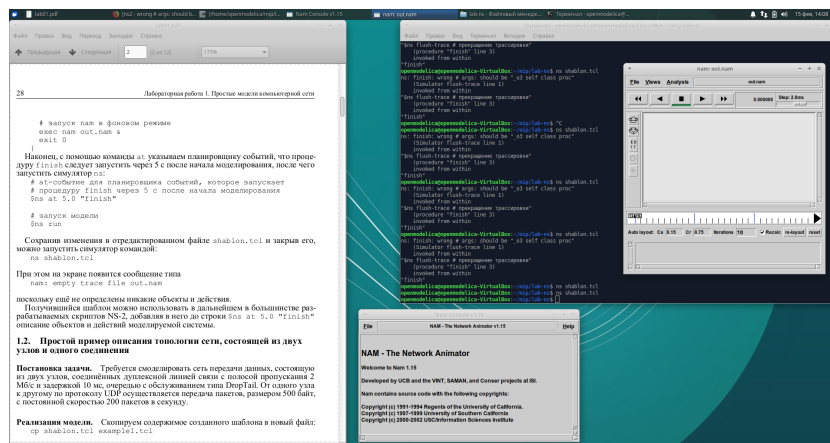
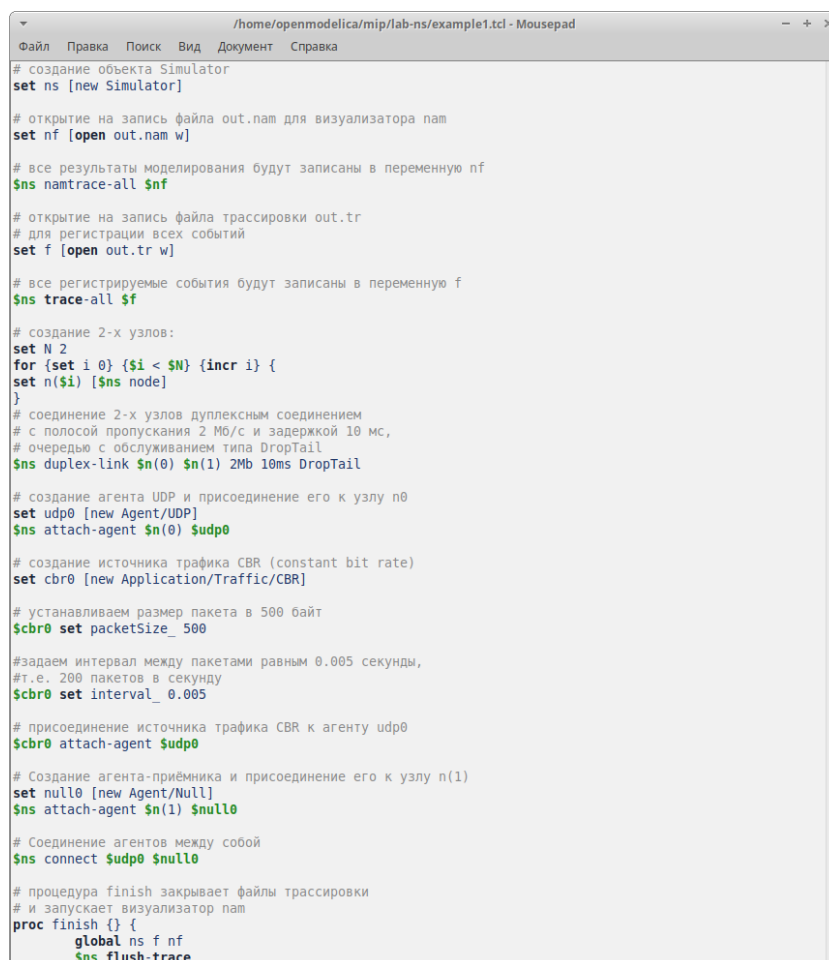


Рис. 4.3: Рис. 3. ns run

Далее, нам была поставлена задача реализовать примитивную модель сети, состоящей из 2 узлов:

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.

По коду, который был указан в тексте лабораторной работы, повторили данную сеть (рис. 4.4, рис. 4.5).



```

/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

# 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,
# очередью 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
$cbr0 set packetSize_ 500

# задаем интервал между пакетами равным 0.005 секунды,
# т.е. 200 пакетов в секунду
$cbr0 set interval_ 0.005

# присоединение источника трафика CBR к агенту udp0
$cbr0 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

# процедура finish закрывает файлы трассировки
# и запускает визуализатор nam
proc finish {} {
    global ns f nf
    $ns flush-trace
}
  
```

Рис. 4.4: Рис. 4. Сеть из 2х узлов

```

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

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
$cbr0 set packetSize_ 500

# задаем интервал между пакетами равным 0.005 секунды,
# т.е. 200 пакетов в секунду
$cbr0 set interval_ 0.005

# присоединение источника трафика CBR к агенту udp0
$cbr0 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

# 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
}

# запуск приложения через 0,5 с
$ns at 0.5 "$cbr0 start"

# остановка приложения через 4,5 с
$ns at 4.5 "$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. 4.5: Рис. 5. Сеть из 2х узлов**

**После этого, в консоли выполнили еще раз `ns run` (рис. 4.6).**

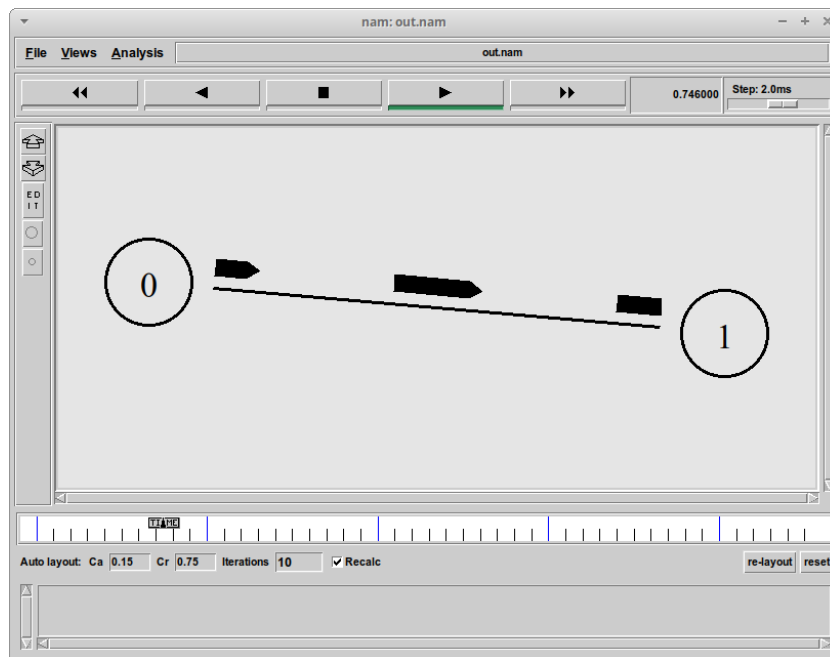


Рис. 4.6: Рис. 6. Первая программа

Здесь мы видим 2 узла (0 и 1) и соединяющую их дуплексную линию связи с полосой пропускания 2 Мб/с. Следующим заданием было реализовать чуть более сложную сеть, состоящую из 4 узлов (рис. 4.7, рис. 4.8): - 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; - TCP-источник на узле n0 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.



```
# 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

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
```

Рис. 4.7: Рис. 7. Реализация сети

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

$cbro set packetSize 500
$cbro set interval 0.005
$cbro 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

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

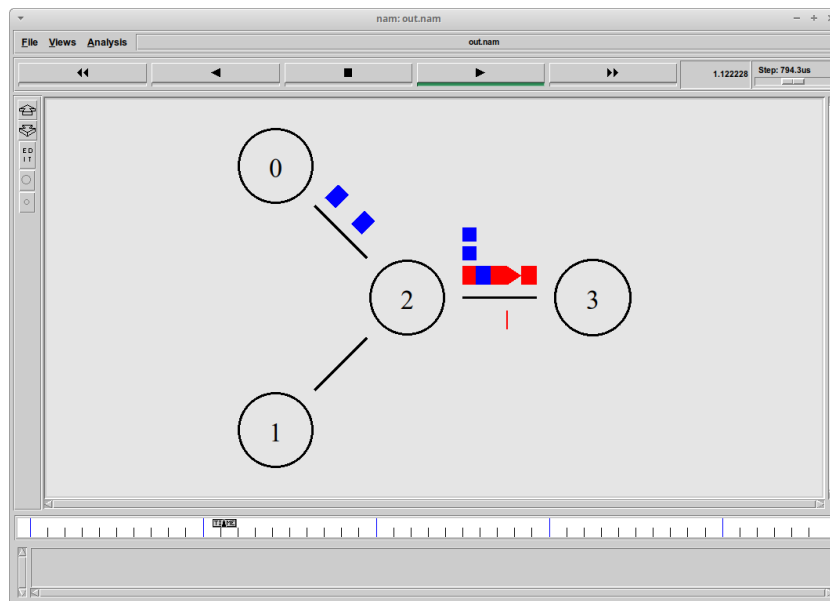
# процедура finish закрывает файлы трассировки
# и запускает визуализатор nam
proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

# 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. 4.8: Рис. 8. Реализация сети**

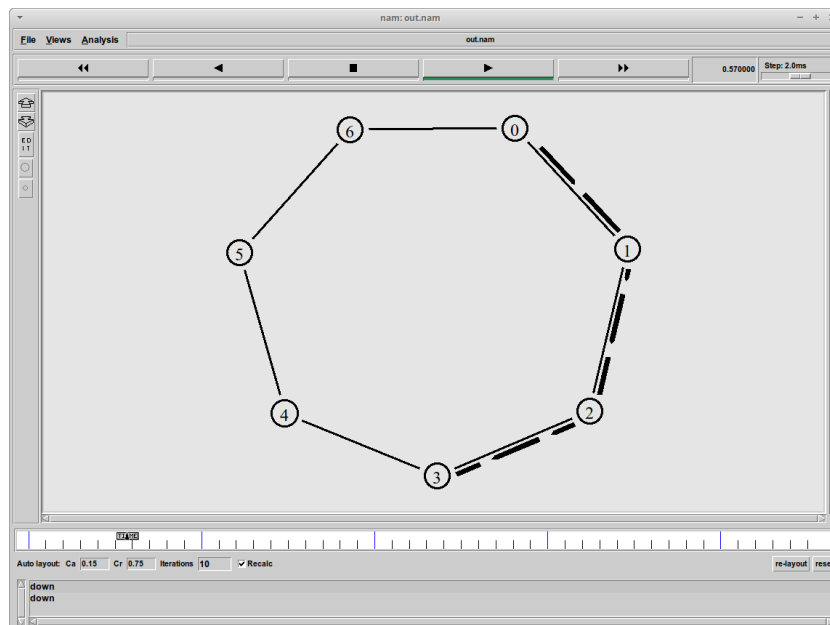
**В 1 секунду у нас заработали оба инициатора, маршрутизация пакетов выглядела следующим образом (рис. 4.9):**



**Рис. 4.9: Рис. 9. Маршрутизация пакетов**

Далее, по примеру из текста лабораторной работы был реализован пример сети с кольцевой топологией (рис. 4.10): - 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.





**Рис. 4.10: Рис. 10. Сеть с кольцевой топологией**

Сеть была настроена таким образом, что пакеты данных должны были ходить из 0 узла в 3 по кратчайшему пути. Также, для наглядности, мы указали отключение линии между 0 и 1 узлами на 1 секунду (с 1 по 2 сек). В момент времени, равный 1 с, пакеты данных пошли по пути 0-6-5-4-3, для достижения цели (рис. 4.11):

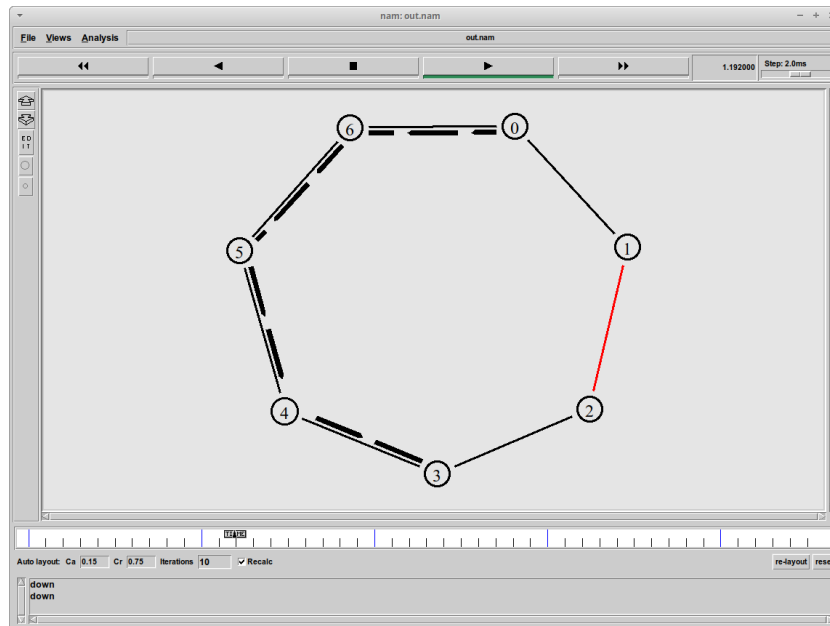


Рис. 4.11: Рис. 11. Изменение маршрутизации пакетов

Последним заданием было реализовать сеть с кольцево-линейной топологией (комбинация линейной и кольцевой топологий). Узлы с 0 по 4 должны были образовывать кольцо, 5 узел соединяться с 1. Также: - 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

Реализованный алгоритм (рис. 4.12, рис. 4.13).

```

# 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]
$ns namtrace-all $nf

# открытие на запись файла трассировки out.tr
# для регистрации всех событий
set f [open out.tr w]
$ns trace-all $f

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}]) 1Mb 10ms DropTail
    }

    set n5 [$ns node]
    $ns duplex-link $n(1) $n5 1Mb 10ms DropTail

    # создание агента TCP и присоединение его к узлу n(0)
    set tcp1 [new Agent/TCP/Newreno]
    $ns attach-agent $n(0) $tcp1

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

    # создание агента-получателя для tcp1
    set sink1 [new Agent/TCPSink/DelAck]
    $ns attach-agent $n5 $sink1
    $ns connect $tcp1 $sink1

    # 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
    }
}

$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 5.5 "finish"

# запуск модели
$ns run
```

Рис. 4.12: Рис. 12. Код упражнения

```

# для регистрации всех событий
set f [open out.tr w]
$ns trace-all $f

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}]) 1Mb 10ms DropTail
    }

    set n5 [$ns node]
    $ns duplex-link $n(1) $n5 1Mb 10ms DropTail

    # создание агента TCP и присоединение его к узлу n(0)
    set tcp1 [new Agent/TCP/Newreno]
    $ns attach-agent $n(0) $tcp1

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

    # создание агента-получателя для tcp1
    set sink1 [new Agent/TCPSink/DelAck]
    $ns attach-agent $n5 $sink1
    $ns connect $tcp1 $sink1

    # 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
    }
}

$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 5.5 "finish"

# запуск модели
$ns run
```

Рис. 4.13: Рис. 13. Код упражнения

После, запустив `ns run` в терминале, получили следующую сеть (рис. 4.14):

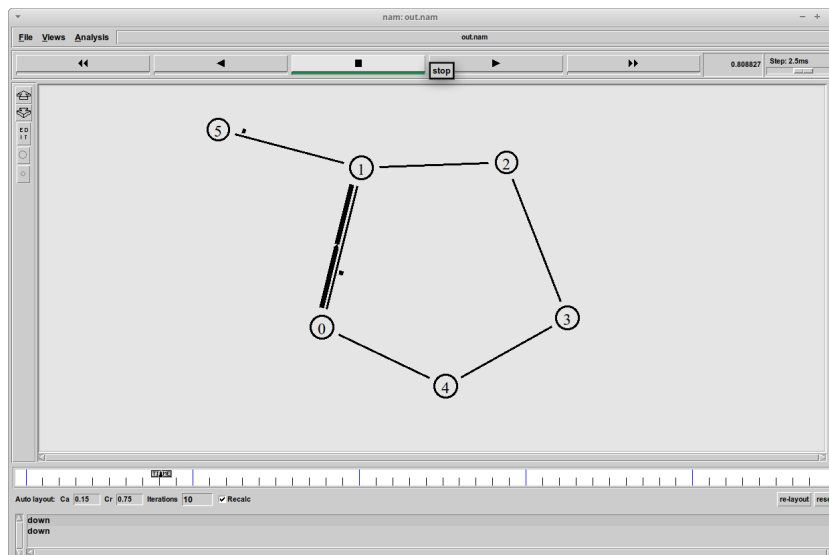


Рис. 4.14: Рис. 14. Сеть из упражнения

В момент времени, равный 1с, связь между 0 и 1 узлами разорвалась, в связи с чем некоторое количество пакетов было потеряно (рис. 4.15):

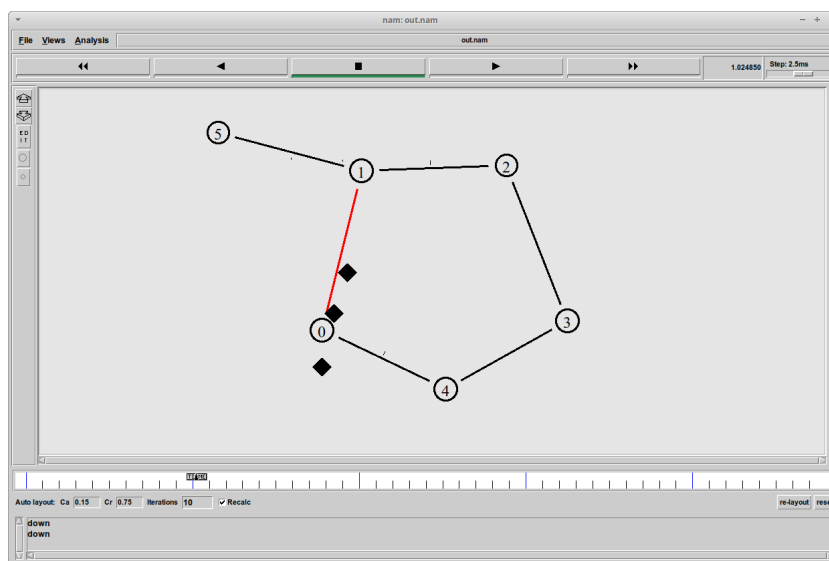
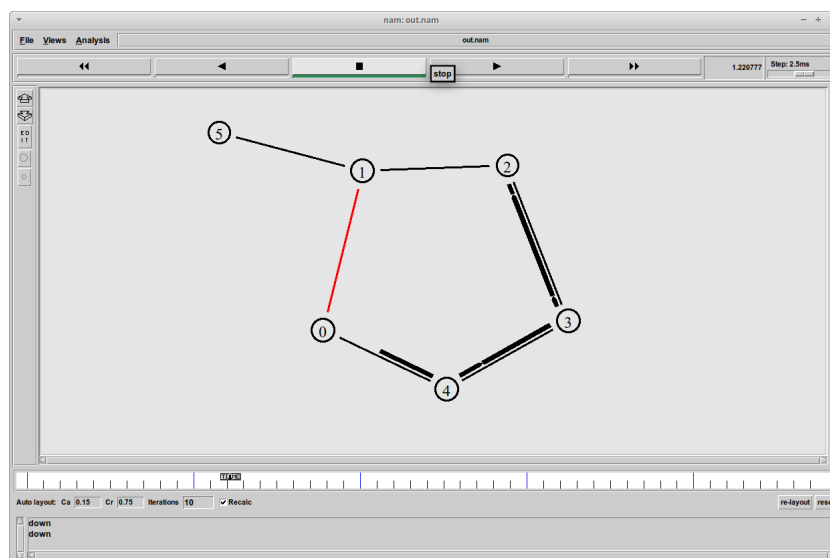


Рис. 4.15: Рис. 15. Потеря пакетов

Спустя несколько секунд реального времени, пакеты пошли по единственно существующему, а-к-а короткому пути (рис. 4.16):



**Рис. 4.16: Рис. 16. Изменение маршрутизации**

## **5 Выводы**

**В ходе выполнения лабораторной работы были получены навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также проведен анализ полученных результатов моделирования.**

## **Список литературы**