

# **Лабораторная работа 1**

**Простые модели компьютерной сети**

Извекова Мария Петровна

# Содержание

<b>Цель работы</b>	<b>5</b>
<b>Задание</b>	<b>6</b>
<b>Теоретическое введение</b>	<b>7</b>
<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>Выводы</b>	<b>22</b>
<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

1	Создание директории . . . . .	9
2	Создание файла . . . . .	9
3	Скрипт . . . . .	10
4	Шаблон . . . . .	11
5	Запуск скрипта . . . . .	11
6	Реализация модели . . . . .	12
7	Сама реализация . . . . .	13
8	Скрипт . . . . .	14
9	Скрипт . . . . .	14
10	Скрипт . . . . .	15
11	Скрипт . . . . .	16
12	Скрипт . . . . .	16
13	Скрипт . . . . .	17
14	Реализация . . . . .	18
15	Скрипт . . . . .	19
16	Реализация . . . . .	20
17	Реализация . . . . .	21

## **Список таблиц**

## **Цель работы**

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

## **Задание**

1. Создание шаблона сценария для NS-2
2. Построение топологии сети из двух узлов и одного соединения
3. Построение усложненной топологии сети
4. Построение кольцевой топологии сети

# 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.

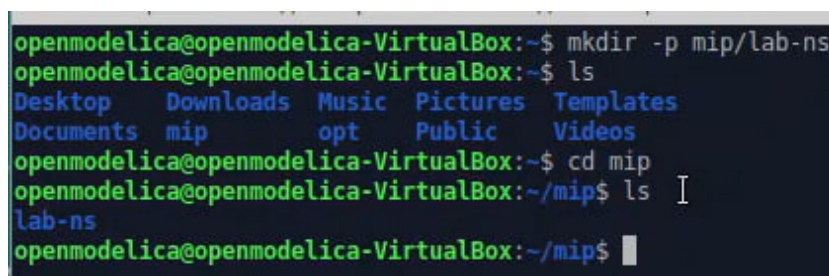
Объекты типа NODE – `$node id` – возвращает идентификатор узла; – `$node neighbors` – возвращает список соседних узлов; – `$node attach agent` – прикрепляет агент типа `agent` к узлу; – `$node detach agent` – отменяет прикрепление агента типа `agent` к узлу; – `$node agent port` – возвращает ссылку на агента, прикрепленного к порту `port` на данном узле, или пустую строку, если порт не используется. – `$node reset port` – отменяет прикрепление всех агентов на данном узле и реинициализирует все переменные, связанные с агентами данного узла. – `$node join-group agent group` – добавляет объект, определяемый объектной ссылкой `agent` для многопользовательской группы с адресом `group`. Это также приводит к выделению соответствующего многопользовательского трафика протоколом групповой работы для обеспечения работы агента `agent`. – `$node allocaddr` – возвращает адреса в многопользовательской группе в возрастающем порядке для каждого соединения, начиная

с `0x8000` и заканчивая `0xFFFF`.



## 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`: (рис. [fig:001] - [fig:002]).



```
openmodelica@openmodelica-VirtualBox:~$ mkdir -p mip/lab-ns
openmodelica@openmodelica-VirtualBox:~$ ls
Desktop  Downloads  Music  Pictures  Templates
Documents  mip      opt    Public   Videos
openmodelica@openmodelica-VirtualBox:~$ cd mip
openmodelica@openmodelica-VirtualBox:~/mip$ ls
lab-ns
openmodelica@openmodelica-VirtualBox:~/mip$
```

Рис. 1: Создание директории

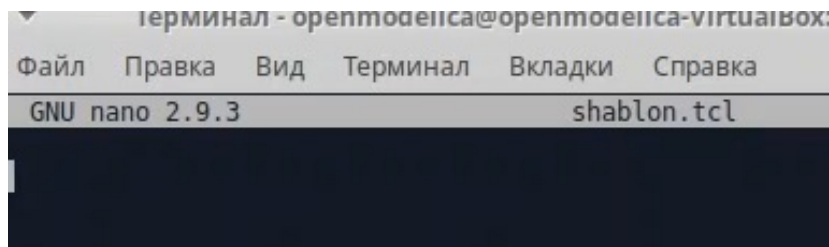


Рис. 2: Создание файла

Создаем скрипт, с помощью которого будет создаваться шаблон, который 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. [fig:003] - [fig:004])

```

# 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: Скрипт

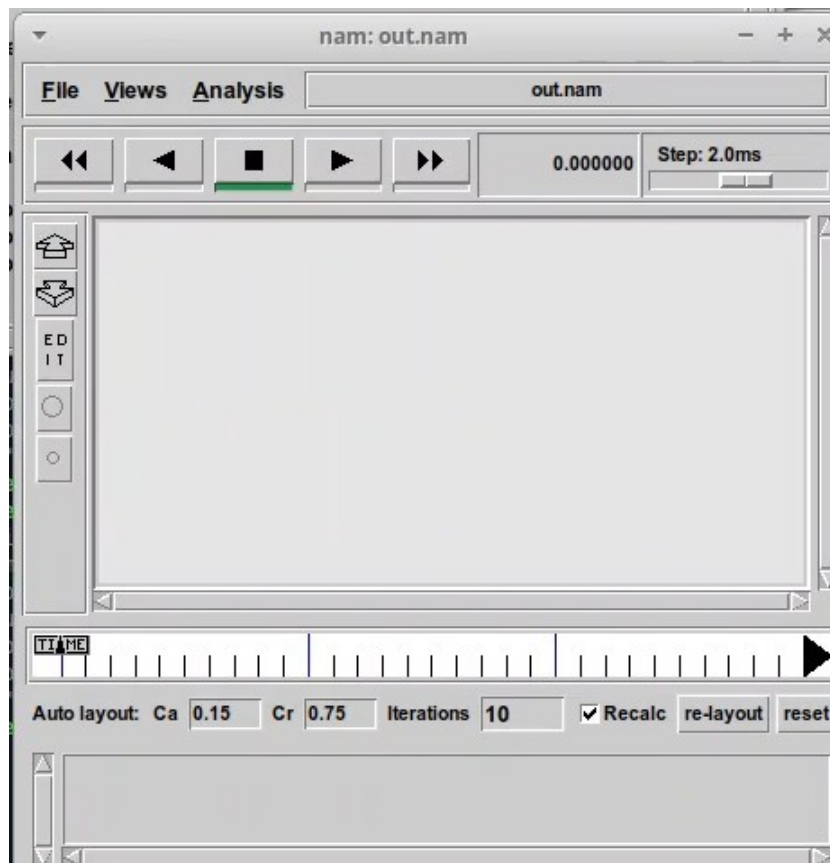


Рис. 4: Шаблон

```

openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ns example1.tcl
ns: finish: wrong # args: should be "_o3 self class proc"
(Simulator flush-trace line 1)
invoked from within
"$ns flush-trace # прекращение трассировки"
(procedure "finish" line 3)
invoked from within
"finish"
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ nam out.nam

```

Рис. 5: Запуск скрипта

Запускаем файл `out.nam` с помощью команды `nam`, сам скрипт запускается с помощью команды `ns`

Моделируем 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. (рис. [-@fig:005])

```

# 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
$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
# 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"

```

Рис. 6: Реализация модели

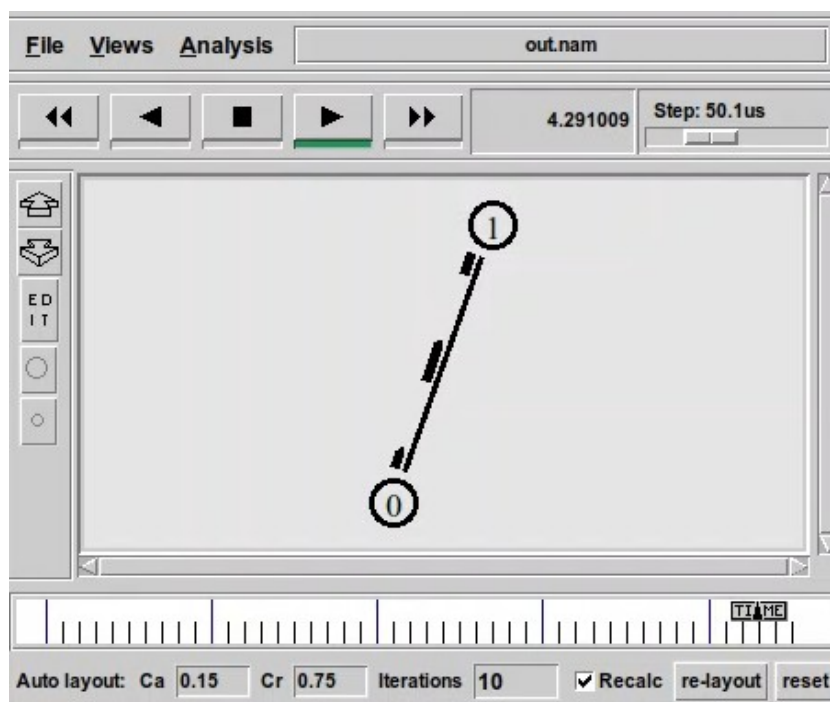


Рис. 7: Сама реализация

На картинке [-@fig:006] можно увидеть два узла, между которыми происходит передача данных.

Усложняем нашу цепь: добавляем два узла, - 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;

```

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. 8: Скрипт

```

# 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]

```

Рис. 9: Скрипт

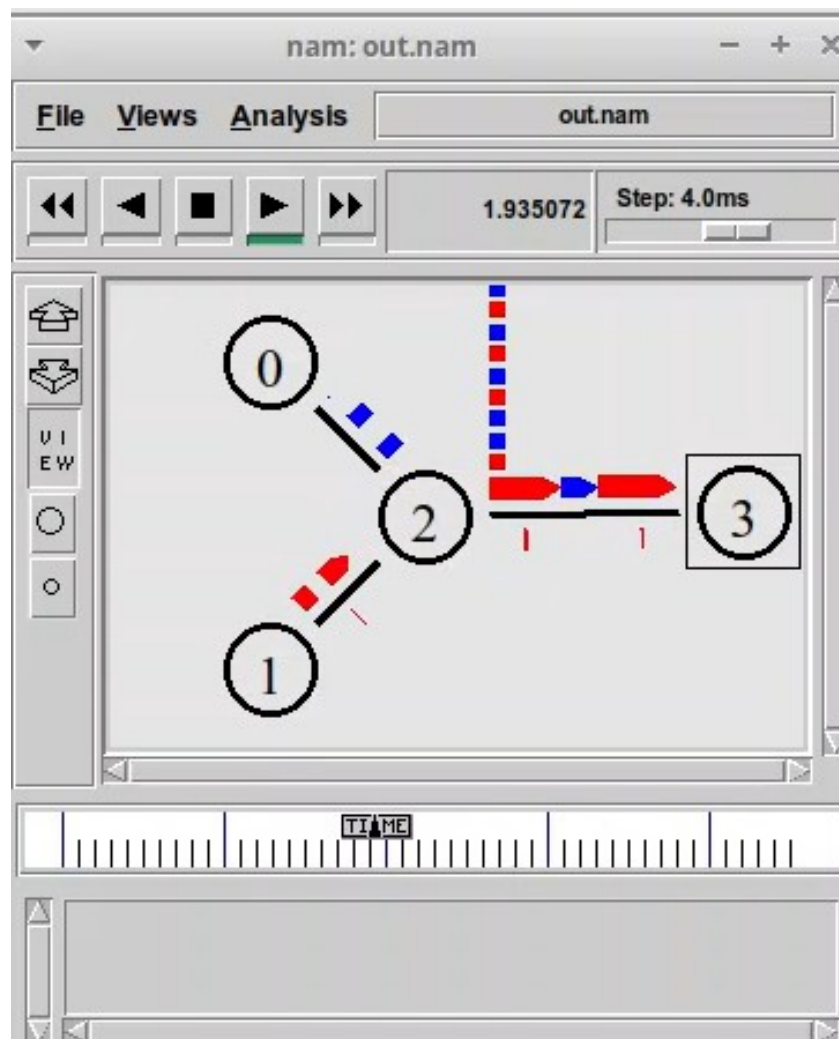


Рис. 10: Скрипт

Так как из узлов 0 и 1 выходят данные одновременно, после прохождения узла 2 данные теряются, так как соединение 2-3 имеет полосу лишь 1 Mb, когда из узлов 0 и 1 суммарно идет 1.6 Mb

Кольцевая топология



```

$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

```

Рис. 11: Скрипт

```

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

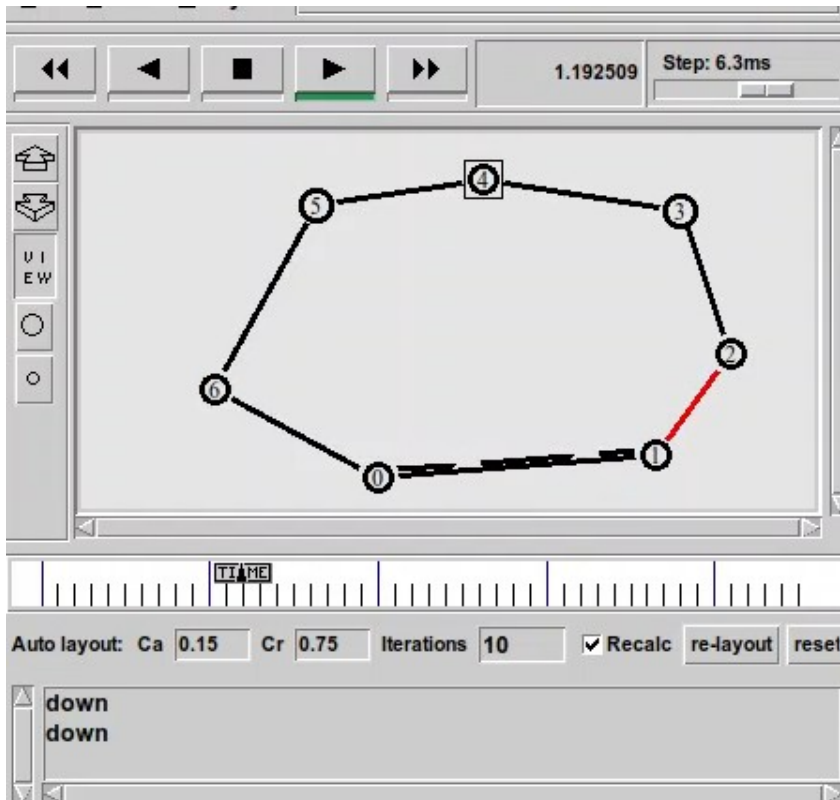
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"

```

Рис. 12: Скрипт

У нас получается передача данных от узла 0 к узлу 3. Происходит потеря данных и узел 1-2 обрывается, данные передаются от 0 к 1





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

```
set ns [new Simulator]
$ns rtproto DV
# открытие на запись файла out.pcap для визуализатора pcap
```

Рис. 13: Скрипт

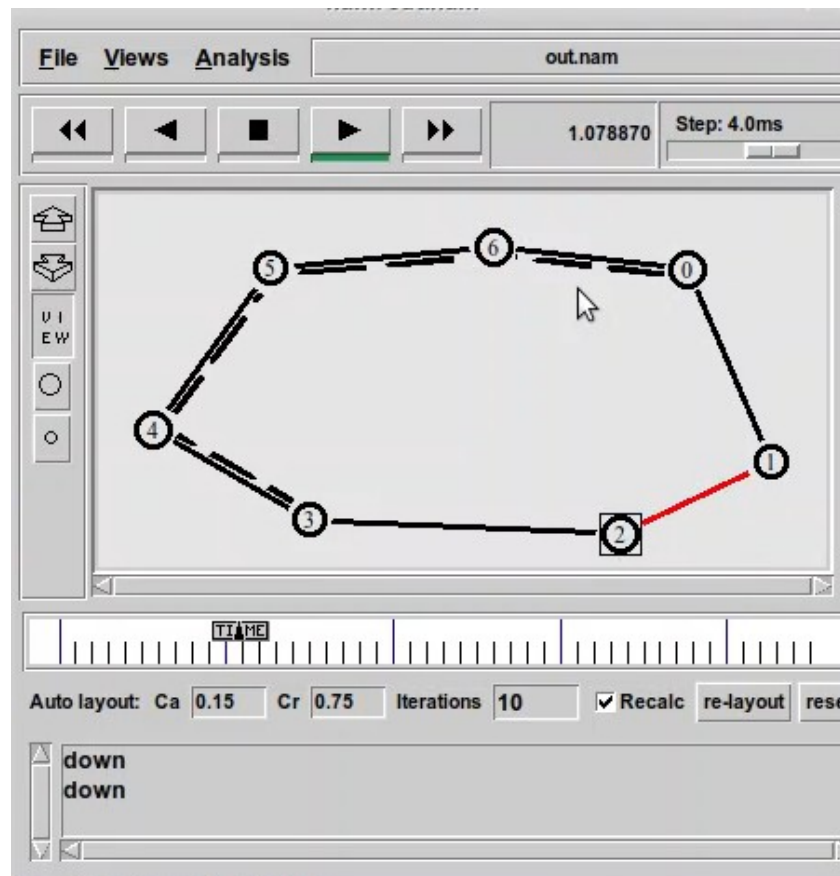


Рис. 14: Реализация

Задание Необходимо построить сеть из 6 узлов, где будет кольцевая и один внешний узел

```

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"
# 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. 15: Скрипт

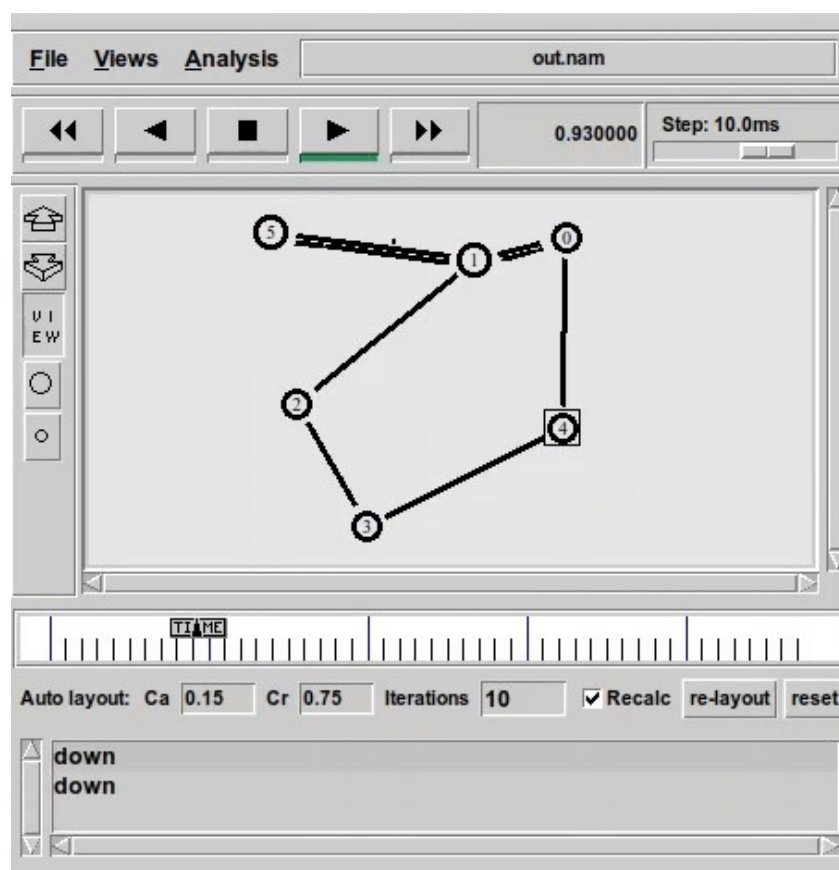


Рис. 16: Реализация

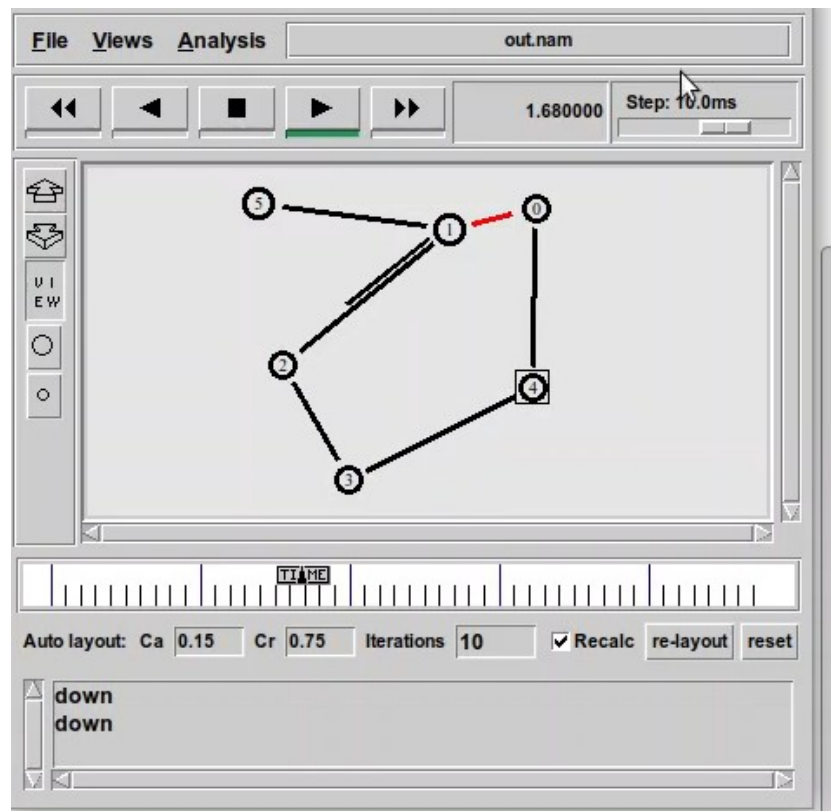


Рис. 17: Реализация

Видим что на рисунке 17 у нас данные от 0 до 5 идут по кратчайшему пути. После обрыва сети один меняют свое направление

## **Выводы**

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

# **Список литературы**

::: Теоретические сведения по имитационному моделированию часть 1