

Лабораторная работа 1

Простые модели компьютерной сети

Шияпова Дарина Илдаровна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
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 усложнённой топологией сети	10
3.4	Пример с кольцевой топологией сети	14
4	Выводы	21

Список иллюстраций

3.1	Создание директорий и файла	6
3.2	Редактирование файла shablon.tcl	7
3.3	Запуск шаблона сценария для NS-2	8
3.4	Пример описания топологии сети, состоящей из двух узлов и одного соединения	9
3.5	Визуализация простой модели сети с помощью nam	10
3.6	Визуализация простой модели сети с помощью nam	11
3.7	Описание усложненной топологии сети	12
3.8	Описание усложненной топологии сети	13
3.9	Описание усложненной топологии сети	14
3.10	Описание кольцевой топологии сети и динамической маршрутизацией пакетов	15
3.11	Передача данных по кратчайшему пути сети с кольцевой топологией	16
3.12	Передача данных по сети с кольцевой топологией в случае разрыва соединения	16
3.13	Передача данных по изменённой кольцевой топологии сети	18
3.14	Передача данных по сети в случае разрыва соединения	19
3.15	Передача данных после восстановления соединения	20

1 Цель работы

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

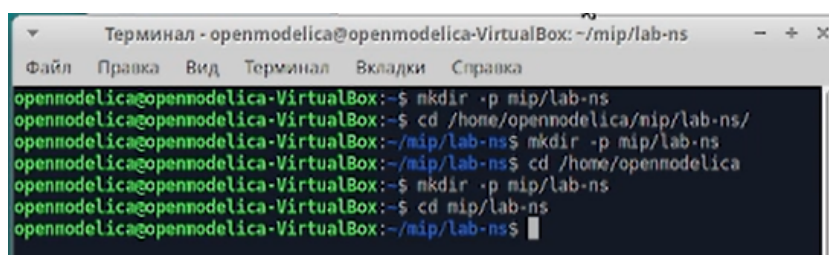
2 Задание

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

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).

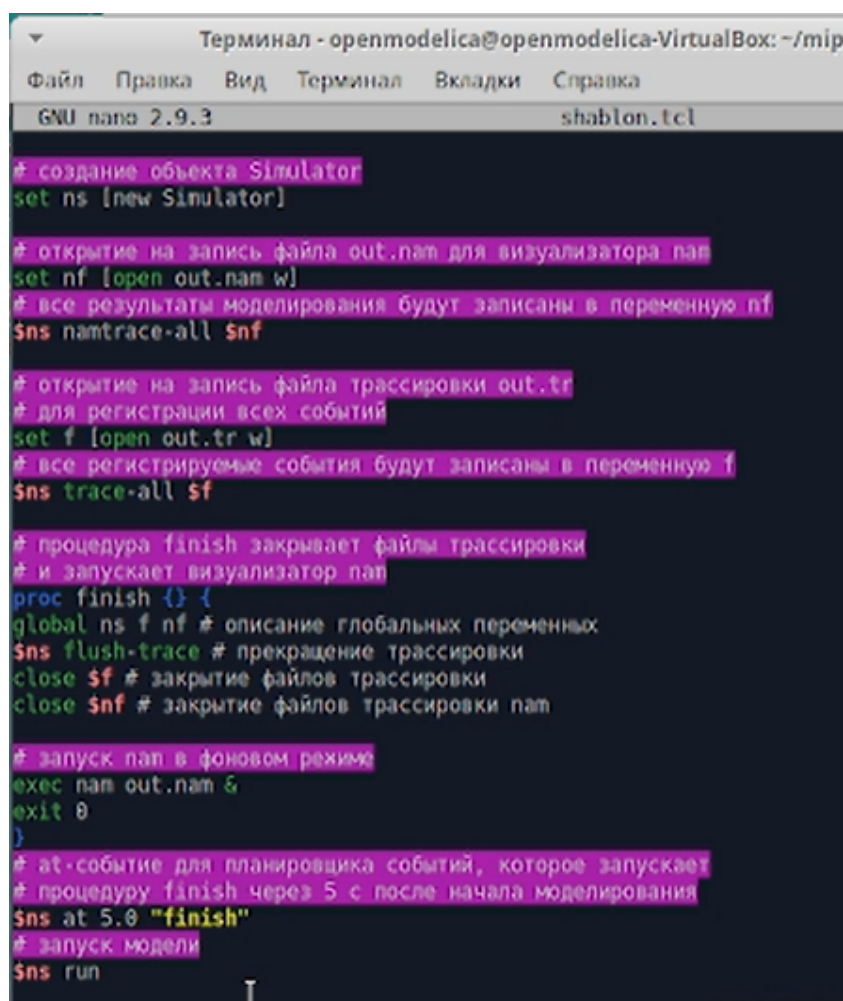


```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
Файл  Правка  Вид  Терминал  Вкладки  Справка
openmodelica@openmodelica-VirtualBox:~$ mkdir -p mip/lab-ns
openmodelica@openmodelica-VirtualBox:~$ cd /home/openmodelica/mip/lab-ns/
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ mkdir -p mip/lab-ns
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cd /home/openmodelica
openmodelica@openmodelica-VirtualBox:~$ mkdir -p mip/lab-ns
openmodelica@openmodelica-VirtualBox:~$ cd mip/lab-ns
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$
```

Рис. 3.1: Создание директорий и файла

Откроем на редактирование файл `shablon.tcl` (рис. 3.2).

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`. С помощью команды `at` указываем планировщику событий, что процедуру `finish` запустим через 5 с после начала моделирования, после чего запустим симулятор `ns`.



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip
Файл  Правка  Вид  Терминал  Вкладки  Справка
GNU nano 2.9.3  shablon.tcl

# 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 файла shablon.tcl

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).

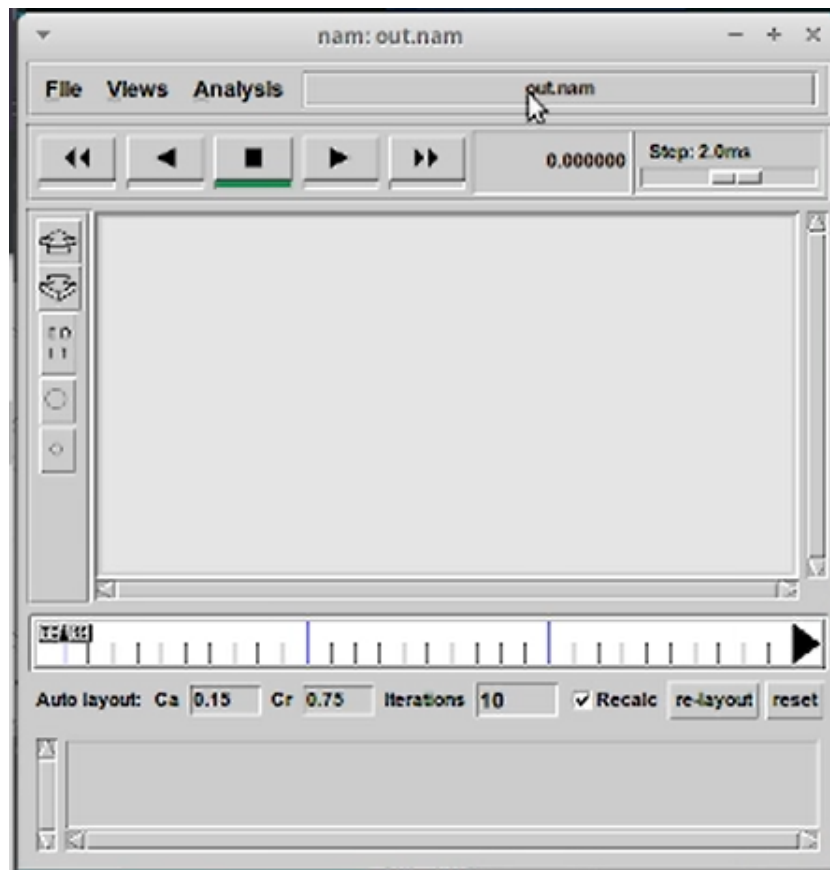


Рис. 3.3: Запуск шаблона сценария для NS-2

3.2 Простой пример описания топологии сети, состоящей из двух узлов и одного соединения

Требуется смоделировать сеть передачи данных, состоящую из двух узлов, соединённых дуплексной линией связи с полосой пропускания 2 Мб/с и задержкой 10 мс, очередь 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.

Скопируем содержимое созданного шаблона в новый файл: `cp shablon.tcl example1.tcl` и откроем `example1.tcl` на редактирование. Добавим в него до строки `$ns at 5.0 "finish"` описание топологии сети. Создадим агенты для

генерации и приёма трафика. Создается 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: $R = \frac{500 \cdot 8}{0.005} = 800000$ /.

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. 3.4).

```
# 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
```

Рис. 3.4: Пример описания топологии сети, состоящей из двух узлов и одного соединения

Сохранив изменения в отредактированном файле и запустив симулятор, получим в качестве результата запуск аниматора `nam` в фоновом режиме (рис. 3.5).

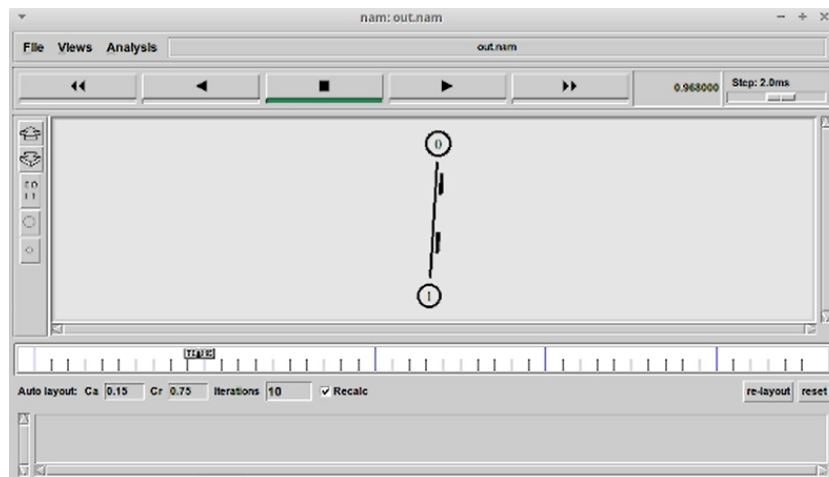


Рис. 3.5: Визуализация простой модели сети с помощью nam

При нажатии на кнопку play в окне nam через 0.5 секунды из узла 0 данные начнут поступать к узлу 1.

3.3 Пример с усложнённой топологией сети

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 ТСП-агент может генерировать, равняется 1KByte)
- ТСП-приёмник генерирует и отправляет АСК 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.

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

```
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. 3.6: Визуализация простой модели сети с помощью `nam`

Создадим агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP (рис. 3.7).

```

# 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

```

Рис. 3.7: Описание усложненной топологии сети

Создадим агенты-получатели. Соединим агенты udp0 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. 3.8).

```

# 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]
$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

```

Рис. 3.8: Описание усложненной топологии сети

Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования (рис. 3.9).

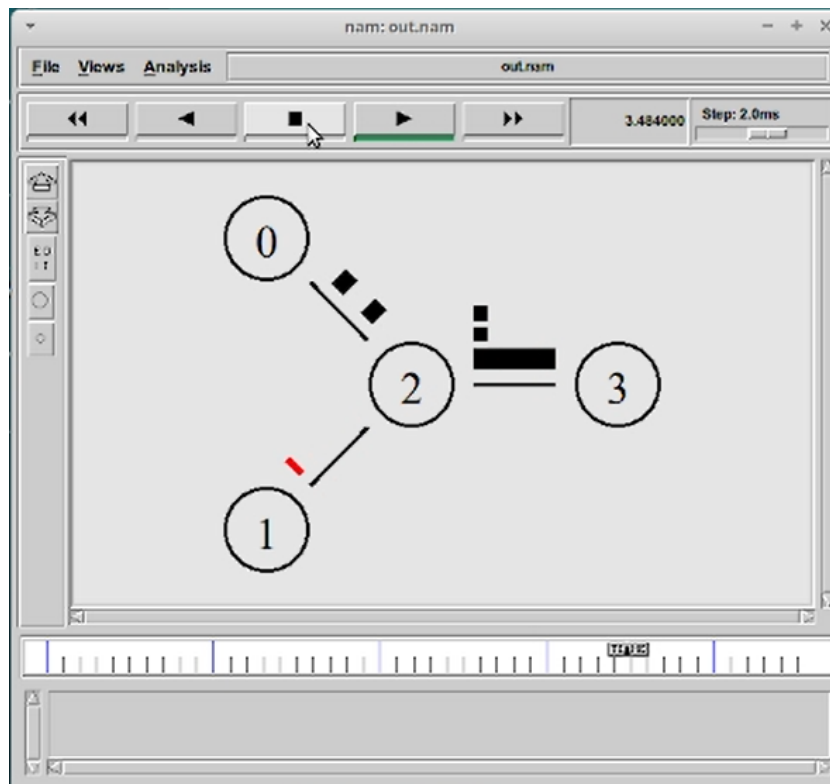


Рис. 3.9: Описание усложненной топологии сети

3.4 Пример с кольцевой топологией сети

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.

Скопируем содержимое созданного шаблона в новый файл: `cp shablon.tcl example3.tcl` и откроем `example3.tcl` на редактирование. Опишем топологию

моделируемой сети (рис. 3.10). 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. 3.11). Добавим команду разрыва соединения между узлами $n(1)$ и $n(2)$ на время в одну секунду, а также время начала и окончания передачи данных.

```
set N 7
for {set i 0} {$i < $N} {incr i} {
  set n($i) [$ns node]
}
#Далее соединим узлы так, чтобы создать круговую топологию:
for {set i 0} {$i < $N} {incr i} {
  $ns duplex-link $n($i) $n([expr ($i+1)%$N]) 1Mb 10ms DropTail
}

# at-событие для планировщика событий, которое запускает
# процедуру finish через 5 с после начала моделирования
$ns at 5.0 "finish"
```

Рис. 3.10: Описание кольцевой топологии сети и динамической маршрутизацией пакетов

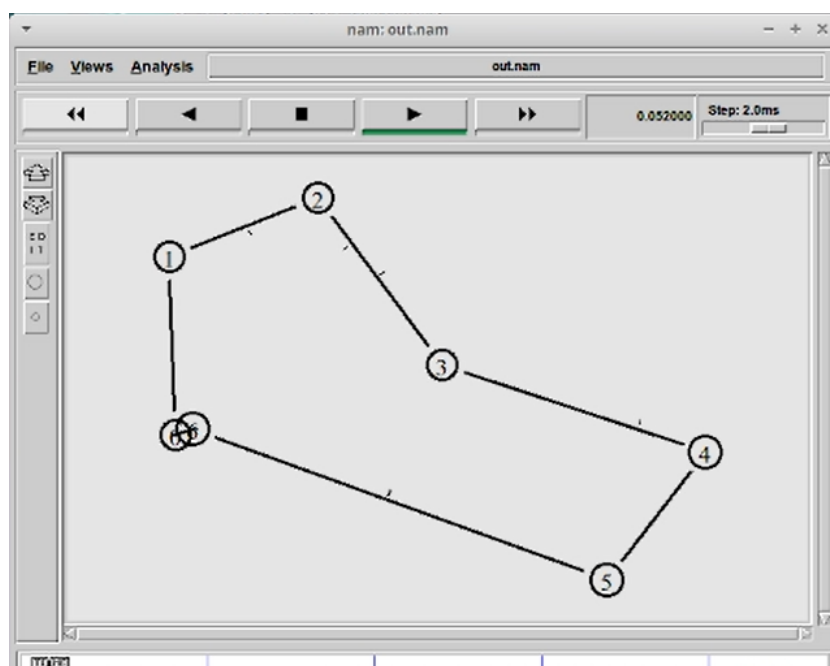


Рис. 3.11: Передача данных по кратчайшему пути сети с кольцевой топологией

Передача данных при кольцевой топологии сети в случае разрыва соединения представлена на рис. 3.12.

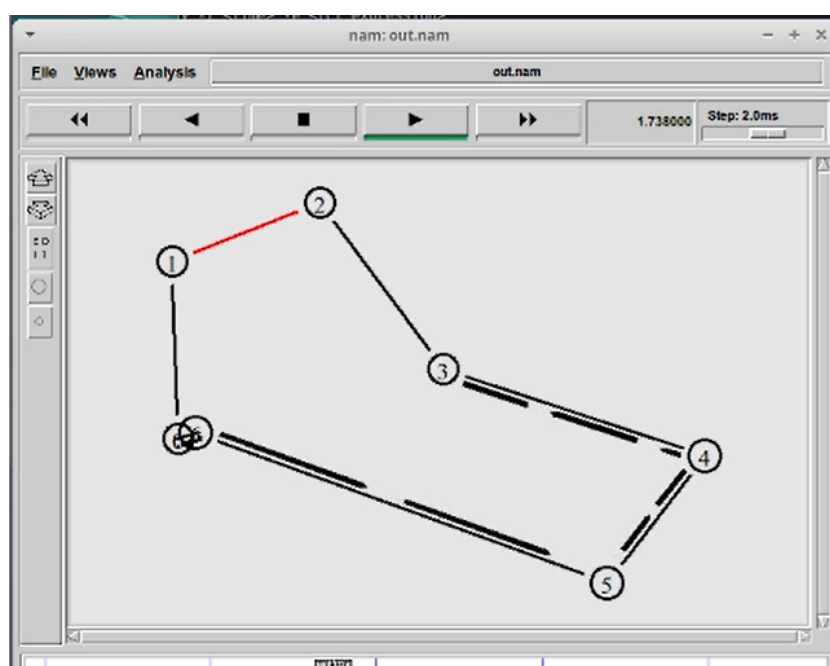


Рис. 3.12: Передача данных по сети с кольцевой топологией в случае разрыва соединения

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. Когда соединение будет разорвано, информация о топологии будет обновлена, и пакеты будут отсылаться по новому маршруту через узлы n(6), n(5) и n(4).

Упражнение

Внесем следующие изменения в реализацию примера с кольцевой топологией 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.

Изменим количество узлов в кольце на 5, а 6 узел n(5) отдельно присоединим к узлу n(1). Вместо агента UDP создадим агента TCP (типа Newreno), а на принимающей стороне используем TCPSink-объект типа DelAck; поверх TCP работает протокол FTP с 0,5 до 4,5 секунд модельного времени. Также зададим с 1 по 2 секунду модельного времени разрыв соединения между узлами n(0) и n(1).

Запустим программу и увидим, что пакеты идут по кратчайшему пути через узел n(1) (3.13).

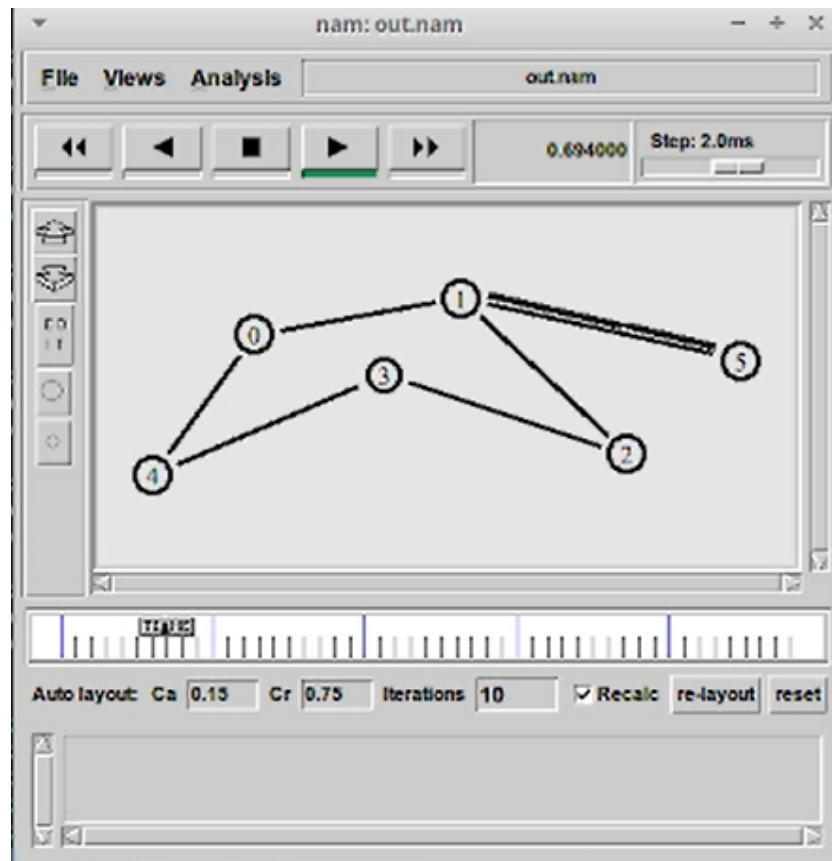


Рис. 3.13: Передача данных по изменённой кольцевой топологии сети

При разрыве соединения часть пакетов теряется, но поскольку данные обновляются пакеты начинают идти по другому пути (3.14).

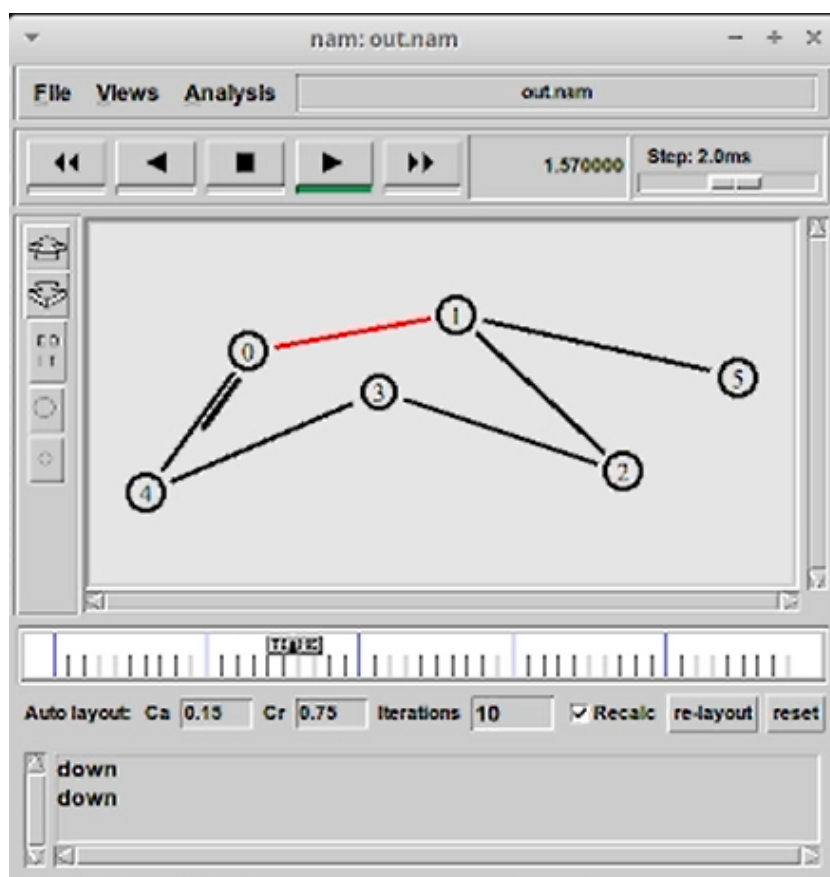


Рис. 3.14: Передача данных по сети в случае разрыва соединения

После восстановления соединения пакеты снова идут по кратчайшему пути (3.15).

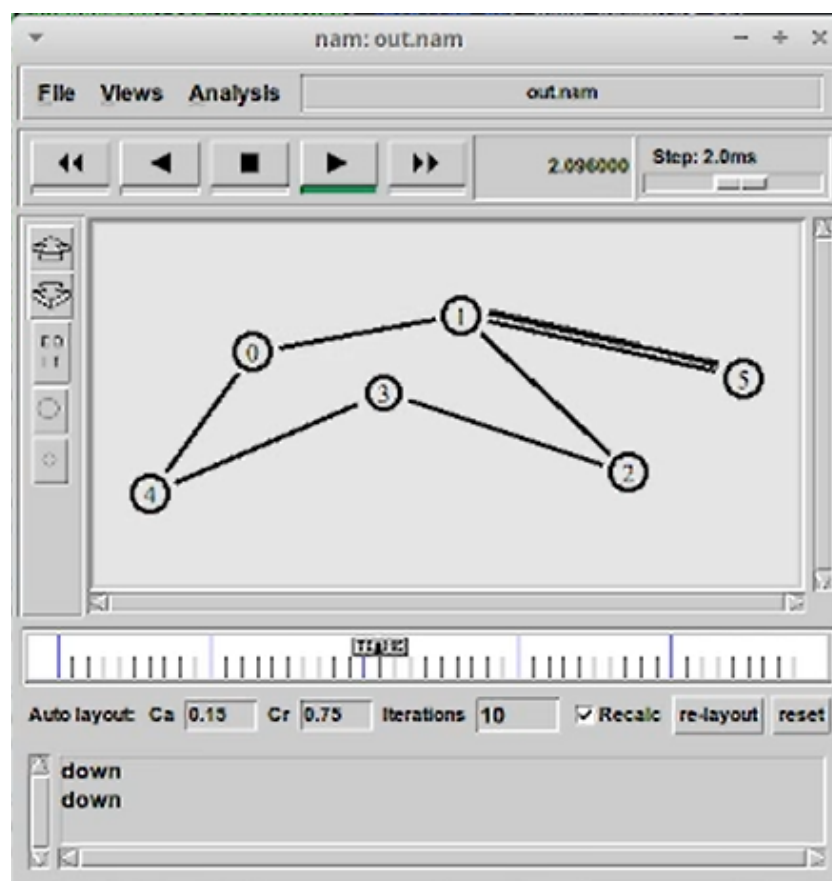


Рис. 3.15: Передача данных после восстановления соединения

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.