

Лабораторная работа 1

Простые модели компьютерной сети

Демидова Екатерина Алексеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
4	Шаблон сценария для NS-2	7
5	Простой пример описания топологии сети, состоящей из двух узлов и одного соединения	10
6	Пример с усложнённой топологией сети	13
7	Пример с кольцевой топологией сети	16
8	Упражнение	20
9	Выводы	25

Список иллюстраций

4.1	Шаблон сценария для NS-2	8
4.2	Запуск шаблона сценария для NS-2	9
5.1	Пример описания простой топологии сети	11
5.2	Визуализация простой модели сети с помощью nam	12
6.1	Пример описания усложненной топологии сети	14
6.2	Мониторинг очереди в визуализаторе nam	15
7.1	Пример с кольцевой топологией сети	17
7.2	Передача данных по кратчайшему пути сети с кольцевой топологией	18
7.3	Передача данных по сети с кольцевой топологией в случае разрыва соединения	19
8.1	Код для упражнения по построению топологии сети	21
8.2	Передача данных по кратчайшему пути сети	22
8.3	Разрыв соединения	23
8.4	Передача данных по сети в случае разрыва соединения	24

1 Цель работы

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

2 Задание

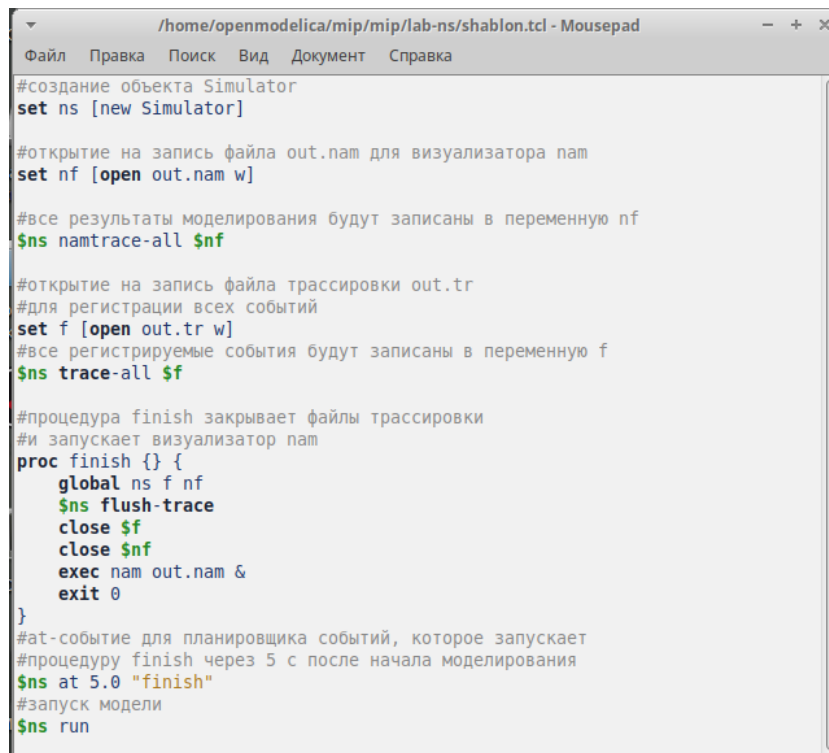
- Создать шаблон сценария для NS-2
- Рассмотреть простой пример описания топологии сети, состоящей из двух узлов и одного соединения
- Рассмотреть пример с усложнённой топологией сети
- Рассмотреть пример с кольцевой топологией сети
- Выполнить упражнение

3 Выполнение лабораторной работы

4 Шаблон сценария для NS-2

В своём рабочем каталоге создадим директорию `tmp`, в которой будут выполняться лабораторные работы. Внутри `tmp` создадим директорию `lab-ns`, а в ней файл `shablon.tcl`. В него запишем шаблон для программ в NS-2.

Создадим объект типа `Simulator`. Затем создадим переменную `nf` и укажем, что требуется открыть на запись `nam`-файл для регистрации выходных результатов моделирования и дадим 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`(рис. [4.1]).



```

/home/openmodelica/mip/mip/lab-ns/shablon.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
}

#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. 4.1: Шаблон сценария для NS-2

Запустив файл шаблона увидим пустую область моделирования(рис. [4.2]).

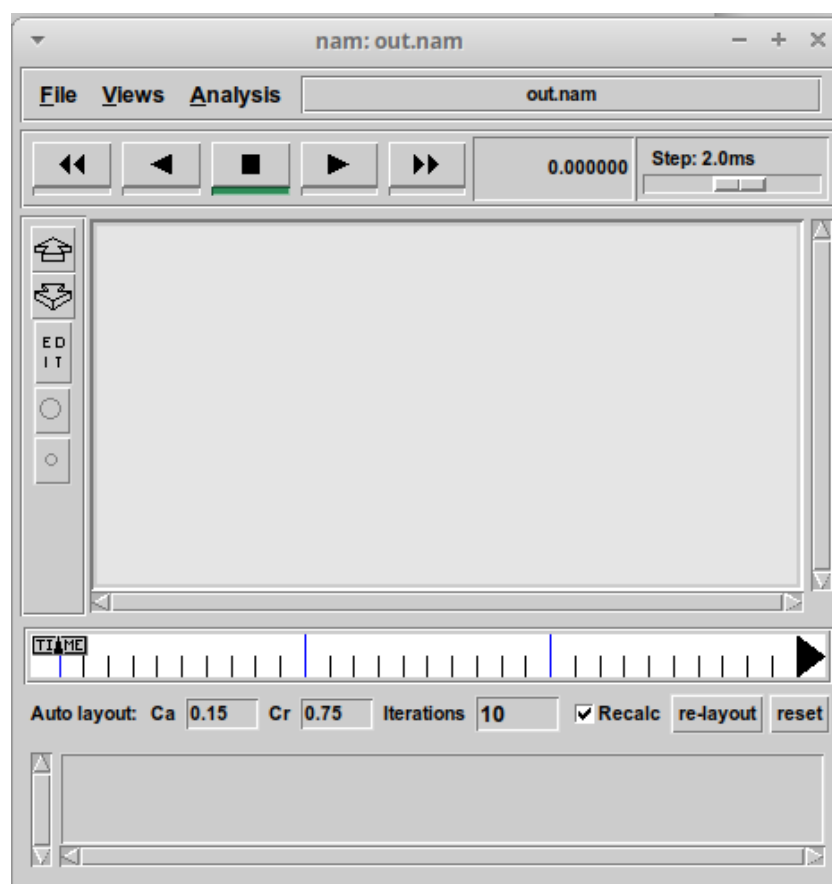


Рис. 4.2: Запуск шаблона сценария для NS-2

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

Смоделируем 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. Создадим узлы n0, к котором прикрепим агент UDP с приложением CBR(источник с постоянной скоростью), и n1, к которому прикрепим Null-агент, который работает как приёмник трафика. И соединим эти агенты между собой(рис. [5.1]).

```

/home/openmodelica/mip/mip/lab-ns/example.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка
}
# 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"

$ns at 5.0 "finish"
#запуск модели
$ns run

```

Рис. 5.1: Пример описания простой топологии сети

При нажатии на кнопку play в окне nam через 0.5 секунды из узла 0 данные начнут поступать к узлу 1(рис. [5.2]).

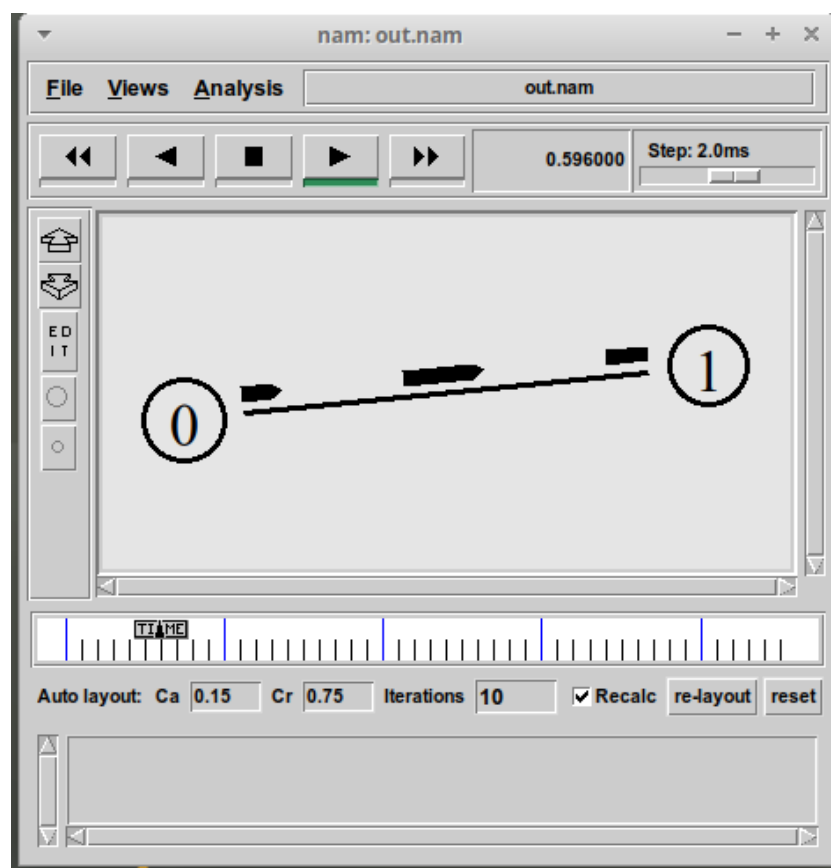



Рис. 5.2: Визуализация простой модели сети с помощью nam

6 Пример с усложнённой топологией сети

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 n_3 (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.

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. Также создадим агентов-получателей и соединим агенты `udp0` и `tcp1` и их получателей. Зададим описание цвета каждого потока и введём отслеживание очереди, ограничение на размер очереди(рис.

[6.1]).



```
set N 4
for {set i 0} {$i < $N} {incr i} {
  set n($i) [create 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"
```

Рис. 6.1: Пример описания усложненной топологии сети

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 Мб, а суммарная – 1,6 Мб. Но соединение $n(2) - n(3)$ имеет полосу лишь 1 Мб. Следовательно, часть пакетов должна теряться. В окне аниматора можно видеть пакеты в очереди, а также те пакеты, которые отбрасываются при переполнении (рис. [6.2]).

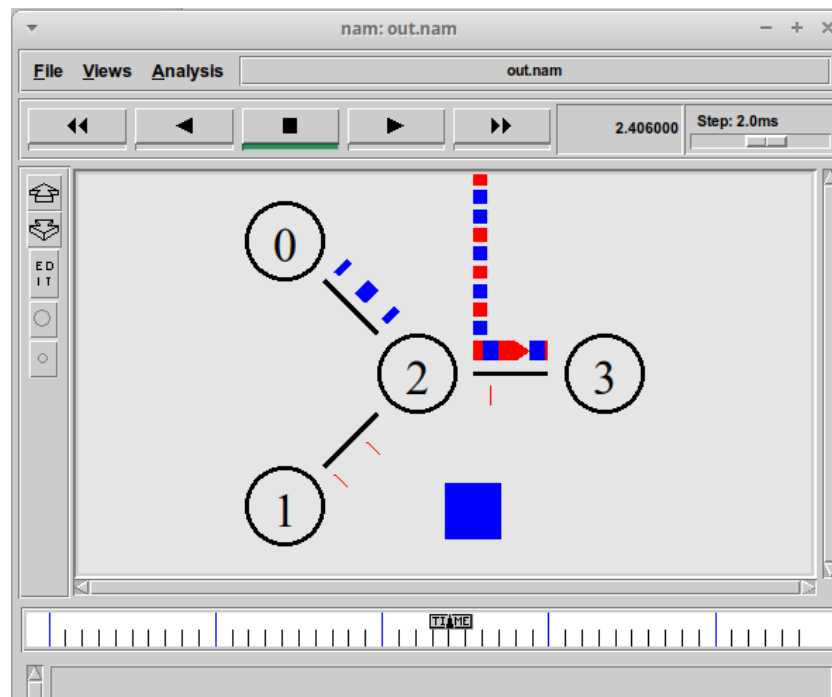


Рис. 6.2: Мониторинг очереди в визуализаторе nam

7 Пример с кольцевой топологией сети

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.

Создадим семь узлов и соединим их в форме кольца. Зададим передачу данных от узла $n(0)$ к узлу $n(3)$ с помощью UDP агента и источника CBR. Добавим команду разрыва соединения между узлами $n(1)$ и $n(2)$ на время в одну секунду, а также время начала и окончания передачи данных. Добавим в начало скрипта после команды создания объекта Simulator, благодаря этому сразу после записки в сети отправляется небольшое количество маленьких пакетов, используемых для обмена информацией, необходимой для маршрутизации между узлами(рис. [7.1]).


```

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. 7.1: Пример с кольцевой топологией сети

При запуске можно увидеть, что пакеты идут по кратчайшему пути через узлы n(1) и n(2)(рис. [7.2]).

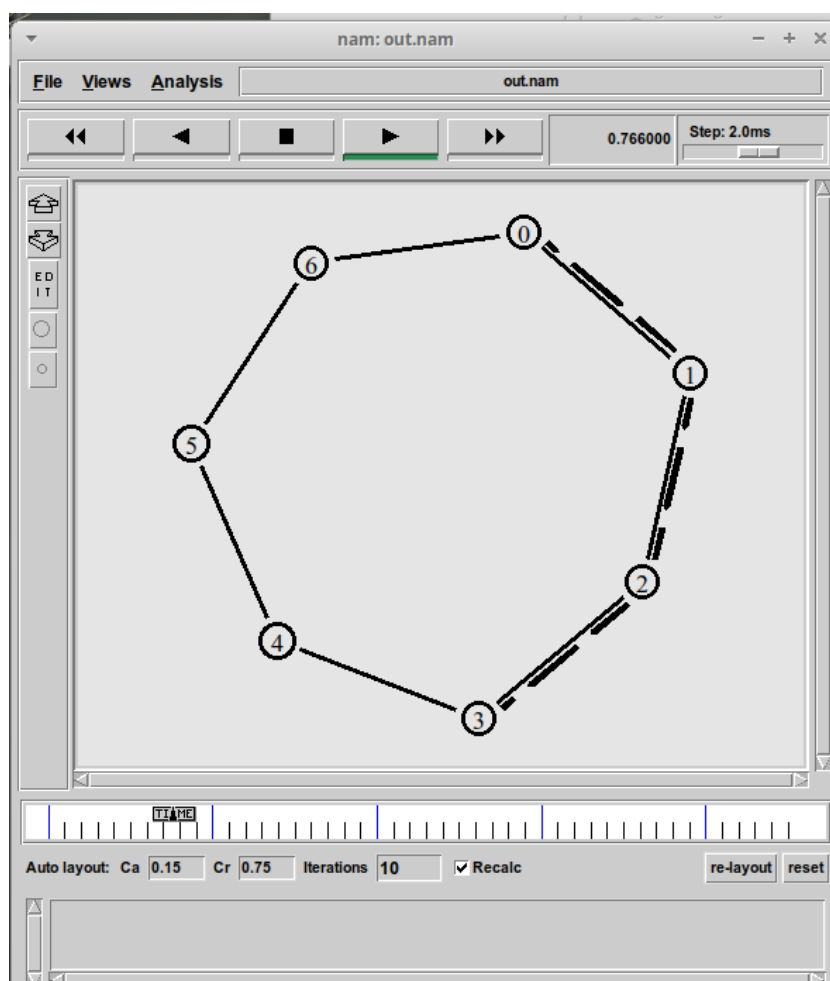


Рис. 7.2: Передача данных по кратчайшему пути сети с кольцевой топологией

Можно увидеть, что при разрыве часть пакетов теряется. Когда соединение разорвано, информация о топологии обновляется, и пакеты отсылаются по новому маршруту через узлы $n(6)$, $n(5)$ и $n(4)$ (рис. [7.3]).

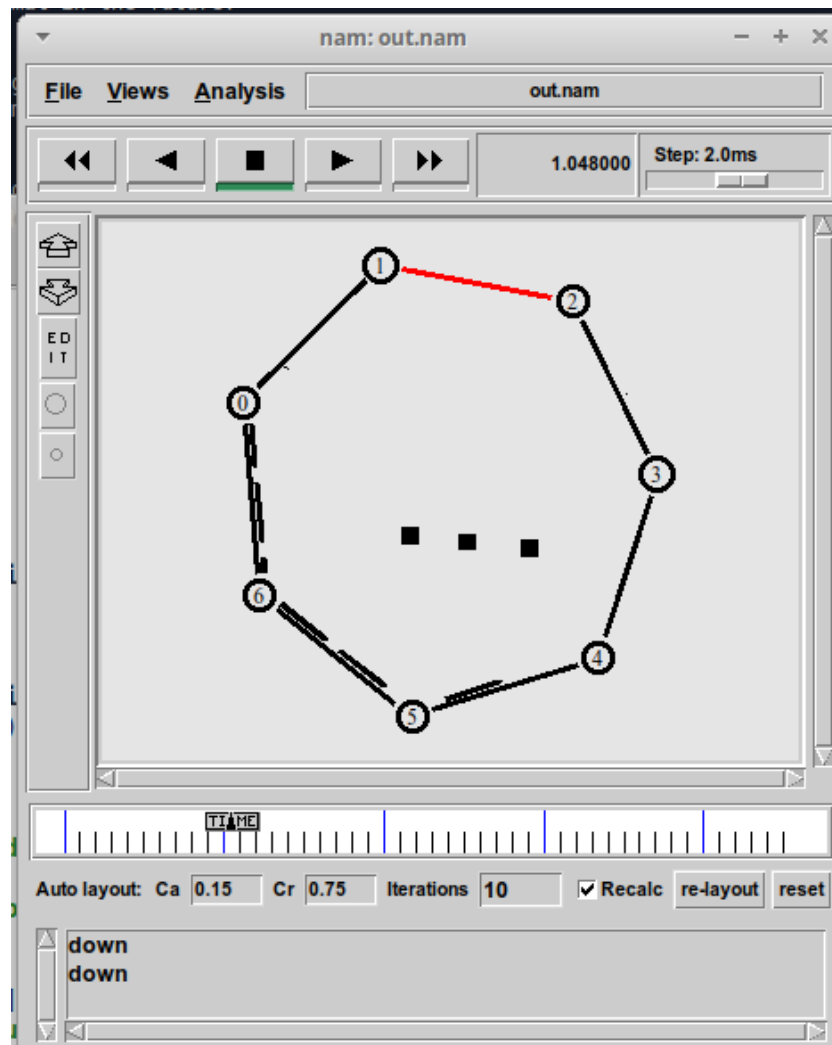


Рис. 7.3: Передача данных по сети с кольцевой топологией в случае разрыва соединения

8 Упражнение

Упражнение Внесите следующие изменения в реализацию примера с кольцевой топологией сети: - топология сети должна соответствовать представленной на рис. 1.7; - 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 колце на 5, а 6 узел $n(5)$ отдельно присоединим к узлу $n(1)$. Вместо агента UDP создадим агента 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 Также зададим с 1 по 2 секунду модельного времени разрыв соединения между узлами $n(0)$ и $n(1)$ (рис. [8.1]).

```
/home/openmodelica/mip/mip/lab-ns/example4.tcl - Mousepad
Файл Правка Поиск Вид Документ Справка

close $nf
exec nam out.nam &
exit 0
}

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

set tcp1 [new Agent/TCP/Newreno]
$ns attach-agent $n(0) $tcp1
# создание приложения FTP
# и присоединение его к агенту tcp1
set ftp [new Application/FTP]
$ftp attach-agent $tcp1

# создание агента-получателя для tcp1
set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(5) $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. 8.1: Код для упражнения по построению топологии сети

При запуске можно увидеть, что пакеты идут по кратчайшему пути через узел n(1)(рис. [8.2]).

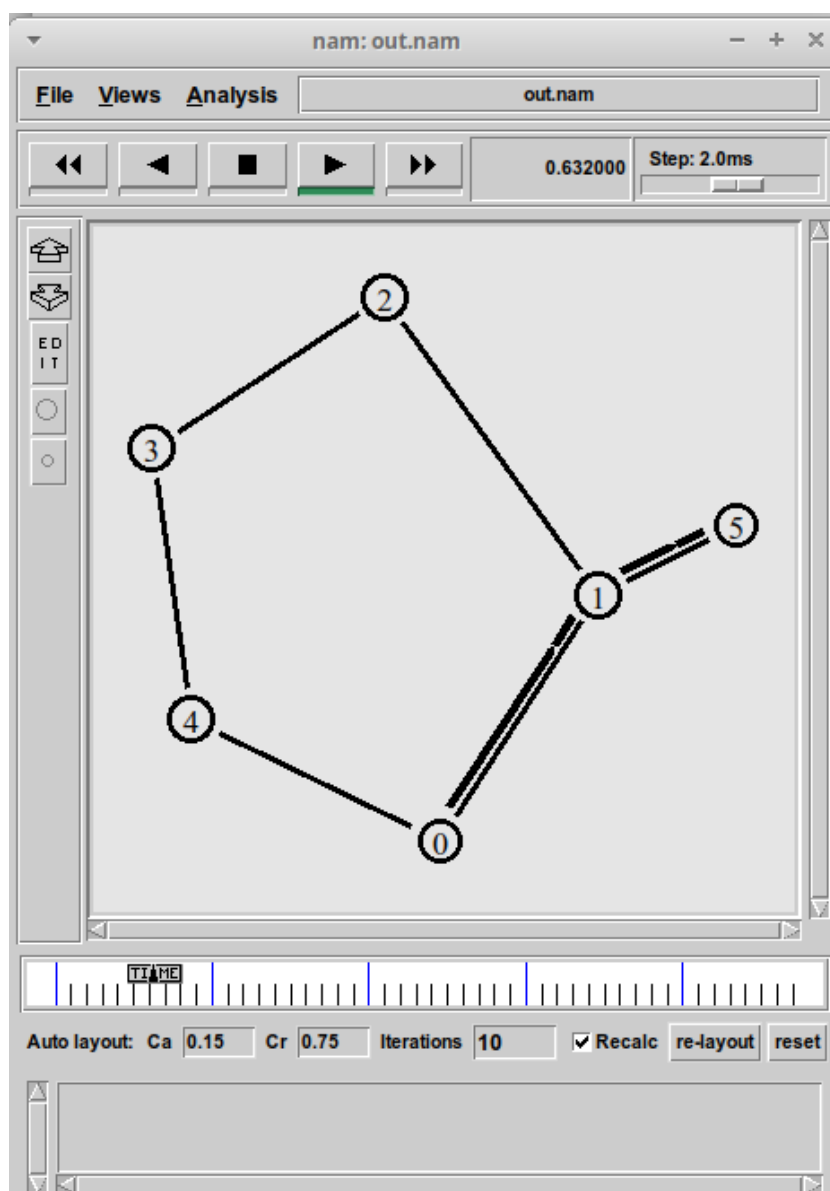


Рис. 8.2: Передача данных по кратчайшему пути сети

При разрыве соединения часть пакетов теряется(рис. [8.3]).

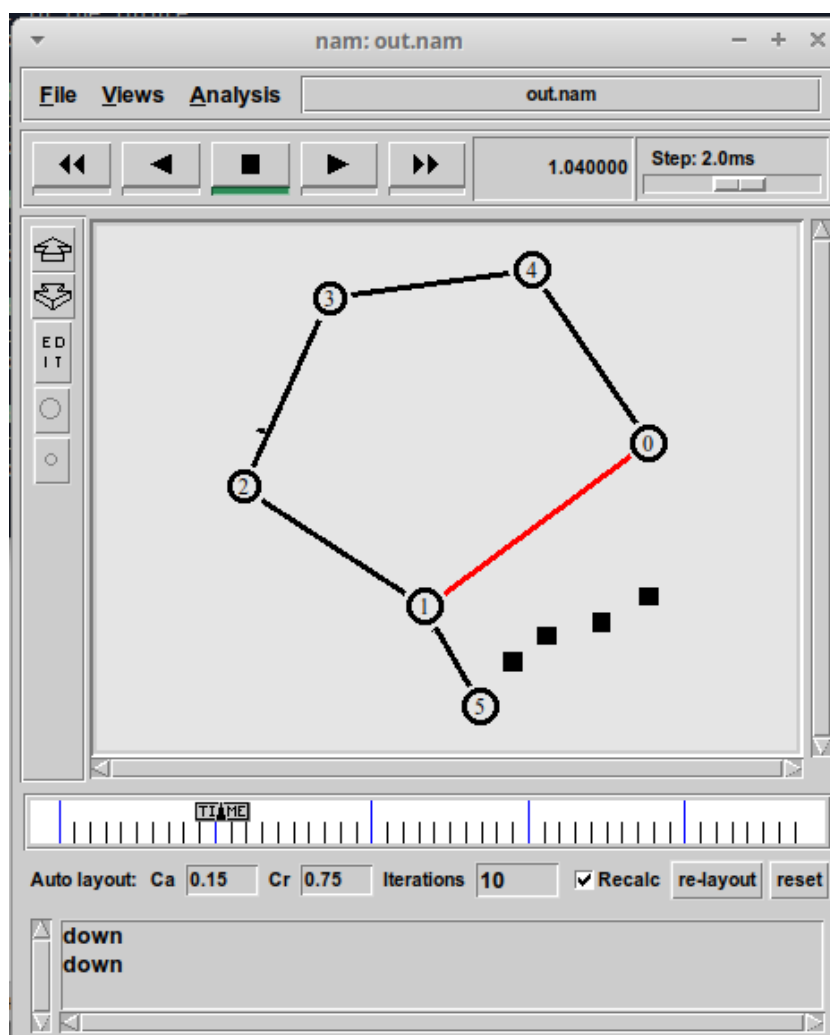


Рис. 8.3: Разрыв соединения

Когда соединение разорвано, информация о топологии обновляется, и пакеты отсылаются по новому маршруту через узлы $n(4)$, $n(3)$, $n(2)$ и $n(1)$ (рис. [8.4]).

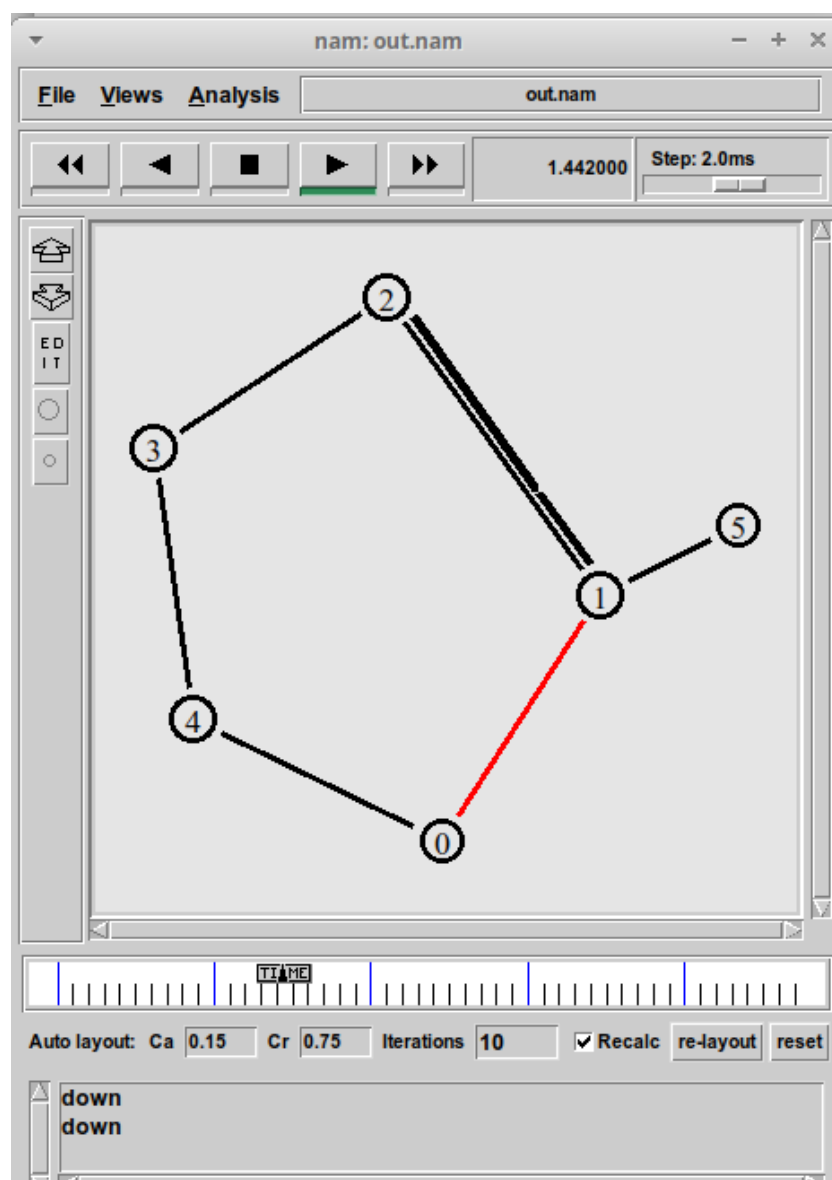


Рис. 8.4: Передача данных по сети в случае разрыва соединения

9 Выводы

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