Простые модели компьютерной сети

Лабораторная работа №1

Баулин Егор Александрович, учебная группа: НКНбд-01-18

Содержание

Список иллюстраций

Цель работы

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

Задание

- Создать шаблон сценария NS-2
- Смоделировать сеть с простой топологией, состоящей из двух узлов и одного соединения
- Смоделировать сеть с усложненной топологией сети
- Смоделировать сеть с круговой топологией сети
- Выполнить упражнение, внесением изменений в сеть с круговой топологией

Выполнение лабораторной работы

Создание шаблона сценария NS-2

- В рабочем каталоге создал директорию mip для лабораторных работ. Внутри создал директорию lab-ns, а в ней файл shablon.tlc при помощи команд:

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

- Отредактировал файл shablon.tlc

1. Создал объект типа Simulator:

```
1 # создание объекта  Simulator
2 set ns [new Simulator]
```

Рис. 0.1: Создание объекта типа Simulator

2. Затем создадал переменную nf и указал, что требуется открыть на запись nam-файл для регистрации выходных результатов моделирования:

```
4 # открытие на запись файла out.nam для визуализатора namset
5 set nf [open out.nam w]
6
7 # все результаты моделирования будут записаны в переменную nf
8 $ns namtrace-all $nf
```

Рис. 0.2: Переменная nf с указанием открытия на запись nam-файла

3. Создал переменную f и открыл на запись файл трассировки для регистрации всех событий модели:

```
10 # открытие на запись файла трассировки out.tr
11 # для регистрации всех событий
12 set f [open out.tr w]
13
14 # все регистрируемые события будут записаны в переменную f
15 $ns trace-all $f
```

Рис. 0.3: Переменная f и открытие на записиь файла трассировки

4. Добавил процедуру finish, которая закрывает файлы трассировки и запускает nam:

```
17 # процедура finish закрывает файлы трассировки
18 # и запускает визуализатор nam
19 proc finish {} {
20         global ns f nf
21         $ns flush-trace
22         close $f
23         close $nf
24         # запуск nam в фоновом режиме
25         exec nam out.nam &
26         exit 0
27 }
```

Рис. 0.4: Добавление процедуры finish

5. С помощью команды at указал планировщику событий, что процедуру finish следует запустить через 5с после начала моделирования, после чего запустить симулятор ns:

```
28 # at-событие для планировщика событий, которое запускает
29 # процедуру finish через 5 с после начала моделирования
30 $ns at 5.0 "finish"
31
32 # запуск модели
33 $ns run
```

Рис. 0.5: Использование команды at для указания планировщику событий

- После сохранения изменений файл можно запустить при помощи команды: ns shablon.tcl

Получившийся шаблон можно использовать для дальнейшего использования, добавляя в него описание объектов и моделируемой системы до строки $ns at 5.0 “finish”

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

Постановка задачи. 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.

- При помощи команды cp shablon.tcl example1.tcl копировал содержимое шаблона в новый файл. Далее открыл его и внёс изменения, согласно топологии сети:

1. Описал топологию сети.

```
29 # создание 2-х узлов:
30 set N 2
31 for {set i 0} {$i < $N} {incr i} {
32         set n($i) [$ns node]
33 }
34
35 # соединение 2-х узлов дуплексным соединением
36 # с полосой пропускания 2 Мб/с и задержкой 10 мс,
37 # очередью с обслуживанием типа  DropTail
38 $ns duplex-link $n(0) $n(1) 2Mb 10ms DropTail
```

Рис. 0.6: Два узла, соединенные дуплексной линией связи

2. Создал агенты для генерации и приёма трафика.

```
# 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. 0.7: Агенты для генерации и приёма

Создаётся 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. Скорость источника: 800000 бит/с.

3. Далее создал Null-агент, который работает как приёмник трафика, и прикрепил его к узлу n1, а также соединил агенты между собой.

```
# 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
```

Рис. 0.8: Агент приёмник и присоединение агентов

4. Для запуска и остановки приложения CBR добавил at-события в планировщик событий (перед командой $ns at 5.0 “finish”).

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

# остановка приложения через 4,5 с
$ns at 4.5 "$cbr0 stop"
```

Рис. 0.9: Планировщик событий с at-событиями

- После сохранения изменений открыл файл и запустил симулятор командой: ns example1.tcl

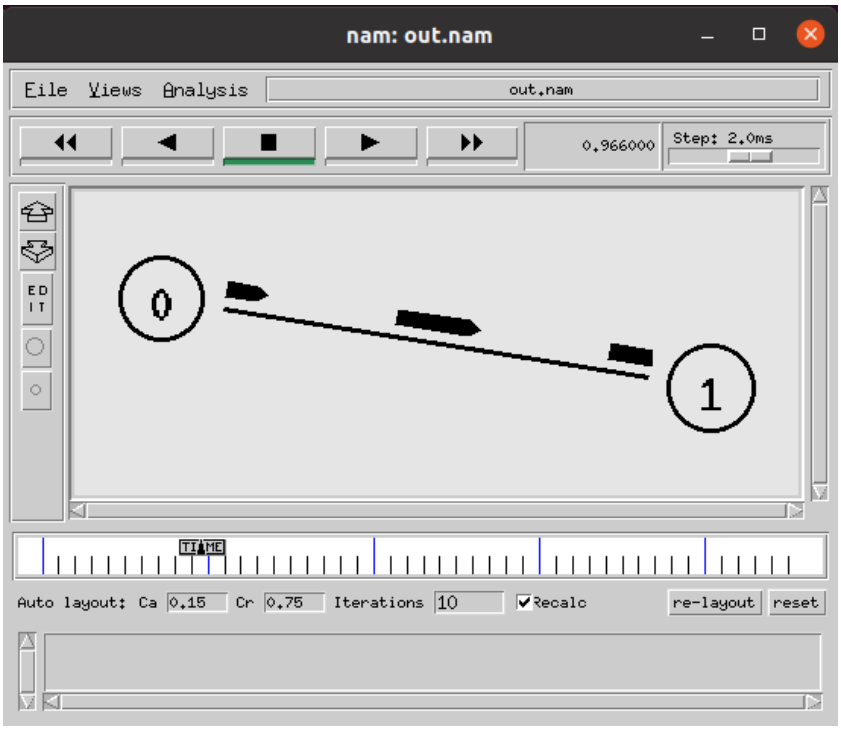

Рис. 0.10: Симуляция первого примера

Пример с усложнённой топологией сети.

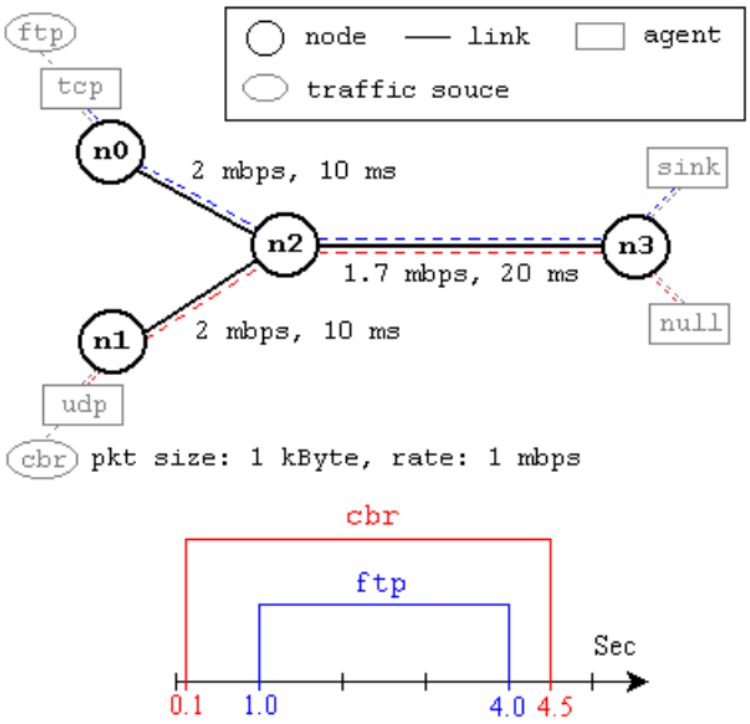

Рис. 0.11: Схема моделируемой сети для второго примера

Постановка задачи. 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;

– 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.

- При помощи команды cp shablon.tcl example2.tcl копировал содержимое шаблона в новый файл. Далее открыл его и внёс изменения, согласно топологии сети:

1. Создадал 4 узла и 3 дуплексных соединения с указанием направления.

```
29 set N 4
30 for {set i 0} {$i < $N} {incr i} {
31         set n($i) [$ns node]
32 }
33 $ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
34 $ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
35 $ns duplex-link $n(3) $n(2) 2Mb 10ms DropTail
36 $ns duplex-link-op $n(0) $n(2) orient right-down
37 $ns duplex-link-op $n(1) $n(2) orient right-up
38 $ns duplex-link-op $n(2) $n(3) orient right
```

Рис. 0.12: Узлы и дуплексные соединение для второго примера

2. Создадал агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP.

```
40 # создание агента UDP и присоединение его к узлу n(0)
41 set udp0 [new Agent/UDP]
42 $ns attach-agent $n(0) $udp0
43
44 # создание источника CBR-трафика# и присоединение его к агенту
   udp0
45 set cbr0 [new Application/Traffic/CBR]
46 $cbr0 set packetSize_ 500
47 $cbr0 set interval_ 0.005
48 $cbr0 attach-agent $udp0
49
50 # создание агента TCP и присоединение его к узлу n(1)
51 set tcp1 [new Agent/TCP]
52 $ns attach-agent $n(1) $tcp1
53
54 # создание приложения FTP
55 # и присоединение его к агенту tcp1
56 set ftp [new Application/FTP]
57 $ftp attach-agent $tcp1
```

Рис. 0.13: Агент UDP с источником CBR и агент TCP с приложением FTP

3. Создал агентов-получателей.

```
59 # создание агента-получателя для udp0
60 set null0 [new Agent/Null]
61 $ns attach-agent $n(3) $null0
62
63 # создание агента-получателя для tcp1
64 set sink1 [new Agent/TCPSink]
65 $ns attach-agent $n(3) $sink1
```

Рис. 0.14: Агенты получатели для второго примера

4. Соединил агенты udp0 и tcp1 и их получателей.

```
67 $ns connect $udp0 $null0
68 $ns connect $tcp1 $sink1
```

Рис. 0.15: Агенты udp1 и tcp1 с получателями

5. Задал описание цветов потока.

```
70 $ns color 1 Blue
71 $ns color 2 Red
72 $udp0 set class_ 1
73 $tcp1 set class_ 2
```

Рис. 0.16: Описание цвета каждлого потока

6. Отслеживание событий в очереди и наложение ограничений на ее размер.

```
75 $ns duplex-link-op $n(2) $n(3) queuePos 0.5
76
77 $ns queue-limit $n(2) $n(3) 20
```

Рис. 0.17: События и размерность очереди

7. Добавил at-события.

```
79 $ns at 0.5 "$cbr0 start"
80 $ns at 1.0 "$ftp start"
81 $ns at 4.0 "$ftp stop"
82 $ns at 4.5 "$cbr0 stop"
```

Рис. 0.18: at-события второго примера

- После сохранения изменений открыл файл и запустил симулятор командой: ns example2.tcl

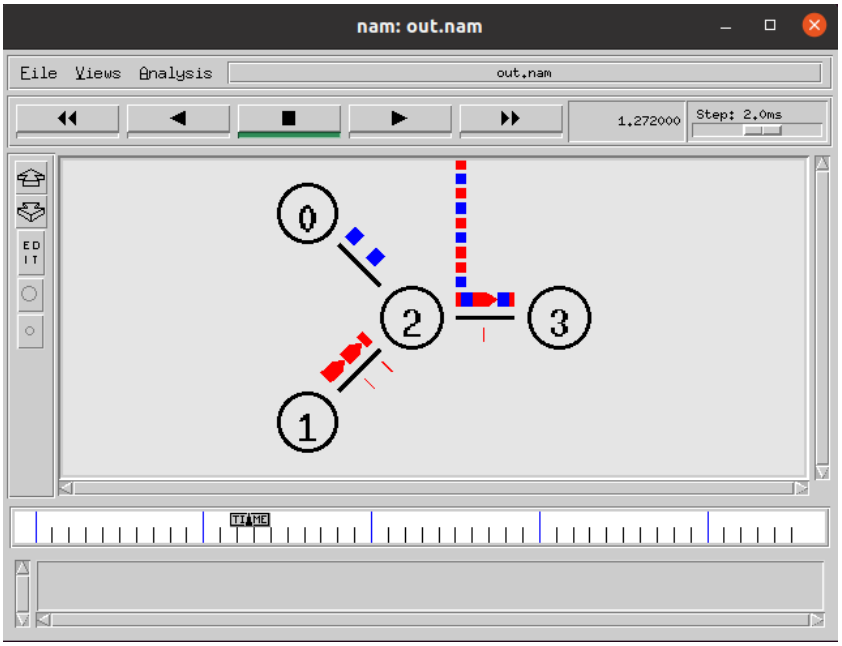

Рис. 0.19: Симуляция второго примера

Пример с кольцевой топологией сети.

Постановка задачи. 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

- При помощи команды cp shablon.tcl example3.tcl копировал содержимое шаблона в новый файл. Далее открыл его и внёс изменения, согласно топологии сети:

1. Описал топологию моделируемой сети и соединил узлы для создания круговой топологии. Каждый узел, за исключением последнего, соединяется со следующим, последний соединяется с первым. Для этого в цикле использован оператор %, означающий остаток от деления нацело.

```
31 set N 7
32 for {set i 0} {$i < $N} {incr i} {
33         set n($i) [$ns node]
34 }
35
36 for {set i 0} {$i < $N} {incr i} {
37         $ns duplex-link $n($i) $n([expr ($i+1)%$N]) 1Mb 10ms DropTail
38 }
```

Рис. 0.20: Создание круговой топологии

2. Задал передачу от узла n(0) к n(3).

```
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
```

Рис. 0.21: Передача данных от n(0) к n(3)

3. Добавил команду разрыва соединения между узлами n(1) и n(2) на время в одну секунду, а также время начала и окончания передачи данных.

```
$ns at 0.5 "$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. 0.22: Разрыв соединения межу узлами

4. Добавил в начало скрипта команду $ns rtproto DV. После запуска в сети отправляется небольшое количество маленьких пакетов, используемых для обмена информацией, необходимой для маршрутизации между узлами. Когда соединение будет разорвано, информацияо топологии будет обновлена, и пакеты будут отсылаться по новому маршруту через узлы n(6), n(5) и n(4).

- После сохранения изменений открыл файл и запустил симулятор командой: ns example3.tcl

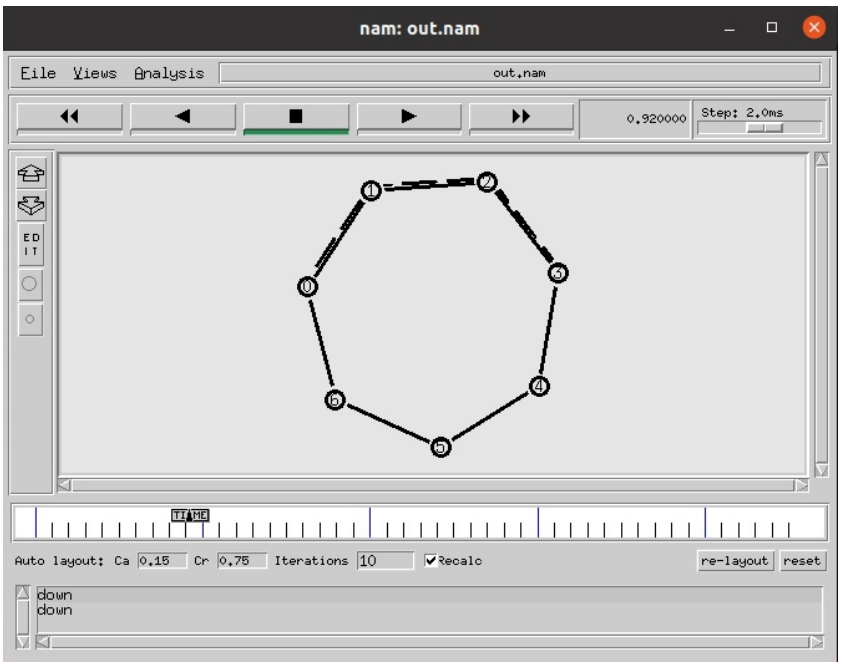

Рис. 0.23: Симуляция сети с круговой топологией

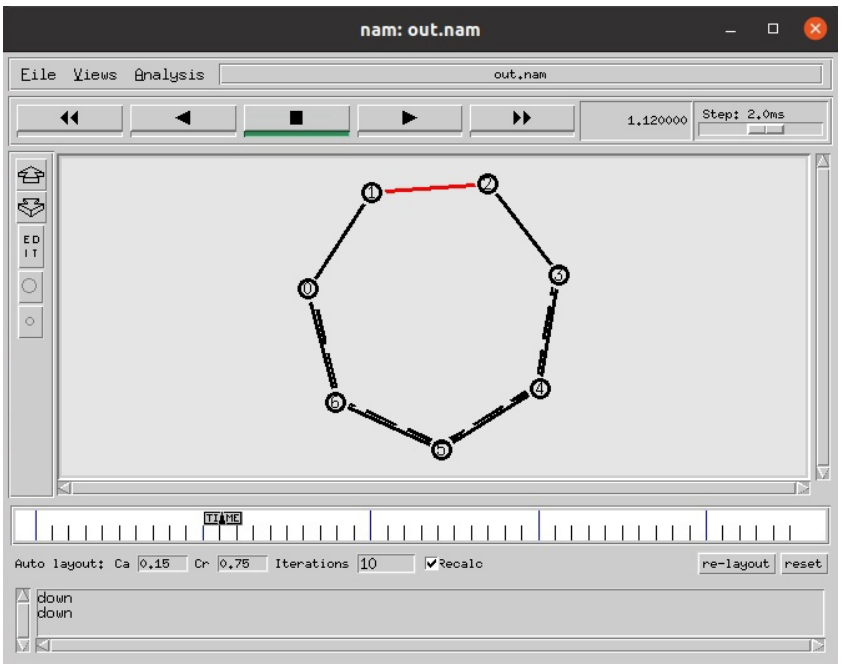

Рис. 0.24: Передача данных при разрыве соединения

Упражнение.

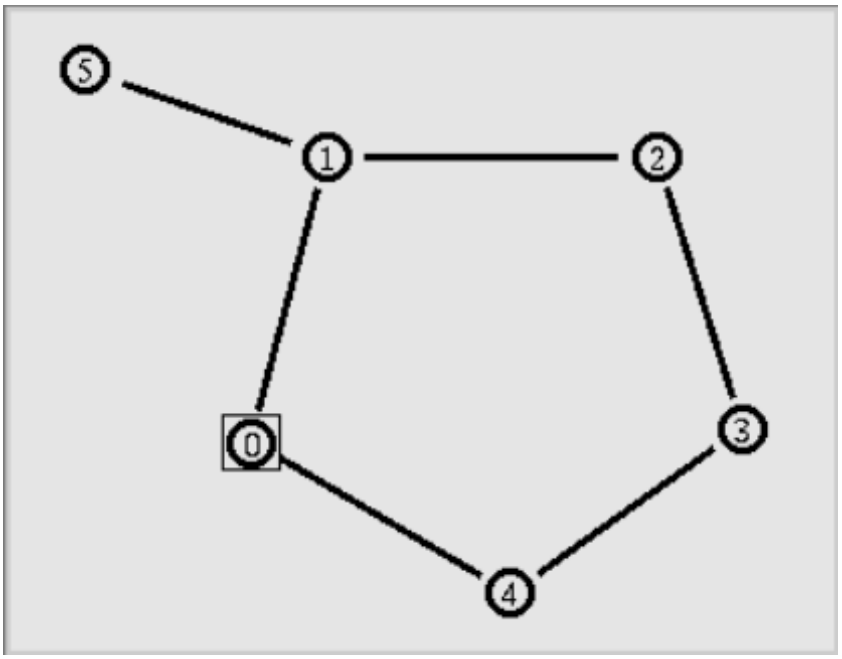

Рис. 0.25: Схема моделируемой сети в упражнении

Описание моделируемой 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.

- Внёс следующие изменения в скрипт третьего упражненения:

```
set N 6
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-1)]) 1Mb 10ms DropTail
}

$ns duplex-link $n(5) $n(1) 1Mb 10ms DropTail
$ns duplex-link-op $n(5) $n(1) orient left-top

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 tcp0 [new Agent/TCP/Newreno]
$ns attach-agent $n(0) $tcp0

set ftp [new Application/FTP]
$ftp attach-agent $tcp0

set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0

set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(5) $sink1

$ns connect $cbr0 $null0
$ns connect $tcp0 $sink1

$ns at 0.5 "$cbr0 start"
$ns at 0.5 "$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"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
```

Рис. 0.26: Изменения для соответвия описанию сети

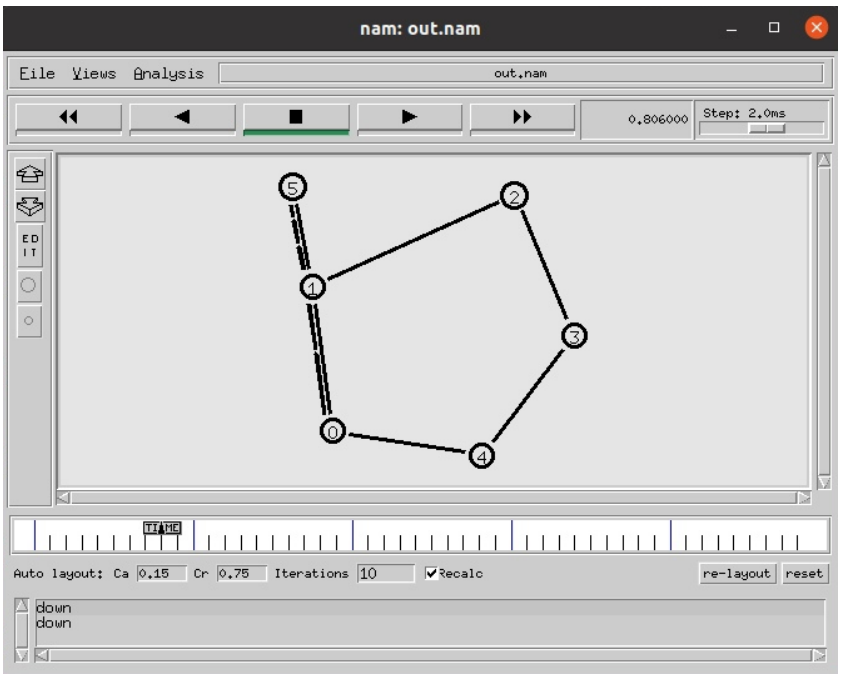

Рис. 0.27: Передача данных в сети

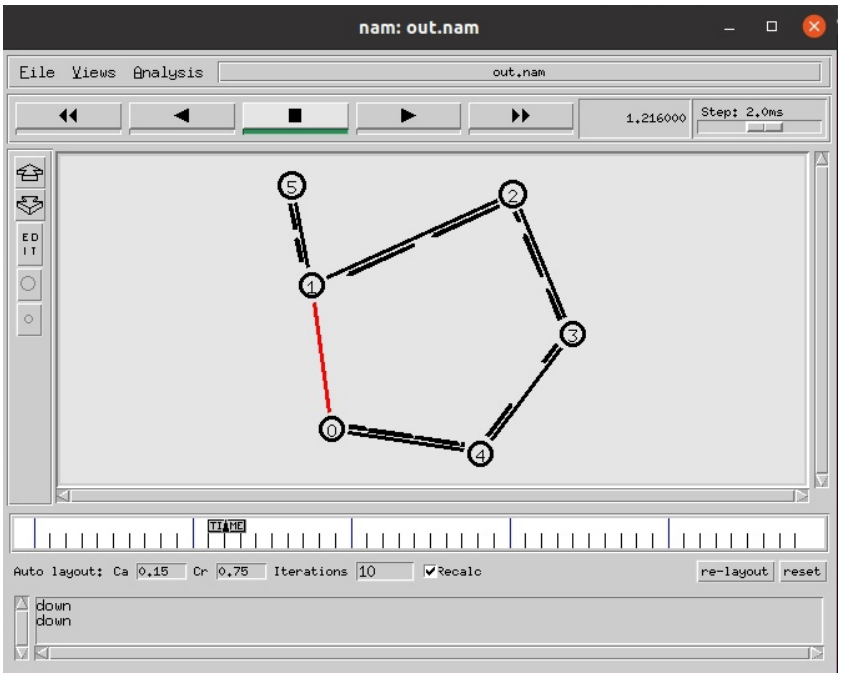

Рис. 0.28: Передача данных в сети при разрыве соединения

Выводы

- Приобрел навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также научился анализировать полученные результаты моделирования.