

Лабораторная работа №1

Имитационное моделирование

Волгин Иван Алексеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	17

Список иллюстраций

3.1	Создание директорий и файла шаблона	7
3.2	Код шаблона описания топологии сети	8
3.3	Запуск файла shablon.tcl	8
3.4	Код второго задания	9
3.5	Результат второго задания	10
3.6	Код третьего задания	11
3.7	Результат третьего задания	12
3.8	Код четвертого задания	13
3.9	Результат четвертого задания. Изначальный маршрут	14
3.10	Результат четвертого задания. Маршрут после разрыва соединения	14
3.11	Код упражнения	15
3.12	Результат упражнения	16

Список таблиц

1 Цель работы

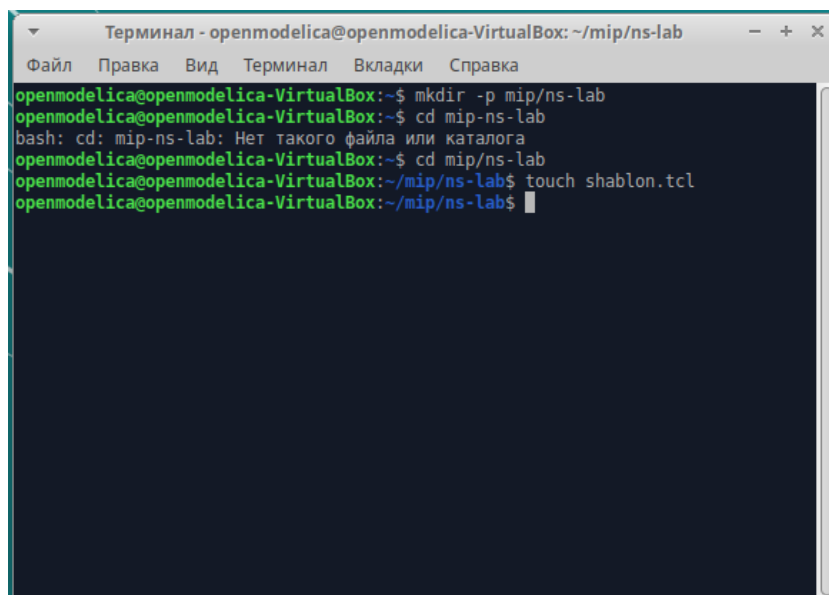
Приобретение навыков моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также анализ полученных результатов моделирования.

2 Задание

1. Создать шаблом сценария для NS-2
2. Создать простой пример описания топологии сети, состоящей из двух узлов и одного соединения.
3. Создать пример усложненной топологической сети.
4. Создать пример кольцевой топологической сети
5. Самостоятельно изменить кольцевую топологическую сеть выполнив дополнительное упражнение.

3 Выполнение лабораторной работы

1. Для начала в своем рабочем каталоге я создал директорию `mip`, в ней создал еще одну директорию `lab-ns`, а в ней файл шаблона описания топологии сети (рис. 3.1).



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/ns-lab
Файл  Правка  Вид  Терминал  Вкладки  Справка
openmodelica@openmodelica-VirtualBox:~$ mkdir -p mip/ns-lab
openmodelica@openmodelica-VirtualBox:~$ cd mip/ns-lab
bash: cd: mip/ns-lab: Нет такого файла или каталога
openmodelica@openmodelica-VirtualBox:~$ cd mip/ns-lab
openmodelica@openmodelica-VirtualBox:~/mip/ns-lab$ touch shablon.tcl
openmodelica@openmodelica-VirtualBox:~/mip/ns-lab$
```

Рис. 3.1: Создание директорий и файла шаблона

2. Далее я написал код шаблона для описания топологии сети (рис. 3.2)

```
*/home/openmodelica/mip/ns-lab/shablon.tcl - Mousepad
Файл  Правка  Поиск  Вид  Документ  Справка
set ns [new Simulator]

set nf [open out.nam w]

$ns namtrace-all $nf

set f [open out.tr w]

$ns trace-all $f

proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit
}

$ns at 5.0 "finish"

$ns run|
```

Рис. 3.2: Код шаблона описания топологии сети

- После этого я запустил файл shablon.tcl и посмотрел корректность его компиляции (рис. 3.3).

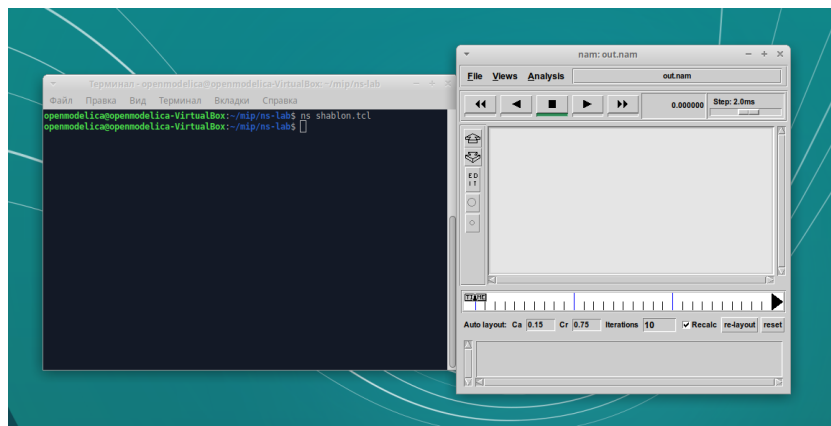


Рис. 3.3: Запуск файла shablon.tcl

- Затем я приступил к выполнению второй задачи, где нужно было создать простой пример топологии сети. 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, очере-

дью 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. Чтобы выполнить задание, я скопировал файл шаблона в новый файл example1.tcl и дописал до строки \$ns at 5.0 “finish” код реализации задания (рис. 3.4). Затем я скомпилировал файл и посмотрел результат (рис. 3.5)

```
set N 2
for {set i 0} {$i < $N} {incr i} {
    set n($i) [$ns node]
}

$ns duplex-link $n(0) $n(1) 2Mb 10ms DropTail

set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(1) $null0

$ns connect $udp0 $null0

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

Рис. 3.4: Код второго задания

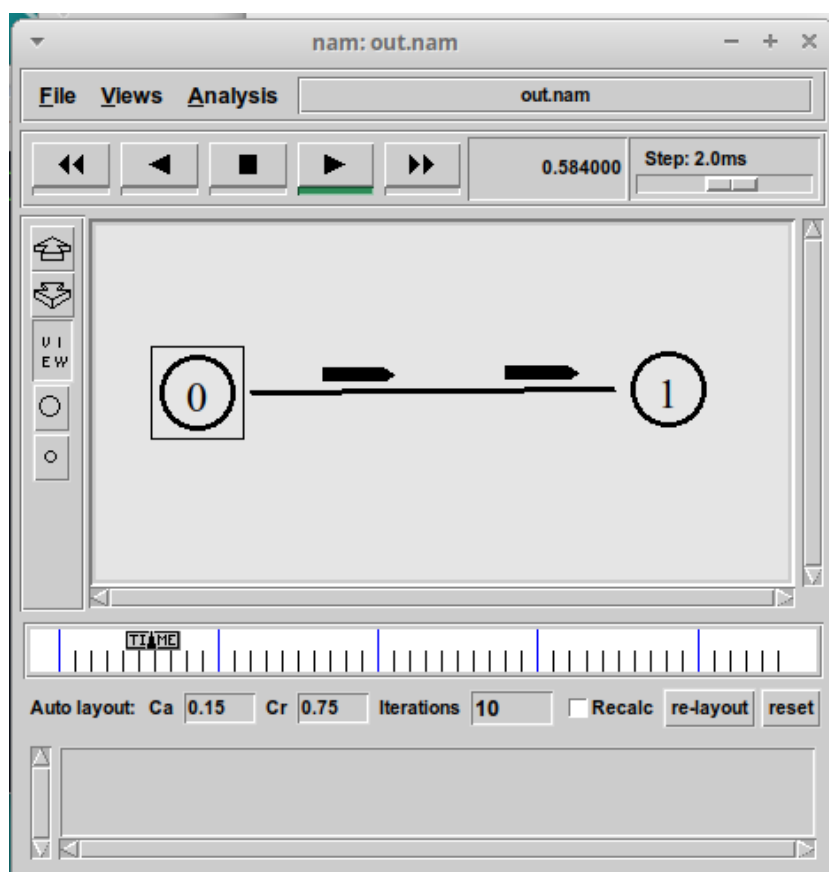


Рис. 3.5: Результат второго задания

5. Далее я приступил к выполнению третьего задания, где нужно было усложнить пример топологии сети. Точная постановка задачи звучит так. – 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

n1, подключён к null-агенту на узле n3; – 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.

Я снова скопировал файл шаблона в новый файл example2.tcl и написал реализацию задания (рис. 3.6). Затем я скомпилировал код и посмотрел корректность выполнения (рис. 3.7)

```
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

set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize 500
$cbr0 set interval 0.005
$cbr0 attach-agent $udp0

set tcp1 [new Agent/TCP]
$ns attach-agent $n(1) $tcp1
set ftp [new Application/FTP]
$ftp attach-agent $tcp1

set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0

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

$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
```

Рис. 3.6: Код третьего задания

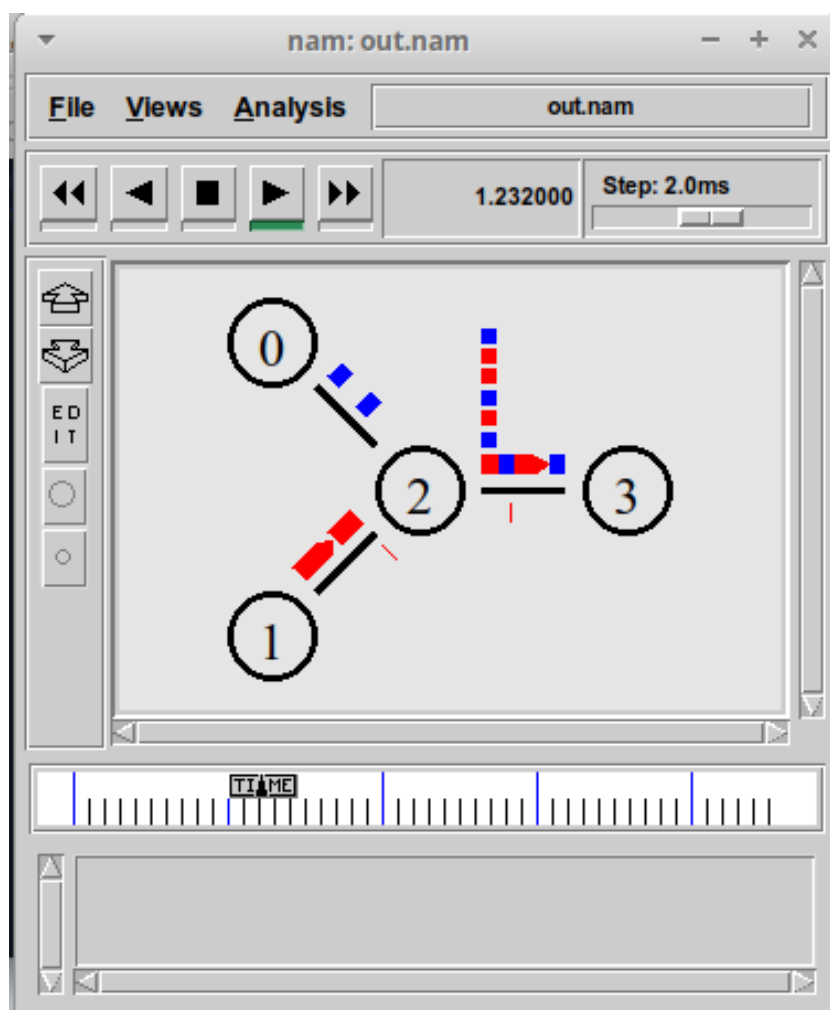


Рис. 3.7: Результат третьего задания

6. Следующим заданием было написание примера кольцевой топологии сети. Постановка задачи: – 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.

Я создал `example3.tcl` на основе файла шаблона и далее написал реализацию задания (рис. 3.8). Затем я скомпилировал файл и посмотрел результаты. Все получилось, сначала сеть работала по одному маршруту (рис. 3.9) и после разрыва одного соединения трафик пошел по другому пути (рис. 3.10)

```

set ns [new Simulator]

$ns rtproto DV

set nf [open out.nam w]

$ns namtrace-all $nf

set f [open out.tr w]

$ns trace-all $f

proc finish {} {
    global ns f nf
    $ns flush-trace
    close $f
    close $nf
    exec nam out.nam &
    exit
}

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"

$ns run

```

Рис. 3.8: Код четвертого задания

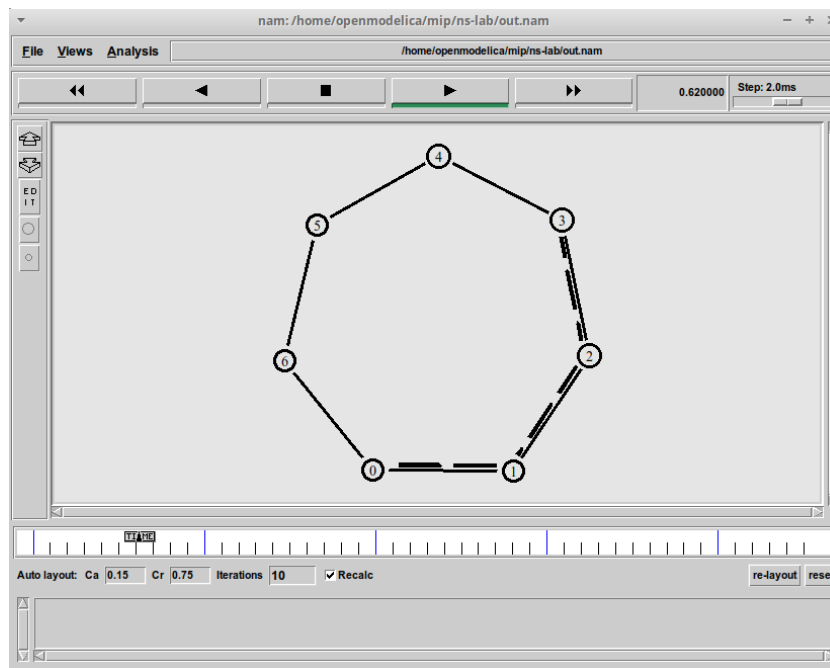


Рис. 3.9: Результат четвертого задания. Изначальный маршрут

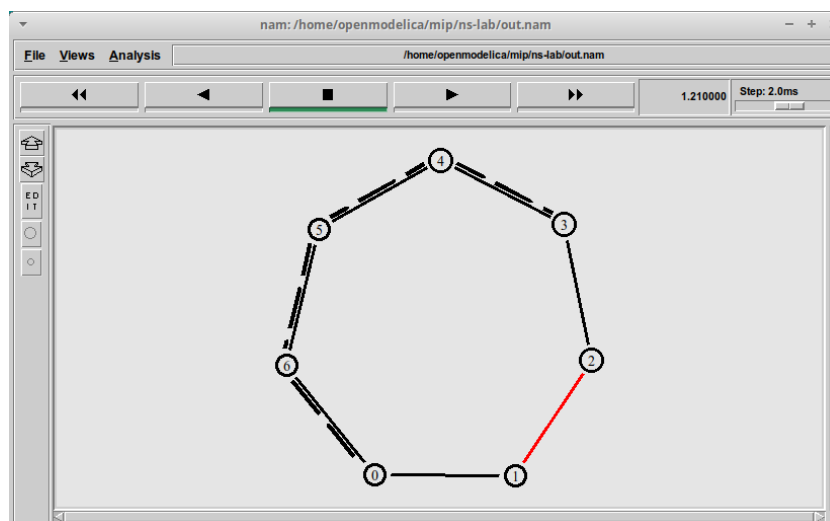


Рис. 3.10: Результат четвертого задания. Маршрут после разрыва соединения

7. Затем я приступил к выполнению упражнения, которое заключалось в том, чтобы переписать код четвертого задания для изменения примера топологии сети. Постановка задачи. – 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. Я написал код для реализации упражнения (рис. 3.11), затем я его скомпилировал и получил удовлетворительный результат (рис. 3.12).

```

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"

$ns run

```

Рис. 3.11: Код упражнения

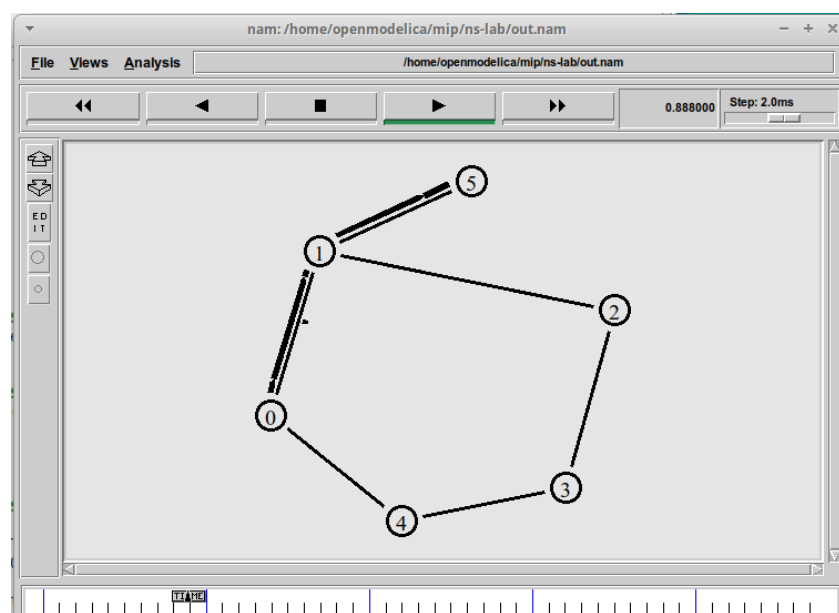


Рис. 3.12: Результат упражнения

4 Выводы

В ходе выполнения лабораторной работы я научился создавать простые примеры топологий сети. Я написал шаблон для реализации следующих заданий. Затем с помощью него создал простейшую топологию, более сложную и кольцевую, а так же самостоятельно выполнил упражнение.