

Отчёт по лабораторной работе №1

Простые модели компьютерной сети

Ибатулина Дарья Эдуардовна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	9
5	Выводы	27
	Список литературы	28

Список иллюстраций

4.1	Создание директории и файла для задания 1	9
4.2	Код для “пустого” процесса	10
4.3	Запуск симуляции процесса	10
4.4	Просмотр графического отображения пустого процесса	11
4.5	Создание файла для образца 1	11
4.6	Код для симуляции сети из двух узлов (1)	12
4.7	Код для симуляции сети из двух узлов (2)	13
4.8	Просмотр процесса	14
4.9	Просмотр сведений о передаваемом пакете	14
4.10	Создание файла для образца 2	14
4.11	Код для усложненной топологии сети	16
4.12	Просмотр процесса (1)	17
4.13	Просмотр процесса (2)	17
4.14	Просмотр процесса (3)	18
4.15	Создание файла для образца 2	18
4.16	Просмотр процесса	19
4.17	Просмотр процесса	20
4.18	Просмотр процесса (1)	21
4.19	Просмотр процесса (2)	21
4.20	Просмотр процесса (3)	22
4.21	Код для самостоятельной работы	23
4.22	Иллюстрация процесса из задания в самостоятельной работе . . .	24
4.23	Просмотр процесса (1)	25
4.24	Просмотр процесса (2)	25
4.25	Просмотр процесса (3)	26

1 Цель работы

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

2 Задание

Смоделировать несколько сетей передачи данных различной сложности с помощью средства имитационного моделирования NS-2.

3 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.

NS-2 является объектно-ориентированным программным обеспечением. Его ядро реализовано на языке C++. В качестве интерпретатора используется язык скриптов (сценариев) OTcl (Object oriented Tool Command Language). NS-2 полностью поддерживает иерархию классов C++ и подобную иерархию классов интерпретатора OTcl.

Обе иерархии обладают идентичной структурой, т.е. существует однозначное соответствие между классом одной иерархии и таким же классом другой. Объединение для совместного функционирования C++ и OTcl производится при помощи TclCl (Classes Tcl). В случае, если необходимо реализовать какую-либо специфическую функцию, не реализованную в NS-2 на уровне ядра, для этого используется код на C++.

Процесс создания модели сети для NS-2 состоит из нескольких этапов: 1. Создание нового объекта класса Simulator, в котором содержатся методы, необходимые для дальнейшего описания модели (например, методы new и delete используются для создания и уничтожения объектов соответственно); 2. Опи-

сание топологии моделируемой сети с помощью трёх основных функциональных блоков: узлов (nodes), соединений (links) и агентов (agents); 3. Задание различных действий, характеризующих работу сети.

Для создания узла используется метод `node`. При этом каждому узлу автоматически присваивается уникальный адрес. Для построения однонаправленных и двунаправленных линий соединения узлов используют методы `simplex-link` и `duplex-link` соответственно.

Важным объектом NS-2 являются агенты, которые могут рассматриваться как процессы и/или как транспортные единицы, работающие на узлах моделируемой сети.

Агенты могут выступать в качестве источников трафика или приёмников, а также как динамические маршрутизирующие и протокольные модули. Агенты создаются с помощью методов общего класса `Agent` и являются объектами его подкласса, т.е. `Agent/type`, где `type` определяет тип конкретного объекта. Например, TCP-агент может быть создан с помощью команды: `set tcp [new Agent/TCP]`

Для закрепления агента за конкретным узлом используется метод `attach-agent`. Каждому агенту присваивается уникальный адрес порта для заданного узла (аналогично портам `tcp` и `udp`). Чтобы за конкретным агентом закрепить источник, используют методы `attach-source` и `attach-traffic`. Например, можно прикрепить `ftp` или `telnet` источники к TCP-агенту. Есть агенты, которые генерируют свои собственные данные, например, CBR-агент (Constant Bit-Rate) — источник трафика с постоянной интенсивностью.

Действия разных агентов могут быть назначены планировщиком событий (Event Scheduler) в определённые моменты времени (также в определённые моменты времени могут быть задействованы или отключены те или иные источники данных, запись и т.д.). Для этого может использоваться метод `at`. Моделирование начинается при помощи метода `run`.

В качестве дополнения к NS-2 часто используют средство визуализации

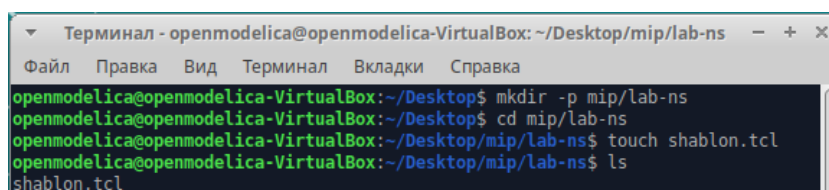
nam (network animator) для графического отображения свойств моделируемой системы и проходящего через неё трафика и пакет **Xgraph** для графического представления результатов моделирования.

Запуск сценария NS-2 осуществляется в командной строке с помощью команды: **ns [tclscript]**

Здесь **[tclscript]** — имя файла скрипта Tcl, который определяет сценарий моделирования (т.е. топологию и различные события). Nam можно запустить с помощью команды **nam [nam-file]** Здесь **[nam-file]** — имя nam trace-файла, сгенерированного с помощью ns.

4 Выполнение лабораторной работы

Создадим директорию для лабораторных работ по ns-2, перейдем в неё и создадим файл, в котором затем напишем код для первого задания (рис. 4.1).



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/Desktop/mip/lab-ns - + x
Файл  Правка  Вид  Терминал  Вкладки  Справка
openmodelica@openmodelica-VirtualBox:~/Desktop$ mkdir -p mip/lab-ns
openmodelica@openmodelica-VirtualBox:~/Desktop$ cd mip/lab-ns
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$ touch shablon.tcl
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$ ls
shablon.tcl
```

Рис. 4.1: Создание директории и файла для задания 1

Напишем код, описывающий “пустой” процесс. Пустым я назвала его потому, что trace file - файл трассировки пустой, поскольку в процессе не задано ни объектов, ни действий. Впоследствии я буду использовать его для моделирования более сложных процессов (рис. 4.2).

```

# 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 # объявление глобальных переменных
    $ns flush-trace
    # запуск nam в фоновом режиме
    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. 4.2: Код для “пустого” процесса

С помощью команды `ns имя_файла` я запустила симуляцию процесса (рис. 4.3).

```

openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$ ns shablon.tcl

```

Рис. 4.3: Запуск симуляции процесса

Открылось графическое окно программы ns-2 с данным процессом. Белое окошко пустое, поскольку в процессе не заданы объекты и действия (рис. 4.4).

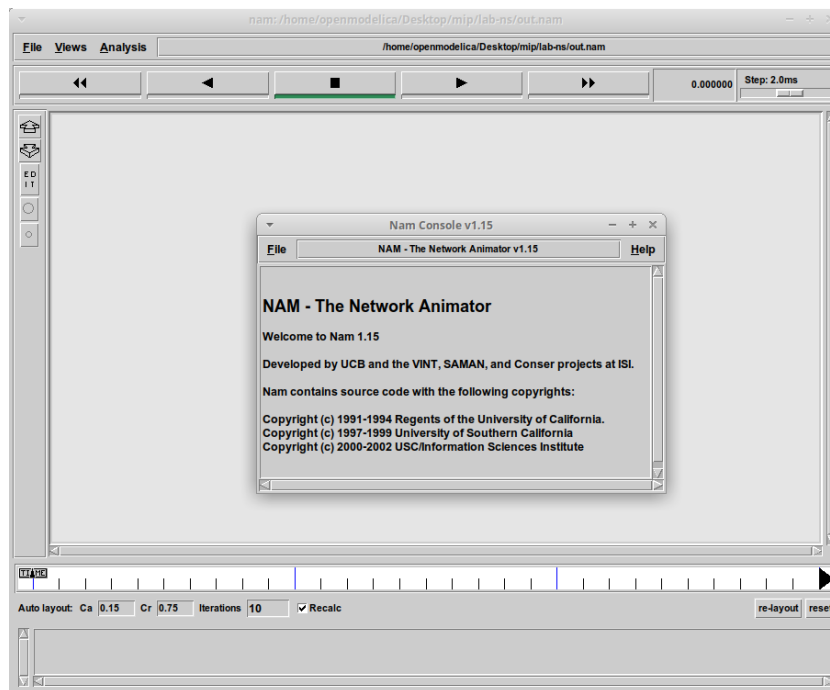


Рис. 4.4: Просмотр графического отображения пустого процесса

Создаем файл *example1.tcl*, чтобы смоделировать простой процесс (рис. 4.5).

```
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$ cp shablon.tcl exampl
e1.tcl
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$ ls
example1.tcl out.nam out.tr shablon.tcl
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$
```

Рис. 4.5: Создание файла для образца 1

Пишем по приведенному в лабораторной работе указанию код для симуляции сети, состоящей из двух узлов. Требование к заданию: 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. (рис. 4.6, 4.7).

```

/home/openmodelica/Desktop/mip/lab-ns/example1.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 # объявление глобальных переменных
    $ns flush-trace
    # запуск nam в фоновом режиме
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}

# 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

```

Рис. 4.6: Код для симуляции сети из двух узлов (1)

```

# 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"

# запуск модели
$ns run

```

Рис. 4.7: Код для симуляции сети из двух узлов (2)

Запускаем процесс и, нажав кнопку *play*, наблюдаем его симуляцию: в окне *nam* через 0.5 секунды из узла 0 данные начнут поступать к узлу 1 (рис. 4.8). Это процесс можно замедлить, выбирая шаг отображения в *nam*. Можно осуществлять наблюдение за отдельным пакетом, щёлкнув по нему в окне *nam*, а щёлкнув по соединению, можно получить о нем некоторую информацию (рис. 4.9).

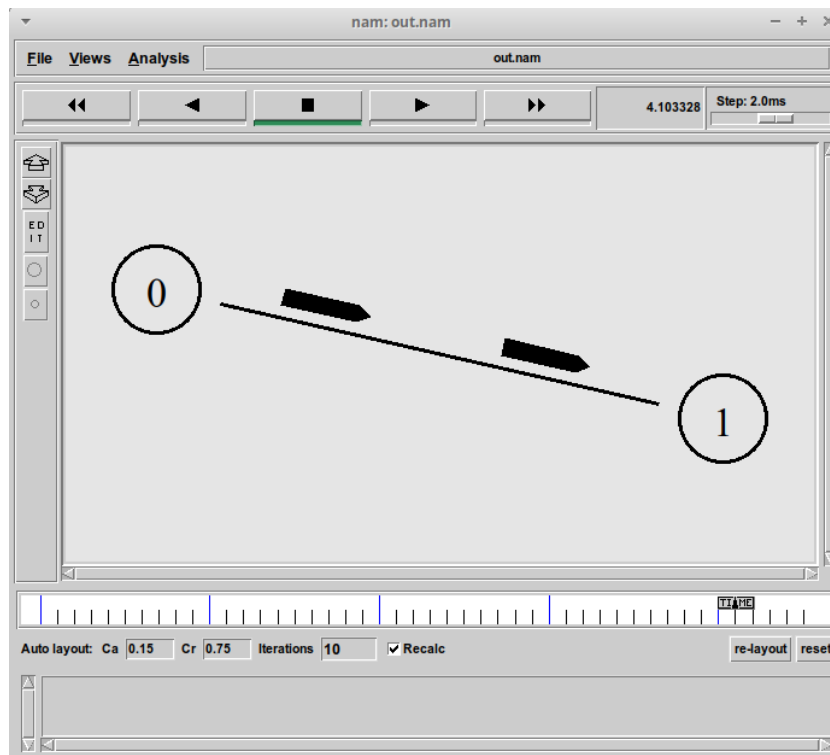


Рис. 4.8: Просмотр процесса

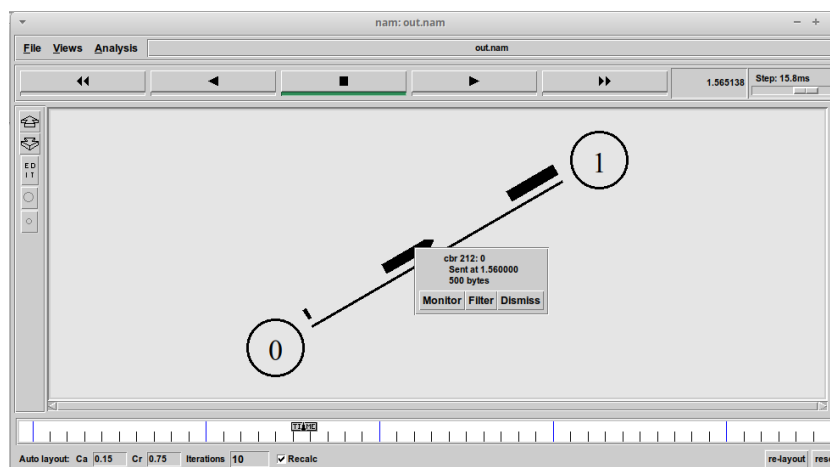


Рис. 4.9: Просмотр сведений о передаваемом пакете

Создаём новый файл для симуляции более сложной топологии сети (рис. 4.10).

```
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$ cp shablon.tcl example2.tcl
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$
```

Рис. 4.10: Создание файла для образца 2

Напишем код для симуляции процесса, отвечающего требованиям:

– 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 (рис. 4.11).

```

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(2) $n(3) 1.7Mb 20ms 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
$ns at 0.1 "$cbr0 start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr0 stop"
# at-событие для планировщика событий, которое запускает
# процедуру finish через 5 с после начала моделирования
$ns at 5.0 "finish"
# запуск модели
$ns run

```

Рис. 4.11: Код для усложненной топологии сети

Запускаем процесс и, нажав кнопку *play*, наблюдаем его симуляцию (рис. 4.12, 4.13, 4.14).

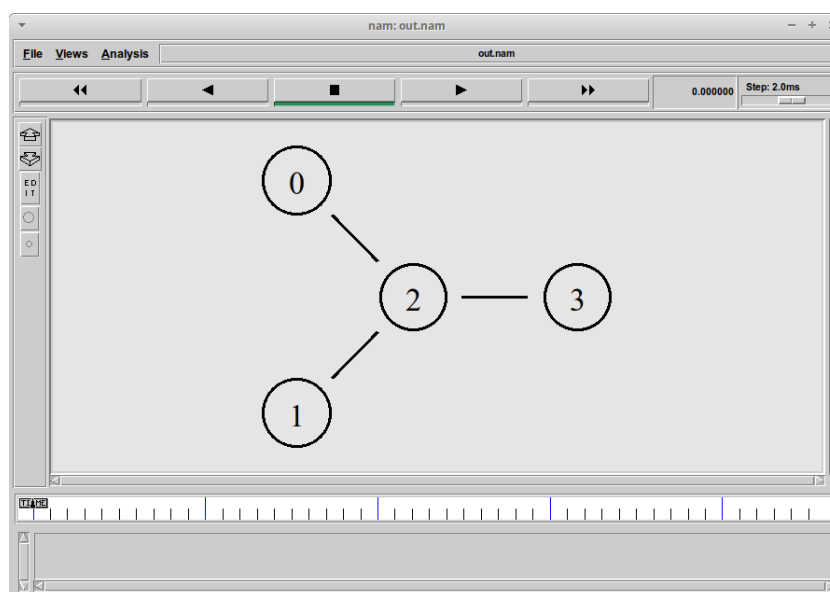


Рис. 4.12: Просмотр процесса (1)

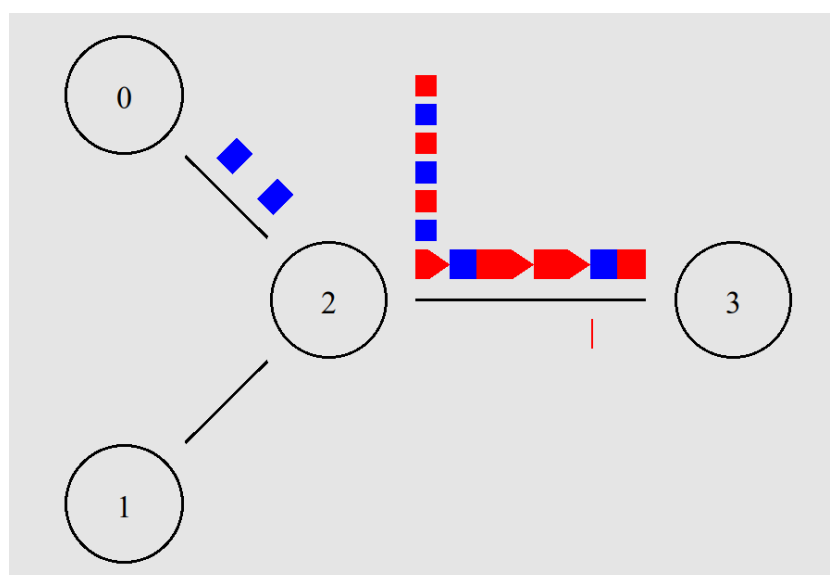


Рис. 4.13: Просмотр процесса (2)

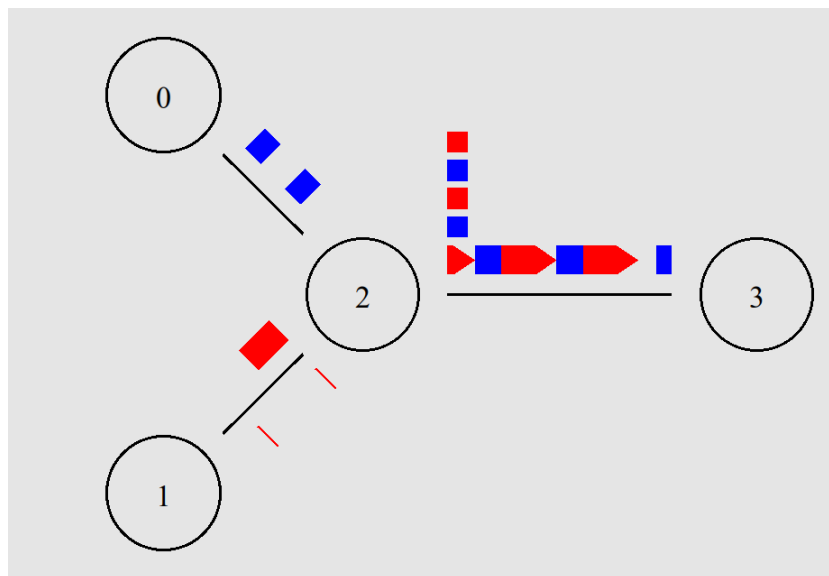


Рис. 4.14: Просмотр процесса (3)

При запуске скрипта можно заметить, что по соединениям между узлами $n(0) \rightarrow n(2)$ и $n(1) \rightarrow n(2)$ 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 Mb, а суммарная — 1.6Mb. Но соединение $n(2) \rightarrow n(3)$ имеет полосу лишь 1 Mb. Следовательно, часть пакетов должна теряться. В окне аниматора можно видеть пакеты в очереди, а также те пакеты, которые отбрасываются при переполнении.

Создадим файл для новой топологии - кольцо (рис. 4.15).

```
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$ cp shablon.tcl example3.tcl
openmodelica@openmodelica-VirtualBox:~/Desktop/mip/lab-ns$
```

Рис. 4.15: Создание файла для образца 2

Затем смоделируем более сложную топологию сети - кольцо из семи узлов. Требования: – 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 $n(2)$;

Напишем код (рис. 4.16).

```
# создание объекта Simulator
set ns [new Simulator]
$ns rtproto DV
# 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 # объявление глобальных переменных
    $ns flush-trace
    # запуск nam в фоновом режиме
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}
set N 7
for {set i 0} {$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"
# 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. 4.16: Просмотр процесса

Запускаем процесс и, нажав кнопку *play*, наблюдаем его симуляцию (рис. 4.17).

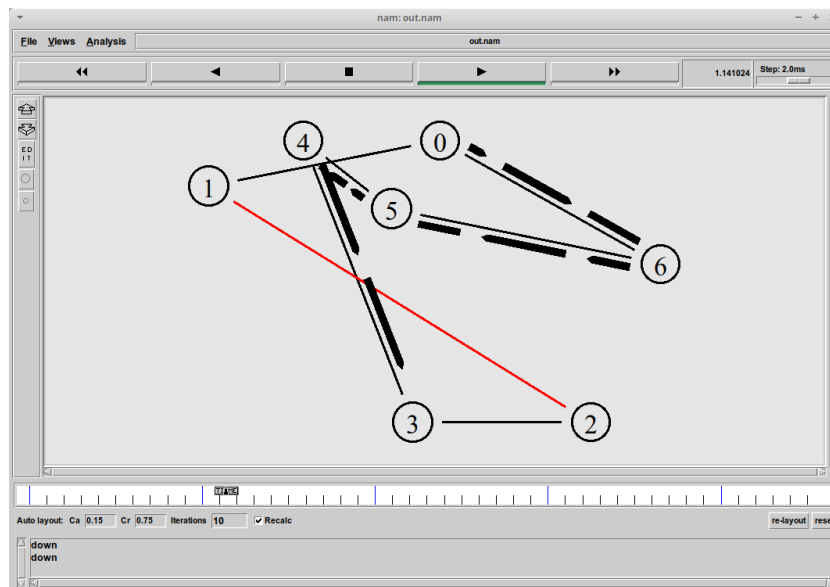


Рис. 4.17: Просмотр процесса

Нажав на кнопку *edit* слева, можно сделать процесс более понятным (красиво перерисовать, чтобы смотрелось действительно как кольцо) (рис. 4.18). Можно заметить, что на секундах с 0й по 1ую данные передаются из 0 в 3 по кратчайшему пути (через 1 и 2), а затем, когда происходит разрыв соединения между 1 и 2 на 2й секунде, данные идут от 0 в 3 по длинному пути (6, 5 и 4), а затем снова идут по кратчайшему пути, когда соединение между узлами 1 и 2 восстанавливается (рис. 4.19, 4.20).

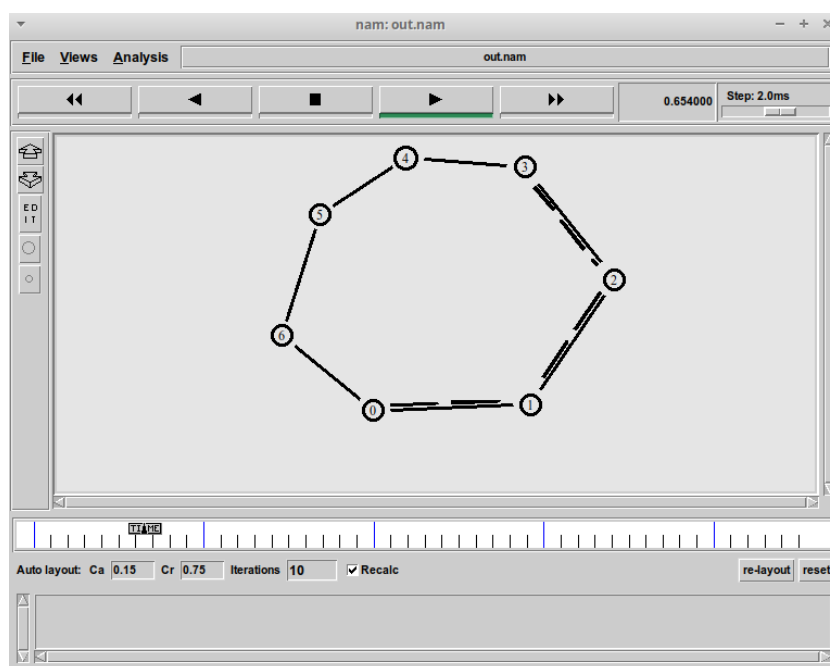


Рис. 4.18: Просмотр процесса (1)

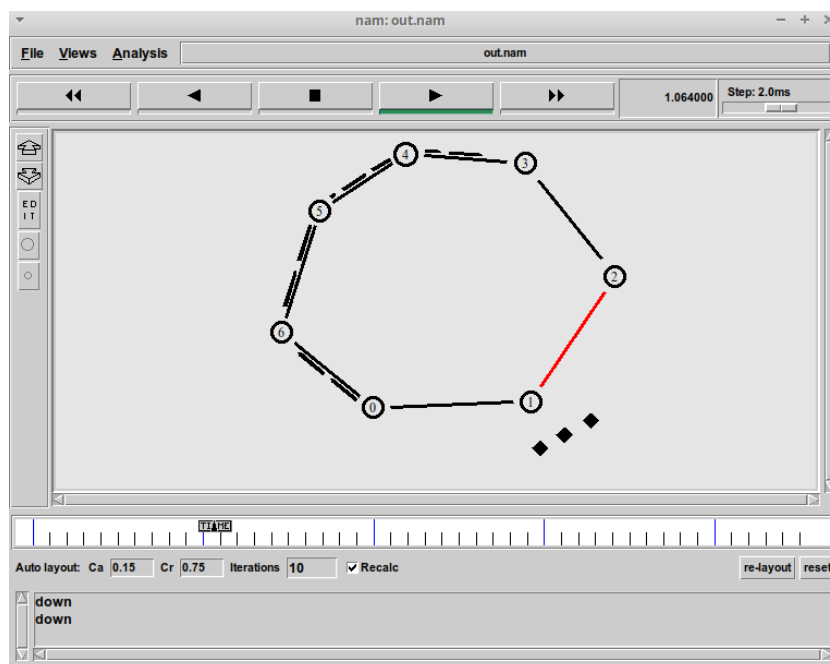


Рис. 4.19: Просмотр процесса (2)

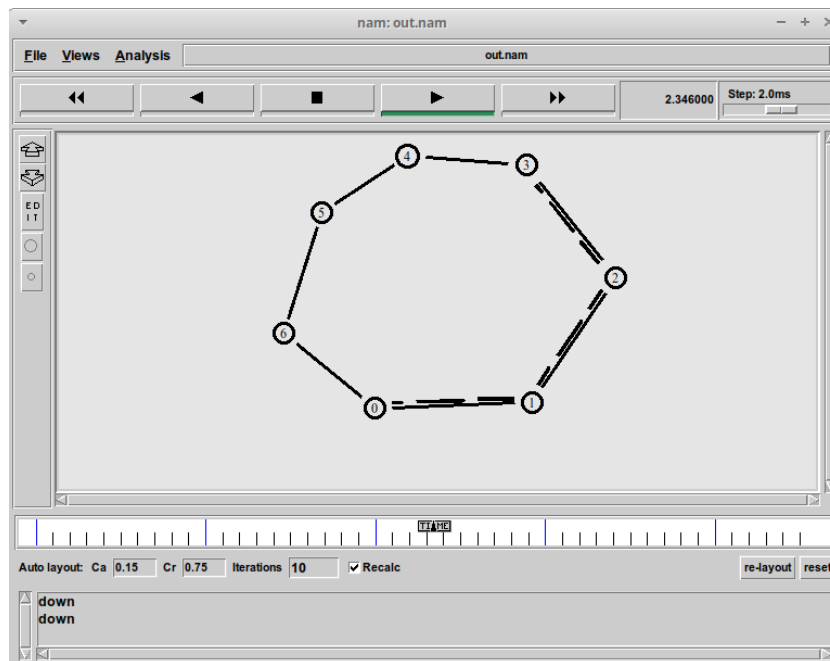


Рис. 4.20: Просмотр процесса (3)

Пришло время для самостоятельной работы. Теперь требуется построить сеть из 6 узлов, 5 из которых образуют кольцо, а из узла 1 выходит узел 5, при этом:

- 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.

Пишем код (рис. 4.21).

```

# создание объекта Simulator
set ns [new Simulator]
$ns rtproto DV
# 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 # объявление глобальных переменных
    $ns flush-trace
    # запуск nam в фоновом режиме
    close $f
    close $nf
    exec nam out.nam &
    exit 0
}
# кол-во узлов в кольцевой части
set N 5
for {set i 0} {$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)%$N]) 1Mb 10ms DropTail
}
# создала отдельно узел под номером 5 (шестой узел), так как он не в кольце
set n5 [$ns node]
# установила соединение между 1-м и 5-м узлом
$ns duplex-link $n5 $n(1) 1Mb 10ms DropTail
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n(0) $tcp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
set tcp_sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n5 $tcp_sink
$ns connect $tcp $tcp_sink
# разрыв соединения между 0 и 1 на 2й секунде (с 1й по 2ю секунду), установка начала и окончания передачи данных
$ns at 0.5 "$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"
# запуск модели
$ns run

```

Рис. 4.21: Код для самостоятельной работы

Я решила включить 5 узлов с 0го по 4й в кольцо, а затем привязать к узлу 1 узел 5 отдельно. Узлы с 0го по 4й связала между собой дуплексными соединениями по порядку, чтобы получилось кольцо. Между первым и пятым узлом установила дуплексное соединение отдельно. Затем подключила Агентов tcp и tcpSink и приложение ftp, установила разрыв между 0 и 1 на 2й секунде (с 1й по 2ую секунду). ftp запускается через полсекнды, и заканчивает работу в 4,5 секунды. Всего передача данных от узла 0 к узлу 5 длится 5 секунд.

Запускаем файл и видим наш процесс (рис. 4.22).

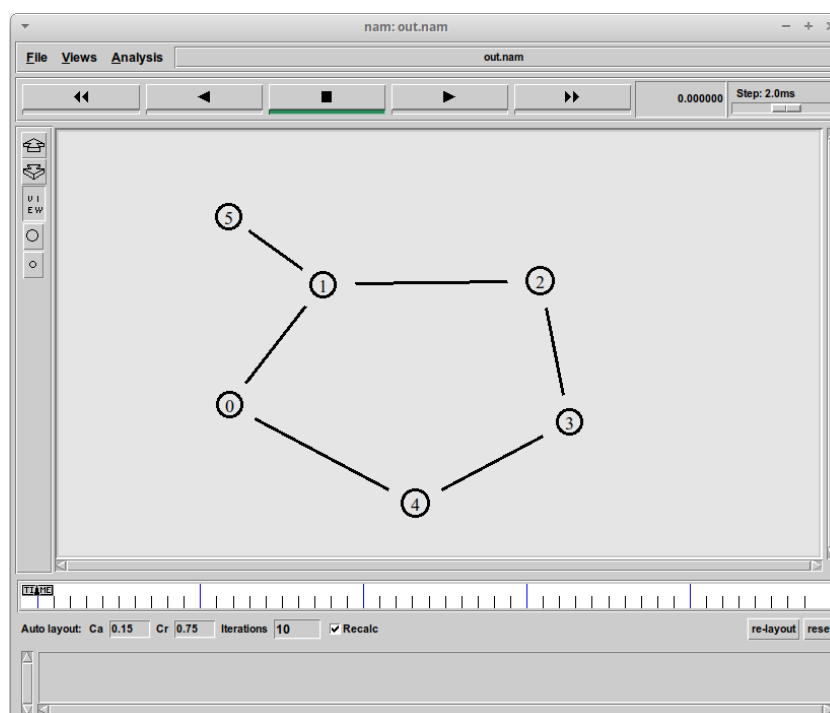


Рис. 4.22: Иллюстрация процесса из задания в самостоятельной работе

Нажав на кнопку *play*, мы увидим, как проходит процесс. Можно заметить, что на секундах с 0й по 1ую данные передаются из 0 в 5 по кратчайшему пути (через 1), а затем, когда происходит разрыв соединения между 0 и 1 на 2й секунде (во время с 1й секунды по 2ую), данные идут от 0 в 5 по длинному пути (4, 3 и 2), а затем снова идут по кратчайшему пути, когда соединение между узлами 0 и 1 восстанавливается (рис. 4.23, 4.24, 4.25).

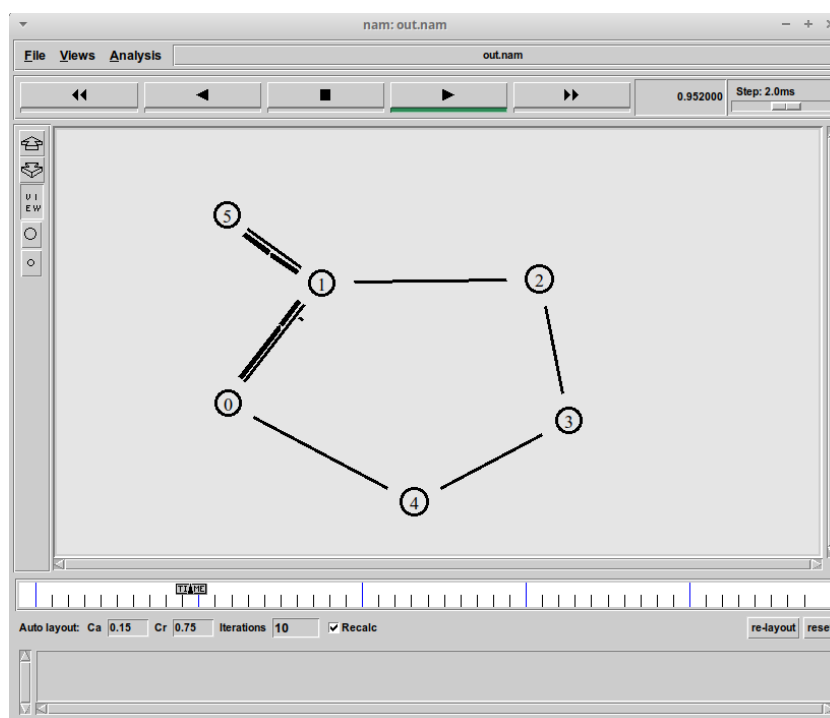


Рис. 4.23: Просмотр процесса (1)

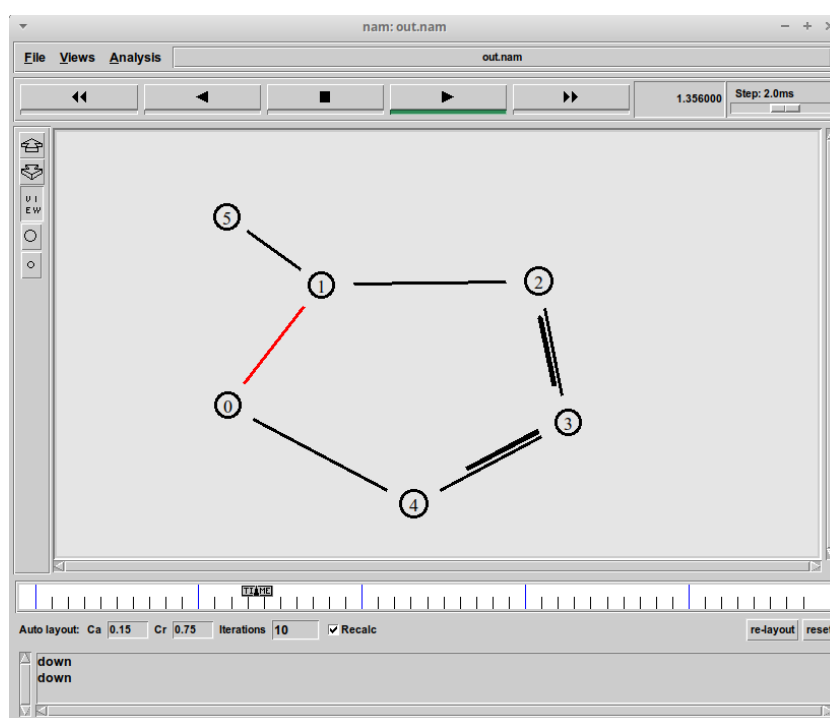


Рис. 4.24: Просмотр процесса (2)

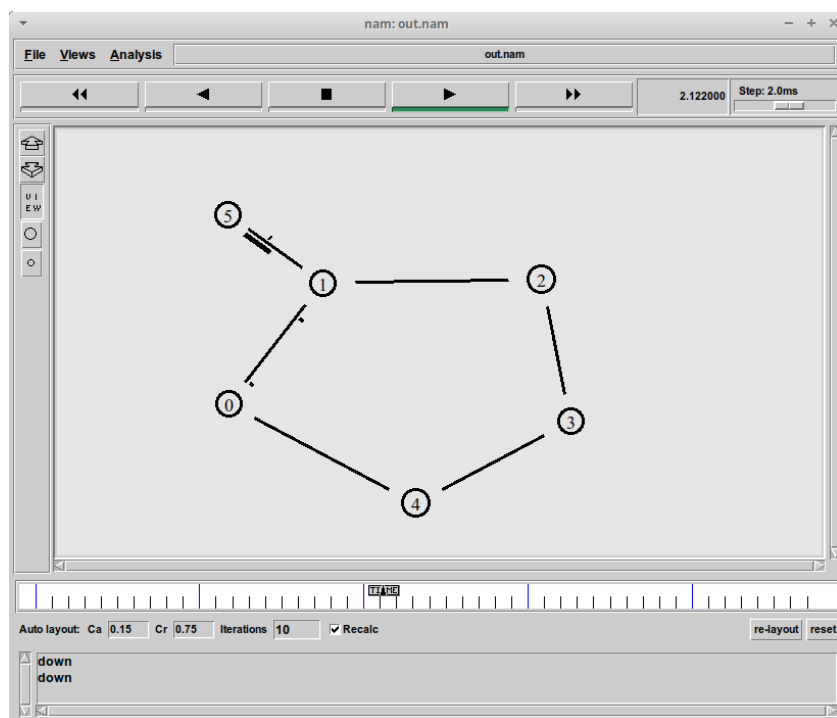


Рис. 4.25: Просмотр процесса (3)

5 Выводы

В результате выполнения лабораторной работы я научилась работать со средством ns-2: моделировать пустой процесс и процессы с простыми и сложными топологиями, а также с динамической маршрутизацией и условиями, накладывающимися на передачу данных.

Список литературы

1. Заборовский В. С. Моделирование и анализ сетей связи с коммутацией пакетов. Network Simulator (Сетевой симулятор ns2). — СПб : Изд-во СПбГТУ, 2001. — 108 с.
2. Exercises on "ns-2" / С. Barakat. — Заявл. 2003.
3. Галкин А. М., Кучерявый Е. А., Молчанов Д. А. Пакет моделирования NS-2: учеб. пособие. — СПб : СПбГУТ, 2007.
4. Заборовский В. С., Мулюха В. А., Подгурский Ю. Е. Моделирование и анализ компьютерных сетей: телематический подход. — СПб: Изд-во СПбГПУ, 2010. — 93 с.