# Лабораторная работа №1

## Простые модели компьютерной сети

Чемоданова Ангелина Александровна

13 февраля 2025

Российский университет дружбы народов имени Патриса Лумумбы, Москва, Россия

# Информация

# Докладчик


- Чемоданова Ангелина Александровна
- Студентка
- Российский университет дружбы народов
- 1132226443@pfur.ru
- https://github.com/aachemodanova


# Цели и задачи

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

# Задание

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

# 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.

```
openmodelica@openmodelica-VirtualBox:~$ mkdir -p mip/lab-ns
openmodelica@openmodelica-VirtualBox:~$ cd mip/lab-ns
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ touch shablon.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ ls
shablon.tcl
```

**Рис. 1:** Создание директорий и файла

Получившийся шаблон можно использовать в дальнейшем в большинстве разрабатываемых скриптов NS-2, добавляя в него до строки $ns at 5.0 “finish” описание объектов и действий моделируемой системы.

# Скрипт шаблона

```
# 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. 2:** Скрипт шаблона

# Симуляция шаблона

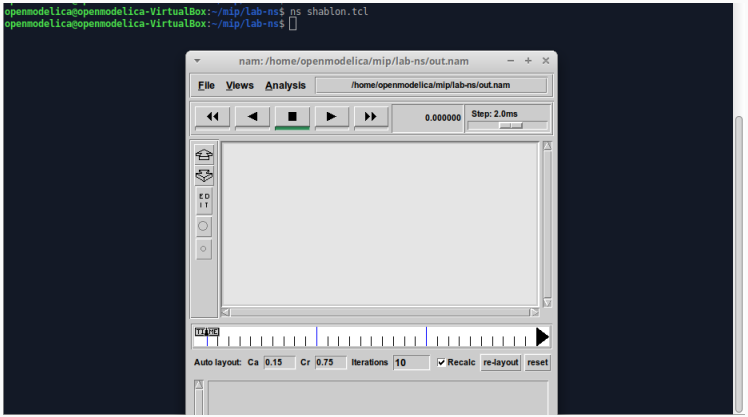


**Рис. 3:** Симуляция шаблона

# 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.

# Реализация модели

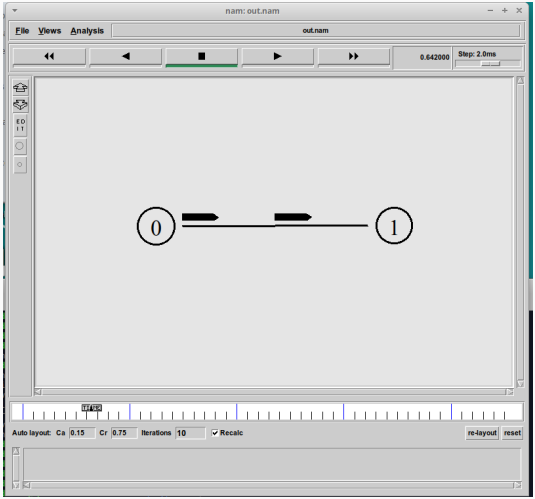


**Рис. 4:** Визуализация простой модели сети с помощью nam

# Скрипт сети из двух узлов и одного соединения

Скопируем содержимое созданного шаблона в новый файл example1.tcl и откроем на редактирование. Создадим агенты для генерации и приёма трафика. Создадим агент UDP и присоединим к узлу n0. К агенту присоединяем приложение. В данном случае — это источник с постоянной скоростью (Constant Bit Rate, CBR), который каждые 5 мс посылает пакет R = 500 байт. 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.

# Скрипт сети из двух узлов и одного соединения

```
# 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,
# очередью 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"
```

**Рис. 5:** Скрипт сети из двух узлов и одного соединения

# Пример с усложненной топологией сети

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.

# Реализация модели.

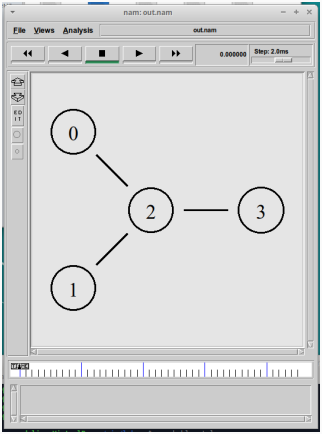


**Рис. 6:** Модель с усложненной топологией сети без симуляции

# Модель с усложненной топологией сети. Симуляция

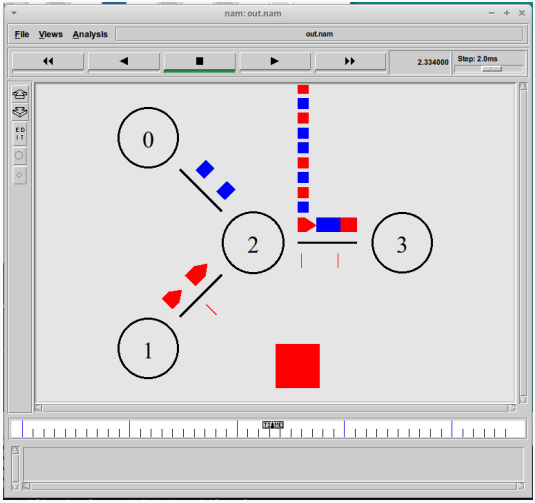


**Рис. 7:** Модель с усложненной топологией сети. Симуляция

# Скрипт модели с усложненной топологией сети

Скопируем содержимое созданного шаблона в новый файл и откроем example2.tcl на редактирование. 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. Создадим агенты-получатели. Соединим агенты udp0 и tcp1 и их получателей. Зададим описание цвета каждого потока. Отслеживание событий в очереди. Наложим ограничения на размер очереди. Добавим at-события.

# Скрипт модели с усложненной топологией сети

```
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

# 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"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr0 stop"
```

# 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.

# Реализация модели.

Скопируем содержимое созданного шаблона в новый файл и откроем example3.tcl на редактирование. Опишем топологию моделируемой сети. 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)

# Передача данных по кратчайшему пути сети с кольцевой топологией

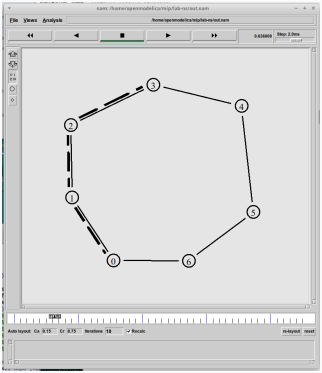


**Рис. 9:** Передача данных по кратчайшему пути сети с кольцевой топологией

# Передача данных по сети с кольцевой топологией в случае разрыва соединения

Добавим команду разрыва соединения между узлами n(1) и n(2) на время в одну секунду, а также время начала и окончания передачи данных. Передача данных при кольцевой топологии сети в случае разрыва соединения.

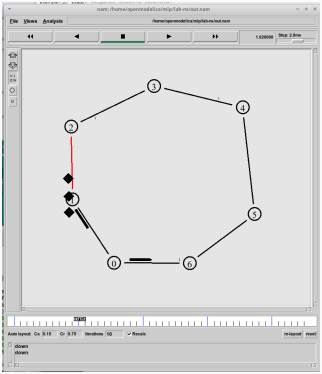


**Рис. 10:** Передача данных по сети с кольцевой топологией в случае разрыва соединения

# Передача данных по сети с кольцевой топологией в случае разрыва соединения

Добавив в начало скрипта после команды создания объекта Simulator, 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).

# Скрипт модели с кольцевой топологией

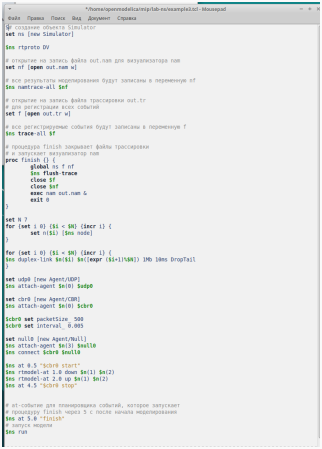


**Рис. 11:** Скрипт модели с кольцевой топологией

# Выполнение упражнения

Упражнение. Внесите следующие изменения в реализацию примера с кольцевой топологией сети: – повторить топологию сети, предоставленную в файле с заданиями к лабораторной работе; – 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.

# Скрипт модели из упражнения

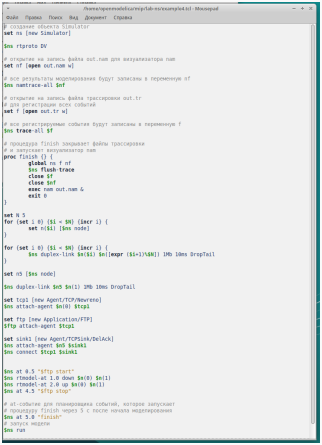


**Рис. 12:** Скрипт модели из упражнения

# Топология сети.

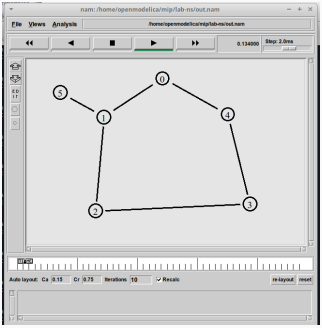


**Рис. 13:** Топология сети из упражнения

# Движение по кратчайшему пути.

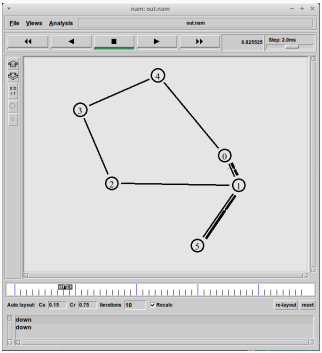


**Рис. 14:** Движение по кратчайшему пути

# Разрыв связи между узлами.

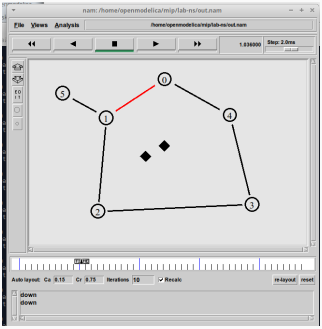


**Рис. 15:** Разрыв связи между узлами

# Движение по длинному пути.

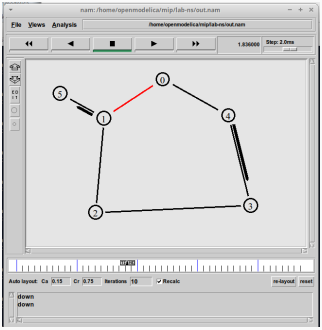


**Рис. 16:** Движение по длинному пути

# Выводы

# Выводы

Мы приобрели навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также провели анализ полученных результатов моделирования.