

Лабораторная работа 1

Простые модели компьютерной сети

Ланцова Я. И.

Российский университет дружбы народов, Москва, Россия

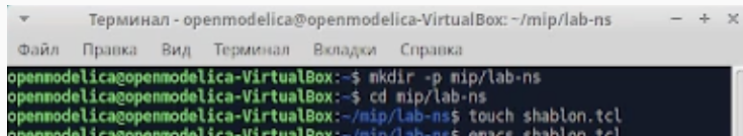
Информация

- Ланцова Яна Игоревна
- студентка
- Российский университет дружбы народов

Приобрести навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также проанализировать полученные результаты моделирования.

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

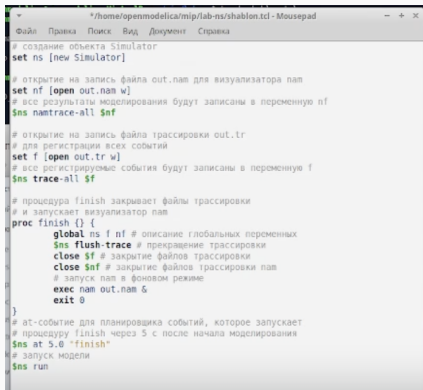
Выполнение лабораторной работы



```
Терминал - openmodelica@openmodelica-VirtualBox: ~/mip/lab-ns
Файл  Правка  Вид  Терминал  Вкладки  Справка
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$ cp shablon.tcl
```

Рис. 1: Создание директорий и файла

Выполнение лабораторной работы



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

# создание объекта Simulator
set ns [new Simulator]

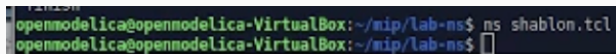
# открытие на запись файла out.nam для визуализатора nam
set nf [open out.nam w]
# все результаты моделирования будут записаны в переменную nf
$ns namtrace-all $nf

# 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 # описание глобальных переменных
    $ns flush-trace # прекращение трассировки
    close $f # закрытие файлов трассировки
    close $nf # закрытие файлов трассировки nam
    # запуск nam в фоновом режиме
    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: Создадим объект типа Simulator. Затем создадим переменную nf



```
openmodelica@openmodelica-VirtualBox:~/nip/lab-ns$ ns shablon.tcl  
openmodelica@openmodelica-VirtualBox:~/nip/lab-ns$
```

Рис. 3: Запустим симулятор командой

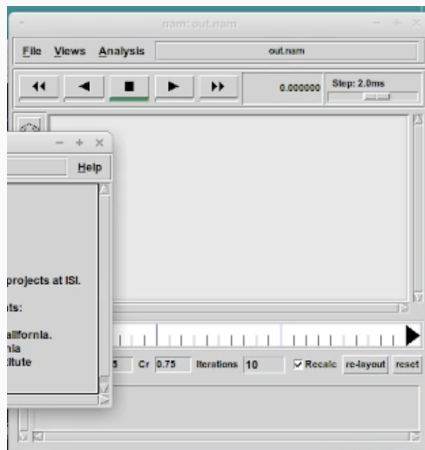
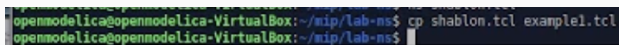


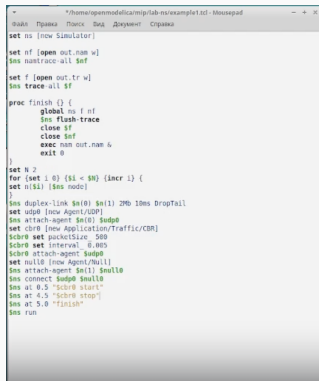
Рис. 4: Увидим пустую область моделирования, поскольку ещё не определены никакие объекты и действия

A screenshot of a terminal window with a dark background. The prompt is 'openmodelica@openmodelica-VirtualBox:~/mip/lab-ns\$'. The command 'cp shablon.tcl example1.tcl' has been entered and executed. The prompt is now 'openmodelica@openmodelica-VirtualBox:~/mip/lab-ns\$' with a cursor at the end.

```
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ cp shablon.tcl example1.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$
```

Рис. 5: Скопируем содержимое созданного шаблона в новый файл

Выполнение лабораторной работы

A screenshot of a text editor window titled "/home/openmodelica/mip/lab-ns/example1.tcl - Mousepad". The editor contains a NetSim configuration script. The script starts with "set ns [new Simulator]" and "set nf [open out.nam w]". It then sets up namespaces for traffic generation and reception. A "proc finish" block handles cleanup. The main loop "for {set i 0} {\$i < \$N} {incr i}" creates nodes and sets up a duplex link between node 0 and node 1. It then creates a UDP agent "udp0" and attaches it to node 0. A CBR source "cbr0" is attached to the UDP agent. The script sets various parameters like packet size, interval, and buffer size. It also sets up a null agent "null0" and connects it to the CBR source. Finally, it sets up a trace and runs the simulation.

```
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 0
}

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/CDR]
$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"
$ns run
```

Рис. 6: Создадим агенты для генерации и приёма трафика. Создается агент UDP и присоединяется к узлу n0

Выполнение лабораторной работы

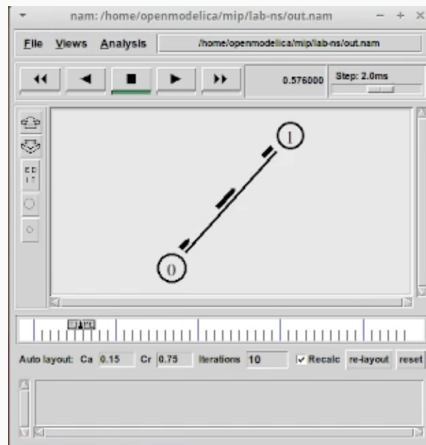
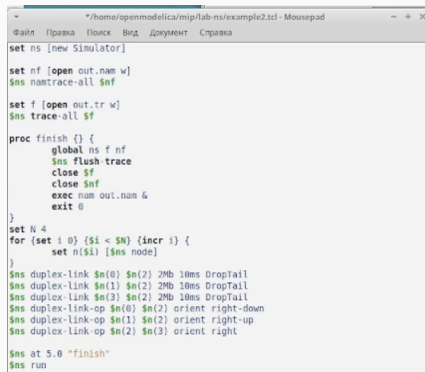


Рис. 7: Сохранив изменения в отредактированном файле и запустив симулятор, получим в качестве результата запуск аниматора nam в фоновом режиме

Выполнение лабораторной работы



```
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 0
}

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

$ns at 5.0 "finish"
$ns run
```

Рис. 8: Откроем example2.tcl на редактирование. Создадим 4 узла и 3 дуплексных соединения с указанием направления

```
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
```

Рис. 9: Создадим агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP

```
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 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"
```

Рис. 10: Создадим агенты-получатели. Соединим агенты udp0 и tcp1 и их получателей.

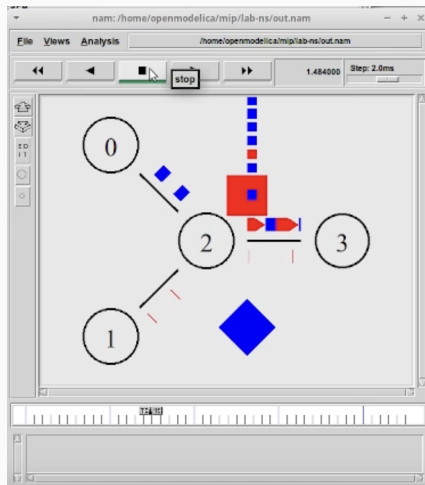
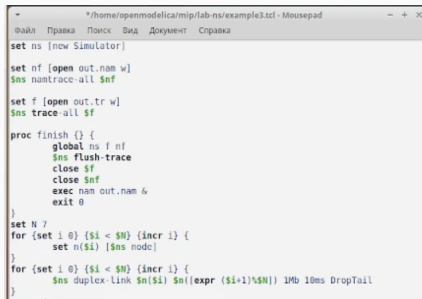


Рис. 11: Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования

A screenshot of a text editor window titled '* /home/openmodelica/mip/lab-ns/example3.tcl - Mousepad'. The window contains a NetSim script. The script starts with 'set ns [new Simulator]', followed by opening output files for node names and traces. It then defines a 'finish' procedure to flush traces and close files. Finally, it sets up a network with 7 nodes and connects them in a ring topology using 'duplex-link' with 1Mb bandwidth and 10ms delay.

```
* /home/openmodelica/mip/lab-ns/example3.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 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
}
```

Рис. 12: Скопируем содержимое созданного шаблона в новый файл. Опишем топологию моделируемой сети. Далее соединим узлы

```
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"
```

Рис. 13: 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 $n(2)$.

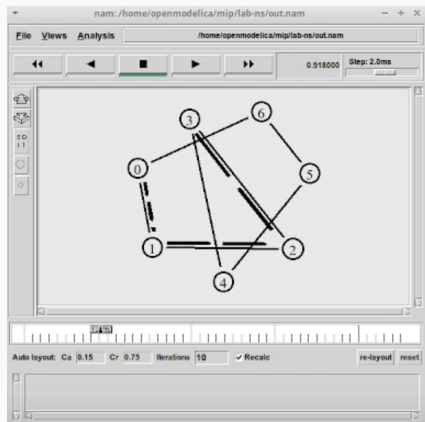


Рис. 14: Добавив в начало скрипта после команды создания объекта Simulator. Увидим, что сразу после запуска в сети отправляется

Выполнение лабораторной работы

```
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}]) 1Mb 10ms DropTail
}
set n5 [$ns node]
$ns duplex-link $n5 $n(1) 1Mb 10ms DropTail
set tcpl [new Agent/TCP/Newreno]
$ns attach-agent $n(0) $tcpl
set ftp [new Application/FTP]
$ftp attach-agent $n(0) $tcpl
set sink1 [new Agent/TCP/Sink/DelAck]
$ns attach-agent $n5 $sink1
$ns connect $tcpl $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. 15: Изменим количество узлов в кольце на 5, а 6 узел n(5) отдельно присоединим к узлу n(1).
Вместо агента UDP создадим агента TCP (типа Newreno)

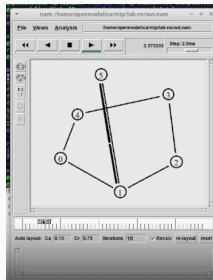


Рис. 16: Запустим программу и увидим, что пакеты идут по кратчайшему пути через узел n(1)

В процессе выполнения данной лабораторной работы я приобрела навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также проанализировала полученные результаты моделирования.