

Лабораторная работа 1

Простые модели компьютерной сети

Клюкин Михаил Александрович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Шаблон сценария для NS-2	7
3.2	Простой пример описания топологии сети, состоящей из двух узлов и одного соединения	8
3.3	Пример с усложненной топологией сети	10
3.4	Пример с кольцевой топологией	14
3.5	Упражнение	18
4	Выводы	24
	Список литературы	25

Список иллюстраций

3.1	Создание директорий и шаблона	7
3.2	Запуск шаблона сценария для NS-2	8
3.3	Визуализация простой модели сети с помощью nam	9
3.4	Передача данных в простой модели сети	10
3.5	Модель сети с усложненной топологией	12
3.6	Модель сети с усложненной топологией	13
3.7	Потеря пакетов в модели с усложненной топологией	14
3.8	Модель сети с кольцевой топологией	15
3.9	Передача данных между узлами $n(0)$ и $n(3)$ по кратчайшему пути	16
3.10	Потеря пакетов при разрыве сети	17
3.11	Маршрутизация данных по сети с кольцевой топологией в случае разрыва сети	18
3.12	Измененная кольцевая топология сети	19
3.13	Передача данных между узлами $n(0)$ и $n(5)$ по кратчайшему пути	20
3.14	Потеря пакетов при разрыве соединения	21
3.15	Передача данных между узлами $n(0)$ и $n(5)$ по альтернативному пути	22
3.16	Передача данных между узлами $n(0)$ и $n(3)$ по кратчайшему пути после восстановления соединения	23

Список таблиц

1 Цель работы

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

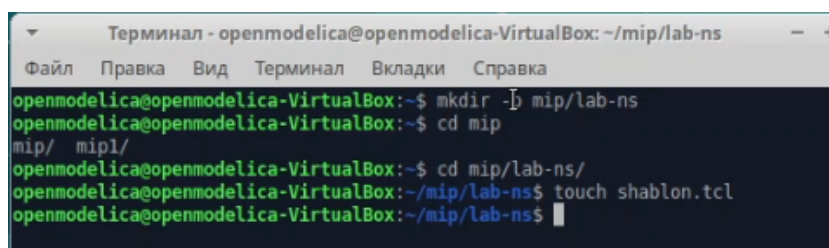
2 Задание

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 усложненной топологией сети;
4. Выполнить пример с кольцевой топологией сети;
5. Выполнить упражнение.

3 Выполнение лабораторной работы

3.1 Шаблон сценария для NS-2

В своем рабочем каталоге создадим директорию `mip`, в которой будут выполняться лабораторные работы. Внутри `mip` создадим директорию `lab-ns`, а в ней файл `shablon.tcl` (рис. 3.1).



```
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
mip/ mip1/
openmodelica@openmodelica-VirtualBox:~$ cd mip/lab-ns/
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$ touch shablon.tcl
openmodelica@openmodelica-VirtualBox:~/mip/lab-ns$
```

Рис. 3.1: Создание директорий и шаблона

Откроем на редактирование файл `shablon.tcl`.

Сначала создадим объект типа `Simulator`. Затем создадим переменную `nf` и укажем, что требуется открыть на запись `nam`-файл для регистрации выходных результатов моделирования. Далее создадим переменную `f` и откроем на запись файл трассировки для регистрации всех событий модели. После этого добавим процедуру `finish`, которая закрывает файлы трассировки и запускает `nam`. Наконец, с помощью команды `at` указываем планировщику событий, что процедуру `finish` следует запустить через 5 с после начала моделирования, после чего запустить симулятор `ns`.

Сохранив изменения в отредактированном файле `shablon.tcl` и закрыв его, запустив симулятор командой `ns shablon.tcl`. Увидим пустую область

моделирования, поскольку еще не определены никакие объекты и действия (рис. 3.2).

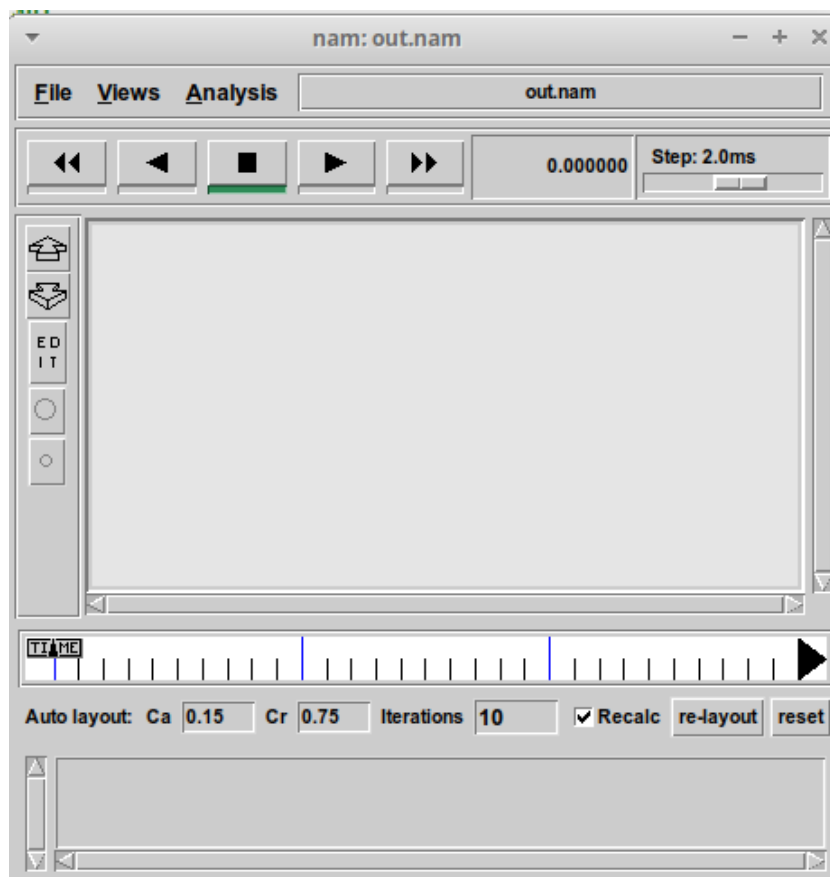


Рис. 3.2: Запуск шаблона сценария для NS-2

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

Скопируем содержимое созданного шаблона в новый файл: `cp shablon.tcl example1.tcl`. Откроем `example1.tcl` на редактирование. Добавим в него до строки `$ns at 5.0 "finish"` описание топологии сети. Создадим агента UDP и присоединим его к узлу `n0`. К агенту присоединим 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 (перед командой `$ns at 5.0 "finish"`). Сохранив изменения в отредактированном файле и запустив симулятор с помощью команды `ns example1.tcl`, получим в качестве результата запуск аниматора nam в фоновом режиме (рис. 3.3).

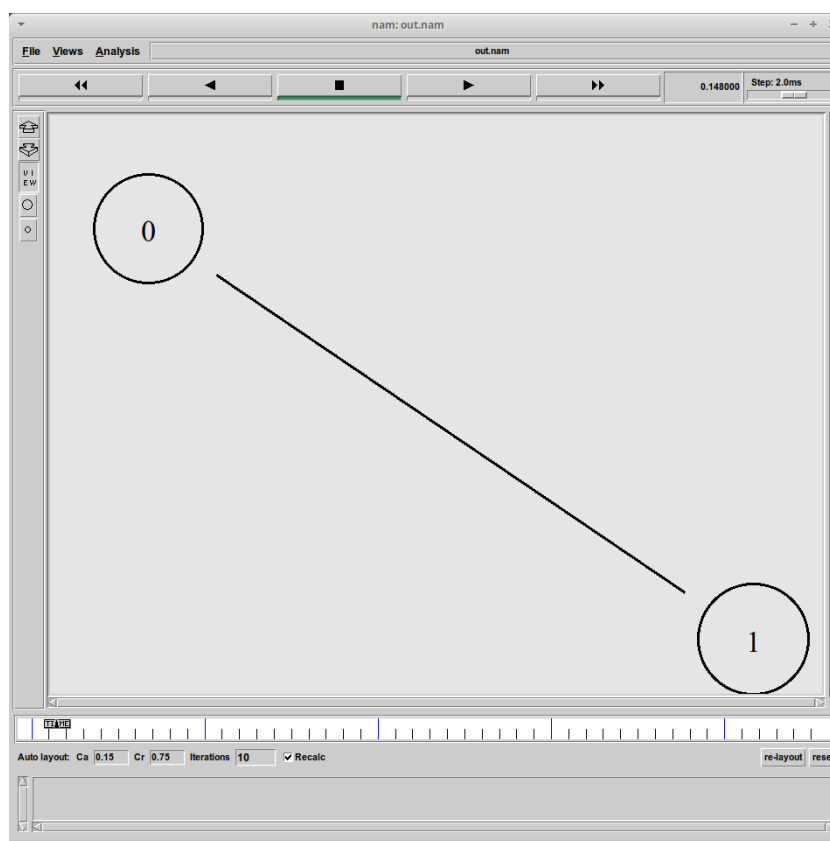


Рис. 3.3: Визуализация простой модели сети с помощью nam

При нажатии на кнопку play в окне nam через 0.5 секунды из узла 0 данные начнут поступать к узлу 1 (рис. 3.4).

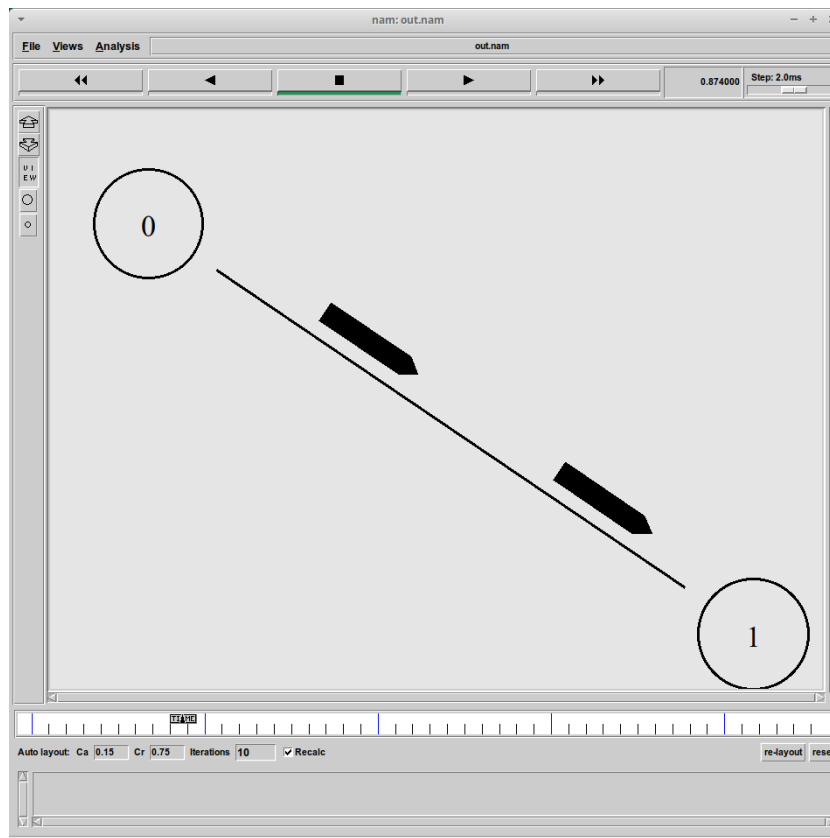


Рис. 3.4: Передача данных в простой модели сети

3.3 Пример с усложненной топологией сети

Описание сети:

- 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 пакет 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.

Скопируем содержимое созданного шаблона в новый файл `cp shablon.tcl example2.tcl` 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 `udr0` и `tcp1` и их получателей. Зададим описание цвета каждого потока. Зададим отслеживание событий в очереди. Наложим ограничения на размер очереди. Добавим `at`-событий.

Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования (рис. 3.5).

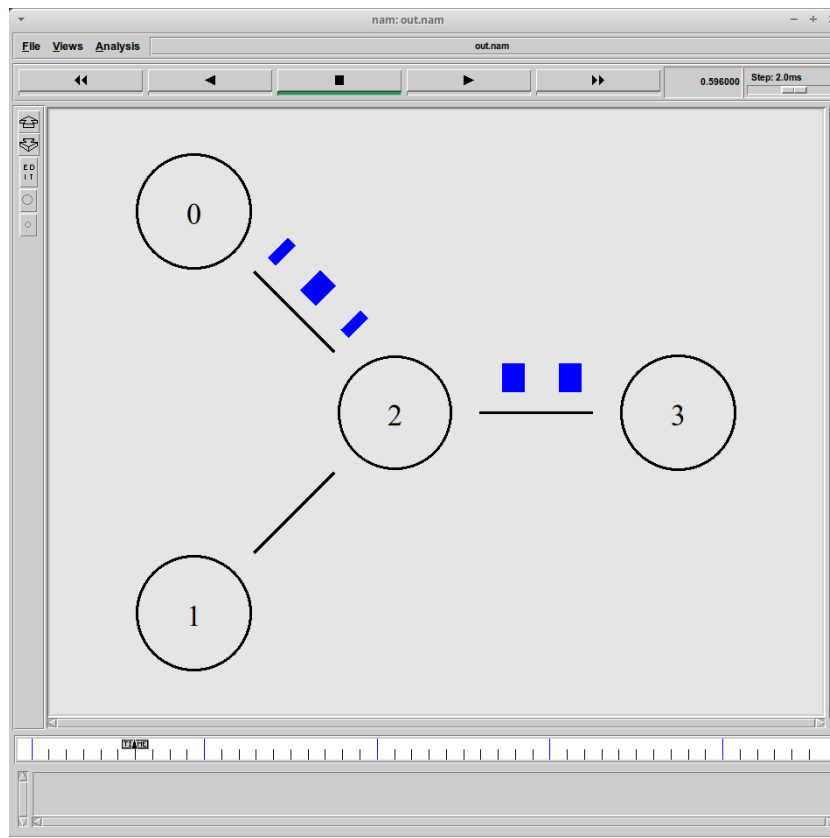


Рис. 3.5: Модель сети с усложненной топологией

При запуске скрипта можно заметить, что по соединениям между узлами $n(0)-n(2)$ и $n(1)-n(2)$ 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 — 1,6 Mb. Но соединение $n(2)-n(3)$ имеет полосу лишь 1 Mb. Таким образом, пакеты, которые в данный момент не могут быть обработаны, попадают в очередь (рис. 3.6).

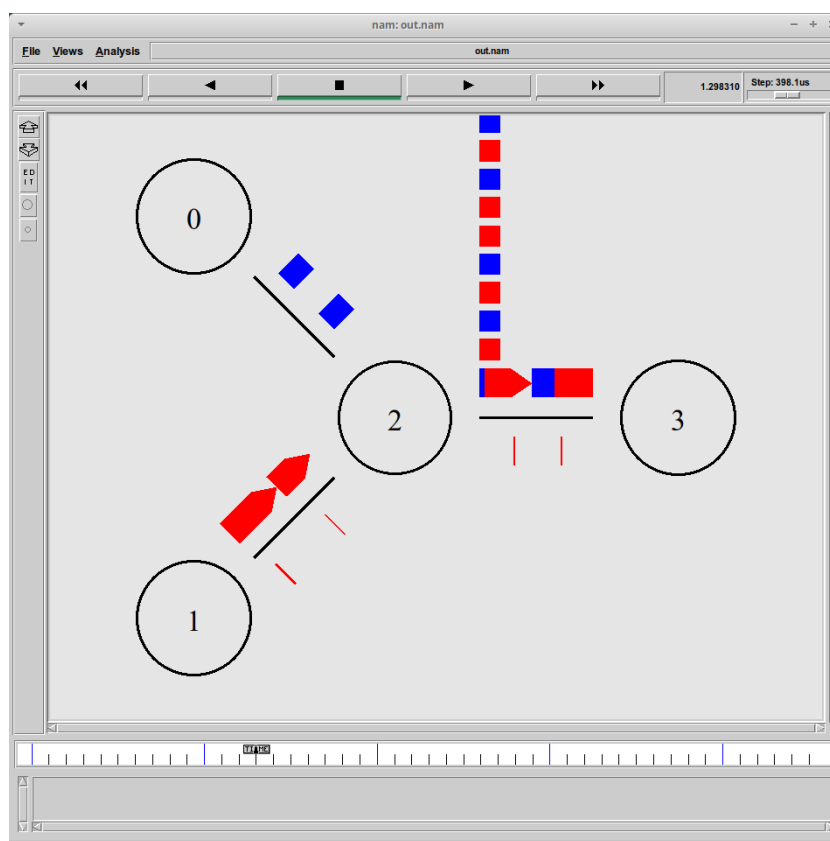


Рис. 3.6: Модель сети с усложненной топологией

В окне аниматора можно видеть пакеты в очереди, а также те пакеты, которые отбрасываются при переполнении (рис. 3.7).

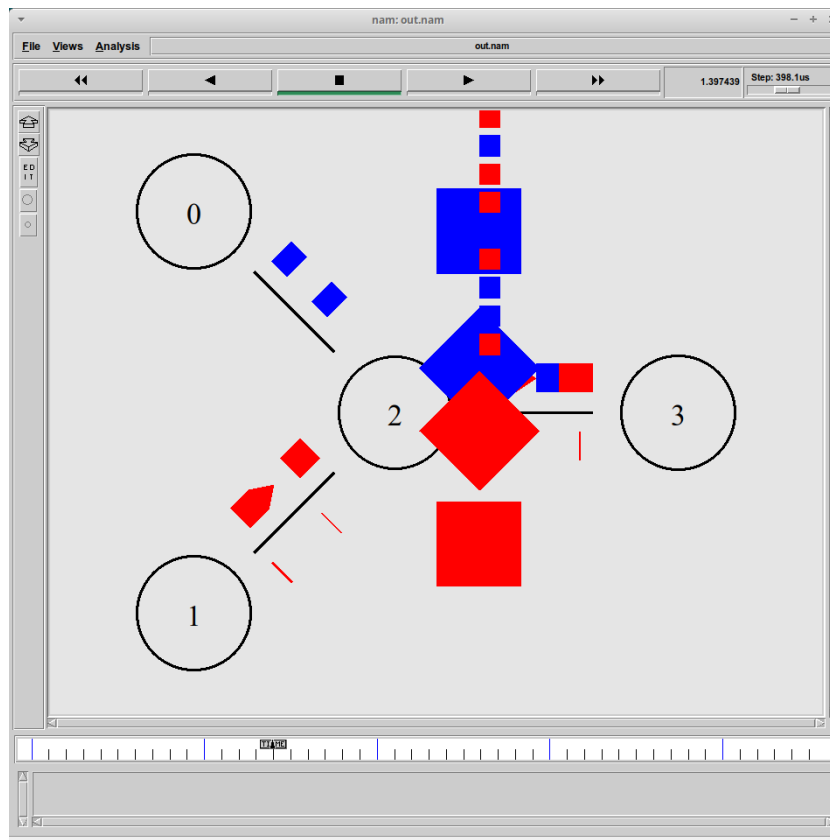


Рис. 3.7: Потеря пакетов в модели с усложненной топологией

3.4 Пример с кольцевой топологией

Описание модели 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.

Скопируем содержимое созданного шаблона в новый файл `cp shablon.tcl example3.tcl` 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 (рис. 3.8).

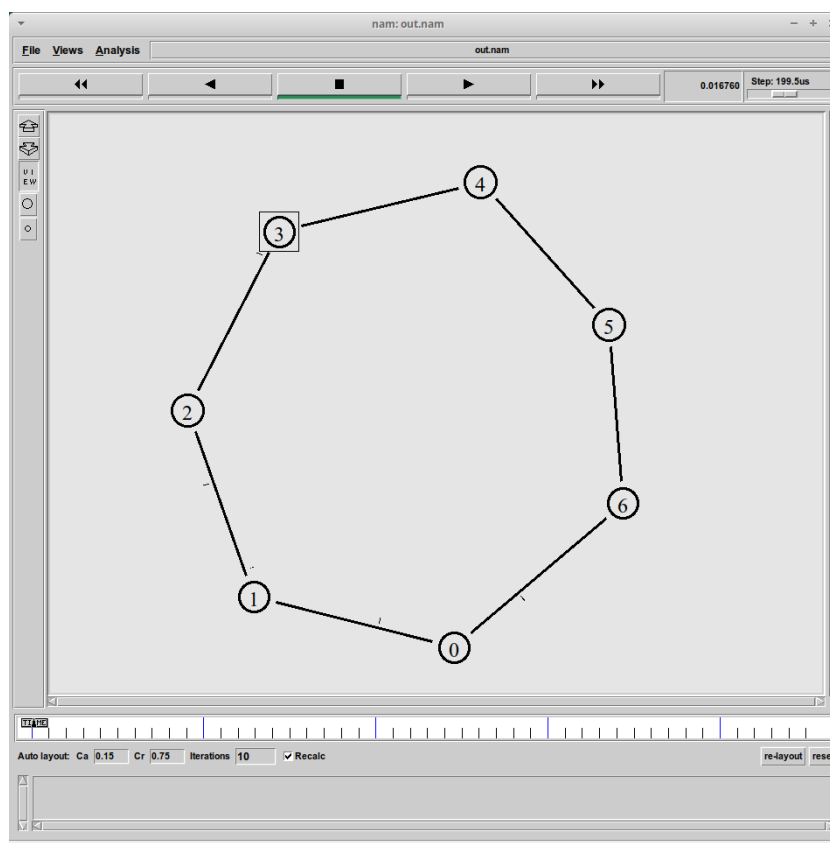


Рис. 3.8: Модель сети с кольцевой топологией

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. 3.9).

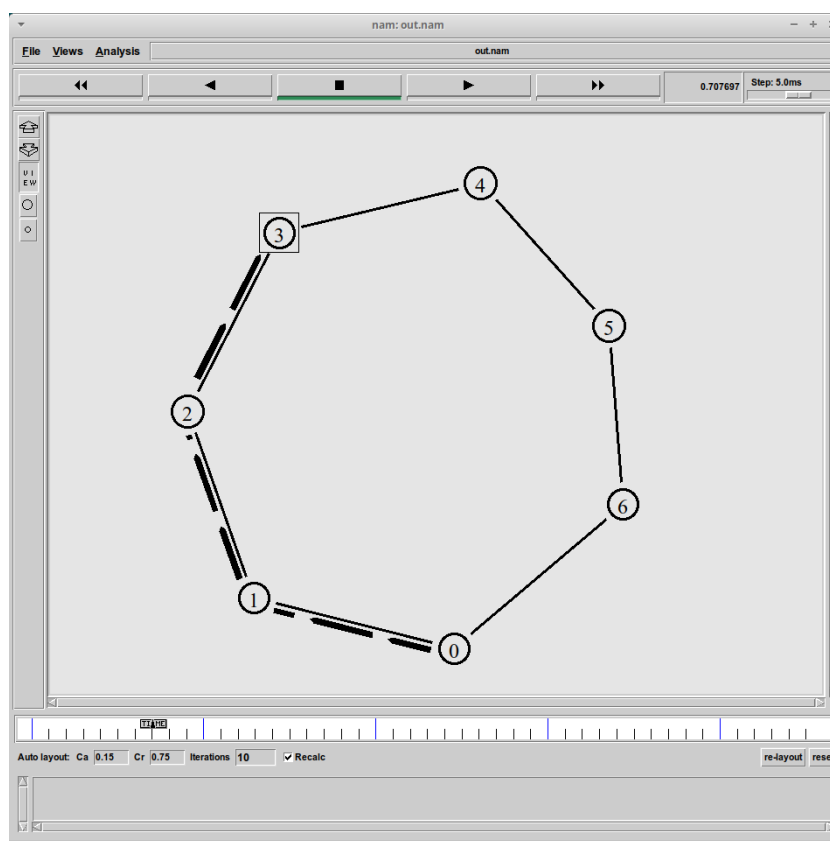


Рис. 3.9: Передача данных между узлами $n(0)$ и $n(3)$ по кратчайшему пути

Добавим команду разрыва соединения между узлами $n(1)$ и $n(2)$ на время в одну секунду (рис. 3.10), а также время начала и окончания передачи данных.

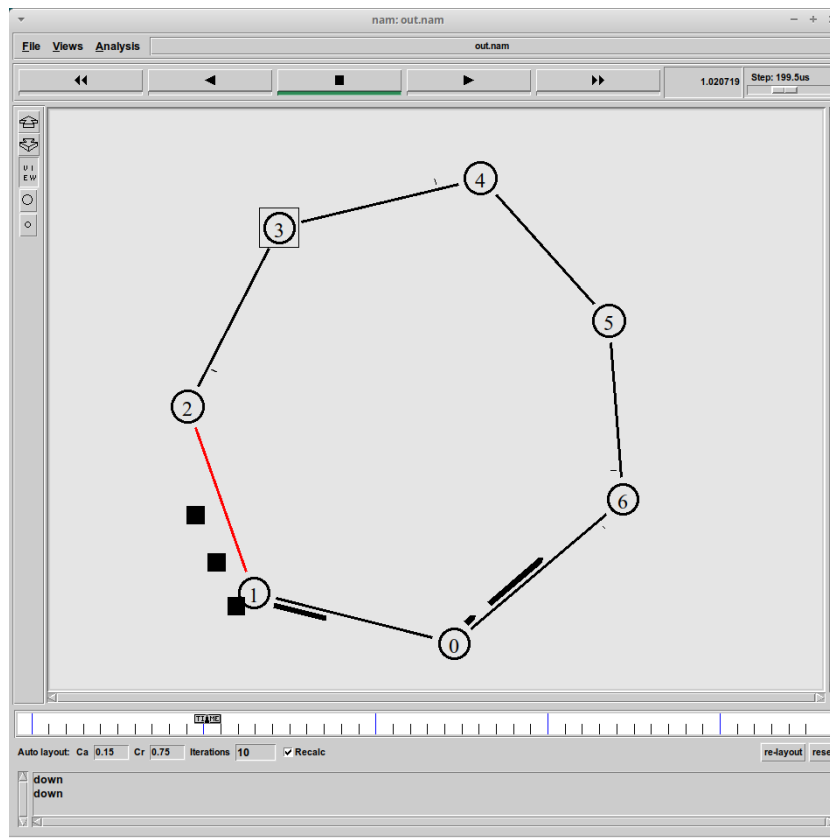


Рис. 3.10: Потеря пакетов при разрыве сети

Добавив в начало скрипта после команды создания объекта Simulator `$ns rtproto DV` увидим, что сразу после запуска в сети отправляется небольшое количество маленьких пакетов, используемых для обмена информацией, необходимой для маршрутизации между узлами (рис. 3.11). Когда соединение будет разорвано, информация о топологии будет обновлена, и пакеты будут отсылаться по новому маршруту через узлы $n(6)$, $n(5)$ и $n(4)$.

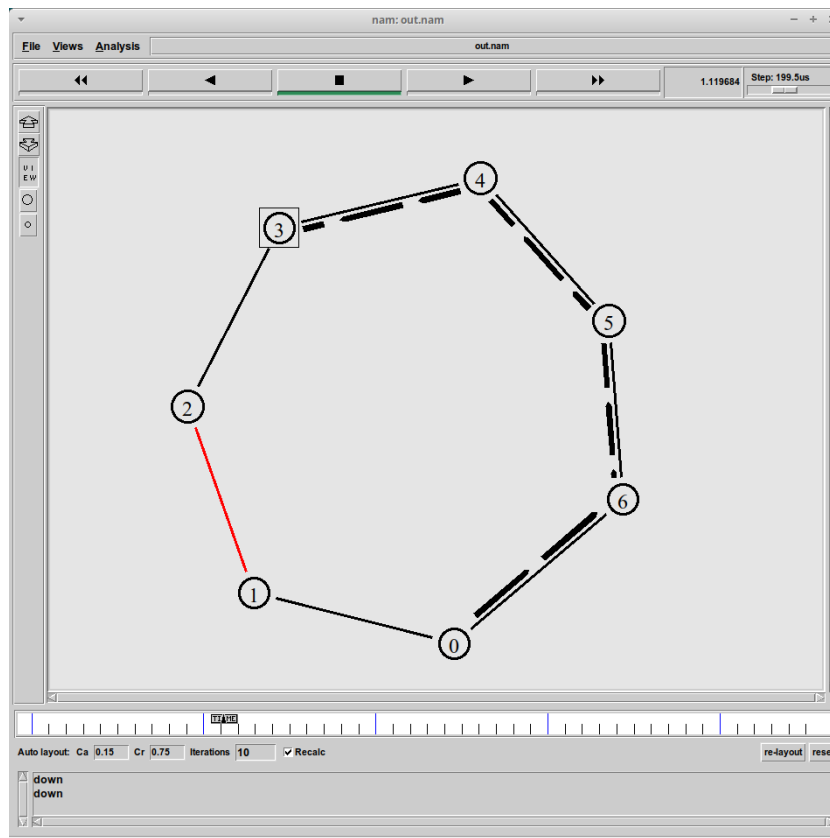


Рис. 3.11: Маршрутизация данных по сети с кольцевой топологией в случае разрыва сети

3.5 Упражнение

Описание 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.

Изменим количество узлов в кольце на 5, а 6 узел $n(5)$ присоединим отдельно к узлу $n(1)$. Вместо агента UDP создадим агент TCP (тип Newreno), а на принимающей стороне используем TCPSink-объект типа DelACK; поверх TCP работает протокол FTP с 0,5 до 4,5 секунд модельного времени. Также зададим с 1 по 2 секунду модельного времени зададим разрыв соединения между узлами $n(0)$ и $n(1)$.

Сама модель представлена на рис. 3.12.

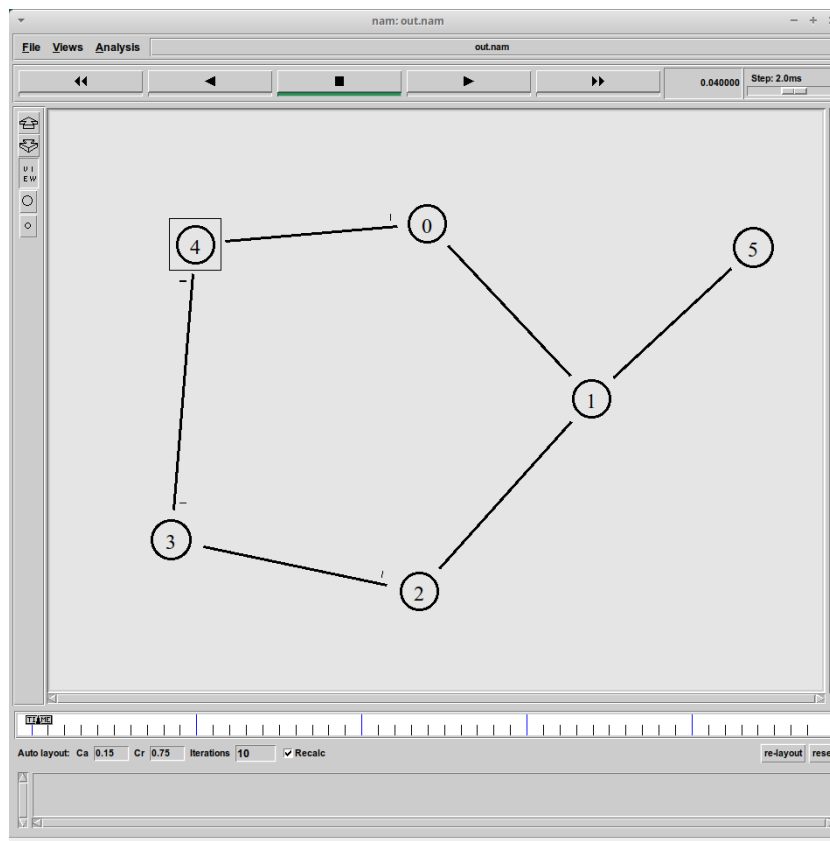


Рис. 3.12: Измененная кольцевая топология сети

Запустим программу и увидим, что пакеты идут по кратчайшему пути через узел $n(1)$ (рис. 3.13).

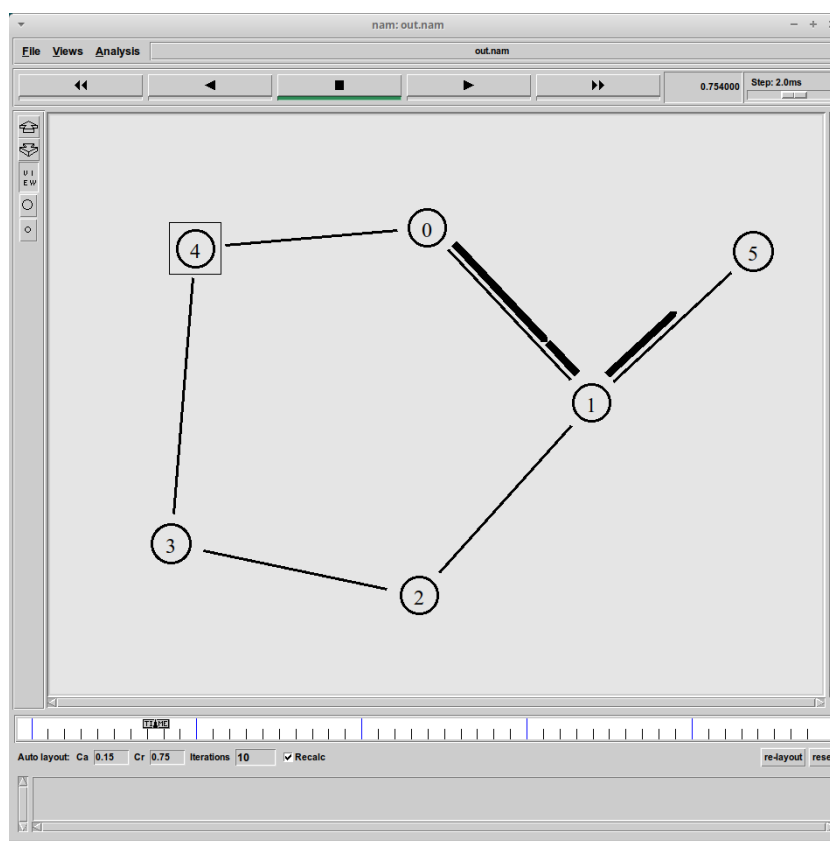


Рис. 3.13: Передача данных между узлами $n(0)$ и $n(5)$ по кратчайшему пути

При разрыве соединения часть пакетов теряется (рис. 3.14).

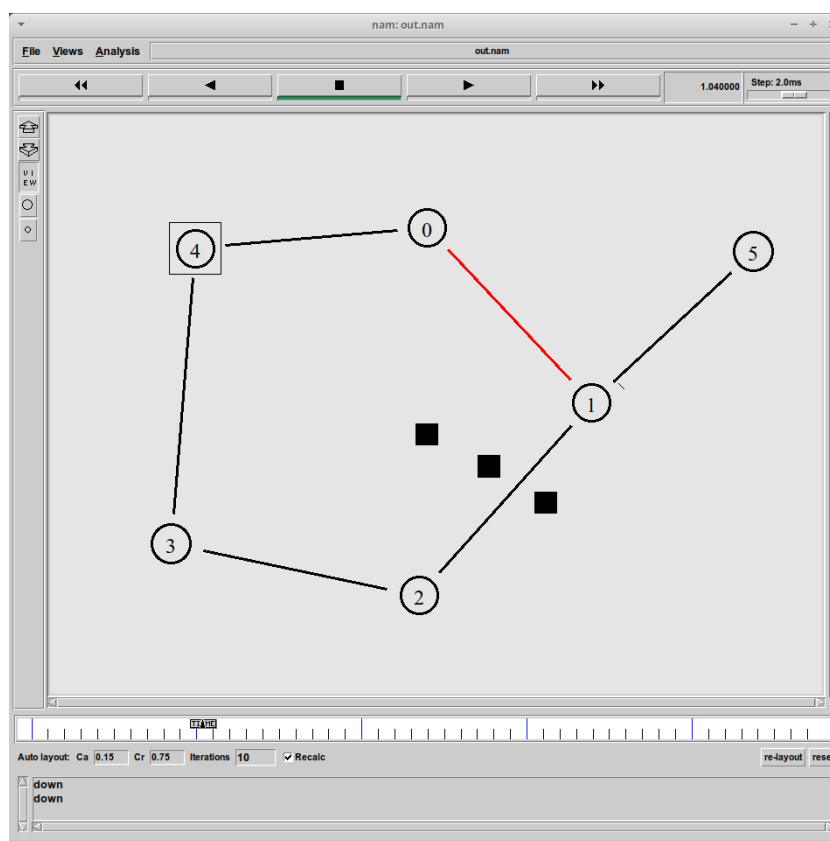


Рис. 3.14: Потеря пакетов при разрыве соединения

Но поскольку данные обновляются, пакеты начинают идти по другому пути (рис. 3.15).

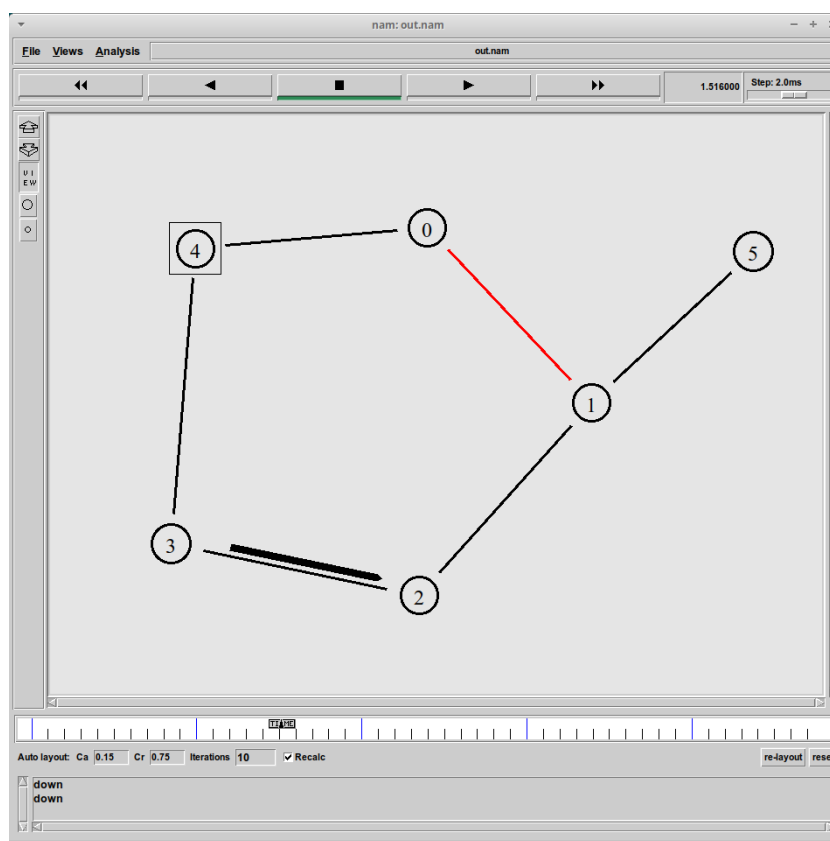


Рис. 3.15: Передача данных между узлами $n(0)$ и $n(5)$ по альтернативному пути

После восстановления соединения пакеты снова идут по кратчайшему пути (рис. 3.16).

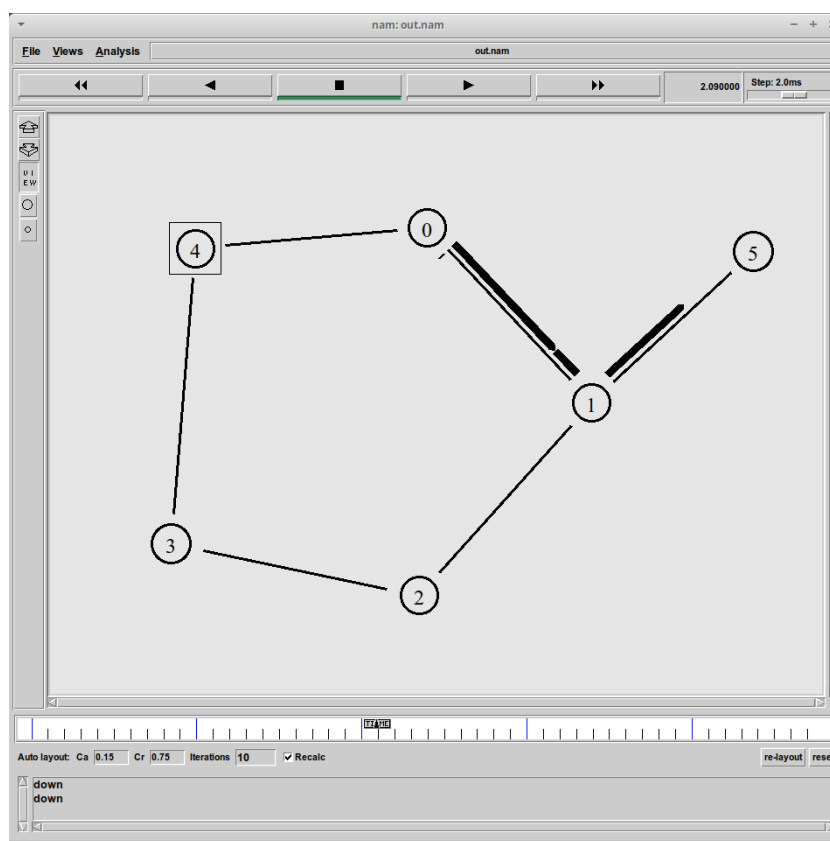


Рис. 3.16: Передача данных между узлами $n(0)$ и $n(3)$ по кратчайшему пути после восстановления соединения

4 Выводы

В процессе выполнения лабораторной работы приобрели навыки моделирования сетей передачи данных с помощью средств имитационного моделирования NS-2, а также проанализировали полученные результаты моделирования.

Список литературы