Лабораторная работа №1

Простые модели компьютерной сети

Астраханцева А. А.

Содержание

# 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. Создание шаблона сценария для NS-2.
2. Выполнение примера описания топологии сети, состоящей из двух узлов и одного соединения.
3. Выполнение примера описания с усложнённой топологией сети.
4. Выполнение примера с кольцевой топологией сети
5. Выполнение упражнения

# 3 Выполнение лабораторной работы

1. Создание шаблона сценария для NS-2. В своём рабочем каталоге создаем директорию mip, в которой будут выполняться лабораторные работы. Внутри mip создаем директорию lab-ns, а в ней файл shablon.tcl: (рис. 1).

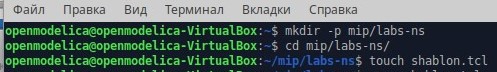


Рис. 1: Создание необходимых директорий и файла

Откроем на редактирование файл shablon.tcl. 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. Полный текст шаблона представлен на скриншоте (рис. 2).

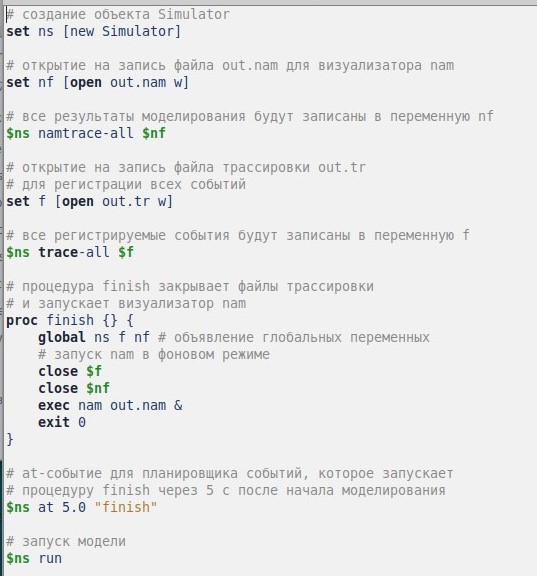


Рис. 2: Скрипт для шаблона

Сохранив изменения в отредактированном файле shablon.tcl и закрыв его, запускаем симулятор. 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 (рис. 3).

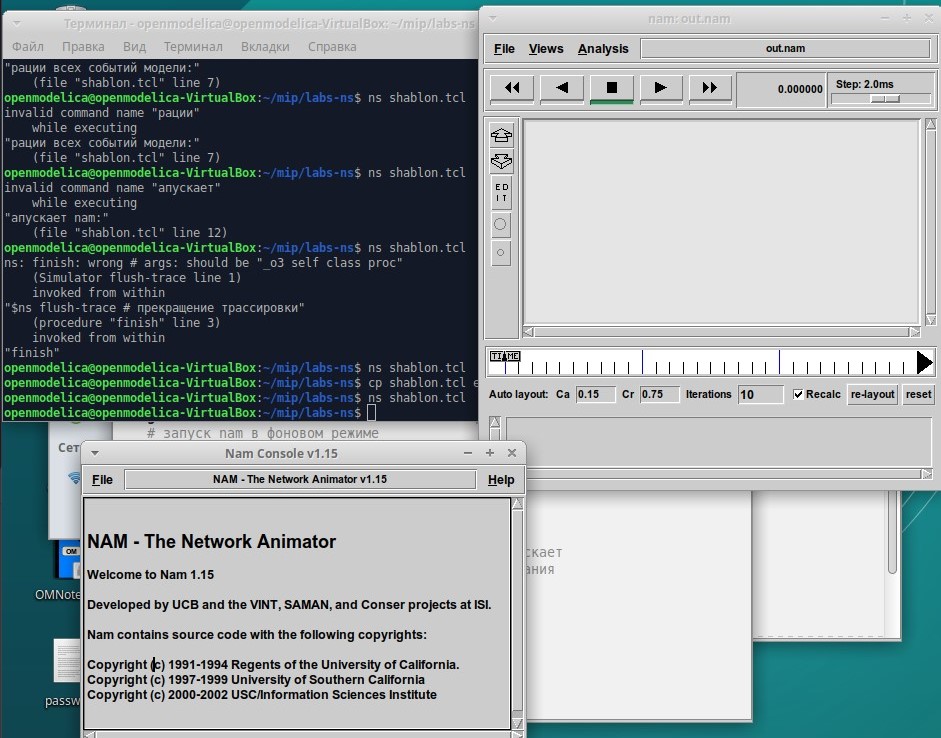


Рис. 3: Запуск симулятора

1. Выполнение примера 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.

Скопируем содержимое созданного шаблона в новый файл example1.tcl (рис. 4).

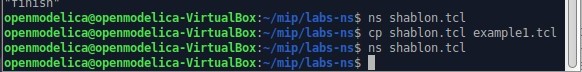


Рис. 4: Копирование шаблона в файл example1.tcl

Откроем example1.tcl на редактирование. Добавим в него до строки $ns at 5.0 “finish” описание топологии сети. Создадим агент UDP и присоединим к узлу n0. В узле агент сам не может генерировать трафик, он лишь реализует протоколы и алгоритмы транспортного уровня. Поэтому к агенту присоединяется приложение. В данном случае — это источник с постоянной скоростью (Constant Bit Rate, CBR), который каждые 5 мс посылает пакет R = 500 байт. Таким образом, скорость источника:

Cоздадим Null-агент, который работает как приёмник трафика, и прикрепим его к узлу n1, после чего соединим агенты между собой. Для запуска и остановки приложения CBR добавим at-события в планировщик событий (перед командой $ns at 5.0 “finish”) (рис. 5).

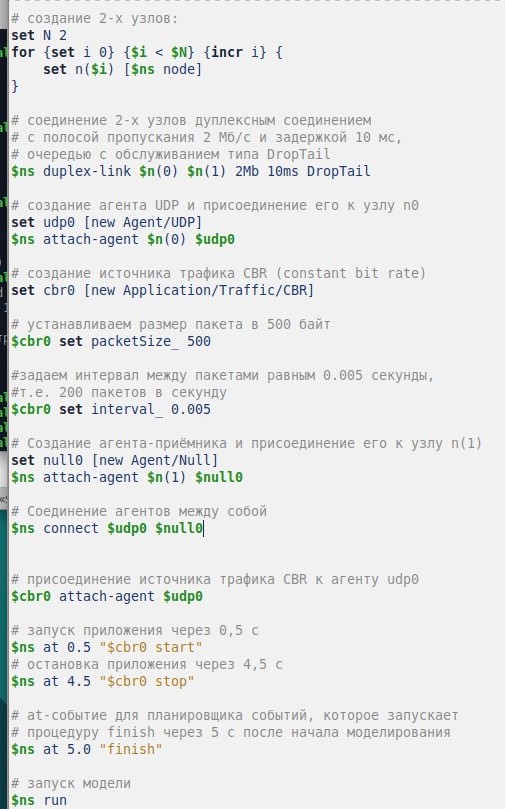


Рис. 5: Скрипт для описания топологии сети, состоящей из двух узлов и одного соединения

Сохраним изменения в отредактированном файле и запустим симулятор. Получим в качестве результата запуск аниматора nam в фоновом режиме. При нажатии на кнопку play в окне nam через 0.5 секунды из узла 0 данные начнут поступать к узлу 1. (рис. 6).

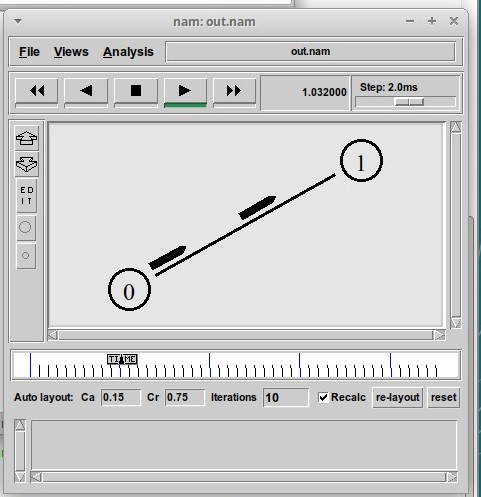


Рис. 6: Анимация сети, состоящей из двух узлов и одного соединения

Этот 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. 7).

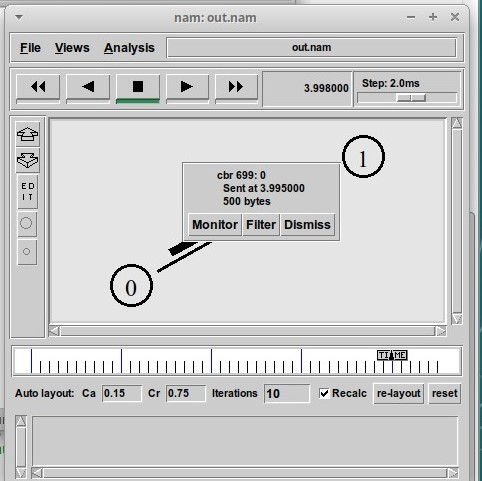


Рис. 7: Наблюдение за отдельным пакетом в аниматоре nam

1. Выполнение примера описания с усложнённой топологией сети.

**Постановка задачи**. Описание моделируемой сети (рис. 2.4): - 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.

Скопируем содержимое созданного шаблона в новый файл example2.tcl (рис. 8).

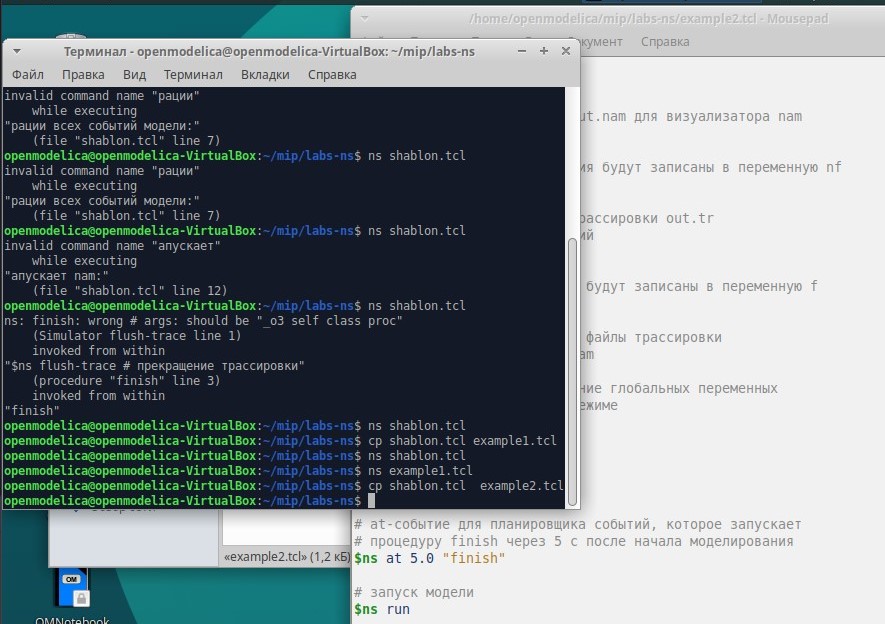


Рис. 8: Копирование шаблона в файл example2.tcl

Откроем example2.tcl на редактирование. Создадим 4 узла и 3 дуплексных соединения с указанием направления:(рис. 9).

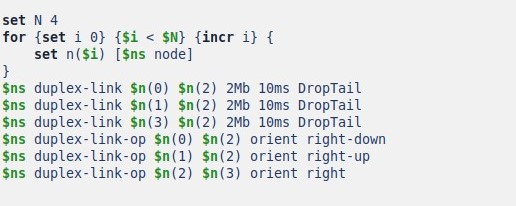


Рис. 9: Создание 4-х узлов и 3-х дуплексных соединения с указанием направления

Создадим агент UDP с прикреплённым к нему источником CBR и агент TCP с прикреплённым к нему приложением FTP: (рис. 10).

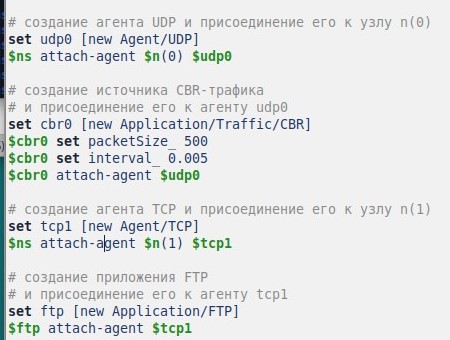


Рис. 10: Создание агента UDP с прикреплённым к нему источником CBR и агента TCP с прикреплённым к нему приложением FTP

Создадим агенты-получатели. Соединим агенты udp0 и tcp1 и их получателей (рис. 11).

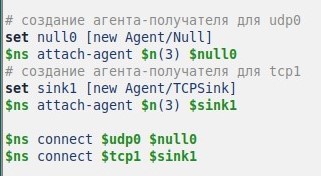


Рис. 11: Создание агентов-получателей. Соединение агентов udp0 и tcp1 и их получателей

Зададим описание цвета каждого потока. Добавим отслеживание событий в очереди и наложение ограничения на размер очереди: (рис. 12).

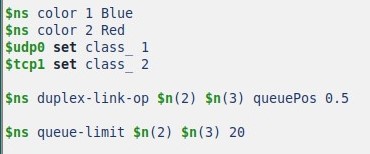


Рис. 12: Задание цвета потока. Отслеживание событий в очереди и наложение ограничения на размер очереди

Добавим at-события: (рис. 13).



Рис. 13: Добавление at-событий

Сохранив изменения в отредактированном файле и запустив симулятор, получим анимированный результат моделирования. 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. Следовательно, часть пакетов должна теряться. В окне аниматора можно видеть пакеты в очереди, а также те пакеты, которые отбрасываются при переполнении. (рис. 14).

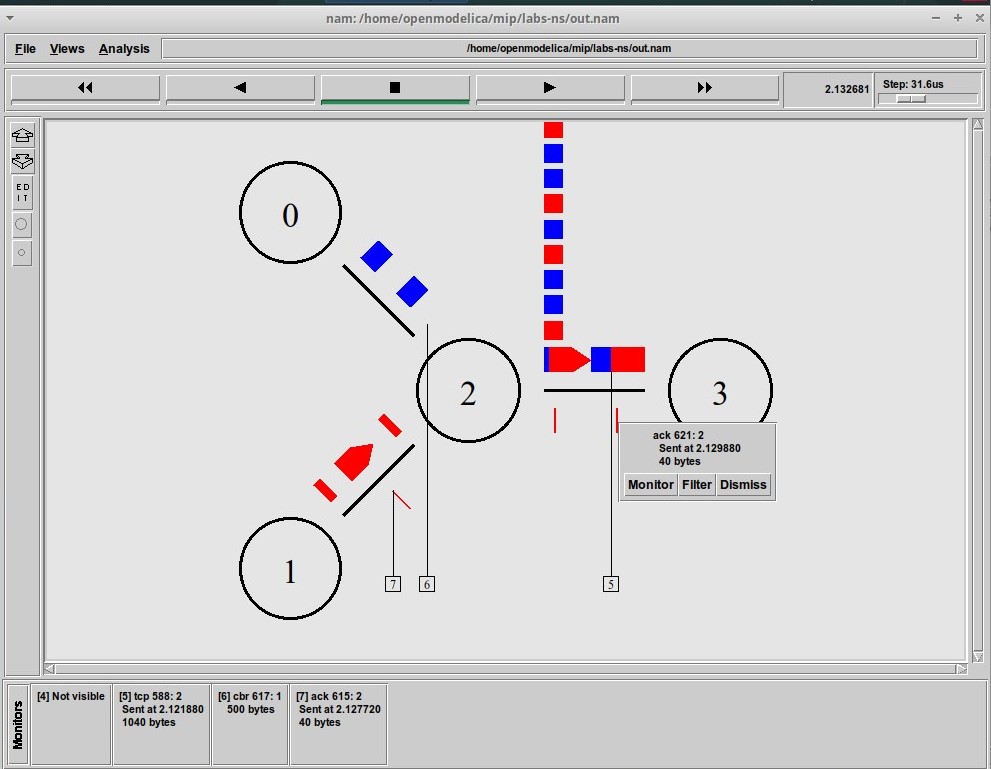


Рис. 14: Запуск анимации сети с усложненной топологией

1. Выполнение примера 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.

Скопируем содержимое созданного шаблона в новый файл example3.tcl (рис. 15).

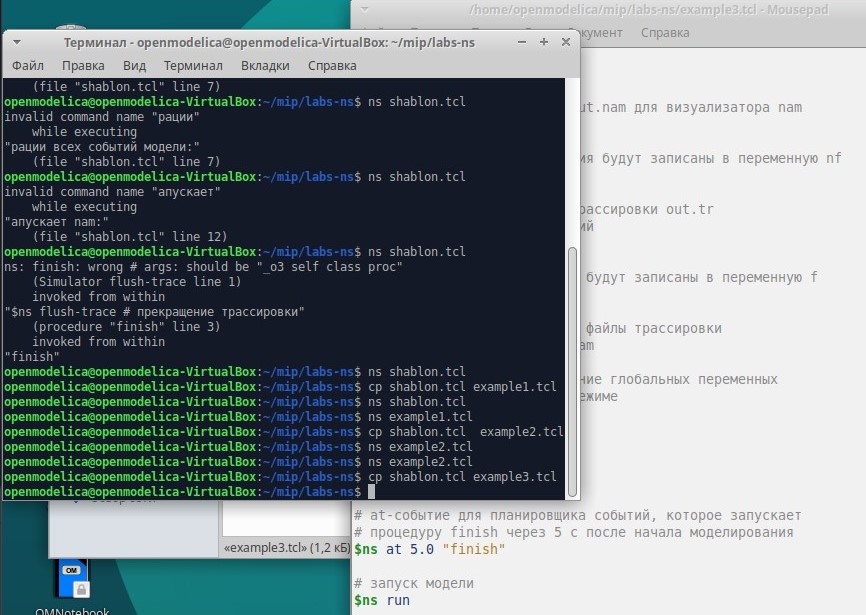


Рис. 15: Копирование шаблона в файл example3.tcl

Откроем example3.tcl на редактирование. Опишем топологию моделируемой сети. Далее соединим узлы так, чтобы создать круговую топологию (рис. 16).



Рис. 16: Создание круговой топологии

Каждый узел, за исключением последнего, соединяется со следующим, последний соединяется с первым. Для этого в цикле используем оператор %, означающий остаток от деления нацело. Зададим передачу данных от узла n(0) к узлу n(3). Добавим команду разрыва соединения между узлами n(1) и n(2) на время в одну секунду, а также время начала и окончания передачи данных: (рис. 17).



Рис. 17: Задание узлов, между которыми будут передаваться данные

Запустим анимацию созданный кольцевой сети. Данные передаются по кратчайшему маршруту от узла n(0) к узлу n(3), через узлы n(1) и n(2) (рис. 18).

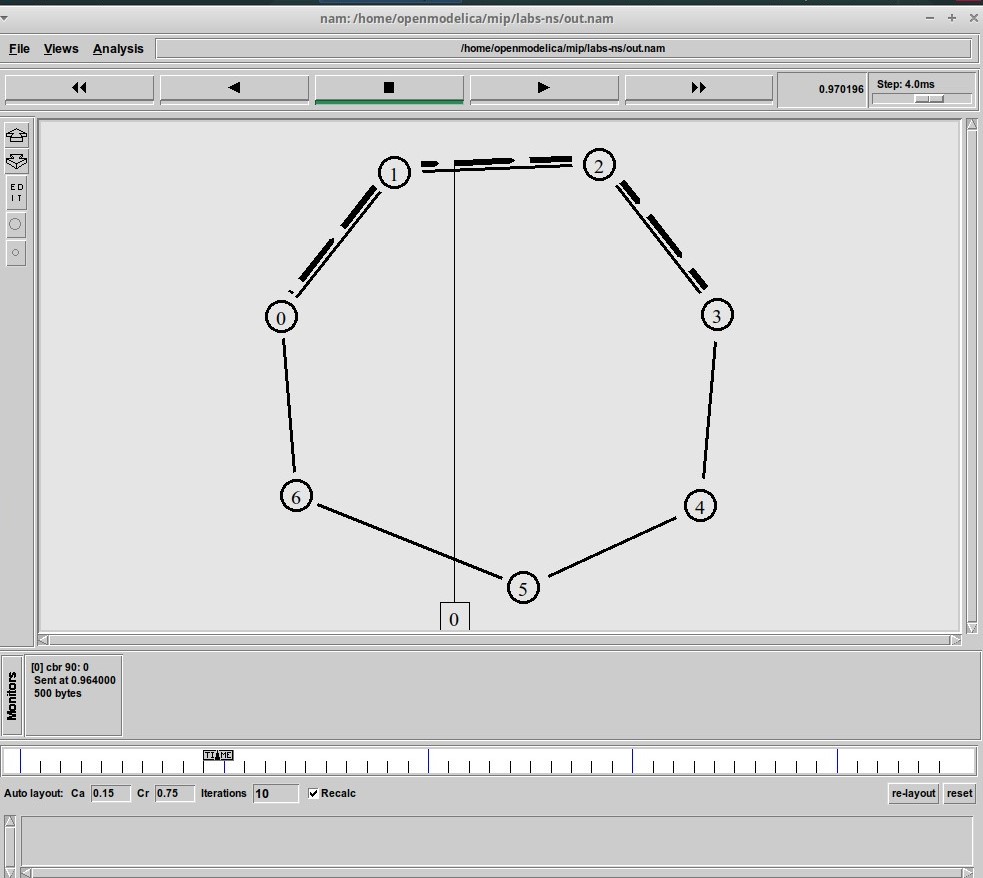


Рис. 18: Анимация круговой топологии

В заданный промежуток времени происходит разрыв цепи. Некоторые данные теряются (рис. 19).

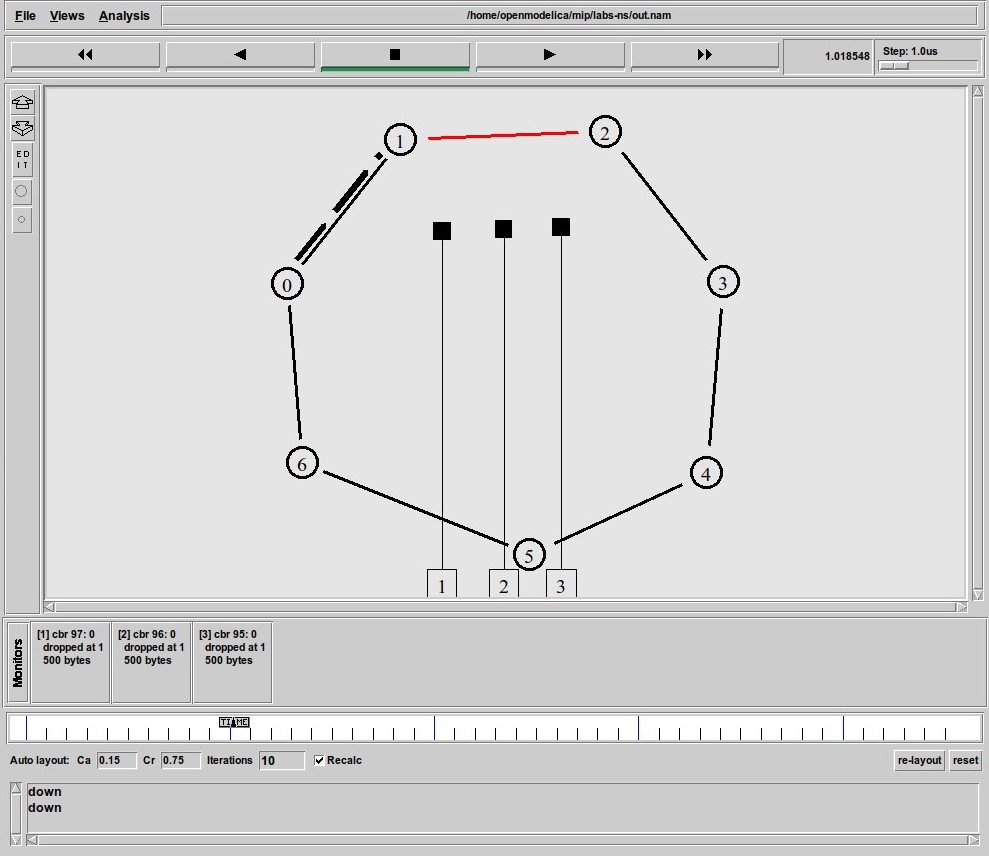


Рис. 19: Анимация круговой топологии с разрывом

Добавим в начало скрипта после команды создания объекта Simulator строчку $ns rtproto DV (рис. 20).

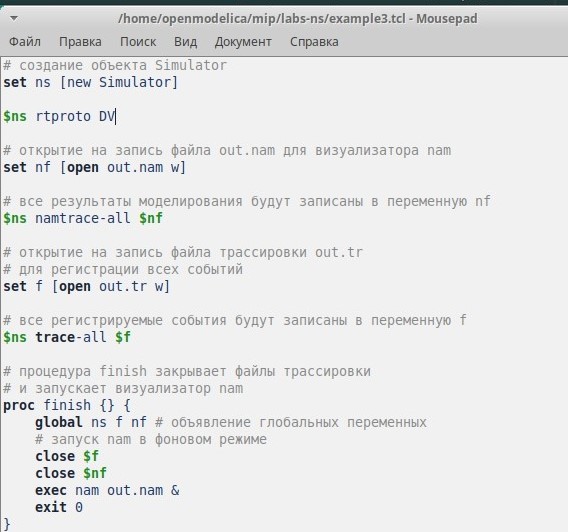


Рис. 20: Добавление строчки для выбора оптимального пути после разрыва

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) (рис. 21).

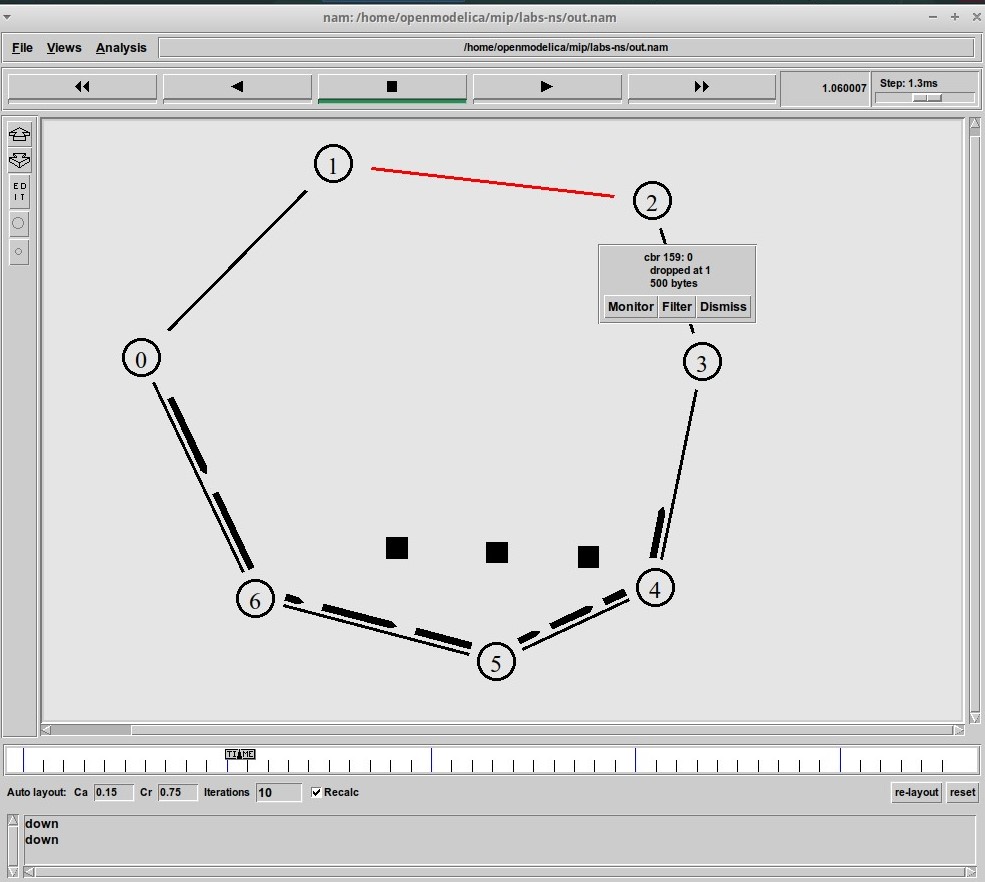


Рис. 21: Анимация выбора оптимального пути после разрыва

1. Выполнение упражнения

**Постановка задачи** Внесите следующие изменения в реализацию примера с кольцевой топологией сети: - топология сети должна соответствовать представленной на рисунке в тексте описания ЛР. - 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.

Скопируем шаблон сценария в новый файл example4.tcl. Откроем файл на редактирование. По условию упражнения создадим 5 узлов. Соединим из кольцевой связью. Соединим узлы n5 и n1. Везде будем использовать дуплексную линию связи с полосой пропускания 1 Мб/с и задержкой 10 мс, очередью с обслуживанием типа DropTail. Создадим агента TCP и свяжем его с узлом n(0). Создадим приложение FTP и соединим его с агентом TCP. Создадим агента получателя, зададим время работы FTP и время разрыва (рис. 22).

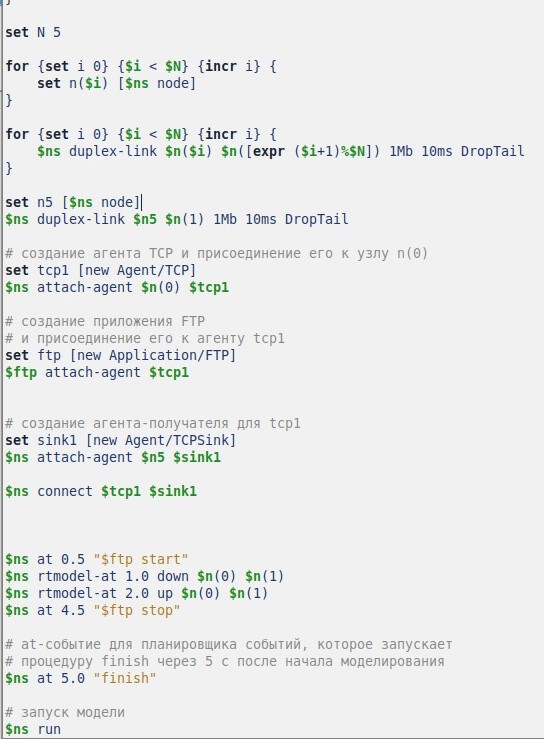


Рис. 22: Скрипт для реализации упражнения

Запустим анимацию созданной цепи. В момент времени 0.5 начинается передача пакетов от узла n(0) к узлу n5. Передаются данные по протоколу TCP, в обратном направлении отправляются Ack сигнал, который означет, что данные успешно получены (рис. 23).

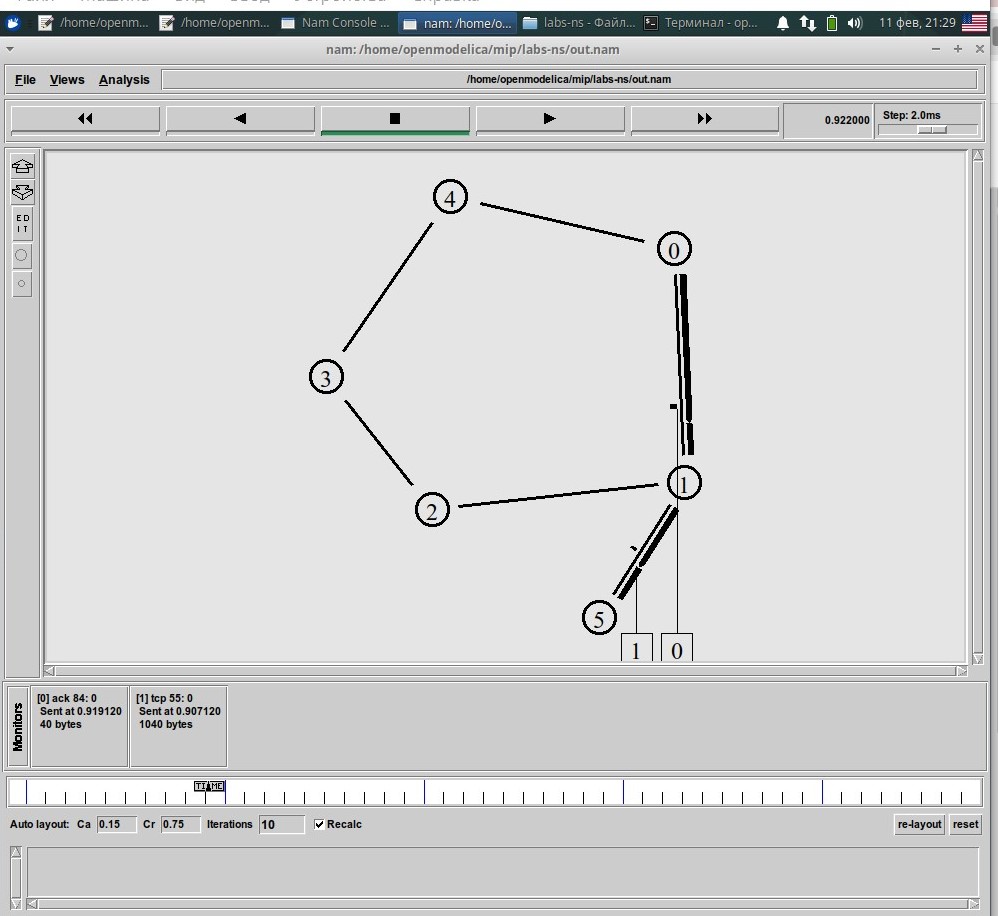


Рис. 23: Анимация скрипта, написанного в рамках упражения

Далее с 1 до 2 секунды происходит разрыв соединения между узлами n(0) и n(1). Теперь данные отправляются по ближайшему возможному маршруту через узлы n(4), n(3) и n(2) (рис. 24).

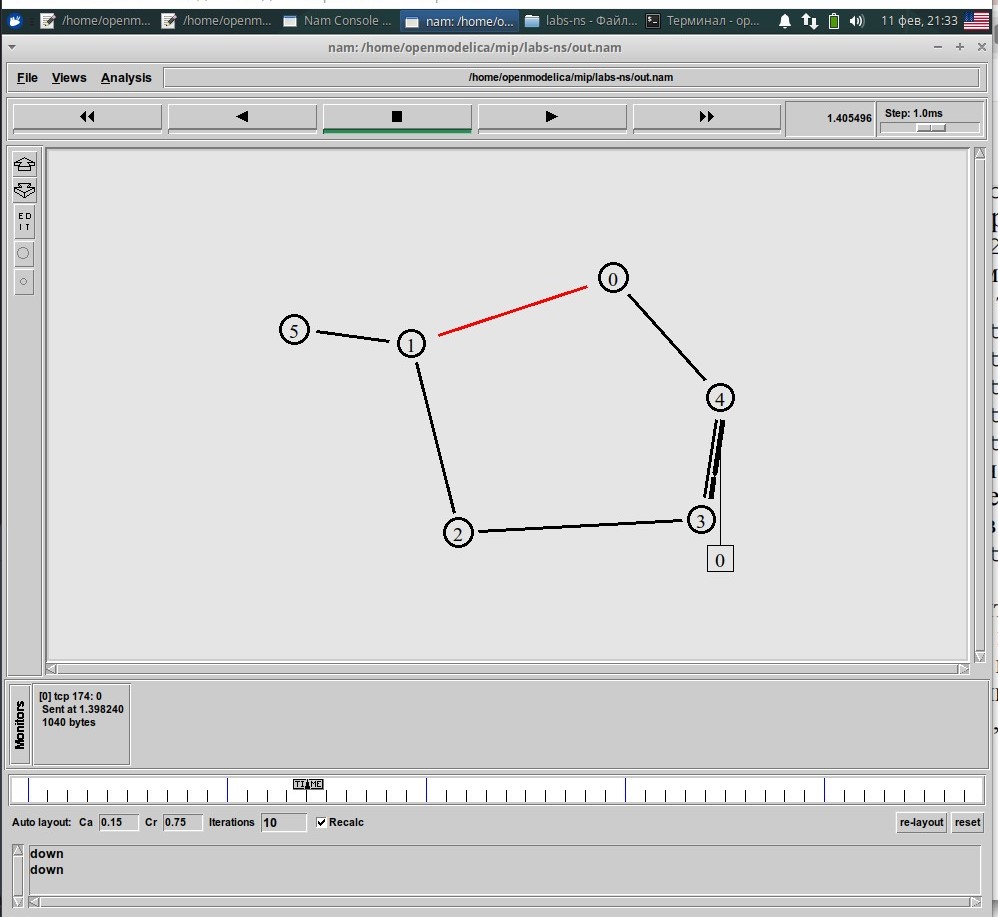


Рис. 24: Анимация скрипта, написанного в рамках упражения

# 4 Выводы

В ходе выполнения лабораторной работы я приобрела навыки моделирования сетей передачи данных с помощью средства имитационного моделирования NS-2, а также провела анализ полученных результатов моделирования.