Томский университет систем управления и радиоэлектроники

С.И. Богомолов

Сети ЭВМ и телекоммуникации

Лабораторный практикум

2012

Министерство образования и науки РФ
Томский университет систем управления и радиоэлектроники

Радиотехнический факультет
Кафедра телекоммуникаций и основ радиотехники

«Утверждаю»
Зав. кафедрой ТОР
_____ Е.П. Ворошилин
_____ 2012 г.

Лабораторный практикум

по дисциплине
«Сети ЭВМ и телекоммуникации»

для студентов направления подготовки 210700.68
«Инфокоммуникационные технологии и системы связи»

Лабораторный практикум составил:
к.т.н., доцент С.И. Богомолов

Томск - 2012 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

ОБЩИЕ ПОЛОЖЕНИЯ

Лабораторный практикум по курсу «Сети ЭВМ и телекоммуникации» имеет целью закрепить и расширить теоретические знания студентов при изучении принципов построения и функционирования устройств и систем коммуникации сетей ЭВМ, ознакомить их с методиками исследования основных компонентов и процессов систем связи, в том числе, и с использованием моделирования устройств и систем связи.

Данный цикл лабораторного практикума, предназначенного для студентов направления подготовки 210700.68 «Инфокоммуникационные технологии и системы связи», содержит описание следующих работ:

Исследование основных компонентов сетевого имитатора NS2

Моделирование сетей ЭВМ с помощью сетевого имитатора NS2

Исследование характеристик протокола TCP с помощью сетевого имитатора NS2

Работы основного цикла ориентированы на исследование отдельных вопросов функционирования вычислительных сетей с помощью сетевого тренажера NS2.

В качестве ознакомления с языком программирования Tcl-OTcl в приложение приведены сведения по этому языку, в том числе. предложены упражнения, предназначенные для получения первичных навыков работы с командным интерпретатором сетевого симулятора NS2.

ПРАВИЛА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Перед выполнением работы студенты на этапе предварительной подготовки:

а) изучают соответствующие разделы теоретического курса;

б) знакомятся с описанием лабораторной работы и подготавливают шаблон отчета по лабораторной работе;

в) выполняют необходимые предварительные расчёты, изложенные в разделе «Предварительная подготовка».

К выполнению лабораторных работ допускается только сту-

денты, выполнившие требования предыдущего раздела, и подтвердившие свою подготовленность при собеседовании с преподавателем.

Лабораторные работы выполняются индивидуально фронтальным методом. При выполнении работ рекомендуется следовать методическим указаниям. Разрешается проведение дополнительных исследований (не в ущерб основному заданию).

В процессе выполнения работы составляется предварительный отчёт, который должен содержать таблицы и графики полученных экспериментально зависимостей.

Если при составлении предварительного отчёта выявится недостаточность или сомнительность полученных данных, то необходимо экспериментально получить недостающие данные и произвести проверку сомнительных результатов.

Работа считается выполненной после утверждения предварительного отчёта преподавателем.

Студенты, не выполнившие работу в часы занятий, выполняют её в специально отведённое время.

СОДЕРЖАНИЕ И ОФОРМЛЕНИЕ ОТЧЕТА

Отчёт по выполненной работе составляется индивидуально каждым студентом.

Отчёт оформляется на листах формата А4 (достаточно электронной копии в редакторе OpenOffice.org Writer). У осей графиков должна быть проставлены числовые значения и единицы размерности.

Отчёт должен содержать:

а) цель исследования;

б) результаты расчётов, полученных на этапе предварительной подготовки;

в) структурные схемы устройств и систем, характеристик которых исследуются в данной работе;

г) структурные схемы измерительных установок для исследования характеристик устройств и систем связи;

д) результаты исследований в виде таблиц, графиков и изображений, получаемых на экранах измерительных приборов;

е) выводы, полученные на основании анализа расчётных и экспериментальных данных.

ЗАЩИТА РАБОТЫ

Лабораторная работа считается выполненной после защиты результатов работы.

При защите результатов работы студент должен представить оформленный отчёт, сохраненные результаты компьютерного эксперимента и продемонстрировать свои знания в следующих разделах работы:

а) структурные схемы исследуемых устройств и систем и основы их функционирования;

б) структурные схемы измерительных установок для исследования характеристик устройств и систем связи и характеристики, которые могут быть получены с помощью данных установок;

в) результаты расчетов предварительной подготовки;

г) теоретическое обоснование полученных экспериментальных зависимостей.

ИССЛЕДОВАНИЕ ОСНОВНЫХ КОМПОНЕНТОВ СЕТЕВОГО ИМТАТОРА NS2

Целью работы «Исследования основных компонентов сетевого имитатора NS2» является знакомство с программным продуктом для имитационного моделирования процессов, происходящих в сетях связи, а также получения первичных навыков работы с сетевым имитатором NS2.

КРАТКИЕ СВЕДЕНИЯ О СЕТЕВОМ ИМИТАТОРЕ NS2

NS2 является программным продуктом, позволяющим осуществить имитационное моделирование сетей связи, разработанный под руководством ряда научных организаций и центров в рамках проекта VINT (Virtual InterNetwork Testbed), организованным DARPA (Defense Research Project Agency) [http://www.isi.edu/nsnam/vint/]. За основу программной реализации выбран пакет network simulator, разработанный в Калифорнийском университете.

NS2 является объектно-ориентированным программным обеспечением, ядро которого реализовано на языке C++. В качестве интерпретатора используется язык скриптов (сценариев) OTcl (Object oriented Tool Command Language). C++ является системным языком и NS2 полностью поддерживает иерархию классов C++.

Использование двух языков программирования объясняется следующими причинами. С одной стороны, для детального моделирования протоколов необходимо использовать системный язык программирования, обеспечивающий высокую скорость выполнения и способный манипулировать достаточно большими объемами данных. С другой стороны для удобства пользователя и быстроты реализации и модификации различных сценариев моделирования привлекательнее использовать язык программирования более высокого уровня абстракции. Такой подход является компромиссом между удобством использования и скоростью.

Системный язык C++ позволяет обеспечит высокую производительность, работу с пакетами потока на низком уровне абстракции модели, модификацию ядра NS2 с целью поддержки но-

вых функций и протоколов.

В качестве языка программирования высокого уровня абстракции используется язык скриптов OTcl, позволяющий обеспечить ряд положительных свойств, присущих языку OTcl: простоту синтаксиса, простоту построения сценария моделирования, возможность соединения воедино блоков, выполненных на системных языках программирования и простую манипуляцию ими.

В NS2 на уровне ядра реализованы большинство известных протоколов сетей связи, например MPLS, IPv6, OSPF, RSVP, протоколы беспроводной связи, web caching и многие другие. Также реализовано целое семейство дисциплин обслуживания очередей: RED, WFQ, CBQ, SFQ и т.д. В рамках NS2 возможна реализация собственного протокола.

NS2 содержит средства анимации результатов моделирования NAM (Network Animator), которое предоставляет графическое воспроизведение результатов проведенного эксперимента: отображение топологии сети, анимация пакетов, узлов, очередей и различные возможности анализа данных. В качестве входных данных для NAM используются файлы, записанные в процессе функционирования NS2, т.е. моделирования сети связи.

Более подробные сведения о сетевом симуляторе NS2 изложены в документе about_ns2_rus.pdf, представленом на сервере S ЛВС кафедры ТОР [S:\БогомоловСИ\NS\Lab1\Metod\], а также на сайтах: http://www.isi.edu/, http://wwwns2.chat.ru/, в частности, в документах http://www.isi.edu/nsnam/ns/ns-documentation.html, http://www.isi.edu/nsnam/ns/tutorial/nsindex.html.

ОСНОВЫ РАБОТЫ С СЕТЕВЫМ ИМИТАТОРОМ NS2

Основное содержание данного раздела использовано из учебных материалов VINT group: NS Tutorial, доступных по адресу http://www.isi.edu/nsnam/ns/tutorial/nsindex.html.

Подготовка шаблона сценария

В данном разделе рассматриваются вопросы разработки Tcl сценария для имитатора NS, моделирующего простую топологию сети: как создавать узлы и линии связи между ними, как посылать данные от одного узла к другому, как контролировать очередь и как запускать аниматор NAM по сценарию моделирования для визуализации его результатов.

Во-первых, создается шаблон, который можно использовать для всех Tcl скриптов. Tcl сценарий можно писать в любом текстовом редакторе, и предлагается назвать этот первый пример 'example.tcl'.

Прежде всего, создается объект имитатор командой

set ns [new Simulator].

Затем открывается файл для записи, который будет использовать аниматор NAM для распечатки данных

set nf [open out.nam w]

$ns namtrace-all $nf.

Первая строка открывает файл 'out.nam' для записи и дает обращается с этим файлом как с переменной 'nf'. На второй строке сообщается, что созданный выше объект имитатор запишет все данные моделирования, предназначенные для аниматора NAM, в этот файл.

Следующий шаг добавляет процедуру 'finish' которая закрывает файл трассировки и запускает NAM:

```
proc finish {} {

    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}.
```

Записанные выше коды будут более понятными после того, как станет видно, что они делают.

Следующая строка передает имитатору: выполнить процедуру 'finish' после 5 секунд времени моделирования

$ns at 5.0 "finish".

Командой 'at' имитатор NS обеспечивает простой способ планирования событий.

Наконец, последняя строка запускает моделирование

$ns run.

Этот файл теперь можно сохранить и попробовать запустить его, набрав в терминале:

'ns example.tcl'.

Так как до сих пор не были определены никакие объекты (узлы, связи, и т.д.) или события, может быть получено сообщение об ошибке типа 'nam: empty trace file out.nam' (либо не будет заметно никакой реакции).

Пересылка данных между узлами

Далее рассматривается создание очень простой топологии из двух узлов, соединенных линией связи.

Следующие две строки определяют два узла:

```
set n0 [$ns node]
set n1 [$ns node].
```

Новый объект узел создается командой '$ns node'. Приведенный выше код создает два узла и присваиваем им для управления имена 'n0' и 'n1'. (Примечание: Код данной секции должен быть вставлен перед строкой '$ns run', или даже лучше, перед строкой '$ns at 5.0 "finish"').

Следующая строка соединяет эти два узла.

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail.
```

Эта строка сообщает объекту имитатору: соединить узлы n0 и n1 двухсторонней связью с полосой пропускания 1Megabit, задержкой 10ms и очередью типа DropTail.

Затем можно сохранить файл (example1.tcl) и запустить на выполнение сценарий

'ns example1.tcl'.

Аниматор NAM стартует автоматически, и должен появиться выходной продукт в виде схемы простейшей сети.

Этот пример позволяет видеть только топологию сети, в которой ничего в ней не происходит, поэтому очередным этапом будет пересылка данных от узла n0 к узлу n1. В имитаторе NS данные всегда пересылаются от одного агента другому. Поэтому следующий шаг должен создать объект агент, который посылает данные от узла n0, и второй объект агент, который получает данные на узле n1.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
```

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0

Эти строки создают UDP (User Datagram Protocol) агента и присоединяют его к узлу n0, затем подключают генератор CBR (Constant Bit Rate) трафика к UDP агенту. Модуль CBR устанавливает постоянную битовую скорость генерирования информации. Строки 7 и 8 не требуют пояснений: Размер пакета устанавливают по 500 байтов, и пакеты будут посылаться каждые 0.005 секунды (то есть 200 пакетов в секунду). Соответствующие параметры для каждого агента определены в руководстве к симулятору NS < http: // www.isi.edu/nsnam/ns/doc/index.html >.

Следующие строки создают нулевого агента, который действует, как приемник трафика, и подключают его к узлу n1.

set null0 [new Agent/Null]

$ns attach-agent $n1 $null0

Затем эти два агента быть связаны друг с другом.

$ns connect $udp0 $null0

И теперь следует сообщить CBR агенту, когда начать посылать данные и когда прекратить посылать. (Примечание: Вероятно лучше поместить следующие строки как раз перед строкой '$ns at 5.0 "finish"').

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

Этот код также не требует пояснений.

Теперь можно сохранить файл (example1a.tcl) и снова начать моделирование. После щелчка на кнопке 'play' в окне аниматора NAM можно будет видеть, что после 0.5 секунд моделирования, узел n0 начнет посылать пакеты данных к узлу n1. С помощью ползунка 'Step' можно изменить темп показа NAM.

Далее можно начать выполнять некоторые эксперименты с NAM и Tcl сценариями. Можно щелкать на любом пакете в окне NAM, чтобы контролировать его. Также можно щелкать непосредственно на линии связи, чтобы получить некоторые графики со статистикой. Рекомендуется также изменить параметры размера пакета 'packetsize' и интервала посылок 'interval' в Tcl сценарии. Полную версию рассмотренного примера можно посмотреть по адресу

<http://www.isi.edu/nsnam/ns/tutorial/examples/example1b.tcl>.

Разработка топологии сети

В этом разделе топология сети задается четырьмя узлами, в которых один узел действует как маршрутизатор, продвигающий дальше данные, которые два узла пересылают четвертому узлу. Поясняется способ, как различить потоки данных от двух узлов от друг друга, и показано, как можно контролировать очередь, чтобы наблюдать, насколько она становится заполненной, и сколько пакетов будет отброшено.

Как всегда, первый шаг должен определить топологию. Следует создать файл 'example2.tcl', используя как шаблон программу предыдущего раздела 'example.tcl'.

Как уже указывалось, программа всегда остается похожей. Всегда следует создавать объект имитатор, моделирование всегда должно начинаться с той же самой команды, и если предполагается управлять аниматором NAM автоматически, нужно всегда открыть файл трассировки, инициализировать его, и определять процедуру, которая закрывает его и запускает NAM.

Для создания четырех узлов в программу следует вставить следующие строки

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

Следующая часть Tcl кода создает три двунаправленных связи между узлами.

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n3 $n2 1Mb 10ms DropTail
```

Теперь программу следует сохранить ('example2.tcl) и запустить сценарий на выполнение. Вероятно, можно заметить, что в окне NAM топология выглядит несколько неуклюже. Для того, чтобы схема выглядела более подходяще, возможно использование кнопки 're-layout' (перепланировка), но было бы лучше иметь немного больше контроля над формированием топологии. Для этого нужно добавить следующие три строки к Tcl сценарию, и стартовать его снова ('example2a.tcl).

```
$ns duplex-link-op $n0 $n2 orient right-down
```

```
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right.
```

Теперь, после просмотра топологии в окне NAM, вероятно понятно, что делает эта программа,.

Нужно заметить, что автоматическая планировка компонентов сети в окне NAM отменяется, так как теперь их размещение производится вручную. Выбор для ориентации связей возможен вправо, влево, вверх, вниз и в комбинации этих ориентаций. Экспериментировать с этими назначениями можно позже, а пока топологию следует оставить как получилось.

Далее создаются два UDP агента с CBR источниками трафика, которых приписывают к узлам n0 и n1. Затем создается агент с отсутствием информации (нуль-агент), приписанный к узлу n3.

```
#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
```

Два CBR агенты должны быть связаны с нуль-агентом.

```
$ns connect $udp0 $null0
$ns connect $udp1 $null0
```

Планирование событий и маркирование потоков

Пусть первый CBR агент, начнет посылать в момент времени 0.5 секунды и остановится в 4.5 секунды, в то время как второй CBR агент стартует в момент времени 1.0 секунда и остановится в 4.0 секунды.

```
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
```

Теперь когда запускается скрипт ' ns example2b.tcl ', заметно, что на линиях связях от n0 до n2 и n1 к n2 имеется большее количество трафика, чем может перенести линия связи от n2 до n3. Это подтверждает и простое вычисление: Мы посылаем 200 пачек в секунду на каждой из первых двух связей, и размер пачки - 500 байтов. Это соответствует полосе пропускания 0.8 мегабита в секунду для линий связей от n0 до n2 и от n1 до n2. Это приводит к полной полосе пропускания 1.6Mb/s, но линия связи между n2 и n3 имеет пропускную способность только 1Mb/s, поэтому очевидно, что некоторые пакеты отбрасываются. Но которые? Оба потока черны, поэтому единственный способ выяснить то, что случается с пакетами, это контролировать их в NAM, щелкая на них. В следующих разделах будет показано, как различить между разными потоками и как наблюдать то, что фактически происходит в очереди на линии связи от n2 до n3.

Добавляются следующие две строки к описаниям CBR агентов.

```
$udp0 set class_ 1
$udp1 set class_ 2
```

Параметр 'fid_' устанавливается после идентификатора потока 'flow id'.

Затем добавляется следующую часть программы к Tcl скрипту, предпочтительно вначале, после того, как был создан объект моделирования, так как она является частью набора заданных значений имитатора.

```
$ns color 1 Blue
$ns color 2 Red
```

Эта программа позволяет устанавливать различные цвета для каждого имени потока id.

Теперь можно снова запускать скрипт ('example2c.tcl), и один поток должен быть синего цвета, в то время как другой красного. Наблюдая соединение от узла n2 к n3 некоторое время, можно заметить, что после некоторого времени распределение между синими и красными пакетами оказывается уже не слишком справедливым. В следующем разделе будет показано, как можно наблюдать эти связи внутри очереди, чтобы выяснить то, что происходит там.

Для того, чтобы контролировать очередь на линии связи от n2 до n3, к программе следует добавить следующую строку

$ns duplex-link-op $n2 $n3 queuePos 0.5

Затем можно снова запустить NS ('example2d.tcl), и будут видны обновленные результаты моделирования.

Теперь можно наблюдать пакеты в очереди, а через некоторое время даже можно видеть, как пакеты падают. Возможно, распределение отброшенных пакетов будет неравномерным, и действительно, нельзя ожидать достаточной справедливости от простой DropTail очереди. Поэтому следует пробовать улучшить организацию очереди, используя SFQ (стохастическая справедливая организация очереди) очередь для связи от n2 до n3. Для этого заменяют описание соединения для линии связи между узлами n2 и n3 на следующую строку ('example2e.tcl)

$ns duplex-link $n3 $n2 1Mb 10ms SFQ.

Теперь организация очереди должна быть более справедливой, и количество отброшенных синих и красных пакетов следует ожидать примерно одинаковым.

ПРЕДВАРИТЕЛЬНАЯ ПОДГОТОВКА

Ознакомиться с описанием сетевого имитатора NS2 и его основных компонентов по материалам данного практикума и рекомендуемых в нем литературных источников.

Изучить раздел «Основы работы с сетевым имитатором NS2» методических указаний по данной лабораторной работе и подготовить необходимые Tcl скрипты для работы с NS2, рассмотренные в данном разделе.

Доработать сценарий example2a.tcl таким образом, чтобы в центре сети был размещен узел, номер которого отмечен знаком “+” в таблице1 (по вариантам). Там же указана ориентация ос-

тальных узлов (по направлению часовой стрелки) относительно центрального узла. Параметры линий связи (frq –полоса пропускания, del – задержка передачи) устанавливать одинаковыми для всей сети.

Таблица 1

	1	2	3	4	5	6	7	8	9	10
n0	+	1,5	3	4,5	+	7,5	9	12	+	3
n1	12	+	6	9	6	+	12	3	1,5	+
n2	3	4,5	+	12	9	12	+	7,5	6	7,5
n3	7,5	9	10,5	+	1,5	3	4,5	+	9	12
Null	n3	n0	n1	n2	n2	n3	n0	n1	n1	n2
Frq	0,5	0,7	0,6	1,0	0,9	1,1	1,2	0,9	0,8	0,7
del	20	15	18	10	12	9	8	16	15	16

Полагать в дальнейшем, что два из трех периферийных узлов работают на передачу пакетов, третий узел, обозначенный в строке «Null» таблицы работает на прием, а центральный узел только продвигает далее полученные пакеты, не добавляя нагрузки.

Результатами предварительной подготовки по данной работе являются файлы Tcl скриптов, подготовленные для моделирования сетей с помощью имитатора NS2.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. С какой целью используется динамическое моделирование систем? Какие задачи оно решает?
2. Дать сравнительную характеристику программных продуктов, предназначенных для моделирования телекоммуникационных сетей.
3. Что представляет собой проект NS2 VINT?
4. Раскрыть архитектуру имитатора NS2.
5. С какой целью в NS2 используется два языка программирования?
6. Как в NS2 отражены реальные характеристики сетевых протоколов, порядок обслуживания очередей?
7. Какие виды ошибок могут быть смоделированы в NS2?
8. Какие средства используются для визуализации в NS2?
9. Какие основные компоненты содержит шаблон Tcl сценария?

10. Пояснить элементы Tcl сценариев, необходимые при создании узлов и связей между ними.
11. Перечислить основные параметры линии связи между узлами и пояснить, как они задаются в Tcl сценарии.
12. Каким образом можно вручную размещать компоненты на схеме сети? Привести примеры.
13. Что такое агенты и какие функции они выполняют?
14. Перечислить известные агенты и описать их основные характеристики.
15. Какие параметры агентов учитываются при моделировании сети и каким образом?
16. Что такое CBR генераторы и как они участвуют в моделировании?
17. Какие параметры CBR генераторов могут быть заданы при моделировании?
18. Что такое планирование событий и как оно реализуется в NS2?
19. Каким образом можно контролировать потоки данных?
20. Для чего и как выполняется маркировка данных?
21. Какие виды организации очереди используются в NS2? Привести примеры.
22. Какие параметры пакета могут быть определены в результате эксперимента?
23. Какими средствами в NS2 отображаются результаты эксперимента? Пояснить.

ЛАБОРАТОРНОЕ ЗАДАНИЕ

1. Ознакомиться с описанием сетевого имитатора NS2 и его основных компонентов по материалам данного практикума и рекомендуемых в нем литературных источников (каталог \Lab2\Metod\). Создать папку с названием Lab_2 на сервере X. Промежуточные и окончательные результаты данной работы сохранять в этом каталоге.

2. Файлы, подготовленные в результате предварительной подготовки, скопировать в созданную папку Lab_2.

3. Открыть окно «Терминал». С помощью команды «Изменить каталог» (cd *path*), где path – путь доступа к каталогу Lab_2, перейти в свой рабочий каталог.

4. Используя файлы, подготовленные в результате предварительной подготовки, выполнить все операции, описные в разделе «Основы работы с сетевым имитатором NS2» методических указаний по данной лабораторной работе.

5. Наблюдать реакцию системы при нажатии на клавишу «relayout», клавиш направления и скорости анимации результатов моделирования, ползунка «step».

6. Остановить моделирование по сценария example1a.tcl в фазе пересылки данных (на интервале моделирования 0,5…4,5 секунды). Справочную информацию о пакете можно вызвать щелчком левой кнопки мыши на образе пакета. Зафиксировать сведения о нескольких соседних пакетах. Подобным же образом ознакомиться с информацией об узлах и линиях связи.

7. Запустить сценарий example2a.tcl, доработанный на этапе предварительной подготовки. Убедиться в соответствии полученного решения исходному заданию. В случае расхождения – доработать программу.

8. Ввести в сценарий моделирования планирование событий. Для этого дополнить доработанный скрипт строками программы по аналогии с файлом example2b.tcl. Параметры CBR источников 1 и 2 (размер пакета ps в байтах, период поступления пакетов in в миллисекундах, цвета маркировки) выбирать в соответствии с таблицей 2. Цвета в таблице закодированы цифрами: 1 – blue, 2 – brown, 3 – green, 4 – red, 5 – yellow.

Таблица 2

	1	2	3	4	5	6	7	8	9	10
ps1	200	400	150	500	250	300	350	500	250	400
ps2	150	500	300	200	500	600	700	200	300	250
in1	4	6	3	5	3	4	4	8	5	7
in2	3	8	6	2	6	8	7	3	6	5
colo	1 3	1 4	1 5	2 3	2 4	2 5	3 4	3 5	4 5	1 2

9. Запустить в NS разработанный в п.8 сценарий. Проверить параметры пакетов и убедиться, что результаты моделирования соответствуют исходному заданию. В случае расхождения – доработать программу.

10. Для контроля пакетов в очереди ввести в программу последнюю строку сценария example2d.tcl (с учетом собственной

схемы сети). Убедиться в результативности изменений. Повторить тоже самое при добавлении в программу последней строки сценария example2e.tcl.

11. Оценить влияние параметров линии связи на величину очереди. Для этого зафиксировать несколько значений полосы пропускания и задержки передачи, в пределах которых наблюдается изменение характера очереди.

12. Путем изменения параметров линии связи экспериментально определить границу полосы пропускания линии связи, при которой начинает формироваться очередь.

13. В отчет следует внести файлы программ, разработанные на этапе предварительной подготовки, а также в результате экспериментальных исследований, включая протоколы работы NS2.

МОДЕЛИРОВАНИЕ СЕТЕЙ ЭВМ С ПОМОЩЬЮ СЕТЕВОГО ИМИТАТОРА NS2

Целью работы «Моделирование сетей ЭВМ с помощью сетевого имитатора NS2» является исследование характеристик сетей ЭВМ путем компьютерного моделирования, направленного на углубление знаний и расширение сведений об основных характеристиках сетей, а также закрепление навыков работы с сетевым имитатором NS2.

КРАТКИЕ СВЕДЕНИЯ О МОДЕЛИРОВАНИИ СЕТЕЙ ЭВМ С ПОМОЩЬЮ СЕТЕВОГО ИМИТАТОРА NS2

Основное содержание данной раздела использовано из учебных материалов VINT group: Jae Chung and Mark Claypool. NS by Example, Technical Report WPI-CS-TR-99-25, Computer Science Department, Worcester Polytechnic Institute, September 1999, а также пособие NS Tutorial, доступное по адресу http://www.isi.edu/nsnam/ns/tutorial/nsindex.html.

Разработка топологии сети

В этом разделе продемонстрирована простая программа NS моделирования с объяснениями, что выполняет каждая строка. В качестве примера рассмотрен Otcl скрипт, который создает простую сетевую конфигурацию и выполняет сценарий моделирования для сети, показанной на рисунке 1. Для выполнения этого моделирования следует поместить файл ns-simple.tcl, размещенный в папке S:\БогомоловСИ\NS\Lab3\Metod\, в свой каталог и запустить его на приглашение командного процессора (ns ns-simple.tcl). (Оригинал вышеупомянутого документа доступен по адресу < http: // nile.wpi.edu/NS/Example/ns-simple.tcl >).

Моделируемая сеть состоит из 4 узлов (n0, n1, n2, n3) и показана на рисунке 1. Дуплексные линии связи между узлами n0 и n2, и n1 и n2 имеют пропускную способность 2 Mbps и время задержки 10 ms. Дуплексная линия связи между узлами n2 и n3 имеет пропускную способность 1,7 Mbps и время задержки 20 ms. Каждый узел использует очередь типа DropTail с максимальным размером 10. К узлу n0 подключен "tcp" агент, у которого установлено соединение с "tcp" агентом приемника ("sink"), подключенного к узлу n3. По умолчанию, максимальный размер па-

кета, который может производить "tcp" агент – 1 KByte. "tcp" агент приемника ("sink") генерирует и посылает ACK пакеты отправителю ("tcp" агенту) и освобождает принятые пакеты. К узлу n1 подключен "udp" агент, который, связан с "null" агентом, подключенным к узлу n3. Назначение "null" агента – только освобождать принятые пакеты. К "tcp" и "udp" агентам прикреплены источники трафика, соответственно, "ftp" и "cbr". Источник "cbr" сконфигурирован так, чтобы генерировать пакеты размером 1 KByte со скоростью 1 Mbps. Источник "cbr" начинает функционирование в 0,1 секунды и оканчивает в 4,5 секунды, а "ftp" начинает в 1,0 секунды и оканчивает в 4,0 секунды.

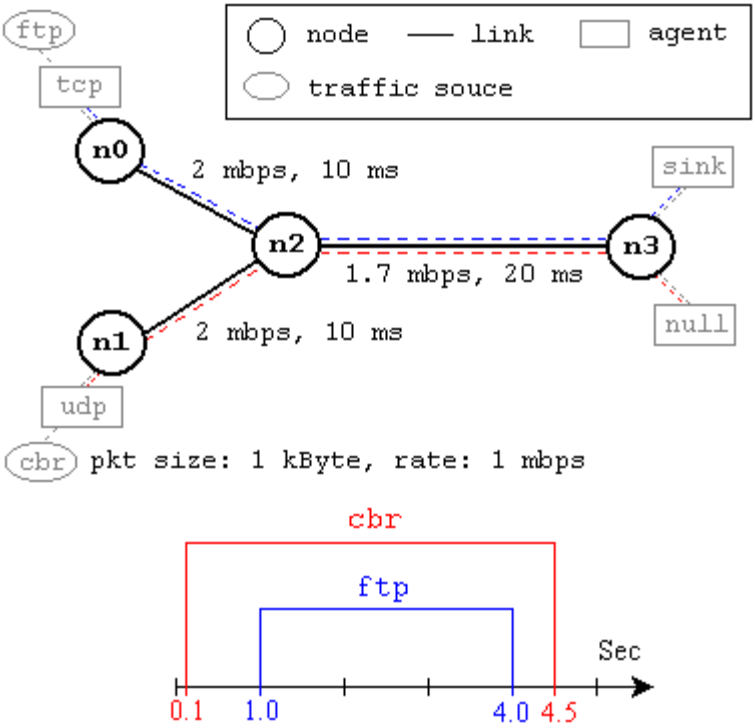

Рис.1. Топология сети и сценарий моделирования

Далее следует объяснение приведенного выше сценария. В общем, NS сценарий начинается с создания экземпляра объекта

моделирования.

- set *ns* [new Simulator]: создает экземпляр объекта NS Simulator и определяет его переменной *ns* (здесь и далее курсив используется для переменных и значений). Эта строка делает следующее:
  - Инициализирует формат пакета (это пока игнорировать).
  - Создает планировщика (значение по умолчанию – календарный планировщик).
  - Выбирает формат адреса по умолчанию (это пока игнорировать)

Объект Simulator содержит методы, которые выполняют следующее:

- Создание составных объектов типа узлов и связей (описано позже)
- Подключение созданных объектов сетевых компонентов (исключая. присоединение агентов)
- Установка параметров сетевых компонентов (главным образом для составных объектов)
- Создание соединения между агентами (исключая. создание соединения между "tcp" и "sink")
- Определение опций отображения NAM
- И т.д.

Большинство методов существуют для установки моделирования и планирования, однако некоторые из них предназначены для отображения NAM. Реализации методов объекта Simulator размещены в файле "opt/ns/2.33/ns-2.33/tcl/lib/ns-lib.tcl".

- *$ns* color *fid color*: для установки цвета пакетов для потока, указанного идентификатором потока (fid). Этот метод объекта Simulator - для отображения NAM и не оказывает никакого эффекта на фактическое моделирование.
- *$ns* namtrace-all *file-descriptor*: Этот метод указывает симулятору делать запись трассировки моделирования в формате ввода NAM. Он также дает имя файлу, в который трассировка будет записана позже командой *$ns* flush-trace. Точно так же метод trace-all используется для записи трассировки моделирования в общем формате.
- proc *finish* {}: после того, как окончено это моделирование,

вызывается командой *$ns* at 5.0 "*finish*". В этой функции определены процессы по окончанию моделирования.

- set *n0* [*$ns* node]: Метод node создает узел. Узел в NS – это составной объект, созданный классификаторами адреса и порта (описанные позже). Пользователи могут создавать узлы, отдельно создавая объекты классификатора порт и адрес и затем соединять их вместе. Однако, этот метод объекта Simulator делает работу более легкой. Подробнее рассмотреть как создан узел можно в файлах: "opt/ns/2.33/ns-2.33/tcl/libs/ns-lib.tcl" и "opt/ns/2.33/ns-2.33/tcl/libs/ns-node.tcl".
- *$ns* duplex-link *node1 node2 bandwidth delay queue-type*: создает две однонаправленные линии связи указанной пропускной способности и задержки, и подключает к двум указанным узлам. В NS выходная очередь узла выполнена как часть канала, поэтому пользователи должны определить тип очереди при создании линии связи. В рассмотренном сценарии моделирования используется очередь типа DropTail. Если нужно использовать очередь типа RED, следует просто заменить слово DropTail на RED. Реализация линии связи в NS будет показана позже. Так же как и узел, линия связи – составной объект, и пользователи могут создавать его подобъекты и подключать их к узлам. Ссылки на исходные тексты могут быть найдены в файлах "opt/ns/2.33/ns-2.33/tcl/libs/ns-lib.tcl" и "opt/ns/2.33/ns-2.33/tcl/libs/ns-link.tcl". Следует обратить внимание, что в компонент линии связи можно вставлять модули ошибок, чтобы моделировать потери связи (фактически пользователи могут создавать и вставлять любые сетевые объекты). Для того, чтобы выяснить, как делать это, следует обратиться к документации NS.
- *$ns* queue-limit *node1 node2 number*: Эта строка устанавливает ограничение очереди двух однонаправленных линий связи, которые соединяют узлы node1 и node2 указанным числом. Доступные методы объекта Simulator следует смотреть на "opt/ns/2.33/ns-2.33/tcl/libs/ns-lib.tcl" и "opt/ns/2.33/ns-2.33/tcl/libs/ns-link.tcl", либо в NS документации для получения дополнительной информации.

- *$ns* duplex-link-op *node1 node2* ...: Следующая пара строк используется для отображения NAM. Чтобы видеть эффекты этих строк, пользователи могут прокомментировать эти строки и попытаться моделировать.

Теперь, когда основная установка сети выполнена, следующее, что нужно сделать – это установить агенты трафика типа TCP и UDP, источники трафика типа FTP и CBR, и прикреплять их, соответственно, к узлам и агентам.

- set *tcp* [new *Agent/TCP*]: Эта строка показывает, как создать TCP агента. Но в общем, пользователи могут таким образом создавать любого агента или источника трафика. Агенты и источники трафика – фактически основные объекты (не составные объекты), главным образом осуществленные в C++ и связанные с OTcl. Однако нет никаких определенных методов объекта Simulator, которые бы создавали эти экземпляры объекта. Чтобы создавать агенты или источники трафика, пользователь должен знать имена классов этих объектов (Agent/TCP, Agnet/TCPSink, Application/FTP и так далее). Эта информация может быть найдена в NS документации или частично в этом документе. Но один ярлык должен указывать на файл "opt/ns/2.33/ns-2.33/tcl/libs/ns-default.tcl". Этот файл содержит заданные по умолчанию значения конфигурируемых параметров настройки для доступных сетевых объектов. Поэтому это работает как хороший индикатор: какие сетевые объекты доступны в NS и каковы их конфигурируемые параметры.
- *$ns* attach-agent *node agent*: Метод attach-agent прикрепляет созданный объект агента к объекту узла. Фактически, что эта функция вызывает метод attach определенного узла, который прикрепляет к себе данного агента. Поэтому, пользователь может делать такую же самую вещь, например, $n0 attach $tcp. Точно так же, каждый объект агента имеет метод attach-agent, который прилагает к себе объект источника трафика.
- *$ns* connect *agent1 agent2*: После того, как созданы два агента, которые будут связаны друг с другом, следующее – это установление логического сетевого соединения между ними. Эта строка устанавливает сетевое соединение, задавая адре-

сатам назначения пару адресов портов.

Полагая, что вся сетевая конфигурация закончена, следующая вещь, которую надо сделать – написать сценарий моделирования (то есть планирование моделирования). Объект моделирования имеет много методов планирования. Однако, тот, который наиболее часто используется – следующей:

$ns at *time "string"*: Этот метод объекта Simulator заставляет планировщика (scheduler_ – это переменная, которая указывает объекту планировщику, созданному командой [new Scheduler] в начале сценария) наметить выполнение указанной строки в данное время моделирования. Например, *$ns* at 0.1 "*$cbr start*" заставит планировщика вызвать метод start CBR объекта источника трафика, который запускает CBR для передачи данных. Обычно, в NS источник трафика не передает реальные данные, но он уведомляет нижележащего агента, что он имеет некоторое количество данных для передачи, и агент, зная только сколько данных есть для передачи, формирует пакеты и посылает их.

После того как все описания процедур сетевой конфигурации, планирования и окончания моделирования сделаны, единственное, что остается – это запустить моделирование. Это выполняется с помощью *$ns* run..

Планирование событий в NS и анализ трассировки

В этом разделе речь идет о дискретных планировщиках событий NS. Главные пользователи планировщика событий – сетевые компоненты, которые моделируют задержку обработки пакета или которым необходимы таймеры. На рисунке 2 показаны сетевые объекты, использующие планировщик событий. Заметим, что есть сетевые объекты, которые выдают событие, и есть объекты, которые обрабатывают событие позднее в намеченное время. Также заметим, что путь данных между сетевыми объектами отличается от пути событий. Действительно, пакеты передаются от одного сетевого объекта другому, используя метод отправителя *send(Packet* p) {target_->recv(p)}* и метод получателя *recv(Packet*, Handler* h = 0)*.

NS имеет два различных типа реализации планировщиков событий. Это планировщики в реальном масштабе времени и в нереальном масштабе времени. Для планировщика в нереальном масштабе времени, доступны три реализации (List, Heap and Cal-

endar - список, куча и календарь), даже при том, что они - все логически исполняют то же самое. Это - из-за обратной совместимости: некоторые раннее выполненные сетевые компоненты, добавленные пользователем (не оригинальные, включенные в пакет) могут использовать определенный тип планировщика не через общедоступные функции, а внутрисистемным взломом.. По умолчанию в нереальном масштабе времени планировщик установлен в значение Calendar. Планировщик в реальном масштабе времени используется как эмулятор, позволяющий тренажеру взаимодействовать с реальной сетью. В настоящее время эмуляция еще не доработана, хотя экспериментальная версия доступна. Далее приведен пример выбора определенного планировщика событий:

```
. .
set ns [new Simulator]
$ns use-scheduler Heap
. .
```

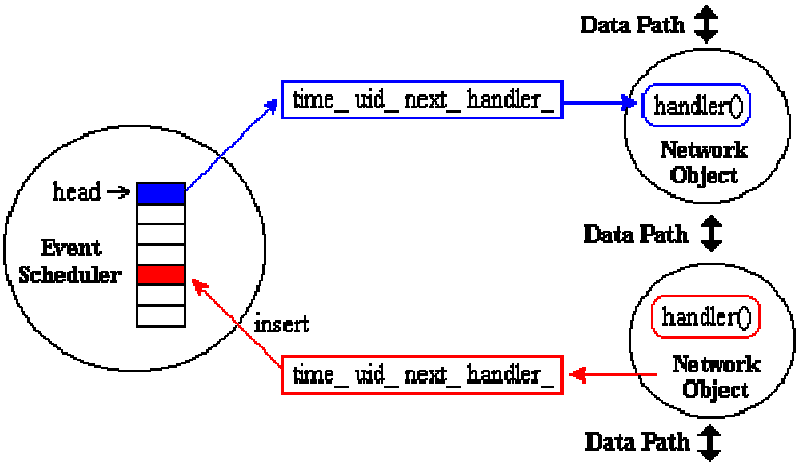

Рис 2. Дискретный планировщик событий

Такое же использование планировщика событий – наметить события моделирования, типа того, когда запустить приложение FTP, когда закончить моделирование, или для сценария моделирования, порожденного прежде выполнения моделирования.

Объект планировщик событий должен сам моделировать методы планирования такие как at *time "string"*, который выпускает специальное событие, называемое AtEvent, в указанное время моделирования *" time"*.. Событие "AtEvent" – фактически потомок класса “Event", который имеет дополнительную переменную для поддержки данной строки *"string"*. Однако, он обработан так же, как нормальный (связанный с пакетом) случай в пределах планировщика событий. Когда моделирование начато, и как только наступает намеченное время для "AtEvent" в очереди событий, "AtEvent" пропускают к обработчику "AtEvent handler" который создан один раз и обрабатывает все AtEvents и OTCL команды, указанные полем *"string"* метода "AtEvent" для выполнения. Далее — версия рассмотренного выше примера с добавленной строкой планирования события моделирования.

```
. . .
set ns [new Simulator]
$ns use-scheduler Heap
$ns at 300.5 "complete_sim"
. . .

proc complete_sim {} {
. . .
}
```

Из приведенного примера можно заметить, что at *time "string*" – метод объекта Simulator (set *ns* [new Simulator]). Но следует помнить, что объект Simulator действует только как интерфейс пользователя, и он фактически вызывает методы сетевых объектов или объектов планирования, которые и делают реальную работу. Следующие строки – частичный список и краткое описание методов объекта Simulator, которые связаны интерфейсом с методами планировщика:

Simulator instproc now # возвращает понятие планировщика текущего времени

Simulator instproc at args # намечает выполнение программы в указанное время

Simulator instproc at-now args # намечает выполнение программы теперь

Simulator instproc after n args # намечает выполнение про-

граммы после n секунд

Simulator instproc run args # старт планировщика

Simulator instproc halt # останов (пауза) планировщика

Рассмотренный ниже раздел демонстрирует пример анализа результатов моделирования. В качестве примера используется тот же самый OTcl сценарий с добавлением нескольких строк для того, чтобы открыть файл трассировки и записывать в него ход моделирования. Сетевая топология и сценарий моделирования формируются также в соответствии с рисунком 1. Для выполнения этого сценария загрузить в свой каталог файл "ns-simple-trace.tcl", размещенный в папке S:\БогомоловСИ\NS\Lab3\Metod\, и набрать в командной строке "ns ns-simple-trace.tcl". (Оригинал программы доступен по адресу <http://nile.wpi.edu/NS/Example/ns-simple-trace.tcl>.

```
# # #
#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Open the Trace file
set tf [open out.tr w]
$ns trace-all $tf

#Define a 'finish' procedure
proc finish {} {
        global ns nf tf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Close the Trace file
        close $tf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}
```

Рис.3. Изменения, вносимые в программу для трассировки

При выполнении Otcl сценария формируется файл трасси-

ровки NAM, который будет использован для ввода в NAM и файл трассировки, называемый "out.tr", который будет использован для анализа результатов моделирования. На рисунке 4 представлен формат файла трассировки а также пример возможных данных трассировки из файла "out.tr".

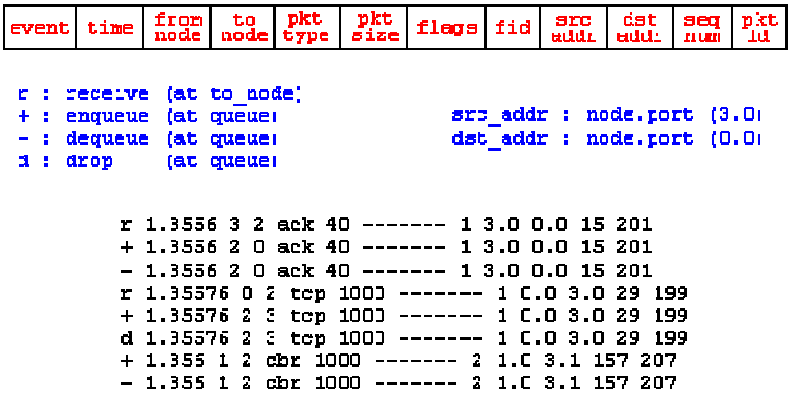

Рис. 4. Пример формата трассировки

Каждая строка трассировки начинается с дескриптора события (+,-, d, r), сопровождаемого временем моделирования (в секундах) этого события, и номерами узлов отправителя и получателя, идентифицирующие линию связи, на которой произошло событие. Здесь приняты следующие условные обозначения: “r” – получение пакета, “+” – добавление пакета к очереди, “-” – снятие пакета из очереди, “d” – удаление пакета из сети. После сведений о типе пакета и его размере (в байтах) следует информации об установленных флагах (в примере показано как"------", поскольку не установлены никакие флаги. В настоящее время, NS применяет только бит уведомления о явной перегрузке (ECN), а остальные биты не используются. Следующее поле – поток идентификаторов (fid) IPv6, которые пользователь может устанавливать для каждого потока при вводе OTcl сценария. Даже в том случае, если поле fid не использовано при моделировании, пользователи могут использовать это поле для целей анализа. Поле fid также используется при задании цвета потока для дисплея NAM. Следующие два поля – адреса источника и при-

емника в формате "узел.порт". Следующее поле показывает номер последовательности пакета протокола сетевого уровня. Заметим, что даже несмотря на то, что UDP реализация не использует номер последовательности, NS сохраняет строчку UDP номеров последовательности пакетов для целей анализа. Последнее поле показывает уникальный идентификатор пакета.

Динамика сетей

В этом разделе приведен пример динамичной сети, где маршрутизация приспосабливается к повреждениям связи. Попутно показано, как можно выполнить большое количество узлов в Tcl массиве вместо предоставления каждому узлу его собственного названия.

Предлагается назвать Tcl сценарий для этого примера 'example3.tcl'. В файл уже может быть вставлен шаблон example.tcl из каталога Lab2.

Как всегда, сначала должна быть создана топология, хотя на сей раз выбирается иной подход, более удобный при создании большей топологии. Следующий блок программы создает семь узлов и хранит их в массиве n().

```
for {set i 0} {$i < 7} {incr i} {
  set n($i) [$ns node]
}
```

Циклы 'for' используются и в других языках программирования, так, что структура понятна. Следует обратить внимание, что массивы, точно так же, как другие переменные в Tcl, не должны быть объявлены первыми.

Теперь следует соединить узлы, чтобы создать круговую топологию. Следующая часть программы поначалу может выглядеть сложноватой.

```
for {set i 0} {$i < 7} {incr i} {
  $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}
```

Этот цикл 'for' соединяет каждый узел массиве со следующим узлом за исключением последнего узла, который соединяется с первым. Чтобы это выполнять, использован оператор '%' (по модулю).

Теперь при запуске программы топология в NAM сначала

может выглядеть немного странной, но после нажатия кнопки 're-layout' схема примет более привычный вид.

Следующим шагом является пересылка некоторых данных от узла n0 к узлу n3.

```
#Create a UDP agent and attach it to node n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0

$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

Представленная выше программа должна выглядеть уже знакомой к этому времени. Единственное различие от прошлых разделов в том, что теперь используются элементы массива узлов.

Если теперь запустить сценарий (предварительно сохранив его под именем 'example3a.tcl'), будет видно, что поток информации выбирает самый короткий путь от узла n0 к узлу n3 через узлы n1 и n2, как и следовало ожидать. Затем добавляется другая интересная особенность. Предполагается нарушение связи между узлами n1 и n2 (которые используются трафиком) в течение секунды.

```
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
```

Вероятно, не слишком трудно понять эти две строки. Теперь можно стартовать сценарий снова (предварительно сохранив его под именем 'example3b.tcl'), и можно будет видеть, что между первой и второй секундами связь будет прервана, и все данные,

посланные от узла n0, будут потеряны.

Далее показано, как использовать динамическую маршрутизацию для решения этой проблемы. В начало Tcl сценария, после того, как был создан объект Simulator, добавляется следующая строка:

$ns rtproto DV

Запустив моделирование снова (предварительно сохранив его под именем 'example3.tcl'), можно будет увидеть, как вначале множество небольших пакетов пробегают по сети. Если замедлить темп NAM достаточно щелкнуть на одном из пакетов, и будет видно, что это – 'rtProtoDV' пакеты, которые используются для обмена маршрутной информацию между узлами. Когда соединение нарушится в 1.0 секунды, маршрутизация снова будет откорректирована, и трафик будет перенаправлен через узлы 6, 5 и 4.

ПРЕДВАРИТЕЛЬНАЯ ПОДГОТОВКА

Ознакомиться с основами моделирования сетей ЭВМ с помощью сетевого имитатора NS2 по материалам данного практикума и рекомендуемых в нем литературных источников, а также повторить основы функционирования сетей по материалам лекций.

Изучить раздел «Краткие сведения о моделировании сетей ЭВМ с помощью сетевого имитатора NS2» методических указаний по данной лабораторной работе и подготовить необходимые Tcl скрипты для работы с NS2, рассмотренные в данном разделе.

Доработать сценарий example3.tcl таким образом, чтобы параметры сети соответствовали варианту таблицы 1. В таблице приняты следующие обозначения:

“N” – общее количество узлов в сети; “frq” – пропускная способность линии связи; “del” – задержка обработки пакета в узле связи (полагать характеристики всех линий связи одинаковыми); “tm” – общее время моделирования системы; “ns” – номер узла отправителя пакетов; “nd” – номер узла получателя пакетов; “nn” – номера узлов, между которыми будет промоделировано нарушение связи; “sr” – момент времени нарушения связи в линии; “sp” – момент времени восстановления связи. При составлении программы не забывать, что нумерация узлов начи-

наются с 0.

Таблица 1

	1	2	3	4	5	6	7	8	9	10
N	8	9	7	8	9	7	8	9	7	8
frq	0,5	0,7	0,6	1,0	0,9	1,1	1,2	0,9	0,8	0,7
del	20	15	18	10	12	9	8	16	15	16
tm	3,2	2,8	3,3	2,9	3,0	2,9	3,2	3,0	2,8	3,3
ns	4	6	3	2	8	5	7	9	1	0
nd	7	9	6	5	5	2	4	6	4	3
nn	5-6	7-8	4-5	3-4	6-7	4-3	7-5	8-7	2-3	1-2
sr	1,1	0,8	1,5	0,9	1,2	1,0	1,1	1,2	0,9	1,3
sp	1,8	1,5	2,1	1,9	1,8	1,7	1,9	1,8	1,7	2,1

Результатами предварительной подготовки по данной работе являются файлы Tcl скрипта, подготовленные для моделирования сетей с помощью имитатора NS2.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. С чего начинается NS сценарий?
2. Каковы результаты действия первой строки NS сценария?
3. Каковы возможности методов объекта Simulator?
4. Как создаются узлы сети в NS симуляторе? Пример программы.
5. На каком уровне стека ЭМВОС (TCP/IP) работают узлы в NS? По каким признакам различаются узлы на этом уровне?
6. Как создаются в NS соединения между узлами?
7. Какие параметры соединения устанавливаются при моделировании сети?
8. На каком уровне стека ЭМВОС (TCP/IP) работают линии связи в NS? По каким признакам различаются работают линии связи на этом уровне?
9. Что такое агенты? Какие параметры агентов устанавливаются при моделировании сети?
10. Как агенты связаны с узлами сети и между собой?
11. На каком уровне стека ЭМВОС (TCP/IP) работают агенты в NS? По каким признакам различаются агенты на

этом уровне?

12. На каком уровне стека ЭМВОС (TCP/IP) работают источники трафика в NS? По каким признакам различаются работают источники трафика на этом уровне?
13. В каких случаях пакет попадает в очередь? Каким образом он ее покидает?
14. Какие параметры очереди устанавливаются при моделировании соединения?
15. При каких условиях пакет удаляется из сети?
16. Что такое планирование событий? Как оно реализуется в NS?
17. С какой целью в NS используется файл трассировки?
18. Раскрыть формат данных в файле трассировки.
19. Пояснить условные обозначения данных файла трассировки.
20. Что такое флаги в заголовке пакета сетевого уровня? С какой целью они используются?
21. Каким образом в NS моделируются аварийные ситуации в сети?
22. Что происходит с пакетом UDP при аварийной ситуации в сети?
23. Что происходит с пакетом TCP при аварийной ситуации в сети?
24. Что такое динамическая маршрутизация сети?
25. Как динамическая маршрутизация реализуется в NS?

ЛАБОРАТОРНОЕ ЗАДАНИЕ

1. Ознакомиться с основами моделирования сетей ЭВМ с помощью сетевого имитатора NS2 по материалам данного практикума и рекомендуемых в нем литературных источников (каталоги \Lab3\Metod\ и \Metod\). Создать папку с названием Lab_3 на сервере X. Промежуточные и окончательные результаты данной работы сохранять в этом каталоге.

2. Программу, разработанную в результате предварительной подготовки, скопировать в созданную папку Lab_3.

3. Открыть окно «Терминал». С помощью команды «Изменить каталог» (cd *path*), где path – путь доступа к каталогу Lab_3, перейти в свой рабочий каталог.

4. Скопировать. файл ns-simple.tcl, размещенный в папке S:\БогомоловСИ\NS\Lab3\Metod\, в свой рабочий каталог и провести анализ этой программы.

5. Запустить файл ns-simple.tcl на приглашение командного процессора (ns ns-simple.tcl). В ходе моделирования исследовать параметры пакетов всех потоков. В отчете отметить типы пакетов, их размеры, интервалы времени между пакетами, номера и моменты передачи первого и последнего пакетов в каждом потоке.

6. На основе файла ns-simple.tcl разработать программу моделирования сети, топология которой соответствует рисунку 5. Символы A, B… F обозначают номера узлов в сети, Ag1… Ag4 – типы агентов, приложенных к узлам, G1… G4 – соответствующие генераторы трафика. При этом параметры сетевых компонентов определяются вариантом таблицы 2. В строке Frq0 указана пропускная способность линии CD, , в строке frq1 – остальных линий связи (в мегабитах за секунду). В строке del0 приведена задержка обработки пакета в линии CD, в строке del1 – за-

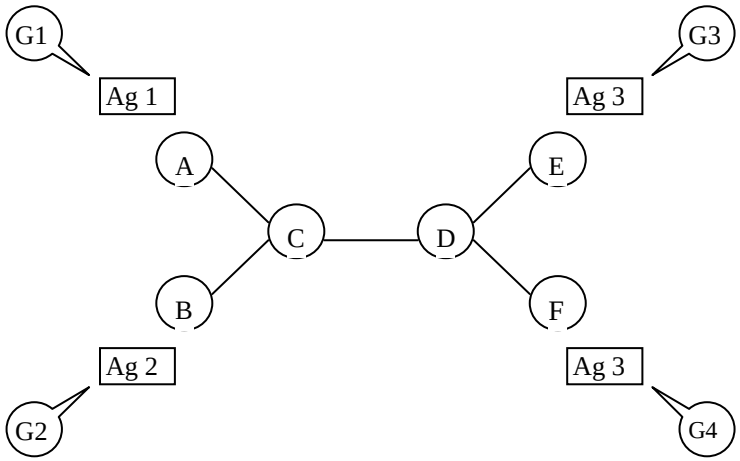

Рис.5. Топология сети

держка обработки пакета в остальных линиях. В строке TCP ука-

заны конечные узлы TCP соединения, в строке UDP – узлы UDP соединения (первым указан передающий узел).

Таблица 2

	1	2	3	4	5	6	7	8	9	10
A	n1	n3	n5	n2	n4	n3	n1	n0	n5	n2
B	n5	n2	n0	n4	n5	n1	n4	n3	n4	n1
C	n0	n4	n1	n3	n0	n4	n2	n4	n3	n5
D	n3	n5	n2	n5	n1	n2	n0	n5	n2	n0
E	n2	n0	n3	n1	n2	n0	n5	n1	n1	n3
F	n4	n1	n4	n0	n3	n5	n3	n2	n0	n4
Frq0	1,8	1,6	1,4	1,7	1,9	1,5	2,1	2,4	2,3	2,2
del0	20	22	26	21	11	24	18	16	17	18
frq1	0,9	0,8	0,7	0,8	0,9	0,7	1,1	1,2	1,2	1,1
del1	40	44	52	42	22	48	36	32	34	36
TCP	EB	FA	BE	AF	EA	FB	AE	BF	FA	EB
UDP	FA	EB	AF	BE	FB	EA	BF	AE	EB	FA

7. Запустить на выполнение подготовленную программу. Убедиться в соответствии функционирования программы исходному заданию.

8. Скопировать. файл ns-simple-trace.tcl, размещенный в папке S:\БогомоловСИ\NS\Lab3\Metod\, в свой рабочий каталог и провести анализ этой программы.

9. Запустить файл ns-simple-trace.tcl на приглашение командного процессора (ns ns-simple-trace.tcl). В ходе моделирования исследовать параметры пакетов всех потоков. В отчете отметить типы пакетов, их размеры, интервалы времени между пакетами, номера и моменты передачи первого и последнего пакетов в каждом потоке.

10. В текстовом редакторе открыть файл out.tr. Выяснить все типы пакетов, передаваемых по сети, их размеры, адреса отправителей и получателей.

11. По файлу out.tr проследить все события (от отправления до получения на конечных узлах), происходящие с пакетом, имеющим номер, равный N+20, где N - номер варианта. Для этого собрать в один файл все строки, описывающие эти события. Уметь объяснить все записи событий, происходящих с этим пакетом. Определить время доставки пакета адресату.

12. Провести анализ функционирования очереди. Для этого найти событие удаления пакета из сети, и провести регистрацию всех событий в этой очереди, предшествующих удалению пакета. Убедиться, что удаление пакета произошло в соответствии с настройками сети.

13. Запустить на выполнение программу, разработанную в результате предварительной подготовки. Убедиться в соответствии функционирования программы исходному заданию. В случае расхождения – доработать программу.

14. В отчет следует внести файлы программ, разработанные на этапе предварительной подготовки и в результате экспериментальных исследований, а также файлы, полученные по результатам анализа материалов моделирования сети.

ИССЛЕДОВАНИЕ ХАРАКТЕРИСТИК TCP С ПОМОЩЬЮ СЕТЕВОГО ИМИТАТОРА NS2

Целью работы «Исследование характеристик TCP с помощью сетевого имитатора NS2» является экспериментальное исследование основных характеристик сетей ЭВМ путем компьютерного моделирования, углубление знаний и расширение сведений об основных протоколах стека TCP/IP (протокола транспортного уровня TCP), а также закрепление навыков работы с сетевым имитатором NS2.

КРАТКИЕ СВЕДЕНИЯ О МОДЕЛЯХ TCP AGENTS ИМИТАТОРА NS2

Основное содержание данной раздела использовано из учебных материалов VINT group: The ns Manual (formerly ns Notes and Documentation). The VINT Project/ Kevin Fall, Kannan Varadhan, 2009. Доступно по адресу: http://www.isi.edu/nsnam/ns/ns-documentation.html. Копия этого руководства размещена в папке S:\БогомоловСИ\NS\Metod\.

Этот раздел описывает функционирование агентов TCP в ns. Есть два основных типа моделей агентов TCP: Однонаправленные агенты и двунаправленные агенты. Однонаправленные агенты далее подразделены на семейства отправителей TCP (которые отрабатывают различные перегрузки и методы контроля ошибок) и получателей ("стоки"). Двунаправленный агент симметричен в том смысле, что он представляет и отправителя и получателя.

В настоящее время поддержаны однонаправленные TCP агенты отправители:

•Agent/TCP - “tahoe” TCP отправитель

•Agent/TCP/Reno - “Reno” TCP отправитель

•Agent/TCP/Newreno - Reno с модификацией

•Agent/TCP/Sack1 - TCP с выборочным повтором (по RFC 2018)

•Agent/TCP/Vegas - TCP Vegas

•Agent/TCP/Fack - Reno TCP с ускоренным подтверждение

•Agent/TCP/Linux – TCP отправитель с поддержкой SACK, выполняемой модулем управления очередью ядра Linux.

На данный момент поддержаны однонаправленные TCP агенты получатели:

•Agent/TCPSink - TCP получатель с одним ACK на пакет

•Agent/TCPSink/DelAck - TCP получатель с управляемой задержкой на ACK

•Agent/TCPSink/Sack1 - получатель с выборочными ACK (по RFC 2018)

•Agent/TCPSink/Sack1/DelAck - Sack1 с DelAck

Двунаправленный экспериментальный получатель поддерживает только форму Reno TCP:

•Agent/TCP/FullTcp

Данный раздел содержит краткий обзор и пример конфигурирования основных TCP агентов отправителей и получателей (получатели не требует никакой конфигурации). Также кратко представлены внутренние основы агента отправителя и описание дополнений для других типов агентов, которые были включены в симулятор.

Однонаправленные отправители

Симулятор поддерживает несколько версий абстрактного отправителя TCP. Эти объекты пытаются фиксировать сущность перегрузок TCP и поведение контроля ошибок, но не предназначены для того, чтобы быть точными копиями реального выполнения TCP. Они не содержат динамического объявления окна, они выполняют вычисления номеров сегмента и номеров ACK целиком в единицах пакетов, нет никаких SYN/FIN соединений establishment/teardown, и никакие данные никогда не передаются (например, никакие контрольные суммы или срочные данные).

Базовый отправитель TCP (Tahoe TCP)

Агент TCP “Tahoe” Agent/TCP выполняет контроль перегрузки в сети и оценку времени кругооборота способом, подобным версии TCP, выпущенной с релизом 4.3BSD “Tahoe” UN’X систем UC Berkeley. Окно переполнения cwnd_ увеличивается на один пакет с новым ACK, полученным в течение медленного пуска (пока cwnd_ < ssthresh_ - порог перегрузки) и увеличивается на 1/cwnd_ для каждого нового ACK, полученного на этапе избегания перегрузки (когда cwnd_ ≥ .ssthresh).

Отклик на перегрузку Tahoe TCP предполагает, что пакет был потерян (из-за перегрузки в сети) когда он обнаруживает

признак NUMDUPACKS (определен в tcp.h, на данный момент равен 3) дубликатов ACK, или когда истекает таймер повторной передачи. В любом случае Tahoe TCP реагирует установкой порога перегрузки ssthresh_ на половину текущего размера окна (минимальное из значений окна переполнения cwnd_ или окна пользователя window _) или 2, какое больше. Затем он инициализирует cwnd_ снова к значению windowInit_ (стартовый размер окна при медленном пуске). Обычно это вынуждает вводить TCP медленным пуском.

Оценка времени кругооборота и выбор таймаута RTO Четыре переменные используются для оценки времени кругооборота и установки таймера повторной передачи: оценка времени кругооборота rtt_, сглаженная оценка времени кругооборота srtt_, отклонение оценки времени кругооборота rttvar_, период тактового генератора tcpTick_ и экспоненциальная константа отсрочки времени кругооборота backoff_. TCP инициализирует rttvar к 3/tcpTick_ и backoff к 1. Когда устанавливается любой будущий таймер повторной передачи, то таймаут устанавливают на текущее время плюс max (bt (a + 4v + 1), 64) секунды, где b – текущее значение backoff, t - значение tcpTick, a - значение srtt, и v - значение rttvar.

Отчеты времени кругооборота прибывают с новым ACKs. Отчет RTT вычисляется как разница между текущим временем и полем “time echo” в ACK пакете. Когда первый отсчет принят, его значение используется как начальное значение для srtt_. Половина первого отсчета используется как начальное значение для rttvar_. Для последующих отсчетов значения сглаженной оценки времени кругооборота srtt_ и отклонения оценки времени кругооборота rttvar_ модифицируются следующим образом:

srtt = 7/8 srtt + 1/8 sample,

rttvar = 3/4 rttvar + 1/4 |sample - .srtt|,

где sample – отсчет времени кругооборота

Конфигурации

Пуск TCP моделирования требует создания и конфигурирования агента, прикрепления источника данных прикладного уровня (генератора трафика), и пуска агента и генератора трафика.

Простые конфигурации

Создание агента

```
set ns [new Simulator] ;# инициализация преамбулы
set node1 [$ns node] ;# постоянное размещение агента на этом узле
set node2 [$ns node] ;# постоянное размещение агента на этом узле
set tcp1 [$ns create-connection TCP $node1 TCPSink  $node2 42]
$tcp set window_ 50 ;# конфигурация TCP агента
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 0.0 "$ftp start"
```

Этот пример иллюстрирует использование симулятором встроенной функции create-connection. Аргументы этой функции: создать агента отправителя, узел отправителя, создать агента получателя, узел получателя, и идентификатор потока ID для использования в соединении. Функция выполняет создание этих двух агентов, установление поля потока ID в агентах, прикрепление агентов источника и приемника к соответствующим узлам, и наконец соединение агентов (то есть установление соответствующих адресов и портов источника и получателя). Значение возвращаемой функции - имя созданного исходного агента.

Источник данных TCP агент TCP не формирует никаких собственных прикладных данных; вместо этого, пользователь моделирования может подключить любой модуль формирования трафика к агенту TCP для генерации данных. Для TCP обычно используются два приложения: FTP и Telnet. FTP представляет оптовую передачу данных большого размера, и telnet выбирает ее размеры передачи беспорядочно из tcplib (см. файл tcplib-telnet.cc).

Другие параметры конфигурации

В дополнение к описанному выше параметру window_ агент TCP поддерживает дополнительные переменные конфигурации. Каждая из переменных, описанных в этом подразделе, является и переменной класса, и переменной экземпляра. Изменение переменной класса изменяет заданное по умолчанию значение для всех агентов, которые созданы впоследствии. Изменение переменной экземпляра определенного агента затрагивает значения,

используемые только этим агентом. Например,

Agent/TCP set window_ 100 ;# Изменяет переменную класса

$tcp set window_ 2.0 ;# Изменяет, window_ только для объекта $tcp

Заданные по умолчанию параметры для каждого агента TCP:

Agent/TCP set window_ 20 ;# максимальное значение размера окна

Agent/TCP set windowInit_ 1 ;# установка/сброс значения cwnd

Agent/TCP set windowOption_ 1 ;# алгоритм избежания перегрузки (1: стандарт)

Agent/TCP set windowConstant_ 4 ;# используется только когда windowOption! =1

Agent/TCP set windowThresh_ 0.002 ;# используется в вычислении усредненного окна

Agent/TCP set overhead_ 0 ;# !=0 добавляет случайный интервал времени между посылками

Agent/TCP set ecn_ 0 ;# TCP должна реагировать на бит ecn

Agent/TCP set packetSize_ 1000 ;# размер пакета, используемый отправителем (байты)

Agent/TCP set tcpTick_ 0.1 ;# таймер гранулирован в секундах (.1– НЕСТАНДАРТ)

Agent/TCP set maxrto_ 64 ;# ограничение на RTO (секунды)

Agent/TCP set dupacks_ 0 ;# счетчик копий ACK

Agent/TCP set ack_ 0 ;# самый высокий полученный ACK

Agent/TCP set cwnd_ 0 ;# окно перегрузки (пакеты)

Agent/TCP set awnd_ 0 ;# усредненный cwnd (экспериментальный)

Agent/TCP set ssthresh_ 0 ;# порог медленного пуска (пакеты)

Agent/TCP set rtt_ 0 ;# отсчет rtt

Agent/TCP set srtt_ 0 ;# усредненный rtt

Agent/TCP set rttvar_ 0 ;# значение отклонения rtt отсчета

Agent/TCP set backoff_ 0 ;# текущий фактор отсрочки RTO

Agent/TCP set maxseq_ 0 ;# максимальный номер посланного сегмента (пакет)

Для большинства экспериментов, вероятно, немногие параметры конфигурации будут требовать модификации. Обычно

включают наиболее изменяемые параметры: window_ и packetSize_. Первый из них ограничивает использования окна TCP, и как рассматривалось, играет роль объявления окна получателя в реальной TCP (хотя оно остается постоянным). Размер пакета по существу функционирует подобно размеру MSS в реальной TCP. Изменяя эти параметры, можно оказать сильное влияние на поведение TCP. Обычно TCP с большими размерами пакета, большими окнами и меньшими временами кругооборота (результат топологии и перегрузки) более агрессивны в запрашивании сетевой пропускной способности.

Другие однонаправленные отправители

Reno TCP Агент Reno TCP agent очень подобен агенту Tahoe TCP, кроме этого он также включает быстрое восстановление в тех случаях, когда текущее окно перегрузки"раздуто" числом копий ACK, полученных отправителем TCP перед получением нового ACK. Понятие “новый ACK” относится к любому ACK, значение которого выше чем наивысшее замеченное до сих пор. Кроме того, агент Reno TCP не возвращается к медленному пуску в течение быстрой повторной передачи. Скорее, он уменьшит установку окна перегрузки на половину текущего окна и сбросит ssthresh_ для соответствия этому значению.

Newreno TCP Этот агент основан на Reno TCP agent, но у которого изменены действия, принимаемые при получении нового ACK. Чтобы выйти на быстрое восстановление, отправитель должен получить ACK для самого высокого посланного номера последовательности. Таким образом, новые “частичные ACK” (т.е. те, которые представляют новые ACK, но не представляют ACK для всех ожидающих выполнения данных) не уменьшают размеры окна (и возможно ведут к останову, характерному Reno).

Vegas TCP Этот агент осуществляет “Vegas” TCP.

Sack TCP Этот агент осуществляет выборочное повторение, основанное на выборочных ACK, обеспечиваемых получателем.

Fack TCP Этот агент выполняет “ускоренный ACK” TCP (модификация Sack TCP).

Linux TCP Этот агент выполняет TCP модули управления перегрузкой, импортированные из Linux ядра. Агент вырабатывает результаты моделирования, которые являются совместимы-

ми на уровне траектории окна перегрузки с поведением Linux хостов.

Пользователи моделирования могут модифицировать или импортировать новые модули управления перегрузкой из исходного кода ядра Linux для этого агента. Модули управления перегрузкой Linux компилированы в бинарный код NS-2. Пользователи могут выбирать различные алгоритмы управления перегрузкой, различные параметры модуля управления перегрузки, и различные Linux параметры TCP для различных экземпляров этого агента. Этот агент поддерживает SACK. Получателю, который поддерживает SACK, рекомендуют работать с этим агентом. Есть обучающая программа для использования этого агента.

Реализация этого агента свободно следует за Linux TCP пакетной обработкой маршрутизации и вызывает исходные коды управления перегрузкой из ядра Linux для изменения управлением перегрузкой, связанного с параметрами (например окно перегрузки, порог медленного пуска и т. д.).

Для достижения результатов моделирования, близких к рабочим характеристикам Linux, этот агент заменяет заданные по умолчанию значения следующих параметров согласно параметрам Linux:

Agent/TCP/Linux set maxrto_ 120

Agent/TCP/Linux set ts_resetRTO_ true

Agent/TCP/Linux set delay_growth_ false

TCP получатели (стоки)

Отправители TCP, описанные выше, представляют однонаправленных отправителей данных. Они должны быть одного ранга с объектами “TCP sink”.

Базовый TCP получатель Основной объект TCP sink (Agent/TCPSink) ответствен за возвращение ACK одноранговому объекту TCP отправителя. Он формирует один ACK на полученный пакет. Размер ACK может быть конфигурирован. Создание и конфигурация объекта TCP sink обычно выполняются автоматически запросом библиотеки (см. create-connection выше).

Параметры конфигурации

Agent/TCPSink set packetSize_ 40

TCP получатель с задержанными ACK Объект задержанный

ACK агент (Agent/TCPSink/DelAck) доступен получателю TCP для моделирования для того, чтобы было менее одного ACK на полученный пакет. Этот объект содержит ограничиваемую переменную interval_, которая задает количество секунд для ожидания между ACK. Получатель задержанного ACK осуществляет агрессивную ACK политику, в силу чего задерживаются только ACK для пакетов, поступающих по порядку. Пакеты вне порядка вызывают немедленную генерацию ACK.

Параметры конфигурации

Agent/TCPSink/DelAck set interval_ 100ms

Sack TCP получатели Получатели TCP с выборочным подтверждением (Agent/TCPSink/Sack1) выполняют SACK генерацию, смоделированную после описания SACK в RFC 2018. Этот объект включает ограничиваемую переменную maxSackBlocks_, которая задает максимальное число блоков информации в ACK, доступном для сохраняемой информации SACK. Заданное по умолчанию значение для этой переменной - 3, в соответствии с ожидаемым использованием SACK с RTTM (см. RFC 2018, раздел 3). Отсроченный и выборочный ACK осуществляются совместно объектом типа Agent/TCPSink/Sack1/DelAck.

Параметры конфигурации

Agent/TCPSink set maxSackBlocks_ 3

Архитектура и внутреннее строение

Основной агент TCP (класс Agent/TCP) создан как совокупность программ для посылки пакетов, обработки ACK, управления окном отправителя и обработкой таймаутов. В общем, каждая из этих программ может быть отменена функцией с тем же самым названием в порождающем классе (поэтому осуществлено столько вариантов отправителя TCP).

Заголовок TCP Заголовок TCP определен структурой hdr_tcp в файле ~ns/tcp.h. Основной агент использует только следующую подгруппу полей:

ts_ / * текущее время посылки пакета источником */

ts_echo_ / * для ACKs: поля timestamp из пакета, связанного с этим ACK */

seqno_ / * номер последовательности для сегмента этих данных или ACK (Примечание: перегрузка!) */

reason_ / * устанавливается отправителем при (повторной)

передаче для того, чтобы проследить причину для отправки */

Функции для отправки данных Заметим, что вообще отправитель TCP фактически никогда не посылает данных (он устанавливают только размер пакета).

send_much(force, reason, maxburst - эта функция пытается посылать столько пакетов, сколько позволяет окно отправителя. Она также сохраняет трассировку того, сколько пакетов послано и ограничено общим количеством maxburst.

Функция output(seqno, reason) посылает один пакет с данным номером последовательности и обновляет максимум переменной номера последовательности отправителя (maxseq _), чтобы сохранить данный номер последовательности, если он пока самый большой из посланных. Эта функция также назначает различные поля в заголовке TCP (номер последовательности, timestamp, причина для передачи). Эта функция также устанавливает таймер повторной передачи, если он уже не в ожидании.

Функции для управления окна Используемое окно отправителя каждый раз задается функцией window(). Она возвращает минимум окна перегрузки и переменную wnd _, которая представляет объявляемое окно получателя.

opencwnd () - эта функция открывает окно перегрузки. Она вызываться, когда прибывает новый ACK. При медленном пуске функция просто увеличивает cwnd_ с каждым полученным ACK. При предотвращении перегрузки – стандартная конфигурация увеличивает cwnd_ на его обратное значение. Другие опции роста окна в течение избегания перегрузки поддерживаются, но они экспериментальны (и не зарегистрированы).

closecwnd (int, how) - эта функция уменьшает окно перегрузки. Это может быть вызвано несколькими способами: при вводе быстрой повторной передачи из-за истечения таймера, или из-за уведомления перегрузки (установлен бит ECN). Эта переменная указывает как должно быть восстановлено окно перегрузки. Значение 0 используется для таймаутов повторной передачи и быстрой повторной передачи в Tahoe TCP. Это обычно заставляет TCP вводить медленный пуск и уменьшать ssthresh_ до половины текущего окна. Значение 1 используется Reno TCP для осуществления быстрого восстановления (которое избегает возвращаться к медленному пуску). Значение 2 используется для со-

кращения окна из-за ECN признака. Оно сбрасывает окно перегрузки к его начальному значению (обычно обусловленным медленным пуском), но не изменяет ssthresh_.

Функции для обработки ACK

recv () - эта функция - основной путь приема для ACK. Обратите внимание, что, так как используется только одно направление потока данных, эта функция должна вызываться только с чистым ACK пакетом (то есть без каких-либо данных). Функция хранит метку времени от ACK в ts_peer _, и проверяет присутствие ECN бита (сокращающего окно отправителя, если назначен). Если ACK - новый ACK, она вызывает newack (), а иначе проверяет наблюдением, является ли он копией последнего замеченного ACK. Если это так, вводит быструю повторную передачу, закрывая окно, сбрасывая таймер повторной передачи и посылая пакет вызовом send_much.

newack () - эта функция обрабатывает "новый" ACK (тот, который содержит номер ACK выше, чем любой замеченный прежде). Функция устанавливает новый таймер повторной передачи, вызывая newtimer (), обновляет оценку RTT, вызывая rtt_update, и обновляет(модифицирует) самые высокие и последние ACK переменные.

Функции для управления таймером повторной передачи Эти функциями служат двум целям: оценке времени кругооборота и установке таймера фактической повторной передачи. rtt_init - эта функция инициализирует srtt_ и rtt_ к нулю, устанавливает rttvar_ в 3/tcp_tick _ и устанавливает множитель backoff в 1.

rtt_timeout - эта функция задает значение таймаута в секундах, которое должны использоваться, чтобы наметить следующий таймер повторной передачи. Она вычисляет это на основании текущих оценок средних значений и отклонений времени кругооборота. Кроме того, она осуществляет возврат экспоненциального таймера Карна для многократных последовательных таймаутов повторной передачи.

rtt_update - эта функция в качестве аргумента берет измеренный RTT и усредняет его для выполнения оценки среднего и отклонения согласно описанию выше. Обратите внимание, что t_srtt_ и t_rttvar оба сохранены с фиксированной точкой (целые числа). Они имеют, соответственно, 3 и 2 бита справа от двоич-

ной точки.

reset_rtx_timer – Эта функция вызывается в течение быстрой повторной передачи или в течение таймаута. Она устанавливает таймер повторной передачи, вызывая set_rtx_timer, и, если вызвано таймаутом, также вызывает rtt_backoff.

rtt_backoff - эта функция возвращает таймер повторной передачи (удваивая его).

newtimer - эта функция вызываема только когда прибывает новый ACK. Если левый край окна отправителя - вне ACK, то вызывается set_rtx_timer, иначе она отменяется, если таймер повторной передачи в ожидании.

Отслеживание динамики TCP

Поведение TCP часто наблюдается созданием графика в координатах: номер последовательности – время. Обычно выполненная трассировка допускает отслеживание на соединении, по которому пройдут пакеты TCP. Поддержаны два метода отслеживания: заданный по умолчанию (используемый для рассмотрения агентов TCP), и расширенный, используемый только для FullTcP.

Трассировка динамики одностороннего TCP Пакеты TCP, сформированные одним из однонаправленных агентов TCP и предназначенные для агента TCP sink, прошедшие через отслеживаемое соединение, выработают строки файла трассировки в формате:

```
+ 0.94176 2 3 tcp 1000 ------ 0 0.0 3.0 25 40
+ 0.94276 2 3 tcp 1000 ------ 0 0.0 3.0 26 41
d 0.94276 2 3 tcp 1000 ------ 0 0.0 3.0 26 41
+ 0.95072 2 0 ack 40 ------ 0 3.0 0.0 14 29
- 0.95072 2 0 ack 40 ------ 0 3.0 0.0 14 29
- 0.95176 2 3 tcp 1000 ------ 0 0.0 3.0 21 36
+ 0.95176 2 3 tcp 1000 ------ 0 0.0 3.0 27 42
```

Точный формат этого файла следа дается в разделе 26.4 руководства [The ns Manual (formerly ns Notes and Documentation) The VINT Project]. При рассмотрении TCP уместны пакеты типа tcp или ack. Их тип, размер, номер последовательности (номер ack для ack пакетов), и время прибытия / отправления / удаления задается полем, соответственно, 5, 6, 11, и 2. Знак “+” указывает прибытие пакета, “-“ – отправление и “d” – удаление. Множество

сценариев обрабатывает этот файл, чтобы выполнить графический вывод или статистические резюме (см., например, ~ns/test-suite.tcl, процедура finish).

Основные команды

Далее следует список команд, используемых при установке/управлении потоками TCP для имитаций:

set tcp0 [new Agent/TCP]

Это создает экземпляр агента TCP. В настоящее время в ns осуществлены несколько разновидностей агентов TCP отправителей и TCP получателей (или стоков). На данный момент доступны TCP отправители: Agent/TCP, Agent/TCP/Reno, Agent/TCP/Newreno, Agent/TCP/Sack1, Agent/TCP/Vegas, Agent/TCP/Fack. TCP получатели, доступные в настоящее время: Agent/TCPSink, Agent/TCPSink/DelAck, Agent/TCPSink/Sack1, Agent/TCPSink/Sack1/DelAck. Имеется также двунаправленная реализация tcp под названием Agent/TCP/FullTcp.

Параметры конфигурации для потока TCP могут быть установлены следующими:

$tcp set window_ <wnd-size>

Для всех возможных параметров конфигурации, доступных для TCP см. раздел 35.1.4 руководства [The ns Manual (formerly ns Notes and Documentation) The VINT Project]. Заданные по умолчанию значения конфигурации могут быть найдены также в ns/tcl/lib/ns-default.tcl.

Далее – пример простой установки TCP соединения:

set tcp [new Agent/TCP] ;# создать агента tcp

$ns_ attach-agent $node_(s1) $tcp ;# связать агента с с узлом отправителя

$tcp set fid_ 0 ;# установить поле идентификатора потока tcp

set ftp [new Application/FTP] ;# создать ftp трафик

$ftp attach-agent $tcp ;# связать ftp трафик с tcp агентом

set sink [new Agent/TCPSink] ;# создать агента tcpsink

$ns_ attach-agent $node_(k1) $sink ;# связать агента с с узлом получателя

$sink set fid_ 0 ;# установить поле идентификатора потока sink

$ns_ connect $ftp $sink ;# действительное подключение источника к стоку

$ns_ at $start-time "$ftp start" ;# запустить поток ftp

ПРЕДВАРИТЕЛЬНАЯ ПОДГОТОВКА

Ознакомиться с основными характеристиками протокола TCP по материалам данного практикума и рекомендуемых в нем литературных источников, а также разделов 8 и 9 учебного пособия В.Г. Козлова, Е.С. Семигук «Программные средства систем связи», электронная версия которого размещена на сервере S ЛВС кафедры ТОР [S:\\ Библиотека кафедры ТОР \ Программные средства систем связи \ (. – Томск: ТМЦДО, 2004. 156 с.)]

Повторить основы функционирования транспортного уровня модели OSI по материалам лекций и рекомендованной литературы. Восстановить в памяти основные принципы и особенности реализации в протоколе TCP организации гарантированной доставки сообщений и механизмов ее регулирования. Уяснить роль и значение таймеров, разобраться с соображениями по выбору временных интервалов.

Изучить раздел «Краткие сведения о моделях TCP Agents имитатора NS2» методических указаний по данной лабораторной работе и подготовить необходимые Tcl скрипты для работы с NS2, рассмотренные в данном разделе.

В качестве основы исследований использовать модель сети, разработанной в п.6 раздела «Лабораторное задание» работы №3 настоящего сборника. Доработать сценарий своего варианта моделирования таким образом, чтобы интенсивность трафика CBR источника составляла 1% пропускной способности линии связи, соединяющей узлы C и D., а длина очереди на этой линии ограничивалась 5 пакетами.

Изменить доработанный выше сценарий моделирования таким образом, чтобы в качестве источника трафика использовать не агент ftp, а агент telnet. Для этого все команды и параметры сценария, содержащие подстроку ftp заменить на подстроку telnet, а также по окончании описания агента telnet для установки метода формирования пакетов ввести в программу строку

$telnet set interval_ 0.

Дополнить разработанные сценарии инструкциями о формировании файлов трассировки для последующего анализа результатов моделирования. Техника получения файла трассировки

приведена в описании лабораторной работы №3 настоящего сборника.

Результатами предварительной подготовки по данной работе являются файлы Tcl скрипта, подготовленные для моделирования сетей с помощью имитатора NS2.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Какие задачи решает протокол TCP?
2. Сопоставить стек TCP и стек OSI.
3. Какие механизмы использует протокол TCP для надежной доставки данных?
4. Какая информация передается в полях «номера портов» и «номера последовательностей» заголовка TCP сегмента?
5. Какая информация передается в полях «флаги» заголовка TCP сегмента?
6. Какая информация передается в полях «размер окна» и «контрольная сумма» заголовка TCP-сегмента?
7. Порядок установления TCP соединения.
8. Как завершается TCP соединение в штатном режиме?
9. Как завершается TCP соединение в особых случаях?
10. Какие состояния можно выделить в процессе TCP соединения?
11. Особенности работы TCP с интерактивными данными.
12. В чем заключается алгоритм Нейгла?
13. Особенности передачи TCP большого объема данных.
14. Особенности реализации алгоритма «скользящее окно» в протоколе TCP.
15. Пояснить механизм «скользящего окна».
16. Из каких соображений выбирается размер окна?
17. В каких случаях в заголовке пакета устанавливается флаг «PUSH»? Как на это реагирует получатель?
18. В чем заключается алгоритм медленного старта?
19. С какой целью и как используется параметр «окно переполнения»?
20. Раскрыть понятие «сокет». В каких полях заголовка содержится информация о нем?
21. В чем заключается квитирование при передаче данных?

22. Особенности квитирования в протоколе TCP.
23. В чем особенности модели агента TCP в симуляторе NS?
24. С какой целью и как определяется время кругооборота?
25. Для какой цели и как в протоколе TCP используются таймеры?

ЛАБОРАТОРНОЕ ЗАДАНИЕ

1. Ознакомиться с основными характеристиками протокола TCP по материалам данного практикума и рекомендуемых в нем литературных источников, а также разделов 8 и 9 учебного пособия В.Г. Козлова, Е.С. Семигук «Программные средства систем связи», электронная версия которого размещена на сервере S ЛВС кафедры ТОР [S:\\ Библиотека кафедры ТОР \ Программные средства систем связи \]. Создать папку с названием Lab_4 на сервере X. Промежуточные и окончательные результаты данной работы сохранять в этом каталоге.

2. Программы, разработанные в результате предварительной подготовки, скопировать в созданную папку Lab_4.

3. Открыть окно «Терминал». С помощью команды «Изменить каталог» (cd *path*), где path – путь доступа к каталогу Lab_3, перейти в свой рабочий каталог.

4. Исследовать процедуру установления соединения. Для этого можно использовать любой сценарий, в котором участвуют агенты TCP. Для определенности использовать сценарий для исследования TCP, разработанный на этапе предварительной подготовки, модифицированный таким образом, чтобы источник CBR был отключен. Это можно выполнить, например, удалением строк программы

$ns at 0.1 "$cbr start"

$ns at 4.5 "$cbr stop",

либо переводя их в режим комментариев (вводом символа #). В отчете отобразить основные этапы установления соединения («троекратного рукопожатия») и отметить все установленные флаги, отсутствующие в модели NS.

5. Подготовить исследования характеристик TCP при передаче больших объемов данных с использованием агента прикладного уровня ftp. В качестве изменяемого параметра использовать интенсивность трафика источника CBR, причем началь-

ное значение интенсивности установить равным 1 % от пропускной способности линии связи, соединяющей узлы C и D, а максимальный размер очереди на этой линии установить равным 5. Время моделирования установить 5 с.

6. Исследовать алгоритм медленного пуска. Для этого запустить сценарий, разработанный на этапе предварительной подготовки. Наблюдать динамику количества пакетов, которые в пределах окна переполнения могут быть посланы до получения квитанции.

7. Дополнить разработанный сценарий инструкциями о формировании файла трассировки для получения возможности постпроцессорного анализа результатов моделирования. Полученный в результате моделирования этого сценария файл out.tr сохранить под (уникальным) другим именем.

8. Повторить исследования по п.п. 6 и 7 при интенсивности трафика источника CBR, равной соответственно 5 %, 20 %, 50 %, 80 %, 95%.

9. Повторить исследования по п.п. 6-8 для максимального размера очереди на линии, соединяющей узлы C и D, равного 10.

10. По результатам исследований по п.п. 6-9 построить графики динамики развития медленного пуска в зависимости от загруженности сети.

11. Исследовать зависимость времени кругооборота от загруженности сети. Время кругооборота измерять от момента отправки пакета до момента получения подтверждения. В таблицу занести данные 10 первых пакетов TCP в начале каждой секунды моделирования.:

Таблица 1

№ пакета	Момент отправления пакета	Момент получения подтверждения	Время кругооборота	Среднее время кругооборота	Среднее абсолютное отклонение

Среднее время кругооборота и среднее абсолютное отклонение времени кругооборота вычислять по первым 10 пакетам ка-

ждой секунды моделирования. Среднее абсолютное отклонение $\Delta t_{ср}$ времени кругооборота вычислять в соответствии с

$$\Delta t_{ср} = \frac{1}{10}\sum_{i=1}^{10}\left|t_i - t_{ср}\right|,$$

где t_i – i-езначение времени кругооборота, $t_{ср}$ - среднее время кругооборота по 10 пакетам.

12. Повторить исследования по п. 11 при интенсивности трафика источника CBR, равной соответственно 5 %, 20 %, 50 %, 80 %, 95%.

13. Повторить исследования по п.п. 11-12 для максимального размера очереди на линии, соединяющей узлы C и D, равного 10.

14. Подготовить исследования характеристик TCP при передаче небольших объемов данных с использованием агента прикладного уровня telnet. 12. Для этого повторить исследования в соответствии с п. 11 при интенсивности трафика источника CBR, равной соответственно 5 %, 20 %, 50 %, 80 %, 95%.

15. Изменить в сценарии параметр interval_ модели telnet на величину 0.5 и повторить исследования по п.14. Зафиксировать изменения. Повторить то же самое для параметра interval_ 0.1.

16. В отчет следует внести файлы программ, разработанные на этапе предварительной подготовки и в результате экспериментальных исследований, файлы, полученные по результатам анализа материалов моделирования сети, а также таблицы и графики экспериментальных данных.

ПРИЛОЖЕНИЕ
ВВЕДЕНИЕ В ЯЗЫК ПРОГРАММИРОВАНИЯ TCL-OTCL

Целью работы «Введение в язык программирования Tcl-OTcl» является получение первичных навыков работы с командным интерпретатором OTcl для предварительной подготовки к выполнению цикла работ по исследованию характеристик функционирования вычислительных сетей с помощью сетевого симулятора NS2.

ОБЩИЕ СВЕДЕНИЯ ПО ВЫПОЛНЕНИЮ РАБОТЫ

Общая часть этого цикла работ ориентирована на использование учебных материалов, размещенных на сайтах разработчиков сетевого симулятора NS2, в частности на сайте http://www.isi.edu/. Электронные копии некоторых материалов размещены в папке S:\БогомоловСИ\NS\Lab1\Metod\.

Первичные сведения о командном интерпретаторе представлены в документе «Введение в Tcl» (Tcl-OTcl.htm), размещенном в каталоге \Lab1\Metod\, а также в собственной справочной системе тренажера NS2.

Дополнительная информация работы с языком программирования Tcl представлена в программе-самоучителе для языка tcl «TclTutor», ссылки на которую размещены по адресу: http://www.msen.com/~clif/TclTutor.html.

Основное внимание следует уделить освоению приемов использования языка программирования Tcl а также подготовки, редактирования и отработки программ и представления результатов программирования.

Для формирования отчета следует использовать протокол сессии «Терминал», либо сохранять результаты моделирования в виде копии экрана путем последовательного переноса на документ графического редактора, например. KolourPaint {кнопка К (аналог кнопки ПУСК в среде Windows) / ГРАФИКА /}, с последующим выбором нужных сегментов экрана и переносом их в текстовый редактор, например, Write.

Лабораторные работы и отчеты по ним выполняются под управлением операционной системы Linux. Общие сведения о

системе Linux представлены на сервере S локальной сети кафедры ТОР и доступны по адресу system:/home/server_S/1_Курсы Linux. (из домашнего каталога пользователя). Более подробную информацию о каждой из команд системы Linux можно получить непосредственно из справочной службы операционной системы. Так, в режиме командной строки эту информацию можно получить, набрав в командной строке запросы вида:

$ info –h

либо

$ man -h,

где символ $ означает приглашение командной строки.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Какие типы данных могут быть использованы в Tcl?
2. Из каких компонентов состоит команда Tcl?
3. Как вносятся комментарии в команду Tcl?
4. Как используются символы пробела в командах Tcl?
5. Как используются символы пробела внутри параметров команд Tcl?
6. Как используются двойные кавычки в командах Tcl?
7. Как используются фигурные скобки в командах Tcl?
8. В чем отличия использования двойных кавычек и фигурных скобок в командах Tcl?
9. Как используются квадратные скобки в командах Tcl?
10. Как организуется подстановка команд в Tcl?
11. В каких случаях квадратные скобки не интерпретируются как команда?
12. Как выполняется подстановка переменных в Tcl?
13. В каких случаях не выполняется подстановка переменных в поле параметров?
14. Как выполняется в Tcl подстановка переменных в массивах?
15. Для какой цели в Tcl используется точка с запятой?
16. Для какой цели в Tcl используется обратная наклонная черта?
17. Что такое backslash-последовательность?
18. Перечислить команды вывода скобок
19. Перечислить команды вывода символов пробела, кавы-

чек и &.

20. Перечислить команды перемещения курсора.
21. Как представить в Tcl восьмеричное число?
22. Как выполняются математические операции в Tcl?
23. Как могут быть определены операнды при выполнении математических операций в Tcl?

ЛАБОРАТОРНОЕ ЗАДАНИЕ

1. Ознакомиться с языком программирования Tcl и его объектно-ориентированным расширением OTcl по материалам, представленным в каталоге \Lab1\Metod\, а также используя собственную службу помощи системы NS2. Создать папку с названием Lab_1 на сервере X. Промежуточные и окончательные результаты данной работы сохранять в этом каталоге.

2. Ознакомиться с содержанием файлов, размещенные в каталоге S:\БогомоловСИ\NS\Lab1\Test\, и скопировать их в созданную папку Lab_1. Файлы представляют собой фрагменты программ, иллюстрирующих работу командного интерпретатора Tcl. Изменить расширение в названии файлов txt на расширение tcl.

3. Исследовать содержимое документа Tcl-OTcl.htm («Введение в Tcl»), одновременно по мере изучения выполняя соответствующие учебные задания из папки Test. Для этого проделать операции по п.4… п.6.

4. Открыть окно «Терминал». С помощью команды «Изменить каталог» (cd *path*), где path – путь доступа к каталогу Lab_1, перейти в свой рабочий каталог.

5. Вызвать тренажер и выполнить подпрограмму, описание которой представлено в файле tcl_ex_1. Для этого набрать на клавиатуре команду:

ns tcl_ex_1.tcl.

Проанализировать результаты работы подпрограммы. В случае необходимости (если появляются сообщения об ошибках) внести в файл соответствующие изменения и вызвать команду повторно.

6. Выполнить операции по п.5 поочередно для файлов tcl_ex_2 … tcl_ex_10 каталога Test.

7. Открыть в программе «Текстовый редактор» файл ex-tcl.tcl

и исследовать его содержимое. Попытаться сформулировать математическое выражение, на основании которого составлена подпрограмма.

8. Отправить этот файл на выполнение симулятором:

ns ex-tcl.tcl.

По результатам работы подпрограммы внести изменения в математическое выражение (если появилась необходимость).

9. Доработать файл ex-tcl.tcl, т.е. внести в него комментарии для каждой команды и сохранить файл под именем ex-tcl_1.tcl.

10. Исследовать содержимое документа OTcl.htm («Введение в OTcl»), одновременно по мере изучения выполняя соответствующие учебные задания из этого же файла. Для этого сохранять фрагменты подпрограмм в соответствующих файлах с расширением tcl.

11. Отправить на выполнение файл ex-otcl.tcl:

ns ex-otcl.tcl.

По результатам работы подпрограммы внести в нее соответствующие комментарии и сохранить файл под именем ex-otcl_1.tcl.

12. Составить подпрограммы, в которых должны быть предусмотрены следующие компоненты:

Составить подпрограммы для вывода на экран своей фамилии, имени и отчества с использованием приема присвоения переменным значений и группировки параметров кавычками и скобками.

Разработать 2 подпрограммы с использованием backslash-последовательностей (печать символов, перемещение по тексту).

Составить подпрограммы для выполнения математических преобразований с использованием подстановки команд:

перемножить числа (день и месяц рождения), к результату прибавить корень квадратный из года рождения и вычесть квадрат номера факультета.

С использованием циклов вывести:

календарь с указанием дня недели для месяца, номер которого совпадает с номером студента в списке группы, для 2010 г.

13. В отчет следует внести протоколы работы с подпрограммами, а также доработанные с учетом добавления комментариев файлы подпрограмм.

ЛИТЕРАТУРА

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. С.-Петербург, изд-во «Питер».2007. - 957с. [40 экз.]

2. The ns Manual./Kevin Fall, Kannan Varadhan. Доступно по адресу: http://www.isi.edu/nsnam/ns/ns-documentation.html.

3. Tutorial for Network Simulator ns /Marc Greis. Доступно по адресу: http://www.isi.edu/nsnam/ns/tutorial/nsindex.html.