УДК 654.026

М.В. Абатурова, В.А. Бабошин, А.С. Гузарев

РАЗРАБОТКА ВИРТУАЛЬНОЙ МОДЕЛИ ДЛЯ ЭМУЛЯЦИИ НИЗКОСКОРОСТНЫХ И НЕСТАБИЛЬНЫХ КАНАЛОВ В СРЕДЕ VIRTUALBOX

Абатурова Марина Владимировна, *окончила Санкт-Петербургский государственный электротехнический университет «ЛЭТИ». Аспирант кафедры автоматики и процессов управления Санкт-Петербургского государственного электротехнического университета «ЛЭТИ». Инженер-программист ОАО «Интелтех». Имеет статьи в области моделирования систем связи, передачи мультимедийной информации по низкоскоростным каналам. [e-mail: voldemar24@yandex.ru].*

Бабошин Владимир Александрович*, окончил Ульяновское высшее военное командное училище связи им. Г.К. Орджоникидзе. Начальник отдела ОАО «НИИ «Рубин», кандидат технических наук, доцент. Имеет статьи в области систем связи специального назначения, систем беспроводного доступа, систем хранения данных. [e-mail: boboberst@mail.ru].*

Гузарев Антон Сергеевич, *окончил Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича. Аспирант кафедры автоматизации предприятий связи Санкт-Петербургского государственного университета телекоммуникаций им. проф. М.А. Бонч-Бруевича. Инженер ОАО «Интелтех». Имеет статьи в области моделирования систем связи, передачи мультимедийной информации по низкоскоростным каналам. [e-mail: audit.tss@mail.ru].*

Аннотация

В настоящей работе рассматривается разработка виртуальной модели в среде VirtualBox для эмуляции низкоскоростных и нестабильных каналов. Для решения данной задачи разрабатывается сетевой программный шлюз, позволяющий управлять изменениями в проходящем трафике для получения статистических данных.

Ключевые слова: мультимедийный трафик, виртуальная модель, низкоскоростные и нестабильные каналы, эмуляция.

Введение

Современный этап развития мировой цивилизации характеризуется переходом от индустриального к информационному обществу, предполагающему наличие новых форм социальной и экономической деятельности, которые базируются на

массовом использовании информационных и телекоммуникационных технологий и, следовательно, на предоставлении пользователям широкого спектра инфокоммуникационных услуг. Вследствие особенности данных услуг сети связи должны обладать следующими свойствами: мультисервисность, широкополосность, мультимедийность, интеллектуальность, инвариантность доступа и многооператорность. Современные мультисервисные сети представляют собой самостоятельный класс сетей, на базе которых может быть осуществлено предоставление широкого набора услуг, в том числе по передаче мультимедийной (многокомпонентной) информации (речь, данные, видео, аудио) [1, 2]. Мультимедийный трафик обладает существенно большими объемами по сравнению с трафиком речи и данных и предъявляет дополнительные требования к пропускной способности, времени и синхронизации передачи данных, что особенно актуально для сетей мобильной связи, включая сети беспроводного доступа. А это, в свою очередь, связано с ограниченной пропускной способностью радиоканалов, подверженных воздействию различных преднамеренных и непреднамеренных помех.

В связи с этим тема, рассматриваемая в данной статье, посвященной моделированию низкоскоростных нестабильных каналов для исследования вопросов адаптации мультимедийного трафика к передаче по «узким каналам», представляется актуальной.

Архитектура программного обеспечения среды моделирования

Современные технологии, используемые при моделировании, позволяют решать достаточно сложные задачи, обеспечивают имитацию сложных многоаспектных процессов, а также систем с большим количеством элементов. Отдельные функциональные зависимости в таких моделях могут описываться весьма громоздкими математическими соотношениями, что затрудняет их корректное математическое описание. Альтернативой этому является применение в задачах исследования сложных систем имитационного моделирования, в частности виртуальных моделей.

В данном случае в качестве инструмента моделирования используется платформа VirtualBox, имеющая модульную архитектуру с четко описанными компонентами и предоставляющая удобные интерфейсы доступа к процессам, эмулируемым виртуальными машинами (ВМ). Данное решение выбрано по причинам удобства использования и возможности интеграции в среды других операционных систем, а также в связи с широким рядом следующих возможностей:

– модульная архитектура с набором интерфейсов доступа к ВМ (GUI (Graphical User Interface), командная строка, удаленно);

– портирование на различные хостовые платформы;

– ВМ может действовать как сервер RDP (Remote Desktop Protocol) и управляться любым клиентом, поддерживающим протокол RDP с функцией USB over RDP;

– функция iSCSI initiator, позволяющая использовать внешние устройства хранения по протоколу iSCSI в качестве виртуальных дисков в гостевой системе без дополнительной поддержки со стороны хостовой ОС.

Программную основу составляет Linux-подобная операционная система с актуальным на данный момент ядром 3.2.0 [3]. Платформа VirtualBox исполняет код гостевой системы прямой передачей инструкций процессору хоста, при этом код, исполняющийся в нулевом кольце гостевой системы, исполняется в первом кольце процессора хоста, не используемом в архитектуре Intel, что снижает нагрузку на него.

Разработка виртуальной модели в среде VirtualBox

Графический интерфейс VirtualBox имеет два основных окна: главное и консоль ВМ. При старте ВМ VirtualBox обычно запускается несколько процессов, которые можно наблюдать в диспетчере задач (Windows) или системном мониторе (Linux):

1. Графический интерфейс окна управления. Процесс, запущенный с параметром startvm, означает, что GUI будет работать в качестве оболочки для ВМ.
2. Сервисный процесс VBoxSVC необходим для того, чтобы отслеживать количество и статусы ВМ, которые могут быть запущены различными способами.
3. ВМ с запущенной в ней гостевой ОС. Она инкапсулирует необходимые детали реализации гостевой ОС и является для хостовой системы обычным приложением.

Опуская промежуточные описания, приведем архитектуру программного обеспечения виртуальной модели и специально разработанного программного шлюза vchannel, за счет которого реализовано удаленное управление параметрами виртуальной модели по протоколу ssh [5].

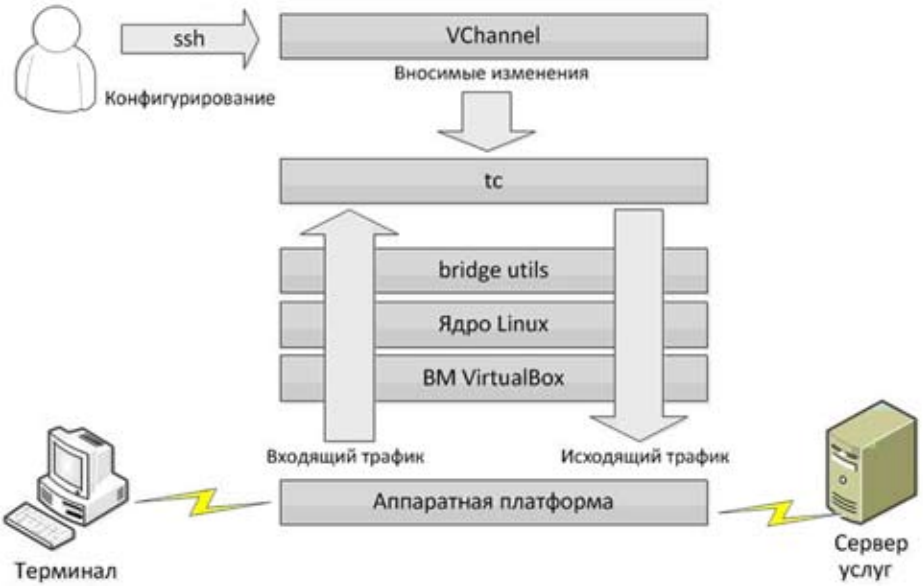

Рис. 1. Архитектура модели

ВМ поддерживает два сетевых интерфейса, физически подключенных к разным тестируемым объектам, в качестве объекта может выступать, например, участок между терминалом и сервером приложений или любой другой участок исследуемой сети. Манипулирование трафиком осуществляется при помощи программы bridge utils (Ver. 1.5.6), а изменение характера трафика выполняется программой tc [6–8]. Кроме того, в предлагаемой версии виртуальной модели предусмотрены процедуры непосредственного управления через интерфейс пользователя виртуального хоста (рис. 1).

Разработка шлюза, вносящего изменения в трафик [3], выполнена с использованием ВМ [6] с целью интеграции виртуальной модели в схему тестирования. Под ВМ понимается программная система, эмулирующая аппаратное обеспечение хост-платформы и исполняющая программы для гостевой платформы [6].

Функционал программы vchannel позволяет вносить следующие изменения [7]:

– задержки;
– ограничение полосы канала;
– перемешивание пакетов;
– потери пакетов;
– эмуляция воздействия аддитивных помех на пакеты данных.

Программирование канала выполняется с помощью конфигурационного файла и запуска специального скрипта. Конфигурационный файл имеет следующий синтаксис:

<rate|delay|loss|duplicate|corrupt|reorder><параметры>.

Эмуляция низкоскоростных радиоканалов выполняется за счет ограничения доступной полосы пропускания. Функция ограничения выполняется средствами виртуального адаптера VirtualBox, диапазон доступных скоростей которого ограничен полосой пропускания 1 Гб/с (для стандарта оптической высокоскоростной связи 10 GBASE).

Механизм очереди реализуется с помощью утилиты *TokenBucketFilter* (*TBF*), которая ограничивает скорость входящего пакетного трафика на уровне запрограммированного порога с возможностью коротких всплесков нагрузки, превышающих данный порог, что характерно для реальных сетей передачи данных [6].

Основным преимуществом механизма *TBF* является то, что он подразумевает создание буфера с токенами, выход которого сопоставляется с выходом очереди. При наличии в буфере свободного токена пакет отправляется в сеть, а токен покидает буфер. В противном случае, при отсутствии в буфере свободного токена, пакеты будут отбрасываться. При конфигурировании данной функции надо учитывать размер буфера: если он будет слишком мал, то все пакеты будут отбрасываться. Размер буфера должен удовлетворять следующему условию:

$$V_{TBF} \geq \frac{P}{800}, \quad (1)$$

где V_{TBF} – размер буфера с токенами;

P – верхний порог полосы пропускания в битах.

В примере, приведенном ниже, полоса пропускания составляет 512 Кбит/с и тогда в соответствии с (1) размер буфера будет равен 640 байтам:

rate512kbit buffer640. (2)

Эмуляция задержки пакетов позволяет определить суммарную задержку передачи данных всех устройств эмулируемого канала с учетом служебной информации. Для локальной сети время суммарной задержки можно определить по формуле:

$$t_{сум} = \sum t_i = t_{switch} + t_{route} + t_{firewall}, \quad (3)$$

где t_{switch} – задержка, вносимая коммутаторами;

t_{route} – задержка, вносимая маршрутизаторами;

$t_{firewall}$ – задержка, вносимая сетевыми экранами.

В случае использования радиоканала число элементов сети увеличится, а суммарные задержки в различном оборудовании можно определить по формуле:

$$t_{сум} = \sum t_i = t_{switch} + t_{route} + t_{firewall} + t_{ШАС} + t_{ППИ} + t_{modem} + t_{radio}, \quad (4)$$

где $t_{ШАС}$ – задержка, вносимая шифрующей аппаратурой;

$t_{ППИ}$ – задержка, вносимая приборами преобразования интерфейсов;

t_{modem} – задержка, вносимая модемами;

t_{radio} – задержка, вносимая радиоканалом.

Фактически данная функция добавляет к пакетам, находящимся в очереди, временной штамп. По достижении указанного времени пакет отправляется, а штамп извлекается из пакета. Следует заметить, что задержка на устройствах зависит и от количества пакетов в очереди, по этой причине задержка не является постоянной величиной. В данной виртуальной модели используется среднеквадратичное отклонение от постоянной величины, в приведенном примере задержка delay будет составлять 200 мс ± 30 мс: *delay 200ms 30ms*.

На практике время задержки может меняться с течением времени в достаточно широких пределах, поэтому кроме среднеквадратичного отклонения время задержки в очереди можно задать законом распределения: Гаусса (normal), Парето (pareto) и нормальное (paretonormal) [10, 11]. Пример реализации распределения Парето имеет вид:

delay 200ms 30ms distribution pareto. (5)

Эмуляция потерь пакетов в канале

Для эмуляции локальной сети можно задать процент потерь пакетов по отношению к переданным и дополнительно – коэффициент корреляции потерь в процентах: *loss 15 %, 20 %.*

Значение корреляции определяет приращение вероятности потери пакета в зависимости от потери предыдущего пакета [6, 7]. Данная функция позволяет учесть необходимость эмуляции пульсирующего трафика при резком изменении числа принимаемых пакетов [5].

В предлагаемой реализации для эмуляции радиоканалов используются стандартные статистические модели *state* или *gemodel*, которые основаны на математическом аппарате Марковских процессов. Марковский процесс – дискретный или непрерывный случайный процесс $X(t)$, который можно полностью задать с помощью двух величин: вероятности $p(x,\ t)$ того, что случайная величина $x(t)$ в момент времени t равна x, и вероятности $p(x_2,\ t_2|x_1,\ t_1)$ того, что если x при $t = t_1$ равен x_1, то при $t = t_2$ он равен x_2. Вторая из этих величин называется вероятностью перехода из состояния x_1 при $t = t_1$ в состояние x_2 при $t = t_2$.

Марковская цепь, применяемая в модели *state*, состоит из двух состояний: всплеск интенсивности передачи пакетов и отсутствие всплеска. Параметр p_{ij} характеризует переходные вероятности указанных состояний.

Модель *state* фактически представляет собой цепь случайных событий, вероятность появления которых зависит от текущего состояния события [12, 13].

Стандартные состояния модели *state*:

– пакет передан в интервал времени без всплесков;
– пакет передан во время всплеска передачи пакетов;
– пакет потерян во время всплеска передачи пакетов;
– пакет потерян в интервал времени без всплесков (рис. 2).

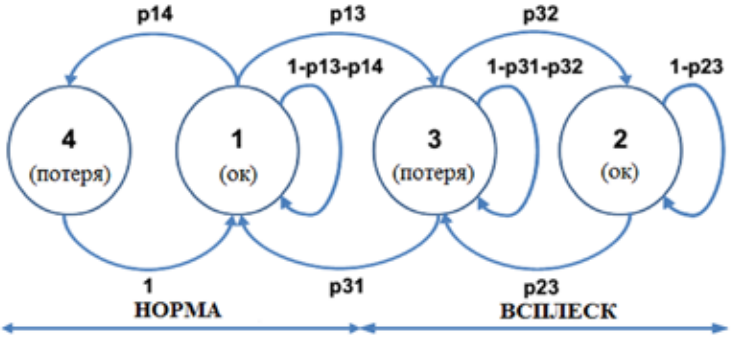

Рис. 2. Граф переходов Марковской цепи модели *state*

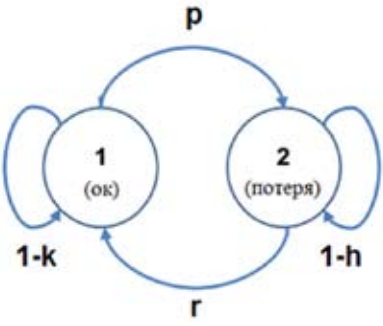

Рис. 3. Граф переходов Марковской цепи модели *gemodel*

В модели *gemodel*

p описывает вероятность перехода к событию потери пакета,

r – вероятность перехода к событию передачи пакета,

$1 - k$ – вероятность передачи пакета, а

$1 - h$ вероятность потери пакета (рис. 3) .

Ниже приведен синтаксис моделей *state* и *gemodel*:

*loss state p*13 [*p*31 [*p*32 [*p*23 [*p*14]]]], (6)

lossgemodel p [*r* [1−*h* [1−*k*]]].

Имеется возможность дублирования пакетов. Для задач эмулирования низкоскоростных и нестабильных каналов это требуется редко, но такая возможность реализована. Можно задать процент дублирования пакетов; в приведенном примере каждые 5 пакетов из 1000 будут дублироваться: *duplicate 0.5%*.

Эмуляция помех в канале

Данная процедура используется для эмуляции аддитивных помех в канале, когда полезная часть принимаемого сообщения смешана с шумом, вызванным внешними воздействиями на канал связи. Ошибка добавляется в произвольное место заданного процента принимаемых пакетов. Дополнительно можно добавить корреляцию ошибок в процентах: *corrupt 18.5%*, *10%,* значение которой определяет приращение вероятности добавления ошибки в пакет в зависимости от ошибки в предыдущем пакете.

Эмуляция перемешивания пакетов

В сетях с большими задержками часто возникает перемешивание последовательно идущих пакетов. Фактически происходит немедленная отправка для определенного числа пакетов, остальные пакеты отправляются с задержкой, заданной параметром *delay*.

В данном решении используется два метода перемешивания пакетов: метод gap и метод reorder. Метод gap перемешивает пакеты с заданным шагом, например каждый 10-й пакет, синтаксис имеет вид: *gap 10*. Метод gap будет полезен при отладке и тестировании исследуемых алгоритмов, для эмуляции более подходит метод reorder, позволяющий задать процент перемешивания пакетов и дополнительно указать коэффициент корреляции: *reorder 15 %, 60 %*. Следует обратить внимание на то, что для использования методов gap или *reorder* параметр *delay* не должен быть нулевым или незаданным.

Заключение

Разработанная программа vchannel выполняет эмуляцию низкоскоростного узкополосного канала с нестабильными характеристиками с возможностями управления параметрами задержек и потерь пакетов, помех и ширины полосы пропускания канала. Эта программа является составным элементом виртуальной модели в среде VirtualBox, что легко позволяет включать ее в необходимые точки сети по сетевому интерфейсу или интегрировать в другую виртуальную модель в ходе решения исследовательских задач.

Разработанная виртуальная модель может быть использована на этапе проектирования для оценки качества предоставления мультимедийных услуг, а также для проверки адаптации технологий действующих сетей к возможностям «узкого канала».

Ниже приведен исходный код основного модуля программы *vchannel*, представленного на языке bash (Bourne again shell).

```
#!/bin/bash
CONF_FILE=/usr/share/etc/vchannel.conf
RATE=$(grep -v «#» rate $CONF_FILE)
DELAY=$(grep -v «#» delay $CONF_FILE)
LOSS=$(grep -v «#» loss $CONF_FILE)
DUPLICATION=$(grep -v «#» duplicate $CONF_FILE)
CORRUPTION=$(grep -v «#» corrupt $CONF_FILE)
REORDERING=$(grep -v «#» –e reorder -egap $CONF_FILE)
if[ $# -eq 0 ]
then
echo “”
echo “Vchannel – утилита управления виртуальным низкоскоростным и нестабильным каналом”
echo “Параметры:”
echo “on включить виртуальный канал”
echo “off выключить виртуальный канал”
echo “status отобразить настройки виртуального канала”
echo “”
echo “Конфигурационный файл /usr/share/etc/vchannel.conf”
exit
fi
if[ $1 = “off” ]
then
echo “выключение виртуального канала...”
tcqdisc del dev eth0 root netem
tcqdisc del dev eth1 root netem
exit
fi
if [ $1 = «show» ]
then
tcqdisc show dev eth0
tcqdisc show dev eth1
exit
fi
if[ $1 = “on” ]
then
echo “включение виртуального канала...”
tcqdisc add dev eth0 root handle 1: tbf $RATE
tcqdisc add dev eth0 parent 1:1 handle 10: netem $DELAY $LOSS $DUPLICATION $CORRUPTION $REORDERING
tcqdisc add dev eth1 root handle 1: tbf $RATE
tcqdisc add dev eth1 parent 1:1 handle 10: netem $DELAY $LOSS $DUPLICATION $CORRUPTION $REORDERING
exit
fi
```

Выбор языка bash связан с тем, что он удовлетворяет стандарту POSIX (Portable Operating System Interface for Unix) и представляет собой усовершенствованную версию командной оболочки Bourne shell. Он поддерживает выполнение скриптов, автодополнение названий файлов и папок, подстановку вывода результата команд, переменные, контроль порядка выполнения, операторы ветвления и цикла [3, 14].

СПИСОК ЛИТЕРАТУРЫ

1. Руководящий технический материал «Принципы построения мультисервисных местных сетей электросвязи». Версия 2.0. – 2005. – 48 с.
2. Рекомендация МСЭ-Т Y.1541 (02/2006). Требования к сетевым показателям качества для служб, основанных на протоколе IP.
3. UNIX and Linux System Administration Handbook (4th Edition). Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley. – 2010.
4. Oracle VM VirtualBox. User Manual. OracleCorporation. 2013. – ULR: https://www.virtualbox.org.
5. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы (4-е издание). – Питер, 2010. – 916 с.
6. Ariane Keller. Packet Filtering and Netem. ETH Zurich. July 20, 2006.
7. Bert Hubert, Thomas Graf, Greg Maxwell, Remco van Mook, Martijn van Oosterhout, Paul B Schroeder, Jasper Spaans, Pedro Larroy. Linux Advanced Routing & Traffic Control HOWTO. 2012.
8. Salsano S., Ludovici F., Ordine A., Giannuzzi D. Definition of a general and intuitive loss model for packet networks and its implementation in the Netem module in the Linux kernel. University of Rome «Tor Vergata». Version 3.1, August, 2012.
9. Борисов В.И., Зинчук В.М. Помехозащищенность систем радиосвязи. Вероятностно-временной подход. – М. : РадиоСофт, 2009.
10. Артамонов Н. Теория вероятностей и математическая статистика. Углубленный курс. – М. : П МГИМО–Университет, 2008.
11. Шикин Е.В., Чхартишвили А.Г. Математические методы и модели в управлении. – М. : Дело, 2002. – 440 с.
12. Казаков В.А. Введение в теорию Марковских процессов. – М. : Советское радио, 1973. – 232 с.
13. Дынкин Е.Б. Марковские процессы. – М. : Физматгиз, 1963. – 432 с.
14. Мендель Купер. Искусство программирования на языке сценариев командной оболочки. Версия 2.8.7 (22 августа 2004 г.). – ULR: http://rus-linux.net/.