КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

УДК 004.414.23

Т. И. Алиев, В. В. Соснин, Д. Н. Шинкарук, М. Ю. Тихонов, Н. Г. Бурмакин

САПР МАРШРУТИЗИРУЕМОЙ КОМПЬЮТЕРНОЙ СЕТИ НА ОСНОВЕ КОМПОНЕНТОВ С ОТКРЫТЫМИ ИСХОДНЫМИ КОДАМИ

Рассматриваются проблемы разработки САПР маршрутизируемых компьютерных сетей, обеспечивающих diffServ-сервисы QoS, с использованием компонентов с открытым исходным кодом на примере систем NS-3 и Qt Framework.

Ключевые слова: *NS-3, QoS, Qt, имитационное моделирование, компьютерная сеть, туннелирование, фильтрация, дисциплины обслуживания, WFQ, PQ, CQ, LLQ.*

Введение. Имитационные модели компьютерных сетей используются для обучения [1], проектирования сетей в составе САПР [2] и исследования сетевых технологий [3]. Необходимость разработки САПР маршрутизируемой компьютерной сети связана с отсутствием в настоящее время готовых программных решений, которые удовлетворяют следующим требованиям:

— открытость исходного кода всех компонентов, что позволяет модифицировать базовые возможности САПР, реализуя сколь угодно сложную логику обработки пакетов;

— компоненты САПР должны иметь лицензии GPL, LGPL или BPL и быть работоспособны в системе Linux, что снижает общую стоимость САПР;

— использование языка C++, что позволяет реализовать в САПР поддержку новых сетевых технологий за счет применения существующих программных кодов стеков протоколов Linux (например, BSD sockets);

— при обработке пакетов должны быть реализованы сложные функции OSI-модели, которые отсутствуют в стандартных пакетах моделирования сетей;

— наличие кроссплатформенного графического пользовательского интерфейса (GUI), содержащего средства для анализа результатов моделирования с детальными сведениями о динамике изменения исследуемых характеристик.

Постановка задачи. Целью работы являлось создание САПР маршрутизируемой компьютерной сети на основе компонентов с открытым исходным кодом. Разрабатываемая САПР должна удовлетворять перечисленным выше требованиям. Для достижения поставленной цели решались следующие задачи:

— выбор средства разработки ядра САПР и его GUI с учетом того, что ядро должно обеспечивать возможность применения статистического моделирования при имитации работы компьютерной сети;

— организация программного взаимодействия между ядром САПР и GUI;

— выбор состава характеристик функционирования компьютерной сети, необходимых для решения задач анализа и синтеза проектных решений, и метода их обработки;

— программная реализация разработанных методов с использованием выбранных средств программирования.

Выбор средства разработки ядра САПР. В работе [4] предлагается для моделирования компьютерных сетей использовать коллекцию библиотек NS-3 (основанную на C++), которая позволяет реализовать большинство базовых функций сети передачи данных. Однако в NS-3 *отсутствуют* штатные средства для программной реализации следующих функций, технологий и требований:

— моделирование отказов линий связи или маршрутизаторов с корректной реакцией узлов сети на отказы, включая реконфигурацию маршрутных таблиц;

— реализация корректного функционирования IP-IP-туннелей;

— реализация фрагментации TCP-трафика в узлах сети;

— полноценное функционирование протокола OSPF, включая рассылку служебных сообщений и реконфигурацию графа сети;

— организация приоритетных очередей с правилами обслуживания PQ, WFQ, LLQ, RPQ+ и CQ [1] с возможностью гибкой классификации трафика;

— расчет по результатам моделирования следующих характеристик: среднее значение и вариация задержки передачи пакетов на каждом маршрутизаторе сети и на всем пути следования пакетов; доля потерь пакетов из-за переполнения буферов, фильтрации или помех в канале связи; размер окна и количество повторных передач для пакетов TCP-потоков.

Для преодоления этой проблемы потребовалась модификация базовых возможностей системы NS-3.

Выбор средства разработки GUI в первую очередь определялся наличием в среде разработки поддержки языка C++, поскольку это позволило использовать в ядре САПР и GUI одинаковые структуры данных в разделяемых файлах с исходными кодами. В таблице приведены наиболее известные кроссплатформенные сетевые симуляторы, имеющие графический пользовательский интерфейс, анализ которых позволил выбрать среду разработки Qt Framework. Эта среда обладает следующими преимуществами:

— известен ряд успешных реализаций сетевых симуляторов в Qt;

— стремительно развивается в OpenSource-среде на протяжении многих лет;

— стандарты Qt поддерживаются крупными компаниями и сообществами;

— является хорошо документированной системой;

— в Qt существуют развитые средства разработки (отладчики, профайлеры);

— изначально проектировалась как кроссплатформенный продукт.

Сетевой симулятор	Язык программирования	Используемая библиотека для создания GUI
GNS-3	Python	Qt
OMNET++ / OMNEST	C++, Tcl	Tk
GloboSim / QualNet	C++	Qt
SSFNet	Java	Swing, JGraph
J-Sim	Java	Swing, JGraph
PacketTracerCisco	C++	Qt
CNet	C, Tcl	Tk
STPSim	Python	wxWidgets
OpenWNS	C++, Python	Qt
NetAnimator	ActionScript	AdobeFlash
NetSim	Java	Swing, JGraph

Кроме приведенных в таблице сред разработки были рассмотрены GTK+, .NET Framework, Ultimate++, Motif, FoxToolkit. Все они в той или иной мере не обеспечивают выполнения перечисленных требований.

Взаимодействия ядра модели и GUI. Разработанная модель, структура которой приведена на рис. 1, состоит из трех исполняемых файлов, два из которых (Конфигуратор и Плеер) разработаны с помощью Qt, а третий (Ядро) — с помощью NS-3. Конфигуратор используется в качестве GUI для изменения параметров модели, Плеер используется в качестве GUI для анализа результатов моделирования. Стрелками показано направление движения входных/выходных данных каждого из исполняемых файлов. Например, созданный с помощью Конфигуратора файл описания модели поступает на вход Ядра, а созданный Ядром файл с историей моделирования поступает на вход в Плеер, в котором можно отследить изменение исследуемых характеристик с течением времени.

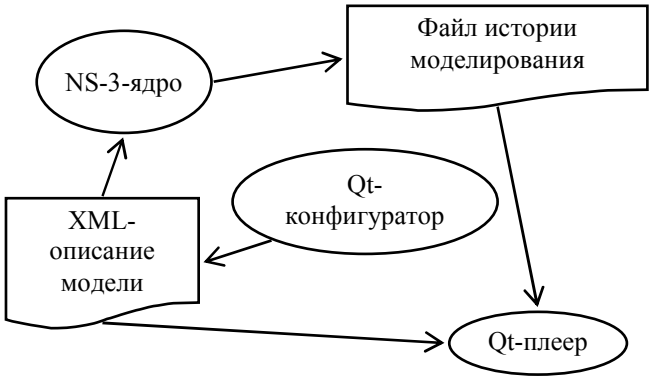

Рис. 1

В файл с историей моделирования записываются временные срезы состояния модели. Важной особенностью такого способа обмена данными между Ядром и Плеером является то, что данные передаются после их окончательного формирования, а не в режиме реального времени при работе имитационной модели. Такой подход обладает следующими достоинствами.

1. Модульность. Разработка Ядра, Конфигуратора и Плеера может выполняться независимо, с использованием описанных файлов как интерфейса между ними.

2. Обратная совместимость. Модель совместима с новыми версиями Qt и NS-3.

3. Возможность работы в демонстрационном режиме. Модель легко трансформировать для работы в этом режиме, для чего достаточно заготовить демонабор файлов с историей моделирования, которые предоставляются потенциальному покупателю со всеми компонентами модели, кроме Ядра.

4. Воспроизводимость. Историю моделирования можно неограниченное количество раз просматривать постфактум на компьютере любой производительности, что невозможно при анализе данных в процессе моделирования „на лету“ (как делается в большинстве имитационных пакетов), поскольку моделирование больших моделей является очень ресурсоемкой задачей.

5. Гибкость процесса анимации, т.е. возможность осуществлять его с любым разрешением по времени, что невыполнимо, если анимация производится одновременно с процессом моделирования.

Модификация базовых возможностей NS-3. Стандартные объекты NS-3 не позволяют реализовать все перечисленные высокоуровневые функции OSI-модели, поэтому при создании САПР на основе NS-3 требуется создавать собственные реализации сетевых устройств, интерфейсов и каналов связи. Из-за этого становится невозможным использование вспомогательных модулей NS-3 — Помощников (Helper), которые скрывают от программиста низкоуровневую работу с множеством объектов модели. Отказ от использования Помощников приводит к усложнению программирования, увеличению количества программного кода и снижению его удобочитаемости. Например, объем программы увеличивается более чем в 3 раза, если для конфигурирования канала связи не использовать Помощник.

Проблема фрагментации TCP-дейтаграмм. При реализации поддержки TCP-соединений между конечными узлами моделируемой сети было обнаружено, что в NS-3 все посылаемые через TCP-сокет пакеты дробятся на фрагменты размером не более 576 байт. Такое „несанкционированное“ дробление приводит к возникновению ошибок при дефрагментации пакетов в маршрутизаторах. Изучение исходных кодов реализации протокола TCP позволило понять, что разработчики NS-3 жестко придерживались стандарта RFC 791 (Internet Protocol [5]), где указана рекомендованная длина посылаемого узлом пакета. При фрагментации TCP-пакетов возникает и другая проблема. Если необходимость фрагментации связана с превышением MTU порта маршрутизатора, то дальнейшую обработку разбитых на фрагменты пакетов можно вести одним из следующих способов:

— присвоить пакетам виртуальные идентификаторы, которые будут хранить информацию о фрагментации;

— изменить алгоритм обработки пакета системой NS-3 в рамках стека протоколов TCP/IP, внеся правки в исходный программный код TCP/IP-стека;

— реализовать сборку фрагментированных пакетов в собственном модуле, который затем подключается в качестве промежуточного уровня TCP/IP-стека;

— использовать стандартные поля заголовка IP-уровня для сохранения информации о фрагментации.

Наилучшим из перечисленных способов является последний — хотя такая схема несколько усложняет фрагментацию и сборку пакетов, она совместима с протоколом [5] и может быть частично реализована встроенными методами NS-3, что упрощает поддержку и дальнейшее развитие модели.

Реализация TCP-сокетов стандартными средствами NS-3. При этом возможны два решения: устанавливать сокеты в рамках одного приложения ими рамках независимых приложений. Анализ показал, что для сбора детальной статистики временных задержек передачи пакетов лучшим является второе решение, так как в этом случае упрощается процедура анализа получаемых конечным узлом пакетов, а также возможно подключить несколько генераторов трафика к одному приемнику. Недостатком такого подхода является то, что в приемнике трафика потребуется установить большое количество приложений, что повысит затраты оперативной памяти при моделировании.

Реализация алгоритма поиска кратчайшего пути OSPF. В разработанной САПР компьютерных сетей маршрутизаторы связаны по схеме point-to-point (P2P), что существенно упрощает реализацию алгоритма маршрутизации. Поэтому возникла задача разработки упрощенной (модифицированной) версии протокола динамической маршрутизации OSPF, которая позволила бы получить те же качественные и количественные результаты, что и полная версия, описанная в стандарте RFC 2328 [6], но при меньших затратах вычислительных ресурсов. Для этого необходимо было выявить основные особенности работы OSPF в сетях point-to-point, найти в протоколе избыточные алгоритмы и неиспользуемые структуры данных в служебных пакетах. Работа протокола OSPF схематично представлена на рис. 2. Анализ показал, что без потери адекватности разрабатываемой версии протокола (по отношению к стандартной) могут быть опущены следующие части протокола OSPF v.2:

— все алгоритмы, связанные с выбором, проверкой работоспособности и функционированием „назначенного“ и „резервного назначенного“ маршрутизаторов (DesignatedRouter и BackupDesignatedRouter — DR и BDR);

— обмен hello-сообщениями и анонсами состояния связей (Link State Advertisement — LSA) по протоколу multicast.

Протокол OSPF содержит несколько типов служебных пакетов, наиболее длинным и сложным по структуре из которых является пакет обновления состояния связей типа 4 — Link State Update (LSU). Длина этого пакета примерно на порядок больше длины любого дру-

гого служебного пакета, поэтому при создании упрощенной версии OPSF имеет смысл модифицировать алгоритм обработки полей только пакетов LSU.

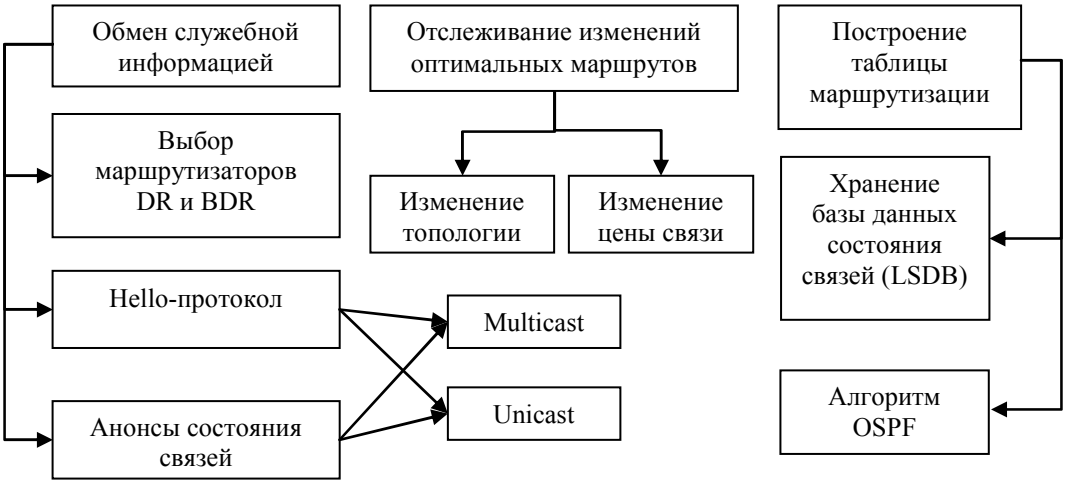

Рис. 2

Модификация протокола OSPF при обработке служебных пакетов позволяет увеличить скорость работы имитационной модели, поскольку дает возможность не рассматривать поля заголовка, которые не обязательны для корректного функционирования в условиях P2P-соединений между маршрутизаторами. Проведенный анализ стандарта [6] показал, что для получения кратчайших маршрутов достаточно следующих полей LSU-пакетов: тип сообщения (1 байт); длина сообщения (2 байта); идентификатор маршрутизатора (4 байта); номер последовательности (2 байта); количество анонсов (4 байта); количество портов (2 байта); адрес порта (4 байта); адрес соседа (4 байта); метрика (4 байта). Сокращение числа заполняемых и анализируемых полей с 27 до 9 позволило упростить протокол OSPF, удалив из него все алгоритмы, связанные с обработкой исключенных из рассмотрения полей. Для обеспечения адекватности модифицированного OSPF было решено пакет, отправляемый в канал, дополнять после перечисленных полей произвольной „набивкой“, равной по длине разнице в байтах между длиной пакета реализованного в модели протокола и стандартного. Таким образом, модифицированный OSPF, корректно рассчитывая маршрутную информацию, не влияет на временные задержки передачи LSU в каналах связи.

Анализ результатов моделирования. Важной особенностью модели является гибкая система сбора результатов ее работы, позволяющая пользователю индивидуально указывать классы трафика, характеристики которых требуется проанализировать. Это реализуется с помощью определения набора фильтров вида: „АИ, МИ, АН, МН, ПИ, ПН, Т, ВИП“, где АИ/МИ — IP-адрес и маска источника, АН/МН — IP-адрес и маска назначения, ПИ/ПН — порт источника/назначения, Т — тип сервиса в поле ToS (type of service) IP-пакета, ВИП — виртуальный идентификатор потока. Например, фильтр вида „192.168.1.2, 255.255.255.0, 192.168.2.3, 255.255.255.0, 25, 665, 3, 77“ из всего множества пакетов выделит только те, которые идут из подсети 192.168.1.0/24 в подсеть 192.168.2.0/24, при этом отправлены с порта 25 на порт 665 с ToS=3 и принадлежат виртуальному потоку № 77. В итоге возможно анализировать показатели качества обслуживания QoS только выбранного трафика. Смысл всех компонентов приведенного фильтра очевиден, кроме поля ВИП. Оно задает идентификатор потока, который присутствует в каждом пакете, но при этом не является частью ни одного из его физически передающихся полей. Использование этой метки позволяет отслеживать движение пакетов внутри туннелей, поскольку поле ВИП не подвергается изменению после любых операций туннелирования в отличие от всех остальных физически существующих полей пакета.

Заключение. Разработанная САПР компьютерной сети на основе компонентов с открытым исходным кодом позволяет решать задачи анализа и синтеза проектных решений при проектировании маршрутизируемых сетей, в которых обеспечивается функционирование QoS-сервисов. Предложенная модульная структурно-функциональная организация САПР позволяет обеспечить обратную совместимость модели с библиотеками Qt и NS-3, воспроизводимость результатов и высокое временно*е* разрешение процесса анимации, а также позволяет получать в качестве результатов характеристики любых классов трафика (включая туннелируемый трафик) на любых узлах моделируемой сети. В ходе создания САПР решен ряд задач моделирования протоколов TCP и OSPF в системе NS-3.

СПИСОК ЛИТЕРАТУРЫ

1. Инструменты для подготовки к сертификационным экзаменам Cisco [Электронный ресурс]: <http://www.cisco.com/en>.

2. Описание программы “NetCracker Network Management” [Электронный ресурс]: <http://netcracker.com/en/products/network_management>.

3. *Oliveira J., Vasseur J.P., Chen L., Scoglio C.* RFC4829. Label Switched Path (LSP) Preemption Policies for MPLS Traffic Engineering. San Diego: The IETF Trust, 2007 [Электронный ресурс]: <http://tools.ietf.org/rfc/rfc4829.txt>.

4. *Соснин В. В., Шинкарук Д. Н.* Моделирование маршрутизатора с поддержкой методов QoS в среде ns-3 // Сб. тр. молодых ученых и сотрудников кафедры ВТ. СПб: СПбГУ ИТМО, 2011. Вып. 2. С. 50—55.

5. *Postel J.* RFC791. Internet Protocol. California, 1981 [Электронный ресурс]: <http://tools.ietf.org/rfc/rfc0791.txt>.

6. *Moy J.* RFC2328. OSPF Version 2. Westford, 1998 [Электронный ресурс]: <http://tools.ietf.org/rfc/rfc2328.txt>.

Сведения об авторах

Тауфик Измайлович Алиев — д-р техн. наук, профессор; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; заведующий кафедрой; E-mail: aliev@d1.ifmo.ru

Владимир Валерьевич Соснин — Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; ассистент; E-mail: vsosnin@mail.ru

Дмитрий Николаевич Шинкарук — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: dimashink@gmail.com

Михаил Юрьевич Тихонов — студент; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: tihmihail@gmail.com

Никита Геннадьевич Бурмакин — аспирант; Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, кафедра вычислительной техники; E-mail: nekit.mail@gmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 08.02.12 г.