РЕАЛИЗАЦИЯ МЕТОДОВ ОБЕСПЕЧЕНИЯ КАЧЕСТВА ОБСЛУЖИВАНИЯ В СРЕДЕ МОДЕЛИРОВАНИЯ NS-3

Д.Н. Шинкарук, Н.Г. Бурмакин (Санкт-Петербург)

1. Введение

Имитационные модели компьютерных сетей используются для обучения [1], проектирования сетей в составе САПР [2], а также для исследования сетевых технологий [3]. Исследование характеристик функционирования маршрутизируемой компьютерной сети при возникновении перегрузок позволяет предупредить возможные проблемы после введения сети в эксплуатацию и сократить расходы на исправление возникающих при этом проблем.

Внедрение политик качества обслуживания (Quality of Service – QoS) позволяет определить, каким образом будут распределены сетевые ресурсы между потоками трафика различных приложений, предъявляющих собственные требования к ресурсам сети, в ситуации ограничения вычислительных возможностей маршрутизатора во время перегрузок.

2. Постановка задачи

В настоящее время для решения задач имитационного моделирования сетей передачи данных применяется широкий спектр программных средств:

наборы библиотек для распространенных языков программирования;

языки моделирования общего назначения;

узкоспециализированные среды моделирования.

Специализированные среды моделирования обладают более широкими возможностями в своей предметной области. В рамках выполняемых исследований авторами была выбрана система ns-3, так как она обладает следующим уникальным набором особенностей:

1. *Открытые исходные коды всех компонентов системы.* Это позволяет детально изучить логику работы системы ns-3 и упрощает модификацию ее базовых возможностей. Кроме того, появляется возможность использовать созданные модели в разработках, требующих полной прозрачности используемого программного обеспечения.

2. *Использование C++ в качестве базового языка для построения моделей.* Это обеспечивает высокую скорость работы модели (в сравнении, например, с системами, J-Sim и NetAnimator, основанными на «медленных» языках Java или Flash). Это является важным преимуществом, если требуется, чтобы модель работала в режиме реального времени, например, при использовании в качестве источника или приемника реального сетевого трафика.

3. *Распространение программных компонентов по лицензии GPL.* Это снижает общую стоимость модели, поскольку все компоненты ns-3 доступны бесплатно.

4. *Работоспособность в системе Linux.* Это также снижает общую стоимость владения моделью, так как не требует покупки дорогих лицензий на платные операционные системы.

5. *Максимальная схожесть модели с реальной сетью.* Внутри модели сети используются абстракции (сетевые интерфейсы, каналы связи, пакеты данных) максимально приближенные к реальным аналогам. Пакеты данных, используемые в модели ns-3, без каких-либо действий со стороны пользователя можно использовать в реальных сетях. Это позволяет использовать модель в качестве генератора или приемника сетевого трафика и интегрировать модель в работающую компьютерную сеть.

Однако в ns-3 отсутствуют встроенные средства для моделирования работы высокоуровневых функций маршрутизаторов, таких как обеспечение качества

обслуживания. Поэтому целью работы авторов стало создание такой ns-3-надстройки, которая позволила бы эффективно преодолеть эту проблему.

3. Расширение базовых возможностей маршрутизатора в ns-3

По умолчанию в системе ns-3 роль маршрутизатора выполняет тот же элемент, что используется при моделировании конечных узлов (класс Node). В качестве маршрутизатора Node выполняет лишь базовые функции сетевого и канального уровней (статическая маршрутизация, операции с заголовками канального уровня).

Для расширения базовых возможностей узла моделируемой сети, авторами был использован стандартный класс приложения (Application). Приложения ns-3 выполняют ту же роль, что и системные и пользовательские приложения в реальных компьютерных сетях [4]. В ns-3 осуществляется установка (подключение) приложений на узлы сети, после чего предоставляется доступ ко всем ресурсам узлов.

Было решено перенести функции классификации трафика и обеспечения качества обслуживания в приложение по аналогии с такими же функциями, выполняемыми ядром Linux в программных маршрутизаторах.

На рис. 1 представлена упрощенная схема реализованного маршрутизатора. С пакетов, попавших на один из N входных портов «Вх. порт *i*» маршрутизатора, снимаются заголовки канального уровня порта. Далее при помощи специальной *callback*-функции, созданной при установке приложения на узел маршрутизатора, пакет попадает на обработку в приложение, где подвергается следующим этапам обработки:

– анализ IP-заголовков пакета;
– фильтрация;
– маршрутизация;
– классификация;
– выполнение алгоритма QoS (помещение пакета в буфер одного из выходных портов «Вых. порт *i*»).

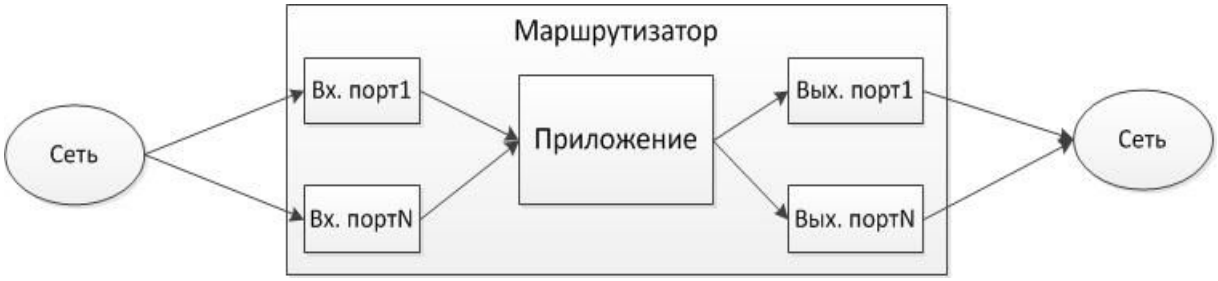

Рис. 1. Схема маршрутизатора

4. Параметризация алгоритмов QoS

В ходе реализации перечисленных выше функциональных возможностей приложения необходимо было создать максимально простой интерфейс, позволяющий пользователю сконфигурировать требуемый алгоритм QoS и при необходимости быстро переключиться на использование другого механизма.

Работу алгоритма распределения сетевых ресурсов можно условно разделить на два этапа: классификация поступивших в маршрутизатор пакетов и выбор пакета для передачи. На этапе классификации происходит анализ заголовков пакетов и их постановка в очередь соответствующего класса. Класс трафика образуется несколькими потоками или является частью одного потока. Поток представляет собой множество пакетов с одинаковыми адресами источника и назначения. В общем случае поток характеризуется также номером протокола транспортного уровня и номером портов источника и приемника. При необходимости классификация на четвертом уровне модели OSI может

учитывать эти параметры и будет конфигурироваться аналогичным образом. В данной модели классификация проводится на третьем уровне, для этого требуется проанализировать три поля заголовка IP: тип обслуживания, адрес источника, адрес назначения (ToS, SA, DA).

В ходе программной реализации алгоритмов распределения ресурсов было выявлено несколько общих свойств, позволивших унифицировать конфигурацию механизмов QoS. Алгоритмы организуют очередь пакетов для каждого класса и хранят в памяти одномерный массив с вычисляемыми после передачи пакета специальными численными значениями. Различие состоит только в алгоритме вычисления этих чисел и в их интерпретации при выборе следующего пакета для передачи. При конфигурации механизма QoS каждому классу трафика ставится в соответствие один параметр. Параметром класса трафика является приоритет для PQ, счетчик байтов для CQ и пропускная способность для WFQ. Все параметры обладают важным свойством: значение каждого параметра пропорционально ширине полосы пропускания, выделяемой данному классу трафика.

Таким образом, был определен следующий набор параметров, достаточный для унифицированной конфигурации механизмов распределения полосы пропускания канала передачи данных:

1. диапазон адресов источника;
2. диапазон адресов назначения;
3. поле Type of Service;
4. качество обслуживания;
5. вес класса.

Параметры 1–3 задают класс трафика, с помощью параметров 4–5 происходит инициализация механизмов QoS. Отдельного внимания заслуживает вес класса: его интерпретация происходит в зависимости от используемого алгоритма QoS. Такое решение возможно благодаря тому, что у любого из рассматриваемых алгоритмов используется только один параметр для каждого класса трафика. Благодаря такому подходу конфигурация алгоритмов QoS может быть полностью унифицирована – используется одинаковое количество параметров с одинаковыми именами.

5. Сравнение модели с существующими реализациями

Как было сказано выше, реализация функций маршрутизатора в виде приложения, устанавливаемого на узел моделируемой сети, была продиктована аналогией с программными маршрутизаторами, основанными на ОС Linux. Однако существуют также другие схемы реализации функций обеспечения качества обслуживания. Например, в маршрутизаторах фирмы Cisco некоторые функции по обработке трафика возложены на устройства сетевых адаптеров, что на рис. 1 соответствует входным и выходным портам маршрутизатора. В этом случае ядро маршрутизатора – приложение, значительно упрощается, также на него снижается нагрузка.

Благодаря открытости и гибкости программного обеспечения, основанного на ns-3, создание моделей систем, аналогичных маршрутизаторам Cisco не требует больших усилий. Основной проблемой в данном случае является закрытость архитектурных особенностей коммерческих маршрутизаторов.

Стоит отдельно отметить удобство и простоту настройки дисциплин обслуживания в описываемой модели маршрутизируемой сети. Например, для конфигурации механизма WFQ в Cisco IOS для одного класса требуется выполнить следующую последовательность действий в командной строке:

1. определить класс трафика с помощью списка доступа;
2. список доступа связать с картой класса;

3. карту класса связать с картой политики;
4. в рамках карты политики указать нужную данному классу полосу пропускания;
5. карту политики связать с выходным интерфейсом.

Предложенный в данной работе способ конфигурации содержит только нужные для работы алгоритма параметры, он лаконичен и интуитивно понятен.

6. Выводы

В результате работы был расширен функционал базовой модели маршрутизатора в системе ns-3, что позволило получить модель маршрутизатора превышающего по возможностям современные открытые программные маршрутизаторы. Также был разработан унифицированный метод параметризации дисциплин обслуживания, позволяющий гибко и быстро настраивать нужный механизм распределения сетевых ресурсов, а также при необходимости легко переключиться на использование другого алгоритма для данного класса трафика.

Литература

1. Описание имитационной модели 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. de, Vasseur J.P., Chen L., Scoglio C.** RFC4829. Label Switched Path (LSP) Preemption Policies for MPLS Traffic Engineering. San Diego: The IETF Trust, 200
4. Linux Advanced Routing & Traffic Control [Электронный ресурс]: <http://www.lartc.org/lartc.html>