Информатика

УДК 004.724:004.057.4

Подсистема маршрутизации Click

Д. С. Кулябов, А. В. Королькова, А. А. Хохлов

Кафедра систем телекоммуникаций
Российский университет дружбы народов
Россия, 117198, Москва, ул. Миклухо-Маклая, 6

В статье описывается программный комплекс Click Modular Router, разработанный в Массачусетском технологическом университете для построения быстрых, гибких и легко управляемых программных маршрутизаторов с поддержкой технологии QoS.

Ключевые слова: routing, QoS, router, Click Modular Router, Click, Linux, Device Polling.

Введение

С момента разработки стека протоколов TCP/IP неотъемлемой частью любой компьютерной сети стали маршрутизаторы, пересылающие пакеты между хостами. Наряду со специализированными компьютерами-маршрутизаторами, которые занимались только маршрутизацией, появилось много модулей для различных операционных систем, которые обеспечивали функциональность обычного компьютера в качестве маршрутизатора.

Разработчики каждой из сетевых операционных систем старались внедрять в свои продукты новые технологии, чтобы их ОС «шли в ногу со временем». Изначально, как в BSD UNIX, так и в Linux системах, вся статическая маршрутизация была представлена лишь простым перенаправлением пакетов между интерфейсами согласно таблице, а динамическая — сторонними программами gated и routed. Через небольшой промежуток времени требования к маршрутизаторам возросли, от них стало требоваться не только уметь пересылать трафик через свои интерфейсы, но и делать множество других вещей — исполнять функции брэндмауэров (firewall) практически на всех уровнях TCP/IP, кроме аппаратного, подсчитывать трафик, составлять статистику, выступать в роли proxy-серверов, ограничивать трафик от одних приложений либо адресов, и, наоборот, выделять больший приоритет трафику от других, осуществлять трансляцию сетевых адресов и их маскарадинг. Так, маршрутизаторы стали не только перенаправлять трафик, а еще и изменять его, маркировать, классифицировать, сортировать, отбрасывать ненужный трафик — то есть управлять трафиком и обрабатывать его.

Естественно, что в развитых сетевых операционных системах — Linux и BSD Unix (например — FreeBSD) стала активно внедряться поддержка всех этих технологий.

Разработчики ядра Linux, начиная с ядер 2.2, ввели в систему программный пакет ipchains (цепочки IP), который впоследствии заменили на пакет iptables (таблицы IP) и систему IPRoute версий 1 и 2, работающие с модулем ядра netfilter и другими модулями. IPRoute 2 совместно с iptables (это оболочка к системе фильтрации трафика ядра) явили собой воистину выдающуюся связку, обладающую практически неограниченными возможностями в обработке трафика. Вот лишь некоторые возможности Linux в области обработки трафика [1]: управление полосой пропускания, разделение и приоретизация трафика, поддержка большого количества механизмов QoS, фильтрация, NAT, firewalling, туннелирование и много другое.

К сожалению, пакет IPRoute2, непрерывно развиваясь и разрастаясь, становится все более трудно управляемым.

Разработчики FreeBSD решили пойти другим путем [2,3]. В 1996 году программисты компании Whistle InterJet Арчи Коббс (Archie Cobbs) и Джулиан Элисчер (Julian Elischer) встали перед задачей реализовать многопротокольный маршрутизатор на базе операционной системы FreeBSD. По некоторым причинам их не устроили уже существующие продукты, и они решили написать свою библиотеку. Ее назвали NetGraph. NetGrpah не имеет недостатков, присущих продуктам — предшественникам, потому что ее изначально проектировали как систему, решающую множество задач, состоящую из модулей. Новые модули могут добавлять любые программисты, поэтому система масштабируема. Ее довольно простое устройство позволяет сторонним людям легко в ней разбираться и улучшать по необходимости.

NetGraph — это модульная сетевая подсистема, которая может использоваться для дополнения существующей сетевой инфраструктуры ядра. Многие существующие работающие модули поставляются с FreeBSD и включают поддержку практически всех сетевых стандартов.

NetGraph понятна для понимания, потому что конфигурация маршрутизатора описывается в виде ориентированного графа, по ребрам которого передается трафик. Это позволяет с легкостью изменять и масштабировать конфигурацию.

Недостатком NetGraph явилось опять же малое количество документации и отсутствие поддержки QoS. Опять же, в последнее время системы Linux получают гораздо большее распространение, чем BSD, и большее количество разработок ведется в этом направлении.

Одной из таких разработок стал программный комплекс Click Modular Router.

1. Описание Click

1.1. История возникновения

Click был разработан в Массачусетском технологическом университете США при поддержке национального агенства DARPA. Впервые авторы (Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, M. Frans Kaashoek) заявили о нем в декабре 1999 года на 17 симпозиуме об архитектуре операционных систем [4]. Click был представлен как гибкая, идеально масштабируемая, быстрая модульная система для построения любых, сколь угодно сложных маршрутизаторов. С тех пор система развивалась, дополнялась новыми модулями (на текущий момент она поддерживает работу с практически любыми сетевыми протоколами и стандартами) и поддерживалась. Обновление ее производится через CVS.

1.2. Идеология Click

В основу идеологии Click положены гибкость, масштабируемость, высокая скорость работы, легкость в управлении и возможность самостоятельно изменять систему.

Гибкость, масштабируемость и легкость в понимании достигаются тем, что конфигурации Click представлены в виде ориентированного графа. Возможность любому программисту изменять и дополнять систему предоставляется в силу того, что Click написан на C++, и в документации есть шаблоны классов, на основе которых можно писать новые компоненты системы.

Высокая скорость обработки трафика достигается за счет механизмов Device Polling и внутреннего механизма аннотаций.

Device Polling [5] — это новая технология работы ядра Linux с устройствами, которая позволяет ускорять обработку пакетов в несколько раз.

Click может заменить всю стандартную маршрутизацию Linux. Эмпирически доказано [4], что скорость его работы больше, хоть и немного, скорости работы стандартного ядра.

Например, на компьютере с процессором Intel Pentium III 700 MHz с двумя сетевыми картами DEC Tulip 100 Mbit/s, операционная система — Linux, ядро —

2.2.4, производительность маршрутизатора, построенного на базе Click Modular Router, по заявлению разработчиков, составила 144000 64-байтных IP пакетов в секунду [4]. Для сравнения, производительность маршрутизатора Cisco 3750 в режиме Fast Switching может доходить лишь до 120000 64-байтных IP пакетов в секунду.

1.3. Архитектура Click

Компоненты системы, которые обрабатывают пакеты, называются ***обработчиками***. Основной код обработчика довольно мал, он включает в себя, в основном, только функции инициализации и передачи пакета, однако может быть дополнен дополнительными функциями, например, передачей размера очереди пакетов.

Для построения маршрутизатора пользователь должен выбрать набор обработчиков и соединить их в ориентированный граф. Ребра графа называются ***соединениями***. Они представляют собой возможные пути следования пакета. Граф является ориентированным по причине того, что пакеты следуют через маршрутизатор в определенном направлении.

Для повышения функциональности пользователи могут создавать свои собственные обработчики, а также модифицировать должным образом имеющиеся. Обработчики можно комбинировать между собой любым образом при помощи каналов (pipes) операционной системы Unix.

Передача пакета по соединению может быть инициирована как источником (push processing), так и пунктом назначения (pull processing). Это повторяет большинство алгоритмов передачи пакетов маршрутизаторами и позволяет создавать полноценные планировщики передачи пакетов.

Обработчик Click представляет собой некий элемент, в котором происходит обработка пакета.

Каждое действие, произведенное программным комплексом Click Router, инкапсулировано в обработчик, начиная от аппаратной обработки и заканчивая подсчетом количества пакетов.

Внутри рабочей конфигурации каждый обработчик представлен объектом C++, соединения реализованы при помощи указателей на объекты, а прохождение пакета представлено вызовом виртуальных функций C++.

Ниже представлены основные свойства обработчика:

- ***класс обработчика:*** принадлежность к определенному классу определяет код, который работает, когда на обработчик попадает пакет;
- ***порты:*** каждый элемент может иметь определенное количество входных и выходных портов. Соединения между обработчиками создаются «между их портами» - между исходящим портом одного обработчика и входящим другого. Два соединения между одними и теми же обработчиками, но разными их портами могут иметь абсолютно разную семантику;
- ***строка инициализации:*** определяет состояние обработчика при первоначальном запуске программного комплекса;
- ***интерфейсы***: каждый обработчик должен иметь как минимум интерфейс для приема и передачи пакетов, однако помимо у обработчика может существовать некоторое количество дополнительных интерфейсов, при помощи которых обработчики обмениваются информацией, полученной в результате исполнения их методов.

На рис. 1 представлен простейший обработчик ***Tee (2)***, принимающий пакеты на единственный входящий порт и копирующий их на два исходящих. Tee — это класс обработчика. Здесь и далее треугольником обозначается входящий порт, а прямоугольником — исходящий. Простейший обработчик представлен на рис. 1.

Цифра 2 в строке инициализации означает запрос на два исходящих порта. В данном примере интерфейсов методов не имеется.

На рис. 2 показана конфигурация, в которой несколько элементов объединены в простейший маршрутизатор, который считает входящие пакеты и отбрасывает их.

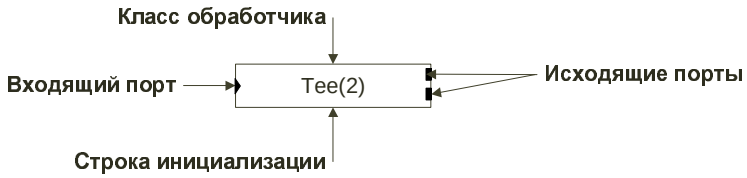

Рис. 1. Простейший обработчик Click.

Рис. 2. Простейший маршрутизатор.

1.4. Push и pull соединения

Теперь рассмотрим два механизма, которые поддерживает Click — передачу пакета по требованию источника (push processing) и по требованию адресата (pull processing). Push–соединения, при которых пакет посылается по инициативе источника, похожи на методы, которые используют все стандартные маршрутизаторы. При Pull–соединениях, наоборот, обработчик, которому пакет предназначается, посылает запрос на источник с тем, чтобы тот вернул ему пакет или нулевой указатель в том случае, если пакета нет. Оба этих способа реализованы при помощи виртуальных функций C++.

Следует заметить, что любой порт обработчика может быть либо push, либо pull типа. Типы портов определяются заранее, при инициализации маршрутизатора. Между портами разных типов соединение не может быть установлено. Существует специальный тип портов, которые, в зависимости от типа порта, с которым планируется взаимодействие, могут принимать разные типы.

Push–соединения используются, например, при поступлении пакетов на сетевую карту. После поступления каждого пакета он передается далее. Pull–соединения можно использовать, например, в том случае, когда для передачи пакета в сеть надо подождать освобождения передающего устройства, которое, как только освободится, инициирует Pull–соединение и передаст пакет.

1.5. Очереди пакетов

Обработчики Click не имеют встроенных буферов для построения очередей необработанных пакетов, однако такие очереди присутствуют в системе в явном виде. Они представлены специальными Queue–обработчиками, которые имеют push — входящий порт и pull — исходящий. Работают по алгоритму FIFO. Строкой инициализации является вместимость обработчика-очереди.

1.6. Механизм поиска обработчиков на основе потоков пакетов

Если обработчику A потребуется использовать обработчик B, он должен сначала найти B. Эта проблема не решается соединениями, так как между A и B может быть не установлено никаких соединений. Вопрос решается следующим образом: либо A ищет B по имени (например, ссылка на B имеется в его строке инициализации), либо использует автоматический механизм, названный механизмом поиска обработчиков на основе потоков пакетов, который на основе аннотаций пакетов определяет местоположение обработчиков в конфигурации.

Как раз этот механизм является одной из причин, по которой Click работает быстрее, чем IPRoute 2.

1.7. Язык конфигурации

Конфигурации системы Click Modular Router пишутся на простом языке, основами которого являются ***декларации*** (объявления) элементов, которые создают элементы, и ***соединения***, которые описывают, каким образом обработчики должны быть соединены. Разработчики называют его C – подобным.

Этот язык является декларативным языком — то есть он описывает, какие обработчики надо создать и как их соединить, но не описывает, каким образом они работают. Это делает конфигурации Click понятными и легко изменяемыми. Простой пример:

frometh0 :: FromDevice(eth0); toeth1 :: ToDevice(eth1); frometh0 -> toeth1;

Как видно, обработчики объявляются при помощи оператора «::», а соединяются — при помощи «->». Этот язык очень легок для усвоения, достаточно часа, чтобы научиться с ним работать в полной мере.

1.8. Способы взаимодействия с операционной системой

Click может работать либо на уровне ядра, для чего ядро придется перекомпилировать, наложив перед этим на его исходные коды специальный патч, либо на пользовательском уровне и взаимодействовать с сетью либо при помощи какого-либо пакетного фильтра, либо при помощи какого-нибудь другого механизма. Работа на пользовательском уровне больше подходит для настройки, написания и отладки конфигураций, в то время как работа на уровне ядра больше подходит для реальных задач, в силу того, что на уровне ядра скорость работы гораздо больше.

Для работы на уровне ядра существуют механизмы для «горячей» замены конфигураций «на лету».

1.9. Реализация обработчиков

Каждый класс обработчика Click соответствует подклассу класса *Element* C++, который имеет примерно 20 виртуальных функций. *Element* включает в себя базовые реализации большинства из них, так что подклассам требуется переопределить 6 или менее виртуальных функций. Всего три функции используются в процессе работы маршрутизатора — push, pull и run_sheduled. Все остальные нужны для идентификации, инициализации, статистики и др.

2. IP маршрутизатор

В этом разделе представлена схематическая модель простейшего маршрутизатора, который может перенаправлять пакеты между двумя сетевыми интерфейсами. Работает он следующим образом.

Обработчик *FromDevice* со строкой инициализации *«eth0»* принимает пакет от интерфейса *eth0* и помещает его в обработчик *Classifier,* который разделяет трафик на ARP–запросы, ARP–ответы и IP–трафик.

ARP–запросы будут передаваться обработчику *ARPResponder*, который будет отвечать на них. Строка инициализации у этого обработчика будет следующая: *ARPResponder(IP_адрес_интерфейса_eth0, MAC_адрес_интерфейса_eth0).* ARP–ответы, которые приходят на интерфейс, очевидно, были получены в результате неких запросов. Их создавал обработчик *ARPQuerier*, ему они и направляются.

IP–пакеты, отфильтрованные обработчиком *Classifier*, направляются обработчику *Paint(1),* который «метит» пакет параметром, заданным в строке инициализации. Это может быть целое число от 1 до 255. Оно означает сетевой интерфейс, откуда пришел пакет (для второго интерфейса будет вызываться обработчик *Paint(2)*). Теперь, при помощи механизма аннотаций, любой обработчик будет знать, с какого интерфейса пришел пакет. Далее пакет поступает обработчику *Strip*, который «отрезает» 14 байт от начала пакета (это указано в его строке

инициализации «14»), эти 14 байт являются ненужным для IP маршрутизации Ethernet–заголовком.

Далее IP–пакет поступает на обработчик *CheckIPHeader*, который проверяет пакет на правильность IP–адреса получателя, источника, вычисляет контрольную сумму пакета и сравнивает ее с той, что имеется в заголовке. Если все проверки проходят успешно, пакет передается на выходной интерфейс неизмененным, если же что-то не так, пакет отбрасывается.

Далее пакет попадает на обработчик *GetIPAddress*, который копирует IP–адрес назначения из заголовка пакета в его аннотацию. Перенаправляет пакет неизменным. Потом пакет попадает в обработчик *LookupIPRoute*, который, имеет в качестве строки инициализации маршрутную таблицу. Этот обработчик начинает собой ветвление дерева, посылая пакет через ряд других обработчиков на первый или второй интерфейс, либо в TCP/IP стек Linux.

Рассмотрим одну из ветвей. После обработчика *LookupIPRoute* пакет попадает в обработчик *DropBroadcasts*, который на основании аннотации-флага, указывающего на широковещательные сообщения для физического соединения, с которого пришел пакет, установленного обработчиком *FromDevice*, отбрасывает эти широковещательные пакеты. Далее обработчиком *CheckPaint* проверяется аннотация, в которой указано, с какого интерфейса пришел пакет. Если оказывается так, что пакет идет в тот же интерфейс, откуда он пришел, то это не совсем корректная ситуация, генерируется ICMP ошибка.

Далее пакет поступает на обработчик *IPGWOptions*, который обрабатывает параметры Record Route и Timestamp.red blackЕсли blackRecordblack blackRouteblack включен, маршрутизатор «вписывает» себя в список пройденных маршрутизаторов. Затем меняется параметр blackTimestampblack, который нужен для измерения времени передачи пакета от одного маршрутизатора к другому. В случае каких-либо ошибок генерируются ICMP–сообщения. Далее пакет передается обработчику *blackFixIpSrc*black со строкой инициализации, которая представляет собой blackIPblack–адрес интерфейса, через который отправится пакет.

Этот обработчик, при условии, что аннотация-флаг Fix Ip Source включена, исправит IP–адрес источника на IP–адрес этого интерфейса. Этот флаг устанавливает обработчик *ICMPError*. Дело в том, что ICMP–ошибки должны приходить именно от этого маршрутизатора, а обработчик *ICMPError* не может предугадать, через какой интерфейс уйдет пакет в сеть, поэтому он просто включает аннотацию-флаг Fix Ip Source, а уже после того, как обработчик *LookupIpRoute* определит интерфейс, через который этот пакет уйдет, FixIpSrc исправит IP–адрес источника на IP–адрес этого интерфейса. Далее пакет передается обработчику *DecIPTTL*, который убавляет на 1 поле TTL заголовка пакета, и, если полученное число не стало равным 0, отправляет пакет дальше, а если TTL истекло, передает пакет обработчику *ICMPError* со строкой инициализации *«11»*, что означает ICMP–ошибку «TTL Expired», который генерирует аналогичную ошибку. Пакеты, которые успешно проходят уменьшение TTL, попадают на обработчик *IPFragmenter* со строкой инициализации *«1500»*. 1500 — это стандартный MTU для IP–сетей. Пакеты, меньшие, чем 1500 байт, обработчик пропускает беспрепятственно сразу. Те, которые больше — он делит на фрагменты, меньшие 1500 байт, формирует из этих фрагментов новые IP–пакеты и передает их на выходной интерфейс. Однако, если входной пакет больше, чем 1500 байт, а в заголовке установлен флаг don't–fragment, обработчик через второй выходной интерфейс посылает пакет обработчику *ICMPError* со строкой инициализации *«3,4»*, что означает ICMP–сообщение «необходима фрагментация», который, в свою очередь, генерирует ICMP–пакет с аналогичной ошибкой.

После обработчика *IPFragmenter* мы получаем готовые к отправке по канальному уровню пакеты. Они поступают в обработчик *ARPQuerier*, строкой инициализации которого является IP–адрес интерфейса. Этот обработчик, во-первых, принимает ARP–ответы из обработчика *Classifier*, рассмотренного выше. Во-вторых, он инкапсулирует полученные IP–пакеты в Ethernet–кадры и отправляет их в очередь на отправку на сетевой интерфейс. Так же этот обработчик посылает в сеть ARP–запросы для распознавания MAC — IP необходимых адресов. Как было сказано выше, обработчик *ARPQuerier* при помощи push–соединения

«кладет» пакеты в очередь *Queue* со строкой инициализации, являющейся целым числом, которое является просто размером этой очереди. В данном примере очередь представляет собой структуру FCFS статического размера. Из очереди при помощи pull–соединения кадры «вытаскивает» обработчик ToDevice со строкой инициализации, являющейся именем сетевого интерфейса в системе, и отправляет их в сеть по мере ее освобождения. Этот обработчик работает на уровне протоколов MAC/LLC. Следует заметить, выше было рассмотрено 4 случая, когда пакет уходил в *ICMPError* обработчики, которые, в зависимости от ситуации, генерировали разные пакеты с ICMP–ошибками. Все эти пакеты передаются на вход обработчику *LookupIPRoute*, и он их направляет в нужные интерфейсы. Ниже представлен рисунок, который наглядно демонстрирует маршрутизатор, представляя его в виде связного графа, узлами которого являются обработчики, а ребрами — соединения. Схематическое изображение этого маршрутизатора представлено на рис. 3.

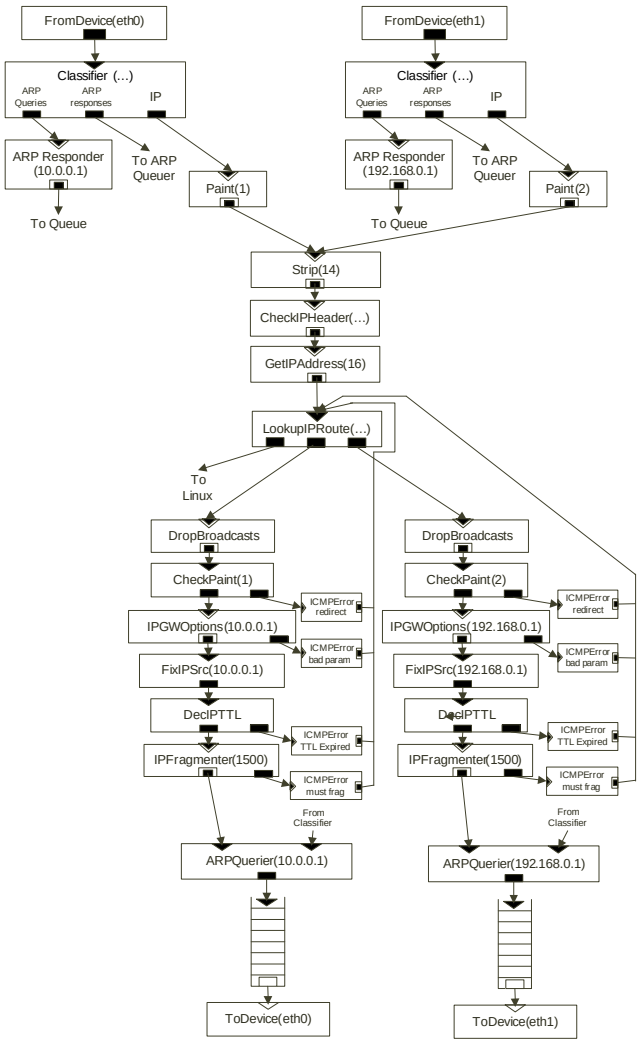

Рис. 3. Конфигурация простого IP — маршрутизатора.

Заключение

В статье рассмотрен программный комплекс, позволяющий реализовывать программные маршрутизаторы на базе Linux систем. Click уже с первого взгляда привлекает своей простотой, гибкостью и логичностью по сравнению с другими программными маршрутизаторами.

Заявленная разработчиками и измеренная эмпирически производительность высока и ограничивается пропускной способностью сетевых интерфейсов на аппаратном уровне. Однако первые эксперименты с сетевыми адаптерами на шине PCI Express показали, что на базе Click можно будет строить сверхбыстрые маршрутизаторы.

Глубокое изучение Click выявило, что он может поддерживать так или иначе практически все алгоритмы управления трафиком, которые присущи современному QoS и модели DiffServ в частности.

Все это позволяет сделать вывод, что у системы Click большое будущее в нише программных маршрутизаторов и что уже сейчас ее можно успешно применять в реальных ситуациях.

Также представляется перспективным применение данного пакета для имитационного моделирования, в рамках пакета ns-click. Данный пакет позволяет построить имитационную модель на ns2, используя при этом реальные модули системы Click.

Литература

1. *Горев А.* Advanced Routing и QoS. — `http://www.opennet.ru/docs/RUS/adv_route_qos/`.
2. *Коббс А.* Все о NetGraph. — 2003. — `http://citrin.ru/netgraph/`.
3. *Беляков С.* 12000 слов о Netgraph и FreeSBD. — 2006. — `http://www.unix.lviv.ua/content/view/33/27/`.
4. *Kohler E., Morris R., Chen B. et al.* The Click Modular Router. — 2000.
5. *Deri L.* Improving Passive Packet Capture: Beyond Device Polling // Proceedings of SANE 2004. — 2004. — `http://luca.ntop.org/`.

UDC 004.724:004.057.4

Routing Subsystem Click

D. S. Kulyabov, A. V. Korolkova, A. A. Khokhlov

Telecommunication Systems Department
Peoples' Friendship University of Russia
6, Miklukho-Maklaya str., Moscow, 117198, Russia

The Click Modular Router, a software for building powerful and flexible routers, is described in this article. It was developed in MIT in 1999. A QoS support is also included in this software.