Как мы 10Gigabit на Linux роутере маршрутизировали

Как то так повелось, что все маленькие домосети в начале 2000 годов в качестве пограничных маршрутизаторов всегда использовали PC based компьютеры: кто на Linux, кто на FreeBSD. Этому способствовали широкая доступность как самих PC based компьютеров, так и недоступность решений именитых производителей в данной области. Недоступность в разных смыслах слова, как по цене, так и физически оборудования, а также литературы по этому оборудованию. Сети переходили со 100 Мбит на 1 Гигабит и, в общем, PC based серверы могли справится с такими скоростями. Ограничение у таких систем всегда было только одно - количество пакетов в секунду, которое сервер маршрутизирует через себя с одного интерфейса на другой. Зато прочие ограничения аппаратных решений: количество маршрутов и т.п., можно считать, отсутствовали. Когда сеть перешагивала гигабитный канал, всегда возможно было поставить 4 сетевых платы (или две двухпортовых) и, в большинстве случаев, работать дальше.

Потом наступили времена 10 Гигабит, когда цена порта начала стремительно падать, и точки обмена трафика начали подключать к себе портами по 10Гигабит. В такой точке обмена трафика количество маршрутов не очень велико, и спокойно помещается в таблицы маршрутизации коммутаторов уровня предприятия. Так как большая часть тяжелого трафика проходила через такие точки, то доступ к ним начали делать через, описанные выше, коммутаторы. А маршрут по умолчанию мог быть по прежнему настроен на некий PC based сервер, который держал BGP с несколькими аплинками, отдающими полные таблицы маршрутизации Интернета. Потом произошел бум на рынке доступа, и все операторы начали давать 2, 4, 8, 10, 25 и до 100 Мбит для конечного пользователя. Это требовало расширения тех каналов, что шли с аплинков, а это приводило к росту нагрузки на PC based сервера. Кто-то обновлял подобные серверы, кто-то переходил на решения именитых зубров провайдерского оборудования, и маршрутизировал уже на них «международный» трафик.

Время шло, сети росли, обрастали кучей дорогого железа, что маршрутизировало растущий трафик, и PC based маршрутизаторы в массовых масштабах как-то были подзабыты, в больших сетях уж точно. А тем временем, производители, как процессоров, так и Ethernet-карт, тоже не сидели на месте.

Когда я впервые увидел описание Intel Ethernet-контроллера на 10 гигабит, с его аппаратной возможностью распараллеливать входящий поток на несколько очередей приема, еще тогда закралась мысль: вот где можно применить многоядерные системы. Но так, как на те времена программная структура ядра Linux, еще не совсем была готова к приему пакетов с разных очередей, не говоря уже про распараллеливание отправки пакетов в сетевую плату, идея была отложена в долгий ящик. И вот, как-то, проводя очередной софт-апгрейд одного из серверов (не помню уже, что делал этот сервер), решил задержать свой взгляд на сетевой части ядра Linux - а что же сменилось там? И к удивлению обнаружил, что появилась параллельная обработка на разных ядрах пакетов из разных очередей, и, главное, параллельная обработка отправки. Вот тогда я принял решение, о проверке возможности маршрутизации 10 Гигабит на PC based сервере. И начал с того момента смотреть, что к чему в ядре Linux, и где могут быть узкие места при параллельной обработке пакетов.

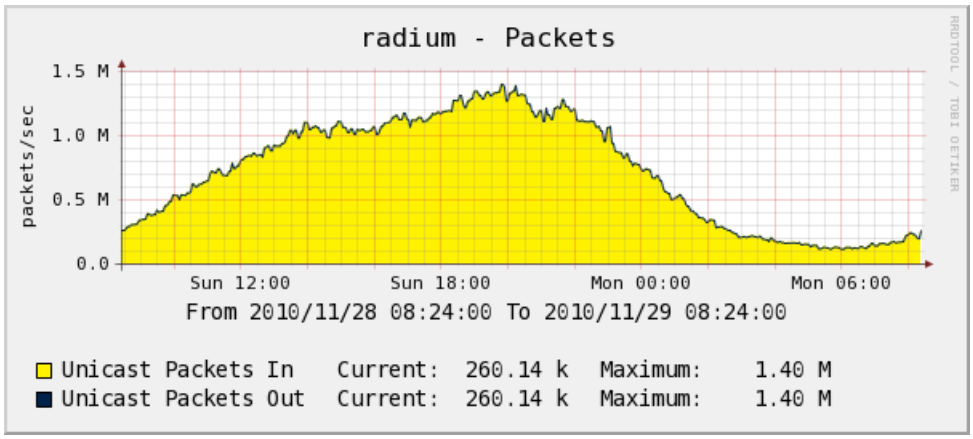
Просмотрев последнюю версию ядра 2.6.36.1 и убедившись, что последнее узкое место в драйвере VLAN было устранено (драйвер при подключение к реальной сетевой создавал столько же виртуальных очередей, как и на реальной сетевой) - а то, что же за это маршрутизатор без поддержки вланов, было решено попробовать-таки 10 Гигабит на Linux. Собравшись с духом, и получив на тестирование оборудование, собрал указанное ядро под 64 бита. Сразу обнаружился баг: последняя очередь сетевой платы вообще не функционирует на отправку пакетов. Порывшись в Интернете, я обнаружил, что баг этот, известен. Обнаружив, что без вланов такого не происходит, еще раз проверил драйвер 8021q. Оказалось, что проблема решалась добавлением одной строчки кода: skb_record_rx_queue(skb,i); в функции vlan_dev_hwaccel_hard_start_xmit. Теперь все было готово и, прогнав пару синтетических тестов, я увидел, что с парой гигабит система вполне справлялась, показывая минимальную загрузку. Большие потоки сгенерировать было просто нечем. Да и кому нужны синтетические тесты? Поэтому было решено проверить решение на реально работающей сети, где суммарный трафик приближается к 10 гигабитам.

Конфигурация сервера:

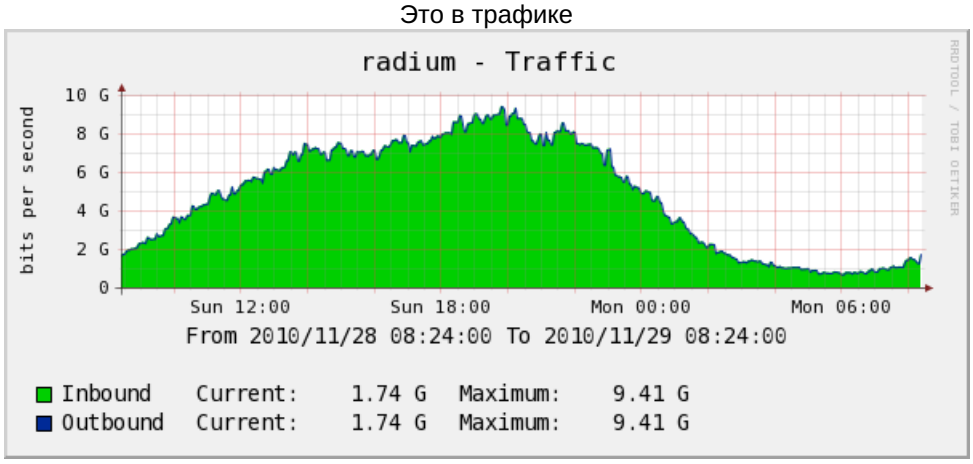
1. Два процессора AMD Opteron 6174;
2. Сетевая плата Intel на базе 82599 с двумя портами 10Gigabit под SFP+;
3. Памяти 16 ГБ, лишь из-за того, что не было планок по 1ГБ, а надо было 8 планок;

Сервер был подключен в ядро сети через SFP+ кабеля. На нем был поднят BGP из пакета quagga, и подняты пиры со всеми аплинками сети (3 full view), а также пиры со всеми паритетами, включая точку обмена трафиком UA-IX. В момент X, трафик был полностью переведен на этот сервер. Убедившись, что на шлюзах доступа абонентов все работает нормально, были убраны правила нарезания скорости, чтобы получить максимальный результат. Вот что мы получили за день работы сервера (это суммарно вход + исход).

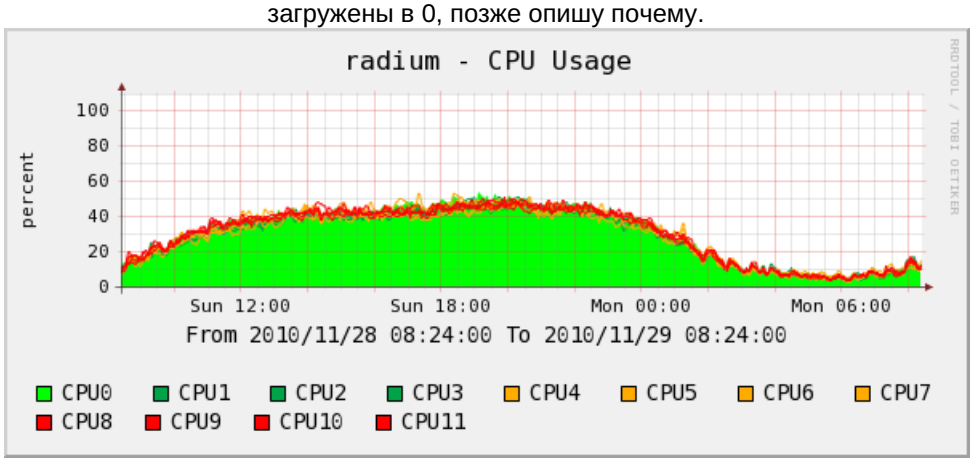
Это в пакетах в секунду



Это в трафике



Это загрузка 12 ядер первого сокета, во втором картина была идентична, кроме последних 8 ядер они были загружены в 0, позже опишу почему.



Во время работы постоянно контролировался пинг на 6 разных серверах CS, которые хостятся у разных операторов, и доступны через точку обмена трафиком. При этом пинг всегда держался в прежних 2-3 мс. Памяти было использовано до 1530 МБ.

Теперь по выводам, которые мы получили с этого тестирования:

1. Хоть и написано у Intel, что до 128 очередей, но для работы под RSS задействовано лишь 16 (остальные - это уже при виртуализации или для особых фильтров), то есть более, чем на 16 ядер оно входящий трафик не распараллелит. Отсюда и недогруз последних 8 ядер, ибо только 16 были задействованы из 24. Было бы 32 ядра, можно было бы попробовать одну сеть на первых 16, а вторую на других 16 ядрах;
2. У драйвера есть возможность указать, сколько очередей можно задействовать с каждой сетевой. По возможности, попробуем еще указать, чтобы использовало лишь 12 очередей - тогда 12 будет на один сокет, и 12 на другой. И повторим тест, дабы увидеть загрузку;
3. Можно бы задействовать два AMD Opteron 6136, тогда 16 ядер получим суммарно, но + 200 МГц процессора;
4. Ошеломляющая скорость перестроения таблицы маршрутизации при отвале одного из пиров с полной таблицей маршрутизации;
5. И главное: можно не снимать со счетов PC based маршрутизацию при скоростях в 10Гигабит.

Для оценки трафика в пике на NetFlow коллектор в этот день приходило чуть более 26 тыс. flows в одну секунду. Если дальше будут еще интересные моменты тестирования такой маршрутизации, то обязательно результаты

выложу. Еще хотелось бы проверить NAT и нарезку скорости, но где такую нагрузку найти? И еще: дисциплины нарезки скорости в Linux пока не умеют распараллеливать нагрузку.

Особую благодарность при проведении тестирования хочется выразить компании Entry, которая предоставила сервер для тестирования.