Балансировка

Тема балансировки одна из самых душещипательных и одновременно зубодробильных после QoS. Как это обычно бывает в сфере связи, всё выглядит очень просто с точки зрения настройки: пяток команд сюда, пяток туда и всё заработало, достаточно поверхностного понимания и придерживаться инструкций. И как обычно, если копнуть глубже при реализации чего-то сложнее, чем сеть провайдера среднего пошиба, то со всех сторон начинают высовывать свой нос проблемы. В этой публикации я позволю себе весьма вольно обращаться с термином "балансировка", иногда, понимая под ним именно балансировку, а иногда и распределение трафика.

Начнём с ареала обитания балансировки. В **агрегированных линиях** балансировка пакетов осуществляется на канальном уровне. **ECMP**/**UCMP** позволяет осуществлять балансировку между несколькими существующими L3-маршрутами (ECMP - равнозначными, UCMP - любыми). **MPLS Traffic Engineering** - это совсем другая стихия со своими особенностями. Да и к балансировке как таковой TE имеет посредственное отношение. Хотя трафик там действительно можно распределять по различным путям. Распределение нагрузки в **BGP** также отдельная тема, над которой бьются лучшие инженеры вместе и по отдельности.

Мы коснёмся лишь первых двух, оставляя TE для одного из следующих выпусков из цикла СДСМ (а BGP уже был рассмотрен). Если говорить именно о балансировке, то способов существует два:

- По пакетам (per packet)
- По потокам (per flow)

Метод балансировки **по пакетам** прост, как дважды два: один пакет туда, другой сюда - Round Robin. Загруженность различных линий получается ровненькая, красивая. Мечта инженера по развитию - не надо покупать 10-Гбитные модули, когда можно 5-6 1-Гбитных объединить в транк и наслаждаться абсолютно равномерным распределением.

Вся эта картина так прекрасна до того самого момента, пока у вас несколько примерно одинаковых линий одной длины идут одним путём. Когда же линии одного транка идут разными путями (разные пути прокладки оптических волокон) или сети с непредсказуемыми параметрами (например, РРЛ), пакеты начинают приходить в неправильном порядке.

TCP с этим умеет бороться. Он собирает полученные пакеты в буфере, раскладывает их по порядку и приложение забирает уже в готовом виде. Так или иначе, некоторые приложения, использующие UDP, тоже могут с этим бороться. Поэтому такая балансировка действительно используется в реальной жизни. Но вот если взять сервисы, вроде, CES, то пиши пропало.

Обычно приложения, использующие такие сервисы, крайне чувствительны к потерям, задержкам и неупорядоченности. Например, 2G базовые станции, которые подключаются по E1, в этом случае мгновенно теряют синхронизацию с BSC. Корень зла в том, что такие протоколы шлют приложению всё без разбору. Представьте себе, что пара таймслотов в E1 поменялась местами? Начнётся всеобщая неразбериха и хаос, и мир погрузится во тьму. И что бы вы ни делали, с пакетной балансировкой вы обречены на непредсказуемые проблемы, ведь счёт идёт на наносекунды. Как раз в этом случае приходит на выручку балансировка по потокам. Благодаря специальному механизму, грубо говоря, один TCP-поток у вас идёт по одной линии, другой - по другой. Тогда данные, принадлежащие одной сессии, дойдут именно в том порядке, в котором должны - в кабеле-то им не грозит перепутаться.

Но и тут нет такого долгожданного всеобъемлющего счастья. Много копий сломано на этой ниве. У одних вендоров это работает хорошо, у других тоже как-то работает. Об этом и поговорим. Разделение потоков может происходить на основе следующих параметров пакета:

- MAC-адрес источника/назначения.
- IP-адрес источника/назначения.
- TCP-порт источника/назначения.
- MPLS-метки.

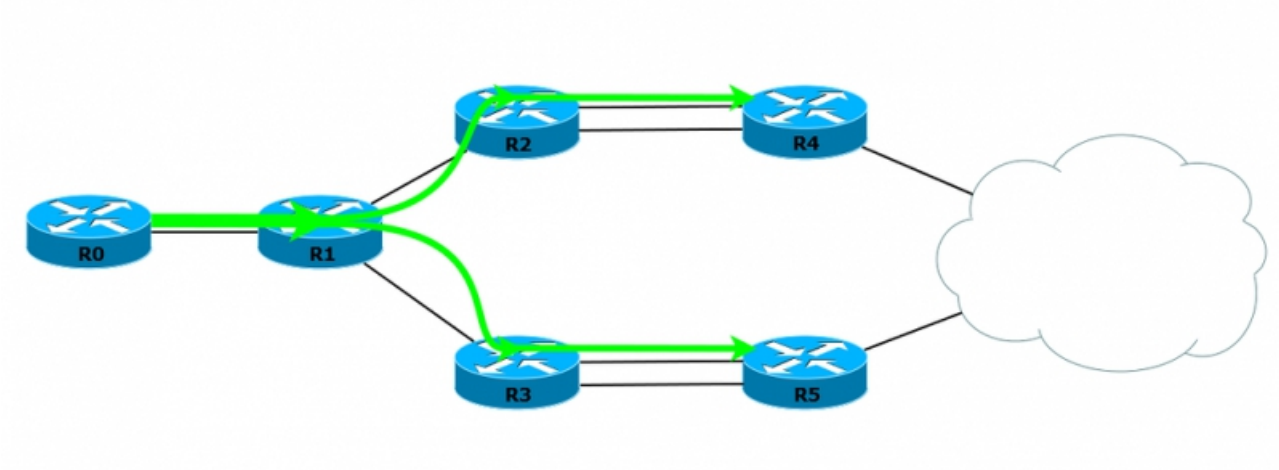
С помощью специальной Hash-функции из набора этих параметров вычисляется некое значение, на основе которого пакет отправляется в тот или иной порт. Естественно, для одинаковых исходных данных всегда будет получаться одинаковый результат, таким образом пакеты одного потока будут попадать в один порт. Первый тонкий момент появляется уже здесь. Обычно, производитель исходит из того, что в транке максимум 8 активных интерфейсов. Поэтому полученный хэш делится на 8, берётся остаток от деления. И эта вот цифра и определяет выходной порт. И если взять три порта в транке, то получается следующая ситуация:

Порядковый номер порта	0	1	2	3	4	5	6	7

Остаток от деления хэша на 8	0	1	2	0	1	2	0	1

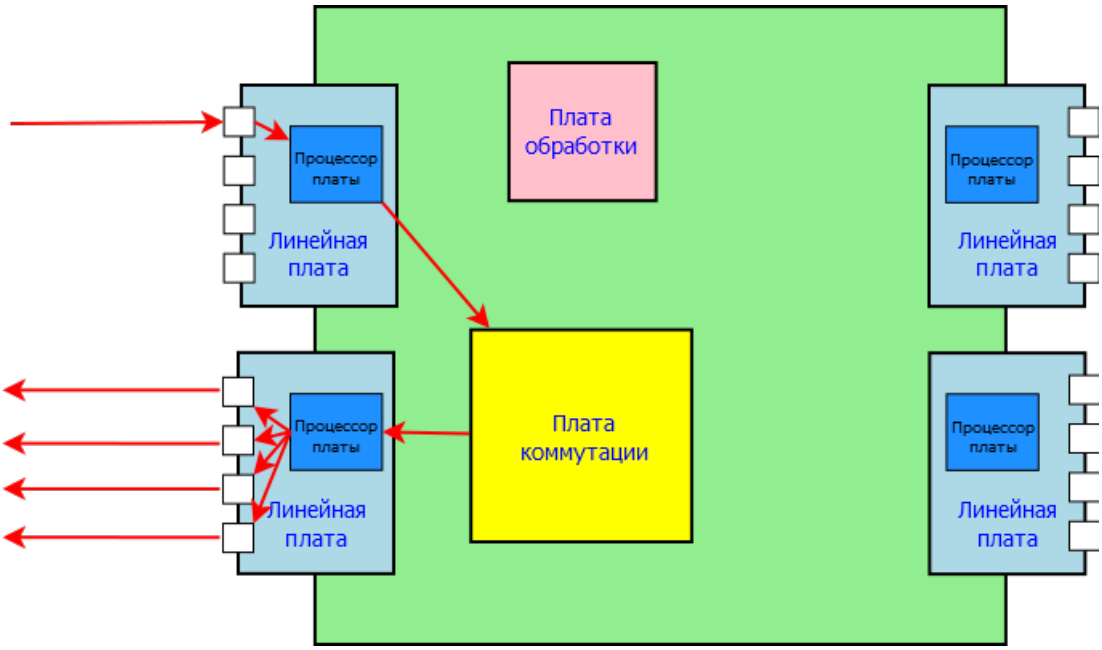
Три потока в первый порт, 3 потока во второй и только 2 в третий. Это не оптимальный алгоритм, но зато наиболее дешёвый. Поэтому общая рекомендация - брать число портов в транке, кратное двум. Кроме того, балансировка плохо себя ведёт на линиях, через которые проходит мало разных потоков. Например, при одном потоке не будет вообще никакой балансировки. Здесь же скрыта ещё одна проблема - если у вас один очень толстый поток и много мелких, то утилизация тоже будет неравномерной. Отсюда же следует, что чем более широкий (читай глубокий) набор параметров вы возьмёте, тем ровнее будет балансировка. Например, в случае маршрутизаторов балансировка по MAC-адресам не даст ничего, по IP-адресам распределение может оказаться неравномерным (на двух линиях, к примеру, 70/30), а вот по TCP-портам уже вполне хорошо. Но эти моменты довольно очевидны и на деле редко встречаются. О них, скорее, важно помнить новичку.

Менее очевидные случаи

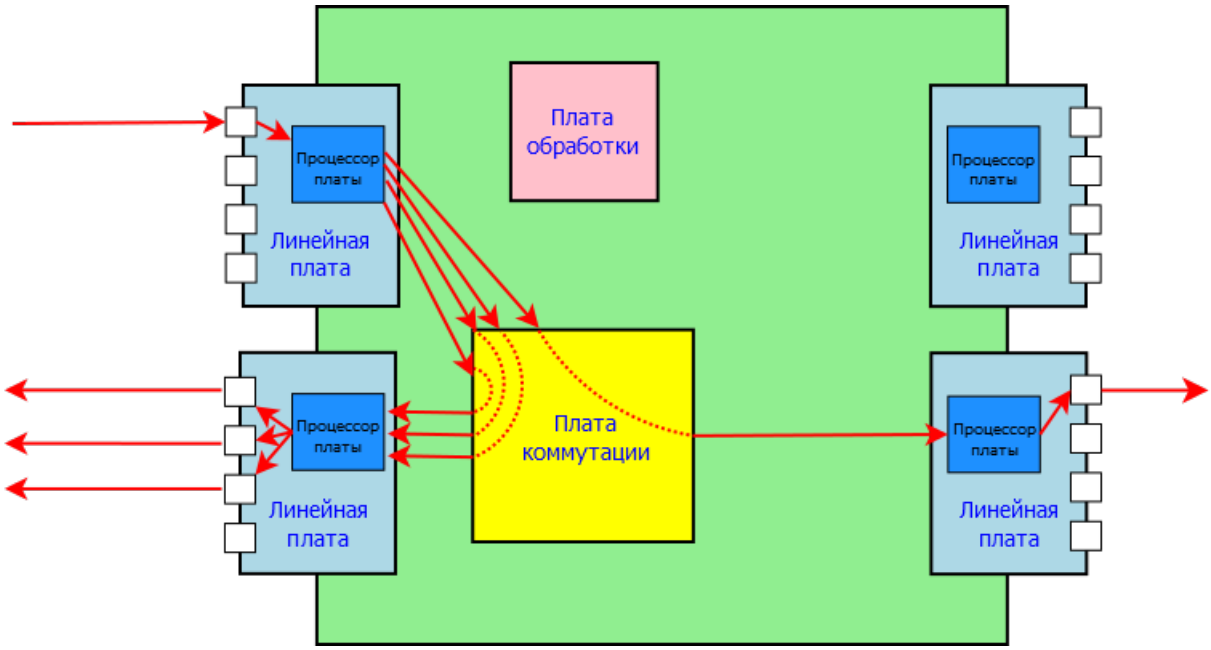


Инженер продумал архитектуру сети и организовал сначала ECMP, а потом двухгигабитный LAG, но наблюдает странную картину - на линии между R1 и R2/R3 балансировка идеальная. R2-R4 - всё уходит в один интерфейс. Дело в том, что на обоих участках используется одинаковая хэш-функция. R1 разделил потоки равномерно. На R2 попал уже сепарированный поток, в котором все параметры, как на подбор, дают одинаковый результат после применения хэш-функции. То есть R2 и рад бы распределить, но с алгоритмами не поспоришь.

А вот вам следующий вопрос: кто в маршрутизаторе озадачен вопросом, в какой интерфейс отправить очередной пакет? Плата, передающая пакет? Плата коммутации? А может, плата, получившая пакет? Логичным кажется предположить, что распределять по портам пакеты должна отсылающая плата - кому как не ей знать о членах транка и направлять путников по верной дорожке на плате?

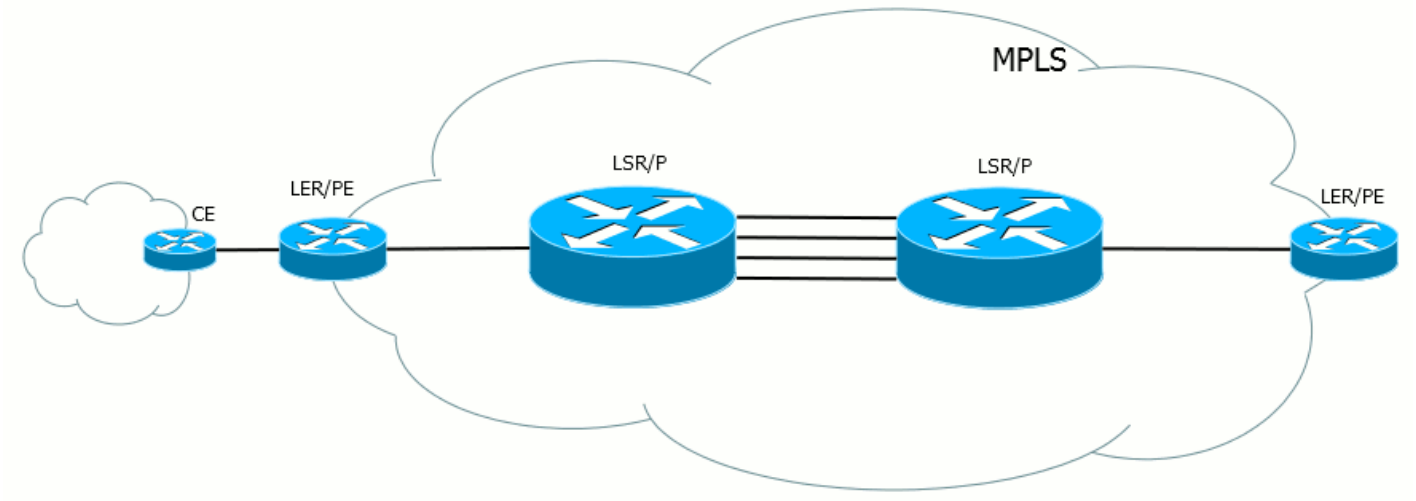


А теперь вспоминаем о том, что транк вполне может включать в себя порты с разных слотов, например, для резервирования не только линии, но и сбоя на плате. В этом случае пакет должен был бы прийти на одну из плат, а потом идти с неё на другую - никто так не проектирует оборудование, я надеюсь. Платы коммутации - более логичный вариант - они соединены со всеми платами и всё про всех знают. Но задача фабрики: правильно, а главное быстро, передать пакет, у неё нет времени на раздумья и расчёт хэшей. Естественно, плате обработки нет никакого дела до отдельных пакетов - она до их уровня не опускается. И последний, кто остаётся в очереди - плата, получившая пакет. С платы обработки на неё уже загружен FIB, поэтому про то, какие члены есть у того или иного транка ей известно, и всё, что остаётся сделать - посчитать хэш и послать в нужном направлении.



И вот тут есть интересный момент. Вы покупаете новую линейную плату для расширения, смотрите её спецификации, тестируете - всё хорошо, ставите на коммерческую сеть, а балансировка работает как-то странно, неправильно. А ноги растут как раз из того, кто принимает решение. Важно понимать, что какую бы крутую вы ни купили плату, за балансировку выходящих с неё пакетов отвечает не она. Впрочем, ситуация эта довольно редкая.

Ещё одна проблема - головная боль многих вендоров - это балансировка MPLS VPN. Если среди прочих у вас есть какой-то жирный VPN, или он и вовсе один, ждите проблем с балансировкой. Во-первых, не каждый маршрутизатор смотрит вглубь MPLS-пакета. Обычно хэш вычисляется на основе меток. Как правило, для этого берётся несколько верхних меток - транспортная, FRR, VPN, CSC и другие. Считается, что это в достаточной мере разнообразные данные. И во многих случаях это действительно так. Но если допустить существование того самого толстого VPN, то меток уже недостаточно - они все будут одинаковыми.

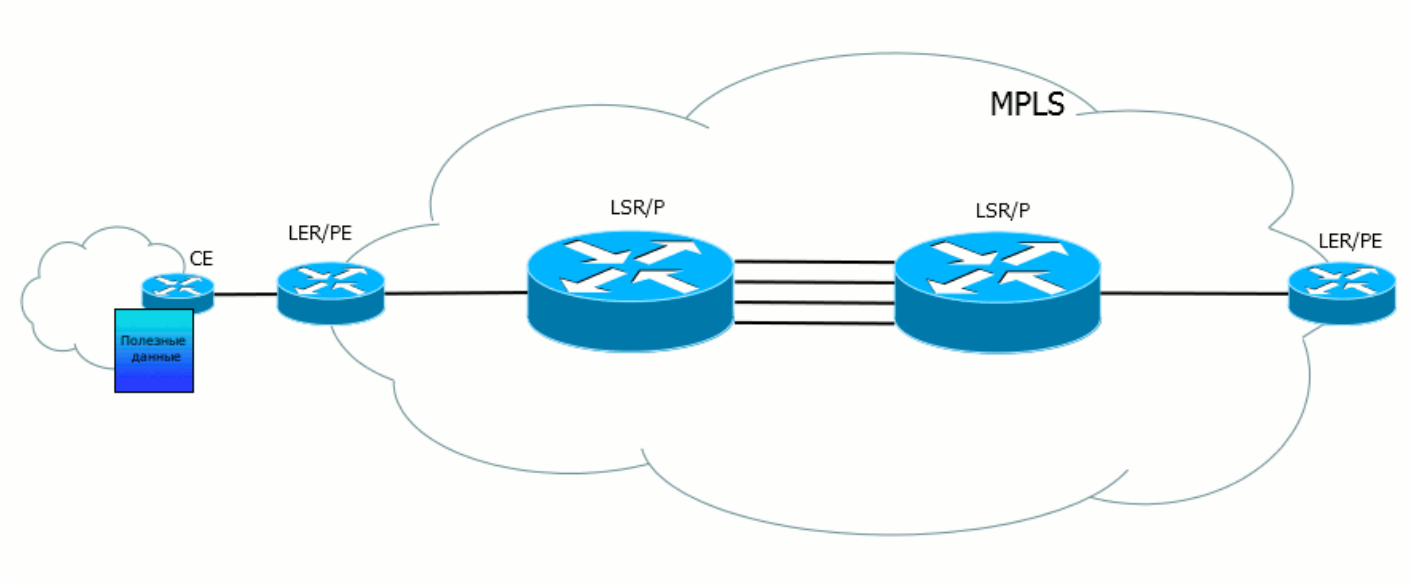


Существуют железки покруче, которые могут проводить глубокий анализ пакета и использовать в хэш-функции помимо меток ещё и IP-адреса и протокол. Иногда даже могут смотреть и порты

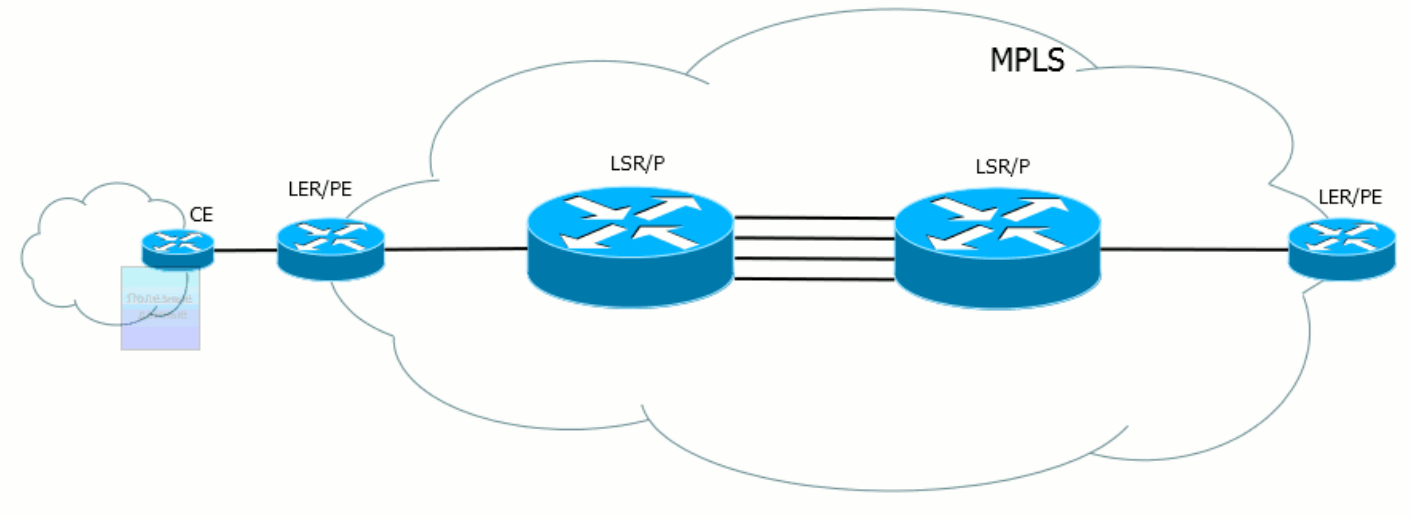
TCP/UDP. Это позволяет добиться ровной балансировки ещё в ряде случаев. Но, строго говоря, это моветон организовывать DPI на высоконагруженных линиях, P-маршрутизаторам и без того есть чем заняться. Да и маршрутизаторы, которые в состоянии это делать, могут стоить неприлично дорого для небольшой компании.

По этой причине существует официальный костыль - Entropy Labels. Звучит таинственно и зловеще - Метки Хаоса. Но суть их довольно простая - добавить разнообразия ~~в жизнь~~ в стек меток MPLS. Есть даже драфт на эту тему. Заботы по разбору IP/TCP заголовков перекладываются на Ingress LSR (иными словами LER или PE). Ему во всех отношениях проще - во-первых, не нужно копаться в ворохе MPLS-меток, потому что на свой интерфейс от CE он получает чистый IP-трафик, во-вторых, объём трафика всё-таки, как правило, несравнимо меньше. Вот этот LER, вообще говоря, формирует весь базовый стек меток - метка VPN и транспортная. Всякие RSVP over LDP, CSC, FRR могут добавиться позже, но они не играют никакой роли. Так вот LER поступает хитро в этом случае - он добавляет элемент разнообразия в стек меток. Для этого у него есть два пути:

- Для одного VPN'а назначать две разные метки VPN в зависимости от содержания пакета;
- Добавлять дополнительную метку в самый низ стека, которая не играет вообще никакой роли.



Хитрость в том, что никакому транзитному маршрутизатору не нужно беспокоиться о том, как обеспечить ровную балансировку. Более того, они даже не подозревают, какие махинации проворачивают LER. Они просто считают хэш на основе набора меток, как обычно. Единственная сложность с этим в том, что надо настраивать дополнительный функционал на граничных маршрутизаторах, возможно, на всех. Плюс, другие концы VPN'а тоже должны знать об этом. Возрастает сложность и падает прозрачность обслуживания и поиска неисправностей. Но есть и промежуточный вариант: анализом содержимого пакетов под MPLS-метками всё же занимается тот LSR, на котором нужна балансировка, но это происходит до собственно вычисления хэш-функции - маршрутизатор анализирует полученный пакет, и сам же и назначает дополнительную метку при первичной обработке. В итоге при вычислении хэша вглубь уже лезть не надо, что упрощает процесс.



Альтернативный вариант решения проблемы - использовать возможности самого MPLS TE: разбивать LAG на составляющие, задавать IP-адреса каждому из интерфейсов отдельно, поднимать IGP, создавая тем самым ECMP/UCMP. Дальше организовывать балансировку средствами Traffic Engineering. Но! Это то ещё удовольствие, и вы встречаете те же проблемы в случае одного жирного VPN с одинаковым набором SIP, DIP и тд. Что же до BGP, который я обещал не трогать, то в нём не идёт речь о попакетной или попотоковой балансировке: там мыслят другими критериями. Добиться здесь хорошего распределения трафика, особенно входящего - это очень шаткий эквилибриум. Вы можете вешать препенды, настраивать Community, но всё может рухнуть от одного чиха инженера из любого вышестоящего провайдера. А бывает и так, что распределение происходит там, где его никто не ожидал.

> Я слышал историю о том, как однажды местечковый провайдер начал наблюдать необычную загрузку интерфейсов у себя - часть трафика с внешнего линка ушла на неожиданное направление, то есть там, где нужно убыло, а где не нужно прибыло. В ходе разбирательств выяснилось, что какой-то мелкий провайдер случайно начал анонсировать Full View в точку обмена трафиком. У местечкового провайдера никакой фильтрации не было, а этот путь оказался по каким-то причинам предпочтительнее других.

Это лишь сливки проблем с балансировкой. У каждого уважающего себя Дон Кихота должна быть хотя бы одна история, как он боролся с ветряными мельницами.