Как построить Гугл

Содержание

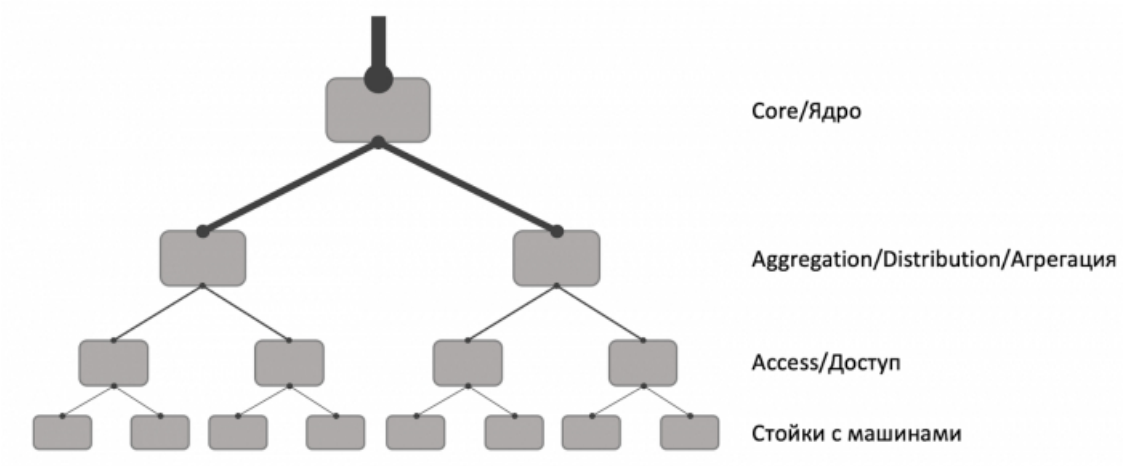
- **Трёхуровневая иерархическая сеть**.
- **Северо-Юг против Запада-Востока**.
- **Сети Клоза**.
  - **Блокируемые и неблокируемые сети и переподписка**.
  - **Многоуровневая сеть Клоза**.
  - **3-Tier, Fat Tree, Clos, Folded Clos, Leaf-Spine?! Аааа!**
  - **Маршрутизация**.
  - **Минусы сети Клоза**.
- **Альтернативные топологии**.
- **Выбор оборудования**.
- **Заключение**.
- **Полезные ссылки**.

Если заглянуть в далёкое прошлое, датацентры жили примерно в таких условиях:

- Преобладающий North-South-трафик от клиента к серверу и обратно.
- L2-сеть в силу требования мобильности виртуальных машин без смены IP-адреса и высокой цены на L3-оборудование.
- Любое сетевое оборудование, обладающее широкой функциональностью, стоило действительно немалых денег.

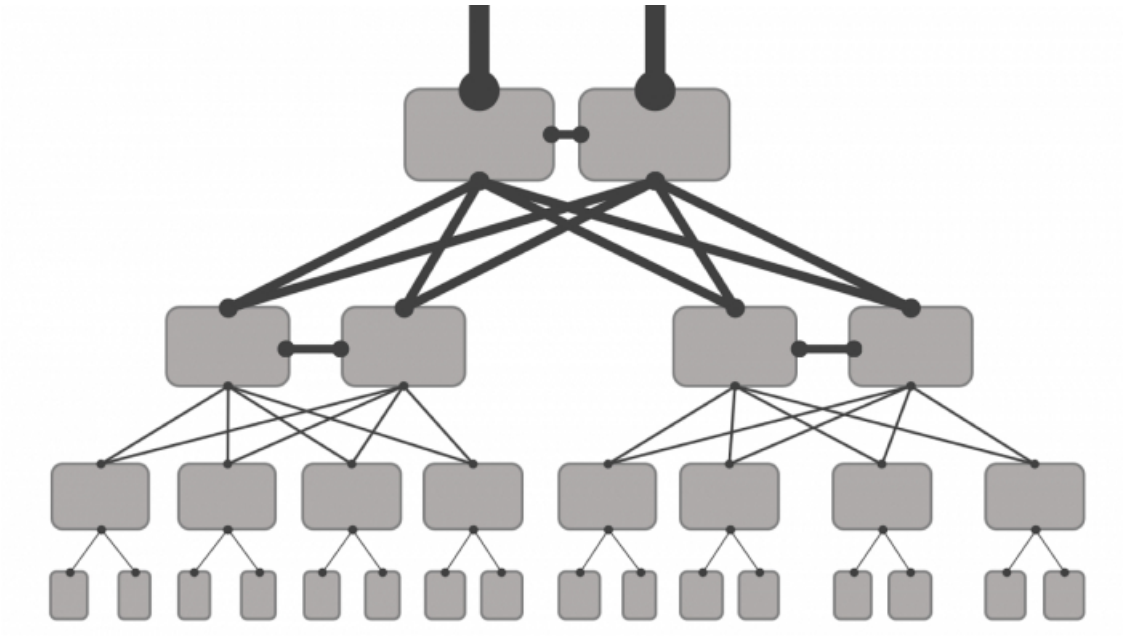
Поэтому традиционная сеть ДЦ была такой:

Трёхуровневая иерархическая сеть - 3-Tier



Это классическая трёхуровневая архитектура сети, известная нам с СДСМ0: Access, Aggreggation/Distribution/Core. Так были построены сети операторов связи и предприятий, а впоследствие её же перенесли в датацентры, потому что только так и умели, да и оборудования другого не было.

Чем дальше в ядро, тем толще линки. 100 Мб/с к машинам, 1 Гб/с от коммутаторов доступа к коммутаторам агрегации. 10 Гб/с (а в то время, возможно, LAG из нескольких гигабитных линков) к ядру сети. И т.д. Потому что весь трафик от серверов сливался в аплинки. Наверху иерархии стояла пара God Boxes, концентрирующих на себе все сервисы и всю маршрутизацию. На них инженеры молились и старались не дышать. Это были модульные шкафы, занимающие полстойки, а то и стойку, а то и кластера из нескольких стоек. С этой топологией была пара проблем. Если взять ситуацию, как на рисунке выше, то отказ любого линка сразу же приводит к выходу из строя части сети. Фактически же, строили сети всё-таки с резервированием, и на каждом уровне обычно стояло по паре устройств:



Да, это требовало богомерзкого STP, но это позволяло обеспечить хоть какую-то стабильность. Здесь **выпадение** линка или устройства приводило к двукратной деградации пропускной способности и, очевидно, работе без резерва. Кроме того, все сервисы при таком подходе сосредотачивались как раз на этих верхних устройствах.

Вывести их из эксплуатации, для обновления, например, означало также потерю половины пропускной способности сети, а то и вовсе полную недоступность сервисов из-за того, что второе устройство настроено неправильно.О, сколько раз на моей памяти инженер со спокойной душой выводил устройство из эксплуатации со словами "второе же есть", и ронял сеть, потому что на втором кто-то в своё время превентивно погасил интерфейс или положил BGP-сессию.

Расширение пропускной способности - это отдельная болезненная история:

- Расширение LAG"ов.
- Закупка линейных карт.
- А если карты ставить больше некуда, то апгрейд железа, на ещё более мощные и большие ящики.

Итого:

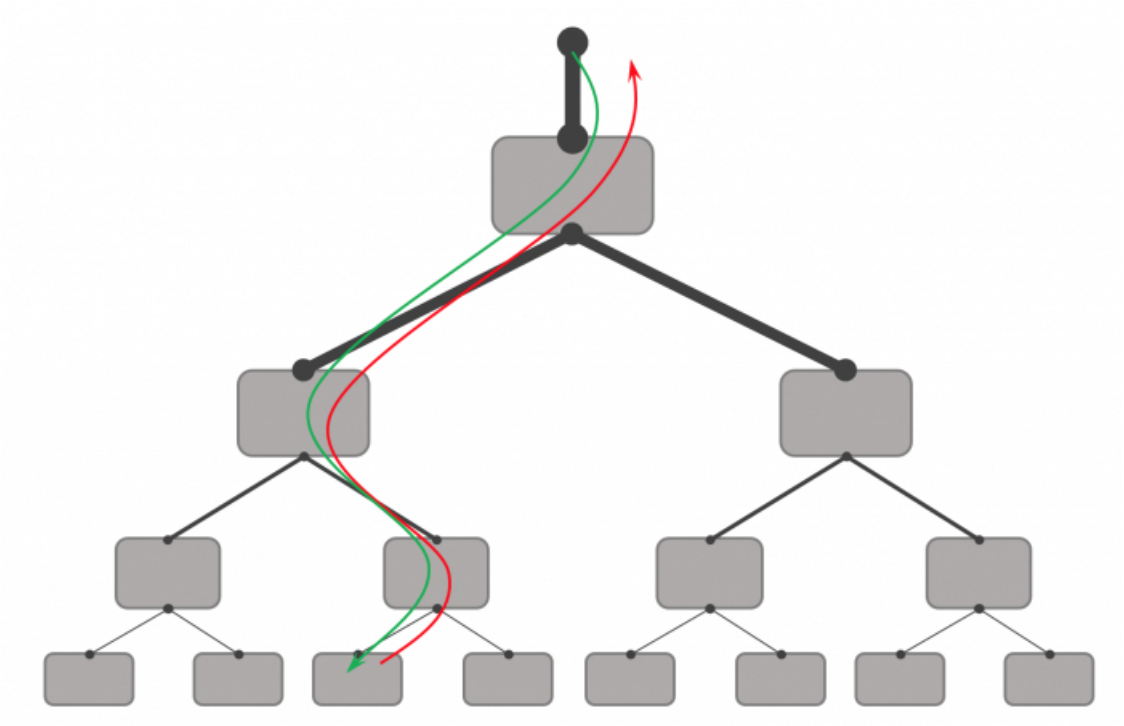
1. Классическая трёхуровневая сеть не может обеспечить лёгкое расширение пропускной способности.
2. Она довольно плохо масштабируется: добавление нового модуля в ДЦ потребует снова расширения линков.
3. Она не может обеспечить высокую степень резервирования - в L2-топологии использование избыточных линков затруднительно из-за "особенностей" MSTP. Соответственно, часть линков просто простаивает. А при выпадении узла есть риск потерять не только пропускную способность, но и всю сеть на некоторое время.
4. В общем, эксплуатировать и поддерживать классическую сеть проблематично.

Любопытно, что L2-топология, обычно используемая в датацентрах прежде, настолько избаловала разработчиков приложений и системных администраторов, что они долгое время не представляли своей жизни без L2. Здесь и кроются истоки TRILL, SPB, FabricPath и прочих технологий, так желающих облегчить жизнь Ops"ам. И только в последние годы прослеживается устойчивая тенденция к L3 до стойки в датацентрах. Однако, было бы лукавством сказать, что L2 отживает своё. Зачастую, он просто переместился на уровень абстракции выше - а именно, в оверлей. Тот же VXLAN благополучно растягивает L2-домен до треска. EVPN тоже умеет его сигнализировать. Однако, общая тенденция такова, что на физической сети, в андерлее, остаётся исключительно L3.

Северо-Юг против Запада-Востока

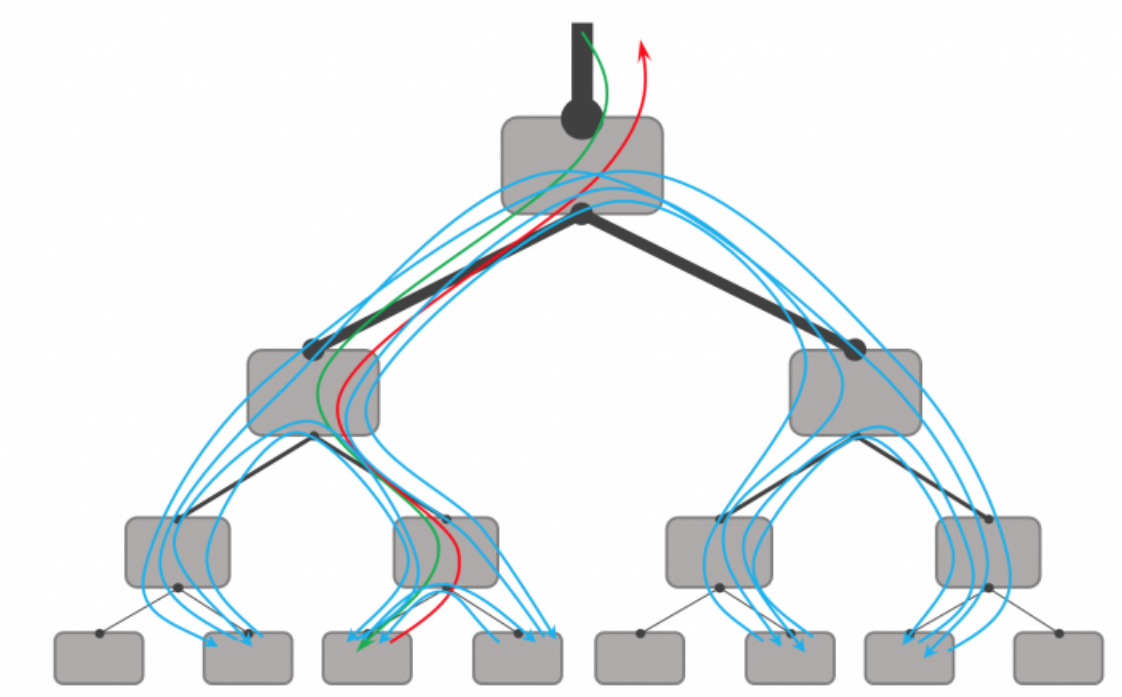
Однако, всё вышеперечисленное в некотором смысле мелочи, которые можно было бы решить. Один переход на L3 снял бы бóльшую часть сложностей. Проблема в том, что изменился мир. Внезапно появились Big Data, Map Reduce, ML, гигантские базы данных, аналитика, контекстная реклама. А теперь и AI из пауэр-поинтов на нас поглядывает. Исторически запрос пользователя полностью

обрабатывался в рамках одного хоста. Был запрос от клиента, который пришёл сверху, и ответ, уходящий обратно наверх. В такой сети преобладает вертикальное (North-South) направление. И иерархическая сеть такой спрос вполне удовлетворяла.

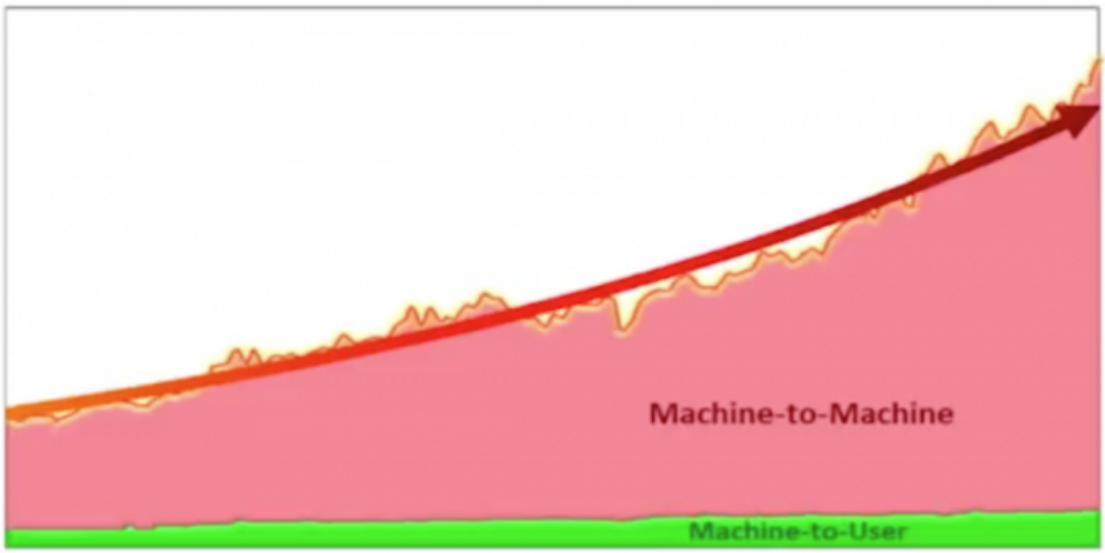


Теперь в ДЦ преобладает горизонтальный (East-West) трафик между серверами. И вот как это получается:

1. Клиент запросил веб-страничку.
2. Его запрос пришёл на балансировщик трафика.
3. Балансировщик перенаправил его на один из фронтендов.
4. Тот послал запрос на один бэкенд-сервер, чтобы получить текстовый контент.
5. На другой сервер, чтобы получить данные о пользователе: пол, возраст, предпочтения, последний поиск.
6. На третий - проанализировать данные по локации.
7. Второй и третий послали новые данные на четвёртый, чтобы пополнить информацию о пользователе в БД, чтобы ещё лучше знать какие отмычки для лифтовых дверей предпочитает среднестатистический житель Ленинского района города Новосибирска, пользующийся Android 9.1.
8. На пятый - сформировать контекстную рекламу.
9. Потом все разом начали отвечать фронтенду.
10. Фронтенд отдал всю страницу обратно.



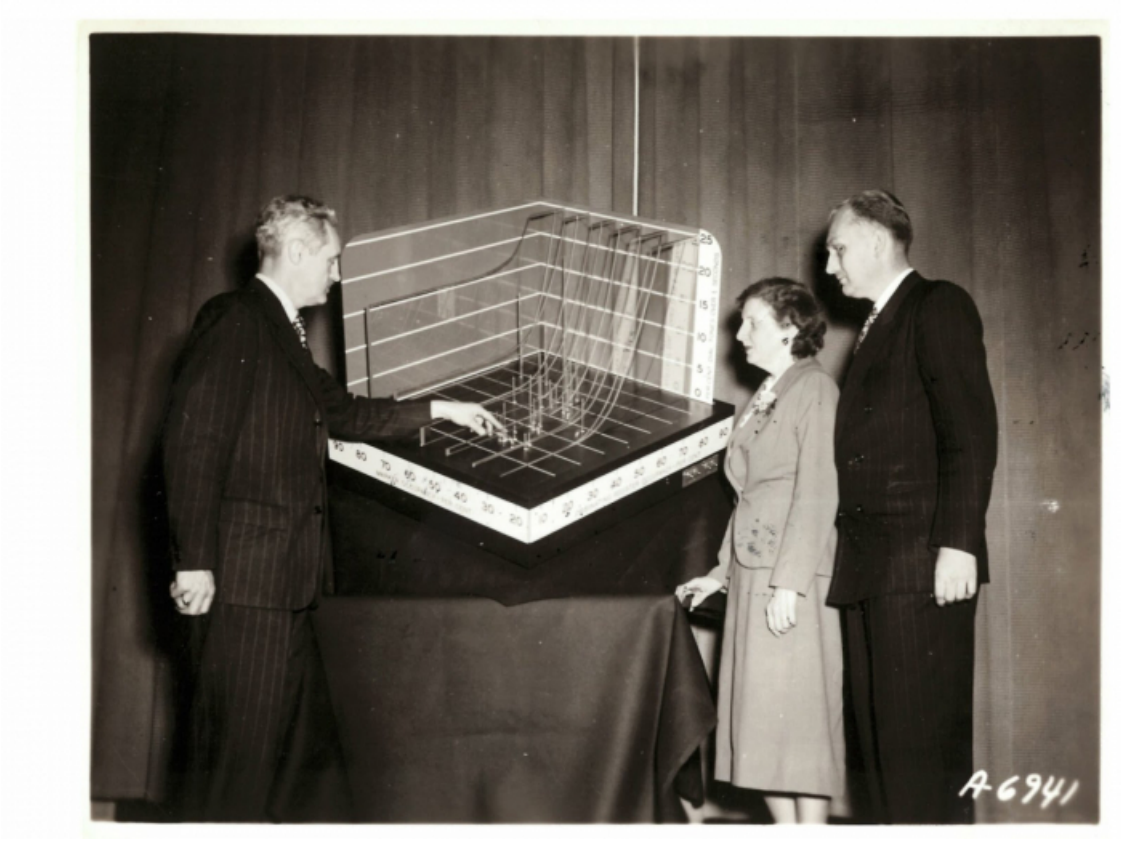
То есть, бóльшая часть событий произошла внутри сети сервис-провайдера. При этом, запрос-ответ внутри ДЦ может многократно превышать по размеру запрос-ответ пользователю. Добавим к этому репликации БД, внутренние хранилища, логи - и всё становится понятно (кивните, если согласны). Вот такой график Facebook показывает на конференциях, иллюстрируя тенденцию к стремительнмоу росту intra-dc-трафика:



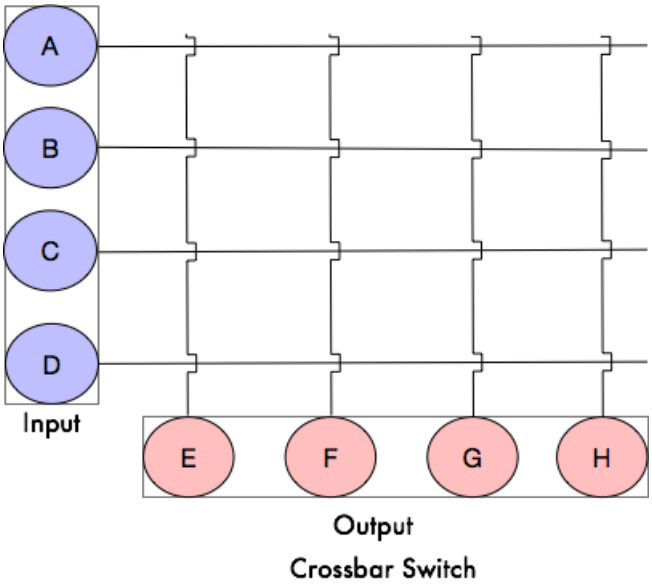
Традиционные сети органически не приспособлены к этому из-за своего акцента на North-South. L2-топологиям из-за низкой утилизации полосы и склонности к штормам тоже отказано (TRILL бы, возможно, спас ситуацию, но где он?). Ну, и "умное" железо, если и не стало резко дешевле, то только из-за всё возрастающей производительности, доходящей сегодня уже до 12,8 Тб/с на одночиповую коробку размером 4 юнита. А так уже любая вафельница умеет в L3+BGP.

Сети Клоза

Современный подход к строительству ДЦ, которые используют гипер-скейлеры и клауд-титаны, вроде Яндекса, Фейсбука и Гугла - развитие идей крутого парня из Bell Laboratories, Чарльза Клоза (который, на самом-то деле, Шарль Кло, ибо француз).

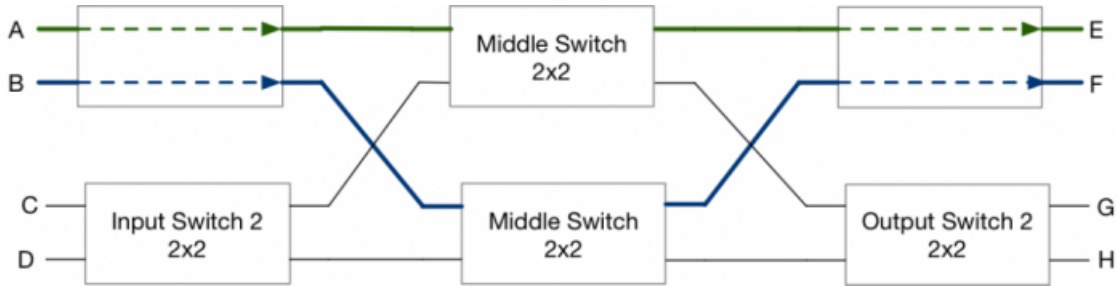


Он в 1953 году, решая проблему проключения звонков через телефонную станцию, предложил неблокируемую сеть, в которой любые входы могут установить соединение с любыми выходами в любой момент времени. Прежде, для решения этой задачи использовался, так называемый, **Crossbar**, соединявший каждый вход с каждым выходом.



Изображение с сайта packetpushers.net

Учитывая, что большая их часть простаивала, решение, и без того сложное в реализации (n^2 соединений), было ещё и неоптимальным. Он предложил разделить входы и выходы дополнительным уровнем коммутации, который позволит простраивать соединения между входами и выходами по запросу. Это и назвали **сетью Клоза**.



Изображение с сайта packetpushers.net

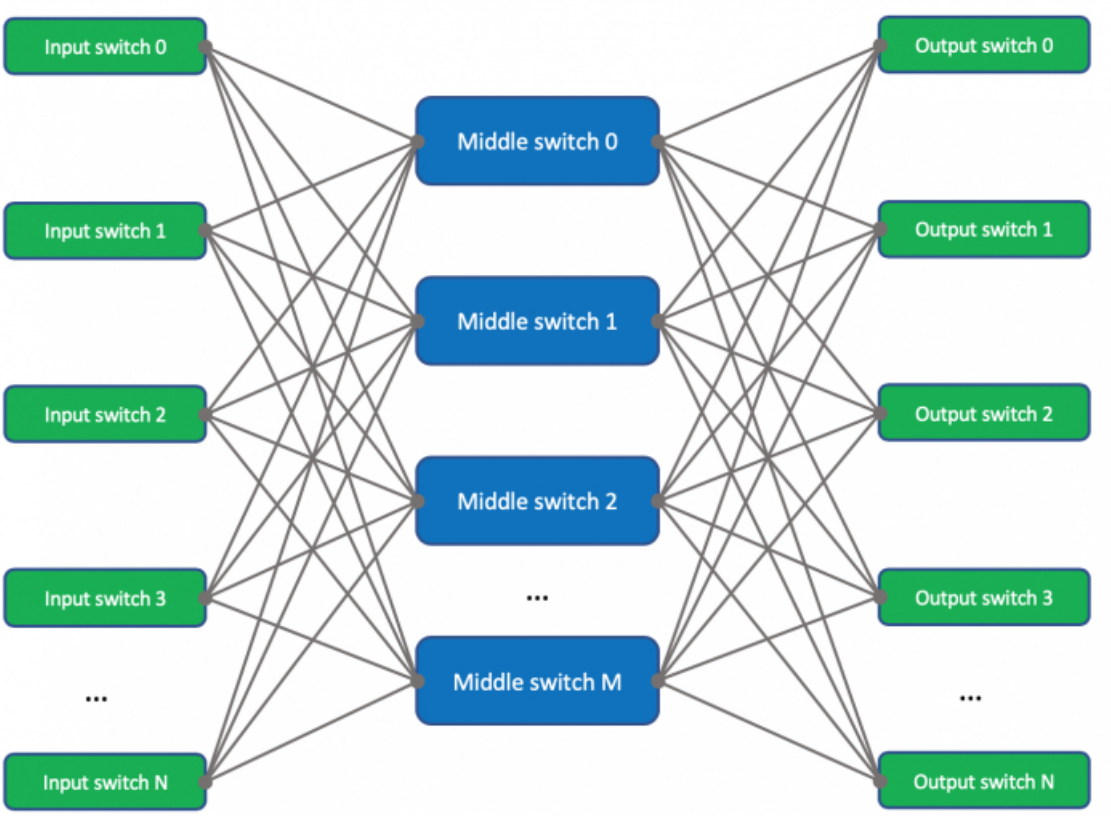
Сложность такой сети значительно ниже, чем у Crossbar, поскольку имеет гораздо меньшее количество точек коммутации: 6n^(3/2)-3n. Вот таблица необходимого числа коммутаций:

TABLE I — CROSSPOINTS FOR SEVERAL VALUES OF N

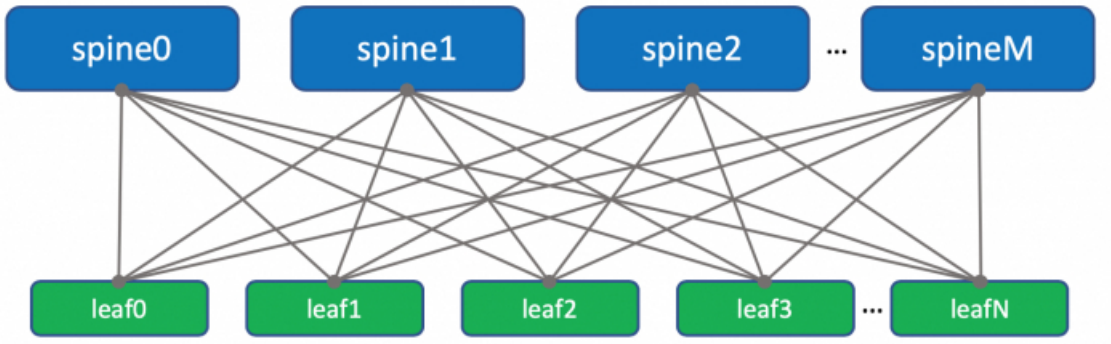
N	Square Array N^2	Three-Stage Array $6N^{3/2} - 3N$
4	16	36
9	81	135
16	256	336
25	625	675
36	1,296	1,188
49	2,401	1,911
64	4,096	2,880
81	6,561	4,131
100	10,000	5,700
...	.....	.....
1,000	1,000,000	186,737
10,000	100,000,000	5,970,000

Уже начиная с 36 конечных точек, сложность топологии Клоза меньше, чем связей "каждый с каждым". При этом оригинальная сеть Клоза подразумевает отсутствие блокировки, то есть любые два входа всегда могут быть скоммутированы друг с другом. Выделяют строго неблокируемые или неблокируемые при перекоммутациях - для нас это сейчас неважно. Фактически же, более рационально использовать сети с низкой вероятностью блокировки.

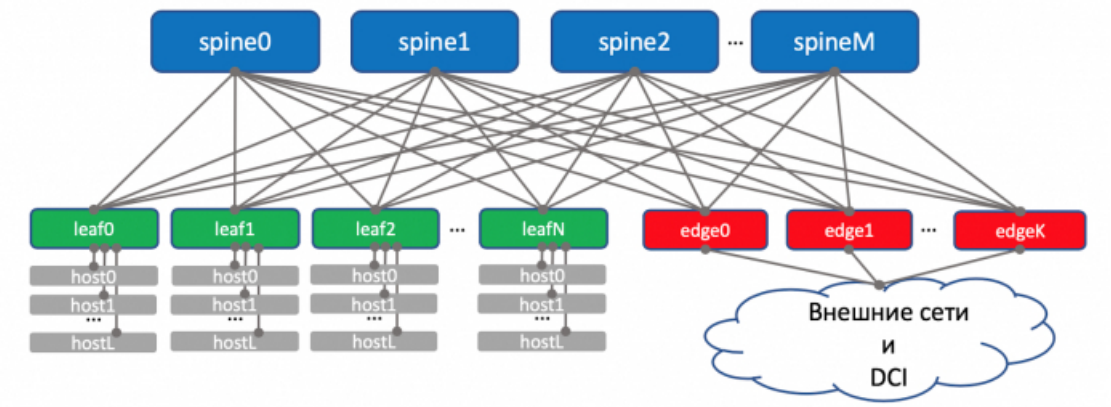
В чистом виде топология Клоза описывала сети с коммутацией каналов. Но в 90-е годы с развитием пакетной коммутации эта идея обрела альтер эго. В модульном сетевом оборудовании стало уже сложно соединять все SerDes"ы одного чипа со всеми SerDes"ами другого, поэтому появился дополнительный уровень, в общем скрытый от пользователей, **фабрики коммутации**, единственной задачей которых было доставить пакеты от входного порта в выходной, убрав при этом необходимость в полносвязной топологии. Это не что иное, как сеть Клоза. Ну и последняя на сегодняшний день инкарнация - это **датацентровые сети**, которые суть - те же фабрики коммутации. Типичный вид трёхуровневой сети Клоза:



Более привычный вид Leaf-Spine - ещё он называется Folded Clos - потому что словно действительно сложен пополам.



Другое название такой сети - **Fat Tree**. Вместо весьма интеллектуального, а, соответственно, и склонного к ошибкам и багам уровня агрегации, появляется примитивный уровень коммутации - Spine, задача которого - очень быстро переложить пакет с одного Leaf на другой. К Leaf"ам подключаются машины. Сами Leaf"ы подключаются к каждому Spine"у. А Spine"ы соответственно ко всем Leaf"ам. Таким образом, между любой парой машин будет существовать большое количество равноценных путей (по количеству спайнов) с всегда одинаковым числом хопов - 3 для сети, изображённой выше. Выход же во внешний мир или в другие ДЦ обычно реализуется через отдельные коробки, которые с точки зрения фабрики выглядят как Leaf-коммутаторы, однако гораздо более функциональные. Называются они Edge-Leaf.



Впрочем, привычная операторам реализация границы датацентра всё же имеет право на жизнь. В этом случае функции Edge выполняют спайны. В таком сценарии интеллект отчасти перемещается на Leaf"ы, отчасти на Edge-Leaf"ы.

В чём плюсы такой сети?

1. Во-первых, в отсутствии неиспользуемых линков (при учёте, что мы отказываемся от L2). ECMP - это воздух для сетей Клоза.
2. Во-вторых, конечно, в широких горизонтальных каналах для East-West-трафика.
3. В-третьих, выпадение одного устройства или линка не влечёт фатальных последствий. Если это был ToR, то пострадает только одна стойка. Если Spine - просядет пропускная способность, но не на 50%, как это было бы прежде, а лишь на 1/n, где n - число спайнов.
4. В-четвёртых, простота вывода спайнов из эксплуатации. Благодаря небольшой деградации и отсутствию интеллекта на этом узле, проводить работы на них не так страшно, как на God-Box"ах
5. В-пятых, масштабируемость. Новые лифы можно безболезненно добавлять пока не кончатся порты на спайнах. Добавлением спайна можно расширить аплинки лифов. Добавлением эджей - полосу пропускания наружу. Не нужно расширять LAG"и и вставлять новые платы. А если вдруг портов потенциально не хватает или хочется объединять модули одного ДЦ в суперфабрику, то можно надстроить ещё один уровень спайнов (или не один). Хотя тут есть нюансы.Такой метод масштабирования называется **горизонтальным** или **Scale out**. Добавляется просто ещё одно типовое устройство, практически линейно увеличивая пропускную способность. Нынче это очень модно. Теперь всё масштабируют горизонтально. Классический подход - **вертикальное** масштабирование **Scale up** - тут добавляют мощностей в текущие устройства, чтобы расширить их возможности. В маршрутизаторы добавляют платы, в сервера - оперативную память и процы.

Здесь может сложиться впечатление, что мы просто перенесли всю сложность с God-Box"ов на граничные маршрутизаторы. Что же, впечатление не обманывает, так оно и есть. Но L3 здесь позволяет, используя ECMP, масштабировать внешнюю связность, просто увеличивая количество граничных коробок (см. выше Scale out). И также легко и безопасно выводить их из работы, как спайны (почти).

Блокируемые и неблокируемые сети и переподписка

Как уже говорилось выше, топологии Клоза были придуманы для сетей с коммутацией каналов. Поэтому свойство "строго неблокируемые" означало, что независимо от количества уже активных каналов, всегда будет возможность соединить свободный вход со свободным выходом. Неблокируемые при перекоммутации - это топологии, в которых тоже всегда возможно найти свободный канал, но, возможно, придётся перекоммутировать существующий активный. Они требуют меньшее количество свитчей.

Также, существуют топологии с низкой вероятностью блокировки. Если переложить эти термины на сеть с коммутацией пакектов, то неблокируемая сеть Клоза описывает в некотором смысле фабрику без переподписки, когда сумма аплинков не меньше суммы даунлинков. Например, схема с тридцатью даунлинками по 10Гб/с и тремя аплинками по 100 Гб/с на Leaf - это фабрика без переподписки. И даже при одновременной передаче данных всеми подключенными хостами, всегда хватит аплинков.

Фактически, это не всегда необходимо. Глядя на пиковую утилизацию аплинков, например, мы видим, что она достигает 80 Гб/с, и если мы не ожидаем рост, то можно делать два аплинка по 100 Гб/с. Получим переподиску 2:3, аналогичную сети Клоза с низкой вероятностью блокировки.

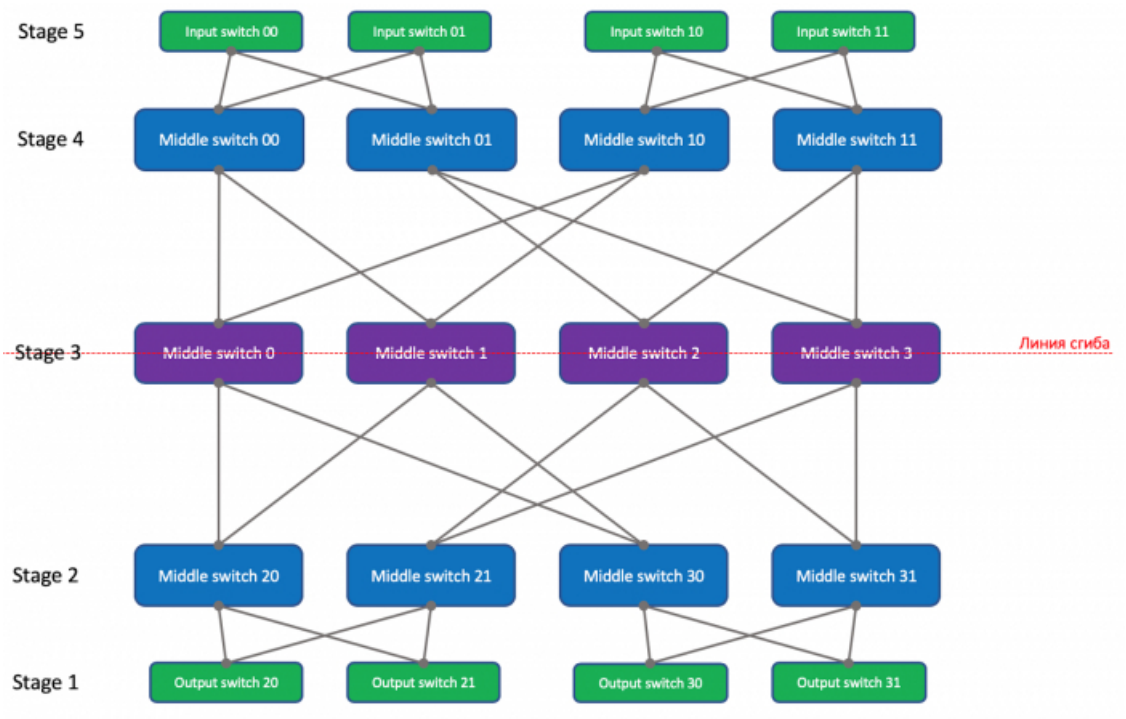
Строить фабрику с переподпиской или без - вопрос на усмотрение сетевых архитекторов. Но возьму на себя смелость сказать, что чаще строят с переподпиской, чтобы сэкономить порты на спайнах.

Касательно пропускной способности стоит ещё отметить ситуацию с кратковременными всплесками трафика (микробёрстами), рождёнными, например, инкастом. С ними не справится и фабрика без переподписки, поскольку мы упираемся обычно не в общую полосу аплинков, а в скорость порта, который не успевает пропустить трафик, начинает его буферизировать, очереди переполнятся и пакеты дропаются. Но это уже совсем другая история.

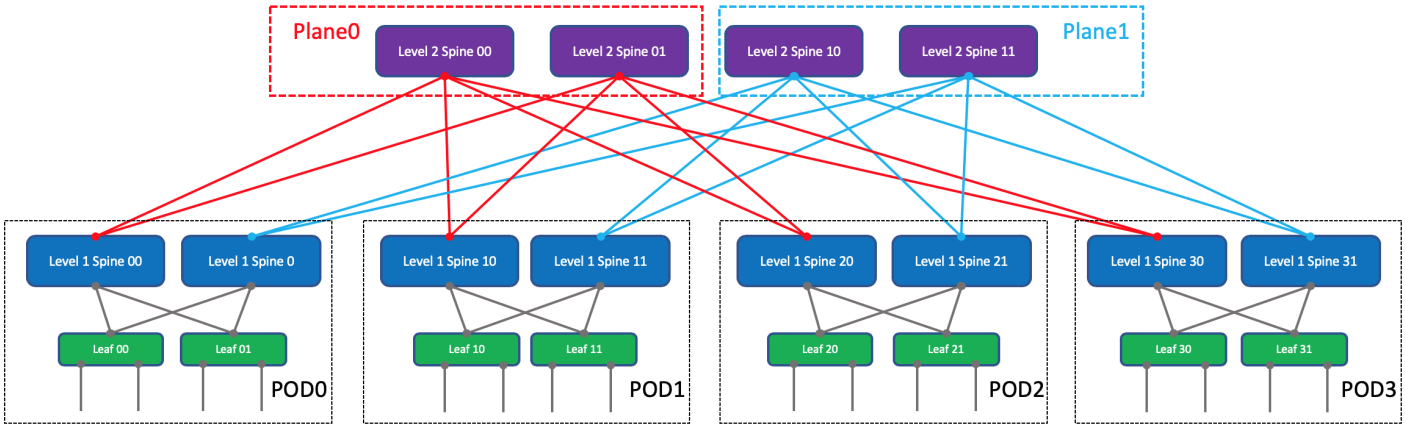
Многоуровневая сеть Клоза

Рассмотренная выше сеть Клоза - трёхуровневая или 3-stage: Input Switch - Middle Switch - Output Switch. Если в ней одни и те же лифы выполняют роль одновременно и передатчиков, и приёмников, её можно сложить пополам по линии Midle Switch. Получается более привычная нам двухуровневая сеть Leaf-Spine.

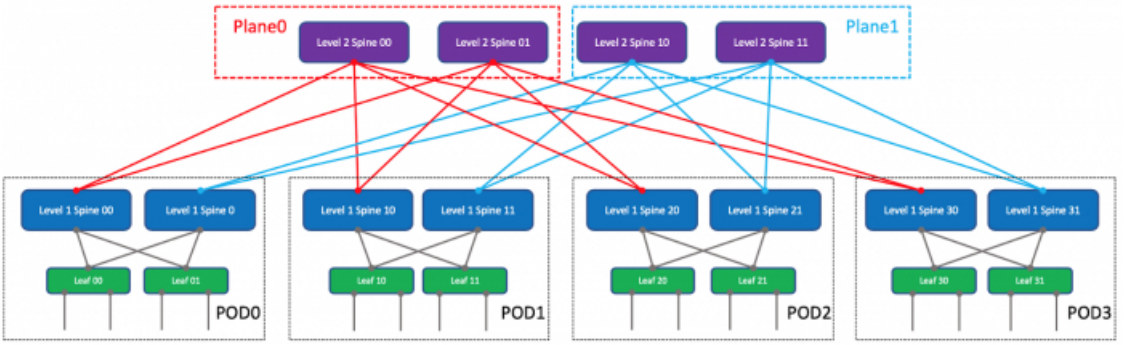
Пятиуровневая сеть Клоза будет выглядеть так: Input Switch - Middle Switch 1 - Middle Switch 2 - Middle Switch 3 - Output Switch. Вот она повёрнутая на 90 градусов:



Если её сложить по линии центральных Middle Switch, получается такая трёхуровневая сеть (folded clos):



Стоит дать пояснения:



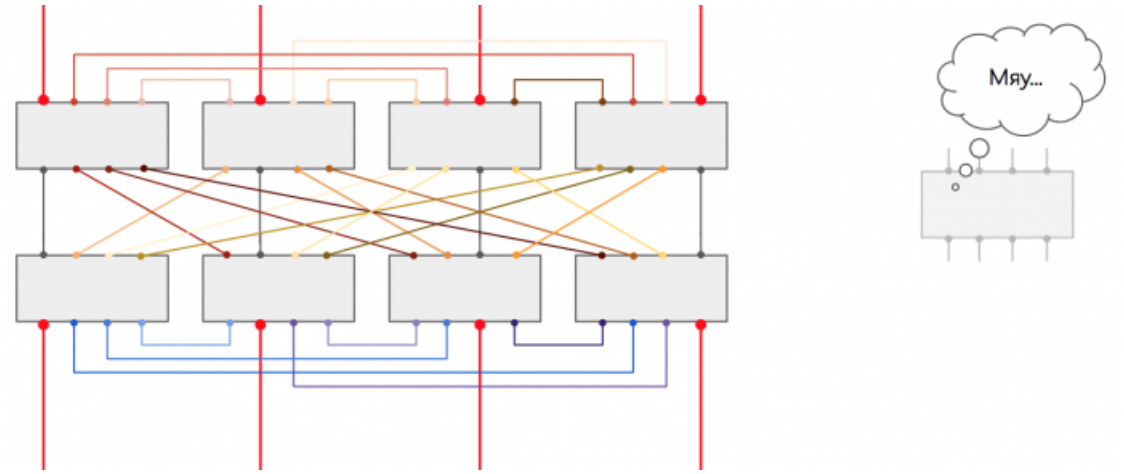
После сворачивания сети Клоза мы получили 4 обособленные группы - их называют **POD**"ами - **Point Of Delivery**. В каждой из них свой набор спайнов - это спайны первого уровня.

POD - это универсальная единица при строительстве датацентров - как кубик лего. Можно сказать, при заказе дополнительных мощностей, просто закупают новый POD и подключают его к новой фабрике. Спайны в одном POD"е далее соединены со спайнами в другом POD"е через спайны второго уровня. Но здесь уже не полная связность: не все спайны первого уровня включены во все спайны второго уровня - напротив, есть разделение по так называемым плоскостям - Plane. По одному спайну

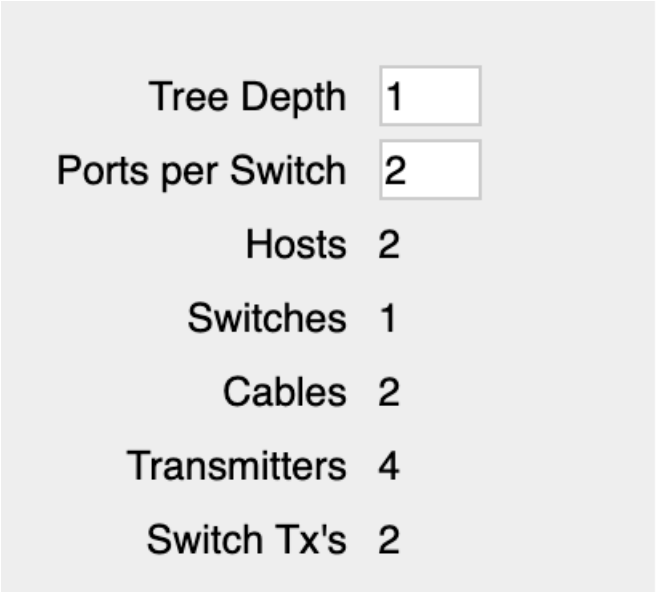
из каждого POD"а выводится в отдельную плоскость, в которой есть свой отдельный набор спайнов второго уровня. Сделано это не из вредности, а потому, что количество портов на спайнах ограничено и сделать топологию полносвязной просто не получится.

На иллюстрации выше - ограничение в 4 порта. И так по числу уровней можно расти дальше.

Сеть Клоза по задумке строится из одинаковых элементов, с одинаковым радиксом (количеством портов). И тут мы имеем ту же самую проблему, что и с чипами:



Чтобы обеспечить неблокируемую (без переподписки) фабрику, аплинки по пропускной способности должны быть такими же, как даунлинки. И тут появляется нюанс при масштабировании. Давайте разбираться. Так, если на коммутаторе всего 2 порта, можно включить всего 1 лиф и 2 машины (или сколько угодно лифов и всё равно 2 машины).





или

Tree Depth	2
Ports per Switch	2
Hosts	2
Switches	3
Cables	4
Transmitters	8
Switch Tx's	6

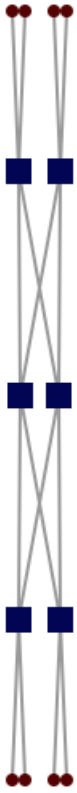
Fat Tree Visualizer
by He "Lonnie" Liu



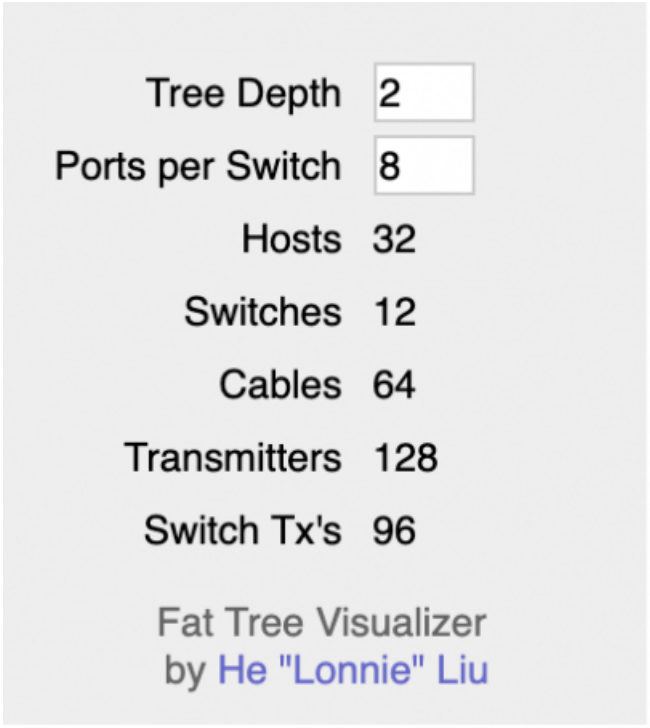
Если 4 порта, то 4 лифа по 2 даунлинка - 8 машин.

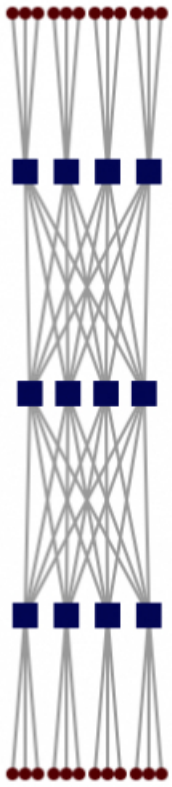
Tree Depth	2
Ports per Switch	4
Hosts	8
Switches	6
Cables	16
Transmitters	32
Switch Tx's	24

Fat Tree Visualizer
by He "Lonnie" Liu



8 - 8 лифов по 4 даунлинка - и 32 машины.





Более близкий к реальности сценарий - 32 порта - 32 лифа по 16 даунлинков - 512 машин.

[Как построить Гугл]

То есть, от числа портов напрямую зависит число машин, которое возможно подключить. При дальнейшем масштабировании всё больше растёт число спайнов. Это при трёхуровневой сети Клоза (лиф-спайн-лиф). Однако если добавить ещё один уровень, то станет дышаться легче:

[Как построить Гугл]

Тогда при 4 портах можно подключить 8 лифов и 16 машин, а при 8 - 128.

[Как построить Гугл]

32-хпортовые свитчи тогда позволят подключить 8192 хоста - в 16 раз больше, чем в трёхуровневой сети Клоза.

[Как построить Гугл]

Посоветую вот этот простой, но изящный инструмент, который рассчитает количество и возможных хостов, и необходимых свитчей, и кабелей. Я склонил репу себе, и вы можете поиграться прямо сейчас. А ещё детальные вычисления для сети Клоза даны тут.

Реальный мир, как обычно, посложнее. Всё же, на самом деле, мы не используем одни и те же коммутаторы на всех уровнях. Типичный тор/лиф сегодня в крупных ДЦ - это 48x10/25G-даунлинков и 6-8x100G-аплинков. Типичный спайн - от 32 до 128x100G. Тут арифметика уже другая. Посчитайте сами. Но очевидно, что подключить в этом случае можно гораздо больше машин.

3-Tier, Fat Tree, Clos, Folded Clos, Leaf-Spine?! Aaaa!

Если у вас такая же реакция, то вам в эту главу! Всё достаточно просто с последними тремя.

Clos - Топология Клоза - это теоретические выкладки о том, как малыми усилиями построить сеть без блокировок.

Folded Clos - свёрнутая топология Клоза - репрезентация топология Клоза, сложенной пополам, в которой входы - они же выходы, а выходы там же, где входы.

Leaf-Spine - это тот же Folded Clos, реализованный в виде сети датацентра.

А дальше посложнее. Складывается ощущение что единой терминологии тут не выработано. Пока я писал эту статью, пришлось 3 раза полностью переписывать некоторые её части, потому что менялось отношение к терминам. В одних источниках **Fat Tree** синонимируется с топологией Клоза, в других - с 3-Tier. Я неоднократно встречал статьи, где современному Leaf-Spine противопоставляется классический Fat-Tree. Хотя в той же википедии Fat Tree изображён в виде Клоза, или в RFC 7938 говорится об их синонимичности. В общем, предлагаю не поддаваться соблазну оставить вопрос без решения. Отныне и навсегда: есть классический 3-Tier: Access, Aggregation, Core - а есть Fat-Tree - он же Leaf-Spine. Но есть и другие. Принимаю аргументированные претензии в комментариях.

Маршрутизация

Для маршрутизации в Клозе стандартом де-факто стал BGP. Почему?

На эту тему есть целый RFC имени Facebook"а и Arista, где рассказывается, как строить **очень крупные** сети датацентров, используя BGP. Читается почти как художественный роман, очень рекомендую для томного вечера. В нём и про топологии Клоза, и про L3-дизайны, и про BGP в качестве единственного протокола маршрутизации, и даже про OPEX с CAPEX"ом. В общем, мне Пётр Лапухов (он один из авторов) идею продал.

Что такое крупный? Это тысячи стоек в одном датацентре, то есть, это тысячи сетевых устройств. Даже при разбиении на OSPF-зоны или ISIS (запрещённый в Российской Федерации протокол маршрутизации)-level"ы, вопросы уместности здесь Link-State протоколов остаются: политики маршрутизации, агрегация маршрутов, а если у нас VRF, а если хочется распространять не только IPv4-unicast маршруты?

LS IGP рассылает обновления на всю зону при любых изменениях линков, в то время как EBGP на месте выбирает лучший маршрут, и если ничего не меняется, то и помалкивает.

Ещё одна особенность IGP-протоколов - периодический фладинг LS-информацией. OSPF раз в 30 минут будет рассылать многие тысячи сообщений, напрягая CPU. Учитывая плотность линков на фабрике, одни и те же LSA много раз проходят по сети. LS IGP получается слишком говорливым. При этом, от BGP как протокола сигнализации между датацентрами, скорее всего, никуда не денешься. А BGP супер-масштабируемый, расширяемый протокол, который уже показал свою эффективность на самой крупной сети нашей Солнечной системы. Десятки и сотни тысяч (даже миллионы) маршрутов он ест на завтрак. Как раз то, что нужно. Тем более мы сокращаем стек используемых технологий - везде BGP.

Если у вас есть вопросы к сходимости BGP, то тут в наши дни всё тоже неплохо: таймеры можно выкрутить, BFD или BGP Next-hop tracking включить, PIC опять же. Время распространения информации при изменении топологии - примерно такое же, что и у LS IGP. Количество информации тоже как минимум не больше - нет периодической рассылки LSDB - только реальные события инициируют сообщения BGP Update. Кроме того, домен распространения обновлений можно сократить агрегацией маршрутов.

Стоит вспомнить, как в середине СДСМ мы наивно смеялись над идеей BGP в качестве IGP, а теперь пришлось к ней обратиться.

Хотя, безусловно, у BGP есть свои недостатки. По сравнению с LS IGP это, безусловно, отсутствие автообнаружения соседей. При настройке администратору нужно указать их в конфигурации, соотвественно держать где-то об этом информацию. Хотя в качестве контраргумента можно привести в пример BGP Logical Link Discovery Protocol и попытки Mellanox облегчить эту задачу.

Во-вторых, это вычисление наилучшего маршрута. В BGP оно несколько грубовато - на основе длины AS Path, Local Preference и MED. Учёта стоимости линков нет (AIGP не работает, потому что IGP нет). В случае датацентровых сетей это не звучит большой проблемой - обычно всё-таки все линки одной полосы и одной стоимости. Хотя для проведения работ, например, чтобы вывести узел из эксплуатации не получится просто задрать кост на интерфейсах - нужно через политики пессимизировать маршруты или навешивать Gshut community.

Итак, IGP не рассматривается как протокол для больших датацентров в принципе. BGP - на сегодня стандарт, но со своими ограничениями. Главное из них - это возможность блэкхола трафика при агрегации маршрутов. Я об этом напишу чуть позже.

А есть ли альтернативы? И да, и нет. Последние несколько лет активно развивается RIFT - Routing In Fat Trees. Все протоколы маршрутизации были разработаны специально для сетей с произвольной топологией и не очень высокой степенью связности между сетевыми узлами. Топология Клоза же напротив является регулярной и плотной. Умные дядьки вдруг решили разрушить полярный мир LS vs DV и смешали их: LS в сторону спайнов, DV в сторону лифов. Детальная информация распространяется только вверх - не вбок и не вниз. Таким образом каждый уровень сети знает все

детали об уровнях ниже, но не про уровни выше. На лифах получается только дефолт.

Кроме того, есть механизм автоматической дизагрегации, который борется с блэкхолом при отказах линков. В ситуации, которую я привёл в секции недостатокв, он начинает анонсировать специфики вниз фабрики, чтобы выбрался путь только через те устройства, которые действительно знают как добраться до адресата. Это объяснение того, почему альтернативы есть. А нет их, потому что RIFT на сегодняшний день реализован на полутора моделях устройств. Более того, несмотря на его плюсы, не все эксперты так уж оптимистично настроены по поводу его будущего. Ну а мы посмотрим.

Минусы сети Клоза

Да, да, они есть, несмотря на всю простоту и красоту подхода.

Во-первых, это количество необходимых кабелей, портов и свитчей. Каждый новый лиф должен быть подключен во все спайны. А каждый новый спайн - во все лифы. Хотя это является естественным свойством этой топологии, и архитектор к нему заведомо готов.

Во-вторых, максимальное количество хостов определено уже на моменте проектирования. Если их понадобится больше, то нужно строить новый кластер/датацентр и соединять его с первым. Очевидно, что это будут две разных инсталляции со сравнительно узкими линками между друг другом, а это автоматически означает, что приложения, активно нагружающие сеть, тоже должны запускаться отдельно в этих кластерах. Если всё-таки для приложений важно, чтобы это был один кластер, то нужно заранее продумывать какое число уровней фабрики будет нужно. А это всегда дополнительные расходы, причём, весьма значительные. И отсюда вытекает следующий недостаток.

В-третьих, фабрика, основанная на топологии Клоза, должна быть построена почти в полном объёме ещё до того, как туда запустят первые сервисы.

[Как построить Гугл]

Например, тут вы не можете не построить одну плоскость, как не можете не поставить один из спайнов в плейне. С одной стороны, это деградация пропускной способности, с другой - отсутствие резервирования, если речь идёт о двух устройствах. Я немного лукавлю, конечно: не всегда нужна сразу вся запланированная пропускная способность, поэтому можно запланировать 32 плейна, а начать с 4 и плавно добавлять их по мере необходимости.

Но! При таких масштабах совсем иначе стоит вопрос коммутаций - это, на самом деле, не самые простые задачи проектирования и экономики. Многие сотни километров оптики должны быть проложены ещё в момент строительства. И на этом фоне, стоимость самих коробок может быть гомеопатически малой. То есть, к моменту запуска нагрузки, сеть должна быть уже практически полностью построена.

В-четвёртых (впрочем это сложно назвать недостатком, скорее особенностью), это маршрутизация. Мы тут используем горячо любимый BGP. И он хорош. Но сценарии отказа и распространения маршрутной информации нужно продумывать очень тщательно. На большой сети мы не можем позволить везде бегать спецификам с торов. Во-первых, их может быть слишком много, что может вызывать исчерпание ресурсов FIB (например, ECMP-групп). Во-вторых, малейшие флапы линков будут вызывать наводнение всей сети обновлениями. Агрегирование маршрутов позволяет решить обе сложности, однако создаёт новую: блэкхолинг трафика. Вот ситуация, которая к нему приведёт:

[Как построить Гугл]

- За Leaf 00 живёт хост 10.0.0.1.
- Leaf 00 анонсирует маршрут 10.0.0.0/24 на Level 1 Spine 00 (и на Spine 01, конечно).
- Рвётся линк между Leaf 00 и Spine 00.
- Маршрут 10.0.0.0/24 исчезает.
- Leaf 01 продолжает анонсировать маршрут 10.0.1.0/24 на Spine 00, зажигая агрегат 10.0.0.0/20.
- Level1 Spine 00 анонсирует этот агрегат на Level2 Spine 00.
- А тот его переанонсирует в другой POD.
- POD1 имеет 4 маршрута 10.0.0.0/20 - через спайны второго уровня: Spine 00, 01, 02, 03.
- Хосты отправляют трафик на DIP: 10.0.0.1.
- Этот трафик балансируется по 4 путям.
- Тот трафик, который приходит на Level 2 Spine 00 далее попадает только на Level 1 Spine 00. Однако на нём нет маршрута к специфику 10.0.0.0/24, а агрегат указывает в Null, поскольку на этом устройстве и зарождён.

То есть, при наличии нескольких рабочих путей, в этом случае четверть трафика будет теряться. Это особенность Клоза: при движении на север есть много путей, а на юг - всегда только один. И с агрегацией нужно быть предельно аккуратным.

Альтернативные топологии

И я тут не про звезду и full mesh. Существуют топологии, в которых можно построить сеть с той же полосой и на то же количество серверов, что и Клоз, но обойтись меньшим количеством свитчей и кабелей. Мы точно не будем здесь развивать эту тему, я просто дам вам отправные точки:

- Flattened butterfly
- Jellyfish
- Dragonfly
- Любопытный сравнительный обзор разных топологий.

Стоит лишь помнить, что это топологии, для которых ещё не разработаны протоколы Control Plane"а. Однако говорят, что их уже используют в HPC - если у вас есть тому свидетельства, буду рад добавить в статью.

Выбор оборудования

Тут, наверняка, нет универсального совета. Но мы попробуем порассуждать. И стоит вспомнить о промежуточном дизайне сети ДЦ, который был между 3-Tier и полноценным многоуровневым Клозом, построенным из однотипных элементов.

Как только начали переходить на L3, сразу появился соблазн использвовать ECMP и ставить много (больше, чем два) агрегирующих устройств. Так, например, в Facebook появились кластера - набор стоек с ToR"ами, подключенными в четыре кластерных свитча. Кластера подразумевали сотни стоек, то есть сотни портов на кластерных свитчах. В те годы это могли быть только большие модульные шкафы с линейными картами, фабриками коммутации, отдельными платами управления, выжирающие электропитание и греющие горячий коридор. С такими модульными коммутаторами есть несколько проблем:

Во-первых, такие устройства имеют сложную внутреннюю архитектуру. Так, каждая линейная карта имеет, как минимум, один чип коммутации (а то и два), свой чип Traffic Management, подключение к общей шине. Фабрики коммутации сами по себе сложны по количеству взаимосвязанных компонентов, хотя и не выполняют интеллектуальные функции. Над всем этим есть VoQ, какие-то общие на коробку диспетчеры. Control Plane работает на управляющей плате и должен синхронно и гарантировано доставить данные в Data Plane всех линейных карт. Эта архитектура, как и управляющий софт, закрыты и крайне сложны для отладки.

Во-вторых, именно из-за наличия такого количества чипов, вероятность того, что где-то начнутся потери - она не только есть, но такие ситуации, увы, часто случаются. Самое в них неприятное, что они крайне тяжело отлавливаются, потому что никак не дают о себе знать - в ошибках на интерфейсах всё чисто, дропы в очередях QoS тоже по нулям, в логах пусто. Да и связность не полностью пропадает - она или с потерями или всё в целом нормально, но отсутствует связность между конкретными префиксами. Учитывая количество путей, по которым раскладывается трафик, отследить где именно и почему что-то теряется, становится задачей нетривиальной и, зачастую, сводящейся к тому, чтобы поочерёдно перезагружать подозрительные устройства (поднимите руки, кому не приходилось этого делать).

Вызваны они бывают как внешниеми факторами, так и внутренними. На моей памяти было два случая, когда происходило замятие контактов на задней части платы, которая вставляется в общую шину. Один раз достаточно сильный инженер засадил линейку в шасси, не заметив, что к контактам прилипла гайка - не только замял контакты, но и сжёг шину замыканием.

Бывают, наборот: слабые физически инженеры, которые недовставят плату, и часть SerDes"ов не работают. Отсюда и частая рекомендация TAC - вытащить и вставить. Гораздо более частая причина - неверно запрограммированный FIB - это уже косяк ПО. Когда запись прогружается из RIB в FIB, что-то может пойти не так, и запись не доедет до чипа.

Спрашивайте своего вендора, как проверять актуальность FIB, причём именно аппаратного - есть ли интересующая вас запись в чипе, а не просто в оперативной памяти линейной карты. В наше время девятинанометровых техпроцессов случаются ситуации, когда бит в ячейке памяти меняет своё значение почти спонтанно. Условно, альфа-частица во время солнечной активности или от куска урана, распадающегося на соседнем юните в стойке, может пролететь через чип и привести к этому. Это повреждает записанную в ячейке информацию или инструкцию и ломает передачу данных.

А **в-третьих**, такие гигантские коробки производит всего несколько вендоров. Само по себе это, возможно, и не является проблемой, но привязанность к производителю может выйти боком. Так или иначе, через подобный дизайн проходили все.

Модная тенденция этой осени - строить фабрику на пицца-боксах - одночиповых коммутаторах

небольшого размера - от одного до четырёх рэк-юнитов. Это в определённом смысле очень выигрышный подход. С одной стороны он позволяет убрать всю эту скрытую архитектурную сложность модульных шасси - если одночиповая коробка ломается, то ломается целиком, а не частично (впрочем, есть у меня для вас пара историй). С другой - они маленькие, дешёвые и производительные - то, что надо в парадигме горизонтального масштабирования. Очень легко докупается ещё одна коробочка на 64x100G порта в качестве спайна для расширения пропускной способности.

И тут как раз происходит смещение с подхода с кластерными свитчами на полноценную 5- (и больше) уровневую топологию Клоза. Сегодня можно найти фантастические по меркам пятилетней давности штуковины. Например, на ToR 48x25G+6x100G, как вам? А 128x100G, умещённые в 4 юнита? 12,8 Тб/с на одном чипе - шутка ли? До едва ли не прошлого года, был здесь нюанс с тем, что производитель чипов для датацентровых коммутаторов был ровно один - Broadcom, заваливший рынок своими Tomahawk/Tomohawk+/2, Trident1/2/3 и Jericho. Сегодня стали появляться мощные конкуренты - Mellanox со своими Spectrum"ами, Barefoot с Tophino, Innovium - и стало уже можно выбирать, и даже немного манипулировать. Некоторые вендоры делают ставку на чипы собственного производства.

Но выбор чипа - это уже тема отдельной статьи.

Заключение

Картинка, как по мне, получается красивая. Но всегда следует помнить, что и инженеры, и админы, и девопсы, и архитекторы решают задачи бизнеса, а не диктуют как бизнесу подстраиваться под классную сеть, которую они придумали. Поэтому место найдётся и для топологий Клоза, и для Fat-Tree, и для MSTP в ядре (увы). Кстати, сеть Клоза на самом деле изобретена не Клозом, а Эдсоном Эрвином на 15 лет раньше, но кого это теперь волнует?

Полезные ссылки

Сети Клоза:

- Оригинальная публикация A study of non-blocking switching networks by Charles Clos. Натурально научный угар. Если не верите, вот вам скринпруф:
  [Как построить Гугл]
- Короткая вводная в сеть Клоза: Clos Networks: What"s Old Is New Again.
- Про сеть Клоза до мельчайших винтиков: Demystifying DCN Topologies: Clos/Fat Trees – Part1 и Part2.
- Поверхностный, но очень понятный разбор устройства сети Facebook: Introduction to Facebook"s data center fabric.
- Scaling Yandex Data Centers – Дмитрий Афанасьев, Яндекс.

О вреде God Boxes в ДЦ:

- Why is Clos (Spine/Leaf) the thing to do in the Data Center?

Подходы к маршрутизации в L3 ДЦ:

- Первый доклад Дмитрия Афанасьева о сетях в ДЦ: Next Hop 2019: конференция по сетям
- RFC 7938. Use of BGP for Routing in Large-Scale Data Centers
- Cumulus. BGP in the data center
- Про оптимизацию BGP: BGP Convergence Optimization

Альтернативные топологии:

- Application of Butterfly Clos-Network in Network-on-Chip
- Flattened butterfly: a cost-efficient topology for high-radix networks.
- Jellyfish: Networking Data Centers Randomly
- Dragonfly+: Low Cost Topology for Scaling Datacenters

Спасибы

- Андрею Глазкову aka @glazgoo за вычитку и правки
- Александру Клименко aka @v00lk за то, что перевернул взгляд на топологии