



@theGluck

Пользователь

57,0

карма

0,0

рейтинг



Профиль

1

Публикации

34

Комментарии

56

Избранное

54

Подписчики

14 мая 2012 в 09:21

Администрирование → Сети для самых маленьких. Часть четвертая. STP

tutorial

Сетевые технологии*, Системное администрирование*

I think that I shall never see

A graph more lovely than a tree.

A tree whose crucial property

Is loop-free connectivity.

A tree that must be sure to span

So packets can reach every LAN.

First, the root must be selected.

By ID, it is elected.

Least-cost paths from root are traced.

In the tree, these paths are placed.

A mesh is made by folks like me,

Then bridges find a spanning tree.

— Radia Joy Perlman

Все выпуски

6. Сети для самых маленьких. Часть шестая. Динамическая маршрутизация

5. Сети для самых маленьких: Часть пятая. NAT и ACL

4. Сети для самых маленьких: Часть четвёртая. STP

3. Сети для самых маленьких: Часть третья. Статическая маршрутизация

2. Сети для самых маленьких. Часть вторая. Коммутация

1. Сети для самых маленьких. Часть первая. Подключение к оборудованию cisco

0. Сети для самых маленьких. Часть нулевая. Планирование

В [прошлом выпуске](#) мы остановились на статической маршрутизации. Теперь надо сделать шаг в сторону и обсудить вопрос стабильности нашей сети.

Однажды, когда вы — единственный сетевой админ фирмы “Лифт ми Ап” — отпросились на полдня раньше, вдруг упала связь с серверами, и директора не получили несколько важных писем. После короткой, но ощутимой взбучки вы идёте разбираться, в чём дело, а оказалось, по чьей-то неосторожности выпал из разъёма единственный кабель, ведущий к коммутатору в серверной. Небольшая проблема, которую вы могли исправить за две минуты, и даже вообще избежать, существенно сказалась на вашем доходе в этом месяце и возможностях роста.

Итак, сегодня обсуждаем:

- проблему широковещательного шторма
- работу и настройку протокола STP и его модификаций (RSTP, MSTP, PVST, PVST+)
- технологию агрегации интерфейсов и перераспределения нагрузки между ними
- некоторые вопросы стабильности и безопасности
- как изменить схему существующей сети, чтобы всем было хорошо

Оборудование, работающее на втором уровне модели OSI (коммутатор), должно выполнять 3 функции: запоминание адресов, перенаправление (коммутация) пакетов, защита от петель в сети. Разберем по пунктам каждую функцию.

Запоминание адресов и перенаправление пакетов: Как мы уже говорили [ранее](#), у каждого свича есть таблица

Реклама

Хабы

[Администрирование баз данных](#)

[Системное администрирование](#)

[Виртуализация](#)

[Восстановление данных](#)

[Настройка Linux](#)

[Сетевые технологии](#)

[Хранение данных](#)

[Резервное копирование](#)

[Антивирусная защита](#)

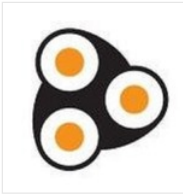
[Облачные вычисления](#)

Все хабы

Яндекс.Директ

[АО СПЕЦКАБЕЛЬ – кабели, муфты...](#)

[deltalogic.ru.com](#)



[Как лечить грибок ногтей? Можно..](#)

[hordmar.ru](#)



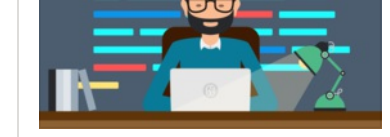
[Интересуешься программированием?](#)

[geekbrains.ru](#)



сопоставления MAC-адресов и портов (aka CAM-table — Content Addressable Memory Table). Когда устройство, подключенное к свичу, посылает кадр в сеть, свич смотрит MAC-адрес отправителя и порт, откуда получен кадр, и добавляет эту информацию в свою таблицу. Далее он должен передать кадр получателю, адрес которого указан в кадре. По идее, информацию о порте, куда нужно отправить кадр, он берёт из этой же CAM-таблицы. Но, предположим, что свич только что включили (таблица пуста), и он понятия не имеет, в какой из его портов подключен получатель. В этом случае он отправляет полученный кадр во все свои порты, кроме того, откуда он был принят. Все конечные устройства, получив этот кадр, смотрят MAC-адрес получателя, и, если он адресован не им, отбрасывают его. Устройство-получатель отвечает отправителю, а в поле отправителя ставит свой адрес, и вот свич уже знает, что такой-то адрес находится на таком-то порту (вносит запись в таблицу), и в следующий раз уже будет переправлять кадры, адресованные этому устройству, только в этот порт. Чтобы посмотреть содержимое CAM-таблицы, используется команда **show mac address-table**. Однажды попав в таблицу, информация не остаётся там пожизненно, содержимое постоянно обновляется и если к определенному мас-адресу не обращались 300 секунд (по умолчанию), запись о нем удаляется.

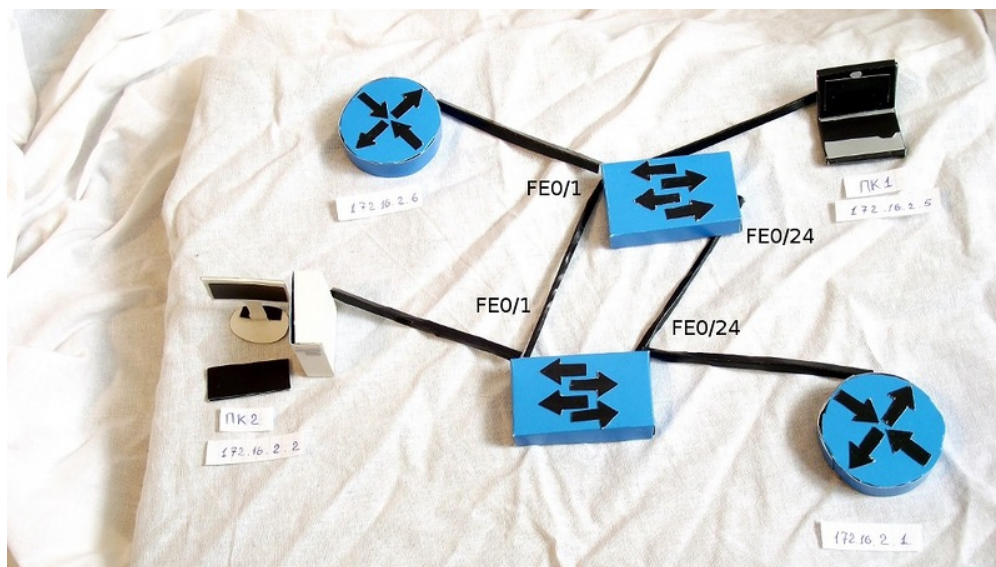
Тут всё должно быть понятно. Но зачем *защита от петель*? И что это вообще такое?



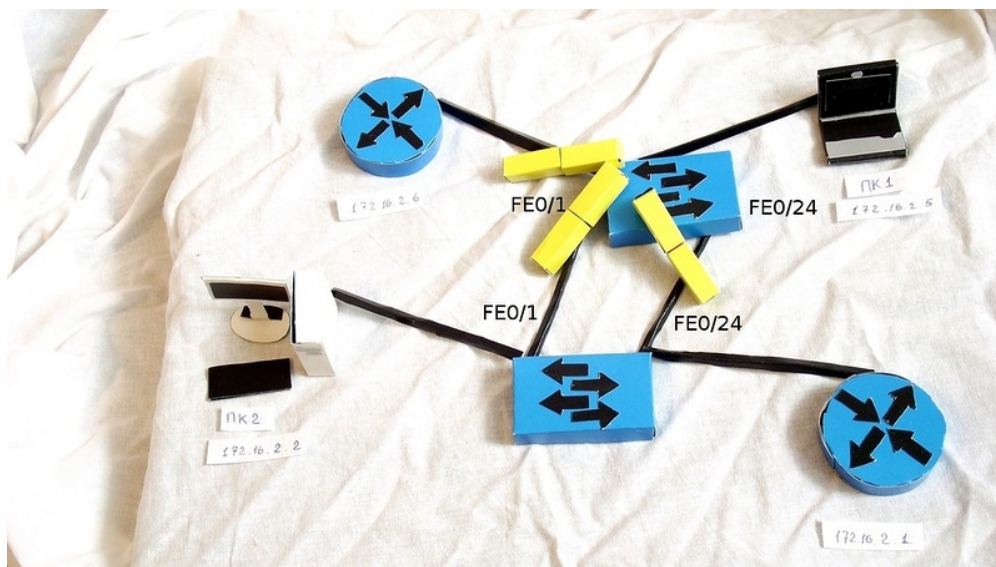
Реклама

Широковещательный шторм

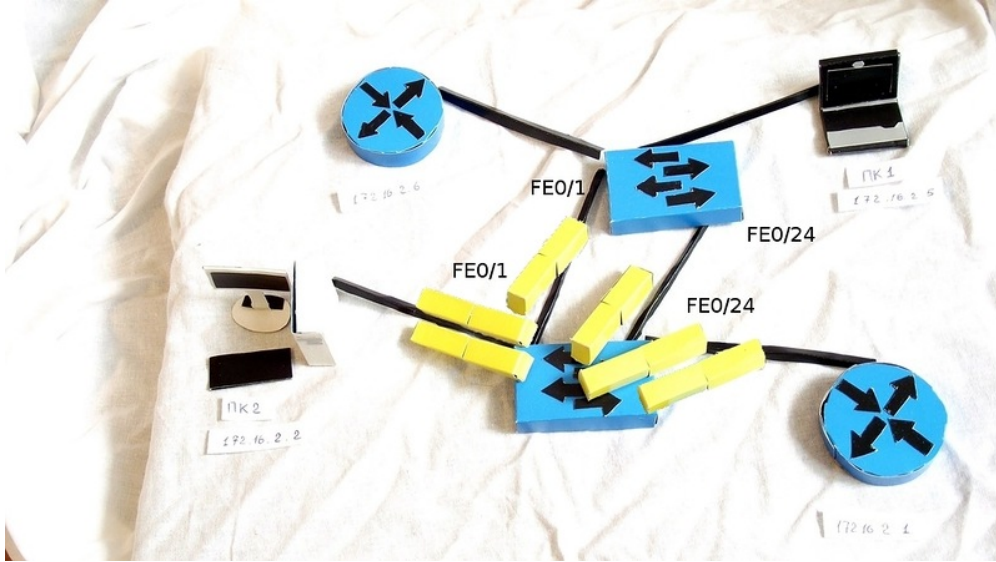
Часто, для обеспечения стабильности работы сети в случае проблем со связью между свичами (выход порта из строя, обрыв провода), используют избыточные линки (redundant links) — дополнительные соединения. Идея простая — если между свичами по какой-то причине не работает один линк, используем запасной. Вроде все правильно, но представим себе такую ситуацию: два свича соединены двумя проводами (пусть будет, что у них соединены fa0/1 и fa0/24).



Одной из их подопечных — рабочих станций (например, ПК1) вдруг приспичило послать широковещательный кадр (например, ARP-запрос). Раз широковещательный, шлем во все порты, кроме того, с которого получили.



Второй свич получает кадр в два порта, видит, что он широковещательный, и тоже шлет во все порты, но уже, получается, и обратно в те, с которых получил (кадр из fa0/24 шлет в fa0/1, и наоборот).



Первый свич поступает точно также, и в итоге мы получаем широковещательный шторм (broadcast storm), который намертво блокирует работу сети, ведь свичи теперь только и занимаются тем, что шлют друг другу один и тот же кадр.



Как можно избежать этого? Ведь мы, с одной стороны, не хотим штормов в сети, а с другой, хотим повысить ее отказоустойчивость с помощью избыточных соединений? Тут на помощь нам приходит STP (Spanning Tree Protocol)

STP

Основная задача STP — предотвратить появление петель на втором уровне. Как это сделать? Да просто отрубить все избыточные линки, пока они нам не понадобятся. Тут уже сразу возникает много вопросов: какой линк из двух (или трех-четырех) отрубить? Как определить, что основной линк упал, и пора включать запасной? Как понять, что в сети образовалась петля? Чтобы ответить на эти вопросы, нужно разобраться, как работает STP.

STP использует алгоритм STA (Spanning Tree Algorithm), результатом работы которого является граф в виде дерева (связный и без [простых циклов](#))

Для обмена информацией между собой свичи используют специальные пакеты, так называемые BPDU (Bridge Protocol Data Units). BPDU бывают двух видов: конфигурационные (Configuration BPDU) и панические "AAA, топология поменялась!" TCN (Topology Change Notification BPDU). Первые регулярно рассылаются корневым свичом (и ретранслируются остальными) и используются для построения топологии, вторые, как понятно из названия, отсылаются в случае изменения топологии сети (проще говоря, подключения/отключения свича). Конфигурационные BPDU содержат несколько полей, остановимся на самых важных:

- идентификатор отправителя (Bridge ID)
- идентификатор корневого свича (Root Bridge ID)
- идентификатор порта, из которого отправлен данный пакет (Port ID)
- стоимость маршрута до корневого свича (Root Path Cost)

Что все это такое и зачем оно нужно, объясню чуть ниже. Так как устройства не знают и не хотят знать своих соседей, никаких отношений (смежности/соседства) они друг с другом не устанавливают. Они шлют BPDU из всех работающих портов на мультикастовый ethernet-адрес **01-80-c2-00-00-00** (по умолчанию каждые 2 секунды), который прослушивают все свичи с включенным STP.

Итак, как же формируется топология без петель?

Сначала выбирается так называемый корневой мост/свич (root bridge). Это устройство, которое STP считает точкой отсчета, центром сети; все дерево STP сходится к нему. Выбор базируется на таком понятии, как идентификатор свича (Bridge ID). Bridge ID это число длиной 8 байт, которое состоит из Bridge Priority (приоритет, от 0 до 65535, по умолчанию 32768+номер vlan или инстанс MSTP, в зависимости от реализации протокола), и MAC-адреса устройства. В начале выборов каждый коммутатор считает себя корневым, о чем и заявляет всем остальным с помощью BPDU, в котором представляет свой идентификатор как ID корневого свича. При этом, если он получает BPDU с меньшим Bridge ID, он перестает хвастаться своим и покорно начинает анонсировать полученный Bridge ID в качестве корневого. В итоге, корневым оказывается тот свич, чей Bridge ID меньше всех.

Такой подход таит в себе довольно серьезную проблему. Дело в том, что, при равных значениях Priority (а они равные, если не менять ничего) корневым выбирается самый старый свич, так как мак адреса прописываются на производстве последовательно, соответственно, чем мак меньше, тем устройство старше (естественно, если у нас все оборудование одного вендора). Понятное дело, это ведет к падению производительности сети, так как старое устройство, как правило, имеет худшие характеристики. Подобное поведение протокола следует пресекать, выставляя значение приоритета на желаемом корневом свиче вручную, об этом в практической части.

Роли портов

После того, как коммутаторы померились айдами и выбрали root bridge, каждый из остальных свичей должен найти один, и только один порт, который будет вести к корневому свичу. Такой порт называется **корневым портом (Root port)**. Чтобы понять, какой порт лучше использовать, каждый некорневой свич определяет стоимость маршрута от каждого своего порта до корневого свича. Эта стоимость определяется суммой стоимостей всех линков, которые нужно пройти кадр, чтобы дойти до корневого свича. В свою очередь, стоимость линка определяется просто- по его скорости (чем выше скорость, тем меньше стоимость). Процесс определения стоимости маршрута связан с полем BPDU "Root Path Cost" и происходит так:

1. Корневой свич посылает BPDU с полем Root Path Cost, равным нулю
2. Ближайший свич смотрит на скорость своего порта, куда BPDU пришел, и добавляет стоимость согласно таблице

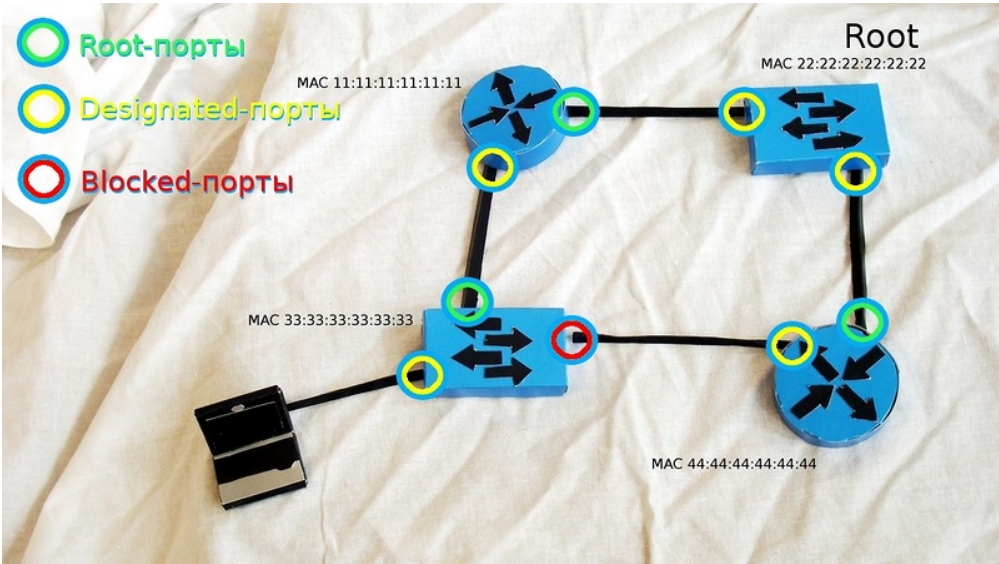
Скорость порта	Стоимость STP (802.1d)
10 Mbps	100
100 Mbps	19
1 Gbps	4
10 Gbps	2

3. Далее этот второй свич посылает этот BPDU нижестоящим коммутаторам, но уже с новым значением Root Path Cost, и далее по цепочке вниз

Если имеют место одинаковые стоимости (как в нашем примере с двумя свичами и двумя проводами между ними — у каждого пути будет стоимость 19) — корневым выбирается меньший порт.

Далее выбираются назначенные (**Designated**) порты. Из каждого конкретного сегмента сети должен существовать только один путь по направлению к корневому свичу, иначе это петля. В данном случае имеем в виду физический сегмент, в современных сетях без хабов это, грубо говоря, просто провод. Назначенным портом выбирается тот, который имеет лучшую стоимость в данном сегменте. У корневого свича все порты — назначенные.

И вот уже после того, как выбраны корневые и назначенные порты, оставшиеся блокируются, таким образом разрывая петлю.



*На картинке маршрутизаторы выступают в качестве коммутаторов. В реальной жизни это можно сделать с помощью дополнительной свитчевой платы.

Чуть раньше мы упомянули состояние блокировки порта, теперь поговорим о том, что это значит, и о других возможных состояниях порта в STP. Итак, в обычном (802.1D) STP существует 5 различных состояний:

- блокировка (blocking): блокированный порт не шлет ничего. Это состояние предназначено, как говорилось выше, для предотвращения петель в сети. Блокированный порт, тем не менее, слушает BPDU (чтобы быть в курсе событий, это позволяет ему, когда надо, разблокироваться и начать работать)
- прослушивание (listening): порт слушает и начинает сам отправлять BPDU, кадры с данными не отправляет.
- обучение (learning): порт слушает и отправляет BPDU, а также вносит изменения в CAM-таблицу, но данные не перенаправляет.
- перенаправление\пересылка (forwarding): этот может все: и посылает\принимает BPDU, и с данными оперирует, и участвует в поддержании таблицы mac-адресов. То есть это обычное состояние рабочего порта.
- отключен (disabled): состояние administratively down, отключен командой **shutdown**. Понятное дело, ничего делать не может вообще, пока вручную не включат.

Порядок перечисления состояний не случаен: при включении (а также при втыкании нового провода), все порты на устройстве с STP проходят вышеприведенные состояния именно в таком порядке (за исключением disabled-портов). Возникает закономерный вопрос: а зачем такие сложности? А просто STP осторожничает. Ведь на другом конце провода, который только что воткнули в порт, может быть свич, а это потенциальная петля. Вот поэтому порт сначала 15 секунд (по умолчанию) пребывает в состоянии прослушивания — он смотрит BPDU, попадающие в него, выясняет свое положение в сети — как бы чего ни вышло, потом переходит к обучению еще на 15 секунд — пытается выяснить, какие mac-адреса “в ходу” на линке, и потом, убедившись, что ничего он не ломает, начинает уже свою работу. Итого, мы имеем целых 30 секунд простоя, прежде чем подключенное устройство сможет обмениваться информацией со своими соседями. Современные компы грузятся быстрее, чем за 30 секунд. Вот комп загрузился, уже рвется в сеть, истерит на тему “DHCP-сервер, сволочь, ты будешь айпишник выдавать, или нет?”, и, не получив искомого, обижается и уходит в себя, извлекая из своих недр айпишник [автонастройки](#). Естественно, после таких экзерсисов, в сети его слушать никто не будет, ибо “не местный” со своим 169.254.x.x. Понятно, что все это не дело, но как этого избежать?

Portfast

Для таких случаев используется особый режим порта — portfast. При подключении устройства к такому порту, он, минуя промежуточные стадии, сразу переходит к forwarding-состоянию. Само собой, portfast следует включать только на интерфейсах, ведущих к конечным устройствам (рабочим станциям, серверам, телефонам и т.д.), но не к другим свичам.

Есть очень удобная команда режима конфигурации интерфейса для включения нужных фиш на порту, в который будут включаться конечные устройства: **switchport host**. Эта команда разом включает PortFast, переводит порт в режим access (аналогично switchport mode access), и отключает протокол PAgP (об этом протоколе подробнее в разделе агрегация каналов).

Виды STP

STP довольно старый протокол, он создавался для работы в одном LAN-сегменте. А что делать, если мы хотим внедрить его в нашей сети, которая имеет несколько VLANов?

Стандарт 802.1Q, о котором мы упоминали в статье о [коммутации](#), определяет, каким образом вланы передаются внутри транка. Кроме того, он определяет один процесс STP для всех вланов. BPDU по транкам передаются нетегированными (в native VLAN). Этот вариант STP известен как **CST** (Common Spanning Tree). Наличие только одного процесса для всех вланов очень облегчает работу по настройке и разгружает процессор свича, но, с другой стороны, CST имеет недостатки: избыточные линки между свичами блокируются во всех вланах, что не всегда приемлемо и не дает возможности использовать их для балансировки нагрузки.

Cisco имеет свой взгляд на STP, и свою проприетарную реализацию протокола — **PVST** (Per-VLAN Spanning Tree) — которая предназначена для работы в сети с несколькими VLAN. В PVST для каждого влана существует свой процесс STP, что позволяет независимую и гибкую настройку под потребности каждого влана, но самое главное, позволяет использовать балансировку нагрузки за счет того, что конкретный физический линк может быть заблокирован в одном влане, но работать в другом. Минусом этой реализации является, конечно, проприетарность: для функционирования PVST требуется проприетарный же ISL транк между свичами.

Также существует вторая версия этой реализации — **PVST+**, которая позволяет наладить связь между свичами с CST и PVST, и работает как с ISL-транком, так и с 802.1q. PVST+ это протокол по умолчанию на коммутаторах Cisco.

RSTP

Все, о чем мы говорили ранее в этой статье, относится к первой реализации протокола STP, которая была разработана в 1985 году Радией Перлман (ее стихотворение использовано в качестве эпиграфа). В 1990 году эта реализация была включена в стандарт IEEE 802.1D. Тогда время текло медленнее, и перестройка топологии STP, занимающая 30-50 секунд (!!!), всех устраивала. Но времена меняются, и через десять лет, в 2001 году, IEEE представляет новый стандарт **RSTP** (он же 802.1w, он же Rapid Spanning Tree Protocol, он же Быстрый STP). Чтобы структурировать предыдущий материал и посмотреть различия между обычным STP (802.1d) и RSTP (802.1w), соберем таблицу с основными фактами:

STP (802.1d)	RSTP (802.1w)
В уже сложившейся топологии только корневой свич шлет BPDU, остальные ретранслируют	Все свичи шлют BPDU в соответствии с hello-таймером (2 секунды по умолчанию)
Состояния портов	
<ul style="list-style-type: none"> — блокировка (blocking) — прослушивание (listening) — обучение (learning) — перенаправление/пересылка (forwarding) — отключен (disabled) 	<ul style="list-style-type: none"> — отбрасывание (discarding), заменяет disabled, blocking и listening — learning — forwarding
Роли портов	
<ul style="list-style-type: none"> — корневой (root), участвует в пересылке данных, ведет к корневому свичу — назначенный (designated), тоже работает, ведет от корневого свича — неназначенный (non-designated), не участвует в пересылке данных 	<ul style="list-style-type: none"> — корневой (root), участвует в пересылке данных — назначенный (designated), тоже работает — дополнительный (alternate), не участвует в пересылке данных — резервный (backup), тоже не участвует
Механизмы работы	
Использует таймеры: Hello (2 секунды) Max Age (20 секунд) Forward delay timer (15 секунд)	Использует процесс proposal and agreement (предложение и соглашение)
Свич, обнаруживший изменение топологии, извещает корневой свич, который, в свою очередь, требует от всех остальных очистить их записи о текущей топологии в течение forward delay timer	Обнаружение изменений в топологии влечет немедленную очистку записей
Если не-корневой свич не получает hello- пакеты от корневого в течение Max Age, он начинает новые выборы	Начинает действовать, если не получает BPDU в течение 3 hello-интервалов
Последовательное прохождение порта через состояния Blocking (20 сек) — Listening (15 сек) — Learning (15 сек) — Forwarding	Быстрый переход к Forwarding для p2p и Edge-портов

Как мы видим, в RSTP остались такие роли портов, как корневой и назначенный, а роль заблокированного разделили на две новых роли: Alternate и Backup. Alternate — это резервный корневой порт, а backup — резервный назначенный порт. Как раз в этой концепции резервных портов и кроется одна из причин быстрого переключения в случае отказа. Это меняет поведение системы в целом: вместо реактивной (которая начинает искать решение проблемы только после того, как она случилась) система становится проактивной, заранее просчитывающей “пути отхода” еще до появления проблемы. Смысл простой: для того, чтобы в случае отказа основного переключится на резервный линк, RSTP не нужно заново просчитывать топологию, он просто переключится на запасной, заранее просчитанный.

Ранее, для того, чтобы убедиться, что порт может участвовать в передаче данных, требовались таймеры, т.е. свич пассивно ждал в течение означенного времени, слушая BPDU. Ключевой фичей RSTP стало введение концепции типов портов, основанных на режиме работы линка- full duplex или half duplex (типы портов p2p или shared, соответственно), а также понятия пограничный порт (тип edge p2p), для конечных устройств. Пограничные порты назначаются, как и раньше, командой spanning-tree portfast, и с ними все понятно- при включении провода сразу переходим к forwarding-состоянию и работаем. Shared-порты работают по старой схеме с прохождением через состояния BLK — LIS — LRN — FWD. А вот на p2p-портах RSTP использует процесс предложения и соглашения (proposal and agreement). Не вдаваясь в подробности, его можно описать так: свич справедливо считает, что если линк работает в режиме полного дуплекса, и он не обозначен, как пограничный, значит, на нем только два устройства- он и другой свич. Вместо того, чтобы ждать входящих BPDU, он сам пытается связаться со свичом на том конце провода с помощью специальных proposal BPDU, в которых, конечно, есть информация о стоимости маршрута к корневому свичу. Второй свич сравнивает полученную информацию со своей текущей, и принимает решение, о чем извещает первый свич посредством agreement BPDU. Так как весь этот процесс теперь не привязан к таймерам, происходит он очень быстро- только подключили новый свич- и он практически сразу вписался в общую топологию и приступил к работе (можете сами оценить скорость переключения в сравнении с обычным STP на видео). В Cisco-мире RSTP называется PVRST (Per-Vlan Rapid Spanning Tree).

MSTP

Чуть выше, мы упоминали о PVST, в котором для каждого влана существует свой процесс STP. Вланы это довольно удобный инструмент для многих целей, и поэтому, их может быть достаточно много даже в некрпной организации. И в случае PVST, для каждого будет рассчитываться своя топология, тратиться процессорное время и память свичей. А нужно ли нам рассчитывать STP для всех 500 вланов, когда единственное место, где он нам нужен- это резервный линк между двумя свичами? Тут нас выручает MSTP. В нем каждый влан не обязан иметь собственный процесс STP, их можно объединять. Вот у нас есть, например, 500 вланов, и мы хотим балансировать нагрузку так, чтобы половина из них работала по одному линку (второй при этом блокируется и стоит в резерве), а вторая- по другому. Это можно сделать с помощью обычного STP, назначив один корневой свич в диапазоне вланов 1-250, а другой- в диапазоне 250-500. Но процессы будут работать для каждого из пятисот вланов по отдельности (хотя действовать будут совершенно одинаково для каждой половины). Логично, что тут хватит и двух процессов. MSTP позволяет создавать столько процессов STP, сколько у нас логических топологий (в данном примере- 2), и распределять по ним вланы. Думаем, нет особого смысла углубляться в теорию и практику MSTP в рамках этой статьи (ибо теории там ого-го), интересующиеся могут пройти по [ссылке](#).

Агрегация каналов

Но какой бы вариант STP мы не использовали, у нас все равно существует так или иначе неработающий линк. А

возможно ли задействовать параллельные линки по полной и при этом избежать петель? Да, отвечаем мы вместе с циской, начиная рассказ о EtherChannel.

Иначе это называется link aggregation, link bundling, NIC teaming, port trunking

Технологии агрегации (объединения) каналов выполняют 2 функции: с одной стороны, это объединение пропускной способности нескольких физических линков, а с другой — обеспечение отказоустойчивости соединения (в случае падения одного линка нагрузка переносится на оставшиеся). Объединение линков можно выполнить как вручную (статическое агрегирование), так и с помощью специальных протоколов: LACP (Link Aggregation Control Protocol) и PAgP (Port Aggregation Protocol). LACP, определяемый стандартом IEEE 802.3ad, является открытым стандартом, то есть от вендора оборудования не зависит. Соответственно, PAgP — проприетарная цисковская разработка.

В один такой канал можно объединить до восьми портов. Алгоритм балансировки нагрузки основан на таких параметрах, как IP/MAC-адреса получателей и отправителей и порты. Поэтому в случае возникновения вопроса: “Хей, а чего так плохо балансируется?” в первую очередь смотрите на алгоритм балансировки.

Тема агрегации каналов заслуживает отдельной статьи, а то и книги, поэтому углубляться не будем, интересующимся- [ссылка](#).

Port security

Теперь расскажем вкратце, как обеспечить безопасность сети на втором уровне OSI. В этой части статьи теория и практическая конфигурация совмещены. Увы, Packet Tracer не умеет ничего из упомянутых в этом разделе команд, поэтому все без иллюстраций и проверок.

Для начала, следует упомянуть команду конфигурации интерфейса **switchport port-security**, включающую защиту на определенном порту свича. Затем, с помощью **switchport port-security maximum 1** мы можем ограничить количество mac-адресов, связанных с данным портом (т.е., в нашем примере, на данном порту может работать только один mac-адрес). Теперь указываем, какой именно адрес разрешен: его можно задать вручную **switchport port-security mac-address *адрес***, или использовать волшебную команду **switchport port-security mac-address sticky**, закрепляющую за портом тот адрес, который в данный момент работает на порту. Далее, задаем поведение в случае нарушения правила **switchport port-security violation {shutdown | restrict | protect}**: порт либо отключается, и потом его нужно поднимать вручную (shutdown), либо отбрасывает пакеты с незарегистрированного мака и пишет об этом в консоль (restrict), либо просто отбрасывает пакеты (protect).

Помимо очевидной цели — ограничение числа устройств за портом — у этой команды есть другая, возможно, более важная: предотвращать атаки. Одна из возможных — истощение CAM-таблицы. С компьютера злодея рассылается огромное число кадров, возможно, широковестьных, с различными значениями в поле MAC-адрес отправителя. Первый же коммутатор на пути начинает их запоминать. Одну тысячу он запомнит, две, но память-то оперативная не резиновая, и среднее ограничение в 16000 записей будет довольно быстро достигнуто. При этом дальнейшее поведение коммутатора может быть различным. И самое опасное из них с точки зрения безопасности: коммутатор может начать все кадры, приходящие на него, рассылать, как широковестьные, потому что MAC-адрес получателя не известен (или уже забыт), а запомнить его уже просто некуда. В этом случае сетевая карта злодея будет получать все кадры, летающие в вашей сети.

DHCP Snooping

Другая возможная атака нацелена на DHCP сервер. Как мы знаем, DHCP обеспечивает клиентские устройства всей нужной информацией для работы в сети: ip-адресом, маской подсети, адресом шлюза по умолчанию, DNS-сервера и прочим. Атакующий может поднять собственный DHCP, который в ответ на запрос клиентского устройства будет отдавать в качестве шлюза по умолчанию (а также, например, DNS-сервера) адрес подконтрольной атакующему машины. Соответственно, весь трафик, направленный за пределы подсети обманутыми устройствами, будет доступен для изучения атакующему — типичная man-in-the-middle атака. Либо такой вариант: подлый мошенник генерирует кучу DHCP-запросов с поддельными MAC-адресами и DHCP-сервер на каждый такой запрос выдаёт IP-адрес до тех пор, пока не истощится пул.

Для того, чтобы защититься от подобного вида атак, используется фишка под названием DHCP snooping. Идея совсем простая: указать свичу, на каком порту подключен настоящий DHCP-сервер, и разрешить DHCP-ответы только с этого порта, запретив для остальных. Включаем глобально командой **ip dhcp snooping**, потом говорим, в каких вланах должно работать **ip dhcp snooping vlan номер(a)**. Затем на конкретном порту говорим, что он может пренаправлять DHCP-ответы (такой порт называется доверенным): **ip dhcp snooping trust**.

IP Source Guard

После включения DHCP Snooping’a, он начинает вести у себя базу соответствия MAC и IP-адресов устройств, которую обновляет и пополняет за счет прослушивания DHCP запросов и ответов. Эта база позволяет нам противостоять еще одному виду атак — подмене IP-адреса (IP Spoofing). При включенном IP Source Guard, каждый приходящий пакет может проверяться на:

- соответствие IP-адреса источника адресу, полученному из базы DHCP Snooping (иными словами, айпишник закрепляется за портом свича)
- соответствие MAC-адреса источника адресу, полученному из базы DHCP Snooping

Включается IP Source Guard командой **ip verify source** на нужном интерфейсе. В таком виде проверяется только привязка IP-адреса, чтобы добавить проверку MAC, используем **ip verify source port-security**. Само собой, для работы IP Source Guard требуется включенный DHCP snooping, а для контроля MAC-адресов должен быть включен port security.

Dynamic ARP Inspection

Как мы уже знаем, для того, чтобы узнать MAC-адрес устройства по его IP-адресу, используется проткол ARP:

посылается широковещательный запрос вида “у кого ip-адрес 172.16.1.15, ответьте 172.16.1.1”, устройство с айпишником 172.16.1.15 отвечает. Подобная схема уязвима для атаки, называемой ARP-poisoning aka ARP-spoofing: вместо настоящего хоста с адресом 172.16.1.15 отвечает хост злоумышленника, заставляя таким образом трафик, предназначенный для 172.16.1.15 следовать через него. Для предотвращения такого типа атак используется фича под названием Dynamic ARP Inspection. Схема работы похожа на схему DHCP-Snooping’a: порты делятся на доверенные и недоверенные, на недоверенных каждый ARP-ответ подвергается анализу: сверяется информация, содержащаяся в этом пакете, с той, которой свич доверяет (либо статически заданные соответствия MAC-IP, либо информация из базы DHCP Snooping). Если не сходится- пакет отбрасывается и генерируется сообщение в syslog. Включаем в нужном плане (планах): **ip arp inspection vlan номер(а)**. По умолчанию все порты недоверенные, для доверенных портов используем **ip arp inspection trust**.

Практика

Наверное, большинство ошибок в Packet Tracer допущено в части кода, отвечающего за симуляцию STP, будьте готовы. В случае сомнения сохранитесь, закройте PT и откройте заново

Итак, переходим к практике. Для начала внесем некоторые изменения в топологию — добавим избыточные линки. Учитывая сказанное в самом начале, вполне логично было бы сделать это в московском офисе в районе серверов — там у нас свич msk-arbat-asw2 доступен только через asw1, что не есть гуд. Мы отбираем (пока, позже возместим эту потерю) гигабитный линк, который идет от msk-arbat-dsw1 к msk-arbat-asw3, и подключаем через него asw2. Asw3 пока подключаем в порт Fa0/2 dsw1. Перенастраиваем транки:

```
msk-arbat-dsw1(config)#interface gi1/2
msk-arbat-dsw1(config-if)#description msk-arbat-asw2
msk-arbat-dsw1(config-if)#switchport trunk allowed vlan 2,3
msk-arbat-dsw1(config-if)#int fa0/2
msk-arbat-dsw1(config-if)#description msk-arbat-asw3
msk-arbat-dsw1(config-if)#switchport mode trunk
msk-arbat-dsw1(config-if)#switchport trunk allowed vlan 2,101-104

msk-arbat-asw2(config)#int gi1/2
msk-arbat-asw2(config-if)#description msk-arbat-dsw1
msk-arbat-asw2(config-if)#switchport mode trunk
msk-arbat-asw2(config-if)#switchport trunk allowed vlan 2,3
msk-arbat-asw2(config-if)#no shutdown
```

Не забываем вносить все изменения в документацию!

Имя устройства	Порт	Название	VLAN	
			Access	Trunk
msk-arbat-gw1	FE0/1	UpLink		
	FE0/0	msk-arbat-dsw1		2,3,101,102,103,104
msk-arbat-dsw1	FE0/24	msk-arbat-gw1		
	GE1/1	msk-arbat-asw1		2,3
	GE1/2	msk-arbat-asw2		2,3
	FE0/1	msk-rubl-asw1		2,101,104
	FE0/19-FE0/23	port-channel 1 (asw3)		2,101,102,103,104
msk-arbat-asw1	GE1/1	msk-arbat-dsw1		2,3
	GE1/2	msk-arbat-asw2		2,3
	FE0/1	Web-server	3	
	FE0/2	File-server	3	
msk-arbat-asw2	GE1/1	msk-arbat-asw1		2,3
	Ge1/2	msk-arbat-dsw1		
	FE0/1	Mail-Server	3	
msk-arbat-asw3	GE1/1	msk-arbat-dsw1		
	FE0/1-FE0/5	PTO	101	
	FE0/6-FE0/10	FEO	102	
	FE0/11-FE0/15	Accounting	103	
	FE0/16-FE0/19	Other		
	FE0/20-FE0/24	port-channel 1 (dsw1)		
msk-rubl-asw1	FE0/24	msk-arbat-dsw1		2,104
	FE0/1-FE0/15	PTO	1	
	FE0/20	administrator	104	

[Скачать актуальную версию документа.](#)

Теперь посмотрим, как в данный момент у нас *самонастроился* STP. Нас интересует только VLAN0003, где у нас, судя по схеме, петля.

```
msk-arbat-dsw1>en
msk-arbat-dsw1#show spanning-tree vlan 3
```


VLAN0003					# Номер влана, в котором работает данный процесс STP	
Spanning tree enabled protocol ieee					# Тип STP	
Root ID	Priority	32771			# Информация о корневом свиче : приоритет	
	Address	0007.ECC4.09E2			# Мас-адрес	
	Cost	4			# Сколько стоит добраться до корневого свича	
	Port	25(GigabitEthernet1/1)			# Через какой порт лучше добраться (Root Port)	
	Hello Time	2 sec	Max Age	20 sec	Forward Delay 15 sec	# Таймеры STP
Bridge ID	Priority	32771	(priority 32768 sys-id-ext 3)		# Информация о текущем свиче: приоритет (стандартный 32768+номер влана)	
	Address	000B.BE2E.392C				
	Hello Time	2 sec	Max Age	20 sec	Forward Delay 15 sec	
	Aging Time	20				
Interface	Role	Sts	Cost	Prio.Nbr	Type	# Состояние портов
<hr/>						
Fa0/2	Desg	FWD	19	128.2	P2p	
Fa0/24	Desg	FWD	19	128.24	P2p	
Gi1/1	Root	FWD	4	128.25	P2p	
Gi1/2	Altn	BLK	4	128.26	P2p	

Итак, какую информацию мы можем получить? Так как по умолчанию на современных свичах работает PVST+ (т.е. для каждого влана свой процесс STP), и у нас есть более одного влана, выводится информация по каждому влану в отдельности, каждая запись предваряется номером влана. Затем идет вид STP: ieee значит PVST, rstp — Rapid PVST, mstp то и значит. Затем идет секция с информацией о корневом свиче: установленный на нем приоритет, его мас-адрес, стоимость пути от текущего свича до корневого, порт, который был выбран в качестве корневого (имеет лучшую стоимость), а также настройки таймеров STP. Далее- секция с той же информацией о текущем свиче (с которого выполняли команду). Затем- таблица состояния портов, которая состоит из следующих колонок (слева направо):

- собственно, порт
- его роль (Root- корневой порт, Desg- назначенный порт, Altn- дополнительный, Back- резервный)
- его статус (FWD- работает, BLK- заблокирован, LIS- прослушивание, LRN- обучение)
- стоимость маршрута до корневого свича
- Port ID в формате: приоритет порта.номер в порта
- тип соединения

Итак, мы видим, что Gi1/1 корневой порт, это дает некоторую вероятность того, что на другом конце линка корневой свич. Смотрим по схеме, куда ведет линк: ага, некий msk-arbat-asw1.

```
msk-arbat-asw1#show spanning-tree vlan 3
```

И что же мы видим?

```
VLAN0003
Spanning tree enabled protocol ieee
Root ID  Priority  32771
      Address    0007.ECC4.09E2
      This bridge is the root
      Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
```

Вот он, наш корневой свич для VLAN0003.

А теперь посмотрим на схему. Ранее, мы увидели в состоянии портов, что dsw1 блокирует порт Gi1/2, разрывая таким образом петлю. Но является ли это оптимальным решением? Нет, конечно. Сейчас наша новая сеть работает точь-в-точь как старая- трафик от asw2 идет только через asw1. Выбор корневого маршрутизатора никогда не нужно оставлять на совесть глупого STP. Исходя из схемы, наиболее оптимальным будет выбор в качестве корневого свича dsw1- таким образом, STP заблокирует линк между asw1 и asw2. Теперь это все надо объяснить недалекому протоколу. А для него главное что? Bridge ID. И он неслучайно складывается из двух чисел. Приоритет- это как раз то слагаемое, которое отдано на откуп сетевому инженеру, чтобы он мог повлиять на результат выбора корневого свича. Итак, наша задача сводится к тому, чтобы уменьшить (меньше-лучше, думает STP) приоритет нужного свича, чтобы он стал Root Bridge. Есть два пути:

1) вручную установить приоритет, заведомо меньший, чем текущий:

```
msk-arbat-dsw1>enable
msk-arbat-dsw1#configure terminal
msk-arbat-dsw1(config)#spanning-tree vlan 3 priority?
<0-61440> bridge priority in increments of 4096
msk-arbat-dsw1(config)#spanning-tree vlan 3 priority 4096
```

Теперь он стал корневым для влана 3, так как имеет меньший Bridge ID:

```
msk-arbat-dsw1#show spanning-tree vlan 3
VLAN0003
Spanning tree enabled protocol ieee
Root ID  Priority  4099
      Address    000B.BE2E.392C
      This bridge is the root
      Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
```

2) дать умной железнке решить все за тебя:

```
msk-arbat-dsw1(config)#spanning-tree vlan 3 root primary
```

Проверяем:

```
msk-arbat-dsw1#show spanning-tree vlan 3
VLAN0003
Spanning tree enabled protocol ieee
Root ID Priority 24579
Address 000B.BE2E.392C
This bridge is the root
Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
```

Мы видим, что железка поставила какой-то странный приоритет. Откуда взялась эта круглая цифра, спросите вы? А все просто- STP смотрит минимальный приоритет (т.е. тот, который у корневого свича), и уменьшает его на два шага инкремента (который составляет 4096, т.е. в итоге 8192). Почему на два? А чтобы была возможность на другом свиче дать команду spanning-tree vlan n root secondary (назначает приоритет=приоритет корневого-4096), что позволит нам быть уверенными, что, если с текущим корневым свичом что-то произойдет, его функции перейдут к этому, “запасному”. Вероятно, вы уже видите на схеме, как лампочка на линке между asw2 и asw1 пожелтела? Это STP разорвал петлю. Причем именно в том месте, в котором мы хотели. Sweet! Зайдем проверим: лампочка — это лампочка, а конфиг — это факт.

```
msk-arbat-asw2#show spanning-tree vlan 3
VLAN0003
Spanning tree enabled protocol ieee
Root ID  Priority  24579
    Address  000B.BE2E.392C
    Cost      4
    Port     26(GigabitEthernet1/2)
    Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec

Bridge ID  Priority  32771 (priority 32768 sys-id-ext 3)
    Address  000A.F385.D799
    Hello Time 2 sec Max Age 20 sec Forward Delay 15 sec
    Aging Time 20

Interface  Role  Sts  Cost   Prio.Nbr Type
-----
Fa0/1      Desg FWD  19     128.1  P2p
Gi1/1      Altn BLK  4      128.25 P2p
Gi1/2      Root FWD  4      128.26 P2p
```

Теперь полюбуемся, как работает STP: заходим в командную строку на ноутбуке PTO1 и начинаем бесконечно пинговать наш почтовый сервер (172.16.0.4). Пинг сейчас идет по маршруту ноутбук-asw3-dsw1-gw1-dsw1(ну тут понятно, зачем он крюк делает — они из разных вланов)-asw2-сервер. А теперь поработаем Годзиллой из SimCity: нарушим связь между dsw1 и asw2, вырвав провод из порта (замечаем время, нужное для пересчета дерева).

Пинги пропадают, STP берется за дело, и за *каких-то* 30 секунд коннект восстанавливается. Годзиллу прогнали, пожары потушили, связь починили, втыкаем провод обратно. Пинги опять пропадают на 30 секунд! Мда-а-а, как-то не очень быстро, особенно если представить, что это происходит, например, в процессинговом центре какого-нибудь банка.

Но у нас есть ответ медленному PVST+! И ответ этот — Быстрый PVST+ (так и называется, это не шутка: Rapid-PVST). Посмотрим, что он нам дает. Меняем тип STP на всех свичах в москве командой конфигурационного режима: spanning-tree mode rapid-pvst

Снова запускаем пинг, вызываем Годзиллу... Эй, где пропавшие пинги? Их нет, это же Rapid-PVST. Как вы, наверное, помните из теоретической части, эта реализация STP, так сказать, “подстиляет соломку” на случай падения основного линка, и переключается на дополнительный (alternate) порт очень быстро, что мы и наблюдали. Ладно, втыкаем провод обратно. Один потерянный пинг. Неплохо по сравнению с 6-8, да?

EtherChannel

Помните, мы отобрали у офисных работников их гигабитный линк и отдали его в пользу серверов? Сейчас они, бедняжки, сидят, на каких-то ста мегабитах, прошлый век! Попробуем расширить канал, и на помощь призовем EtherChannel. В данный момент у нас соединение идет от fa0/2 dsw1 на Gi1/1 asw3, отключаем провод. Смотрим, какие порты можем использовать на asw3: ага, fa0/20-24 свободны, кажется. Вот их и возьмем. Со стороны dsw1 пусть будут fa0/19-23. Соединяем порты для EtherChannel между собой. На asw3 у нас на интерфейсах что-то настроено, обычно в таких случаях используется команда конфигурационного режима default interface range fa0/20-24, сбрасывающая настройки порта (или портов, как в нашем случае) в дефолтные. Packet tracer, увы, не знает такой хорошей команды, поэтому в ручном режиме убираем каждую настройку, и тушим порты (лучше это сделать, во избежание проблем)

```
msk-arbat-asw3(config)#interface range fa0/20-24
msk-arbat-asw3(config-if-range)#no description
msk-arbat-asw3(config-if-range)#no switchport access vlan
msk-arbat-asw3(config-if-range)#no switchport mode
msk-arbat-asw3(config-if-range)#shutdown
```

```
ну а теперь волшебная команда

msk-arbat-asw3(config-if-range)#channel-group 1 mode on
```

```
то же самое на dsw1:

msk-arbat-dsw1(config)#interface range fa0/19-23
msk-arbat-dsw1(config-if-range)#channel-group 1 mode on
```

поднимаем интерфейсы asw3, и вуаля: вот он, наш EtherChannel, раскинулся аж на 5 физических линков. В конфиге он будет отражен как interface Port-channel 1. Настраиваем транк (повторить для dsw1):

```
msk-arbat-asw3(config)#int port-channel 1
msk-arbat-asw3(config-if)#switchport mode trunk
msk-arbat-asw3(config-if)#switchport trunk allowed vlan 2,101-104
```

Как и с STP, есть некая трудность при работе с etherchannel в Packet Tracer’е. Настроить-то мы, в принципе, можем по вышеописанному сценарию, но вот проверка работоспособности под большим вопросом: после отключения одного из портов в группе, трафик перетекает на следующий, но как только вы вырубаετε второй порт — связь теряется и не восстанавливается даже после включения портов.

Отчасти в силу только что озвученной причины, отчасти из-за ограниченности ресурсов мы не сможем раскрыть в полной мере эти вопросы и посему оставляем БОльшую часть на самоизучение.

Материалы выпуска

- Новый план коммутации
- Файл РТ с лабораторной.
- Конфигурация устройств
- STP или STP
- Безопасность канального уровня
- Агрегация каналов

По сложившейся традиции все неотвеченные вопросы безымянными читателями хабра задаются в блоге цикла в ЖЖ.

За видео и помощь в подготовке статьи спасибо @eucariot

сети для самых маленьких, STP, RSTP, Etherchannel, 802.1d, 802.1w, port security

↑ +66 ↓

👁 218k

★ 1161

@theGluck

карма 57,0

рейтинг 0,0

- Похожие публикации
- +89

Сети для самых маленьких. Часть пятая. ACL и NAT

👁 193k

★ 1428

💬 47
- +100

Сети для самых маленьких. Часть третья. Статическая маршрутизация

👁 211k

★ 1518

💬 49
- +84

Сети для самых маленьких. Часть вторая. Коммутация

👁 288k

★ 1700

💬 45

Реклама помогает поддерживать и развивать наши сервисы

Подробнее

Спецпроект

Самое читаемое

Администрирование

Сейчас

Сутки

Неделя

Месяц

+10

7 мер защиты сервера

👁 7,3k

★ 139

💬 6

+6

Сервер с лицензией Windows за 180 руб./мес. — Самый дешёвый VDS

👁

7,1k

★

40

💬

24

+3

«Прозрачный» Squid с разграничением доступа

👁

2,5k

★

69

💬

4

+40

Настройка UEFI-загрузчика. Самое краткое руководство в мире

👁

22,1k

★

369

💬

80

+6

Автосбор данных о выполненных заданиях в MS SQL Server

👁


1,8k

★


34

💬


0

 **bazil11** 14 мая 2012 в 10:46 #


Спасибо!

 **theGluck** 14 мая 2012 в 10:57 # h ↑ +1 ↑ ↓


Завсегда пожалуйста. Опыт по публикации, практически, первый, жду пожеланий и отзывов.

 **slam007** 14 мая 2012 в 11:11 # +1 ↑ ↓

Так и знал, что «падение» сети — это проделки Бендера =)

 **eucariot** 14 мая 2012 в 11:39 # h ↑ 0 ↑ ↓

Ну что вы, он помогать призван.

 **JDima** 14 мая 2012 в 11:58 # +3 ↑ ↓

Не заметил один момент, совершенно критический, и почему-то игнорируемый во всех курсах циски.

Portfast на нагруженных свитчах смертельно опасен!


- Суть в следующем.
- 1) На любых коммутаторах BPDU обрабатываются программно.
 - 2) Частота рассылки BPDU — по дефолту 2 секунды.
 - 3) При втыкании патч-корда порт переходит в forwarding мгновенно.

По меркам коммутатора, через который проходит много (десятков) гигабит, 2 секунды — вечность. За это время мозги свитчу выносятся напрочь, и он уже не может загасить порт (errdisable в случае BPDU Guard или просто stp blocking — неважно), ресурсов для обработки BPDU нет, ибо большинство бродкастов в сети — ARP пакеты, долетающие до CPU свитча. И вот у нас шторм по всему L2 сегменту, который не хочет прекращаться сам по себе. Это не голая теория, я сам однажды на это нарвался (как дебил сказал человеку «воткни новый свитч в любые два порта»).


Понятно, что на аксессной пользовательской железе с парой сотен мегабит и 100мб интерфейсами до пользователей такая катастрофа маловероятна, я больше про ЦОДы говорю.

Потому ОБЯЗАТЕЛЬНО задействовать Storm Control везде, где только можно. Он реализован аппаратно, и он позволяет коммутатору успешно работать и при шторме. Ведь один из самых мерзких моментов шторма — то, что и добраться до самого свитча при отсутствии out-of-band management крайне непросто.

Кстати — как минимум cat6500 прекрасно отзывается по консоли даже когда шторм выжрал все ресурсы.


 **theGluck** 14 мая 2012 в 12:02 # h ↑ 0 ↑ ↓

очень ценно, спасибо


 **JDima** 14 мая 2012 в 12:23 # h ↑ 0 ↑ ↓

Про это можно почитать например тут: blog.ioshints.info/2012/04/stp-loops-strike-again.html

И еще момент. Хорошая сеть — та сеть, где вообще нет STP колец, и ни один линк не блокируется, с сохранением полной отказоустойчивости (STP задействован лишь в качестве предохранительного механизма). Обычно используют агрегацию линков плюс технологии, позволяющие завязать один port-channel на двух или более разных шасси (стеки на 3750, VSS на cat6500, vPC на нексусах и т.д. — у каждого вендора есть нечто подобное). Хардкор — полноценный роутинг на L2, TRILL и протоколы на его основе (у циски — Fabricpath). Но его пока мало где встретишь, и то кроме ЦОДов он нигде не используется.

 **theGluck** 14 мая 2012 в 12:33 # h ↑ 0 ↑ ↓

на хабре была интересная [статья](#) на этот счет. В комментариях нешуточные дебаты. В идеале- да, согласен. Но суровая реальность сурова.

 **JDima** 14 мая 2012 в 12:40 # h ↑ 0 ↑ ↓

В комментариях нешуточные дебаты.
Да там даже обсуждать нечего. Любой, кто говорит, что STP хорош хоть чем-то кроме дешевого решения для организации базовой избыточности либо в качестве предохранителя, является ослом.

Хотя смотрю, кому-то хватило мозгов сделать по spanning-tree vlan X, раз как бы теоретически колец нет. Ну-ну... Никто ведь не говорит, что STP надо отключать. Просто он не должен ничего никогда

Что обсуждают

Сейчас	Вчера	Неделя	Месяц
В дцатый раз про собеседования	89		
Из Firefox 52 удалят API уровня заряда аккумулятора для сохранения приватности пользовательских данных	30		
Как запустить ClickHouse своими силами и выиграть джекпот	21		
Практическое пособие «Как вывести из себя программиста»	43		
Управляем автоматом на Groovy/Java. Как ЧПУ станку в домашней мастерской не превратиться в мульт героев “двое из ларца”	14		

блокировать кроме как на полминуты максимум при появлении физики.



paralon 15 мая 2012 в 10:32 # h i

0 ↑ ↓

А не подскажите, как CCNP, что на необходимость не нужно STP говорит 642-813 SWITCH? А то я пока на 642-902 ROUTE застрял.

Или вышеописанная вами теория и есть, то что рекомендует cisco?

Спасибо за ответ



Legh 15 мая 2012 в 12:04 # h i

+1 ↑ ↓

в 642-813 половина — это STP и его вариации. содержимое топика — необходимый, но недостаточный минимум для сдачи.

Удачи с роутингом, больше практикуйтесь!



JDima 15 мая 2012 в 13:34 # h i

+1 ↑ ↓

Понятия не имею про SWITCH, я когда-то BCMSN сдавал. Но тогда данный экзамен был не архитектурным, а техническим. Думаю, с тех пор мало что изменилось. И крайне сомневаюсь, что в SWITCH будут вопросы про VSS/vPC, ибо у человека, не администрирующего 6500/N5k, мало шансов потрогать эти технологии.

Для решения архитектурных вопросов следует выйти за рамки сертификаций и читать SRND, где куда более актуальная информация. Вот например про «отказ от STP»:

www.cisco.com/en/US/docs/solutions/Enterprise/Data_Center/DC_3_0/DC-3_0_IPInfra.html#wp1044510

Да и вообще советую прочитать всю эту статью, там много полезной информации.



nicolnx 14 мая 2012 в 15:36 # h i

0 ↑ ↓

На том BCMSN (SWITCH) что был я этот момент ни разу не игнорировался. Но даже без него понятно, что

%Warning: portfast should only be enabled on ports connected to a single host. Connecting hubs, concentrators, switches, bridges, etc... to this interface when portfast is enabled, can cause temporary bridging loops.
Use with CAUTION

при включении одного — не просто так написан.

Отсюда вывод — вы включили portfast там где включать его было нельзя, получив тот самый temporary bridging loop, и дело тут вовсе не в нагрузке...



JDima 14 мая 2012 в 15:48 # h i

0 ↑ ↓

«can cause temporary bridging loops»

Именно что скорее «constant». Циска на моей памяти не говорила, что подобный луп может вынести весь L2 сегмент до того момента, когда администратор руками разъединит кольцо, а не на 2 секунды до первой BPDU.

Мне на самом деле тогда сильно «повезло». Местный умудрился попасть в 2 из 4-х портов, в которые попасть было нельзя.

«вы включили portfast там где включать его было нельзя»

Немного неправильный подход. Сеть должна быть отказо- и дуракоустойчивой. То есть вытыкание или втыкание любого проводка не должно приводить к катастрофе. А то получается на «попал в аварию и не сработала подушка» ответ «не надо было попадать в аварию».

У нас на пользовательском аксесе раз в пару месяцев BPDU guard блокирует порты, потому что кто-то умный втыкает торчащий из розетки патч-корд в соседнюю розетку. Суть моего комментария сводится к тому, что не следует полагаться на механизмы STP в этом случае. Ну ладно, на пользовательском порту можно сказать «sw port-security max 3», это должно помочь, а вот в виртуализированном ЦОДе с этим куда сложнее. Потому всегда нужны дополнительные меры защиты. Но: они не всегда поддерживаются. Взять Nexus 2000 — у них нет никакого storm control. То есть кто-то соединил соседние порты патч-кордом, и минимум кусок сети с большой вероятностью лег.



nicolnx 14 мая 2012 в 15:58 # h i

0 ↑ ↓

> Циска на моей памяти не говорила, что подобный луп может вынести весь L2 сегмент

Циска вроде бы никогда не умалчивала того факта что bridging loop — это то чего стоит избегать любой ценой.

>> «вы включили portfast там где включать его было нельзя»

> Немного неправильный подход. Сеть должна быть отказо- и дуракоустойчивой.

Вы сами отключили защиту от дурака там где этого было делать нельзя. portfast по умолчанию не задействован на всех портах. При его включении в консоли отдает ворнинг с капсом CAUTION. Таким образом вы сняли ремни безопасности, убрали подушку, — и тапку в пол. А потом заявляете что, мол, ездить на машине смертельно опасно...

> У нас на пользовательском аксесе раз в пару месяцев BPDU guard блокирует порты, потому что кто-то умный втыкает торчащий из розетки патч-корд в соседнюю розетку. Суть моего комментария сводится к тому, что не следует полагаться на механизмы STP в этом случае.

Получается, portfast у вас на access портах отключен, раз там bpdu летят?

Взять Nexus 2000 — у них нет никакого storm control. То есть кто-то соединил соседние порты патч-кордом, и минимум кусок сети с большой вероятностью лег.

Если есть вероятность образование петли, то отключать portfast и глушить там stp — преступление против человечества. И рано поздно это даст по голове. Еще раз — should only be enabled on ports connected to a single host. А дуру можно много чего сломать...



JDima 14 мая 2012 в 16:15 # h i

0 ↑ ↓

Вы сами отключили защиту от дурака

Ну вот представим себе, что portfast на порту абсолютно необходим. Некая система не может ждать по полминуты после поднятия линка. Выходит, кто-то соединил два порта — и приплыли...

Получается, portfast у вас на access портах отключен, раз там bpdu летят?

Вы путаете portfast с bpduguard.

```
#show spanning-tree interface fa4/10 detail
Port 202 (FastEthernet4/10) of VLAN0200 is designated forwarding
Port path cost 19, Port priority 128, Port Identifier 128.202.
Designated root has priority 32968, address 001a.a18b.bec0
Designated bridge has priority 32968, address 001a.a18b.bec0
Designated port id is 128.202, designated path cost 0
Timers: message age 0, forward delay 0, hold 0
Number of transitions to forwarding state: 0
The port is in the portfast mode
Link type is point-to-point by default
Bpdu guard is enabled by default
BPDU: sent 1099568, received 0
```

Вот здоровый аксесный порт. Если соединить его с fa4/11, то с вероятностью 99,999% fa4/10 уйдет в blocking, сразу после получения BPDU, и пользователи ничего не заметят.

Еще раз — should only be enabled on ports connected to a single host. А сдурю можно много чего сломать...

Если человек полагается на «авось», т.е. надеется на то, что никто не соединит соседние порты патч-кордом, то ему вообще нечего делать в IT. Необходимо заранее предусмотреть любые возможные сценарии, ведущие к аварии, и предпринять меры по недопущению как можно большего числа этих аварий. И всегда исходить из того, что пользователь — идиот; вспышки на солнце происходят каждый день; глюк в софте на сетевом железе случается раз в два дня и т.д.

Вы же явно сторонник подхода «если система падает от чиха, то надо запретить чихать, а не устранить источник влияния чиха на стабильность». Так не бывает.



nicolnx 14 мая 2012 в 16:38 # h i

0 ↑ ↓

> Необходимо заранее предусмотреть любые возможные сценарии, ведущие к аварии,
> и? предпринять меры по недопущению как можно большего числа этих аварий.

Надо понимать, в комплекс мер по предотвращению у вас входит игнорирование предупреждений вендора о возможности образования петель при тех телодвижениях что вы совершаете?



JDima 14 мая 2012 в 16:51 # h i

0 ↑ ↓

А есть альтернативы? Вы сможете, к примеру, обеспечить работу заливки по PXE без portfast? Нет? Значит, исходим из того, что portfast — данность, от которой следует отталкиваться. Но если пользователь соединит патч-кордом две розетки и сегмент ляжет минут на 10, то виноват будет не пользователь, а сетевик. И даже заклинание «spanning-tree portfast bpduguard default» делу не поможет.

Еще раз. В документации по portfast'у где-то пишется, что после получения BPDU порт моментально переходит в blocking, а затем проходит через нормальный процесс STP. И кольцо должно возникнуть максимум на 2 секунды, что называется не «катастрофа», а «инцидент, который скорее всего останется незамеченным». Я же говорю, что на практике все может быть куда хуже. И мало кто об этом знает. В курсе BCMSN (который Вы явно пропустили, судя по «portfast не рассылает BPDU» :)) про такой исход дела ни слова не было.



nicolnx 14 мая 2012 в 17:15 # h i

0 ↑ ↓

> А есть альтернативы? Вы сможете, к примеру, обеспечить работу заливки по PXE без portfast? Нет?

В условиях взаимоисключающих параграфов (portast на потенциально петлеопасных портах+ доступ идиотов к этим портам) вопрос несколько выходит за рамки spanning tree. Более надежным решением тут видится создание топологии беспетельной от природы с использованием Vlan per port например или Private VLAN.

Еще как ненадежный вариант — portfast trunk: When you enable PortFast between two switches, the system will verify there are no loops in the network before bringing the blocking trunk to a forwarding state. —

www.cisco.com/en/US/docs/switches/lan/catalyst4000/7.4/configuration/guide/stp_enha.html#wp1019 — возможно, на некоторых платформах оно действительно will verify there are no loops in the network before

> В документации по portfast'у где-то пишется, что после получения BPDU порт моментально переходит в blocking, а затем проходит через нормальный процесс STP. И кольцо должно возникнуть максимум на 2 секунды

Все правильно. При условии что продавленный арп-штормом control-plane с другой стороны будет в состоянии отправить этот BPDU например. И что этот BPDU не дропнется на перегруженном порту еще до того как он дойдет до процессора. Нигде не написано что максимальное время бродкаст-шторма при включенном portfast на портах и запетлявшей топологии = 2с. Зачем же это додумывать?

У каждой технологии есть границы применимости, соответственно их нужно либо не нарушать, либо не использовать технологию. А если уже странного хочется, то хотябы понимать где костыли стоят.



JDima 14 мая 2012 в 17:45 # h i

0 ↑ ↓

Более надежным решением тут видится создание топологии беспетельной от природы с

использованием *Vlan per port* например или *Private VLAN*.
На 4500-м катализе с 300 портами, с текущей конфигурацией по 48 портов на VLAN, по VLANу на порт? Издевается? Большого бреда в жизни не слышал. Мы вроде как о корпоративной сети говорим, а не о провайдере. И думаю, понятно, почему и private vlan идет лесом по определению.

Еще как ненадежный вариант — portfast trunk

Еще одна глупость. Portfast trunk — тот же обычный portfast, предназначенный для подключения транком конечных устройств, не знающих про STP и неспособных создать петлю. Например, гипервизора. Или роутера с сабинтерфейсами. Но и у гипервизора может поехать крыша, и если он вдруг объединит в бридж оба порта — будет network meltdown по моему сценарию. Где-то про такое писали, у кого-то такое было.

При условии что продавленный арп-штормом control-plane с другой стороны будет в состоянии отправить этот BPDU например.

А вот про это в доках не говорится ни явно, ни намеками. И не каждый догадается, что на обработку BPDU может и не хватить ресурсов.

И что этот BPDU не дропнется на перегруженном порту еще до того как он дойдет до процессора.

Дропнется в направлении «in» на перегруженном порту? Любопытно... А на отправку у него и так безусловный приоритет. Иначе типичный congestion обернулся бы кошмарным расколбасом по всей сети, с постоянными перестроениями топологии.

Нигде не написано что максимальное время бroadcast-шторма при включенном portfast на портах и запетлявшей топологии = 2с. Зачем же это додумывать?

Написано, что порт при получении BPDU проходит через нормальный процесс STP.

«Максимум 2 секунды» подразумевается. Не будут же они как для идиотов все разжевывать.

У каждой технологии есть границы применимости, соответственно их нужно либо не нарушать, либо не использовать технологию.

Вы предложили две совершенно бредовые альтернативы.

И уж конечно перед рассмотрением других технологий следует хорошенько изучить первоначальную ;) Например, поставить везде storm-control broadcast level 1 — и сеть при шторме не упадет, будут лишь маленькие глючки с полагающимися на бродкаст сервисами (ARP, DHCP), да и то вряд ли — ресурсы свитча не будут исчерпаны, и он спокойно положит закольцованные порты.

«Не нарушать»? Это куда не годится. Надо сделать так, чтобы и при кратковременном нарушении ничего плохого не происходило. А для этого для начала надо хорошо знать матчасть.

Вы ведь даже не знали, что portfast'овый порт рассылает BPDU :)

И не вздумайте на собеседовании говорить «а давайте каждому по VLANу создадим»...



rapidspacerocket 5 марта 2014 в 15:58 # h ↑



Portfast выполняет ещё одну важную функцию, кроме быстрого up'а порта — он уменьшает количество сообщений topology change, которые приводят к стиранию таблиц MAC-адресов во всём коммутируемом сегменте, в результате чего следует флуд unknow unicast. Т.е. при каждом включении/выключении порта конечным пользователем, вы получаете TCN и последующую очистку mac-tables на всех свичах в домене STP/RSTP. Защита от дурака — это BPDU Guard для STP. В RSTP защита встроенная, при получении BPDU на Edge Port он автоматически перестают рассматриваться, как Edge Port и попадает под обычные действия алгоритма RSTP.



JDima 5 марта 2014 в 16:38 # h ↑



Фу, некропостер!

| Защита от дурака — это BPDU Guard для STP

На самом деле в указанном мной выше контексте особой разницы между BPDU guard и нормальным механизмом STP нет. В обоих случаях при получении BPDU порт будет заблокирован, так как перестанет быть portfast. В одном из них он имеет шанс скоро разблокироваться, если договорится с соседом, в другом останется заблокированным навеки или до истечения recovery таймера.



casperrr 21 декабря 2012 в 08:45 # h ↑



Ну, не так уж игнорируется, по крайней мере в доках (даже на русском)

img.nag.ru/projects/setup/b85/e0b321501dd4f33283adebd69b4fa1db.pdf

В данном примере устройство А является мостом, порт p1 которого уже выполняет пересылку. Для порта p2 включена функция PortFast. Устройство В является концентратором. После подключения второго кабеля к А порт p2 переходит в режим пересылки и образует петлю между p1 и p2. Данная петля разрывается как только p1 или p2 получает BPDU, которое переводит один из этих двух портов в режим блокировки. **Однако с таким типом временных петель существует проблема. Если трафик петли слишком интенсивный, у моста может возникнуть проблема успешной передачи BPDU, которое приведет к разрыву петли.** Это может привести к значительной задержке сходимости или в крайних случаях остановить работу сети.



wonderworld 14 мая 2012 в 14:04 #



Отличный цикл статей. Большое спасибо!



eucariot 14 мая 2012 в 14:06 # h ↑




Пожалуйста. Через месяц по ACL. Оставляйтесь на связи.



slam007 14 мая 2012 в 15:32 #




Анимация хорошая, можно начальству показывать.

 eucariot 14 мая 2012 в 16:25 <#> [h](#) [↑](#) [↓](#) 0

А для чего начальству это видеть?

 slam007 14 мая 2012 в 17:14 <#> [h](#) [↑](#) [↓](#) 0


Наверное вы не сталкивались с любопытным начальством, которое далеко от IT и которое может воткнуть висящий из свитча патч-корд в тот же свитч.

 eucariot 14 мая 2012 в 17:21 <#> [h](#) [↑](#) [↓](#) 0

Случай миловал, все начальники (или почти все) были умнее меня.

 mcdebugger 14 мая 2012 в 16:50 <#> [↑](#) [↓](#) 0

В тексте приведены порты fa0/1, fa0/24, на изображениях fe0/1, fe0/24. Я что-то подзабыл с прошлых статей, или опечатка?

 mcdebugger 14 мая 2012 в 16:51 <#> [h](#) [↑](#) [↓](#) 0


(в абзаце про Широковещательный шторм)

 eucariot 14 мая 2012 в 16:52 <#> [h](#) [↑](#) [↓](#) 0

Вообще, конечно, пересортица пошла. FE — FastEthernet, Fa — сокращение от него же, которое обычно в командной строке пишешь. В общем, это одно и то же.

 gissarsky 15 мая 2012 в 08:56 <#> [↑](#) [↓](#) +1

Отличная статья, нравится оформление и доступность информации.

 theGluck 15 мая 2012 в 09:07 <#> [h](#) [↑](#) [↓](#) 0

спасибо, мы старались

 paralon 15 мая 2012 в 10:45 <#> [↑](#) [↓](#) 0

Дичайше буду рекомендовать весь цикл ваших статей, тем кто готовится к CCNA.

Жаль, что я сдавал экзамен раньше чем вы начали это делать — думаю срок самостоятельного обучения сильно бы сократился.

Успехов!

 eucariot 15 мая 2012 в 10:51 <#> [h](#) [↑](#) [↓](#) 0

Спасибо. Я очень надеюсь, что мы ещё затронем интересные вам темы: динамическая маршрутизация, BGP, MPLS, ну и подкаст, конечно.

 theGluck 15 мая 2012 в 10:54 <#> [h](#) [↑](#) [↓](#) +1

Спасибо. Но рекомендовать наш цикл я бы стал только в качестве дополнительного материала к официальному курсу. Ну и ничто не заменит Джереми, конечно, если уж говорить о дополнениях :)

 Levor 15 мая 2012 в 10:57 <#> [↑](#) [↓](#) 0

С интересом читаю ваш цикл статей, просто и понятно всё объясняете. Обязательно буду рекомендовать всем, кому может пригодиться.

Вот если б вы начали этак на полгода раньше, просто цены б не было...

P.S. А когда планируете про MPLS рассказывать?

 eucariot 15 мая 2012 в 11:01 <#> [h](#) [↑](#) [↓](#) 0

MPLS это пока всё-таки мысль. И весьма отдалённая, поэтому даже не планировали ещё. Возможно, к концу лета или осенью.

 eucariot 15 мая 2012 в 11:02 <#> [h](#) [↑](#) [↓](#) 0

Я таких комментариев, кстати, получаю очень много. Такое ощущение, что опоздал везде и всюду.)

 Levor 15 мая 2012 в 11:06 <#> [h](#) [↑](#) [↓](#) 0

Судя по количеству добавивших в избранное, очень многим статьи будут полезны! А тем, кто недавно сдал CCNA (вроде меня), никогда не будет лишним повторить материал.

 eucariot 15 мая 2012 в 11:08 <#> [h](#) [↑](#) [↓](#) 0

Лишь бы в избранное добавляли потому, что интересно, а не с мыслями: «Вот простыня, будет время, может, прочитаю»))

 ValdikSS 15 мая 2012 в 11:27 <#> [↑](#) [↓](#) 0

Eucariot, у тебя второй аккаунт, что ли?


 eucariot 15 мая 2012 в 11:28 <#> [h](#) [↑](#) [↓](#) 0

Это же мой соавтор. В каждом топике, вроде, об этом говорим)

 rdntw 22 января 2014 в 23:00 <#> [↑](#) [↓](#) 0

Для обмена информацией между собой свичи используют специальные **пакеты**

всё такие пакеты? не кадры?

 **theGluck** 23 января 2014 в 08:41 # h ↑ 0 ↑ ↓

кадры, да

Только зарегистрированные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

Интересные публикации

GT

Ведущие производители чипов присматриваются к фотолитографии в глубоком ультрафиолете 1

H

Археология программиста 0

GT

Mars One готовят к продаже для выхода на фондовую биржу 5

H

Дорогой Хабр, я хочу чтобы ты лучше слышал своих юзернеймов 25


GT

Путин попросил правительство подумать о введении НДС на покупки в зарубежных интернет-магазинах 62

Яндекс.Директ

Список научных конференций 2016

naukaip.ru



Реклама

Войти

Регистрация

Разделы

Публикации

Хабы

Компании

Пользователи

Q&A

Песочница

Инфо

О сайте

Правила

Помощь

Соглашение

Услуги

Реклама

Спецпроекты

Тарифы

Контент

Семинары

Разное

iOS приложение

Android приложение

Помощь стартапам

© TM

Служба поддержки

Мобильная версия

