Динамическая маршрутизация. Введение.

КОМПЬЮТЕРНАЯ АКАДЕМИЯ «ШАГ»

Cisco | Networking Academy®
Mind Wide Open™

Рассмотрим недостатки статической маршрутизации:

1. Большая административная нагрузка, ложащаяся на администратора, связанная с необходимостью внесения записей о маршрутах во все доступные сети

2. Высокая вероятность ошибки администратора при внесении большого количества записей в таблицы маршрутизации

3. Проблемы, возникающие при отказе линий связи, из-за того, что маршрутизаторы не могут самостоятельно находить новые маршруты, и необходимо вмешательство администратора

Если бы маршрутизаторы умели посредством обмена друг с другом информацией о доступных сетях самостоятельно или частично самостоятельно строить свои таблицы маршрутизации, находя оптимальные маршруты при текущей топологии сети, то это бы решило перечисленные выше проблемы:

1. Администраторам не пришлось бы выполнять работу по составлению таблиц маршрутизации
Была бы сведена к минимуму возможность ошибки администратора

2. При изменении топологии связей маршрутизаторы находили бы новые маршруты в рамках работающих в настоящий момент линий связи

Разумеется, при этом возникает ряд новых вопросов:

1. Разработка эффективных протоколов (протоколов маршрутизации), с помощью которых маршрутизаторы могли бы обмениваться информацией о доступных сетях.

2. Дополнительный сетевой трафик, порождаемый протоколами маршрутизации

3. Администрирование протоколов маршрутизации

Интернет не является хаотическим нагромождением сетей и маршрутизаторов с «беспорядочными» связями друг с другом. Интернет имеет вполне четкую структуру:

существует некоторая магистраль Интернета, к которой подключаются крупные сети (например, провайдеры), получающие большие блоки адресов, например /19, затем уже к сетям провайдеров подключаются сети конечных пользователей. При этом множество маршрутизаторов не нуждается в том, чтобы знать маршруты во все сети.

1. Маршрутизатор небольшой компании, подключенной к одному провайдеру должен использовать помимо маршрута к подключенной сети лишь маршрут по умолчанию, ведущий к маршрутизатору провайдера

2. Маршрутизатор сети крупной компании, подключенной к одному провайдеру должен знать маршруты ко всем сетям своей компании и маршрут по умолчанию, ведущий к маршрутизатору провайдера

3. Маршрутизаторы крупных сетей (провайдеров или очень крупных компаний), подключенных непосредственно к магистрали Интернет должны знать маршруты в рамках своей части сети и маршруты в прочие сети, ведущие на граничные маршрутизаторы, подключенные непосредственно к магистрали

Интернет настолько большая система, что бесструктурный обмен данными обо ВСЕХ маршрутах между ВСЕМИ маршрутизаторами абсолютно лишен смысла и породит чудовищный трафик.

Кроме того, распространение всем маршрутизаторам обновленных сведений о новых или отказавших сетях приведет к тому, что уточнение маршрутных таблиц никогда не прекратится, так же порождая огромный трафик.

Возникает вопрос о некоторой структуризации Интернет с точки зрения маршрутизации.

Интернет с точки зрения маршрутизации рассматривается как двухуровневая система. Крупные сети, составляющие Интернет (например, сети ISP провайдеров) называют автономными системами (Autonomous System, AS).

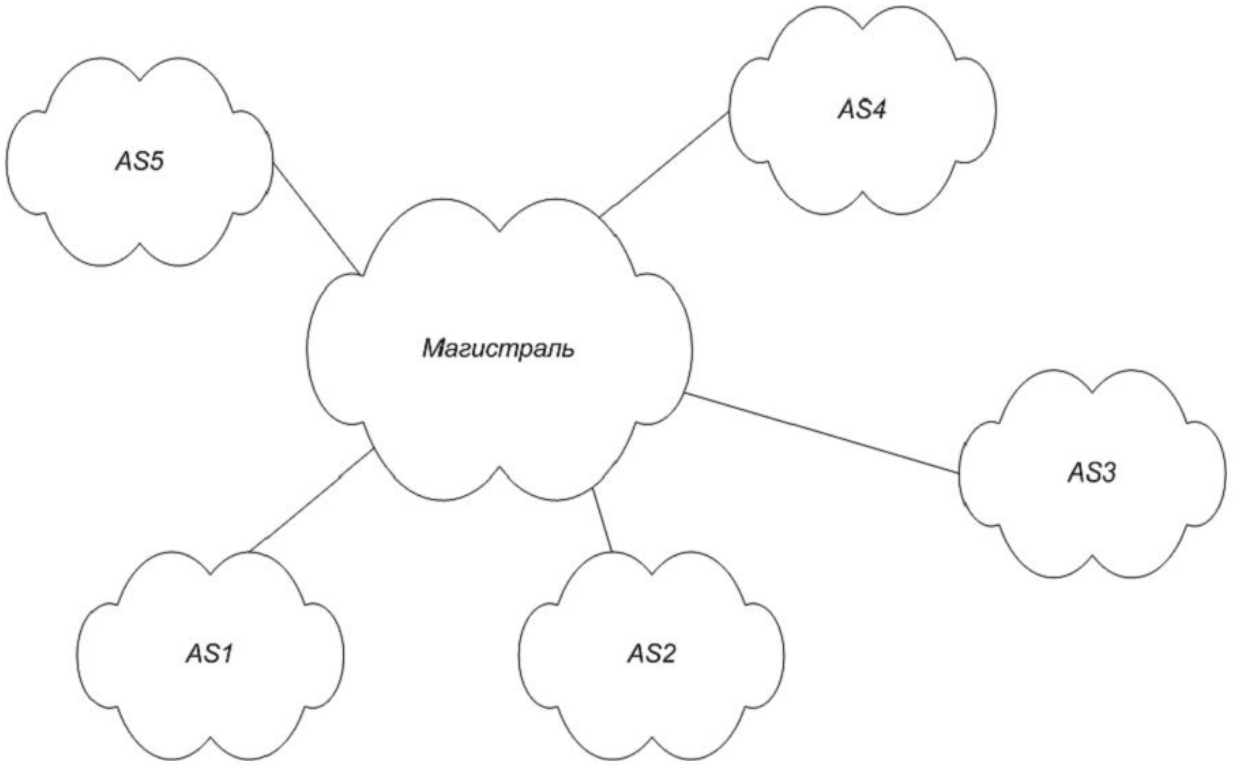
В рамках этих автономных систем маршрутизаторы решают задачи двух типов:

1. маршрутизация внутри автономной системы, т.е. между составляющими автономную систему узлами (внутрисистемная маршрутизация)

2. маршрутизация за пределы автономной системы, т.е. в другие автономные системы (межсистемная маршрутизация).

Каждая автономная система находится в одном административном управлении и организует внутрисистемную маршрутизацию таким образом, какой посчитает нужным. Магистраль Интернет, которая в некотором смысле, то же является автономной системой, так же находится в одном административном управлении.

Упрощенно структуру Интернет с точки зрения деления на автономные системы можно изобразить следующим образом:



В каждой автономной системе задача о внутрисистемной маршрутизации может решаться произвольным образом и принятие решения о внутрисистемной маршрутизации целиком лежит на администраторе AS. При этом, различные автономные системы могут применять внутри разные подходы к маршрутизации, т.е. применять статическую маршрутизацию, или какой-то протокол динамической маршрутизации.

Магистраль, как автономная система, тоже должна иметь какой либо, единый для всей магистрали, способ организации таблиц маршрутизации, причем для магистрали, очевидно, не подходит статическая маршрутизация, необходимо применять маршрутизацию динамическую.

Каждая автономная система внутри использует ту технику построения маршрутных таблиц, которая ей удобна (статическая маршрутизация или выбранный протокол динамической маршрутизации).

ВСЯ магистраль использует один протокол динамической маршрутизации и все граничные маршрутизаторы всех автономных систем должны поддерживать этот протокол динамической маршрутизации, принятый на магистрали.

Задачи, решаемые внутри автономной системы и на магистрали достаточно различны. Поэтому существующие протоколы маршрутизации принято делить на два класса:

1. протоколы внутренней маршрутизации

2. протоколы внешней маршрутизации.

Первые применяются внутри автономных систем и из вышесказанного следует, что возможно одновременное существование и использование многих внутренних протоколов маршрутизации, так как в различных автономных системах могут применяться различные протоколы внутренней маршрутизации.

С одной стороны магистраль – одна автономная система и в ней может одновременно применяться только один протокол в одно время. С другой стороны по мере развития Интернет сложность маршрутизации на магистрали росла (и растет) и протокол внешней маршрутизации, применяемый на магистрали Интернет со временем может измениться.

Внутренние протоколы:

1. RIP (Routing Information Protocol, протокол информирования о маршрутах) первый протокол динамической маршрутизации, впервые внедрен в BSD системах в начале 80-х годов, простой в реализации протокол.

2. IGPR (Interior Gateway Routing Protocol, протокол маршрутизации внутреннего шлюза) разработан компанией Cisco Systems в середине 80-х, более эффективен, нежели RIP, имеет с ним идеологически достаточно много общего. Является форменной разработкой и поэтому поддерживается только в продукции Cisco Systems.

3. EIGRP (Enhanced Interior Gateway Routing Protocol, протокол маршрутизации внутреннего шлюза) так же разработан компанией Cisco Systems (начало 90-х) и поддерживается только в продукции Cisco Systems, как следует из названия является расширением IGPR.

4. OSPF (Open Shortest Path First, открытая реализация алгоритма «кратчайший путь первым»), открытый протокол, предоставляющий возможность строить на его основе очень большие, даже внутренне структурированные автономные системы, сложен, но очень эффективен.

5. IS-IS (Intermediate System-Intermediate System) разработан ISO как протокол маршрутизации сетевого протокола стека OSI (CLNP), адаптирован для маршрутизации на базе протокола IP.

Внешние протоколы:

1. GGP (Gateway Gateway Protocol, протокол взаимодействия шлюзов), статус historic, нами рассмотрен не будет

2. EGP (Exterior Gateway Protocol, протокол внешних шлюзов), статус historic, нами рассмотрен не будет

3. BGP (Border Gateway Protocol, протокол граничных шлюзов) применяется сегодня для маршрутизации на магистрали Интернет, будет рассмотрен нами обзорно.

Существует два базовых подхода, которые используются для организации взаимодействия маршрутизаторов друг с другом с целью динамического построения таблиц маршрутизации:

1. Дистанционно векторный алгоритм (DVA, Distance Vector Algorithm) или алгоритм Беллмана-Форда.

2. Алгоритм состояния связей (LSA, Link State Algorithm) или алгоритм Дейкстра.

Предположим, что существует некоторая совокупность сетей и маршрутизаторов, которые связывают эти сети.

Пусть изначально в таблицы маршрутизаторов не были занесены никакие записи о доступных сетях. Значит ли это, что маршрутизаторы в этом случае вообще не были сконфигурированы и маршрутизаторы не имеют маршрутов ни в какие сети? «Нет» - на оба вопроса: интерфейсам маршрутизаторов необходимо присвоить IP адреса, следовательно, совсем не конфигурировать маршрутизаторы нельзя, а раз интерфейсы маршрутизаторов имеют IP адреса, то маршрутизаторы сразу при старте знают маршруты в подключенные сети.

Существует некоторое множество маршрутизаторов и сетей. Конечной целью работы протокола маршрутизации будет ситуация, когда каждый маршрутизатор знает маршруты во все сети системы, однако на начальном этапе каждый маршрутизатор знает только маршруты во все подключенные сети.

Возникает вопрос: как маршрутизаторы могут обмениваться маршрутной информацией, если ее изначально НЕТ, чем именно обмениваются маршрутизаторы, если они изначально ничего не знают. На самом деле совокупное знание маршрутизаторов на этапе старта работы алгоритма динамической маршрутизации как раз и охватывает знание маршрутов во все сети, но только каждый маршрутизатор обладает частью этого знания, т.е. имеет маршруты только в подключенные непосредственно к нему сети.

Тогда, маршрутизаторам просто необходимо обменяться друг с другом этой информацией. Т.е. сделать так, чтобы частичные знания каждого были доступны всем, и когда маршрутизаторы «сложат» свои знания о маршрутах в известные им сети, у каждого маршрутизатора будет информация о маршрутах во все сети, а не только в подключенные. И, таким образом, цель динамической маршрутизации будет достигнута.

Алгоритм Беллмана-Форда (Distance Vector Algorithm).

В соответствии с данным алгоритмом, каждый маршрутизатор через известные промежутки времени рассылает через все свои интерфейсы сведения обо всех сетях, маршруты к которым маршрутизатору известны (это и называется вектором расстояний), такая рассылка называется «объявление маршрута» или «анонс маршрута».

Каждая сеть описывается номером сети и метрикой, т.е. стоимостью доставки в эту сеть. Рассылка обычно делается широковещательно на канальном и сетевом уровне.

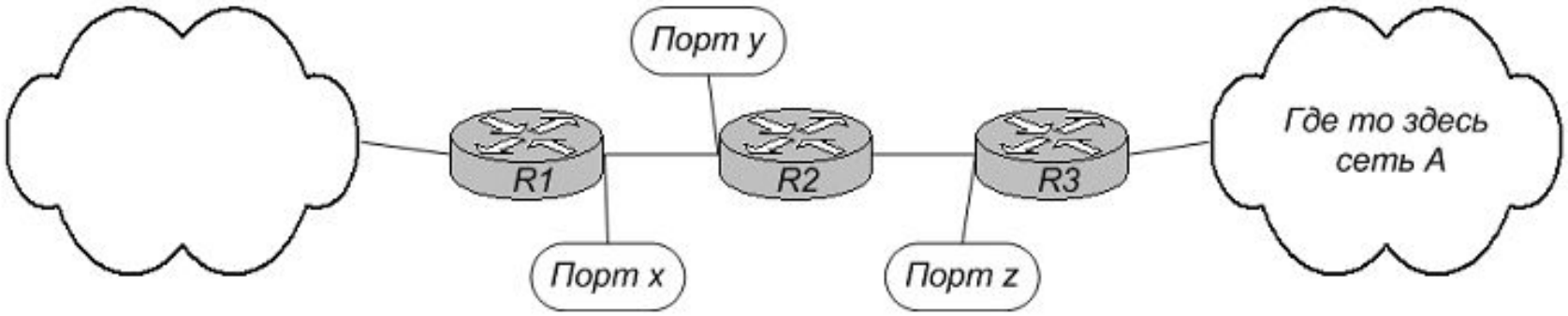
Когда маршрутизатор получает такое сообщение от своего соседа, он анализирует его следующим образом: из сообщения маршрутизатор выделяет информацию о каждой сети, о которой объявил соседний маршрутизатор, после чего анализирует каждое такое сообщение.

Рассмотрим, как анализируется одно сообщение о доступности той или иной сети:

- если маршрут в сеть, которая анонсируется в данном объявлении, неизвестен маршрутизатору получателю сообщения, то маршрутизатор получатель сообщения из данного объявления может получить информацию о том, как достичь данной сети. Как говорилось выше, в объявлении упоминаются только номера сетей и метрики, но не упоминаются адреса шлюзов, через которые объявляющий маршрутизатор достигает данной сети.

Но эти адреса совершенно не интересуют маршрутизатор получатель сообщения, так как тот маршрутизатор, через который достигает данной сети маршрутизатор отправитель, не находится (скорее всего, обычно) в одной канальной сети с маршрутизатором получателем сообщения.
Маршрутизатор, получивший сообщение будет достигать данной сети через маршрутизатор, пославший то сообщение, которое обрабатывается.

Проанализируем схему:



Пусть, в наборе сетей справа, где-то есть сеть А. Маршрутизатор R2 узнал об этом от маршрутизатора R3, и имеет в свое таблице маршрутизации очевидный маршрут вида:

Сеть назначения	Следующий маршрутизатор	Метрика
Сеть А	Порт Z	N

Маршрутизатор R2 объявляет об этом, отправляя широковещательно через все свои порты, в частности через порт Y о доступности известных ему сетей в частности и сети A. Это сообщение попадает на порт X маршрутизатора R1. Всообщении присутствует только номер сети A и метрика (цена доставки) известная маршрутизатору R2. Действительно, зачем маршрутизатору R1 знать, через какой порт маршрутизатор R2 достигает сети A (порт Z), если порт Z не достижим на канальном уровне маршрутизатору R1? Совершенно не необходимо.

Узнав, что маршрутизатору R2 известен маршрут в сеть A, маршрутизатор R1 тоже может заключить, что ему известен маршрут в сеть A, причем адресом следующего маршрутизатора, через который маршрутизатор R1 будет достигать сети A, будет тот порт, от которого он получил информацию о доступности сети A, т.е. порт, пославший обрабатываемый в настоящее время пакет - порт Y.

Таким образом, объявляющему маршрутизатору не нужно посылать в своих анонсах адреса следующих шлюзов - получатели этих объявлений и без этой информации понимают, что для них адрес следующего маршрутизатора это адрес пославшего объявление порта.

При этом, маршрутизатор R1 получил сведения о цене доставки в сеть A, которая известна маршрутизатору R2, для самого R1 цена доставки в сеть A очевидно выше. Если в качестве метрики использовать количество переходов, то очевидно, маршрутизатор R1 должен увеличить на единицу, полученную от маршрутизатора R2 метрику и записать в свою таблицу маршрутизации следующий маршрут:

Сеть назначения	Следующий маршрутизатор	Метрика
Сеть A	Порт Y	N++

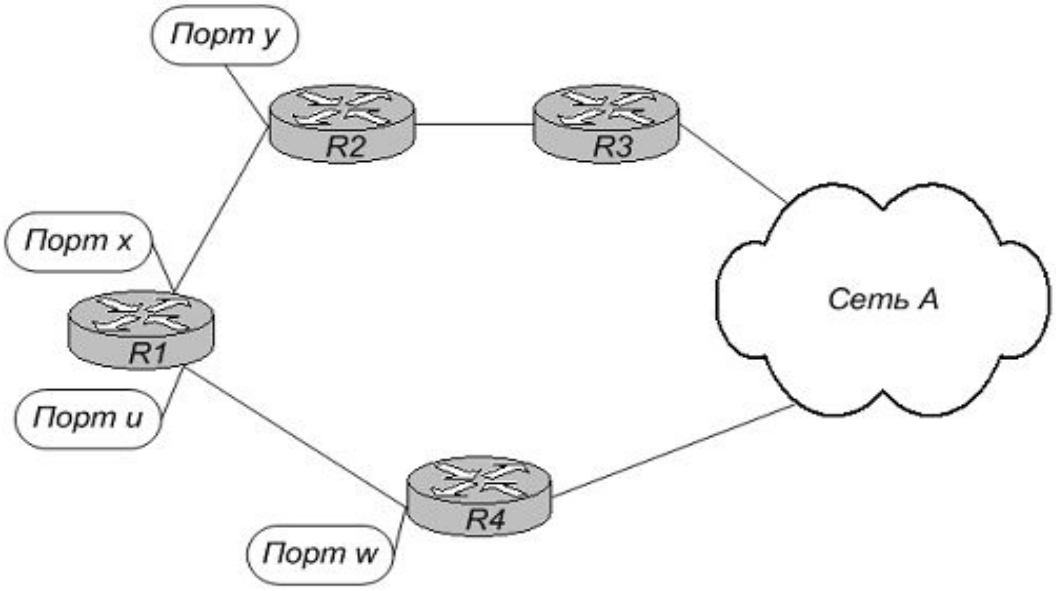
Таким образом, маршрутизатор R1 узнал от маршрутизатора R2 маршрут в сеть A.

Далее рассмотрим, что будет, если маршрут в сеть А и ранее был известен маршрутизатору R1. В таком случае возникает вопрос: каким маршрутом в сеть А выгоднее пользоваться маршрутизатору R2: тем, который был ему известен ранее или тем, который ему предлагает маршрутизатор R2.

Это вопрос решается с помощью метрики: если увеличенная на единицу метрика м аршрута в сеть А, полученного от маршрутизатора R2 меньше метрики маршрута в сеть А, ранее известного маршрутизатору R1, то, очевидно - предлагаемый маршрутизатором R2 маршрут в сеть А более предпочтителен (меньше стоит) нежели маршрут в сеть А, ранее известный маршрутизатору R1.

И тогда маршрутизатору R1 необходимо заменить ранее известный ему маршрут в сеть A на маршрут, пролегающий через маршрутизатор R2. А если увеличенная на единицу метрика маршрута в сеть A, полученного от маршрутизатора R2 более либо равна метрике маршрута в есть A, ранее известного маршрутизатору R1, то такой маршрут менее (или не более) предпочтителен, нежели известный ранее маршрут и маршрутизатору R1 нет необходимости менять свою запись в таблице маршрутизации относительно сети A, тогда данное объявление просто игнорируется.

Рассмотрим пример:



Маршрутизатор R4, к которому подключена сеть A объявляет о ней через порт W с метрикой 1, данное объявление получает маршрутизатор R1 через порт U, увеличивает метрику на 1 и заносит в свою таблицу маршрутизации запись вида:

Сеть A	Порт W	2

В то же время маршрутизатор R3, который тоже подключен к сети A объявляет о ее доступности маршрутизатору R2, который считает ее доступной с метрикой 2. Маршрутизатор R2 далее объявляет о доступности сети A с метрикой 2 через порт Y, это объявление получает маршрутизатор R1 через порт X, и маршрутизатор R1 таким образом узнает о доступности через порт Y сети A с метрикой 3.

Так как у маршрутизатора R1 уже есть запись в маршрутной таблицу о доступности сети A с метрикой 2, то информация о доступности этой же сети по другому маршруту с худшей метрикой будет отброшена.

Таким образом, работа данного алгоритма строится на следующих принципах:

1. Изначально все маршрутизаторы знают только маршруты в подключенные сети, метрика этих маршрутов составляет 0 или 1 (в зависимости от конкретного протокола)

2. Маршрутизаторы рассылают через все свои интерфейсы сведения о подключенных сетях и расстояний до них (0 или 1) Получая аналогичные объявления от других аршрутизаторов, маршрутизаторы узнают маршруты в сети, непосредственно подключенные к маршрутизаторам соседям.

3. Новые рассылки маршрутизаторы делают уже с учетом информации полученной от соседей, распространяя, таким образом, при втором объявлении сведения о большем количестве доступных сетей. Тогда, на каждом «круге» объявлений информация о доступности сетей в составной сети распространяется все большему числу маршрутизаторов.

4. Так продолжается до тех пор, пока все маршрутизаторы сети не узнают о доступности всех сетей, входящих в составную сеть и не построят полных таблиц маршрутизации.

Свое название - дистанционно векторный, алгоритм Беллмана-Форда получил вследствие того, что маршрутизаторы при взаимодействии с соседями, рассылают список доступных сетей и расстояний до них, такое объявление и называется вектор расстояний.

В случае, если в составной сети появится со временем новая сеть, информацию о доступности этой сети включит в свои объявления тот маршрутизатор, к которому она подключена. О доступности этой сети узнают его соседи, потом соседи соседей и так далее, пока все маршрутизаторы составной сети не узнают о доступности новой сети. При этом администратору не придется вносить вручную изменения в таблицы маршрутизации всех маршрутизаторов составной сети, т.е. таким образом (частично) достигается одна из преследуемых целей – уменьшение «ручного» труда администраторов, по конфигурированию маршрутных таблиц.

Однако, появление новых сетей не единственное изменение, которое может произойти в составной сети, кроме этого, могут выходить из строя маршрутизаторы и линии связи. В случае выхода из строя тех или иных линий связи или маршрутизаторов алгоритму необходим механизм, позволяющий маршрутизаторам удалять записи из своих таблиц маршрутизации, которые описывают маршруты в недоступные сети.

В алгоритме Беллмана-Форда это реализуется следующим образом: маршрутизаторы, к которым были подключены недоступные теперь сети, обязаны перестать включать информацию о доступности этих сетей в свои объявления.

Маршрутизаторы, имеющие косвенные сведения (т.е. полученные не на основании того факта, что данная сеть подключена к данному маршрутизатору, а полученные от других маршрутизаторов) о доступности тех или иных сетей, должны удалять из своей маршрутной таблицы записи о маршрутах в те сети, о доступности которых не было объявлений в течение некоторого интервала времени и, разумеется, не включать информацию о доступности этих сетей в свои собственные объявления.

Таким образом, когда некоторый маршрутизатор перестает объявлять о доступности подключенной к нему сети из-за отказа сети или отказа интерфейса самого маршрутизатора, соседние маршрутизаторы через некоторое время «забывают» о доступности этой сети и также перестают объявлять об этой сети. После чего, соседи, которые получали информацию о доступности данной сети от этих маршрутизаторов тоже «забывают» и так далее.

Тогда, в случае добавления в составную сеть новых сетей информация об их доступности постепенно распространяется всем маршрутизаторам составной сети, в случае недоступности некоторой сети информация об этом тоже постепенно распространяется ко всем маршрутизаторам составной сети.

Данный алгоритм весьма прост в реализации и позволяет получить желаемые результаты, впрочем, он не лишен и недостатков:

1. DVA протоколы порождают интенсивный сетевой трафик, так как маршрутизаторы достаточно часто должны рассылать свои сообщения, при этом так как в сообщении рассылается по сути вся таблица маршрутизации, то размер этих сообщений может быть велик.

2. Если уменьшить частоты объявлений маршрутизаторов, то информация о любом изменении доступности тех или иных сетей (появление или исчезновение сетей) будет распространяться медленно.

3. Информация о недоступных сетях в любом случае распространяется достаточно медленно, так как таймер «забывания» о доступности сети очевидно должен быть заметно больше таймера, регулирующего как часто маршрутизаторы должны рассылать свой вектор расстояний.

4. Маршрутизаторы не знают истинной топологии связей, а лишь владеют информацией о расстояниях до доступных сетей.

5. Данный алгоритм может приводить к появлению маршрутных петель при отказе некоторой сети, т.е. таких ситуаций, когда маршрутизаторы пересылают друг другу пакет в некоторую сеть, занимая тем самым пропускную способность линий связи. Более детально об этой проблеме DVA и о методах борьбы с этим явлением будет изложено позднее.

Далее рассмотрим второй подход к организации протоколов динамической маршрутизации – алгоритм Дейкстра или LSA (Link State Algorithm), так же этот алгоритм называется SPF (Shortest Path First).

Данный алгоритм заметно сложнее рассмотренного выше алгоритма DVA, и базируется на следующих принципах:

1. В алгоритме LSA маршрутизаторы, в отличие от DVA знают полную топологию связей в сети

2. Маршрутизаторы НЕ получают «куски» таблицы маршрутизации от других маршрутизаторов, вместо этого полная таблица маршрутизации строится каждым маршрутизатором на основании знания полной топологии связей сети

Работа протокола, базирующегося на алгоритме LSA, состоит из двух основных этапов:

1. сначала маршрутизаторы обмениваются друг с другом информацией о состоянии всех связей сети (или точнее AS), т.е. получают информацию о том, какие маршрутизаторы имеют друг с другом работающие связи. При этом все маршрутизаторы AS получают полную базу данных связей в своей AS.

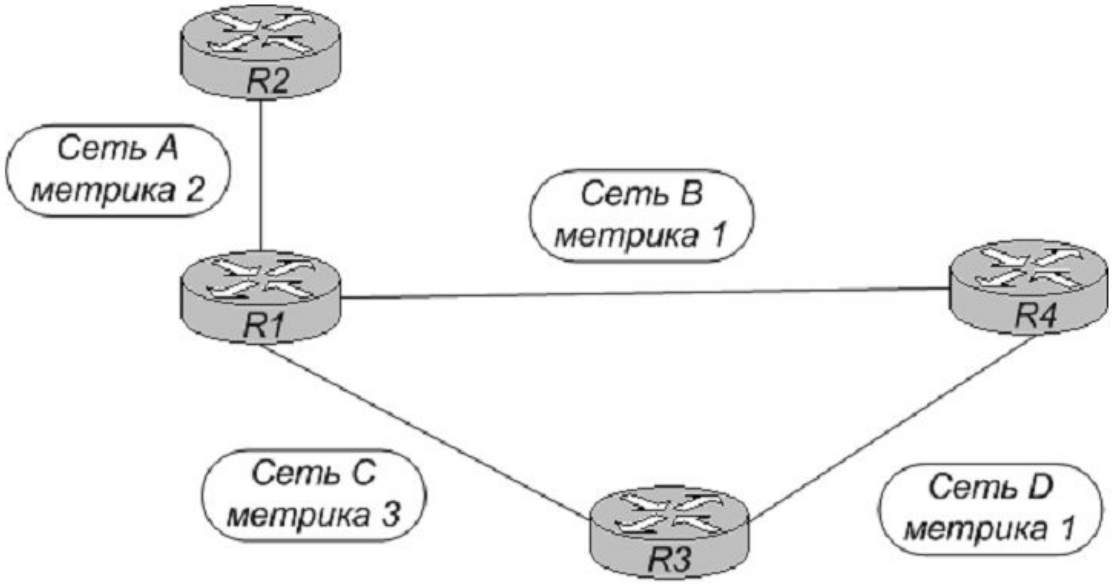
2. На втором этапе, когда каждый маршрутизатор владеет полной базой данных состояния связей в AS, каждый маршрутизатор на основании этой базы данных локально (путем локально проводимых вычислений) находит оптимальные маршруты во все сети, входящие в AS с помощью алгоритма SPF.

При этом, для обеспечения динамического реагирования на изменения в сети, соседние маршрутизаторы продолжают постоянно обмениваться друг с другом специальными сообщениями, с помощью которых проверяется, не отказали ли какие то связи и не появились ли какие то новые.

В таких пакетах не пересылается таблица маршрутизации, данные пакеты служат лишь для проверки того, что связь действует, поэтому размер сообщений такого рода крайне мал.

Если ни одна связь в AS не отказала, и не появилось никаких новых связей, то очевидно, таблицы маршрутизации, построенные маршрутизаторами адекватны, в противном случае нужно снова построить базу данных состояния связей AS, на основании которой затем маршрутизаторы снова построят таблицы маршрутизации. Таким образом, протоколы, базирующиеся на алгоритме LSA, приводят к меньшему трафику, но дополнительно нагружают маршрутизаторы расчетами.

Рассмотрим работу LSA на примере простой сети. Проанализируем только маршрутизаторы, соединенные связями точка-точка. И, пока, схема не будет содержать сети с узлами. Опишем порядок нахождения маршрутов не в конечные сети, а к маршрутизаторам, к которым подключены эти самые сети с узлами.



Алгоритм состояния связей рассматривает возможность учета не только количества переходов, которые необходимо совершить, чтобы достигнуть сети назначения, но, вообще говоря, может работать с любой метрикой. Полагается, что каждая сеть обладает некоторой метрикой, показывающей эффективность той или иной сети с той или иной точки зрения. Маршрут будет оптимальным, если суммарная метрика сетей, его образующих, минимальна.

Например, метрика может отражать пропускную способность сети (тогда чем выше скорость сети, тем меньше метрика и наоборот), задержки в сети или просто каждая сеть имеет одинаковую метрику (например, 1), что означает, что результирующий маршрут будет вычислен, по сути, на основании минимизации количества переходов.

Так как, возможность учета различных метрик часто применяется в протоколах семейства LSA, принимаем, что сети, составляющие данную автономную систему, имеют различные метрики, указанные на рисунке.

От кого -> К кому	Связь	Метрика
R1->R2	A	2
R1->R3	C	3
R1->R4	B	1
R2->R1	A	2
R3->R1	C	3
R3->R4	D	1
R4->R1	B	3
R4->R3	D	1

Для работы алгоритма SPF на каждом маршрутизаторе строится база данных состояния связей, представляющая собой полное описание всех связей между всеми маршрутизаторами системы. Пусть таблицы маршрутизации для каждого маршрутизатора построены и образуют единую базу данных. Рассмотрим, как будет выглядеть такая база данных для нашей сети, не забудем отметить, что база данных состояния связей идентична на всех маршрутизаторах AS.

Алгоритм Дейкстра позволяет каждому маршрутизатору на основании такой таблицы найти оптимальный маршрут к каждому маршрутизатору, т.е.
вычислить список сетей, соединяющих данных маршрутизатор с произвольным маршрутизатором по кратчайшему пути. Рассмотрим этот алгоритм.

Обозначим S тот маршрутизатор, который ищет на основании приведенной выше базы данных маршруты к прочим маршрутизаторам. Обозначим E множество маршрутизаторов, кратчайшие пути к которым уже найдены. Обозначим R множество оставшихся маршрутизаторов. Обозначим O упорядоченный список путей, некоторое вспомогательное множество, которое понадобятся нам в дальнейшем; P – кратчайший путь во вспомогательном списке O.

Алгоритм:

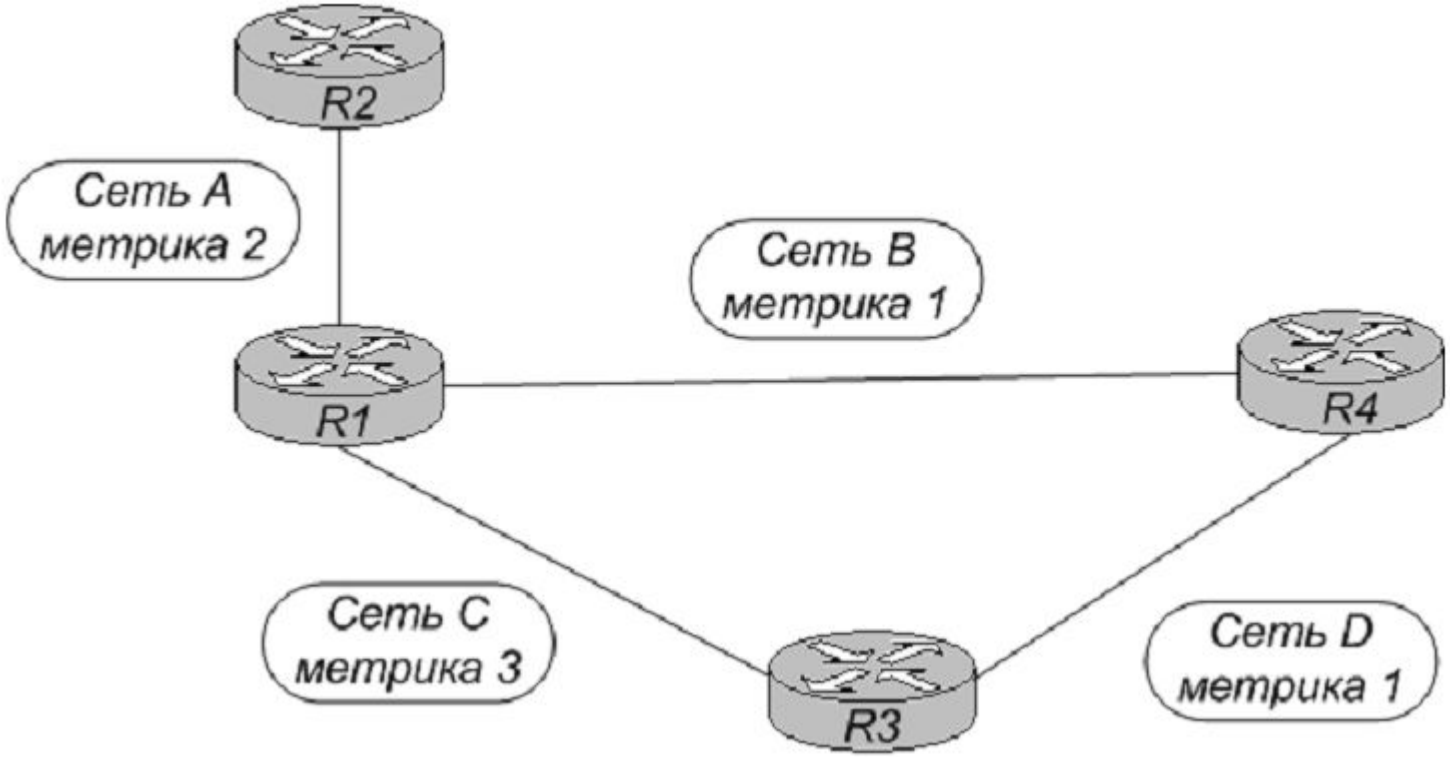
1. В множество E занесем маршрутизатор S, так как маршрут к нему тривиален, в множество R – все остальные маршрутизаторы, так как кратчайшие пути к ним еще не известны. Поместим в множество O все пути, начинающиеся, в соответствии с приведенной выше базой данных, из S, отсортировав их в порядке возрастания метрик.

2. Если O пуст, то отметить все вершины в R как недостижимые и закончить работу алгоритма. Смысл данной проверки на первом шаге очевиден: если у маршрутизатора нет ни одной связи, то он не сможет достичь ни одного маршрутизатора и они (а так же подключенные к ним сети) являются недостижимыми.

3. Рассмотрим P – кратчайший путь во вспомогательном списке O. Удалим P из O. Рассмотрим V – маршрутизатор, к которому ведет P. Если маршрутизатор V принадлежит множеству E, перейти к шагу 2. Иначе, P является кратчайшим путем из S в V (записывается как V:P), переносим V из множества R в множество E.

4. Строим новый набор путей, подлежащих рассмотрению, путем добавления к пути P всех односегментных путей, начинающихся из V. Метрика каждого нового пути равна сумме метрики P и метрики соответствующего односегментного отрезка, начинающегося из V. Добавить новые пути в упорядоченный список O, поместив их на места в соответствии со значениями метрик. Переходим на шаг 2.

Рассмотрим применение алгоритма Дейкстра на примере данной сети:



при нахождение кратчайших путей для маршрутизатора R3.

Тогда S=R3, E=R3, так как к маршрутизатору R3 кратчайшего пути искать не нужно.
В множество R занесем R1, R2, R4, так как кратчайшие пути к этим маршрутизаторам еще не найдены.
В множество O запишем пути D, C в порядке возрастания метрик. Так как множество O не пусто, переходим к пункту 3.

Шаг 1. В качестве P выберем путь D, так как сеть D ведет к маршрутизатору R4, V=R4. Так как R4 не принадлежит множеству E, то кратчайший путь к маршрутизатору R4 и есть путь P. Отметим этот факт в таблице результатов (маршрут к R4 есть D), удаляем P из O, переносим V из R в E. В результате: множество E содержит R3 и R4, множество R содержит R2 и R1, список O содержит C. Выполнив третий шаг алгоритма, переходим к четве ртому: к пути P (это путь D) добавляем все пути, начинающиеся из R4, таких путей два B и D. В итоге получаем пути DD (метрика 2) и DB (метрика 2). Итого, перед переходом на шаг 2 и началом нового цикла расчетов наши множества имеют вид:

E=R3, R4

R=R1, R2

O=DD, DB, C

Результат: кратчайший путь к R4 – D (или R4:D)

Шаг 2. В качестве P выберем первый путь из множества O, т.е. DD. Двигаясь по этому пути из R3 мы снова попадаем в R3, т.е. V=R3. Так как V (т.е. R3) принадлежит множеству E, то исключив P (т.е. DD) из множества Опереходим к пункту 2. Множества имеют вид:

E=R3, R4

R=R1, R2

O=DB, C

Результат: кратчайший путь к R4 – D (или R4:D)

Шаг 3. В качестве P выберем первый путь из множества O, т.е. P=DB. Двигаясь по этому пути от маршрутизатора R3 попадаем на маршрутизатор R1, таким образом V=R1. Так как V не принадлежит множеству E, то это означает, что кратчайший путь от R3 к R1 найден и этот кратчайший путь есть P, т.е. DB. Добавим этот факт в таблицу результатов и удалим P из множества O. Переносим V из множества R в множество E. Выполняем четвертый этап алгоритма, строим новые пути. Из V=R1 существует три односегментных пути (A, B, C), добавим их к P=DB, получаем пути DBA (метрика 4), DBB (метрика 3) и DBC (метрика 5). Добавим эти пути, в множество O, произведем сортировку множества O по возрастанию метрики. Множества имеют вид:

E=R3, R4, R1

R=R2

O=C, DBB, DBA, DBC

Результаты: кратчайший путь к R4 – D (или R4:D)
кратчайший путь к R1 – DB (или R1:DB)

Шаг 4. В качестве P выберем первый путь из множества O, т.е. P=C. Двигаясь по этому пути от маршрутизатора R3 попадаем на маршрутизатор R1, таким образом V=R1. Так как V принадлежит множеству E, исключаем P=C из множества O и переходим к следующему шагу. Множества имеют вид:

E=R3, R4, R1

R=R2

O=DBB, DBA, DBC

Результаты: кратчайший путь к R4 – D (или R4:D)

кратчайший путь к R1 – DB (или R1:DB)

Шаг 5. В качестве P выберем первый путь из множества O, т.е. P=DBB. Двигаясь по этому пути от маршрутизатора R3 попадаем на маршрутизатор R4, таким образом V=R1. Так как V принадлежит множеству E, исключаем P=DBB из множества O и переходим к следующему шагу. Множества имеют вид:

E=R3, R4, R1

R=R2

O=DBA, DBC

Результаты: кратчайший путь к R4 – D (или R4:D)

кратчайший путь к R1 – DB (или R1:DB)

Шаг 6. В качестве P выберем первый путь из множества O, т.е. P=DBA. Двигаясь по этому пути от маршрутизатора R3 попадаем на маршрутизатор R2, таким образом V=R2. Так как V не принадлежит множеству E, то это означает, что мы нашли кратчайший путь от R3 к R2 и этот кратчайший путь есть P, т.е. DBA. Добавим в таблицу результатов соответствующую запись, удалим P из множества O, перенесем V из множества R в множество E. Выполняем четвертый этап алгоритма, строим новые пути. Из V=R2 существует один односегментный путь A, добавим их к P=DBA, получаем путь DBAA (метрика 6). Добавим этот путь, в множество O, производим сортировку множества O по возрастанию метрики. Множества имеют вид:
E=R3, R4, R1, R2
O=DBC, DBAA

Результаты: кратчайший путь к R4 – D (или R4:D)
кратчайший путь к R1 – DB (или R1:DB)
кратчайший путь к R2 – DBA (или R2:DBA)

Шаг 7. В качестве P выберем первый путь из множества O, т.е. P=DBC. Двигаясь по этому пути от маршрутизатора R3, попадаем на тот же маршрутизатор R3, таким образом, V=R3. Так как V принадлежит множеству E, исключим P=DBC из множества O и перейдем к следующему шагу. Множества имеют вид:

E=R3, R4, R1, R2

R=

O=DBAA

Результаты: кратчайший путь к R4 – D (или R4:D)

кратчайший путь к R1 – DB (или R1:DB)

кратчайший путь к R2 – DBA (или R2:DBA)

Шаг 8. В качестве P выберем первый путь из множества O, т.е. P=DBAA. Двигаясь по этому пути от маршрутизатора R3, попадаем на маршрутизатор R1, таким образом, V=R1. Так как V принадлежит множеству E, исключим P=DBAA из множества O и перейдем к следующему шагу. Множества имеют вид:

E=R3, R4, R1, R2

R=

O=

Результаты: кратчайший путь к R4 – D (или R4:D)
кратчайший путь к R1 – DB (или R1:DB)
кратчайший путь к R2 – DBA (или R2:DBA)

Шаг 9. Так как множество O пусто, работу алгоритма необходимо завершить. Так как множество R пусто, то недостижимых маршрутизаторов нет.

Результатом работы алгоритма SPF является таблица кратчайших путей ко всем маршрутизаторам сети от маршрутизатора R3. Разумеется, все остальные маршрутизаторы сети, выбрав в качестве S себя, так же нашли кратчайшие пути ко всем остальным маршрутизаторам. Итак, таблица кратчайших путей от R3 ко всем остальным маршрутизаторам имеет вид:

кратчайший путь к R4 – D (или R4:D)
кратчайший путь к R1 – DB (или R1:DB)
кратчайший путь к R2 – DBA (или R2:DBA)

Теперь на основании таблицы кратчайших путей маршрутизатору R3 необходимо построить таблицу маршрутизации ко всем остальным маршрутизаторам сети (пока рассматривается упрощенный случай- т.е. рассматриваются маршруты не в сети, а к маршрутизаторам). Очевидно, в соответствии с концепцией IP маршрутизации маршрутизатору нет необходимости знать ВЕСЬ маршрут к цели, а достаточно знать лишь адрес первого маршрутизатора. Для нахождения таблицы маршрутизации из таблицы кратчайших путей достаточно отсечь в каждом найденном кратчайшем пути все переходы кроме первого. Маршрутизатор, к которому ведет первая связь в кратчайшем пути, и будет являться следующим маршрутизатором для данного маршрута.

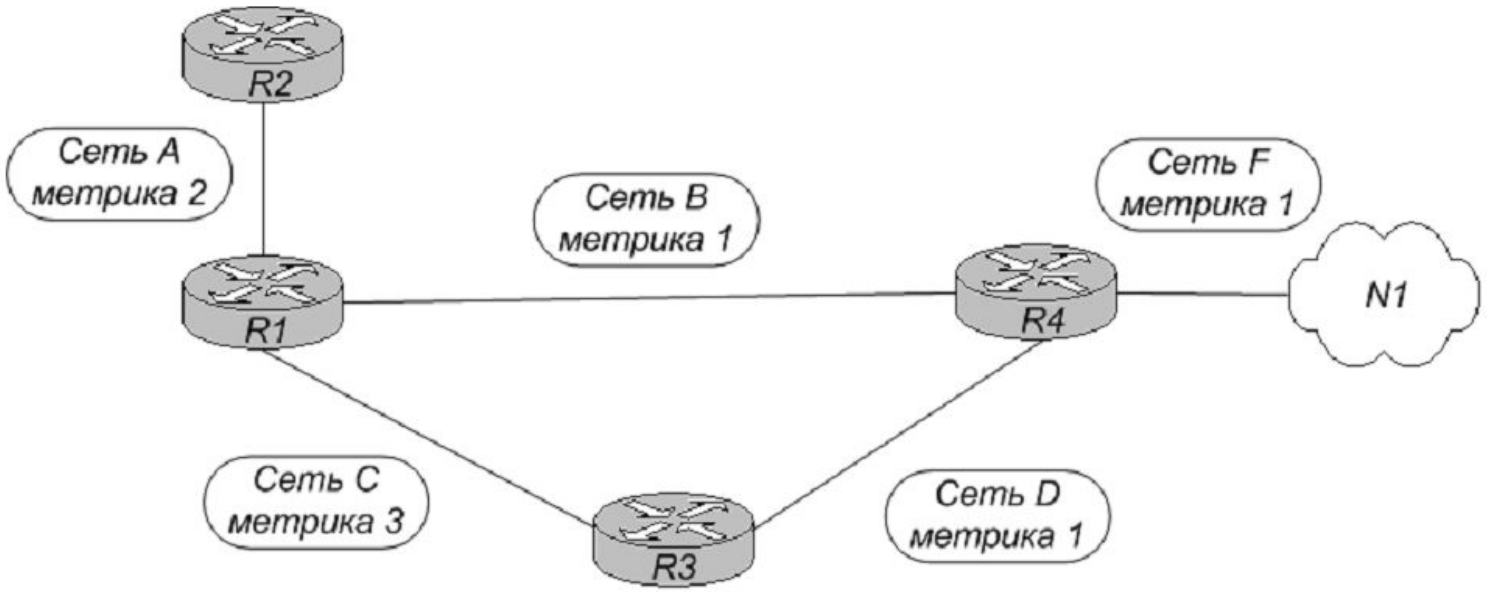
Почему же найденные с помощью алгоритма результаты являются кратчайшими путями?

- начиная от точки S, т.е. от того маршрутизатора, для которого строится таблица алгоритм записывает все связи, ведущие от этого маршрутизатора в множество и УПОРЯДОЧИВАЕТ этот список по возрастанию метрики, т. е. всегда сперва рассматривает пути с минимальной метрикой или кратчайшие пути. Пройдя по заведомо кратчайшему из путей на первом шаге, алгоритм куда то попадает. Обратите внимание, что алгоритм НЕ ищет кратчайшего пути К НЕКОТОРОМУ маршрутизатору, вместо этого алгоритм ищет просто кратчайшие пути и смотрит, куда они приводят.

Если алгоритм попадает на некоторый маршрутизатор, то он совершенно однозначно может сделать вывод, что на ЭТОТ маршрутизатор не существует более короткого пути потому, что если бы он существовал, то этот путь из множества О был бы обработан РАНЕЕ, нежели тот путь, который найден. Т.е. анализ делается кратчайших путей, а затем уже выясняется, куда они ведут. Поэтому все найденные алгоритмом пути – кратчайшие.

Добавим в рассматриваемую систему сети с конечными узлами, или тупиковые сети.
Положим, что к маршрутизатору R4 подключена некоторая тупиковая сеть N1. Тупиковая сеть содержит в себе узлы, которые так же можно добавить в рассмотренную выше схему SPF наравне с маршрутизаторами (т.е. рассматривать каждый узел как вершину графа) для того, чтобы все маршрутизаторы могли вычислить маршруты к этим узлам, не смотря на то, сами узлы не строят своих таблиц маршрутизации. Решение добавить в работу алгоритма все узлы наравне с маршрутизаторами и рассчитывать, как приведено выше маршруты ко всем узлам, очевидно, не слишком эффективно, так как приведет к значительному возрастанию вычислительной нагрузки, ложащейся на маршрутизаторы, притом что, маршруты ко всем узлам одной тупиковой сети очевидно идентичны.

Для того, чтобы оптимизировать нагрузку на вычислительные ресурсы маршрутизаторов поступают следующим образом - ВСЮ тупиковую сеть как единое целое рассматривают в рамках работы алгоритма SPF наравне с маршрутизаторами, т.е. считают вершиной графа.



В таком случае в базе данных связей появится запись о том, что маршрутизатор R4 связан с сеть N1 связью F, при этом информации о том, что сеть N1 связана с маршрутизатором R4 не будет. База данных связей будет иметь вид:

От кого -> К кому	Связь	Метрика
R1->R2	A	2
R1->R3	C	3
R1->R4	B	1
R2->R1	A	2
R3->R1	C	3
R3->R4	D	1
R4->R1	B	3
R4->R3	D	1
R4->N1	F	1

При этом работа алгоритма SPF ни коим образом не меняется. Рассмотрим работу SPF в случае наличия такой тупиковой сети для маршрутизатора R3.

Итак, исходные данные:

E=R3

R=R1, R2, R3, N1

O=C, D

Результаты: пока нет

Шаг 1. Все как и в первом примере: P=D, так как сеть D ведет к маршрутизатору R4, то V=R4. V не принадлежит E, поэтому кратчайший путь к R4 и есть P. Зафиксируем этот факт в таблице результатов, удалим P из O, перенесем V из R в E. Затем в пути P добавим все пути, начинающиеся из R4, таких путей три F, B и D, в итоге получим пути DF (метрика 2), DD (метрика 2) и DB (метрика 2). Итого, перед переходом на шаг 2 и началом ново го цикла
расчетов множества имеют вид:
E=R3, R4
R=R1, R2, N1
O=DF, DD, DB, C
Результат: кратчайший путь к R4 – D (или R4:D)

Шаг 2. P =DF. V=N1. Так как N1 не принадлежит E, то найден кратчайший путь к N1, этот путь N1:DF. Удалим P из O. Так как из N1 никакие пути не ведут (в базе данных связей, связи тупиковая сеть - маршрутизатор не вносятся в базу данных), то ничего с множеством делать не нужно, просто переходим к третьему шагу алгоритма.
E=R3, R4, N1
R=R1, R2
O=DD, DB, C
Результаты: кратчайший путь к R4 – D (или R4:D)
кратчайший путь к N1 – DF (или N1:DF)

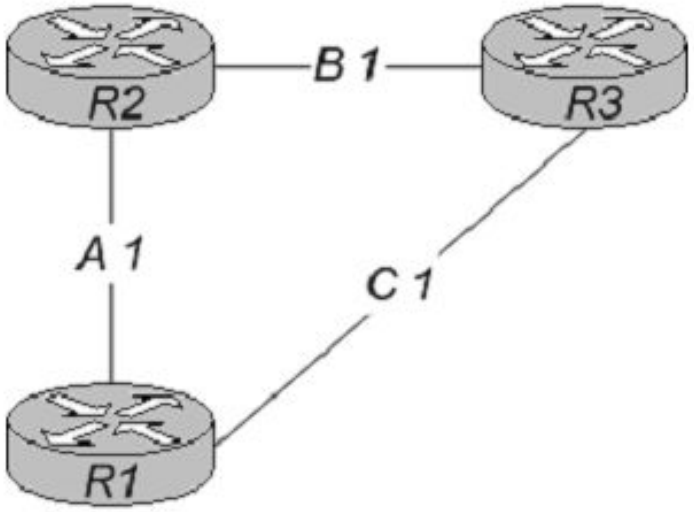
Очевидно, что дальнейшая работа алгоритма не отличается от прошлого случая с учетом того, что все остальные шаги из прошлого примера в этом примере будут фигурировать с увеличенными на 1 номерами.

Еще одним достоинством алгоритма SPF является возможность расширения алгоритма для нахождения не только кратчайшего, но и всех возможных маршрутов, затем протокол динамической маршрутизации может использовать эту информацию для перераспределения отправляемых пакетов по нескольким маршрутам с целью балансировки нагрузки на линии связи (отказоустойчивость не имеется ввиду, так как в случае отказа одной из линий связи маршрутизаторы создают новую базу данных связей и заново находят маршруты с помощью SPF).

Заставить маршрутизаторы находить не только кратчайшие пути, но любые возможные пути между маршрутизаторами на первый взгляд очень просто - достаточно внести простейшую модификацию в третий пункт нашего алгоритма: если V принадлежит E, то достаточно отметить P как альтернативный маршрут в V. Из рассмотренного выше принципа рассмотрения в первую очередь кратчайших путей следует, что найденный путь будет хуже, чем уже известный, но таким образом можно найти все, а не только кратчайшие пути.

Если протокол маршрутизации захочет балансировать нагрузку на линии связи, перераспределяя трафик между альтернативными маршрутами (это определяется не самим алгоритмом SPF, а протоколом маршрутизации), отправляя, например, трафик к цели обратно пропорционально метрикам всех возможных маршрутов, то возможно серьезная проблема: могут возникать частичные маршрутные петли.

Рассмотрим пример:

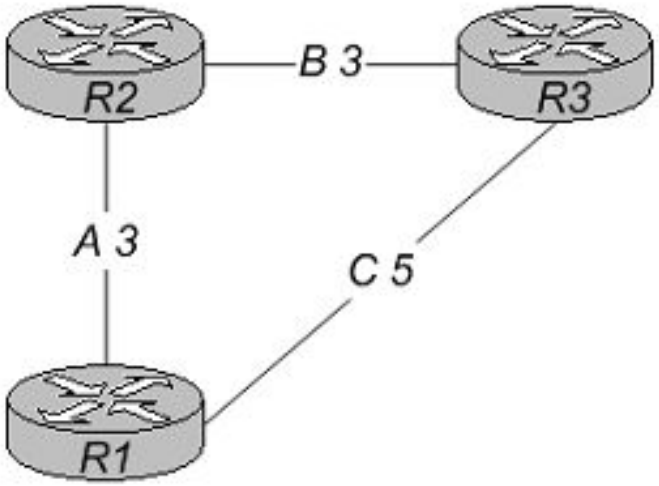


Допустим маршрутизатор R1 отправляет данные на маршрутизатор R3 с помощью некоторого протокола маршрутизации, который использует балансировку нагрузки между альтернативными маршрутами, тогда по маршруту C (метрика 1) будет отправляться 2/3 трафика, а по маршруту AB (метрика 2) будет отправляться 1/3 трафика. Предположим, что маршрутизатор R2 поддерживает ту же технику, в таком случае маршрутизатор R2 тоже имеет два маршрута к маршрутизатору R3: B (метрика 1) и AC (метрика 2) и отправляет обратно маршрутизатору R1 1/9 от общего числа тех пакетов, которые маршрутизатор R1 отправил. Итого, примерно 1/9 пакетов вернется на R1, 1/27 часть пакетов сделает это дважды и так далее. Налицо проблема.

Можно сказать, что эта проблема не касается SPF: он только предлагает маршруты, причем позволяет найти кратчайший, в таком неэффективном использовании линий связи виноват сам протокол маршрутизации, выполняющий столь бессмысленное перенаправление пакетов. Однако, изменения все же проще внести в работу алгоритма SPF таким образом, чтобы подобные ситуации были не возможны. Необходимо ограничить число альтернативных маршрутов, которые находит SPF, рассмотрим, как это делается.

Вводится следующее правило: если узел X отправляет данные в узел Y, он может пересылать их
через узел Q только в том случае, если Q ближе к Y, чем X к Y. В разобранном выше примере, следуя этому правилу, R1 не может посылать данные в R3 через R2, поскольку R2 не ближе к R3 , чем R1.

Однако такая посылка возможна, если связи между узлами имеют метрики как в следующем примере:



Допустим маршрутизатор R1 отправляет данные на маршрутизатор R3 с помощью некоторого протокола маршрутизации, который использует балансировку нагрузки между альтернативными маршрутами, тогда и по маршруту C (метрика 5) и маршруту AB (метрика 6) будут отправляться данные. Действительно, R2 ближе к R3 нежели R1 к R3. Но при этом маршрутизатор R2 не станет отправлять часть этих данных обратно к R1, так как R1 дальше от R3 нежели сам R1. Фактически работает следующее правило: использовать можно только те маршруты, которые ведут «вперед», но не «назад».

Можно без особого труда заставить протокол SPF находить помимо кратчайшего маршрута только те альтернативные маршруты, которые удовлетворяют описанному выше правилу, и как следствие с помощью SPF можно строить маршрутные таблицы, лишенные описанного выше недостатка. Формирование базы данных состояния связей в алгоритме LSA делается с помощью объявлений, которыми (широковещательно или с помощью групповой адресации) обмениваются соседние маршрутизаторы друг с другом.

Таким образом, преимущества алгоритма LSA состоят в том, что маршрутизаторы владеют полной информацией о топологии сети, алгоритм порождает менее интенсивный сетевой трафик, так как маршрутизаторы не рассылают постоянно свою таблицу маршрутизации, а обмениваются друг с другом короткими пакетами с целью проверки работоспособности каналов связи. В свою очередь это означает, что LSA протоколы более приспособлены работать в крупных AS нежели DVA, так как чем больше маршрутов в сети, тем более интенсивный трафик порождают протоколы семейства DVA. Недостатки LSA в основном состоят в значительной вычислительной нагрузке, которая ложится на маршрутизаторы и как следствие, аппаратные требования к маршрутизаторам для поддержки протоколов, основанных на алгоритмах LSA значительно выше.

Рассмотренные алгоритмы построения маршрутных таблиц – DVA и LSA позволяют автоматизировать работу администраторов связанную с настройкой маршрутизации в крупных сетях. Но это не означает полное исключение необходимости управления сетевыми структурами со стороны администратора. В этом случае, главной задачей администратора является правильная настройка протоколов маршрутизации использующих вышеприведенные алгоритмы, а это возможно только в случае полного понимания принципов работы алгоритмов - на уровне способности прогнозировать любое действие маршрутизатора работающего под управлением заданного протокола маршрутизации.

Рассмотренные алгоритмы построения маршрутных таблиц – DVA и LSA позволяют автоматизировать работу администраторов связанную с настройкой маршрутизации в крупных сетях. Но это не означает полное исключение необходимости управления сетевыми структурами со стороны администратора. В этом случае, главной задачей администратора является правильная настройка протоколов маршрутизации использующих вышеприведенные алгоритмы, а это возможно только в случае полного понимания принципов работы алгоритмов - на уровне способности прогнозировать любое действие маршрутизатора работающего под управлением заданного протокола маршрутизации.