Маршрутизация със следене състоянието на връзката.

Йерархична маршрутизация

Link State vs. distance vector

Информацията в един distance vector (DV) е сравнима с пътния знак.

Докато link state (**LS**) протоколите са пътна карта.

LS рутерът има пълна картина на мрежата и и по-трудно ще вземе неправилно решение.

DV маршрутизират по слухове.

Link State. Особености.

LS – подава на съседите си информация за директно свързаните връзки и състоянието им (затова е link state).

Тази информация се подава от рутер на рутер, всеки я копира, без да я променя. Всеки рутер има идентична информация за мрежата.

Всеки рутер сам за себе си изчислява най-добрите пътища до съответните префикси.

Алгоритъм на Dijkstra. Един от AI алгоритмите.

За сведение:

https://www.goodrequest.com/blog/the-introduction-to-artificial-intelligence-algorithms-5-dijkstra

LS протоколите се базират на алгоритъма на E. W. Dijkstra - shortest path.

Dijkstra е един от най-простите алгоритми за намиране на най-късият път в граф.

Алгоритъм на Dijkstra

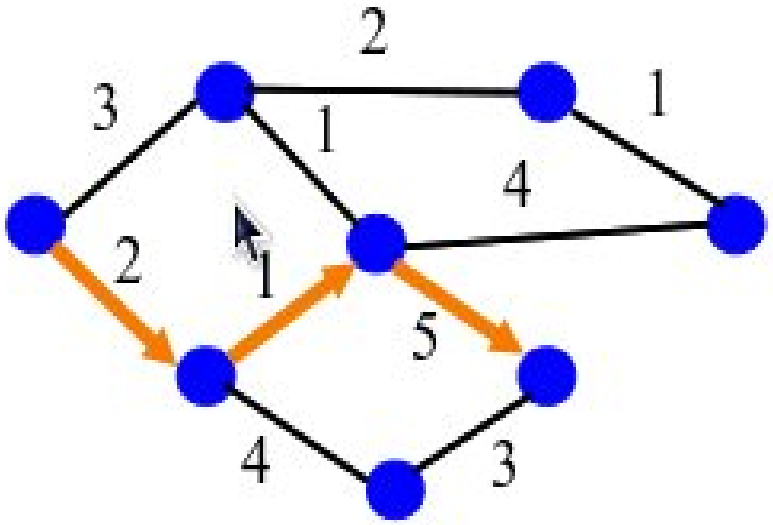
Алгоритъмът на Dijkstra е алгоритъм за намиране на най-къс път в граф от даден връх до всички останали върхове.

Теглата на ребрата на графа трябва да са положителни числа.

Резултатът от алгоритъма е дърво на оптималните пътища от дадения връх до всички останали.

Т.е Shortest Path Tree (SPF).

Алгоритъм на Dijkstra



Пет основни действия

При маршрутизацията със следене състоянието на връзката (link state routing) всеки маршрутизатор трябва да извършва следните пет основни действия:

1. Откриване на съседните маршрутизатори и техните мрежови адреси.
2. Измерване на стойностите на връзките до съседните маршрутизатори.

Пет основни действия

3. Конструиране на пакети с информация за състоянието на връзките.
4. Изпращане на тези пакети до всички останали маршрутизатори.
5. Изчисляване на най-късия път до всеки маршрутизатор в мрежата.

В резултат на тези пет действия се събира и разпространява до всички маршрутизатори информация за цялата топология на мрежата (**LSDB**).

Откриване на съседните маршрутизатори

След включването на един маршрутизатор неговата първа задача е да научи кои са съседите му.

Това се постига чрез изпращане на “exo” пакет по всяка от изходящите линии на маршрутизатора.

От своя страна, всеки от съседите отговаря като съобщава името си. Това име трябва да бъде уникално в мрежата. (Например, IP адрес на някой от интерфейсите.)

Откриване на съседните маршрутизатори

Ако два или повече маршрутизатора са свързани в мрежа с общодостъпно (broadcast) предаване (например Ethernet), откриването на съседите е малко по-сложно.

Трябва да се установи съседство „всеки с всеки“, което е свързано с генериране на големи обеми служебен трафик.

Затова се въвежда главен възел (designated router), който установява съседство с останалите възли (рутери).

link state packets

След като събере необходимата информация за състоянието на връзките си, следващата задача на маршрутизатора е да конструира пакет, който съдържа следната информация:

уникалното име на рутера-подател, пореден номер, срок на годност и списък със съседите на подателя, като за всеки съсед е указана цената на връзката до него.

Определянето на момента, в който трябва да бъдат подготвени и изпратени пакетите, е важна задача.

link state packets

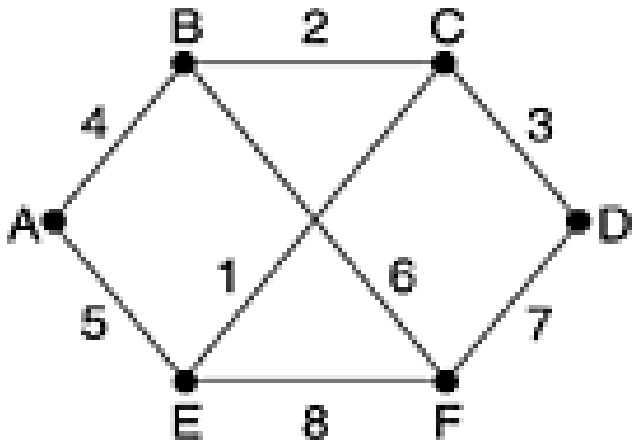
Един възможен начин е това да става през определени равни интервали от време.

Друга по-добра възможност е пакетите да се подготвят и изпращат само при промяна в топологията на мрежата - след отпадане или поява на нов съсед или промяна в цената на някоя връзка.

Нека да разгледаме следната примерна мрежа. Ребрата имат етикети със съответното времезакъснение.

link state packets

Пакетите със състоянието за връзките за шестте маршрутизатора изглеждат по следния начин:



Link State Packets

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

Разпространяване на **link state packets**

Най-съществената част на алгоритъма е надеждното доставяне на пакетите с информацията за състоянието на връзката до всички маршрутизатори.

За разпространението на пакетите се използва методът на наводняването (flooding). При него всеки пакет се изпраща по всички линии, освен линията по която е пристигнал.

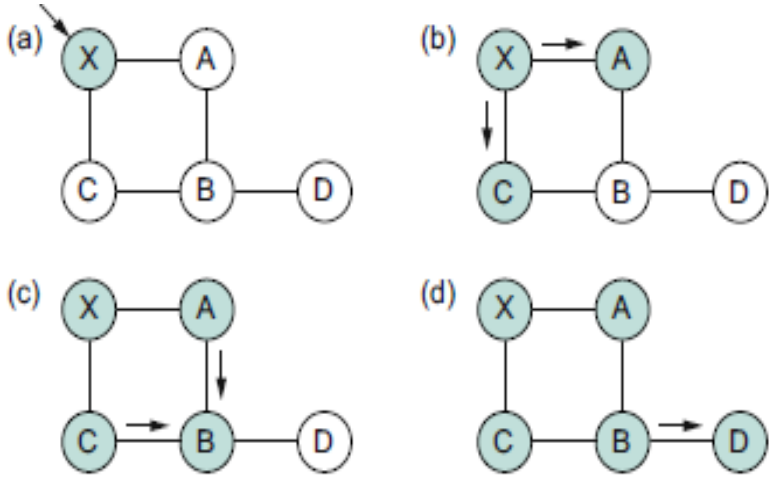
Разпространяване на **link state packets**

Обработката на всеки пристигнал пакет започва с проверка дали пакетът има по-голям пореден номер в сравнение с най-големия пореден номер, който е пристигнал до този момент от този източник.

Ако номерът е по-голям, информацията от пакета се записва в таблицата с информация за състояние на връзките и пакетът се предава по останалите линии.

Ако номерът е по-малък или равен, пакетът се отхвърля.

Разпространяване (**Flooding**) на **link state packets**



Пореден номер

Този алгоритъм има някои проблеми.

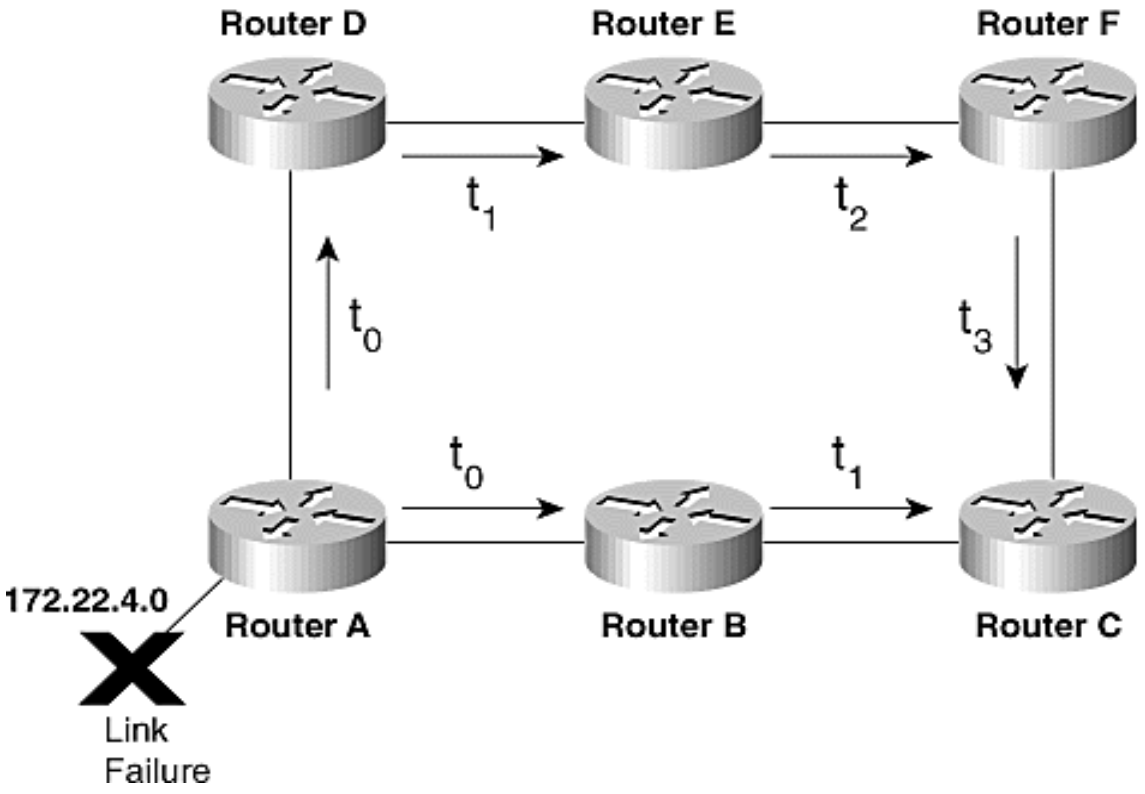
Ако поредният номер не е достатъчно голям, той бързо се превърта.

Затова се използват 32-битови поредни номера.

Ако пристига 1 пакет/s, то за превъртане на номера ще са необходими около 137 години.

Ако пристига 1 пакет/10 s (на практика) - 1361 години.

Пореден номер. Пример.



Пореден номер. Пример.

Събитие: префикс 172.22.4.0 (рутер A) отпада.

A разпространява (flood-ва) LSA на съседи B и D. B и D на техните съседи и т.н.

На рутер C: пристига LSA от B в момент t_1, и влиза в топол. БД на C и се отправя към F.

В t_3, пристига същото копие на LSA по пътя A-D-E-F-C.

Рутер C вече има LSA в топол. БД и няма да го отправи към B. Поредният номер на LSA от F е равен на LSA от B.

Пореден номер. Пример.

Второ събитие: 172.22.4.0 пада и след това се вдига.

Рутер A изпраща LSA за падането с номер 166.

След това нов LSA за вдигането с номер 167.

Рутер C получава “падналото LSA” и след това “вдигнато LSA” по пътя A-B-C, но по-късно LSA от A-D-E-F-C path.

Какво щеше да стане, ако нямаше пореден номер (sequence number)?

Но LSA 167 вече е в топол. БД и закъсняло LSA 166 не може да заблуди рутер C.

Поле за срок на годност

В полето за срок на годност маршрутизаторът-подател указва продължителността на интервала от време в секунди, през който пренасяната от него информация трябва да се счита за валидна.

Всеки маршрутизатор, който получи даден пакет намалява с единица стойността на това поле преди да го предаде към своите съседи.

Поле за срок на годност

Освен това, след като маршрутизаторът запише данните от пакета в своята таблица, той продължава да намалява срока на годност на тези данни на всяка следваща секунда.

Ако срока на годност стане 0, данните се изтриват.

По този начин се премахва опасността остаряла информация за състоянието на връзките да се разпространява и използва прекалено дълго време от маршрутизаторите.

Прилагане на SPF алгоритъма

E. W. Dijkstra: “Да се конструира дърво с минимална обща дължина между всичките ***n*** възела.”

(Дървото е граф с един и само един път между всеки два възела.)

Как става:

Дъгите се разделя на три множества:

Прилагане на SPF алгоритъма

I. Дъгите, присвоени на “строящото се” дърво;

II. Дъгите, от които се избира следващата за добавяне в I;

III. Останалите дъги.

Възлите се делят на две множества:

A. Свързаните с дъги от множество I,

B. Останалите (една и само една дъга от II води до един от тези възли).

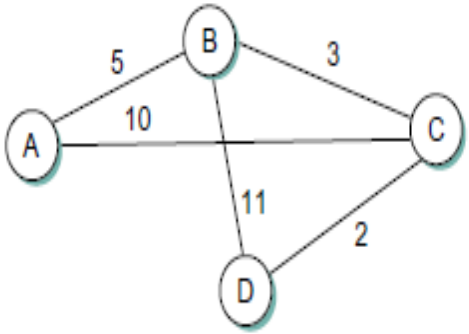
Прилагане на SPF алгоритъма

SPF Tree Database. БД, съответства на **множество I**.

Кандидатска БД. БД, съответства на **множество II**.

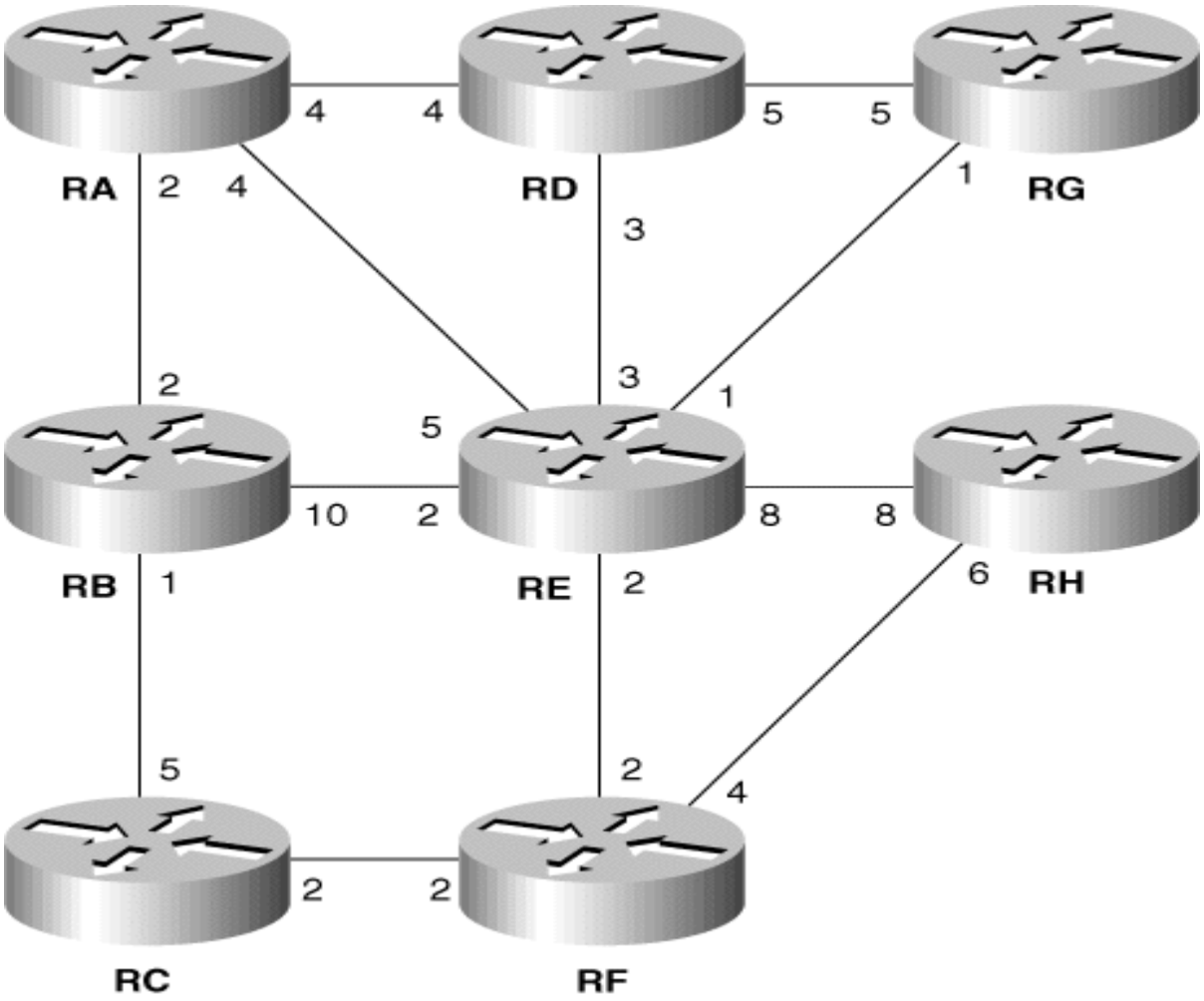
Link State Database. Топологичната БД, съответства на **множество III**.

SPF алгоритъм. Прост пример.



1	(D,0,–)	
2	(D,0,–)	(B,11,B) (C,2,C)
3	(D,0,–) (C,2,C)	(B,11,B)
4	(D,0,–) (C,2,C)	(B,5,C) (A,12,C)
5	(D,0,–) (C,2,C) (B,5,C)	(A,12,C)
6	(D,0,–) (C,2,C) (B,5,C)	(A,10,C)
7	(D,0,–) (C,2,C) (B,5,C) (A,10,C)	

SPF алгоритъм. По-сложен пример.



Link State Database. Извлечение.

Router ID	Neighbor	Cost
RA	RB	2
RA	RD	4
RA	RD	4
RB	RA	2
RB	RC	1
RB	RE	10
RC	RB	5
RC	RF	2
RD	RA	4

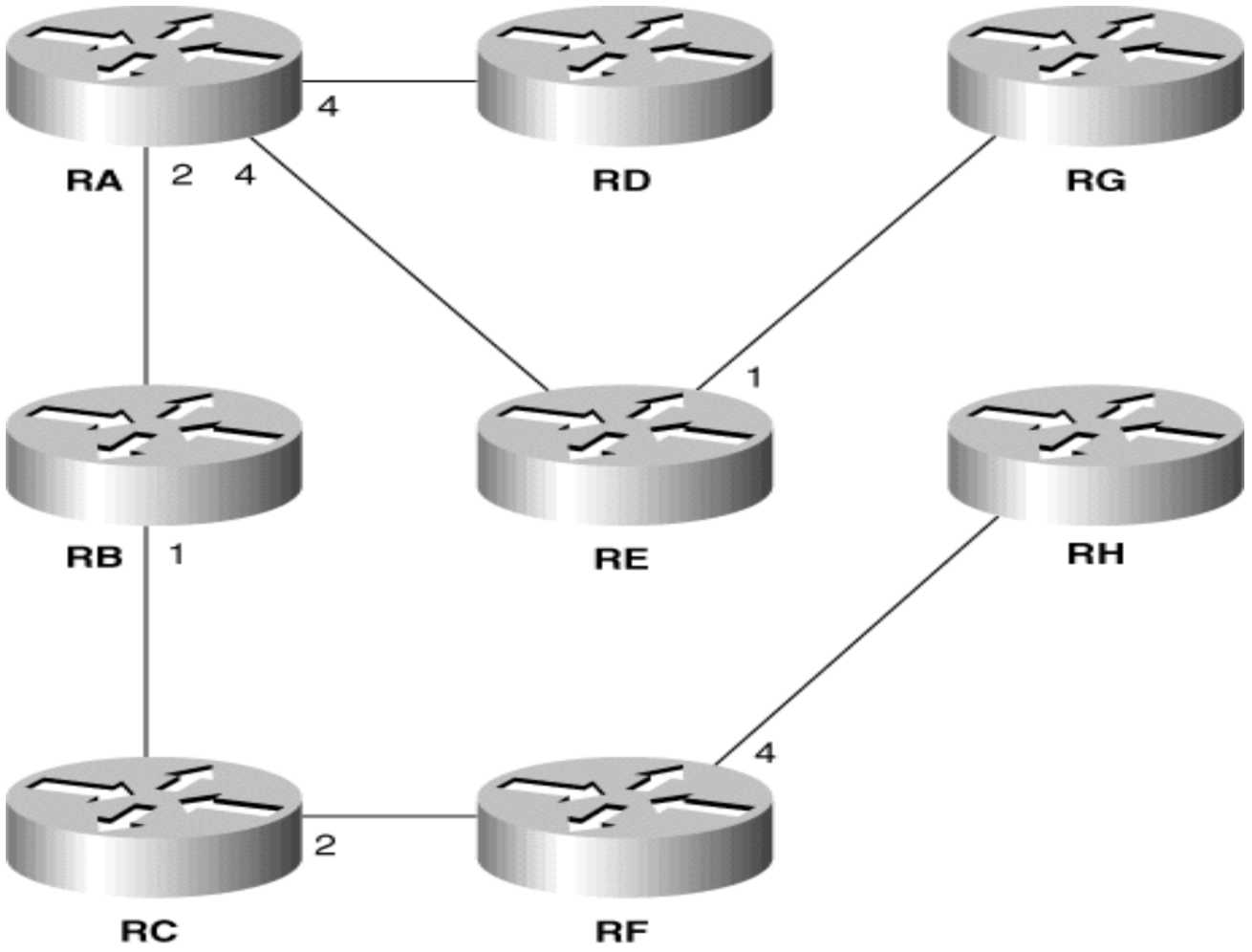
Канд. БД	Стойност до Root	SPF Tree	Описание
		RA,RA,0	Router A-**root**.
RA,RB,2 RA,RD,4 RA,RE,4	2 4 4	RA,RA,0	Добавят се съседите на RA
RA,RD,4 RA,RE,4 RB,RC,1 RB,RE,10	4 4 3	RA,RA,0 RA,RB,2	(RA,RB,2) с най-ниска стойност
RA,RD,4 RA,RE,4 RC,RF,2	4 4 5	RA,RA,0 RA,RB,2 RB,RC,1	(RB,RC,1) с най-ниска стойност от канд. БД

RA,RE,4 RC,RF,2 RD,RE,3 RD,RG,5	4 5 7 9	RA,RA,0 RA,RB,2 RB,RC,1 RA,RD,4	(RA,RD,4) и (RA,RE,4) са с 4 от RA; (RC,RF,2) = 5. (RA,RD,4) се добавя. (RD,RE,3) отпада.
RC,RF,2 RD,RG,5 RE,RF,2 RE,RG,1 RE,RH,8	5 9 6 5 12	RA,RA,0 RA,RB,2 RB,RC,1 RA,RD,4 RA,RE,4	(RA,RE,4) се добавя. Съседите на RE в Канд. БД. “Най-скъпият” път до RG отпада.
RE,RF,2 RE,RG,1 RE,RH,8 RF,RH,4	6 5 12 9	RA,RA,0 RA,RB,2 RB,RC,1 RA,RD,4 RA,RE,4 RC,RF,2	(RC,RF,2) се добавя, съседите му – в Канд. БД. “Най-скъпият” път до RH отпада.

Пресмятане на SPF

RF,RH,4	RA,RA,0 RA,RB,2 RB,RC,1 RA,RD,4 RA,RE,4 RC,RF,2 RE,RG,1	(RE,RG,1) се добавя. Всички съседи на RG са в дървото, нищо не се добавя в Канд. БД.
	RA,RA,0 RA,RB,2 RB,RC,1 RA,RD,4 RA,RE,4 RC,RF,2 RE,RG,1 RF,RH,4	(RF,RH,4) се добавя. Няма повече кандидати. **Shortest path tree е готово.**

Дървото на най-късия път



Изчисляване на новите маршрути. Голям обем памет.

Изчислените маршрути се записват в маршрутните таблици.

Необходимата памет за съхраняване на информацията за състоянието на връзките за мрежа с ***n*** маршрутизатори, всеки от които има по ***k*** съседи е пропорционална на ***nk***.

Така големите по размер мрежи изискват използване на маршрутизатори с голям обем памет.

Йерархична маршрутизация

С увеличаването на размерите на мрежата нараства обемът на маршрутните таблици, което изисква повече памет и процесорно време за тяхната обработка.

Това налага въвеждането на йерархично маршрутизиране, при което мрежата се разделя на **области**.

Маршрутизаторите в една област знаят всичко за вътрешната структура на своята област, но не знаят вътрешната структура на останалите области.

Йерархична маршрутизация е най-подходяща при протоколите със следене на състоянието на връзката. (DV не може, не са рекурсивни.)

Области в LS протоколите