14. 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

LS протоколите - shortest path first или distributed database, се базират на алгоритъма на E. W. Dijkstra - shortest path.

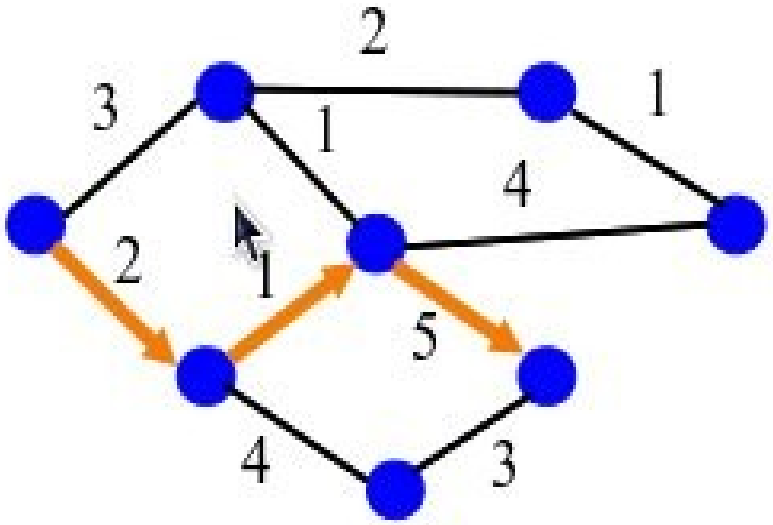
Такива протоколи са:

- Open Shortest Path First (OSPF) за IP;

- Intermediate System to Intermediate System (IS-IS) на ISO за CLNS и IP;

- Optimized Link State Routing Protocol (OLSR - rfc3626).

Алгоритъм на Dijkstra



CLNS

CLNS (Connectionless Network Service).

OSI Network Layer услуга (за разлика от CONS - Connection-Oriented NS) не изисква да се установи канал.

CLNS маршрутизира съобщения до дестинацията независимо от други – message switching.

В OSI CLNS е услуга, осигурена от CLNP (Connectionless Network Protocol) и използвана от TP4 (Transport Protocol Class 4).

CLNS

Но CLNP не се използва Internet, вместо него IP.

Все пак много телеком операторите още използват CLNP.

Защото IS-IS (OSI layer 3 protocol) е признат от ITU-T като протокол за управление на SDH (Synchronous Digital Hierarchy).

Около нас – академичната мрежа в Румъния.

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.
3. Конструиране на пакети с информация за състоянието на връзките.

Пет основни действия

4. Изпращане на тези пакети до всички останали маршрутизатори.
5. Изчисляване на най-късия път до всеки маршрутизатор в мрежата.

Резултат от тези действия

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

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 “ехо” 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 предаване (например Ethernet), откриването на съседите е малко по-сложно.

Един възможен начин за представяне на връзките между тях е да се въведе допълнителен възел, който да отговаря на общата среда за предаване.

Измерване на цените на връзките

Всеки маршрутизатор трябва да може да определи време-закъснението до своите съседи.

Най-простият начин е маршрутизаторът да изпрати "echo" пакет към всеки свой съсед на който трябва директно да се отговори.

Времето от изпращането на "echo" пакета до получаване на отговора се дели на две и по този начин се получава времето-закъснение до съответния съсед.

За по-точно измерване, този процес може да се повтори няколко пъти и да се вземе средната стойност.

Този метод предполага, че връзките са симетрични, което не винаги е вярно.

Измерване на цените на връзките

Друг въпрос е дали при измерването да се взима предвид натовареността на възлите.

Разликата се постига в зависимост от това кога маршрутизаторът стартира измерването:

когато пакетът постъпва в съответната изходяща опашка или когато пакетът се придвижи в началото на опашката.

Включването на натовареността на възлите има предимства и недостатъци.

Предимството е, че от две линии, които имат еднаква скорост за по-къса ще се счита по-ненатоварената линия. Това ще доведе до по-голяма ефективност.

Измерване на цените на връзките

Недостатъкът може да се илюстрира със следния пример.

Нека една мрежа е разделена на две части, които са свързани чрез две линии *A* и *B*.

Да предположим, че в даден момент по-голямата част от трафика между двете части на мрежата минава по линия *A*.

Тогава при следващото изчисляване на маршрутните таблици трафикът ще се насочи към по-добрата линия *B*.

Този процес ще се повтаря циклично и ще доведе до нестабилност в работата на мрежата.

link state packets

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

Пакетът трябва да съдържа уникалното име на подателя, 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

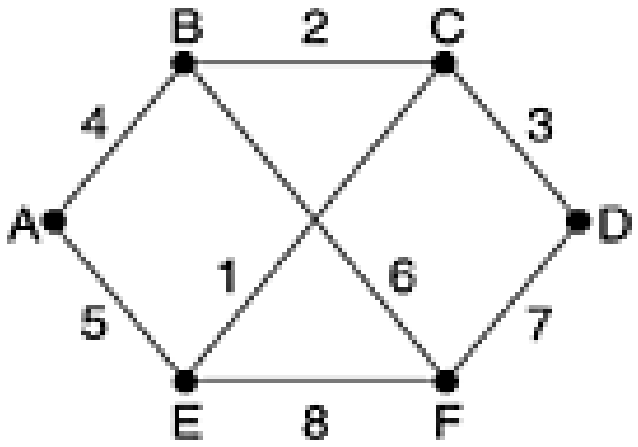
Един възможен начин е това да става през определени равни интервали от време.

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 време-закъснение.

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.

Пореден номер

Този алгоритъм има някои проблеми.

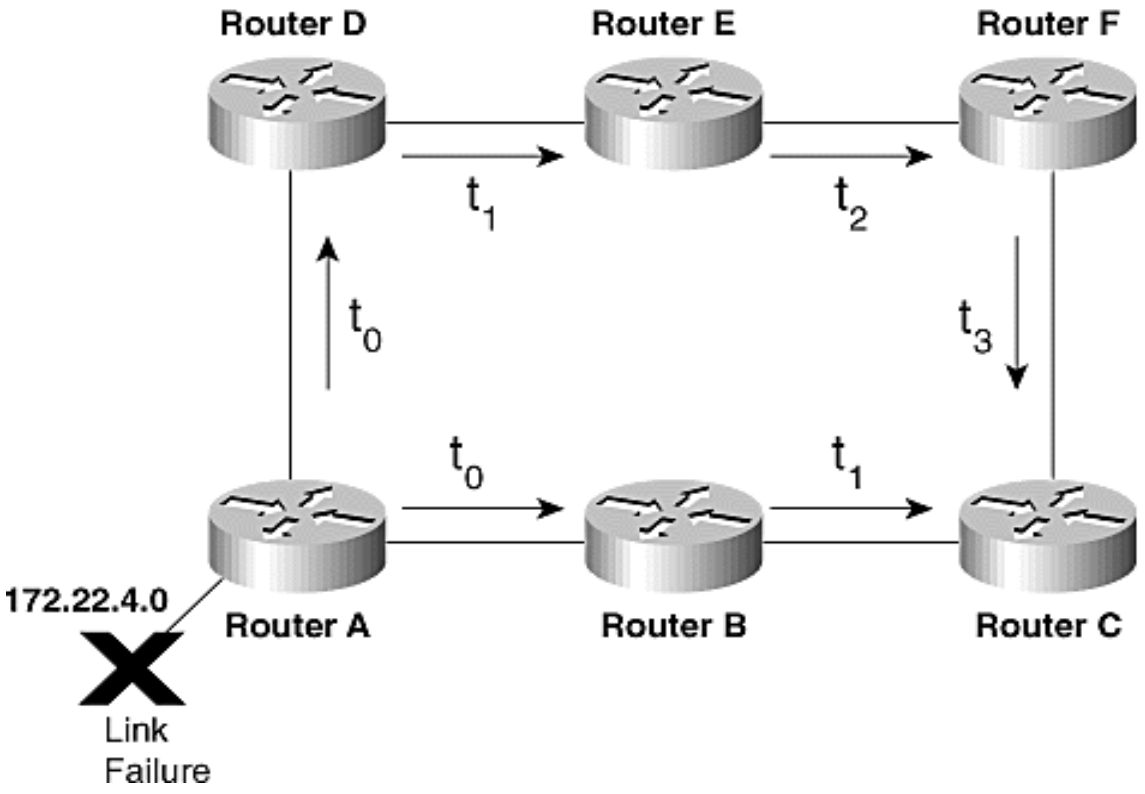
Ако поредният номер не е достатъчно голям, той може да се превърти.

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.

Таблица с информация за състоянието на връзките

Таблицата с информация за състоянието на връзките, която се използва от маршрутизатор *B* примера в слайд 17 изглежда примерно по следния начин:

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Таблица с информация за състоянието на връзките

Всеки ред от таблицата съответства на пристигнал, но все още необработен пакет.

Полето *Source* е източникът на пакета, *Seq* е поредният му номер, *Age* е срокът на годност.

С всеки пакет се свързват флагове за изпращане (*send flags*) и флагове за потвърждение (*ACK flags*) за всяка от изходните линии на маршрутизатора *B*.

Флаговете за изпращане указват по кои линии трябва да се изпрати пакета.

Флаговете за потвърждение указват по кои линии да се изпрати потвърждение за получаването на пакета.

Таблица с информация за състоянието на връзките

Ако в *B* пристигне дубликат на някой от пакетите в таблицата, то съответните флагове трябва да се актуализират.

Например, ако пристигне дубликат на пакета със състоянието на *C* от *F*, преди този пакет да бъде препратен към *F*, то флаговете за изпращане на пакета ще се променят на 100, а флаговете за потвърждение - на 011.

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).

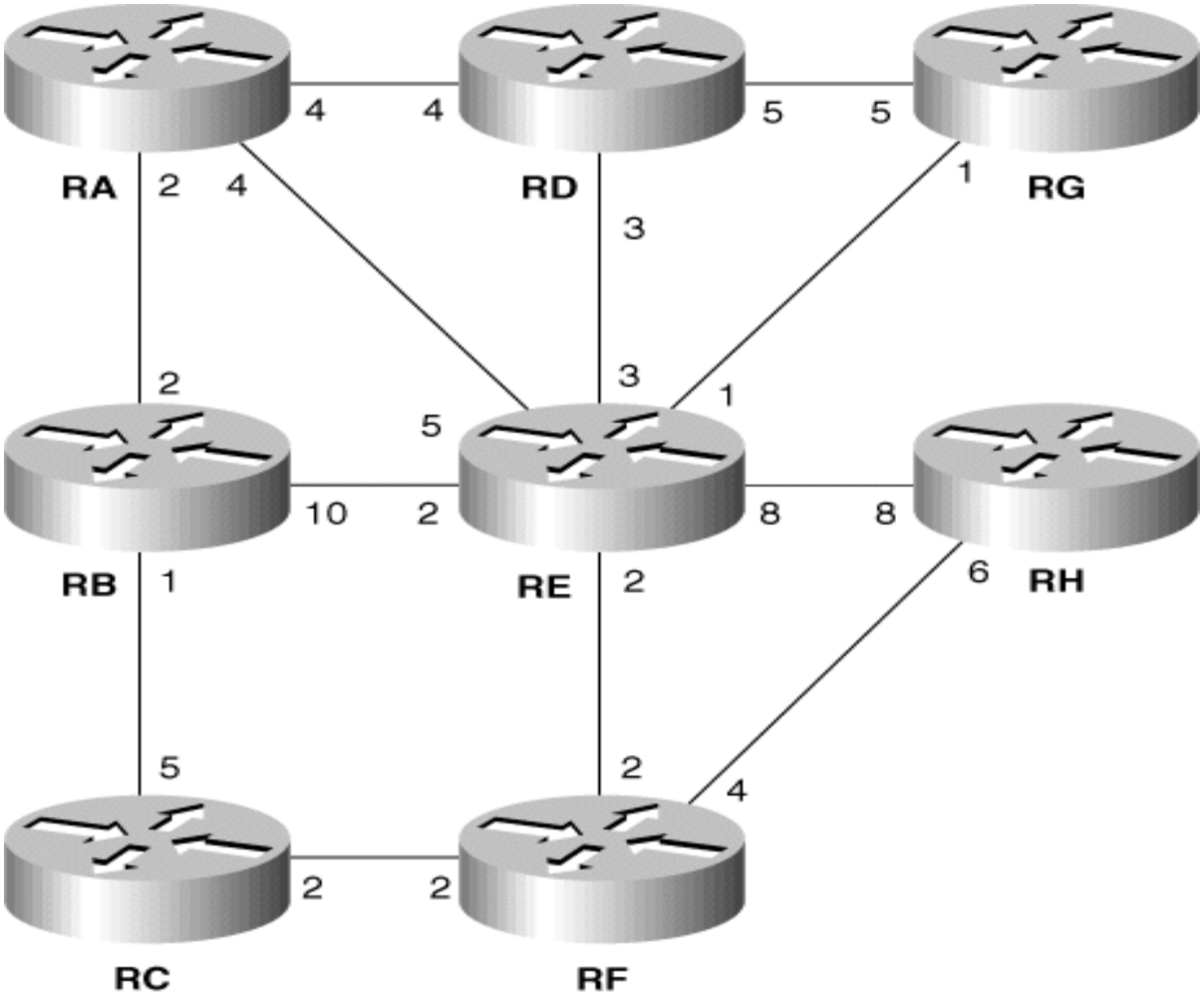
Прилагане на SPF алгоритъма

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

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

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

Таблица за състоянието на връзките. Link State Database.



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

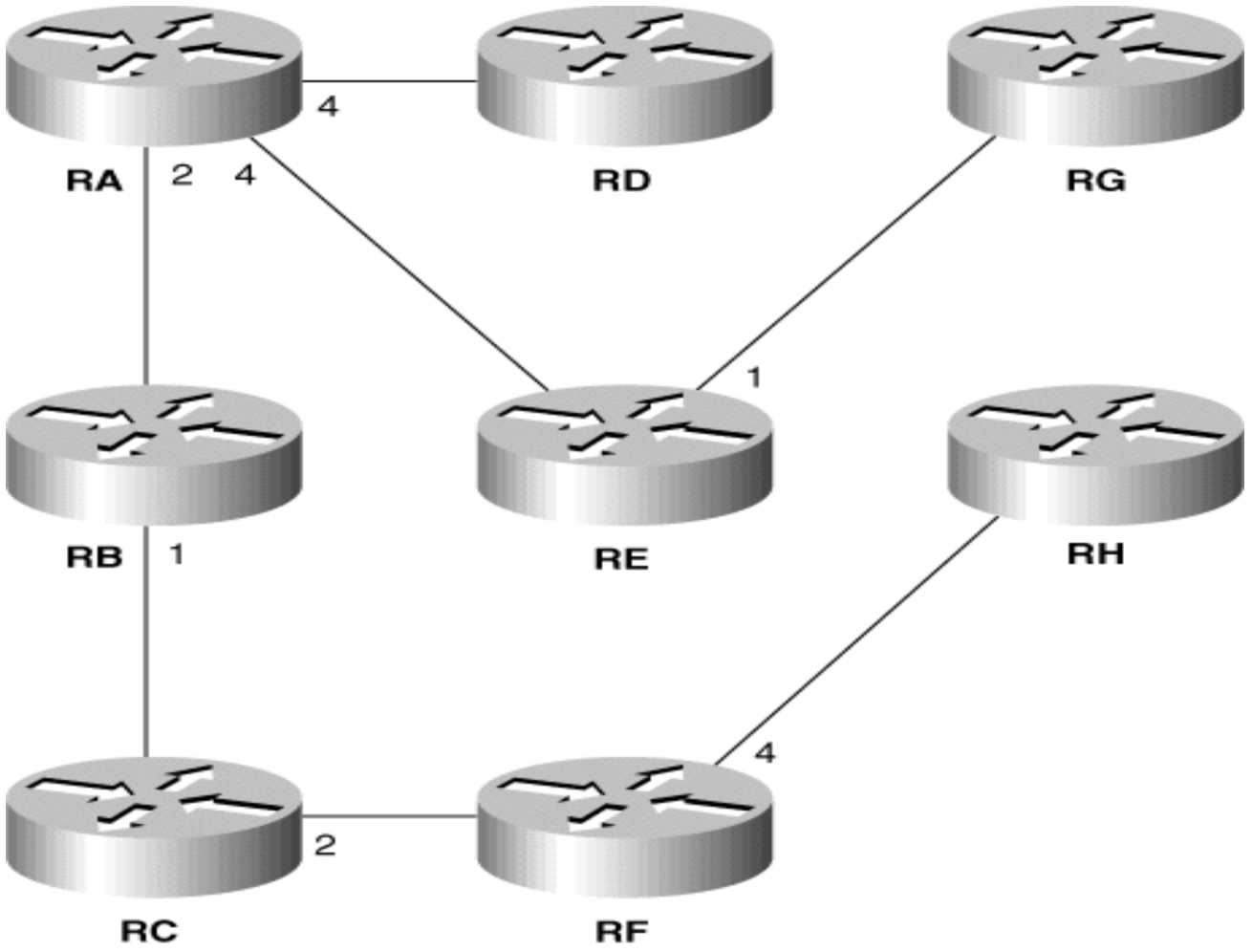
Канд. БД	Стойност до Root	SPF Tree	Описание
		RA,RA,0	Router A-**root**.
RA,RB,2 RA,RD,4 RA,RE,4	2 4 4	RA,RA,0	Добавят се съседите на RA
RA,RD,4 RA,RE,4 RB,RC,1 RB,RE,10	4 4 3	RA,RA,0 RA,RB,2	(RA,RB,2) с най-ниска стойност
RA,RD,4 RA,RE,4 RC,RF,2	4 4 5	RA,RA,0 RA,RB,2 RB,RC,1	(RB,RC,1) с най-ниска стойност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

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.

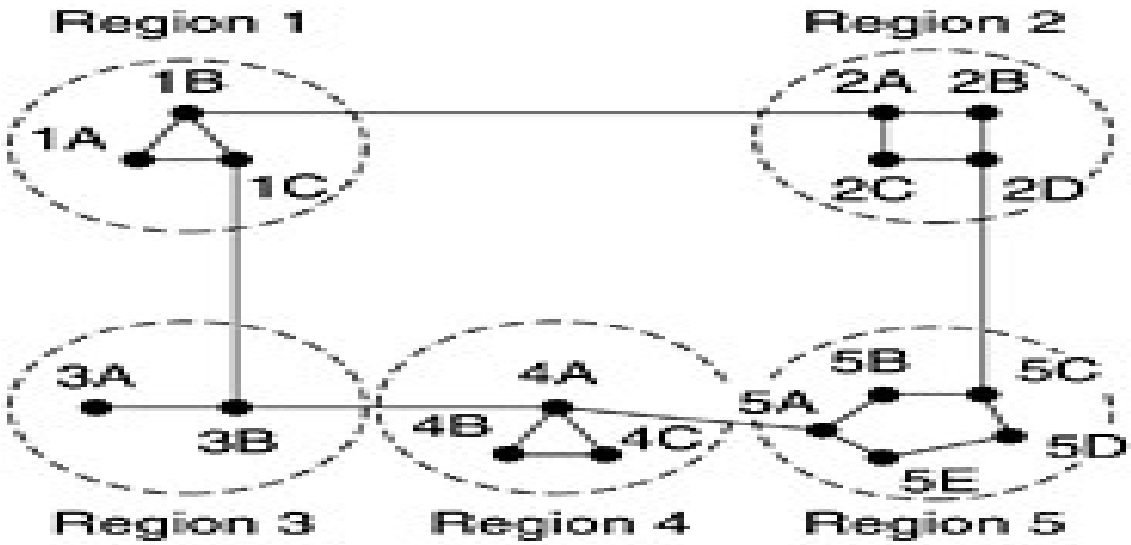
Това налага въвеждането на йерархично маршрутизиране, при което мрежата се разделя на **области**.

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

За по-големи мрежи може да е необходима йерархия с повече от две нива.

Пример на мрежа с йерархична маршрутизация на две нива

Като пример да разгледаме следната мрежа с йерархична маршрутизация на две нива. Метриката е в хопове.



Пример (прод.)

Пълната таблица за маршрутизатора *1A* съдържа 17 реда и има следния вид:

Dest.	Line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Пример (прод.)

Съкратената таблица за маршрутизатора *1A* съдържа 7 реда и има следния вид:

Dest.	Line	Hops
1A	—	—
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Пример (прод.)

В нея са запазени маршрутите към направленията в "област 1".

Маршрутите към направленията в останалите области са обобщени в един ред, като се използва маршрутизатора от съответната област, който е най-близо до маршрутизатора *1A*.

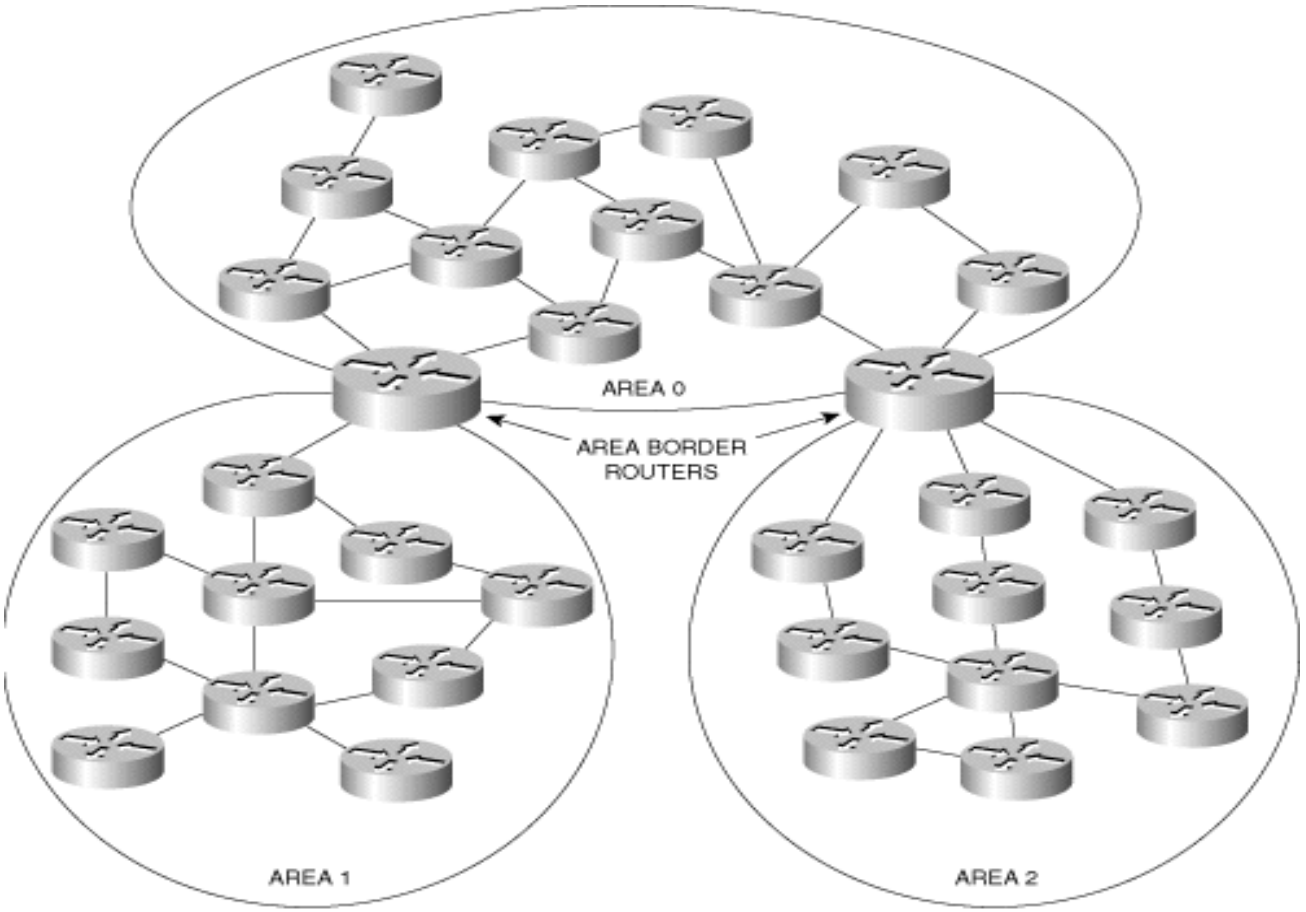
Спестяването на памет от маршрутните таблици има отрицателен ефект - някои от пътищата увеличават своята дължина.

Пример (прод.)

Например, най-късият път от *1A* до *5C* минава през "област 2", но при йерархично маршрутизиране ще се използва пътят през "област 3", тъй като това е по-добре за повечето маршрутизатори от "област 5".

Ако *n* е броят на маршрутизаторите в една мрежа, може да се покаже, че оптималният брой области, всяка с по равен брой маршрутизатори, е най-близкото цяло до $\sqrt{n}$

Области в LS протоколите



Области в LS протоколите

Област (***area***) е подмножество от рутери в общата мрежа. (DV не може, не са рекурсивни.)

Разделянето на области се налага:

- колкото е **по-голяма БД** със състояние на връзките, изисква **повече памет** от DV.

- сложният алгоритъм - повече CPU време и цикли от DV.

- "*flooding*" на LS пакети vs. *bandwidth*, но OSPF и др. - частични updates, когато е необходимо.

LS vs. DV

Размяна на съобщения

LS: с **n** възела, **E** връзки, **O(nE)** изпратени съобщения

DV: разменят се само **м/у съседи**

Скорост на сходимост

LS: висока

DV: ниска

Зацикляния

Проблем “Count-to-infinity”

Стабилност: рутер не е в ред?

LS:

Възелът рекламира неточна “*link cost*”

Всеки възел изчислява **собствената си таблица**

DV:

DV възел рекламира **неточен “*path cost*”**

Таблицата на даден възел се използва от други (**грешката се разпространява**)