13. Разпределена маршрутизация с дистантен вектор

Малко история

Маршрутизиращ протокол с дистантен вектор (distance vector protocol) е използван отначало в ARPANET.

По-късно в Интернет намира широко приложение като RIP (Routing Information Protocol).

Основите на тези алгоритми са поставени от Белман (1957 г.), Форд и Фолкерсън (1962 г.).

Затова са известни като алгоритми Белман-Форд или Форд-Фолкерсън.

Само на Cisco Systems – IGRP и EIGRP.

Основни принципи

Distance Vector – рутерите се анонсират (рекламират) като вектори:

Посока - адреса на следвашия възел (next hop) и изходящия интерфейс и

Разстояние (метрика), напр. брой възли до дестинацията (hop count).

Основни принципи

Маршрутизаторите (рутери) в тези случаи не знаят целия път до крайната точка (дестинация).

DV използва само:

1. Посока или интерфейс, по който да се отправи пакета.
2. Разстоянието до дестинацията.

Общи положения

При маршрутизацията с дистантен вектор (distance vector routing) всеки маршрутизатор изгражда и поддържа маршрутна таблица, в която всеки ред съдържа адрес на местоназначение, адрес на следващата стъпка към това местоназначение по най-добрия известен до момента път и дължината на този път (метрика).

Периодически маршрутизаторите изпращат на съседите си цялата или част от маршрутната таблица.

Предимства и недостатъци

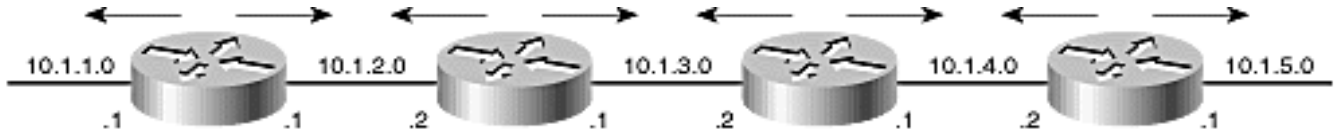
DV алгоритмите:

- не товарят процесора и паметта;
- лесни са за реализация и поддръжка;

Но

- Периодическите update-и отнемат пропускателна способност от потребителите.

Distance Vector в действие



Router A

	NET	VIA	HOPS
t_0	10.1.1.0	--	0
	10.1.2.0	--	0
t_1	10.1.1.0	--	0
	10.1.2.0	--	0
	10.1.3.0	10.1.2.2	1
t_2	10.1.1.0	--	0
	10.1.2.0	--	0
	10.1.3.0	10.1.2.2	1
	10.1.4.0	10.1.2.2	2
t_3	10.1.1.0	--	0
	10.1.2.0	--	0
	10.1.3.0	10.1.2.2	1
	10.1.4.0	10.1.2.2	2
	10.1.5.0	10.1.2.2	3

Router B

NET	VIA	HOPS
10.1.2.0	--	0
10.1.3.0	--	0
10.1.2.0	--	0
10.1.3.0	--	0
10.1.1.0	10.1.2.1	1
10.1.4.0	10.1.3.2	1
10.1.2.0	--	0
10.1.3.0	--	0
10.1.1.0	10.1.2.1	1
10.1.4.0	10.1.3.2	1
10.1.5.0	10.1.3.2	2
10.1.2.0	--	0
10.1.3.0	--	0
10.1.1.0	10.1.2.1	1
10.1.4.0	10.1.3.2	1
10.1.5.0	10.1.3.2	2

Router C

NET	VIA	HOPS
10.1.3.0	--	0
10.1.4.0	--	0
10.1.3.0	--	0
10.1.4.0	--	0
10.1.2.0	10.1.3.1	1
10.1.5.0	10.1.4.2	1
10.1.3.0	--	0
10.1.4.0	--	0
10.1.2.0	10.1.3.1	1
10.1.5.0	10.1.4.2	1
10.1.1.0	10.1.3.1	2
10.1.3.0	--	0
10.1.4.0	--	0
10.1.2.0	10.1.3.1	1
10.1.5.0	10.1.4.2	1
10.1.1.0	10.1.3.1	2

Router D

NET	VIA	HOPS
10.1.4.0	--	0
10.1.5.0	--	0
10.1.4.0	--	0
10.1.5.0	--	0
10.1.3.0	10.1.4.1	1
10.1.4.0	--	0
10.1.5.0	--	0
10.1.3.0	10.1.4.1	1
10.1.2.0	10.1.4.1	2
10.1.4.0	--	0
10.1.5.0	--	0
10.1.3.0	10.1.4.1	1
10.1.2.0	10.1.4.1	2
10.1.1.0	10.1.4.1	3

Метрика

Предполага се, че всеки маршрутизатор знае метриката на връзките до своите съседи.

Ако метриката е брой стъпки или маршрутизатори до местоназначението (хопове), разстоянието до всеки съсед е 1.

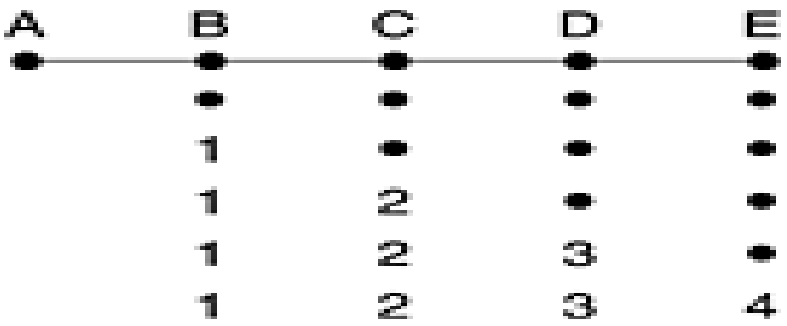
Ако метриката е натоварване на възела, разстоянието до всеки съсед е броя на пакетите в изходящата опашка към този пакет.

Ако метриката е времезакъснение, маршрутизаторът периодично изпраща “ехо” пакети до съседните му маршрутизатори и измерва закъснението на техния отговор.

Недостатък

Сериозен недостатък на маршрутизиращите алгоритми с дистантен вектор е **ниската им скорост на сходимост**.

Добрите новини се разпространяват бързо в мрежата, но лошите новини обикновено изискват твърде голям брой периодични съобщения за да достигнат до всички маршрутизатори.



Добавяне на обект

Нека маршрутизаторът *A* в началото не е включен в мрежата.

Всички останали маршрутизатори знаят това - в маршрутната им таблица към направлението *A* е записано ∞ (достатъчно голямо число, трябва да е поне с единица повече от диаметъра на мрежата). Това е отразено на първия ред по-горе.

След включването на *A* останалите маршрутизатори научават за това събитие чрез няколко обмена на своите вектори на разстоянията, всеки от които се извършва едновременно между всички съседни маршрутизатори.

Добавяне на обект

При първата обмяна *B* научава от *A* за път с дължина 0 до *A* и записва в своята таблица, че *A* е на разстояние 1.

В този момент останалите маршрутизатори все още не са научили за включването на *A*. Това е отразено на втория ред по-горе.

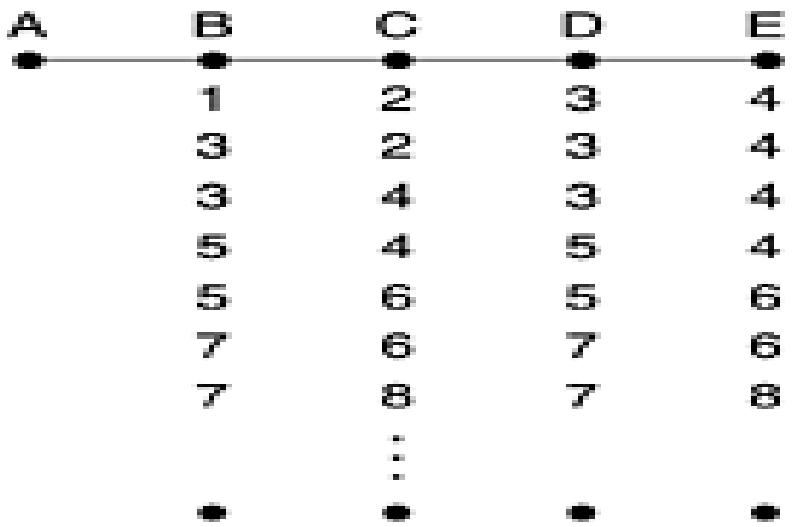
При следващия обмен *C* научава, че от *B* съществува път до *A* с дължина 1 и записва в своя вектор път до *A* през *B* на разстояние 2 и т.н.

Добавяне на обект

По-общо в мрежа с диаметър *k* хопа са необходими най-много *k* размени на съобщения за разпространяване на новината за появил се по-добър път.

Изключване на обект

Да разгледаме друг пример.



Изключване на обект

Нека всички маршрутизатори в началото са включени в мрежата.

Да предположим, че *A* спира да работи или се прекъсва връзката от *A* до *B*, което от гледна точка на *B* е същото.

При първия обмен *B* не получава информация от *A*, но получава информация от *C*, че има път до *A* с дължина 2.

B не знае, че пътя от *C* до *A* минава през него - от негова гледна точка би могъл да съществува друг независим път от *C* до *A*, затова *B* записва в таблицата си в реда за *A* път с дължина 3 и следваща стъпка *C*.

Изключване на обект

D и *E* не променят маршрутните си таблици при първия обмен на векторите на разстоянията.

На следващия обмен *C* научава за два възможни пътя до *A*, и двата с дължина 4, единият през *B*, другият през *D.*

C избира и записва в маршрутната си таблица единия от тях в зависимост от реда на обработването на съобщенията от *B* и *D*.

Count to Infinity

Резултатът от продължаващия обмен е отразен в следващите редове по-горе. Той ще продължи, докато стойностите по направленията към *A* и в четирите машрутизатора не достигнат ∞.

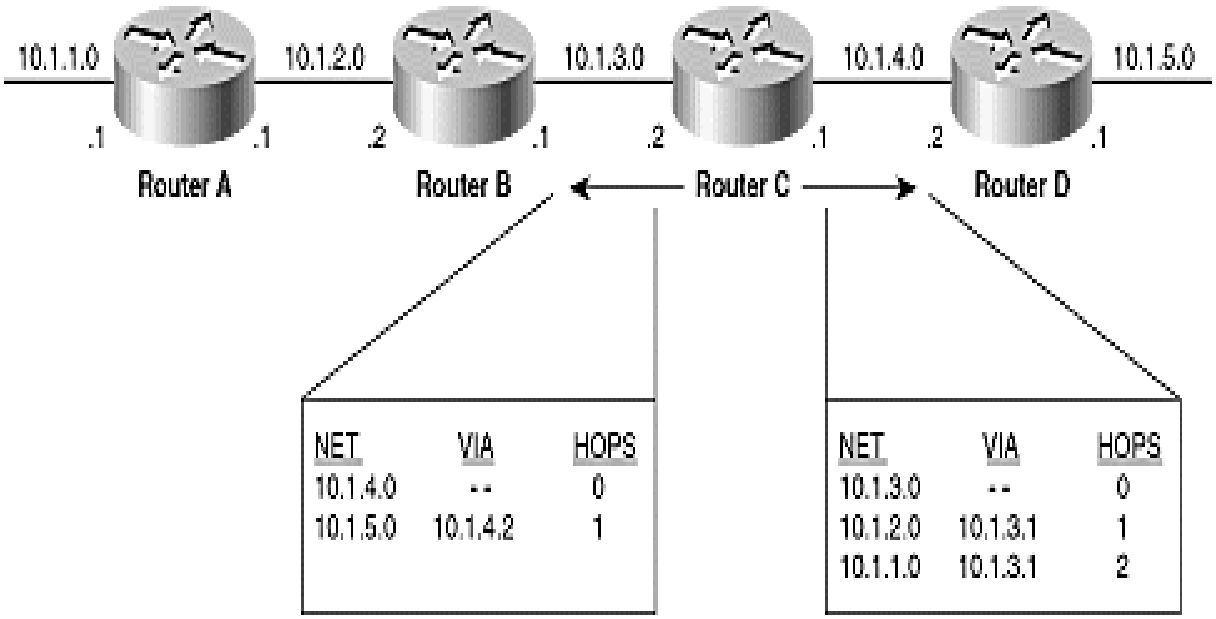
Този проблем се нарича **броене до безкрайност** (**count to infinity**).

Split horizon

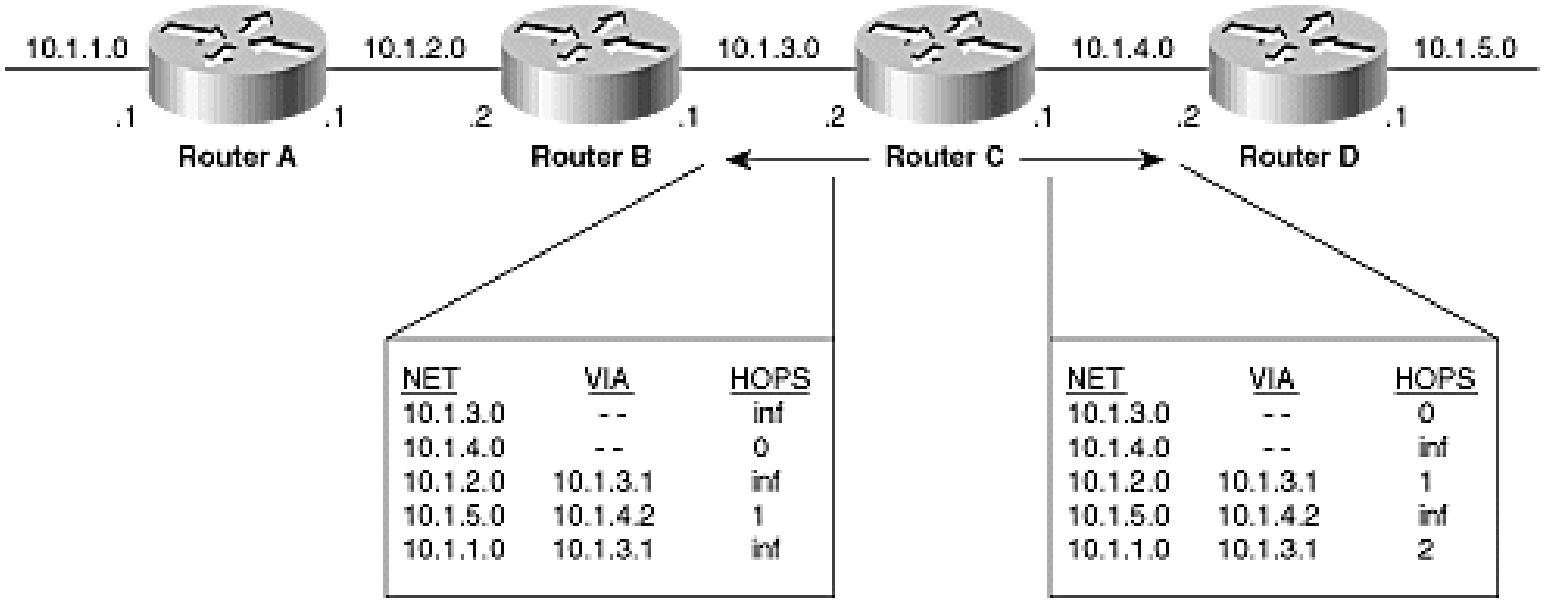
Едно частично негово решение е т.н. **разделяне на хоризонта** (**split horizon**).

При него се въвежда ново правило - ако в маршрутната таблица на *X* в реда за *Y* е записана следваща стъпка *Z*, то *X* не изпраща към *Z* информация за маршрута към *Y*.

Split horizon



Split horizon wi poisoned reverse



"По-добре лоша новина отколкото никаква". Рутер B получава фалшива информация, че 10.1.1.0 е достижима през C.

Обикновен Split horizon няма да се справи.

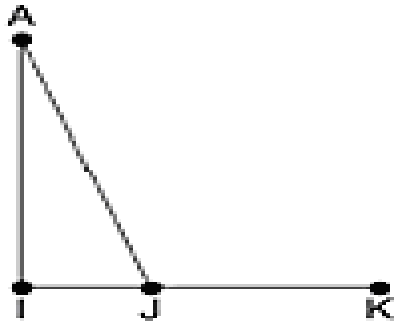
Частично решение - **split horizon**

В горния пример на втория обмен на вектори, *C* не изпраща към *B* информация за маршрута към *A*, тъй като маршрутът от *C* към *A* минава през *B*.

Въвеждането на разделяне на хоризонта не решава напълно проблема броене до безкрайност.

Да разгледаме следния пример.

split horizon



В началото *A* и *I* имат пътища с дължина 2 стъпки до *K* през *J*, а *J* има път с дължина 1 до *K*.

Да предположим, че връзката между *J* и *K* отпадне.

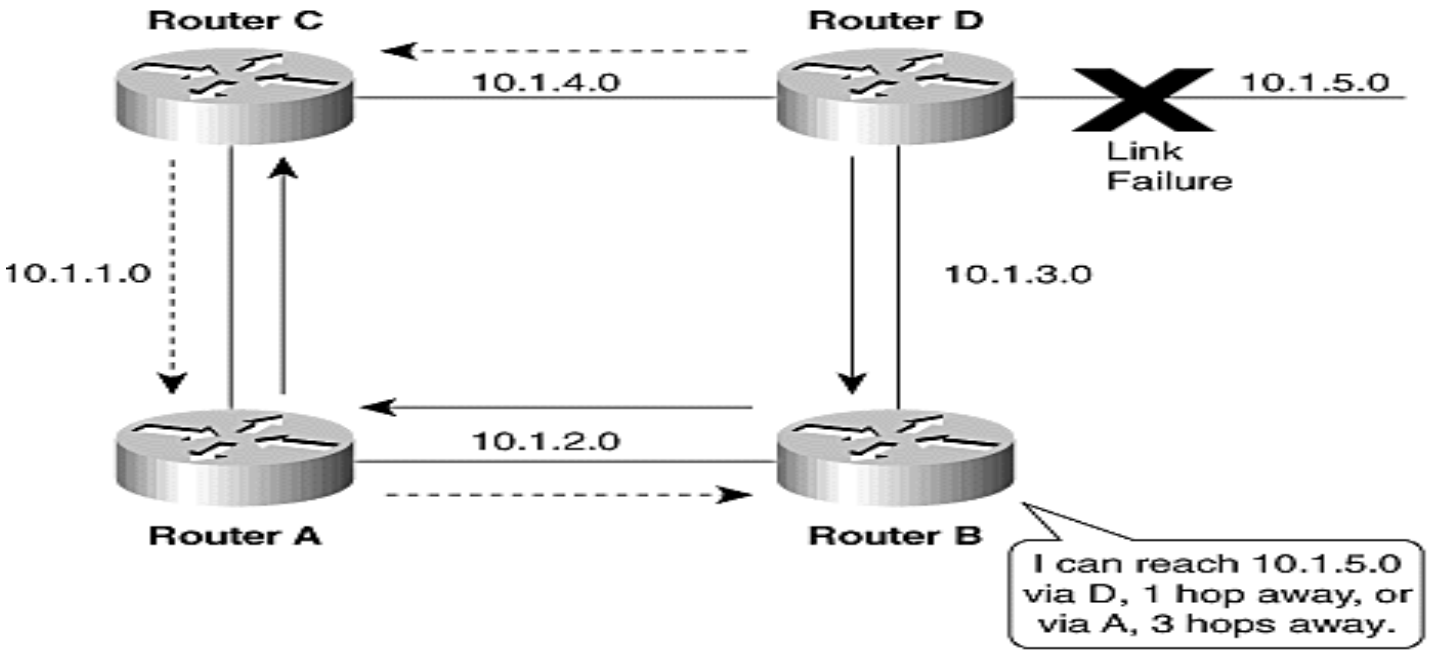
split horizon

Тогава на първия последвал обмен *J* няма да получи информация за нов път до *K* през *A* или *I* по правилото за разделяне на хоризонта и правилно ще заключи, че *K* е недостижим.

На следващия обмен *A* и *I* научават, че няма път до *K* през *J*, но *A* научава за път до *K* през *I* с дължина 3 и *I* научава за път до *K* през *A* с дължина 3.

Така въпреки разделянето на хоризонта, *A* и *I* ще броят до безкрайност.

Броене до безкрайност



Тук Split horizon не помага.

Едно решение: ограничаване hop count (=15), но конвергенция около 7 минути при update = 30 s.

Недостатъчно, затова:

Triggered Updates

Незабавното изпращане на извънредни съобщения за обновяване (Triggered Updates) намалява вероятността за поява на цикли.

Когато един маршрутизатор промени метриката за даден маршрут, той трябва веднага да изпрати извънредно съобщение.

Например, връзката J-K отпада.

Triggered Updates

J ще съобщи веднага на A и I, че:

$$d(J\text{-}K) = \infty$$

A и I записват в маршрутните си таблици, че K е недостижим и веднага съобщават това на своите съседи.

При следващо редовно (периодично) обновяване няма да възникне “броене до безкрайност” между A и I. В таблиците им няма да има остаряла информация за K.

Hold down

Triggered Updates не достига едновременно до всички маршрутизатори.

Маршрутизатор, още неполучил съобщението, би изпратил редовно периодично съобщение за обновяване с неточна информация до друг маршрутизатор, който вече има актуалната информация за променен маршрут.

Hold down

Защита срещу това: hold down таймер.

След изпращане или получаване на triggered update маршрутизаторът стартира hold down таймер.

До неговото нулиране не приемат съобщения за обновяване на променения маршрут.