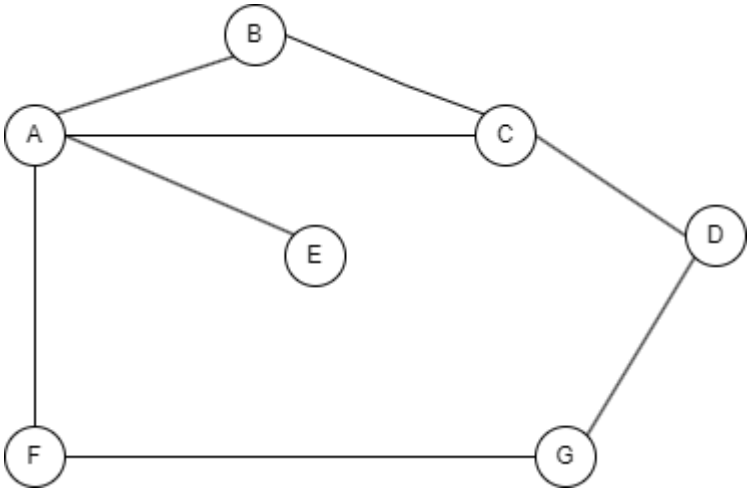
15. Vector-Distanta

Ideea in spatele acestui algoritm sta in numele sau vector-distanta. Fiecare nod constuieste un vector in care stocheaza distantele cate toate nodurile si distribuie acest vector la nodurile invecinate.Presupunerea pe care o face routarea vector-distanta este ca fiecare nod stie costul legaturii pentru fiecare vecin al sau. Aceste costuri pot fi configurate de un manager de retea. O legatura care nu este activa se noteaza cu infinit.



	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Pentru a intelege mai uosr cum functioneaza un astfel de algoritm vom folosi figura si tabelul de mai sus unde costul fiecarei legaturi este marcata cu 1 asfel incat calea cu costul cel mai mic sa fie una cu cele mai putine hopuri.

De notat ca fiecare nod stie doar linia pe care este situat. Consideram fiecare rand ca o lista de distante de la un nod la celelalte, reprezentant viziunea initiala a nodului. Astfel, A crede initial ca poate ajunge la B, dar nu la D. Tabela de rutare stocata de A va fi folosita pentrua a ajunge la orice nod la care e legat direct. Astfel tabela de rutare initiala a lui A va arata astfel:

Destinatie	Cost	Umatorul Hop

B	1	B
C	1	C
D	∞	--
E	1	E
F	1	F
G	∞	--

In urmatorul pas al algoritmului fiecare nod trimite, celor direct conectate, lista lui de rutare. De exemplu, nodul F ii spune nodului A ca poate ajunge la nodul G cu un cost de 1; A de asemenea stie ca poate ajunge la F cu un cost de 1, asa ca adauga costul pentru a ajunge la G, inca 1 ajungand la un cost de 2, care este mai mic decat costul intial de infint. Astfel, A inregistreaza ca poate ajunge la G cu un cost de 2 prin F. Asemanator A invata ca de la C ca poate ajunge la D adaugand la constul initial de 1 inca 1 ajungand la 2 si asa mai departe.

In acest punct, A isi face update la tabela de rutare cu costuri si urmatoarele hopuri la toate nodurile din retea astfel:

Destinatie	Cost	Umatorul Hop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

In absenta schimbarii topologiei nu este nevoie decat de cateva mesaje schimbate intre noduri pentru ca fiecare nod sa aiba o tabela completa de rutare. Avand astfel urmatorii vectori de distanta:

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Exista doua circumstante in care un nod trimite infromatii despre rutele sale, prima fiind un update periodic. In acest caz, fiecare nod trimite un update chiar daca nu exista o modificare, spunand celorlalte noduri ca este activ chiar daca nu sunt schimbari. Al doilea motiv pentru care trimite un update este cand un link devine inactiv si anunta celalate noduri cauzand o modificare in rutele din tabela. Cand este declansat un update cauzat de o moficiare poate determina ca fiecare nod sa isi recaluleze si sa transimta mai departe noile informatii la celalate noduri cu care comunica direct.

Pentru a intelege ce se intampla cand un link cade, sa consideram ce se intampla cu F cand detecteaza ca nu mai are legatura cu G. Atunci F seteaza o noua distanta catre G ca infinit si paseaza informatia catre A, din moment ce A stia ca este un drum de 2 hopuri pana la G prin F, A va seta si el distanta catre G ca infinit. Dar la umatorul update de la C, A va invata ca este o distanta de 2 hopuri de la C pana la G . Asfel A va avea o noua ruta de 3 hopuri prin C pana la G.

Exista cazuri in care anumite evenimente pot destabiliza reteaua. Sa presupunem ca legatura intre A si E cade, in urmatorul update. A spune ca distanta catre E este infinit dar B si C zice ca exista o distnata de 2 noduri catre E. In functie de momentul in care sunt trimise mesajele se poate ajunge la urmatorul eveniment: Nodul B afland ca poate avea un drum la E prin C cu 2 hopuri trage concluzia ca poate ajunge cu un total de 3 hopuri la E si trimite informatia catre A; nodul A concluzionand ca poate ajunge la E printr-un total de 4 hopuri trimite informatia catre C; nodul C crede ca poate ajunge prin 5 hopuri si tot asa. Acest ciclu merge pana cand se ajunge la un numar destul de mare cat sa se considere infinit. In tot acest timp nici unul din noduri nu stie ca E nu este accesibil si tabela de rutare nu se stabilizeaza. Aceasta problema se numeste numaratoarea pana la infinit.

Sunt cateva soluti partiale la aceasta problema. Prima este alegerea unui numar mic ca aproximare a infinitului. De exemplu putem decide ca numarul maxim de hopuri sa nu depaseasca 16 in retea asfel algand 16 ca aproximare a infinitului. Desigur, poate prezenta o problema in cazul in care se pot forma distante mai mari de 16 hopuri.

O tehnica de a imbunatatii timpul in care se stabilizeaza reteaua este numita "split horizon". Ideea fiind, ca atunci cand un nod trimite informatiile despre rute, nu trimte informatii despre rutele pe care le-a invatat de la un nod catre acelasi nod. De exemplu daca B are ruta (E,2,A) in tabel, atunci stie ca a invatat ruta de la A, atunci cand B trimite update-ul de rute catre A, nu include ruta (E,2) in acel update. O varianta mai avansata a lui split horizon numit "split horizon with poison reverse" in care B trimite inapoit ruta catre A, dar pune o informatie negativa in ruta pentru a se asigura ca A nu va folosi pana la urma B pentru a ruta informatii catre E.

Din nefericire ambele metode nu ajuta la viteza de stabilizare a protocolului, si acesta fiind un dezavantaj major fata de alte protocoale cum ar fi link-state routing.