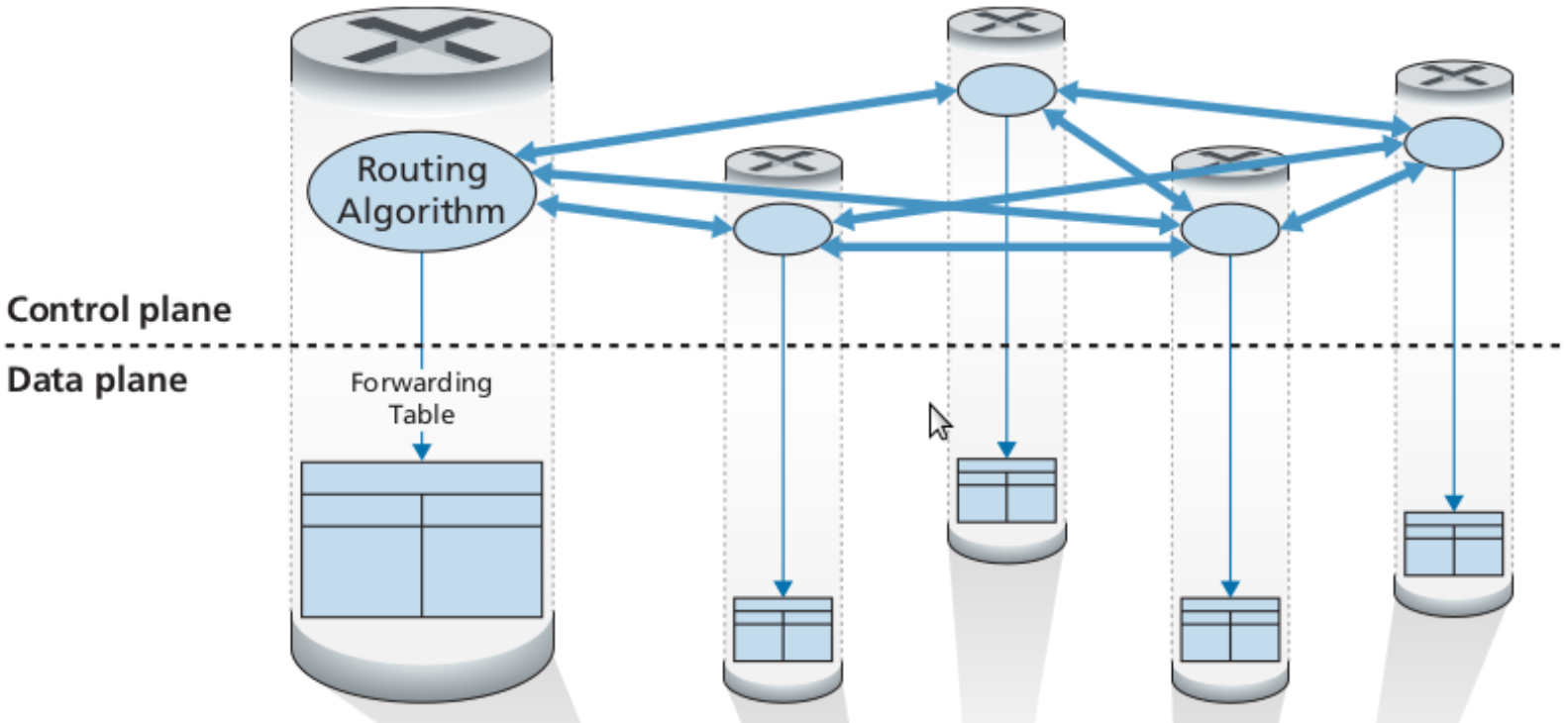
Capa de Red

October 20, 2025

Plano de Control

Per-router control

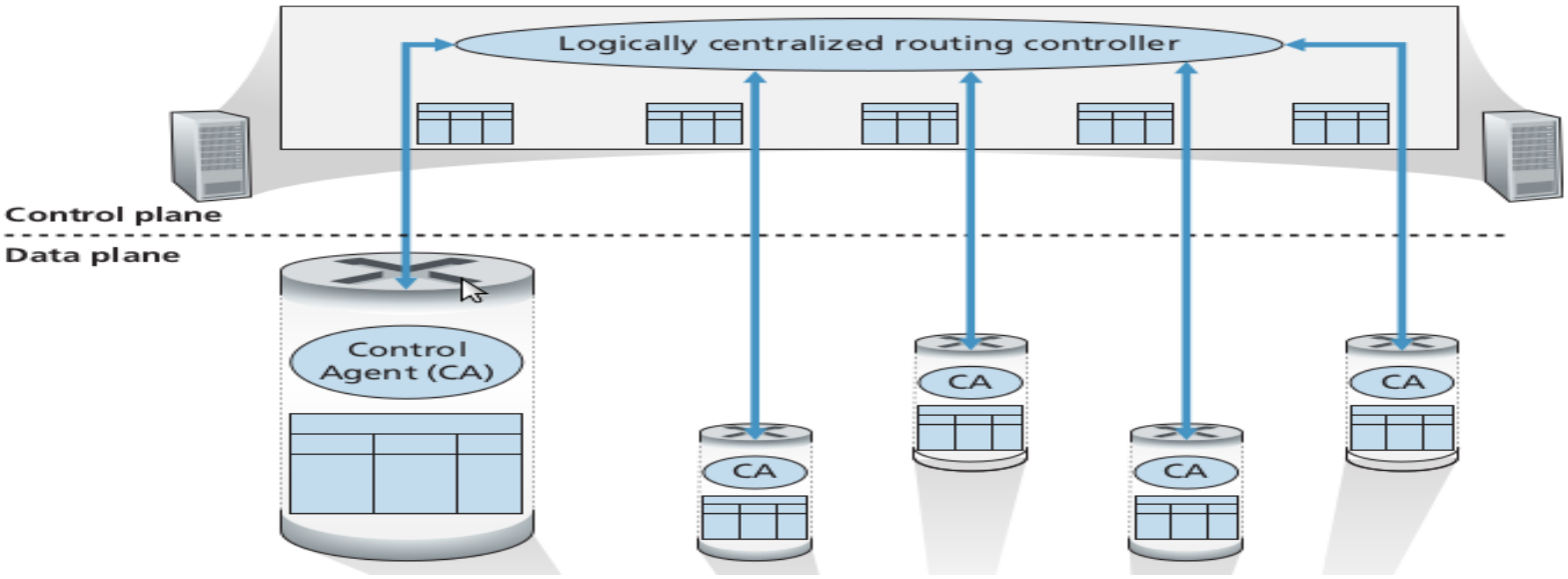
Cada ruter tiene un componente de enrutamiento que se comunica con los componentes de enrutamiento en otros ruters para calcular los valores de su tabla de reenvío.



Plano de Control

Logically centralized control

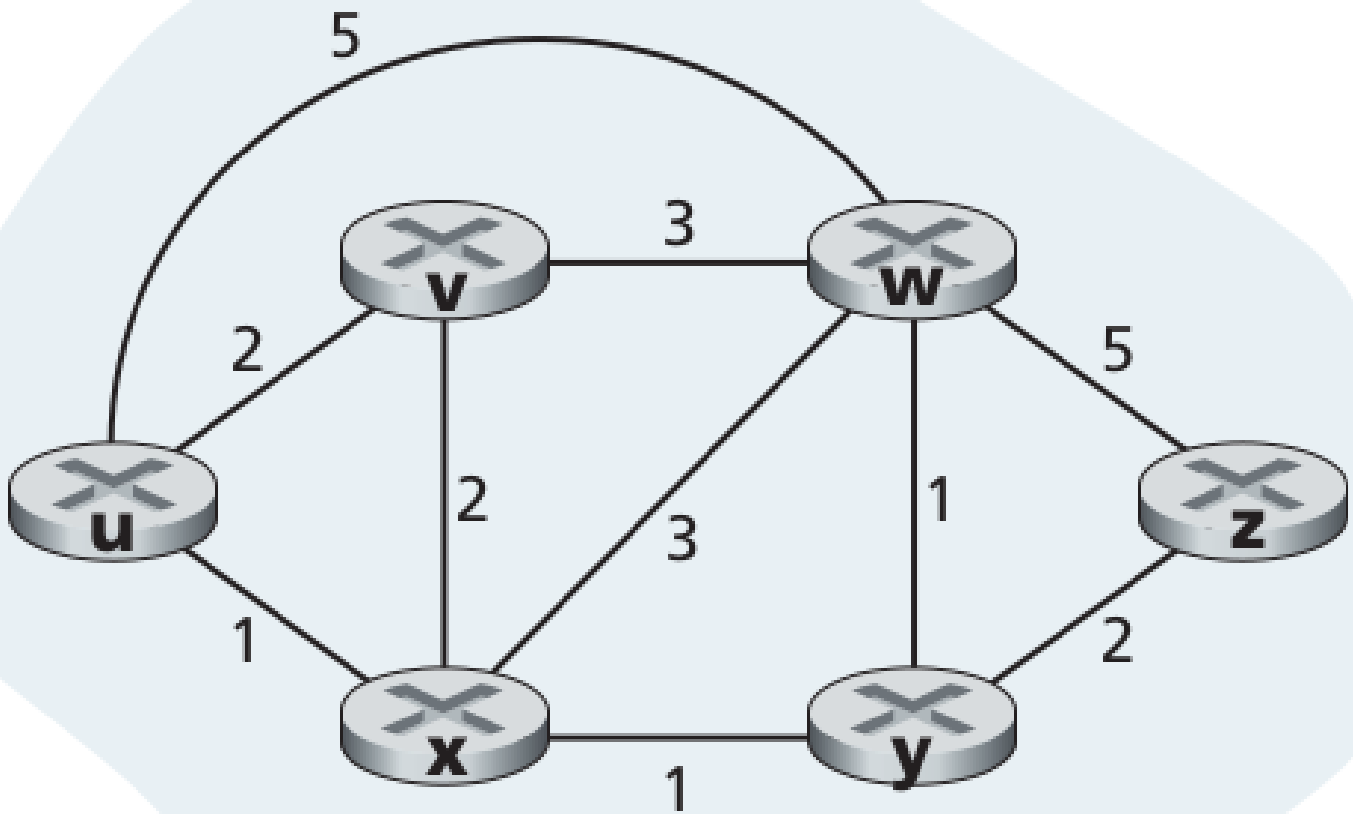
Un controlador lógicamente centralizado calcula y distribuye las tablas de reenvío que utilizará cada uno de los ruters.



El controlador interactúa con un agente de control (CA) en cada uno de los routers para configurar y administrar la tabla de flujo de ese router.

Algoritmos de Ruteo

Abstracción de una red con un grafo G = (N, E)



Algoritmos de Ruteo

Objetivo

Determinar un buen camino o ruta desde el router origen hasta el router destino, a través de una red de routers.

Algoritmos de Ruteo

- Algoritmos Centralizados: Calcula la ruta de costo mínimo entre un origen y un destino utilizando el conocimiento global y completo acerca de la red. Algoritmos de estado de enlaces (LS, Link-State).
- Algoritmos Descentralizados: Calcula la ruta de costo mínimo entre un origen y un destino de manera distribuida. Ningún nodo tiene toda la información acerca del costo de todos los enlaces de la red. Algoritmo de vector de distancias (DV, Distance-Vector).

The Link-State (LS) Routing Algorithm

Cada nodo difunde (hace broadcast) de su información al resto de los nodos. El resultado de esto, es que todos los nodos tienen una visión **completa e idéntica** de la red. Cada nodo puede entonces ejecutar el algoritmo LS y calcular el mismo conjunto de rutas de costo mínimo.

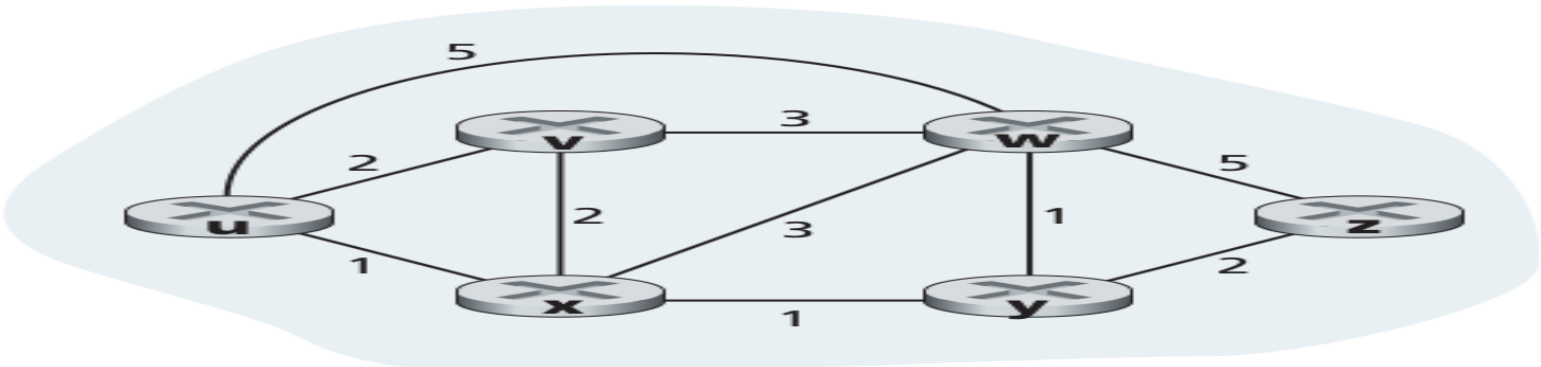
The Link-State (LS) Routing Algorithm

The link-state routing algorithm we present below is known as Dijkstra's algorithm, named after its inventor.

- D(v): costo de la ruta de costo mínimo desde el nodo de origen al destino **v**, para cada iteración del algoritmo.
- p(v): nodo anterior (vecino de **v**) a lo largo de la ruta de costo mínimo desde el origen hasta **v**.
- N': subconjunto de nodos; **v** pertenece a N' si la ruta de costo mínimo desde el origen hasta **v** se conoce de forma definitiva.

Link-State (LS) Algorithm for Source Node u

```
1   Initialization:
2    N' = {u}
3    for all nodes v
4      if v is a neighbor of u
5        then D(v) = c(u,v)
6      else D(v) = ∞
7
8   Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for each neighbor v of w and not in N':
12           D(v) = min(D(v), D(w)+ c(w,v) )
13    /* new cost to v is either old cost to v or known
14     least path cost to w plus cost from w to v */
15  until N'= N
```



Link-State (LS) Algorithm

Cuando el algoritmo LS termina, tenemos para cada nodo su predecesor a lo largo de la ruta de coste mínimo desde el nodo de origen. Para cada predecesor, también tenemos su predecesor, y así de este modo podemos construir la ruta completa desde el origen a todos los destinos.

Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Algoritmo de ruteo por vector de distancias (DV)

Mientras que el algoritmo LS es un algoritmo que emplea información global, el algoritmo por vector de distancias (DV) es asíncrono y distribuido.

Ecuación de Bellman-Ford

Sea $d_x(y)$ el costo de la ruta de costo mínimo desde el nodo x al nodo y.

$$d_x(y) = min_v c(x, v) + d_v(y) \quad (1)$$

donde min_v se calcula para todos los vecinos v de x.

Distance-Vector (DV)

La solución de la ecuación de Bellman-Ford proporciona las entradas de la tabla de reenvío del nodo x.

$$d_x(y) = min_v c(x, v) + d_v(y) \tag{2}$$

¿Qué debe comunicar un nodo a sus vecinos?

Distance-Vector (DV)

Cada nodo x mantiene la siguiente información:

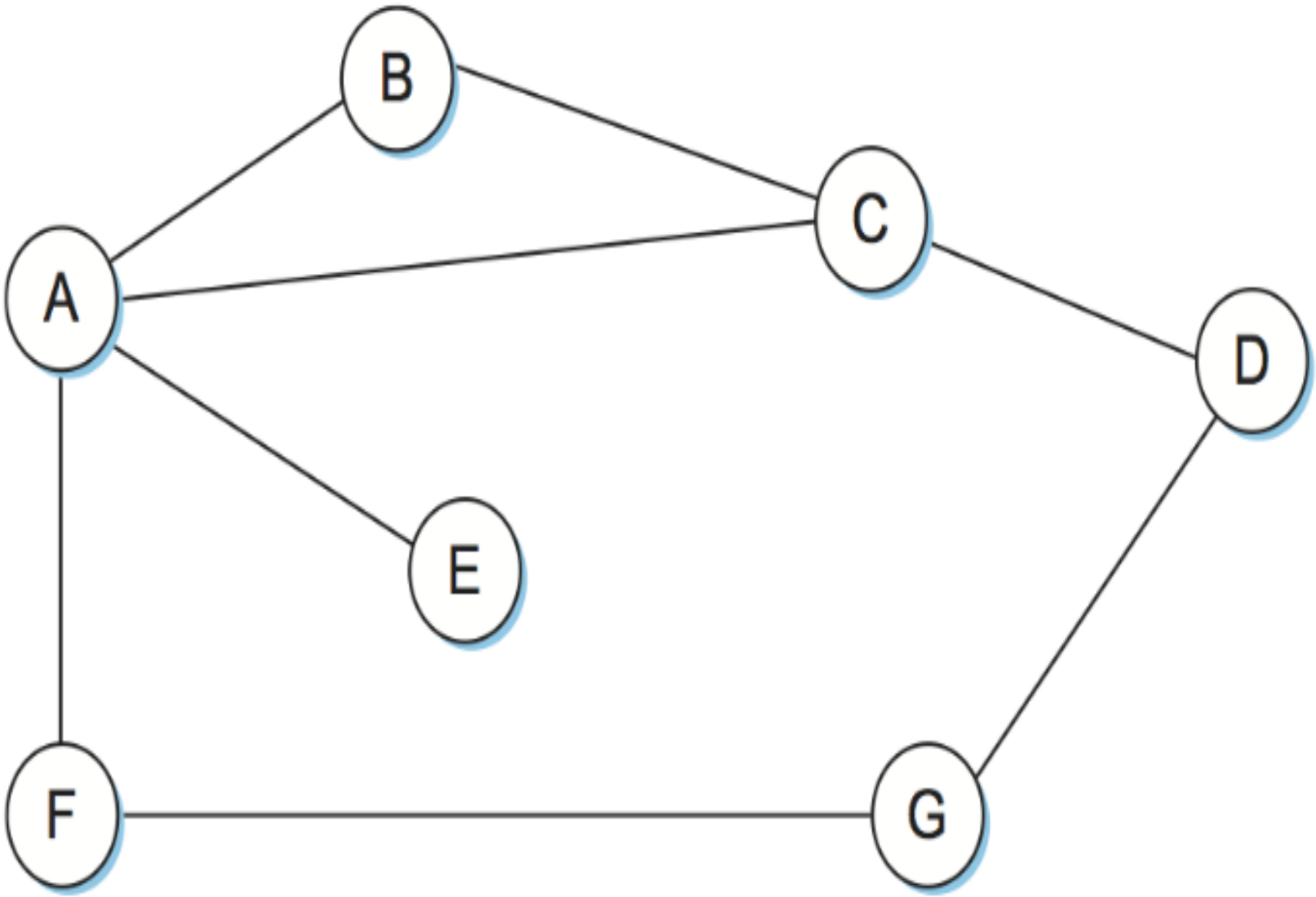
- Para cada vecino v, el costo $c(x, v)$.
- Su vector de distancias, el cual contiene la estimación que x hace de su costo hacia todos los destinos.
- Los vectores de distancias de cada uno de sus vecinos.

Distance-Vector (DV) Algorithm

En cada nodo x:

```
1  Initialization:
2    for all destinations y in N:
3       D_x(y) = c(x,y) /* if y is not a neighbor then c(x,y) = ∞ */
4    for each neighbor w
5       D_w(y) = ? for all destinations y in N
6    for each neighbor w
7       send distance vector  D_x = [D_x(y): y in N]  to w
8
9  loop
10    wait   (until I see a link cost change to some neighbor w or
11              until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14        D_x(y) = min_v{c(x,v) + D_v(y)}
15
16 if D_x(y) changed for any destination y
17       send distance vector D_x  = [D_x(y): y in N]  to all neighbors
18
19 forever
```

Distance-Vector: Ejemplo



Vectores de Distancias

En nodo A

A	B	C	D	E	F	G

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

En nodo E:

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

En nodo E:

A	B	C	D	E	F	G
1	∞	∞	∞	0	∞	∞

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

En nodo E:

A	B	C	D	E	F	G
1	∞	∞	∞	0	∞	∞

En nodo F:

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

En nodo E:

A	B	C	D	E	F	G
1	∞	∞	∞	0	∞	∞

En nodo F:

A	B	C	D	E	F	G
1	∞	∞	∞	∞	0	1

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

En nodo E:

A	B	C	D	E	F	G
1	∞	∞	∞	0	∞	∞

En nodo F:

A	B	C	D	E	F	G
1	∞	∞	∞	∞	0	1

En nodo G:

Vectores de Distancias

En nodo A

A	B	C	D	E	F	G
0	1	1	∞	1	1	∞

En nodo B:

A	B	C	D	E	F	G
1	0	1	∞	∞	∞	∞

En nodo C:

A	B	C	D	E	F	G
1	1	0	1	∞	∞	∞

En nodo D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

En nodo E:

A	B	C	D	E	F	G
1	∞	∞	∞	0	∞	∞

En nodo F:

A	B	C	D	E	F	G
1	∞	∞	∞	∞	0	1

En nodo G:

A	B	C	D	E	F	G
∞	∞	∞	1	∞	1	0

Nos paramos en nodo G

VD de G:

A	B	C	D	E	F	G
∞	∞	∞	1	∞	1	0

Nos paramos en nodo G

VD de G:

A	B	C	D	E	F	G
∞	∞	∞	1	∞	1	0

Recibe los vectores distancia de:

Nos paramos en nodo G

VD de G:

A	B	C	D	E	F	G
∞	∞	∞	1	∞	1	0

Recibe los vectores distancia de: **F y D**

VD de F:

A	B	C	D	E	F	G
1	∞	∞	∞	∞	0	1

VD de D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

Nos paramos en nodo G

VD de G:

A	B	C	D	E	F	G
∞	∞	∞	1	∞	1	0

Recibe los vectores distancia de: **F y D**

VD de F:

A	B	C	D	E	F	G
1	∞	∞	∞	∞	0	1

VD de D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

G entonces recomputa su propio vector distancia

Nos paramos en nodo G

VD de G:

A	B	C	D	E	F	G
∞	∞	∞	1	∞	1	0

Recibe los vectores distancia de: **F y D**

VD de F:

A	B	C	D	E	F	G
1	∞	∞	∞	∞	0	1

VD de D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

G entonces recomputa su propio vector distancia

VD de G:

A	B	C	D	E	F	G
2	∞	2	1	∞	1	0

Nos paramos en nodo G

VD de G:

A	B	C	D	E	F	G
∞	∞	∞	1	∞	1	0

Recibe los vectores distancia de: **F y D**

VD de F:

A	B	C	D	E	F	G
1	∞	∞	∞	∞	0	1

VD de D:

A	B	C	D	E	F	G
∞	∞	1	0	∞	∞	1

G entonces recomputa su propio vector distancia

VD de G:

A	B	C	D	E	F	G
2	∞	2	1	∞	1	0

Como el VD de G cambió, debe informarlo a sus vecinos

Vector Distancia: problema

- Supongamos que el enlace de A con E se cae, y A avisa sobre este hecho a sus vecinos.
- Sin embargo, dependiendo del momento exacto de los eventos, podría suceder que el nodo B, sabiendo que C llega a E con dos saltos, concluye que puede alcanzar E en 3 saltos, y le comunica esa información a A.
- Luego, A utiliza esa información para calcular una ruta a E de 4 saltos, vía B.
- y así sucesivamente...

Vector Distancia: problema

- Supongamos que el enlace de A con E se cae, y A avisa sobre este hecho a sus vecinos.
- Sin embargo, dependiendo del momento exacto de los eventos, podría suceder que el nodo B, sabiendo que C llega a E con dos saltos, concluye que puede alcanzar E en 3 saltos, y le comunica esa información a A.
- Luego, A utiliza esa información para calcular una ruta a E de 4 saltos, vía B.
- y así sucesivamente...
- Problema de *cuenta al infinito*.
- *poisson-reverse*