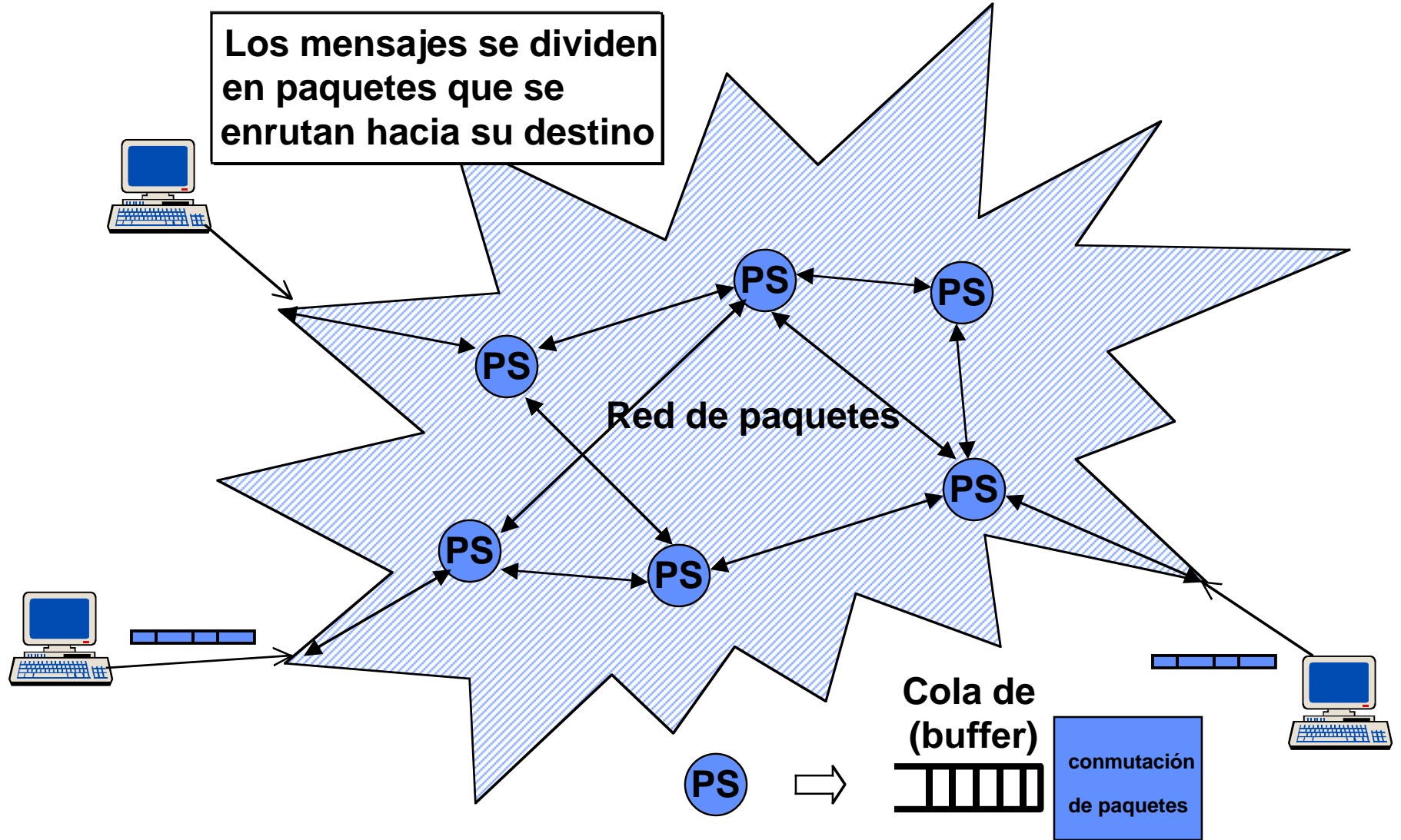

Clase 21: Enrutamiento en redes de transmisión de datos

Eytan Modiano

Redes conmutadas por paquetes



Enrutamiento

- **Hay que elegir las rutas para varios pares origen-destino (pares O/D) o para varias sesiones:**
 - **Enrutamiento de datagramas: la ruta se elige según una base paquete por paquete**

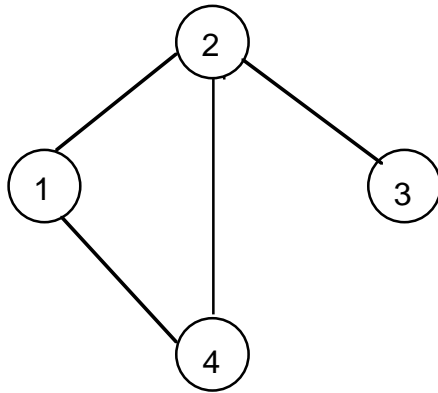
Utilizar un enrutamiento de datagramas es un modo sencillo de dividir las rutas
 - **Enrutamiento de circuito virtual: la ruta se elige según una base sesión por sesión**
 - **Enrutamiento estático: la ruta se elige según un modo preestablecido, basado en los pares O/D**

Enrutamiento de amplia difusión

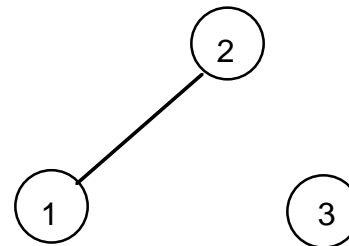
- **Se dirige un paquete desde un origen hacia todos los nodos de la red**
- **Posibles soluciones:**
 - **Inundación (*flooding*): cada nodo envía el paquete a todos los enlaces de salida:
Se descartan los paquetes recibidos por segunda vez**
 - **Enrutamiento de árbol de expansión: se envía el paquete a lo largo de un árbol que incluye todos los nodos de la red**

Grafos

- Un grafo $G = (N,A)$ consta de un conjunto finito no vacío de nodos N y un conjunto de pares de nodos A llamados arcos (enlaces o aristas)



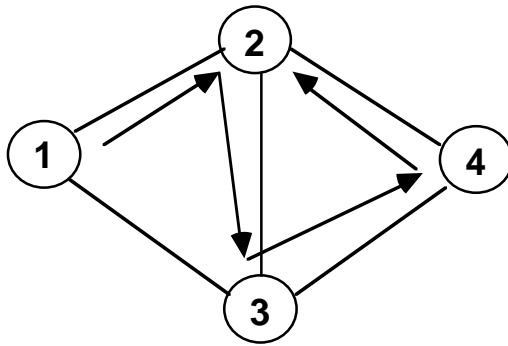
$$N = \{1,2,3,4\}$$
$$A = \{(1,2),(2,3),(1,4),(2,4)\}$$



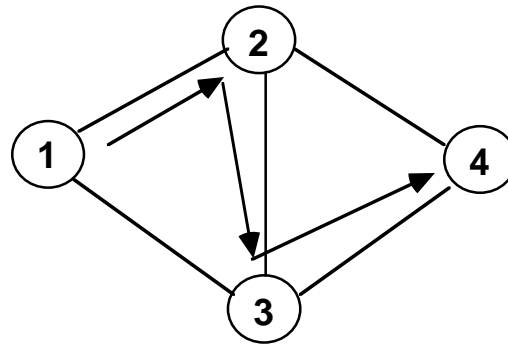
$$N = \{1,2,3\}$$
$$A = \{(1,2)\}$$

Caminos y rutas

- Un camino (walk) es una secuencia de nodos (n_1, n_2, \dots, n_k) en la que cada par de nodos adyacentes es un arco.
- Una ruta (path) es un camino sin nodos repetidos



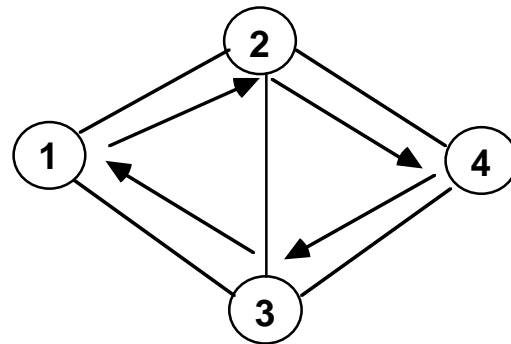
Camino (1,2,3,4,2)



Ruta (1,2,3,4)

Ciclos

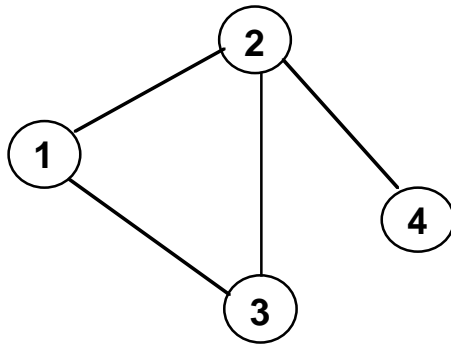
- Un ciclo es un camino (n_1, n_2, \dots, n_k) con $n_1 = n_k$, $k > 3$, y sin nodos repetidos excepto $n_1 = n_k$



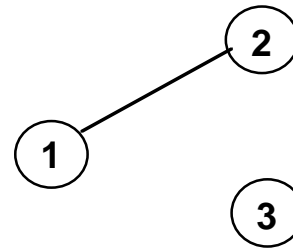
Ciclo (1,2,4,3,1)

Grafo conexo

- Un grafo es conexo si existe una ruta entre cada par de nodos



Conexo

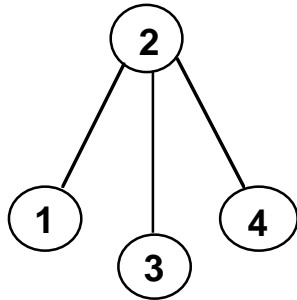


Inconexo

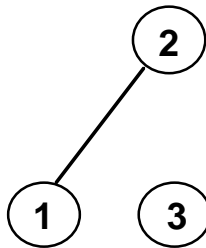
- Un grafo inconexo se puede separar en dos o más componentes conexos

Árboles y grafos acíclicos

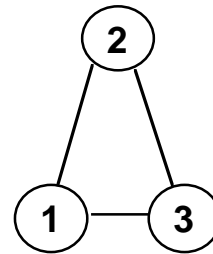
- Un grafo acíclico es un grafo sin ciclos
- Un árbol es un grafo acíclico conexo



**Acíclico,
conexo**



**Inconexo
no árbol**

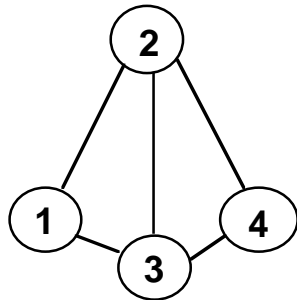


**Cíclico,
no árbol**

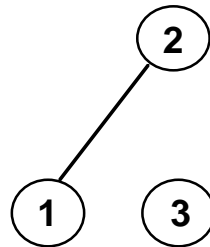
- El número de arcos en un árbol es siempre uno menos que el número de nodos:
 - Demostración: empezar con un nodo arbitrario y añadir un nodo cada vez que se añade un arco => N nodos y N-1 enlaces. Si se añade un arco sin añadir un nodo, el arco deberá ir a un nodo que ya esté en el árbol formando, así, un ciclo

Subgrafos

- $G' = (N', A')$ es un subgrafo de $G = (N, A)$ si:
 - 1) G' es un grafo
 - 2) N' es un subconjunto de N
 - 3) A' es un subconjunto de A
- Un subgrafo se obtiene eliminando nodos y arcos de un grafo:
 - Nota: los arcos adyacentes a un nodo eliminado también deben ser eliminados



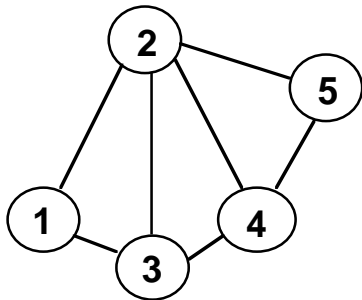
– Grafo



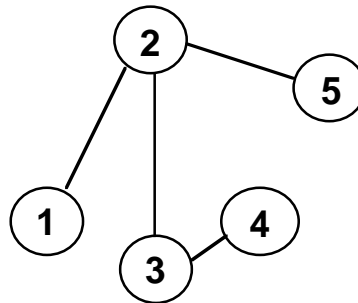
Subgrafo G' de G

Árboles de expansión (Spanning trees)

- $T = (N', A')$ es un árbol de expansión de $G = (N, A)$ si:
 - T es un subgrafo de G con $N' = N$ y T es un árbol



Grafo G



Árbol de expansión de G

Árboles de expansión

- **Los árboles de expansión son útiles para difundir y recoger información de control en las redes; a veces son útiles también para el enrutamiento**
- **Para difundir datos desde el nodo n:**
 - El nodo n realiza una amplia difusión de los datos en todos los arcos del árbol adyacente
 - Otros nodos retardan los datos en los arcos de otros árboles adyacentes
- **Para recoger datos en el nodo n:**
 - Todas las ramas del árbol (todas, salvo la n) envían datos
 - Los otros nodos (todos, salvo el n) esperan la recepción de datos en todos sus arcos salvo en uno adyacente y, a continuación, envían los datos recibidos más los propios por el arco restante

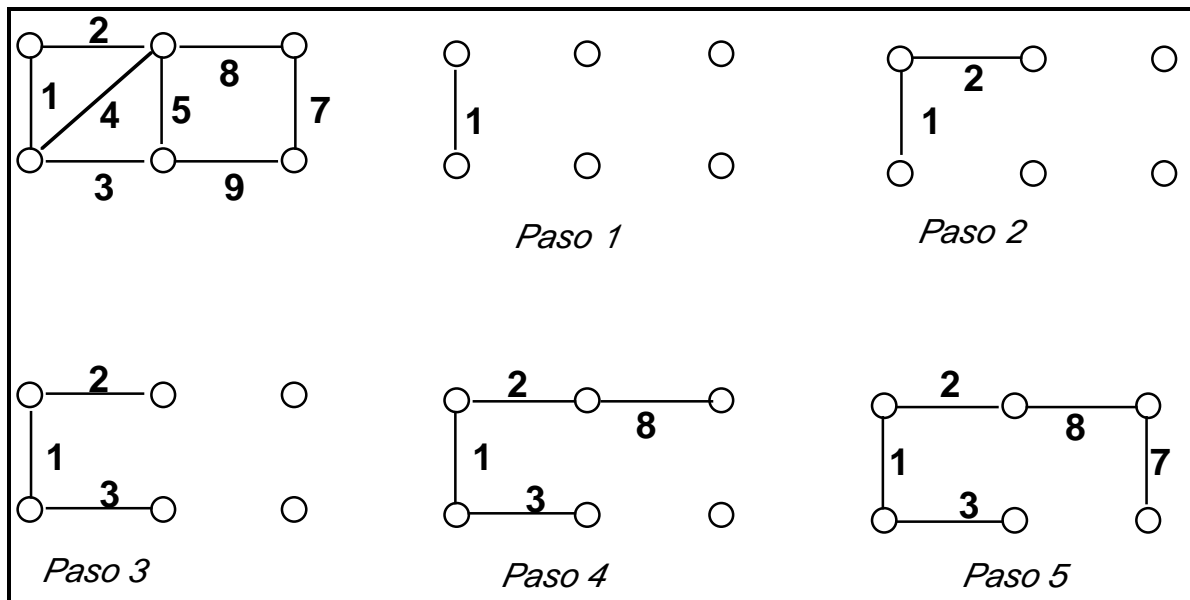
Construcción general de un árbol de expansión

- **Algoritmo para construir un árbol de expansión para un grafo conexo $G = (N,A)$:**
 - 1) **Seleccionar cualquier nodo n en N ; $N' = \{n\}$; $A' = \{ \}$**
 - 2) **Si $N' = N$, entonces parar ($T=(N',A')$ es un árbol de expansión)**
 - 3) **Elegir $(i,j) \in A$, $i \in N'$, $j \notin N'$**
 $N' := N' \cup \{j\}$; $A' := A' \cup \{(i,j)\}$; ir al paso 2
- **La conectividad de G garantiza que se puede elegir un arco en el paso 3 siempre que N' sea distinto de N**
- **¿Es único el árbol de expansión?**
- **¿Qué contribuye a un buen árbol de expansión?**

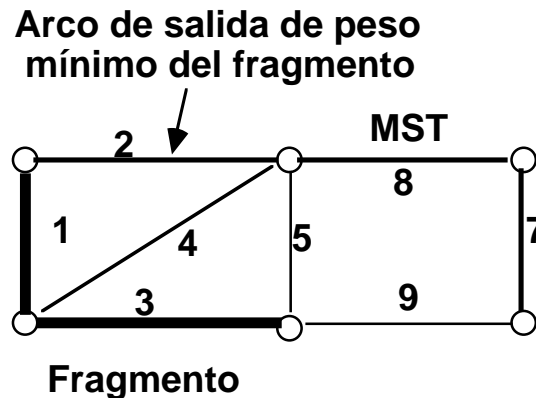
Árbol de expansión de peso mínimo (MST)

- **Pasos de un algoritmo MST genérico:**
 - Dado un conjunto de subárboles de un MST (llamados fragmentos) añadir un arco de salida de peso mínimo a algún fragmento
- **Prim-Dijkstra: empezar con un sólo nodo arbitrario como fragmento**
 - Añadir un arco de salida de peso mínimo
- **Kruskal: empezar con cada nodo como fragmento**
 - Añadir el arco de salida de peso mínimo, minimizado sobre todos los fragmentos

Algoritmo de Prim-Dijkstra



Ejemplo del algoritmo de Kruskal



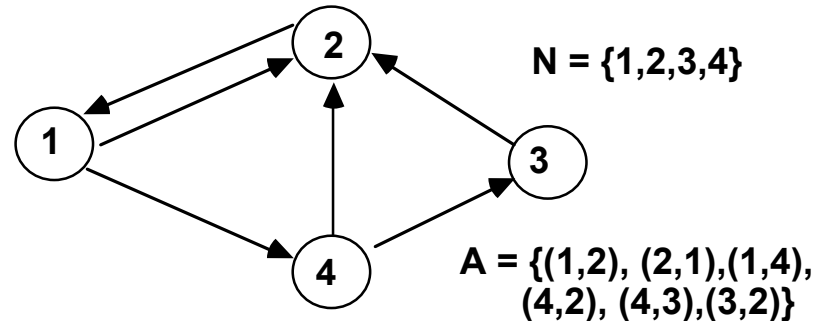
- **Suponer que los arcos de peso 1 y 3 son un fragmento:**
 - Analizar cualquier árbol de expansión utilizando estos arcos y el arco de peso 4, que es un arco de salida del fragmento
 - Suponer que el árbol de expansión no utiliza el arco de peso 2
 - Eliminar el arco de peso 4 y añadir el arco de peso 2 da lugar a otro árbol de menor peso
 - Así, un arco de salida de peso mínimo del fragmento debe estar en el MST.

Enrutamiento del camino más corto

- **Cada enlace tiene un coste que refleja:**
 - La longitud del enlace
 - El retardo del enlace
 - La congestión
 - El precio
- **El coste puede variar con el tiempo**
- **La longitud de la ruta es la suma de todos los costes a lo largo de la ruta**
- **El camino más corto es el camino con la mínima longitud**
- **Algoritmos del camino más corto:**
 - Bellman-Ford: versiones distribuidas y centralizadas
 - Algoritmo de Dijkstra
 - Muchos otros

Grafos dirigidos (digrafos)

- Un grafo dirigido (digrafo) $G = (N,A)$ es un conjunto finito no vacío de nodos N y un conjunto de pares de nodos ordenados A llamados arcos directos



- Camino dirigido: (4,2,1,4,3,2)
- Ruta dirigida: (4,2,1)
- Ciclo dirigido: (4,2,1,4)
- El mejor modo para representar las redes de datos es con digrafos aunque, por lo general, los enlaces tienden a ser bidireccionales (el coste puede ser distinto en cada dirección)
 - Para simplificar, en nuestros ejemplos utilizaremos enlaces bidireccionales de costes iguales

Algoritmo de Bellman Ford

- **Calcula las rutas más cortas desde un nodo de origen dado (por ejemplo, el nodo 1) a todos los demás nodos**
- **Idea general:**
 - Primero se calcula la ruta más corta de un solo arco
 - Luego, la ruta más corta de al menos dos arcos, etc.
 - Sea $d_{ij} = \infty$ si (i,j) no es un arco
- **Establecemos $D_i(h)$ como la distancia más corta desde 1 hasta i usando, como máximo, h arcos:**
 - $D_i(1) = d_{1i}$; $i \neq 1$ $D_1(1) = 0$
 - $D_i(h+1) = \min \{j\} [D_j(h) + d_{ji}]$; $i \neq 1$ $D_1(h+1) = 0$
- **Si todos los pesos son positivos, el algoritmo termina en $N-1$ pasos**

Ejemplo - Bellman Ford

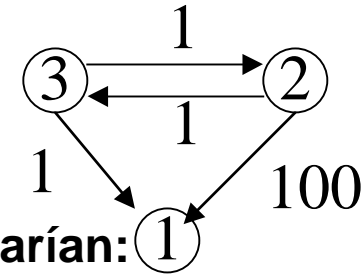
Bellman Ford distribuido

- **Los costes de los enlaces pueden variar con el tiempo:**
 - Cambios en las condiciones del tráfico
 - Fallos de los enlaces
 - Movilidad
- **Cada nodo mantiene su propia tabla de enrutamiento:**
 - Es necesario actualizar la tabla regularmente para que refleje los cambios de la red
- **Establecemos D_i como la distancia más corta desde el nodo i hasta el destino**
 - $D_i = \min \{j\} [D_j + d_{ij}]$: ecuación de actualización
- **Cada nodo (i) actualiza regularmente los valores de D_i utilizando la ecuación anterior:**
 - Cada nodo mantiene los valores de d_{ij} para sus vecinos, así como los valores de D_j recibidos de sus vecinos
 - Utiliza dichos valores para calcular D_i y envía un nuevo valor de D_i a sus vecinos
 - Si no se producen cambios en la red, el algoritmo obtendrá las rutas más cortas en N pasos como máximo

Reacción lenta ante errores de los enlaces

- Empezar con $D_3=1$ y $D_2=100$

- Tras una iteración, el nodo 2 recibe $D_3=1$ y $D_2 = \min [1+1, 100] = 2$



- En la práctica, las longitudes de los enlaces a veces varían:

- Supongamos que falla el enlace entre los nodos 3 y 1 (es decir, $d_{31}=\infty$)
- El nodo 3 se actualizará $D_3 = d_{32} + D_2 = 3$
- En el siguiente paso se actualizará el nodo 2: $D_2 = d_{23} + D_3 = 4$

- Tendrán que pasar casi 100 iteraciones antes de que el nodo 2 encuentre la ruta correcta al nodo 1

- Posibles soluciones:

- Difundir también información de rutas
- Esperar antes de reenrutar por un camino, con el correspondiente aumento de coste
El nodo próximo al enlace fallido debería anunciar $D=\infty$ durante algún tiempo para evitar bucles

Algoritmo de Dijkstra

- **Calcula el camino más corto desde un nodo de origen dado a todos los demás nodos:**
 - Requiere que los pesos de los arcos no sean negativos
- **El algoritmo se desarrolla en etapas:**
 - Etapa k: en ella se calcula cuáles son los k nodos más cercanos al origen
 - Etapa k+1: dados los k nodos más próximos al nodo de origen, se averigua el k+1
- **Observación clave: la ruta hacia los k+1 nodos más cercanos incluye sólo nodos que se encuentren entre los k nodos más cercanos**
- **Sea M el conjunto de nodos ya incorporados por el algoritmo:**
 - Empezar con $D_n = d_{sn}$ para todo n (D_n = distancia del camino más corto desde el nodo n al nodo de origen)
 - Repetir hasta que $M=N$
 - Encontrar el nodo $w \notin M$ que presenta la siguiente distancia de menor coste al nodo de origen
 - Añadir w a M
 - Actualizar las distancias: $D_n = \min [D_n, D_w + d_{wn}]$ (para todos los nodos $n \notin M$)
 - Obsérvese que la actualización de D_n sólo es necesaria para los nodos que ya no están en M y que la actualización requiere únicamente el cálculo de una nueva distancia atravesando el nodo w recientemente añadido

Ejemplo de Dijkstra

Implementación del algoritmo de Dijkstra

- **Versión centralizada:** un solo nodo tiene información sobre la topología y calcula las rutas:
 - Posteriormente, las rutas se pueden difundir ampliamente por el resto de la red
- **Versión distribuida:** cada nodo i difunde $\{d_{ij} \text{ todo } j\}$ a todos los nodos de la red; luego, todos los nodos pueden calcular los caminos más cortos al resto de los nodos:
 - Protocolo OSPF (Open Shortest Path First) utilizado en Internet

Enrutamiento en Internet

- **Sistemas autónomos (AS)**
 - Internet está dividida en sistemas autónomos, cada uno de ellos bajo el control de una única autoridad
- **Los protocolos de enrutamiento se pueden clasificar en dos categorías:**
 - **Protocolos interiores:** operan dentro de un AS
 - **Protocolos exteriores:** operan entre sistemas autónomos
- **Protocolos interiores:**
 - **Generalmente utilizan algoritmos del camino más corto**
 - Vector distancia: basado en el Bellman-ford distribuido
 - Protocolos del estado de los enlaces: basados en el Dijkstra "distribuido"

Protocolos del vector distancia

- **Basados en el algoritmo de Bellman-Ford distribuido:**
 - Los nodos intercambian información de la tabla de enrutamiento con sus vecinos
- **Ejemplos:**
 - **Protocolos de información de enrutamiento (RIP)**
 - La métrica utilizada es el conteo de saltos ($d_{ij}=1$)
 - La información de enrutamiento se intercambia cada 30 segundos
 - **Protocolo de enrutamiento de pasarela interior (IGRP)**
 - Propiedad de CISCO
 - La métrica tiene en cuenta la carga
 - $D_{ij} \sim 1/(\mu-\lambda)$ (retardo estimado a través del enlace)

 - Se actualiza cada 90 segundos
 - Capacidad de enrutamiento multiruta

Protocolos de estado de los enlaces

- **Basados en el algoritmo de la ruta más corta de Dijkstra:**
 - Evita los bucles
 - Los routers monitorizan el estado de sus enlaces de salida
 - Los routers difunden el estado de sus enlaces dentro del AS
 - Cada nodo conoce el estado de todos los enlaces y puede calcular todas las rutas utilizando el algoritmo de Dijkstra:
 - Sin embargo, los nodos sólo envían el paquete al siguiente nodo de la ruta, con la dirección de destino de los paquetes. El siguiente nodo consultará la dirección en la tabla de enrutamientos
- **Ejemplo: OSPF (Open Shortest Path First) usado con frecuencia en Internet**
- **Los protocolos de estado de los enlaces suelen generar menos tráfico de "control" que los del vector distancia**

Enrutamiento entre dominios

- **Utilizado para enrutar paquetes a través de diferentes AS**
- **Opciones:**
 - **Enrutamiento estático: rutas configuradas manualmente**
 - **Enrutamiento del vector distancia:**
 - Protocolo de pasarela exterior (EGP)**
 - Protocolo de pasarela de borde o fronteriza (BGP)**
- **Problemas:**
 - **¿Qué coste tiene para la "métrica" el uso del enrutamiento por vector distancia?**
 - Problemas de relaciones: un proveedor de red A puede no querer que los paquetes de B circulen por su red o bien los dos proveedores pueden llegar a un acuerdo**
 - Problemas de coste: los proveedores de red se pueden cobrar entre ellos por la entrega de paquetes**