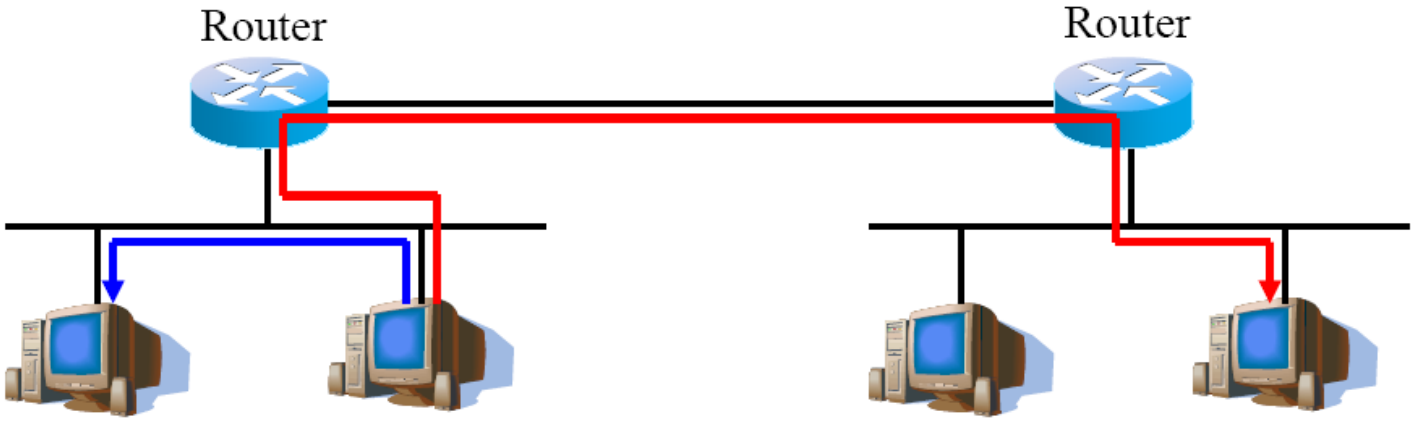
ET4230

CHAPTER 3: NETWORK LAYER

ROUTING ALGORITHM

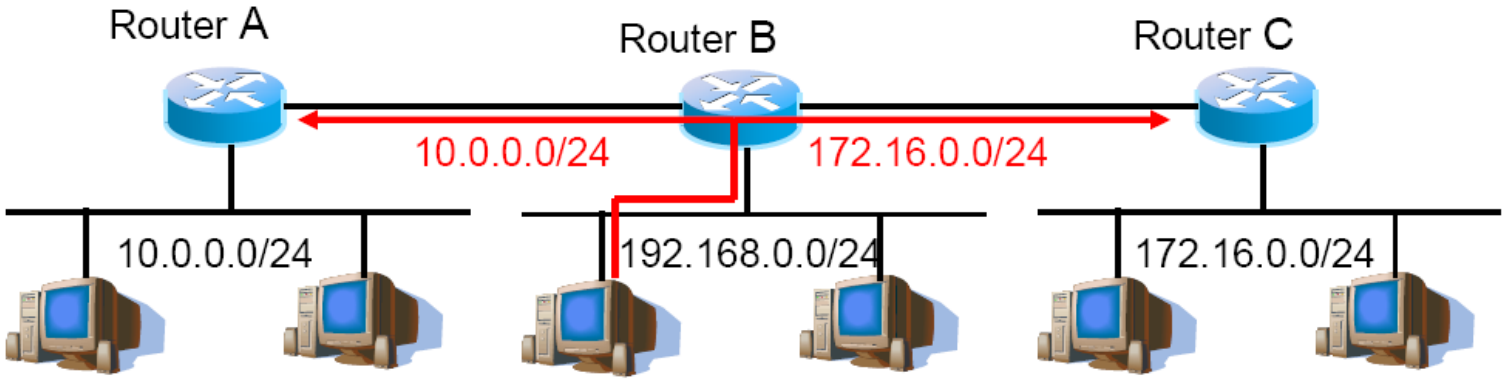
TỔNG QUAN

- KHÁI NIỆM
  - Quá trình chọn đường đi qua các nút mạng để tới đích một cách tối ưu
  - Định tuyến ở lớp mạng sử dụng mô hình định tuyến hop-by-hop
- THÀNH PHẦN
  - Bảng định tuyến (routing table)
  - Giải thuật định tuyến



TỔNG QUAN

- BẢNG ĐỊNH TUYẾN
  - Danh sách các đường đi có thể từ nguồn đến đích
  - Địa chỉ đích/mặt nạ mạng, Router kế tiếp (interface, metris)



10

Lưu ý quy tắc: *No routes, no reachability*!

dest. network	net. mask	next hop	interface	metrics
10.0.0.0	255.255.255.0	A' IP addr.	1	1
172.16.0.0	255.255.255.0	C' IP addr.	2	1

TỔNG QUAN

- NGUYÊN TẮC ĐỊNH TUYẾN CỦA ROUTER

Longest prefix match

Ví dụ: địa chỉ đích: 11.1.2.5 và bảng định tuyến sau

Network	Next hop
11.0.0.0/8	A
11.1.0.0/16	B
11.1.2.0/24	C

11.1.2.5 = 00001011.00000001.00000010.00000101

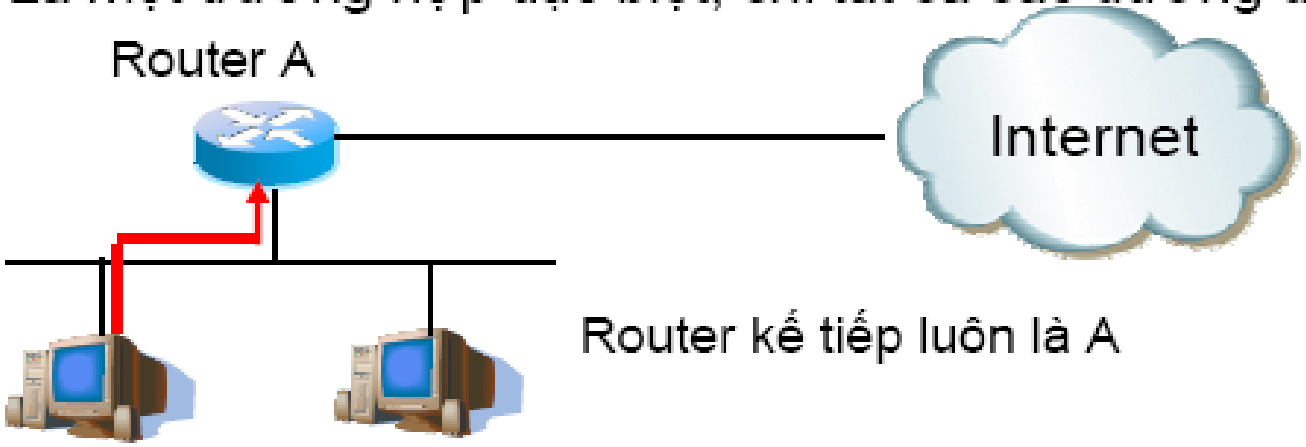
11.0.0.0/8 = 00001011.00000000.00000000.00000000

11.1.0.0/16 = 00001011.00000001.00000000.00000000

11.1.2.0/24 = 00001011.00000001.00000010.00000000

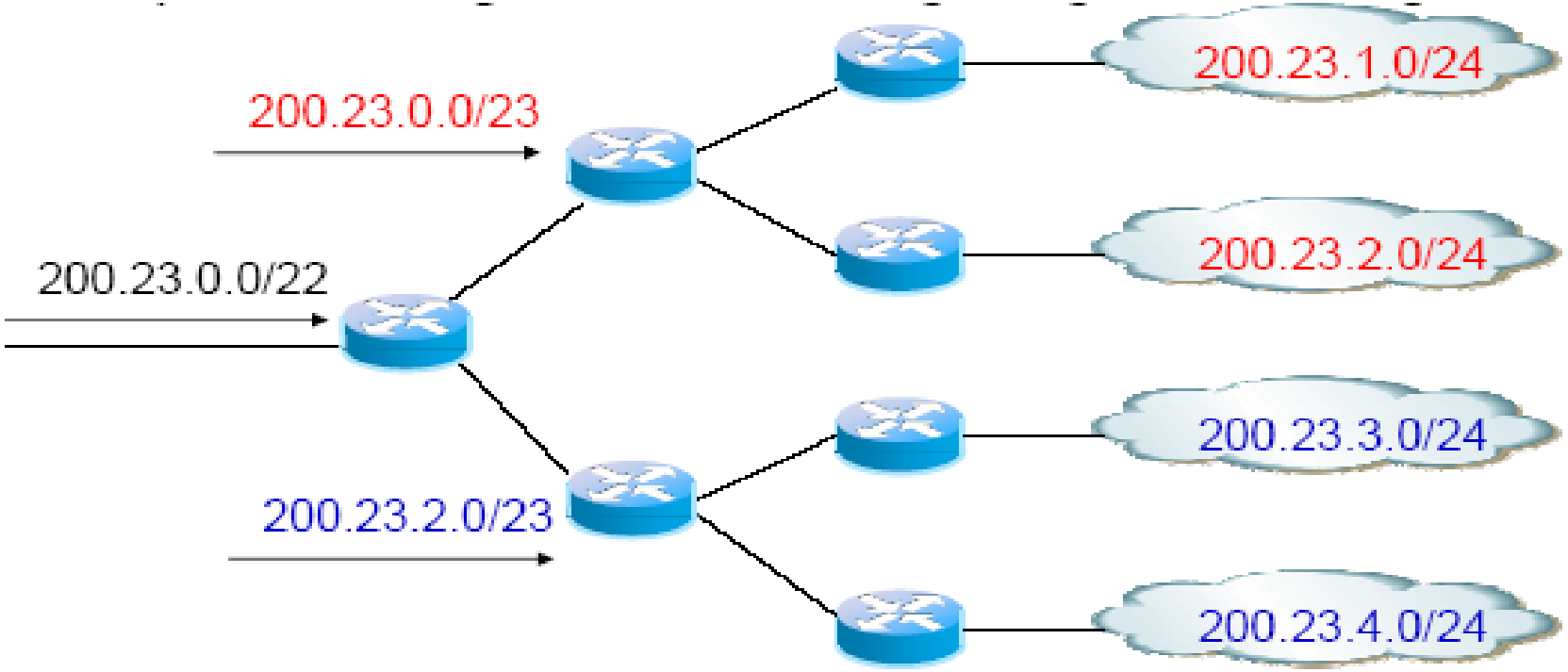
TỔNG QUAN

- ĐƯỜNG ĐI MẶC ĐỊNH
  - Nếu đường đi không tìm thấy trong bảng chọn đường
    - Đường đi mặc định trỏ đến một router kết tiếp
    - Trong nhiều trường hợp, đây là đường đi duy nhất
  - 0.0.0.0/0
    - Là một trường hợp đặc biệt, chỉ tất cả các đường đi



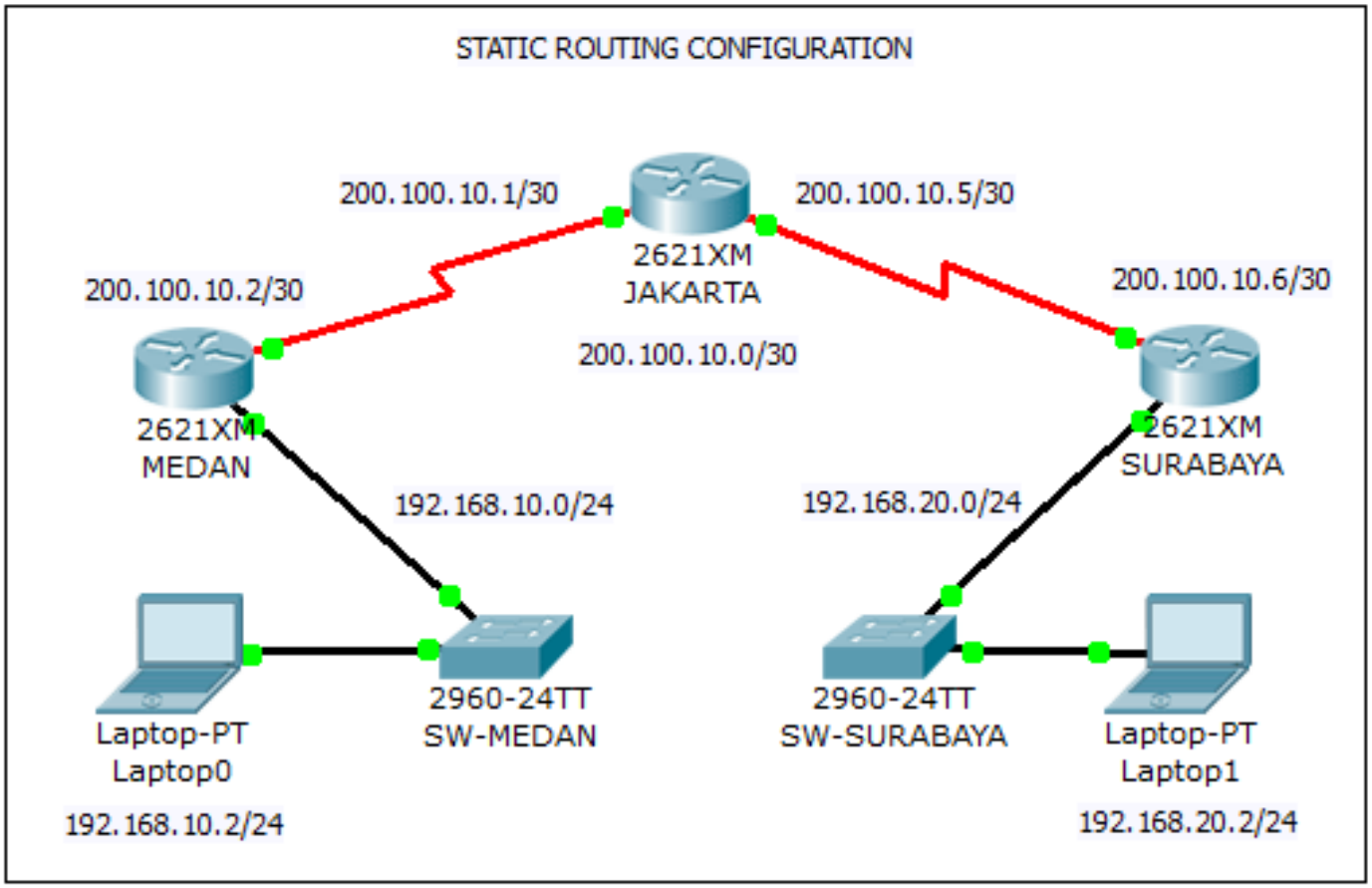
TỔNG QUAN

- KẾT HỢP ĐƯỜNG ĐI



CÁC PHƯƠNG PHÁP ĐỊNH TUYẾN

- ĐỊNH TUYẾN TĨNH
  - Không sử dụng giao thức định tuyến
  - Cập nhật bảng định tuyến thủ công



CÁC PHƯƠNG PHÁP ĐỊNH TUYẾN

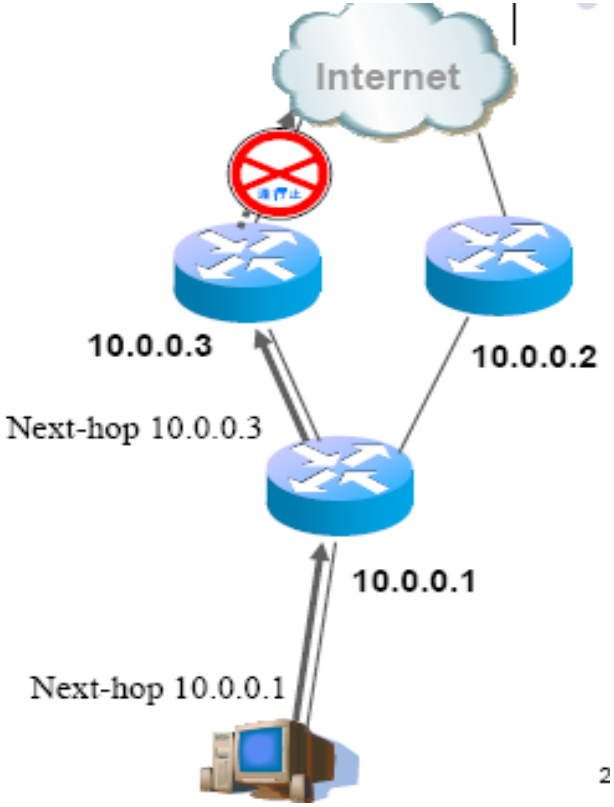
- VÍ DỤ

- Khi có sự cố:
  - Không thể nối vào Internet kể cả khi có tồn tại đường đi dự phòng
  - Người quản trị mạng cần thay đổi

Bảng chọn đường của 10.0.0.1 (1 phần)

Prefix	Next-hop
0.0.0.0/0	10.0.0.3

Kết nối bị lỗi



CÁC PHƯƠNG PHÁP ĐỊNH TUYẾN

- ĐỊNH TUYẾN ĐỘNG
  - Sử dụng các giao thức định tuyến
  - Lựa chọn tuyến dựa trên thông tin trạng thái hiện thời của mạng
    - Đáp ứng tính thời gian thực
  - Mô hình tập trung: được xây dựng từ hệ thống tính toán định tuyến
    - Thu thập thông tin vào một nút mạng
    - Sử dụng các giải thuật tìm đường đi trên đồ thị
    - Phân bổ bảng định tuyến từ nút trung tâm đến các nút
  - Mô hình phân tán:
    - Mỗi nút tự xây dựng bảng định tuyến riêng
    - Sử dụng các giao thức định tuyến
    - Được sử dụng phổ biến trong thực tế

CÁC PHƯƠNG PHÁP ĐỊNH TUYẾN

- VÍ DỤ

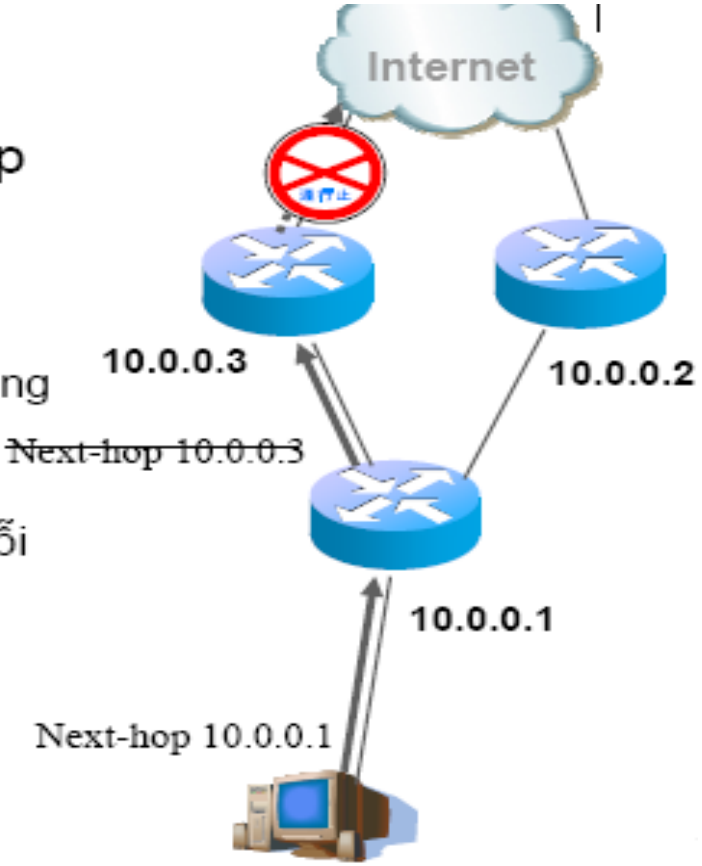
- Khi có sự cố:
  - Đường đi thay thế được cập nhật một cách tự động

Bảng chọn đường của 10.0.0.1 (1 phần)

Prefix	Next-hop
0.0.0.0/0	**10.0.0.2**
~~0.0.0.0/0~~	~~10.0.0.3~~

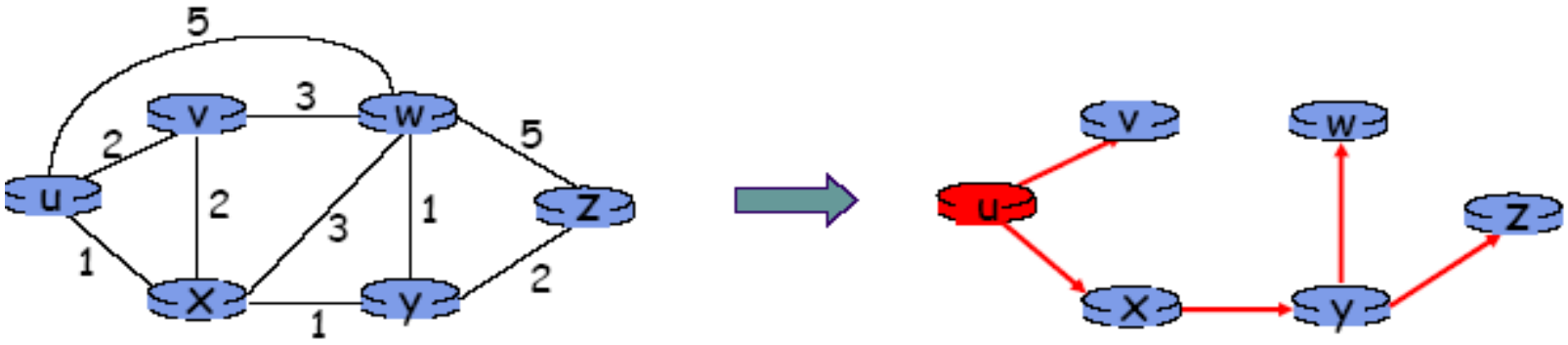
Kết nối dự phòng

Kết nối bị lỗi



CÁC THUẬT TOÁN ĐỊNH TUYẾN

- MỤC TIÊU
  - Tìm đường đi ngắn nhất từ một nút gốc tới các nút còn lại
  - Xây dựng cây theo đường ngắn nhất (shortest path tree - SPT)
- BIỂU DIỄN MẠNG BẰNG ĐỒ THỊ



- SPT – Shortest Path Tree
- Các cạnh xuất phát từ nút gốc và tới các lá
- Đường đi duy nhất từ nút gốc tới nút v, là đường đi ngắn nhất giữa nút gốc và nút v
- Mỗi nút sẽ có một SPT của riêng nút đó

Bellman-Ford

- DVA – Distance Vector Algorithm
  - Dùng thuật toán Bellman-Ford

- Input
  - Đồ thị G(V, E) trong đó V là tập đỉnh, E là tập cạnh có trọng số
  - Đỉnh nguồn S: $S \in V$

- Output
  - Đồ thị có chu trình âm $\rightarrow$ không tồn tại đường đi ngắn nhất
  - Đường đi ngắn nhất từ đỉnh nguồn S đến tất cả các đỉnh còn lại

- Ký hiệu
  - $D(h)_i$: đường đi ngắn nhất từ node nguồn S đến node i có tối đa h đoạn (link).
  - d_{ij}: trọng số trên cạnh nối từ node i đến node j
    d_{ij} = 0 nếu i trùng j
    d_{ij} = Eij nếu i khác j

Bellman-Ford

Giải thuật

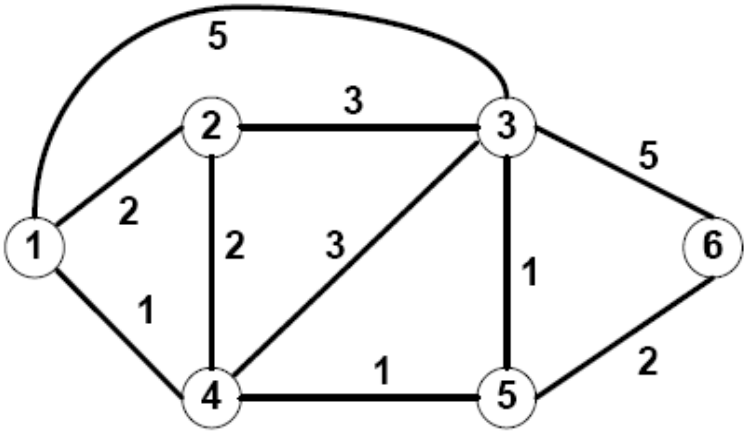
- Bước 1: khởi động
  - $D(1)_N = d_{SN}$, $\forall N \in V\backslash\{S\}$

  (đường đi ngắn nhất từ S đến N có tối đa 1 đoạn)

- Bước 2: cập nhật đường đi ngắn nhất
  - $D(h+1)_N = \min \{D(h)_j + d_{jN}\} \forall j \in V\backslash\{S\}$

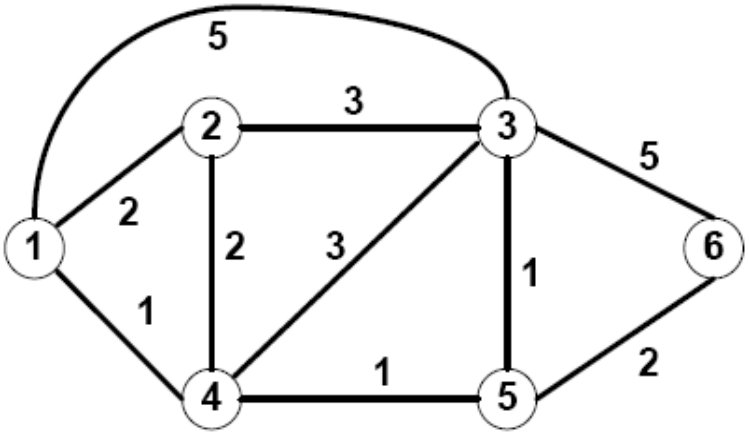
- Bước 3: lặp lại bước 2 cho đến khi không có đường đi mới nào ngắn hơn được tìm thấy thì dừng

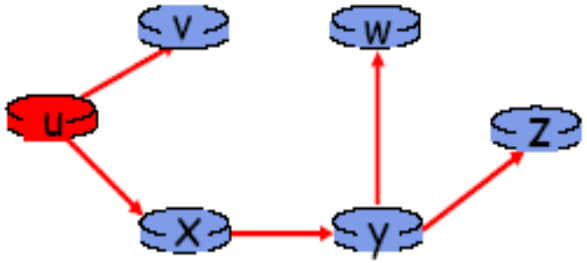
- Kết quả $D(h)_N$ sẽ là đường đi ngắn nhất từ node nguồn S đến node N



B1: tìm đường đi ngắn nhất đi qua 1 cạnh
B2: tìm đường đi ngắn nhất đi qua 2 cạnh

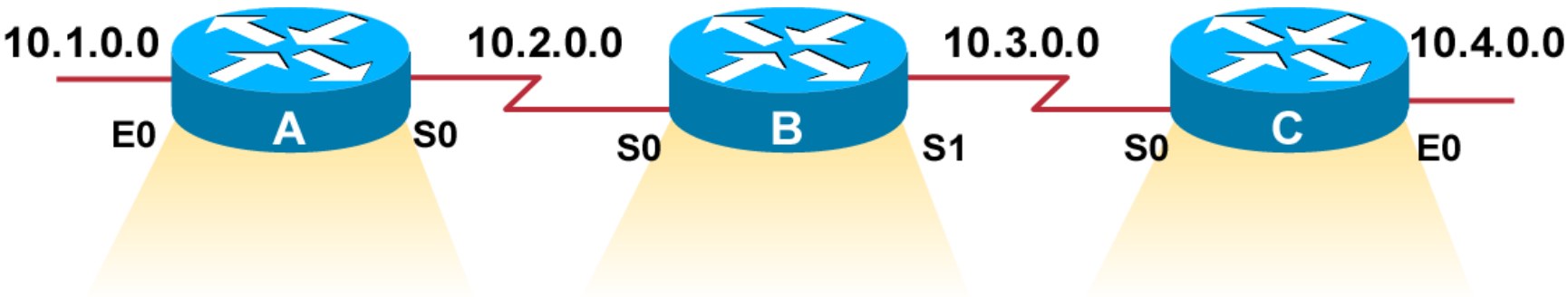
Bellman-Ford





Lần	Node 2		Node 3		Node 4		Node 5		Node 6	
chạy	$D(h)_2$	Path	$D(h)_3$	Path	$D(h)_4$	Path	$D(h)_5$	Path	$D(h)_6$	Path
1	2	1–2	5	1–3	1	1–4	∞	---	∞	---
2	2	1–2	4	1–4–3	1	1–4	2	1–4–5	10	1–3–6
3	2	1–2	3	1–4–5–3	1	1–4	2	1–4–5	4	1–4–5–6
4	2	1–2	3	1–4–5–3	1	1–4	2	1–4–5	4	1–4–5–6

Count to Infinity



Routing Table		
10.1.0.0	E0	0
10.2.0.0	S0	0
10.3.0.0	S0	1
10.4.0.0	S0	2

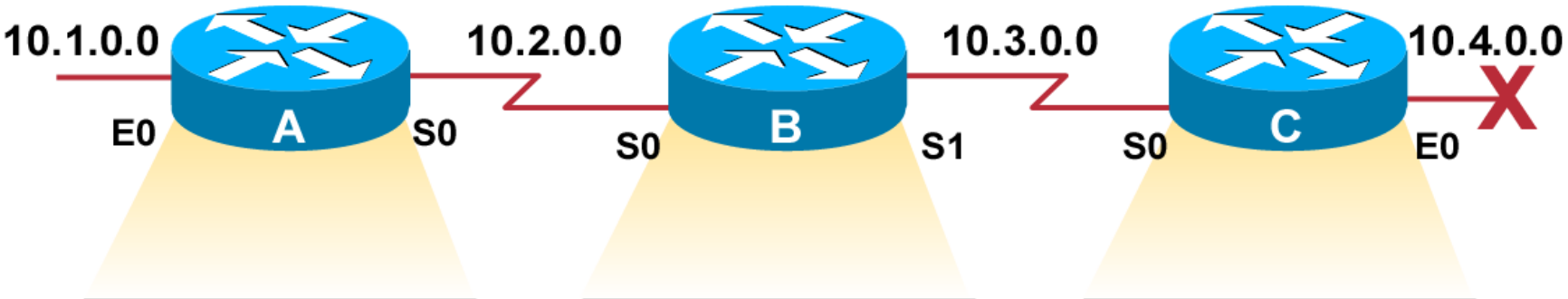
Routing Table		
10.2.0.0	S0	0
10.3.0.0	S1	0
10.4.0.0	S1	1
10.1.0.0	S0	1

Routing Table		
10.3.0.0	S0	0
10.4.0.0	E0	0
10.2.0.0	S0	1
10.1.0.0	S0	2

ICND20GR_205

Khả năng bị phân kì, lặp vô hạn

Count to Infinity



Routing Table		
10.1.0.0	E0	0
10.2.0.0	S0	0
10.3.0.0	S0	1
10.4.0.0	S0	2

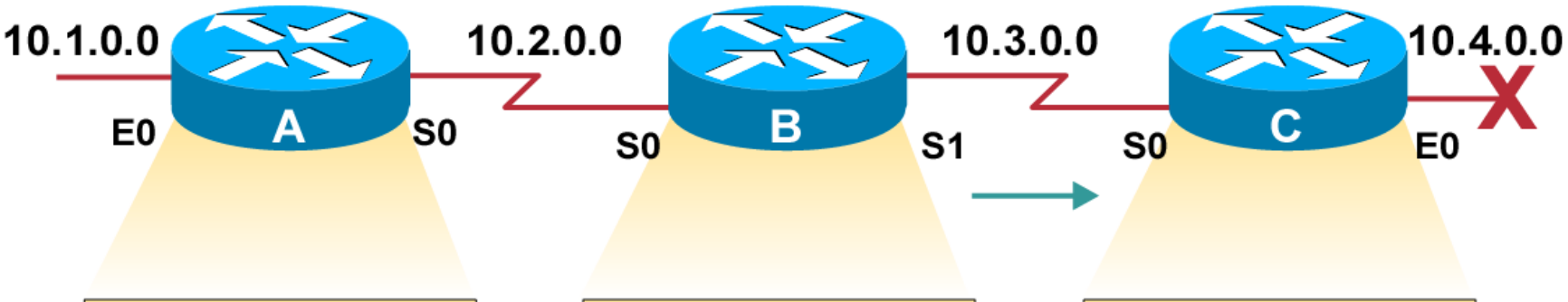
Routing Table		
10.2.0.0	S0	0
10.3.0.0	S1	0
10.4.0.0	S1	1
10.1.0.0	S0	1

Routing Table		
10.3.0.0	S0	0
10.4.0.0	E0	Down
10.2.0.0	S0	1
10.1.0.0	S0	2

ICND20GR_206

Router A không biết mạng 10.4.0.0 bị lỗi

Count to Infinity



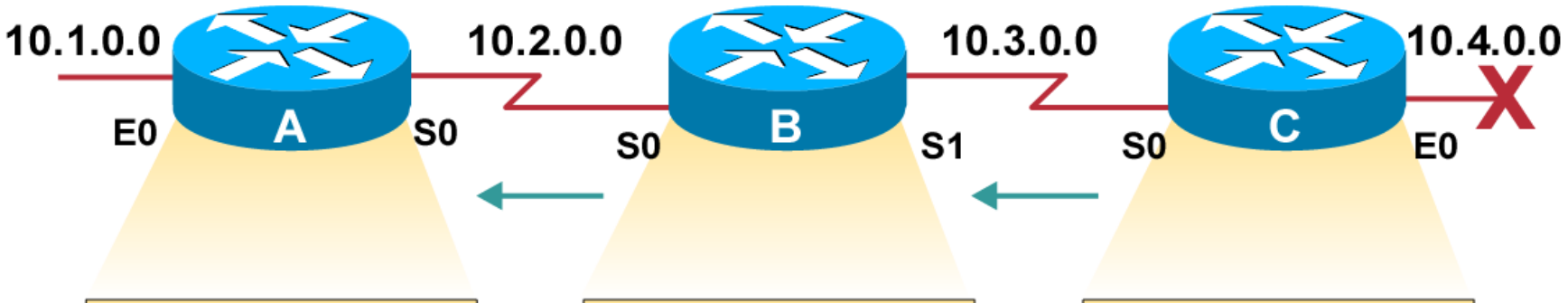
Routing Table		
10.1.0.0	E0	0
10.2.0.0	S0	0
10.3.0.0	S0	1
10.4.0.0	S0	2

Routing Table		
10.2.0.0	S0	0
10.3.0.0	S1	0
10.4.0.0	S1	1
10.1.0.0	S0	1

Routing Table		
10.3.0.0	S0	0
10.4.0.0	S0	2
10.2.0.0	S0	1
10.1.0.0	S0	2

ICND20GR_207

Count to Infinity



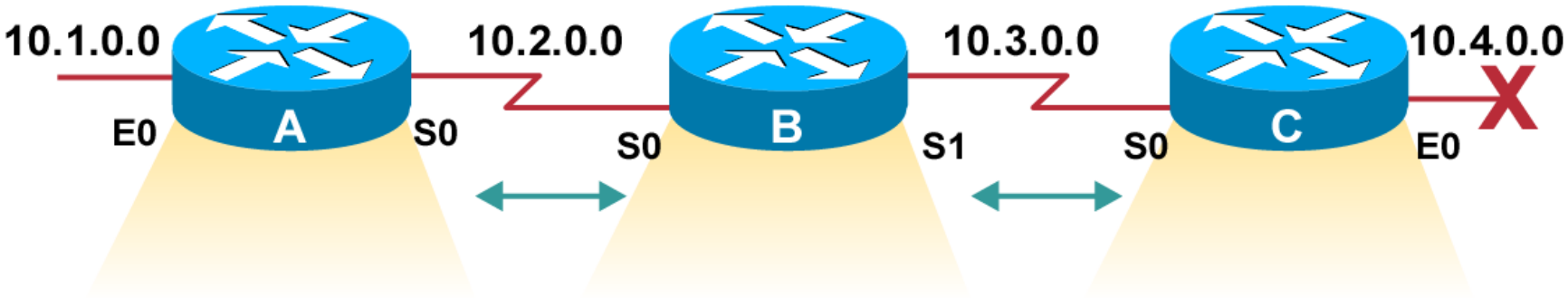
Routing Table		
10.1.0.0	E0	0
10.2.0.0	S0	0
10.3.0.0	S0	1
10.4.0.0	S0	4

Routing Table		
10.2.0.0	S0	0
10.3.0.0	S1	0
10.4.0.0	S1	3
10.1.0.0	S0	1

Routing Table		
10.3.0.0	S0	0
10.4.0.0	S0	2
10.2.0.0	S0	1
10.1.0.0	S0	2

ICND20GR_208

Count to Infinity



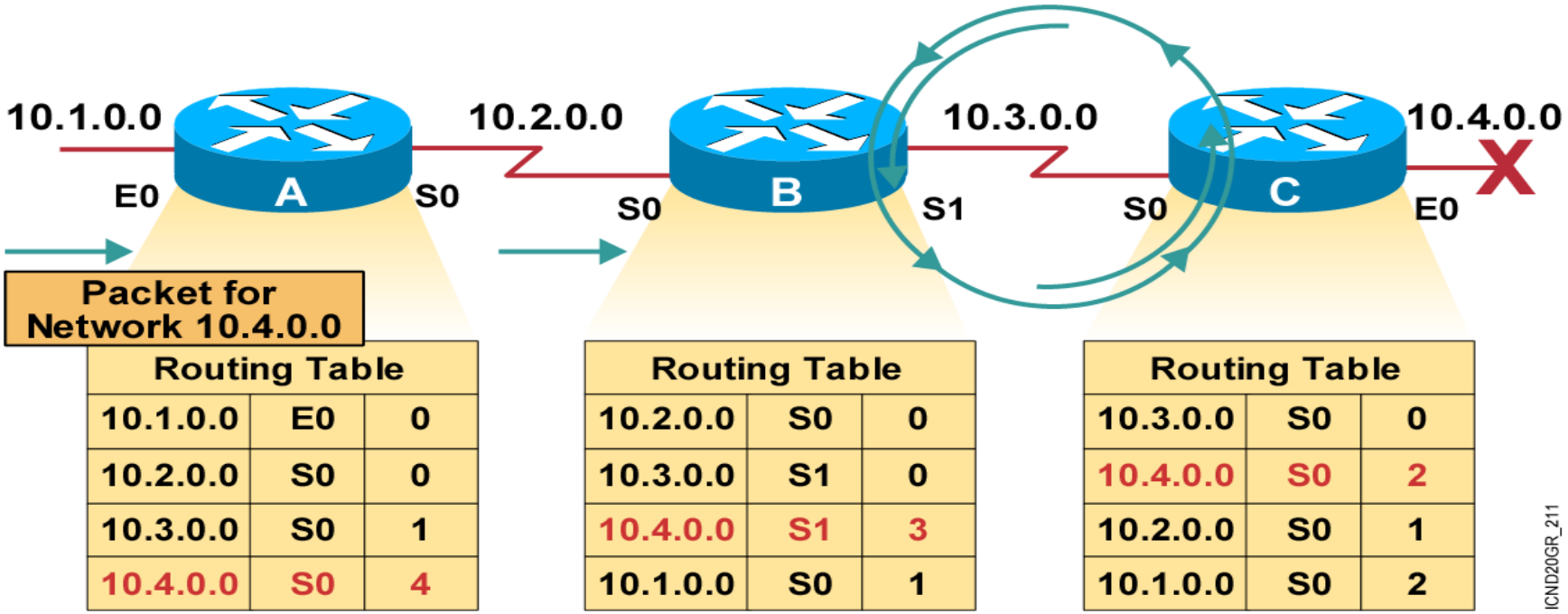
Routing Table		
10.1.0.0	E0	0
10.2.0.0	S0	0
10.3.0.0	S0	1
10.4.0.0	S0	6

Routing Table		
10.2.0.0	S0	0
10.3.0.0	S1	0
10.4.0.0	S1	5
10.1.0.0	S0	1

Routing Table		
10.3.0.0	S0	0
10.4.0.0	S0	4
10.2.0.0	S0	1
10.1.0.0	S0	2

ICND20GR_209

Count to Infinity



Routing Table		
10.1.0.0	E0	0
10.2.0.0	S0	0
10.3.0.0	S0	1
10.4.0.0	S0	4

Routing Table		
10.2.0.0	S0	0
10.3.0.0	S1	0
10.4.0.0	S1	3
10.1.0.0	S0	1

Routing Table		
10.3.0.0	S0	0
10.4.0.0	S0	2
10.2.0.0	S0	1
10.1.0.0	S0	2

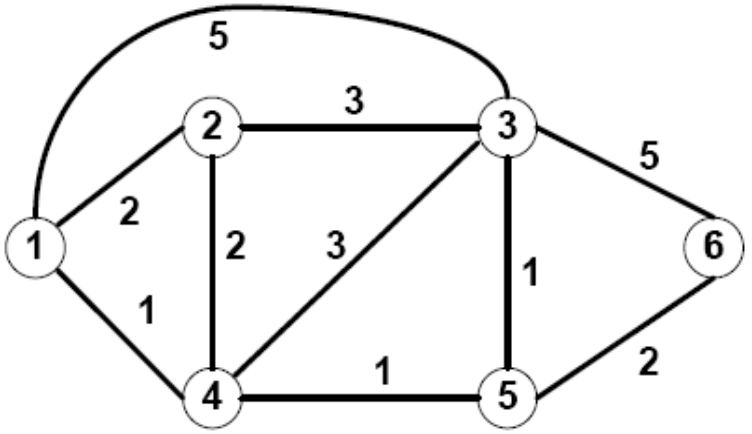
ICND20GR_211

link-state advertisement (LSA)

- Thuật toán định tuyến trạng thái liên kết
  Nếu 1 nút bị lỗi thì tất cả các node sẽ biết
  - Mỗi nút phải duy trì "bản đồ" toàn mạng
    tăng lưu lượng bản tin trong mạng
  - Thuật toán Dijkstra (dùng trong **OSPF**)

    - Input
      - Đồ thị G(V, E) trong đó V là tập đỉnh, E là tập cạnh có trọng số không âm
      - Đỉnh nguồn S: $S \in V$
    - Output
      - Đường đi ngắn nhất từ đỉnh nguồn S đến tất cả các đỉnh còn lại
    - Ký hiệu
      - D_j : đường đi ngắn nhất từ node nguồn S đến node i tại bước chạy hiện hành của giải thuật
      - M: tập các đỉnh đã xét tại bước chạy hiện hành của giải thuật
      - d_{ij}: trọng số trên cạnh nối từ node i đến node j
        $d_{ij} = 0$ nếu i trùng j
        $d_{ij} = E_{ij}$ nếu i khác j

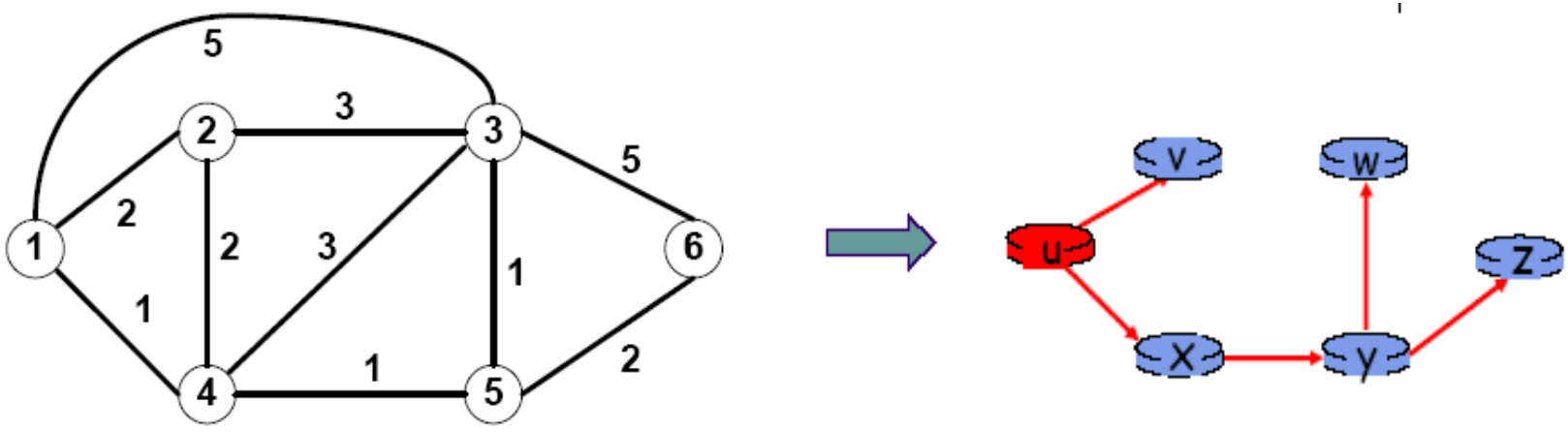
link-state advertisement (LSA)



- Giải thuật
  - Bước 1: khởi động
    - M = {S}
    - $D_i = d_{si}$ (các cạnh nối trực tiếp với S)
  - Bước 2: cập nhật đường đi ngắn nhất
    - Chọn đỉnh $N \in V\backslash M$ sao cho: $D_N = \min \{D_i\}$ $\forall i \in V\backslash M$
    - $M = M \cup \{N\}$
    - $D_j = \min \{D_j, D_N + d_{Nj}\}$ $\forall j \in V\backslash M$
  - Bước 3: lặp lại bước 2 cho đến khi M=V
  - Kết quả D_i sẽ là đường đi ngắn nhất từ node nguồn S đến node i

Chọn đỉnh ngắn nhất đến S sau đó từ đỉnh đó tính ra các đỉnh khác

link-state advertisement (LSA)



Lần chạy	M	Node 2 D_2	Node 2 Path	Node 3 D_3	Node 3 Path	Node 4 D_4	Node 4 Path	Node 5 D_5	Node 5 Path	Node 6 D_6	Node 6 Path
1	1	2	1 – 2	5	1 – 3	1	1 – 4	∞	---	∞	---
2	1 , 4	2	1 – 2	4	1 – 4 – 3	1	1 – 4	2	1 – 4 – 5	∞	---
3	1 , 4 , 2	2	1 – 2	4	1 – 4 – 3	1	1 – 4	2	1 – 4 – 5	∞	---
4	1 , 4 , 2 , 5	2	1 – 2	3	1 – 4 – 5 – 3	1	1 – 4	2	1 – 4 – 5	4	1 – 4 – 5 – 6
5	1 , 4 , 2 , 5 , 3	2	1 – 2	3	1 – 4 – 5 – 3	1	1 – 4	2	1 – 4 – 5	4	1 – 4 – 5 – 6
6	1 , 4 , 2 , 5 , 3 ,6	2	1 – 2	3	1 – 4 – 5 – 3	1	1 – 4	2	1 – 4 – 5	4	1 – 4 – 5 – 6

So sánh Dijkstra vs. Bellman-Ford

- Bellman-Ford
  - Việc tính toán cho node n phải biết các thông tin về chi phí liên kết của các node kề của n và chi phí tổng cộng từ node nguồn s đến các node kề của node n
  - Mỗi node cần lưu trữ tập các chi phí và các đường đi tương ứng đến các node khác
  - Có thể trao đổi thông tin với các node kề trực tiếp
  - Có thể cập nhật thông tin về chi phí và đường đi dựa trên các thông tin trao đổi với các node kề và các thông tin về chi phí liên kết
- Dijkstra
  - Mỗi node cần biết topology toàn bộ mạng
  - Phải biết chi phí liên kết của tất cả các liên kết trong mạng
  - Phải trao đổi thông tin với tất cả các node khác trong mạng

Lịch sử Định tuyến trong ARPANET

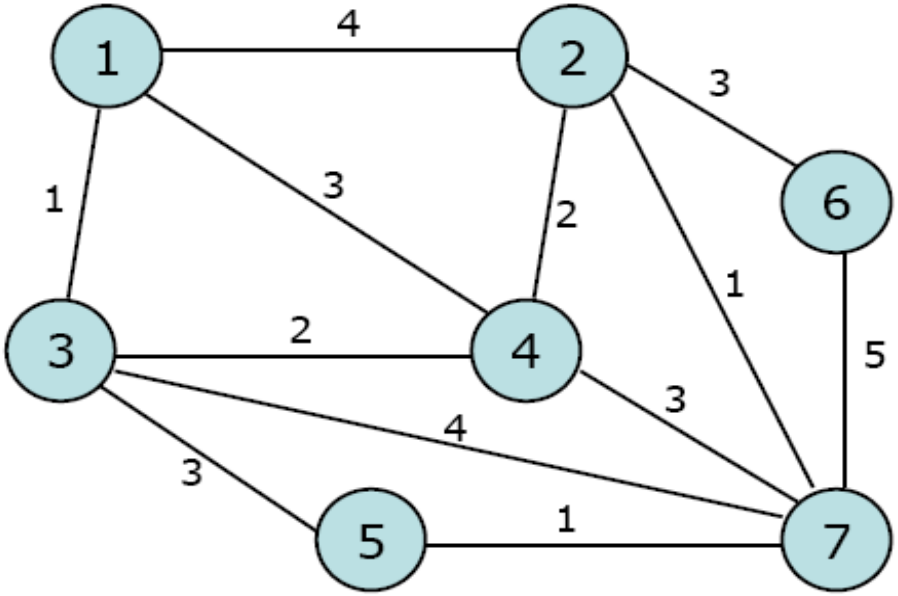
- Thế hệ đầu tiên (1969)
  - Distributed adaptive
  - Dùng thời gian trễ ước tính làm tiêu chuẩn để đánh giá hiệu quả
  - Dùng giải thuật tìm đường Bellman-Ford
    - Các node trao đổi thông tin (các vector thời gian trễ) với các node kề
    - Cập nhật bảng tìm đường dựa trên thông tin đến
  - Không quan tâm đến tốc độ đường truyền, chỉ quan tâm chiều dài hàng đợi tại các node
  - Chiều dài hàng đợi không phải là cách đo chính xác của thời gian trễ
    - Đáp ứng chậm với nghẽn mạch

Lịch sử Định tuyến trong ARPANET

- Thế hệ thứ 2 (1979)
  - Dùng thời gian trễ làm tiêu chuẩn đánh giá hiệu quả
  - Thời gian trễ được đo trực tiếp
  - Dùng giải thuật tìm đường Dijkstra
  - Thích hợp cho mạng có tải trung bình hoặc nhẹ
  - Khi mạng tải nặng, có ít tương quan giữa thời gian trễ đo được và thời gian trễ gặp phải
- Thế hệ thứ 3 (1987)
  - Việc tính toán chi phí của liên kết đã được thay đổi
  - Thời gian trễ trung bình được đo trong 10 giây cuối
  - Bình thường hóa dựa trên giá trị hiện tại và kết quả trước đó

BÀI TẬP

- Tìm đường ngắn nhất từ node 1
  - Theo giải thuật Dijkstra
  - Theo giải thuật Bellman-Ford



BÀI TẬP

- Tìm đường ngắn nhất từ node A
  - Theo giải thuật Dijkstra
  - Theo giải thuật Bellman-Ford