UIT
Trường Đại học
Công nghệ Thông tin

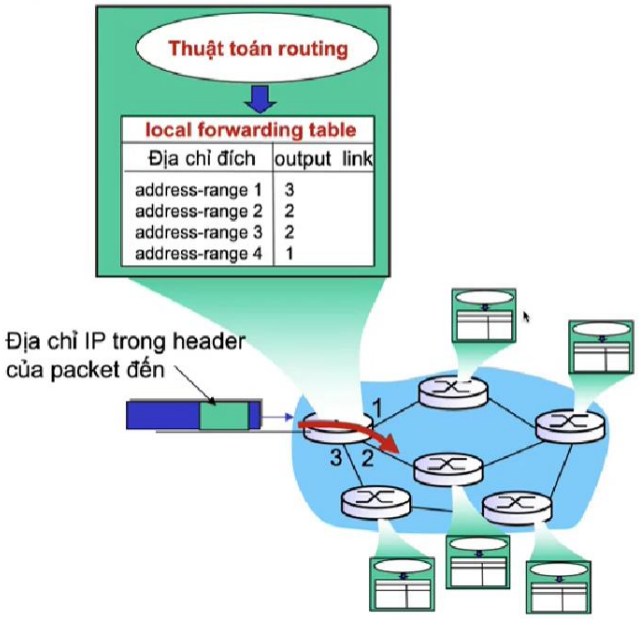
Khoa Khoa học
và Kỹ thuật Thông tin

NHẬP MÔN MẠNG MÁY TÍNH

Chương 4: Tầng Mạng

(Tiếp Theo)

*** Tác động lẫn nhau giữa routing và forwarding**



Routing: mang tính global (toàn cục), xác định được đường đi có "chi phí" thấp nhất, hay còn gọi là tốt nhất cho gói tin từ đầu cuối này đến đầu cuối khác trong Mạng. Sử dụng thuật toán routing ta thu được output là bảng Forwarding (Forwarding Table), dựa vào bảng này ta có thể biết được Gói tin có địa chỉ bao nhiêu sẽ đi ra ở đầu ra nào.

Forwarding: sau khi gói tin đi đến interface của router thì việc đưa nó đến interface output phù hợp sẽ dựa vào forwarding table.

4.5. Các Thuật Toán Routing

Phân loại thuật toán Routing

- Nút (node) là router
- Neighbor là nút láng giềng, kết nối trực tiếp với router, 1 router có thể có nhiều neighbors.

Toàn cục hay Phân cấp?

- Toàn cục (global):

+ Tất cả các router trong mạng thu thập toàn bộ thông tin kết nối của toàn bộ mạng.

+ Sử dụng giải thuật tìm đường đi trên độ thị (Dijkstra) ⇒ Thuật toán Link State.

+ Phân phối bảng định tuyến từ trung tâm tới các routers khác trong mạng.

- Phân cấp (Decentralized):

+ Mỗi router tự xây dựng bảng chọn đường riêng dựa trên kết nối với các nút láng giềng với nó → lặp đi lặp lại quy trình tính toán và trao đổi thông tin với neighbor → dần tìm ra con đường có chi phí ngắn nhất tới đích.

+ Sử dụng giải thuật định tuyến: Distance vector.

+ Được sử dụng phổ biến hơn trong thực tế.

4.5.1. Link State

→ *Sử dụng thuật toán Dijsktra*

- Các nút biết các thông tin của tất cả các nút trong mạng bằng cách quảng bá các gói tin trạng thái (link state broadcast) chứa các thông tin định danh, chi phí của các liên kết với neighbor của nó.

- Kết quả của việc này là tất các nút đều có cái nhìn giống nhau và toàn diện về cấu trúc mạng. Mỗi nút sau đó có thể tự chạy giải thuật **Link State** và tính toán đường đi ngắn nhất giống như các nút khác.

- **Giải thuật Dijkstra** tính toán con đường ngắn nhất từ 1 nút nguồn đến tất cả các nút khác trong mạng bằng việc lặp đi lặp lại k lần, sẽ biết được đường đi có chi phí thấp nhất của k đích.

- Các ký hiệu:

+ c(x, y): chi phí kết nối từ nút x đến nút y, = ∞ nếu x không trực tiếp kết nối với y.

+ D(v): chi phí của đường đi tốt nhất từ nút nguồn tới đích v sau khi thực hiện giải thuật.

+ p(v): nút trước của v trên đường đi ngắn nhất từ nguồn tới đích.

+ N': tập các nút mà đường đi ngắn nhất đã được biết.

- *Giải thuật Link State cho nút nguồn u:*

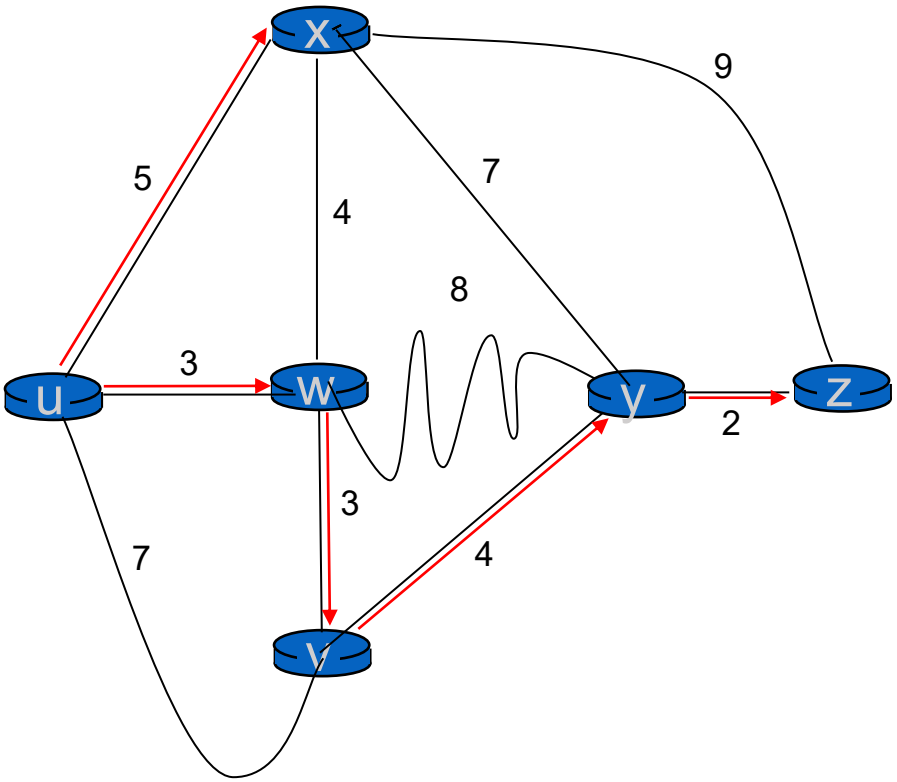
+ Bước khởi tạo ban đầu.

+ Sau đó là 1 vòng lặp. Số lần lặp bằng số nút trong mạng

→ Kết quả: các con đường ngắn nhất từ u đến mỗi nút trong mạng.

1 Khởi tạo
2 N' = { u }
3 Với nút v bất kì
4 Nếu v là láng giềng của u
5 Thì D(v) = c(u,v)
6 Ngược lại D(v) = ∞
7
8 Lặp
9 Tìm w không trong N' mà D(w) có giá trị nhỏ nhất đến nút
10 trong N'
11 Cập nhật D(v) cho mỗi láng giềng v của w mà không
12 nằm trong N':
13 D(v) = (D(v), D(w) + c(w,v))
14 Đến khi hết tất cả các nút trong N'

Ví dụ:



Bước	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
Khởi tạo	u	7, u	**3, u**	5, u	∞	∞
1	uw	6. w		**5, u**	11, w	∞
2	uwx	**6, w**			11, w	14, x
3	uwxv				**10, v**	14, x
4	uwxvy					**12, y**
5	uwxvyz					

Giải thích: - Trong bước khởi tạo, các chi phí đường đi ngắn nhất từ nút nguồn u tới các láng giềng liên kết trực tiếp với nó (v, w, x) lần lượt là (7; 3; 5). Các chi phí tới (y, z) được thiết lập là ∞ vì nó không kết nối trực tiếp tới u.

- Ở vòng lặp đầu tiên, trong số các nút v, w, x, y, z, chưa có nút nào được thêm vào N'; và thấy rằng nút có đường đi với chi phí nhỏ nhất là w với D(w) = 3 (kết quả của vòng lặp tìm chi phí trước), vì vậy w được thêm vào N'.

- Sau đó, ta sẽ cập nhật D cho tất cả các nút không nằm trong N' còn lại giống như dòng 13 trong giải thuật phía trên: Chi phí tới v (ban đầu là 7, trước nút đó là u) được cập nhật lại thành 6 thông qua nút trước đó được tìm thấy với chi phí thấp hơn là w. Chi phí tới nút x thì không thay đổi. Tương tự như v, x đối với lần lượt y và z, chi phí đến y (thông qua w) được cập nhật lại thành 11, và z giữ nguyên ∞.

- Ở vòng lặp thứ 2, nút x được tìm thấy là có chi phí đường đi nhỏ nhất (5), nên ta đưa x vào N'. Tập N' lúc này gồm (u, w, x). Chi phí của các nút còn lại chưa có trong N' (v, y, z) được cập nhập qua dòng 13 trong giải thuật ở trên, kết quả được chỉ ra trong dòng bước 2 ở bảng tính toán trên.

- Tiếp tục cho đến khi N' tập hợp tất cả các nút.

→ Khi thuật toán LS kết thúc, chúng ta biết được nút trước của mỗi nút trên con đường đi ngắn nhất từ nút nguồn. Với mỗi nút trước, ta lại

có nút trước của nó, vì vậy chúng ta có thể xây dựng một đường đi hoàn chỉnh từ nguồn cho tới mọi đích.

- **Độ phức tạp** khi tính toán với n nút là **O(n2)** với n(n+1)/2 phép so sánh.

- *Có một vấn đề nảy sinh, đó là sự dao động của đường đi tốt nhất*, khi chi phí dựa trên lưu lượng của đường đi có thể thay đổi khiến cho giải thuật Link State tạo ra kết quả khác. Sự dao động này có thể xảy ra ở bất kỳ giải thuật nào.

+ Một giải pháp đề ra đó là các chi phí liên kết không dựa vào lưu lượng mạng, nhưng đây là một giải pháp không thể chấp nhận được khi mục tiêu của định tuyến là tránh sự tắc nghẽn ở các liên kết.

+ Một giải pháp khác hợp lý hơn đó là đảm bảo các bộ định tuyến không cùng chạy thuật toán Link State trong cùng một thời điểm. Tuy nhiên, nghiên cứu cho thấy các bộ định tuyến có thể tự đồng bộ hóa với nhau → Một cách xử lí đó là cho các bộ định tuyến ngẫu nhiên thời gian nó gửi ra thông tin quảng bá đường liên kết.

4.5.2. Distance Vector

Ý tưởng:

- Mỗi nút định kỳ nhận ước lượng distance vector từ một hoặc nhiều neighbor nối trực tiếp với nó.
- Sau khi nhận DV (Distance Vector) mới, nó cập nhật lại DV cũ dùng công thức Bellman-Ford:

$$d_x(y) = \min\{c(x,v) + d_v(y)\}$$

Với:

  - $d_x(y)$: Chi phí đường đi nhỏ nhất từ x đến y.
  - $c(x,v)$: Chi phí từ x đến v (lân cận của x).
  - $d_v(y)$: Chi phí đường đi nhỏ nhất từ v đến y (lân cận của v)

- Chỉ khi DV thay đổi, nút mới thông báo đến các nút láng giềng.
- Các nút lặp đi lặp lại quá trình này cho đến khi không còn thông tin trao đổi giữa các láng giềng.

→ Giải thuật là thuật toán lặp đi lặp lại, bất đồng bộ, và phân tán

Giải thuật:

- Mỗi nút bắt đầu với $D = [D_x(y): y \in N]$ là vector của ước lượng chi phí từ x tới các nút y trong N. Với giải thuật DV, mỗi nút x duy trì các thông tin định tuyến sau:
  - Đối với mỗi láng giềng x, chi phí c(x, v) từ x với láng giềng nối trực tiếp, v.
  - $D_x = [D_x(y): y \in N]$, là vector khoảng cách của nút x chứa chi phí ước lượng của x tới tất cả các đích y trong N.
  - $D = [D_v(y): y \in N]$, là vector khoảng cách của mỗi láng giềng v thuộc N của x.
- Từ thời điểm này tới thời điểm khác, mỗi nút gửi một bảng các vector khoảng cách tới các láng giềng của nó. Khi một nút x nhận vector khoảng cách từ bất kì láng giềng v nào của nó, nó cập nhật lại vector khoảng cách của nó bằng công thức Bellman-Ford như sau:

$$D_x(y) = \min_v\{c(x, v) + D_v(y)\}$$

- Nếu như sau khi cập nhật mà vector khoảng cách của x thay đổi thì nút x, sau đó, sẽ gửi bảng vector khoảng cách vừa cập nhật của nó cho các láng giềng của nó. Các láng giềng này cũng sẽ cập nhật các vector khoảng cách của nó.