Giải thuật chọn đường

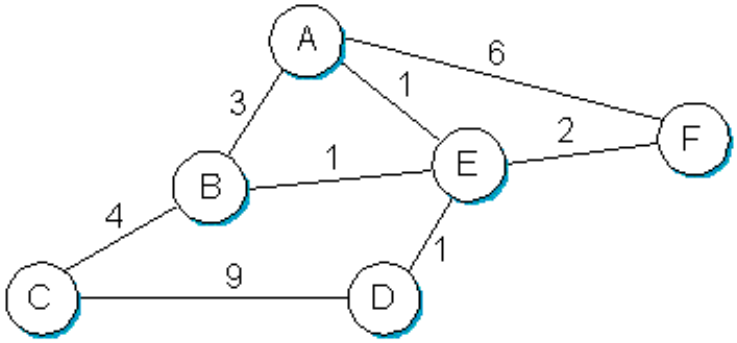
Bởi:

unknown

Giải thuật chọn đường

Giới thiệu

Vạch đường về bản chất là một bài toán trong lý thuyết đồ thị. Hình 6.4 thể hiện một đồ thị biểu diễn cho một mạng.



Mạng được biểu diễn như một đồ thị (H6.4)

Các nút trong đồ thị (được đánh dấu từ A đến F) có thể là các host, switch, router hoặc là các mạng con. Ở đây chúng ta tập trung vào một trường hợp các nút là các router. Các cạnh của đồ thị tương ứng với các đường nối kết mạng. Mỗi cạnh có một chi phí đính kèm, là thông số chỉ ra cái giá phải trả khi lưu thông trên nối kết mạng đó.

Vấn đề cơ bản của việc vạch đường là tìm ra đường đi có chi phí thấp nhất giữa hai nút mạng bất kỳ, trong đó chi phí của đường đi được tính bằng tổng chi phí khi đi qua tất cả các cạnh làm thành đường đi đó. Nếu không có một đường đi giữa hai nút, thì độ dài đường đi giữa chúng được xem như bằng vô cùng.

Mục tiêu của giải thuật chọn đường

- Xác định đướng đi nhanh chóng, chính xác.
- Khả năng thích nghi được với những thay đổi về hình trạng mạng.
- Khả năng thích nghi được với những thay đổi về tải đường truyền.
- Khả năng tránh được các nối kết bị tắt nghẽn tạm thời

- Chi phí tính toán để tìm ra được đường đi phải thấp

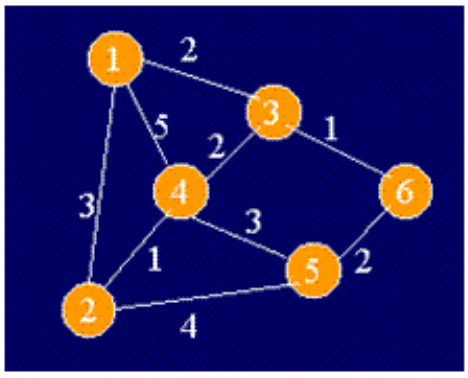
Phân loại giải thuật chọn đường

Giải thuật chọn đường có thể được phân thành những loại sau:

- Chọn đường tập trung (Centralized routing): Trong mạng có một Trung tâm điều khiển mạng (Network Control Center) chịu trách nhiệm tính toán và cập nhật thông tin về đường đi đến tất cả các điểm khác nhau trên toàn mạng cho tất cả các router.
- Chọn đường phân tán (Distributed routing): Trong hệ thống này, mỗi router phải tự tính toán tìm kiếm thông tin về các đường đi đến những điểm khác nhau trên mạng. Để làm được điều này, các router cần phải trao đổi thông tin quan lại với nhau.
- Chọn đường tĩnh (Static routing): Trong giải thuật này, các router không thể tự cập nhật thông tin về đường đi khi hình trạng mạng thay đổi. Thông thường nhà quản mạng sẽ là người cập nhật thông tin về đường đi cho router.
- Chọn đường động (Dynamic routing): Trong giải thuật này, các router sẽ tự động cập nhật lại thông tin về đường đi khi hình trạng mạng bị thay đổi.

Các giải thuật tìm đường đi tối ưu

Đường đi tối ưu từ A đến B là đường đi "ngắn" nhất trong số các đường đi có thể. Tuy nhiên khái niệm "ngắn" nhất có thể được hiểu theo nhiều ý nghĩa khác nhau tùy thuộc vào đơn vị dùng để đo chiều dài đường đi. Đối với các router, các đại lượng sau có thể được sử dụng để đo độ dài đường đi:



Mô hình hóa mạng thành đồ thị (H6.5)

- Số lượng các router trung gian phải đi qua (HOP)
- Độ trì quản trung bình của các gói tín
- Cước phí truyền tin

Mỗi giải thuật chọn đường trước tiên phải chọn cho mình đơn vị đo chiều dài đường đi.

Để xác định được đường đi tối ưu, các giải thuật chọn đường sử dụng phương pháp đồ thị để tính toán. Trước tiên, nó mô hình hóa hình trạng mạng thành một đồ thị có các đặc điểm như sau:

- Nút là các router.
- Cạnh nối liền 2 nút là đường truyền nối hai router.
- Trên mỗi cạnh có giá đó là chiều dài đường đi giữa 2 router thông qua đường truyền nối hai router .
- Chiều dài đường đi từ nút A đến nút B là tổng tất cả các giá của các cạnh nằm trên đường đi. Nếu không có đường đi giữa 2 router thì xem như giá là vô cùng.

Trên đồ thị này sẽ thực hiện việc tính toán tìm đường đi ngắn nhất.

Giải thuật tìm đường đi ngắn nhất Dijkstra

Mục đích là để tìm đường đi ngắn nhất từ một nút cho trước trên đồ thị đến các nút còn lại trên mạng.

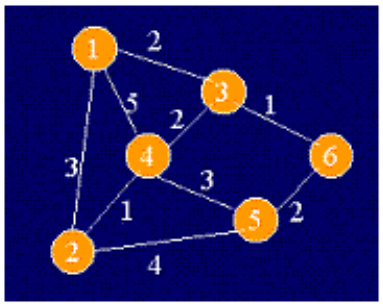
Giải thuật được mô tả như sau:

- Gọi:
  - S là nút nguồn cho trước
  - N: là tập hợp tất cả các nút đã xác định được đường đi ngắn nhất từ S.
  - Di: là độ dài đường đi ngắn nhất từ nút nguồn S đến nút i.
  - l_{ij}: là giá của cạnh nối trực tiếp nút i với nút j, sẽ là ∞nếu không có cạnh nối trực tiếp giữa i và j.
  - Pj là nút cha của của nút j.
- Bước 1: Khởi tạo
  - N={S}; Ds=0;
  - Với ∀i≠S: Di=lsi , P_i=S
- Bước 2: Tìm nút gần nhất kế tiếp
  - Tìm nút **i** ∉ N thoả Di= min (Dj) với j ∉ N
  - Thêm nút i vào N.
  - Nếu N chứa tất cả các nút của đồ thị thì dừng. Ngược lại sang Bước 3
  - Bước 3: Tính lại giá đường đi nhỏ nhất
    - Với mỗi nút j ∉ N: Tính lại Dj= min{ Dj, D**i**+ l**i**$_j$} ; Pj=i;
    - Trở lại Bước 2

Ví dụ: Cho mạng có hình trạng như đồ thị hình H6.6:

Tìm đường đi ngắn nhất từ nút 1 đến các nút còn lại.

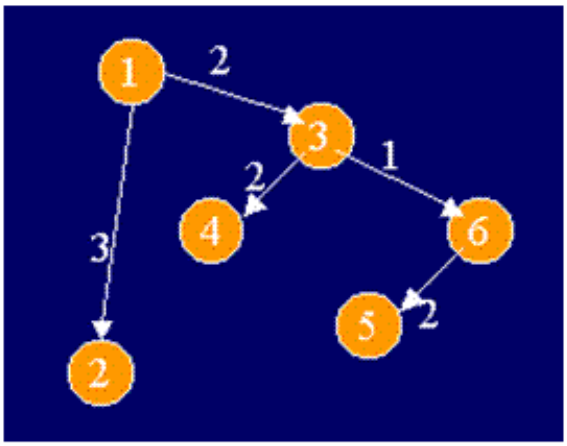
Áp dụng giải thuật ta có:



Hình trạng mạn (H6.6)

- S=1
- Các bước thực hiện được mô tả như sau:

Lần lặp	N	D_2	D_3	D_4	D_5	D_6	P_2	P_3	P_4	P_5	P_6
Khởi tạo	{1}	3	2	5	∞	∞	1	1	1	1	1
1	{1,3}	3	2	4	∞	3	1	1	3	1	3
2	{1,3,2}	3		4	7	3	1		3	2	3
3	{1,3,2,6}			4	5	3			3	6	3
4	{1,3,2,6,4}			4	5				3	6	
5	{1,3,2,6,4,5}				5					6	



Cây đường đi ngắn nhất từ nút 1 (H6.7)

Từ kết quả trên ta vẽ được cây có đường đi ngắn nhất từ nút số 1 đến các nút còn lại như hình H6.7. Từ cây đường đi ngắn nhất này, ta xác định được rằng: để đi đến các router router 4, 5, 6 , bước kế tiếp router 1 cần gởi gói tin đến là router số 3 (next hop).

Chú ý, đường ngắn nhất này chỉ đúng theo hướng từ nút số 1 về các nút còn lại và chỉ đúng cho nút số 1 mà thôi.

Thông thường giải thuật Dijkstra được sử dụng theo mô hình chọn đường tập trung. Trong đó, Trung tâm điều khiển mạng sẽ tìm cây đường đi ngắn nhất cho từng router trên mạng và từ đó xây dựng bảng chọn đường tối ưu cho tất cả các router.

Giải thuật chọn đường tối ưu Ford-Fulkerson

Mục đích của giải thuật này là để tìm đường đi ngắn nhất từ tất cả các nút đến một nút đích cho trước trên mạng.

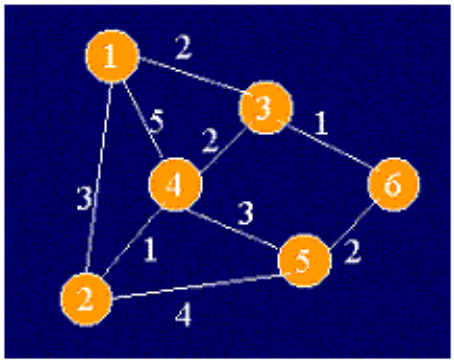
Giải thuật được mô tả như sau:

- Gọi
  - d là nút đích cho trước
  - D_i là chiều dài đường đi ngăn nhất từ nút i đến nút d.
  - C_i là nút con của nút i
- Bước 1: Khởi tạo:
  - Gán $D_d = 0$;
  - Với $\forall i \neq d$: gán $Di = \infty$; $Ci = -1$;
- Bước 2: Cập nhật giá đường đi ngắn nhất từ nút i đến nút d
  - $Di = \min\{ l_{ij} + Dj\}$ với $\forall j \neq i \Rightarrow Ci = j$;
  - Lặp lại cho đến khi không còn Di nào bị thay đổi giá trị

Ví dụ, cho sơ đồ mạng có hình trạng như đồ thị hình H6.8. Hãy tìm đường đi ngắn nhất từ nút khác trên đồ thị đến nút 6.

Áp dụng giải thuật ta có:

- d=6
- Các bước thực hiện được mô tả như sau:

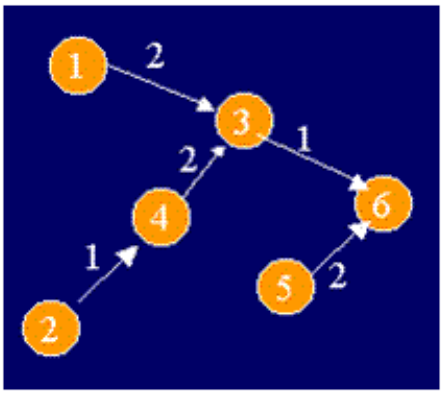


Hình trạng mạng (H6.8)

Lần lặp	D_1	D_2	D_3	D_4	D_5	C_1	C_2	C_3	C_4	C_5
Khởi tạo	∞	∞	∞	∞	∞	-1	-1	-1	-1	-1
1	∞	∞	**1**	**3**	**2**	-1	-1	**6**	**3**	**6**
2	**3**	**4**	1	3	2	**3**	**4**	6	3	6
3	3	4	1	3	2	3	4	6	3	6

Từ kết quả trên ta vẽ lại được cây đường đi ngắn nhất từ các nút khác nhau về nút đích số 6 như H6.9. Cây này cho phép xác định đường đi tối ưu từ những nút khác nhau về nút số 6. Chẳng hạn tại nút 1, để đi về nút số 6 thì bước kế tiếp phải đi là nút số 3. Tương tự, tại nút 2, để đi về nút số 6 thì bước kế tiếp phải đi là nút số 4.

Giải thuật này được sử dụng theo mô hình phân tán. Ở đó mỗi router sẽ tự tính toán, tìm cây có đường đi ngắn nhất từ các nút khác về nó. Từ đó suy ra đường đi tối ưu cho các nút khác đến nó và gởi các đường đi này đến từng nút trên mạng.

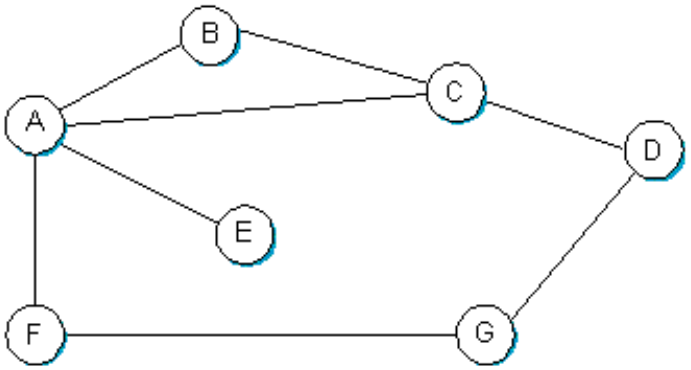


Cây đường đi ngắn nhất về nút 6 (H6.9)

Giải pháp vạch đường Vector Khoảng cách (Distance Vector)

Ý tưởng của Distance-Vector như sau: Mỗi nút thiết lập một mảng một chiều (vector) chứa khoảng cách (chi phí) từ nó đến tất cả các nút còn lại và sau đó phát vector này đến tất cả các nút láng giềng của nó. Giả thiết đầu tiên trong Distance-Vector là: mỗi nút phải biết được chi phí của các đường nối từ nó đến tất cả các nút láng giềng có đường nối kết trực tiếp với nó. Một nối kết bị đứt (down) sẽ được gán cho chi phí có giá trị vô cùng.

Để xem xét giải thuật vạch đường Distance-Vector hoạt động như thế nào, cách dễ nhất là xem xét đồ thị được cho như trong hình H6.10



Một mạng làm ví dụ trong giải thuật Distance-Vector (H6.10)

Trong ví dụ này, chi phí trên mỗi đường nối đều được đặt là 1. Chúng ta có thể biểu diễn sự hiểu biết của các nút về khoảng cách từ chúng đến các nút khác như trong bảng H6.11

Thông tin được lưu tại các nút	Khoảng cách đến nút						
	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

Các khoảng cách ban đầu được lưu tại mỗi nút (H6.11)

Chúng ta có thể xem mỗi một hàng trong bảng 6.11 như là một danh sách các khoảng cách từ một nút đến tất cả các nút khác. Khởi đầu, mỗi nút đặt giá trị 1 cho đường nối kết đến các nút láng giềng kề nó, ∞ cho các đường nối đến tất cả các nút còn lại. Do đó, lúc đầu A tin rằng nó có thể tìm đến B qua một bước nhảy (hop) và rằng nó không thể đi đến D được. Bảng vạch đường lưu tại A thể hiện những niềm tin mà A có được, ngoài ra còn lưu thêm nút kế tiếp mà A cần phải đi ra để đến một nút nào đó. Khởi đầu, bảng vạch đường của nút A trông giống như trong bảng 6.12

Đích (Destination)	Chi phí (Cost)	Nút kế tiếp (Next Hop)
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-

Bảng vạch đường khởi đầu tại nút A (H6.12)

Bước kế tiếp trong giải thuật vạch đường Distance-Vector là: mỗi nút sẽ gởi một thông điệp đến các láng giềng liền kề nó, trong thông điệp đó chứa danh sách các khoảng cách mà cá nhân nút tính được. Ví dụ, nút F bảo nút A rằng F có thể đi đến nút G với chi phí là 1; A cũng biết được rằng nó có thể đến F với chi phí là 1, vì thế A cộng các chi phí lại thành chi phí đi đến G là 2 thông qua F. Tổng chi phí là 2 này nhỏ hơn chi phí vô cùng lúc đầu, do đó A ghi lại nó có thể đi đến G thông qua F với chi phí là 2. Tương tự, A học được từ C rằng, nó có thể đi đến D thông qua C với chi phí là 2, và chi phí này nhỏ hơn chi phí cũ là vô cùng. Cùng lúc A cũng học từ C rằng, nó có thể đi đến B thông qua C với chi phí là 2, nhưng chi phí này lại lớn hơn chi phí cũ là 1, vì thế thông tin mới này bị bỏ qua.

Tại thời điểm này, A có thể cập nhật lại bảng chọn đường của nó với chi phí và nút ra kế tiếp để có thể đi đến tất cả các nút khác trong mạng. Kết quả được cho trong bảng H6.13

Đích (Destination)	Chi phí (Cost)	Nút kế tiếp (Next Hop)
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Bảng vạch đường cuối cùng tại nút A (H.6.13)

Nếu không có sự thay đổi về hình trạng mạng nào, chỉ cần vài cuộc trao đổi thông tin vạch đường giữa các nút trong mạng thì mọi nút đều có được thông tin vạch đường hoàn hảo. Quá trình đem thông tin vạch đường nhất quán đến mọi nút trong mạng được gọi là sự hội tụ (convergence). Hình 6.14 chỉ ra thông tin về chi phí cuối cùng từ một nút đến các nút khác trong mạng khi quá trình vạch đường đã hội tụ.

Thông tin được lưu tại các nút	Khoảng cách đến nút						
	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

Các khoảng cách cuối cùng được lưu tại mỗi nút (H6.14)

Nét đẹp của loại giải thuật phân tán như trên nằm ở chỗ nó cho phép tất cả các nút đạt được thông tin vạch đường mà không cần phải có sự hiện diện của bộ điều khiển trung tâm nào.

Còn có vài chi tiết làm cho giải thuật Distance-Vector hoàn hảo hơn. Thứ nhất, chú ý rằng có hai tình huống khác nhau mà tại đó một nút quyết định gởi thông tin vạch đường của mình cho các nút láng giềng kề bên. Tình huống đầu tiên là sự cập nhật theo chu kỳ (periodic update). Trong tình huống này, mỗi nút tự động gởi bản cập nhật thường xuyên, ngay cả khi không có thay đổi gì trong đó cả. Điều này giúp cho các nút khác biết được nút hiện tại đang còn sống. Vả lại việc cập nhật thường xuyên còn giúp cho các nút láng giềng có thể có được thông tin vạch đường nhanh chóng trong trường hợp thông tin của chúng bị mất. Tần số phát thông tin vạch đường đi có thể khác nhau tùy vào giải thuật, chúng thường có giá trị từ vài giây đến vài phút. Tình huống thứ hai gọi là sự cập nhật do bị kích hoạt (triggered update). Tình huống này xảy ra mỗi khi có sự thay đổi thông tin trong bảng vạch đường của nút. Nghĩa là mỗi khi bảng vạch đường có sự thay đổi, nút sẽ gởi bản cập nhật này cho các láng giềng của mình.

Bây giờ ta xem xét điều gì xảy ra khi một đường nối kết hay một nút bị hỏng. Những nút đầu tiên phát hiện ra điều này sẽ gởi thông tin vạch đường mới cho láng giềng của chúng ngay, và thông thường hệ thống sẽ ổn định với tình trạng mới một cách nhanh chóng. Còn đối với câu hỏi làm sao nút phát hiện ra sự cố, có nhiều câu trả lời khác nhau. Cách tiếp cận thứ nhất là: một nút liên tục kiểm tra đường nối tới các nút láng giềng khác bằng cách gởi các gói tin điều khiển tới chúng và kiểm tra xem nó có nhận được các gói tin trả lời hay không. Cách tiếp cận khác là: một nút phát hiện ra một đường nối kết (hay nút ở đầu kia của đường nối) gặp sự cố khi nó không nhận được thông tin vạch đường một cách định kỳ từ láng giềng của nó.

Ví dụ, xem xét điều gì sẽ xảy ra khi F phát hiện ra đường nối từ nó đến G bị hỏng. Đầu tiên, F đặt chi phí của đường nối từ nó đến A thành vô cùng và gởi thông tin này đến A. Do A đã biết là cần 2 bước để đi từ nó đến G thông qua F, A sẽ đặt lại chi phí từ nó đến G là vô cùng. Tuy nhiên, với bản cập nhật kế tiếp từ C, A phát hiện ra rằng có một

đường đi dài 2 hops từ C đến G, do đó nó sẽ cập nhật lại đường đi từ nó đến G dài 3 hops thông qua C. Và khi A quảng cáo thông tin này cho F, F lại cập nhật lại đường đi dài 4 hops đến G thông qua A.

Không may là: một số tình huống phát sinh lỗi khác lại làm cho mạng mất ổn định nghiêm trọng. Giả sử nối kết từ A đến E bị đứt. Trong những chu kỳ cập nhật sau, A thông báo đường đi từ nó đến E dài vô cùng, nhưng B và C lại quảng cáo đường đi từ chúng đến E dài 2 hops. Nếu các bản cập nhật được định thời để phát cùng lúc, B sẽ sửa lại độ dài đường đi từ nó đến E là 3 thông qua C, C sửa lại độ dài đường đi từ nó đến E là 3 thông qua B. Sau đó A lại nghe B và C quảng cáo độ dài đường đi từ chúng đến E là 3 và giả sử A chọn B là nút kế tiếp để đi đến E, nó sẽ cập nhật lại độ dài đoạn đường là 4. Đến chu kỳ kế tiếp, B nghe C nói độ dài từ C đến E là 3 nên cập nhật lại độ dài đường đi từ B đến E là 4 thông qua C, C thì làm ngược lại: sửa lại con đường từ nó đến E là 4 thông qua B. Rồi lại đến lượt A nghe B sửa lại độ dài từ A đến E là 5 thông qua B. Sự thể sẽ tiếp diễn cho đến khi các độ dài tăng đến một số có thể coi là vô cùng. Nhưng tại thời điểm này, không nút nào biết là E không thể đến được, và các bảng vạch đường trong mạng luôn không ổn định. Tình huống này được gọi là vấn đề "*đếm tới vô cùng*" (count-to-infinity problem).

Có vài giải pháp giải quyết một phần vấn đề "*đếm tới vô cùng"*. Giải pháp thứ nhất là dùng một số khá nhỏ để coi như gần bằng vô cùng. Ví dụ như chúng ra có thể quyết định số lượng bước nhảy (hop) tối đa để đi qua một mạng là không quá 16, và do đó ta chọn 16 là số gần bằng vô cùng. Con số này ít ra cũng giới hạn được thời gian mà các nút có thể phải bỏ ra để đếm tới vô cùng. Tuy nhiên giải pháp này có thể gặp vấn đề nếu một số nút mạng được chia tách và mạng có thể cần nhiều hơn 16 bước nhảy để duyệt hết nó.

Một kỹ thuật khác dùng để cải thiện thời gian dùng để ổn định hóa mạng được gọi là kỹ thật "*chia tầm nhìn"* (split horizon). Ý tưởng là: khi một nút gởi một bảng cập nhật vạch đường cho các láng giềng của nó, nó sẽ không gởi những thông tin vạch đường mà nó đã nhận từ một nút láng giềng ngược lại chính nút láng giềng đó. Ví dụ như nếu B có một đường đi (E, 2, A) trong bảng vạch đương của nó, B chắc hẳn phải biết rằng nó học con đường này từ A, vì thế mỗi khi B gởi thông tin cập nhật cho A nó sẽ không gởi con đường (E, 2) trong đó. Tuy nhiên giải pháp này chỉ tốt khi nó xoay quanh 2 nút mà thôi.

Giải pháp chọn đường "Trạng thái nối kết" (Link State)

Vạch đường kiểu Link-state là một ví dụ thứ hai trong họ giao thức vạch đường bên trong một miền. Giả thiết đầu tiên trong Link-state cũng khá giống trong Distance-vector: Mỗi nút được giả định có khả năng tìm ra trạng thái của đường nối nó đến các nút láng giềng và chi phí trên mỗi đường nối đó. Nhắc lại lần nữa: chúng ta muốn cung cấp cho mỗi nút đủ thông tin để cho phép nó tìm ra đường đi có chi phí thấp nhất đến bất kỳ đích nào. Ý tưởng nền tảng đằng sau các giao thức kiểu Link-state là rất đơn giản:

Mọi nút đều biết đường đi đến các nút láng giềng kề bên chúng và nếu chúng ta đảm bảo rằng tổng các kiến thức này được phân phối cho mọi nút thì mỗi nút sẽ có đủ hiểu biết về mạng để dựng lên một bản đồ hoàn chỉnh của mạng. Giải thuật Link-state dựa trên hai kỹ thuật: sự phân phối một cách tin cậy thông tin về trạng thái các đường nối kết; và sự tính toán các đường đi từ kiến thức tổng hợp về trạng thái các đường nối kết.

Làm ngập một cách tin cậy (Reliable Flooding)

"Làm ngập" là quá trình thực hiện cam kết: "tất cả các nút tham gia vào giao thức vạch đường đều nhận được thông tin về trạng thái nối kết từ tất cả các nút khác". Như khái niệm "làm ngập" ám chỉ, ý tưởng cơ sở của Link-state là cho một nút phát thông tin về trạng thái nối kết của nó với mọi nút láng giềng liền kề, đến lượt mỗi nút nhận được thông tin trên lại chuyển phát thông tin đó ra các nút láng giềng của nó. Tiến trình này cứ tiếp diễn cho đến khi thông tin đến được mọi nút trong mạng.

Cụ thể hơn, mỗi nút tạo ra gói tin cập nhật, còn được gọi là gói tin trạng thái nối kết (link-state packet – LSP), chứa những thông tin sau:

- ID của nút đã tạo ra LSP
- Một danh sách các nút láng giềng có đường nối trực tiếp tới nút đó, cùng với chi phí của đường nối đến mỗi nút.
- Một số thứ tự
- Thời gian sống (time to live) của gói tin này

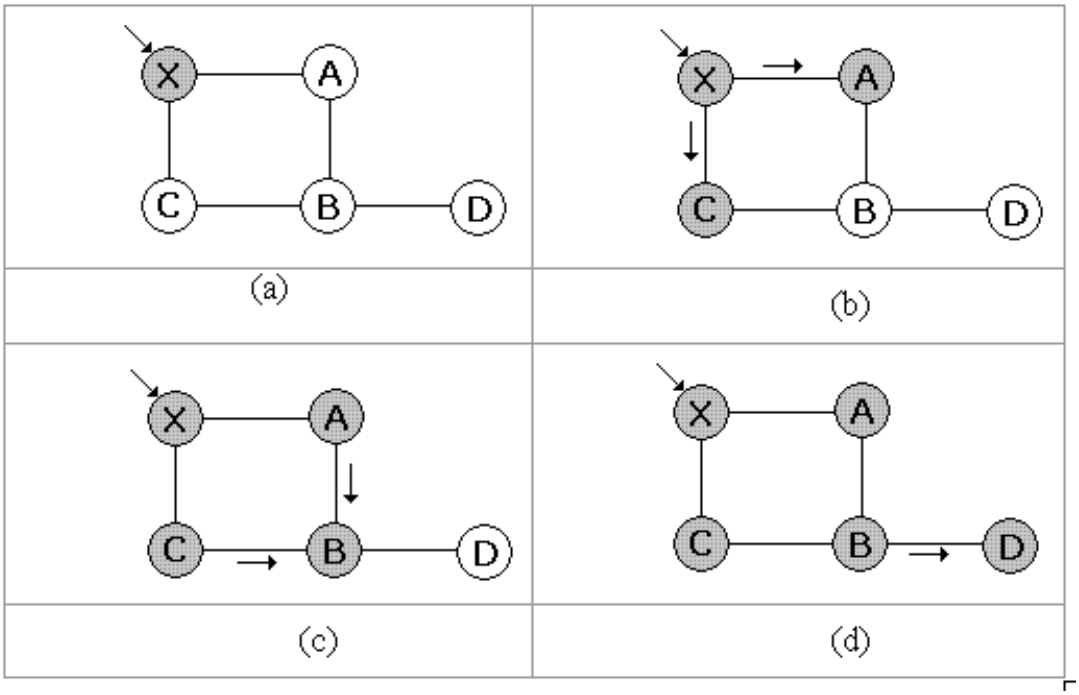
Hai mục đầu là cần thiết cho việc tính toán chọn đường; hai mục sau cùng được sử dụng để giúp cho quá trình làm ngập thật chắc. Tính tin cậy ở đây đòi hỏi việc đảm bảo các nút trong mạng có được thông tin có phiên bản mới nhất, do có nhiều LSP trái ngược nhau từ một nút được phát lên mạng. Đảm bảo việc làm ngập có thể tin cậy được là một việc khó (Ví dụ, một phiên bản cũ của giao thức vạch đường link-state trong ARPANET đã làm cho mạng này bị tê liệt vào năm 1981).

Việc làm ngập được thực hiện như sau: Đầu tiên, việc truyền các LSP giữa các nút kề nhau được bảo đảm tính tin cậy bằng cách sử dụng cơ chế báo nhận (acknowledgement) và làm lại khi bị lỗi (retransmission) giống như ở tầng liên kết dữ liệu. Tuy nhiên, cần thực hiện thêm một số bước để đảm bảo việc phát một LSP từ một nút đến tất cả các nút khác trong mạng là đáng tin cậy.

Giả sử nút X nhận được một phiên bản LSP có nguồn gốc từ nút Y nào đó. Chú ý rằng nút Y có thể là bất kỳ router nào ở trong cùng một miền với X. X kiểm tra xem nó đã có bất kỳ phiên bản LSP nào từ Y không. Nếu không, nó sẽ lưu LSP này. Nếu có, X sẽ so sánh hai số thứ tự trong hai LSP. Nếu LSP mới đến có số thứ tự lớn hơn số thứ tự của LSP có sẵn, X cho rằng LSP mới đến là mới hơn, và do đó X sẽ thay LSP cũ bằng phiên bản mới này. Ngược lại, với một số thứ tự nhỏ hơn (hoặc bằng), LSP mới đến sẽ bị coi

là cũ hơn cái đang có sẵn (hoặc ít ra là không mới hơn), và vì thế nó sẽ bị bỏ qua, không cần phải làm gì thêm. Nếu LSP mới đến là cái mới hơn, nút X sẽ gởi một phiên bản của LSP này ra tất cả các nút láng giềng liền kề nó ngoại trừ nút láng giềng vừa gởi cho nó phiên bản LSP mới vừa đề cập. Đến phiên các nút láng giềng của X lại xoay qua phát tán LSP mới này sang các nút láng giềng khác. Việc "LSP không được gởi ngược lại nút vừa phát ra nó" sẽ giúp dẫn đến điểm dừng của quá trình phát tán LSP này. Sự phát tán dây chuyền có điểm dừng này sẽ đảm bảo cho việc đem phiên bản LSP mới nhất đến tất cả các nút trong mạng.

Hình H6.15 thể hiện một LSP được dùng làm ngập một mạng nhỏ. Hình (a) thể hiện X nhận được một LSP mới; (b) X đẩy LSP mới ra A và C; (c) A và C đẩy LSP qua B; (d) B đẩy LSP qua D và quá trình làm ngập kết thúc.



Việc làm ngập mạng với các gói tin LSP (H6.15)

Cũng giống như trong giải thuật Distance-Vector, sẽ có hai tình huống mà một nút quyết định gởi LSP đến các nút láng giềng: gởi một cách định kỳ hoặc gởi do bị kích hoạt.

Một trong những ưu tiên hàng đầu của cơ chế làm ngập (flooding) là phải đảm bảo đem thông tin mới nhất đến mọi nút trong mạng càng nhanh càng tốt và các thông tin cũ phải được rút ra không cho lưu thông trên mạng nữa. Thêm nữa, rất là lý tưởng nếu ta có thể giảm thiểu lượng thông tin vạch đường lưu chuyển trên mạng – một kiểu phí tổn theo cách nhìn của nhiều người.

Một phương pháp cần thiết để giảm thiểu phí tổn dành cho việc vạch đường là tránh gởi các LSP trừ trường hợp hết sức cần thiết. Điều này có thể thực hiện được bằng cách sử

dụng các bộ định thời (timer) có giá trị rất lớn – thường là kéo dài hàng giờ - dùng để định kỳ phát các LSP.

Còn để đảm bảo rằng thông tin cũ phải được thay thế bởi thông tin mới, các LSP sẽ mang số thứ tự. Mỗi khi một nút phát LSP mới, nó sẽ tăng số thứ tự lên 1. Không giống như hầu hết các giao thức khác, số thứ tự trong LSP sẽ không được đếm xoay vòng (modulo), vì thế trường chứa số này phải đủ lớn (ví dụ như 64 bit). Nếu một nút bị chết (down) và sau đó được khởi động lại, nó sẽ khởi động trường số thứ tự lại bằng 0. Nếu một nút bị chết quá lâu, tất cả các LSP của nút đó sẽ bị mãn kỳ (timed out); ngoài ra, nếu cuối cùng nút này lại nhận được LSP của chính nó với số thứ tự lớn hơn bản gốc, nút có thể lấy số lớn hơn làm số thứ tự mới.

Các LSP cũng mang theo thời gian sống của nó (Time to live - TTL). Điều này dùng để đảm bảo các LSP cũ rút cuộc cũng bị xóa khỏi mạng. Một nút luôn luôn giảm trường TTL của một LSP mới đến nó đi 1 trước khi chuyển LSP này ra các nút láng giềng. Khi trường TTL còn 0, nút phát lại LSP với TTL = 0, điều đó sẽ được thông dịch bởi tất cả các nút trong mạng như là tín hiệu cho phép xóa LSP đó.

Tính toán chọn đường trong Link State

Khi một nút có một phiên bản LSP từ tất cả các nút khác trong mạng, nó đã có thể tính toán ra bản đồ hoàn chỉnh cho hình thái của mạng, và từ bản đồ này nút có thể quyết định con đường tốt nhất đến tất cả các nút còn lại trong mạng. Giải pháp chọn đường chính là giải thuật tìm đường đi ngắn nhất Dijkstra.

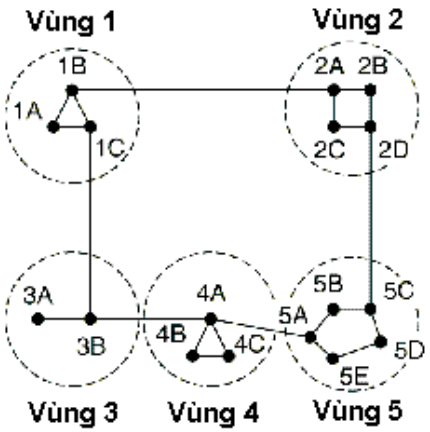
Vạch đường phân cấp (Hierarchical Routing)

Khi mạng tăng kích thước, kích thước bảng vạch đường của các router tăng theo. Không chỉ bộ nhớ của router bị tiêu hao quá nhiều cho việc trữ các bảng vạch đường, mà CPU còn phải tốn nhiều thời gian để quét bộ nhớ và cũng cần nhiều băng thông hơn để truyền những thông tin chọn đường này. Rồi cũng sẽ đến lúc mạng máy tính phát triển đến mức không một router nào có đủ khả năng trữ một đầu mục thông tin về một router khác, vì thế việc vạch đường phải phát triển theo đường hướng khác: vạch đường phân cấp.

Khi việc vạch đường phân cấp được áp dụng, các router được chia thành những vùng (domain). Trong mỗi vùng, mỗi router biết cách vạch đường cho các gói tin đi đến được mọi đích trong nội vùng của nó, nhưng không biết gì về cấu trúc bên trong của các vùng khác. Khi nhiều vùng được nối kết với nhau, đương nhiên mỗi vùng được công nhận tính độc lập để giải phóng các router trong các vùng đó khỏi việc phải tìm hiểu hình trạng của các vùng khác.

Với những mạng thật lớn, kiến trúc phân cấp hai mức có thể sẽ không đủ; có thể cần phải nhóm các vùng lại thành liên vùng, nhóm các liên vùng thành khu vực...

Hình H6.16 thể hiện một mạng được vạch đường phân cấp gồm hai mức có năm vùng. Bảng vạch đường đầy đủ của router A gồm có 17 mục từ như trong hình H6.16(b). Khi vạch đường được thực hiện theo kiểu phân cấp, bảng vạch đường của A giống như bảng H6.16(c), có mọi mục từ chỉ đến các router cục bộ giống như trước, tuy nhiên các mục từ chỉ đến các vùng khác lại được cô đặc lại thành một router. Do tỉ lệ các router trong các vùng tăng, vì thế cách làm này giúp rút ngắn bảng vạch đường.



(a)

Bảng vạch đường đầy đủ của nút 1A

Đích	Lối ra	Chi phí
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Bảng vạch đường phân cấp của host 1A

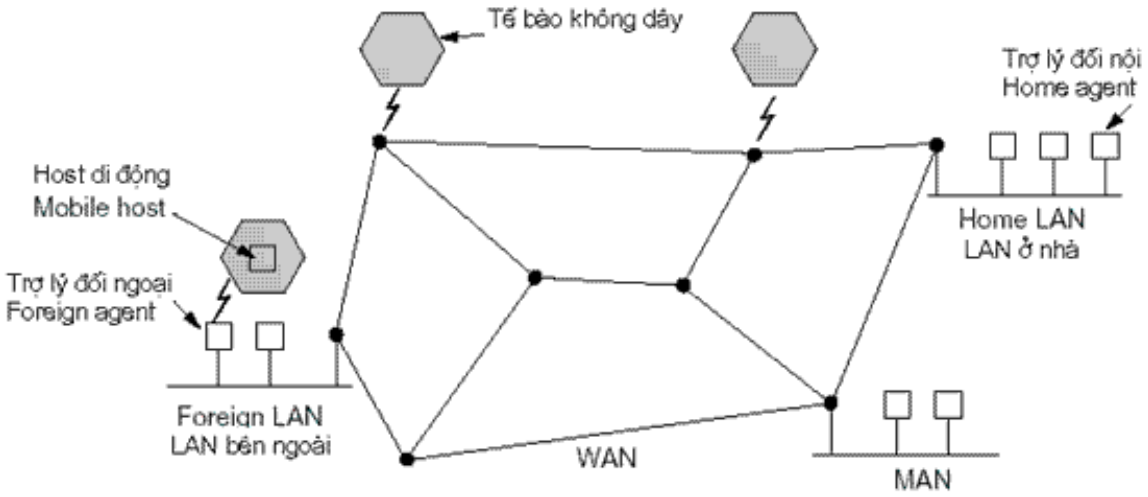
Đích	Lối ra	Chi phí
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Vạch đường phân cấp (H6.16)

Vạch đường trong mạng di động

Ngày nay, hàng triệu người đang sở hữu máy tính xách tay, và thông thường họ muốn đọc email cũng như truy xuất các hệ thống tập tin cho dù họ đang ở bất kỳ nơi nào trên thế giới. Việc sử dụng các host di động này dẫn đến một vấn đề phức tạp mới: để vạch đường cho gói tin đến host di động, trước tiên phải tìm ra nó đã. Chủ đề về tích hợp các host di động lại thành một mạng là tương đối mới, tuy vậy trong phần này chúng ta sẽ phác thảo ra một số vấn đề phát sinh và chỉ ra các giải pháp khả thi.



Mô hình mạng có hệ thống không dây (H6.17)

Mô hình mạng mà các nhà thiết kế thường sử dụng được chỉ ra trong hình H6.17. Ở đây chúng ta có một mạng WAN bao gồm vài router và host. Mạng WAN được dùng để nối kết các mạng LAN, MAN, các tế bào mạng không dây (Wireless cell).

Các host không bao giờ di chuyển được gọi là cố định, chúng được nối vào mạng bởi các đường cáp đồng hoặc quang. Ngược lại, chúng ta sẽ phân biệt hai loại host khác: loại di cư (migratory host) và loại lang thang (roaming host). Loại host di cư về bản chất là host cố định, nhưng chúng thỉnh thoảng lại chuyển từ địa bàn (site) này đến địa bàn mạng kia, và chúng chỉ có thể sử dụng mạng mới khi được nối kết vật lý vào đấy. Loại host lang thang thực chất vừa chạy vừa tính toán, nó muốn duy trì các nối kết mạng ngay cả khi đang di chuyển. Chúng ta sẽ sử dụng thuật ngữ "host di động" để ám chỉ hai loại di động vừa nói đến, tức là chúng đã đi khỏi nhưng lại muốn duy trì liên lạc về nhà.

Tất cả các host được giả sử đều có vị trí mạng nhà (home location) và vị trí này không bao giờ thay đổi. Các host cũng có địa chỉ lâu dài tại nhà (home address) và địa chỉ này có thể được dùng để xác định vị trí mạng nhà của nó, cũng giống như số điện thoại 84-071-831301 chỉ ra số đó ở Việt Nam (mã 084), thành phố Cần Thơ (mã 071). Mục tiêu của việc vạch đường trong hệ thống có các host di động là phải đảm bảo có thể gởi được gói tin đến host di động sử dụng địa chỉ tại nhà của nó và làm cho các gói tin đến được host di động một cách hiệu quả cho dù host này có ở đâu đi nữa.

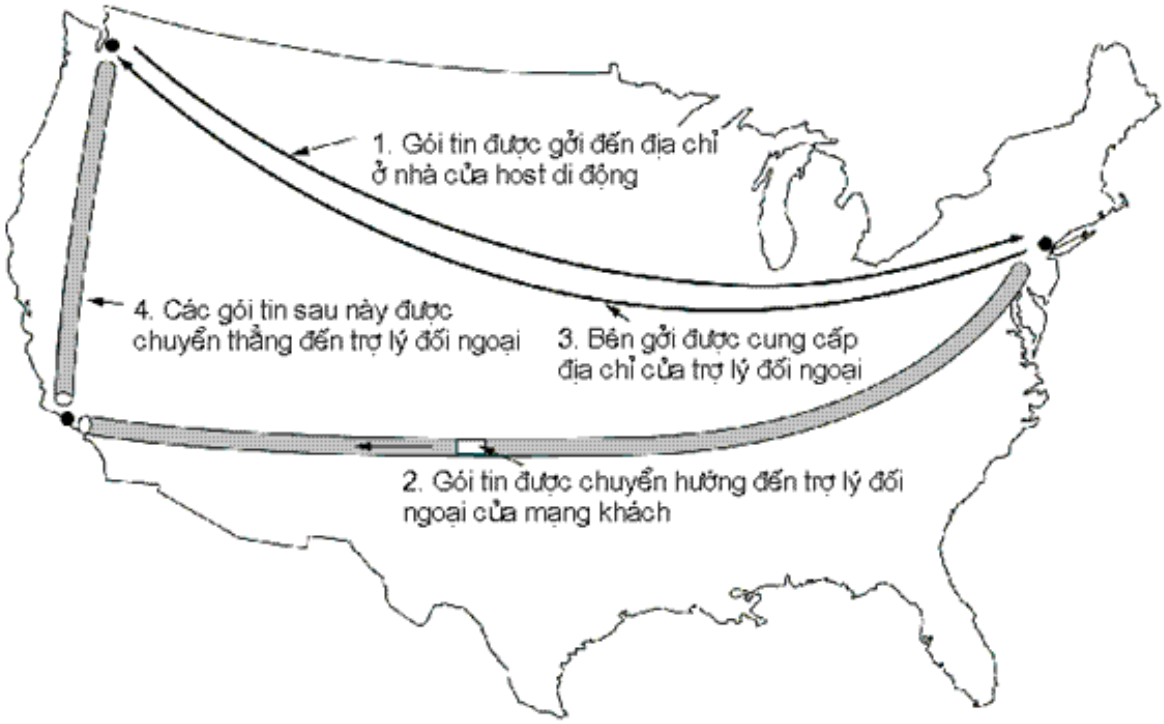
Trong mô hình ở hình H6.17, WAN được chia thành các đơn vị nhỏ, ở đây chúng ta gọi là khu vực (area), thường là LAN hoặc tế bào mạng không dây. Mỗi khu vực có một hoặc nhiều trợ lý đối ngoại (foreign agent - FA) – đó là những tiến trình làm nhiệm vụ theo dõi các host khách đang viếng thăm khu vực của mình. Thêm vào đó, mỗi khu vực còn có một trợ lý đối nội (home agent - HA), làm nhiệm vụ theo dõi những host có nhà nằm trong khu vực nhưng hiện đang viếng thăm khu vực khác.

Khi một host đi vào một khu vực mới (có thể là host này muốn thường trú trong mạng LAN mới hoặc chỉ đi ngang cell này thôi), nó phải đăng ký với trợ lý đối ngoại ở đó. Thủ tục đăng ký diễn ra như sau:

1. Theo chu kỳ, mỗi trợ lý đối ngoại sẽ phát ra những thông điệp thông báo sự hiện diện của nó cùng với địa chỉ. Một host mới tới sẽ chờ lắng nghe thông báo này. Nếu host cảm thấy nó đã chờ lâu nhưng không nhận được thông báo, host có thể tự phát câu hỏi: Có bất kỳ trợ lý đối ngoại nào ở đây không?
2. Host di động đăng ký với trợ lý đối ngoại, cung cấp thông tin về địa chỉ ở nhà, địa chỉ MAC và một số thông tin về an ninh khác.
3. Trợ lý đối ngoại liên hệ với trợ lý đối nội ở nhà của host đó và nói: Một host của ông đang ở đây. Thông điệp mà trợ lý đối ngoại gởi cho trợ lý đối nội bên kia chứa địa chỉ mạng của trợ lý đối ngoại đó. Thông điệp này còn chứa thông tin an ninh dùng để thuyết phục trợ lý đối nội bên kia rằng host di động của nó thực sự đang ở đó.
4. Trợ lý đối nội bên kia kiểm tra thông tin an ninh, trong đó có một tem thời gian, để chứng tỏ được rằng tem này vừa được tạo ra trong vòng vài giây. Và nếu kết quả kiểm tra là tốt đẹp, nó sẽ bảo trợ lý đối ngoại bên kia tiến hành làm việc.
5. Khi trợ lý đối ngoại nhận được sự chấp thuận của trợ lý đối nội bên kia, nó tạo ra một đầu mục trong các bảng quản lý và thông báo cho host di động rằng: Bạn đã đăng ký thành công.

Lý tưởng nhất là khi một host di động rời khỏi một cell, nó phải thông báo với trợ lý đối ngoại ở đó để xóa đăng ký. Nhưng đa phần người sử dụng thường tắt máy ngay khi sử dụng xong.

Khi một gói tin được gởi đến một host di động, đầu tiên gói tin đó được gởi đến mạng LAN nhà của host đó (bước 1 trong hình H6.18.



Vạch đường trong mạng di động (H6.18)

Bên gởi, ví dụ đang ở Tiền Giang, gởi gói tin đến mạng nhà của host di động ở Cần Thơ. Giả sử host di động đang ở Đồng Tháp. Trợ lý đối nội ở Cần Thơ tìm ra được địa chỉ tạm thời của host di động, đóng gói gói tin đó và chuyển cho trợ lý đối ngoại của mạng ở Đồng Tháp (bước 2). Đến phiên trợ lý đối ngoại ở Đồng Tháp mở gói gói tin đó và phát cho host di động thông tin dưới dạng khung thông tin ở mức liên kết dữ liệu.

Sau đó trợ lý đối ngoại ở Đồng Tháp sẽ bảo bên gởi ở Tiền Giang hãy đóng gói và gởi gói tin trực tiếp đến Đồng Tháp (bước 3). Từ đó trở về sau, nhưng gói tin mà bên gởi muốn gởi cho host di động được gởi trực tiếp đến trợ lý đối ngoại tại Đồng Tháp, rồi được trợ lý đối ngoại phát trực tiếp đến host (bước 4).