Evolved Fault Tolerant Routing, Broadcasting and Load Balancing in Hyper-DeBruijn-Aster Network

Nguyễn Hồng Thái, Phạm Minh Trí, Nguyễn Chí Ngọc
Dept. of Telecommunication
Hô Chi Minh City of University Technology, South Vietnam
nhthai2005@gmail.com, minhtriphamlth@yahoo.com, ncngoc@hcmut.edu.vn

Abstract

In this paper, we investigate the properties of Hyper-DeBruijn-Aster Networks in order to evolve the Fault Tolerant Routing, Broadcasting and Load Balancing, besides, we study the new concept of Shortest Routing so that we can apply it into the framework multiprocessors network.

By giving new method of routing basing on features of duplex mapping in Hyper-DeBruijn-Aster Network, we can achieve more possibilties in routing than other recent researchers investigated, that refers to increase the perfomances of Fault Tolerant and Broadcasting. Our paper also focus on the proposed Shortest Routing algorithm in Hyper-DeBruijn-Aster network.

1. Giới thiệu:

Trong bài viết này, đầu tiên chúng tôi trình bày một họ của mạng gọi là Hyper-DeBruijn-Aster network và sau đó là những vấn đề truyền thông được nghiên cứu. Các tính chất của mạng De-Bruijn trình bày rằng mạng này là điển hình tốt cho thế hệ công nghệ tương lai, sau hypercube [7]. Nó hỗ trợ nhiều ứng dụng quan trọng từ việc tạo ra mạng De-Bruijn [9,11]. Điểm mạnh của Hypercube là khá lớn. Còn điểm mạnh trong k-cube là degree và diameter là không có mối liên hệ với nhau. Mà Hypercube thừa hưởng từ k-cube nên tận dụng được những điểm mạnh đó [7]. Và một điểm mạnh nữa mà chúng tôi tìm thấy đó là nếu kết hợp giữa De-Bruijn và Hypercube trong shortest routing thì rất là hiệu quả. Vấn đề này chúng tôi sẽ trình bày trong phần Hyper-DeBruijn-Aster graph.

Việc định tuyến trong De-Bruijn Graph được điều tra bởi Samantham, Liu và Mao [1,2,6], tuy nhiên những giải thuật của họ không thể đạt được với shortest path nếu như có node bị hỏng trên đường đi [7]. Những vấn đề về broadcasting trong De-Bruijn graph cũng vừa được điều tra bởi Esfahanian, Ganesan, Ohring [8,10]. Tuy nhiên, những giải thuật của họ chỉ làm việc ở một mạng binary de Bruijn. Và điều đó được tác giả Nguyễn Chí Ngọc trong bài viết [7], cải thiện khá tốt. Tuy nhiên, tác giả Ngọc vẫn chưa tối ưu trong vấn đề *fault tolerant routing và broadcasting*. Việc tối ưu đó, chúng tôi làm bằng cách kết hợp Hypercube và De-Bruijn graph tạo ra một graph gọi là Hyper-DeBruijn-Aster graph (nó có thể hiểu là Hypercube-DeBruijn-Asterisk và chúng tôi ký hiệu nó là *HD**). HD* là sự kết hợp những ưu điểm của Hypercube, De-Brujin với miền bit mở rộng thay vì là binary de Bruijn graph. Và *HD** còn có một ưu điểm khác đó là có thể thực hiện được *shortest routing*. Còn đối với bài viết [3] của Elango Ganesan, Dhiraj K Pradhan, các tác giả đã ứng dụng được Hyper-DeBruijn graph. Tuy nhiên, các tác giả chỉ dừng lại ở Binary de Bruijn graph và giải thuật DeadLock-Free routing. Với giải thuật của chúng tôi, đọc giả sẽ dễ hiểu hơn vì nó dựa trên kiến thức và nền tảng của hypercube, deBruijn và giải thuật *shorest rounting*. Đây là những mảng mà người độc phần lớn đã biết. Còn trong bài viết [5] của Wei Shi và Pradip K Srimani, tác

giả đã đưa ra một giải thuật định tuyến khá hay là kết hợp giữa Hypercube và Butterfly để tạo ra một graph mang tên là Hyper-Butterfly và dùng giải thuật định tuyến theo đường đi ngắn nhất. Trong bài viết tác giả cho là Hyper-DeBruijn graph là không có quy tắc, chưa tối ưu trong vấn đề *fault tolerant routing* và nó chưa tối ưu mà còn lại phức tạp. Tuy nhiên, những khuyết điểm đó chúng tôi sẽ khắc phục trong bài viết này. Mặt khác, với Hyper-DeBruijn-Aster và với giải thuật *Shortest Routing* mà chúng tôi sẽ trình bày dưới đây, thì mạng Hyper-DeBruijn-Aster còn có thể làm nhiệm vụ *cân bằng tải* và *xử lý động bộ*, điều này đóng góp rất nhiều vào những giải thuật *xử lý song song* và *phân tán dữ liệu*. Do đó có thể nói rằng, nó cải tiến rất nhiều trong việc tăng hiệu suất mạng truyền thông đa xử lý.

2. Hyper-DeBruijn-Aster graph

2.1 Hypercube Graph H_n và De Bruijn Graph B_n

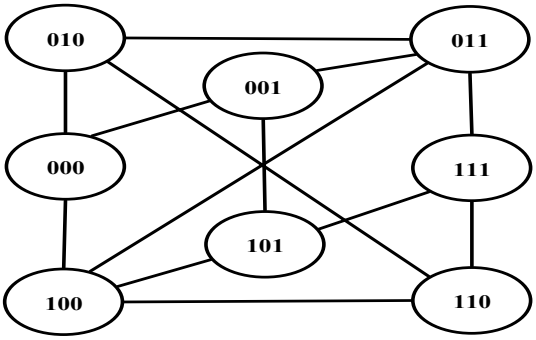
Hypercube Graph bậc m, ***H(m)*** bao gồm tập hợp các node Z_2^m. Có một cạnh nằm giữa 2 đỉnh nếu và chỉ nếu các nhãn node của chúng khác nhau chính xác 1 bit.

Binary de Bruijn Graph bậc n, ***B(n)*** bao gồm các tập đỉnh Z_2^n.

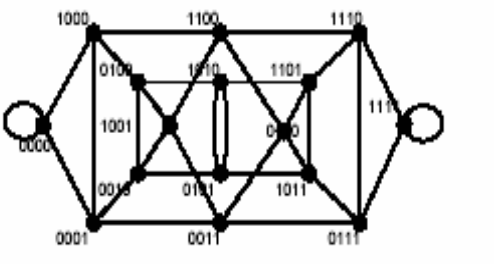
Cho $\alpha,\beta \in Z_2$ và $x \in Z_2^{n-2}$, mỗi node có dạng là $\alpha x \beta$, được kết nối tới:

- $x\beta\alpha$ thông qua shuffle arc
- $x\beta\bar{\alpha}$ thông qua shuffle-exchange arc
- $\beta\alpha x$ thông qua inverse-shuffle arc
- $\bar{\beta}\alpha x$ thông qua inverse-shuffle-exchange arc

Tương tự vậy, nếu ta thay đổi cho $\alpha,\beta \in Z_d$ và $x \in Z_d^{n-2}$ (với $d \geq 2$), thì rõ ràng ta có thể mở rộng số liên kết trên mỗi node cũng như mở rộng số node lân cận. Và điều này có thể cải thiện vấn đề *Fault Tolerant Routing và Broadcasting* trong mạng De-Bruijn.



Hình 1: Hypercube graph H(3)



Hình 2: De Bruijn graph D(2,2)

Nhận xét 1: *D(d,n) là một graph vô hướng có quy tắc đối xứng, có degree là 2d, diameter là n và có số node là* $N(D_{dn}) = d^n$ *[7].*

Nhận xét 2: *H(m) là một graph vô hướng có quy tắc đối xứng, có degree là m, diameter làm m và có số node là* $N(D_m) = 2^m$ *[5].*

2.2 Hyper-DeBruijn-Aster Graph *HD*(m,d,n)*

Với sự mở rộng DeBruijn graph ở trên kết hợp với mạng hypercube, chúng tôi đã tìm ra một họ mạng mới, đó là *Hyper-De-Bruijn-Aster bậc (m,n)* và *hệ số mở rộng DeBruijn là d*, ký hiệu là *HD*(m,d,n)*. *HD*(m,d,n)* là một graph tích *H(m)xD(d,n)*.

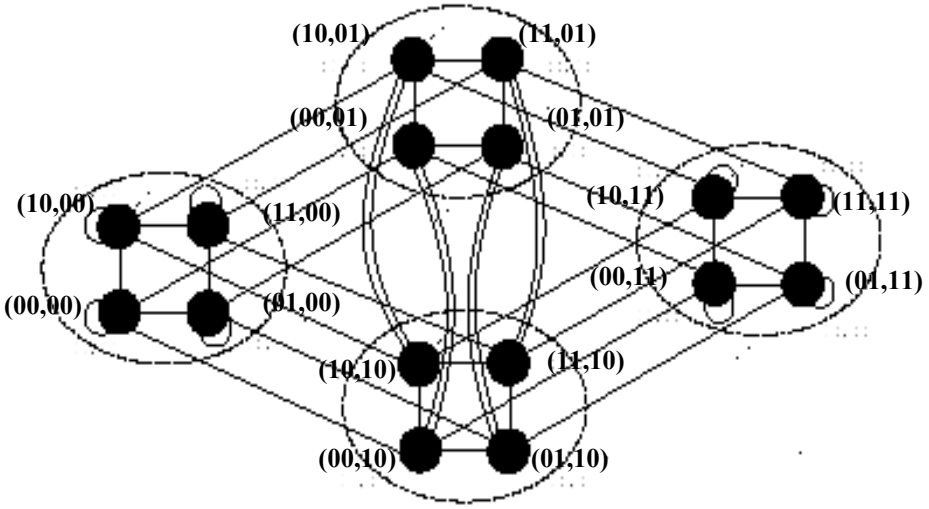
Ta dễ dàng thấy rằng một node $\langle x_{m-1}x_{m-2}\ldots x_0 y_{n-1}y_{m-2}\ldots y_0\rangle$ nối đến những node dưới đây đối với phần lân cận deBruijn:

$\langle x_{m-1}x_{m-2}\ldots x_0 \alpha y_{n-1}y_{m-2}\ldots y_1\rangle$

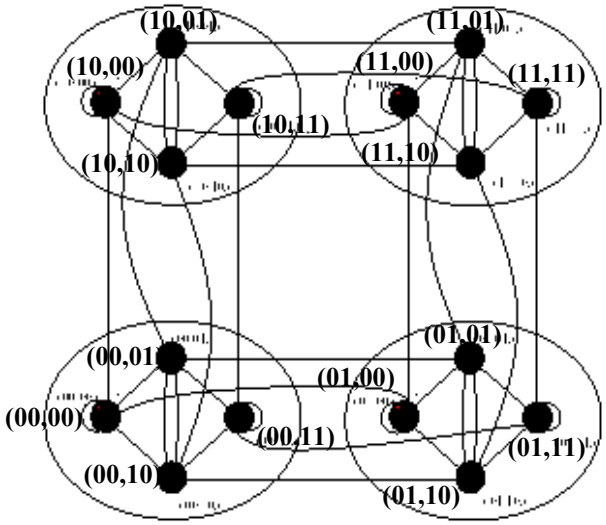
$\langle x_{m-1}x_{m-2}\ldots x_0 y_{n-1}y_{m-2}\ldots y_1\alpha\rangle$, với $\alpha \in Z_d$

Và những node dưới đây đối với phần lân cận Hypercube

$\langle x_{m-1}x_{m-2} \dots \bar{x}_i x_i x_{i-1} \dots x_0 y_{n-1} y_{m-2} \dots y_0 \rangle$ với $0 \le i \le m-1$



Hình 3: Trình bày *HD*(2,2,2)*



Hình 4: Một dạng trình bày khác của *HD*(2,2,2)*

Nhận xét 3: *HD*(m,d,n) là một graph vô hướng có quy tắc đối xứng, có degree là m+2d, diameter là m + n và có số node là* $N(HD^*) = 2^m x d^n$.

Nhận xét 4: *Địa chỉ HD* gồm 2 phần: phần đầu của địa chỉ HD*, đó chính là phần địa chỉ của Hypercube graph còn phần sau của địa chỉ HD*, đó chính là phần địa chỉ của DeBruijn graph.*

Chứng minh 1: Vì Hypercube graph và De-Bruijn graph là 2 loại graph đối xứng (nhận xét 1 và 2). Một graph HD*(m,d,n), ta có thể phân ra 2 phần, đó là Hypercube graph và De-Bruijn graph. Và mỗi phần của Hypercube bao gồm một DeBruijn graph hay với cách phân khác thì mỗi phần của DeBruijn graph, nó bao gồm một Hypercube graph. Do đó, HD* graph cũng là graph đối xứng (hình 3,4).

Chứng minh 2: Từ phần chứng minh 1. Ta thấy nếu một graph *HD*(m,d,n)*, thì với cách phân chia theo hình 4, ta có m khối, đó chính là m hypercube graph. Trong mỗi khối đó có d^n node. Do đó, số node của *HD*(m,d,n)* là $2^m x d^n$ hay có thể hiểu một cách dàng là $N(HD^*) = N(H_m)xN(D_{nd})$.

Chứng minh 3:degree của *HD** : *m + 2d.*

Định nghĩa 1: *degree $d_G(v)$ của một đỉnh v trong graph G là số cạnh của graph G hướng tới v, mỗi lần tính vòng tính là 2 cạnh [4] trang 10, như ở node 0000 và 1111 ở hình 2.*

Degree $d(H_m) = m$ (nhận xét 1) và $d(D(d,n)) = 2xd$ (nhận xét 2). Và trên mỗi node sẽ có những liên kết đến một số node của phần DeBruijn graph cũng như phần Hypercube vừa được trình bày ở trên.

Nên $d(HD^*) = d(H_m) + d(D(d,n))$ hay $d(HD^*) = m + 2d$.

Chứng minh 4: Diameter của *HD**: *m+ n*.

Định nghĩa 2: *diameter của graph G D_G là khoảng cách lớn nhất giữa 2 đỉnh của graph G [4] trang 14.*

Với đặc tính của *HD** graph thì một node v(h,d) đi đến node v(h',d'). Thì tiến trình định tuyến sẽ từ *v(h,d) → v(h',d) → v(h',d')* hoặc từ *v(h,d) → v(h,d') → v(h',d')*.

Do đó, $D_G(HD^*) = D_G(Hm) + D_G(D(d,n))$ hay $D_G(HD^*) = m + n$.

Thông số	Node	Degree	Diameter	Fault-Tolerance
Hypercube H(m+n)	2^{m+n}	m+n	M+n	m+n
Butterfly B(m+n)	$(m+n)x2^{m+n}$	4	3n/2	4
De Bruijn D(d,m+n)	d^{m+n}	2xd	M+n	2xd-2
HyperButterfly HB(m,n)	$nx2^{m+n}$	m+4	m+3n/2	m+4
Hyper-DeBruijn-Aster HD*(m,d,n)	$2^m x d^n$	m+2xd	M+n	m+2xd-2

3. Shortest Routing và Diameter trong *HD*(m,d,n)*

Việc định tuyến theo phương pháp gọi là *Shortest Routing* từ điểm đến điểm trong *HD*(m,d,n)* là cực kỳ đơn giản và dễ hiểu. Như ở phần *nhận xét 4* và phần *chứng minh 4* vừa thảo luận ở trên thì việc định tuyến từ 2 node bất kỳ v(h,d) đến v(h',d') dùng *Shortest Routing* có thể được thiết lập như sau:

- Đi từ v(h,d) → v(h',d) dùng giản đồ *Shortest Routing* trong một hypercube graph.
- Đi từ v(h',d) → v(h',d') dùng giản đồ *Shortest Routing* trong một De Bruijn graph.
- Hoặc ta có thể đảo ngược trình tự trên.

Nhận xét 5*: Từ việc thảo luận trên ta có thể rút ra một công thức, đó là:*
ShortestRouting(HD) = ShortestRouting(H)+ShortestRouting(D)* (*)

Và như thế với việc đảo ngược trình tự trên thì vấn đề Shortest Routing trong mạng HD là không đổi.*

Từ *nhận xét 5*, cho thấy vấn đề *Shortest Routing* trong mạng *HD** là rất hiệu quả. Và điều này làm cho *HD** graph trở thành ưu điểm hơn hẳn so sánh với các mô hình graph khác như hyper-cube, DeBruijn, Butterfly … Tuy nhiên, với Hyper-DeBruijn-Aster graph thì *HD**

đã tăng số cạnh trên một node và điều này cho phép ta có thể tăng số đường *Shortest Routing* trong mỗi graph con DeBruijn lên, thêm vào đó ta kết hợp giữa Hypercube và DeBruijn graph. Điều này cho phép ta tăng số đường *Shortest Routing* lên rất nhiều. Và với đặc điểm này ta so sánh lại với Hypercube, DeBuijn hay Butterfly thì đây là một lợi thế trong vấn đề *Shortest Routing*. Tuy nhiên, vấn đề *Shortest Routing* trong bài [5] của Wei Shi and Pradip K Srimani, các tác giả đề cập có phần giống với cách thức định tuyến của chúng tôi nhưng các tác giả chỉ thực hiện *Shortest Routing* trong mạng Hyper-Butterfly, không đưa ra quy luật cho mạng Hyper-DeBruijn và chỉ nói rằng mạng Hyper-DeBruijn là không có quy tắc và bảng so sánh của tác giả chỉ dừng lại ở mạng Hyper-DeBruijn trong đó phần DeBruijn chỉ dừng lại ở Binary DeBruijn graph. Chúng tôi đã cải tiến nó, làm cho nó trở nên hiệu quả hơn, đó là đưa ra Hyper-DeBruijn-Aster (kiến trúc mạng vừa được đề cập ở những phần trên) và chúng tôi đã áp dụng *Shortest Routing* trên mạng này một cách hiệu quả với việc tìm ra quy luật (*). Tiếp theo, chúng tôi sẽ trình bày một giải thuật định tuyến mà chúng nghiên cứu trong mạng Hyper-DeBruijn-Aster. Với giải thuật này, các node trong mạng sẽ làm việc nhẹ nhàng hơn, nhanh hơn và đặc biệt là hiệu quả trong trong vấn đề *Fault Tolerant Routing, Broadcasting và cân bằng tải*. Vì với cách trình bày trên cùng với các nhận xét, chứng minh và bảng so sánh trên ta thấy Hyper-DeBruijn-Aster graph mà chúng tôi đưa ra là hơn hẳn so với những graph khác như Hypercube, DeBruijn, Butterfly và HyperButterfly. Và trong bài viết [5] của Wei Shi and Pradip K Srimani, các tác giả đã so sánh graph của mình, đó là HyperButterfly graph có những ưu điểm hơn hẳn so với Hypercube, DeBruijn, Butterfly và Hyper-DeBruijn. Dưới đây, chúng tôi sẽ so sánh ưu điểm giữa HyperButterfly graph và graph mà chúng tôi đề nghị.

Giả sử Hyper-DeBruijn-Aster là $HD^*(m_1,d_1,n_1)$, trong đó m_1 là bậc của hypercube, n_1 là bậc của DeBruijn, và d_1 là chỉ số mở rộng bit trên graph ($d_1 \geq 2$).

Và Hyper-Butterfly là $HB(m_2,n_2)$, trong đó m_2 là bậc của hypercube còn n_2 là bậc của Buterfly.

Và giả sử việc thiết kế mạng là như nhau tức cùng số node:

Ta có: $2^{m_1} d_1^{n_1} = n_2 2^{m_2+n_2}$

$$\Leftrightarrow m_1 + n_1 \log_2 d_1 = m_2 + n_2 + \log_2 n_2 (1)$$

Do đó, degree của HD*: $d_{HD^*} = m_1+2xd_1 \geq m_1 +4$ thì rõ ràng nếu d_1 càng lớn thì d_{HD^*} càng lớn và nếu như ta cho $m_1 = m_2 \rightarrow d_{HD^*} \geq d_{HB}$. Điều này làm mạng *HD** hiệu quả hơn *HB* trong vấn đề *Fault-Tolerant Routing và Broadcasting*.

Bên cạnh đó, Diameter của *HD**: $D_{HD^*} = m_1+n_1= m_2+n_2 + \log_2 n_2 - n_1 \log_2 d_1 + n_1$.

Rõ ràng khi d_1 tăng lên thì D_{HD^*} giảm và nếu $d_1 \geq 2n_2$ thì $D_{HD^*} \leq D_{HB}$. Điều này cho ta thêm một minh chứng nữa là mạng *HD** hiệu quả hơn *HB* trong vấn đề *Shortest Routing*.

Và cuối cùng là :

$$Fault-Tolerance(HD^*) = m_1 + 2d_1 - 2 (2)$$

Rõ ràng nếu d_1 tăng thì *Fault-Tolerance(HD*)* tăng lên. Trong khi, *Fault-Tolerance(HB)* chỉ bằng $m_2 + 4$.

Với những gì vừa trình bày trên, cho thấy Hyper-DeBruijn-Aster tối ưu hơn so với Hyper-Buttfly trong vấn đề *Fault-Tolerant Routing, Shortest Routing và Broadcasting*.

Sau đây, chúng tôi sẽ trình bày giải thuật Shortest Routing trong mạng Hyper-DeBruijn-Aster mà chúng tôi nghiên cứu.

<u>*Bước 1:*</u> Dùng giải thuật *Shortest Routing* trong mạng DeBruijn để tìm ra những đường đi ngắn nhất đi từ $d_S \rightarrow d_D$. Đây là bước tính lý thuyết dựa vào cơ sở hạ tầng

của mạng mà không phụ thuộc và trạng thái của mạng lúc định tuyến.

Bước 2: Thực hiện định tuyến trong mạng *HD**, việc định tuyến của mạng ở bước này sẽ phụ thuộc vào tình trạng của mạng. Cách thức định tuyến sẽ tiến hành như sau:

- Ban đầu, nó sẽ xét đến đường đi ngắn nhất thứ nhất (chọn bất kỳ một trong số đường đi ngắn nhất vừa tìm được ở bước 1). Và nó sẽ định tuyến đến node thứ nhất trên đường đi thứ nhất đó. Nó sẽ kiểm tra kết nối nếu thành công thì nó sẽ định tuyến đến đó. Nếu không thành công thì nó sẽ kiểm tra đến node thứ 1 của đường ngắn nhất thứ 2, và nếu vẫn không thành công thì nó sẽ kiểm tra đến node thứ 1 của đường tiếp theo. Cứ như thế mà đến đường cuối cùng. Còn nếu đến node thứ 1 của đường cuối cùng mà vẫn không thành công thì nó sẽ thực hiện việc định tuyến đến một hypercube thứ 1 (lân cận hypercube của nó). Trong đó, hypercube thứ nhất được xác định bằng cách đảo ngược 1 bit khác nhau giữa phần địa chỉ hypercube của địa chỉ nguồn và đích (sự so sánh khác biệt này có thể từ trái sang phải hoặc từ phải sang trái). Nếu không thành công với hypecube lân cận thứ 1 thì tiếp tục xét đến hypercube lân cận thứ 2. Nếu đến hypercube lân cận cuối cùng mà không thành công thì chúng ta sẽ quay lại bước 1 với giả sử là các node lân cận của địa chỉ nguồn vừa xét ở trên là không tham gia vào việc tìm ở bước 1. Sau đó, ta sẽ tiếp tụ lại từ đầu bước 2 và ở bước này việc tìm nếu xãy tra lỗi đến tất cả các node lân cận của phần DeBruijn mà gặp lỗi thì nó bỏ qua việc thử đến các hypercube lân cận và tiếp tục lại bước 1 như trên. Vì tính chất của *HD** là để đi từ node v(h,d) → v(h',d') thì nó sẽ lần lượt v(h,d) →v(h,d') →v(h',d') nên việc thực hiện giải thuật *Shortest Routing* như trên là tối ưu.

- Khi nó đã định tuyến được đến node tiếp theo thì ta tiếp tục định tuyến đến node tiếp theo với giải thuật tương tự như trên. Và bây giờ địa chỉ nguồn cho việc định tuyến tiếp theo, nó là địa chỉ của node tiếp theo này. Trường hợp, việc định tuyến từ node này đến tất cả các node lân cận (trừ node vừa định tuyến trước đó) bị lỗi thì nó gởi lại thông báo cho node định tuyến trước đó biết. Để node ngay trước đó, nó định tuyến lại đến node khác. Hoặc thời gian của node ngay trước đó bị timeout thì node ngay trước đó cũng sẽ định tuyến lại với hypercube lân cận khác.

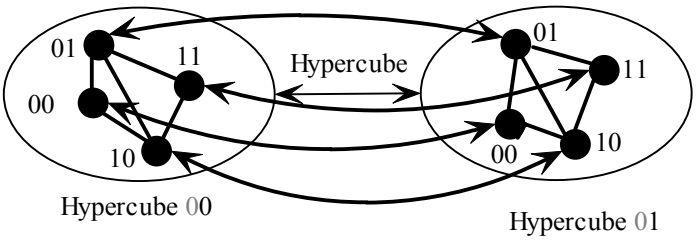
Nhận xét 6: Với Hyper-DeBruijn với việc mở rộng bit, cho phép ta tăng số đường đi ngắn nhất giữa 2 node trong mạng Hyper-DeBruijn. Hơn nữa, mạng Hypercube cũng cho ta nhiều đường đi ngắn nhất giữa 2 điểm. Kết hợp với công thức (*), cho ta một graph Hyper-DeBruijn-Aster với số đường đi ngắn nhất tăng lên rất nhiều. Nếu giả sử đường đi ngắn nhất từ 2 điểm trong phần DeBruijn mà không có sự tham dự của Hypercube đi từ $d_S \rightarrow d_D$, nó đi qua là N node (kể cả node nguồn và node đích). Và giả sử số bit khác nhau giữa địa chỉ nguồn và đích của phần là M thì số đường đi ngắn nhất giữa 2 điểm bất kỳ trong mạng Hyper-DeBruijn-Aster là NxM! đường.

Nhận xét 7: Ưu điểm của giải thuật trên là chỉ cần tính đường ngắn nhất của phần De-Bruijn graph mà không cần phải tính đường đi ngắn nhất của toàn mạng. Điều này cho phép ta tăng tốc độ xử lý và giảm gánh nặng cho mạng.

Nhận xét 8: Việc định tuyến trong phần Hypercube, ta chỉ cần xét số bit khác nhau giữa địa chỉ nguồn và địa chỉ đích, đó chính là số node hypercube lân cận có thể gởi tối đa trong giải thuật *Shortest Routing*. Do đó, có thể tính được số đường đi ngắn nhất của Hypercube một cách dễ dàng bằng số bit khác nhau giữa địa chỉ nguồn và địa chỉ đích lấy giai thừa. Đây là

đặc điểm giúp ta tìm ra đường đi ngắn nhất trong hypercube một cách dễ dàng và nhanh chóng mà không cần phải tính toán.

Nhận xét 9: Ta thấy rằng mạng Hyper-DeBruijn-Aster có đặc tính ánh xạ theo cạnh của hypercube, được minh họa ở hình 5. Do đó, nếu việc định tuyến từ một node đến một node kế tiếp trong cùng một Hypercube bị lỗi thì nó có thể chuyển sang một Hypercube kế tiếp. Tuy nhiên, ta vẫn giữ được phần địa chỉ De-Bruijn, tức là có thể định tuyến theo hướng ban đầu, thay vì trên Hypercube ban đầu thì nó là một Hypercube khác.



Hình 5: Từng phần tử của Hypercube 00 sẽ ánh xạ sang Hypercube 01

Ví dụ: để định tuyến từ node (00,01) → (01,11). Thì ban đầu nó dùng giải thuật định tuyến *Shortest Routing De-Bruijn* trên cùng một Hypercube. Tức sẽ đi từ (00,01) → (00,11) → (01,11). Nếu node (00,11) bị lỗi thì nó sẽ thực hiện đi từ (00,01) → (01,01). Ta thấy địa chỉ của phần De-Bruijn vẫn không đổi. Và tiếp tục định tuyến theo hướng từ 01 → 11 của phần De-Bruijn, tức là đi từ (01,01) → (01,11). Với cách định tuyến như trên, mạng cho phép node định tuyến tiếp theo bị lỗi mà vẫn đảm bảo vấn đề *Shortest Routing.* Và như giải thuật vừa trình bày trên và ví dụ minh chứng vừa trình bày thì nếu các node lân cận của node tiếp theo bị lỗi hết thì node tiếp theo đó sẽ gởi một thông điệp báo lỗi không thể định tuyến được về lại node trên nó. Để node trên nó sẽ đi theo đường đi ngắn nhất thứ 2 hoặc là gởi sang Hypercube lân cận. Và với giải thuật như vậy, ta vẫn đảm bảm vấn đề *Shortest Routing*.

Nhận xét 10: Việc định tuyến theo giải thuật Shortest Routing như trên, ta thấy rằng có sự tỷ lệ giữa độ dài đường đi và số đường đi ngắn nhất. Và điều này phụ thuộc vào số bit khác nhau trong phần hypercube giữa địa chỉ nguồn và đích. Nếu số bit khác nhau tăng thì đường đi sẽ dài hơn và như thì số đường đi ngắn nhất sẽ tăng lên. Và điều này phù hợp thực tế, vì đường đi càng dài thì xác suất lỗi cũng tăng lên và ngược lại.

Với giải thuật trên, trong trường hợp không lỗi và giả sử số node trên đường đi ngắn nhất của phần De-Bruijn là N_D và số bit khác nhau giữa địa chỉ nguồn và đích là M_H thì số node mà nó định tuyến qua sẽ bằng $N_D + M_H \leq D_{HD*}$, D_{HD*} là Diameter của *HD**.

4. Fault Tolerance của *HD*(m,d,n)*

Định nghĩa 3: *Fault Tolerance của một mạng là khả năng của mạng có thể tiếp tục vận hành đúng đắn ngay cả khi một hay nhiều node bị lỗi.*

Việc áp dụng giải pháp *Shortest Routing* vào một graph nào đó thì cơ bản ta sẽ tìm ra một đường ngắn nhất và cho định tuyến theo đường đó. Điều này rất tốt đối với việc tăng tốc định tuyến hay giảm thiểu chi phí hay số node đi qua … nhưng nó sẽ không thể định tuyến thành công nếu một node nào đó trên đường đi bị lỗi. Tuy nhiên, với graph *HD**, nó cho phép chúng ta tăng số đường đi ngắn nhất lên. Điều này, cho phép ta thừa hưởng được ưu điểm của việc *Shortest Routing* và đồng thời cải tiến được vấn đề *Fault Tolerance*. Hơn nữa, việc áp dụng giải thuật *Shortest Routing* mà chúng tôi

nghiên cứu vào *HD**, điều này làm cải tiến được vấn đề *Shortest Routing*, tức nếu vấn đề lỗi tại một node lân cận nào đó thì nó sẽ giải quyết ngay tại node đó hay node trên nó mà không cần phải yêu cầu gởi lại từ địa chỉ nguồn nguyên thủy.

Trong một mạng bất kỳ và yêu cầu là một giải thuật định tuyến tốt nhất thì khi định tuyến từ 1 điểm → 1 điểm thì điều kiện tối thiểu là một trong những node lân cận của địa chỉ nguồn và một trong những node lân cận của đia chỉ đích và các Hypercube trung gian phải hoàn toàn tốt. Ngược lại, mạng không thể hoạt động được. Vài giải thuật *Shortest Routing* chỉ không thực hiện thành công trong *HD** chỉ xãy ra trong trường hợp như trên. Điều này cho thấy, giải thuật *Shortest Routing* của chúng tôi trong *HD** là tối ưu trong vấn đề *Fault Tolerance*.

Giả sử, node bất kỳ trong mạng có địa chỉ $<x_0x_1x_2..x_i\ldots x_{n-1}>$, $0 \le i \le n-1$

Với mạng *HD*(m,d,n)* thì số node lân cận của môt node bất kỳ là N_{HD^*} bằng:

$N_{HD^*} = m + 2d$ $\leftrightarrow \exists x_i \mid x_i \neq x_{i+2}, 0 \le i \le n-3 \quad (3)$
$N_{HD^*} = m + 2d - 1$ $\leftrightarrow \left\{ \begin{array}{c} \forall x_i \mid x_i \neq x_{i+2}, 0 \le i \le n-3 \\ x_0 \neq x_1 \end{array} \right\} \quad (4)$
$N_{HD^*} = m + 2d - 2$ $\leftrightarrow \forall x_i \mid x_i = x_{i+1}, 0 \le i \le n-2 \quad (5)$

Do đó, *Fault Tolerance* của mạng *HD** khi áp dụng giả thuật *Shortest Routing* bằng N_{HD^*} -1. Hay nói cách khác, khả năng chịu lỗi trên đường định tuyến giữa 2 node bất kỳ sử dụng giải thuật *Shortest Routing* nó phụ thuộc vào vị trí node bị lỗi. Do đó, độ phức tạp của giải thuật cũng phụ thuộc vào vị trí lỗi của node trong mạng.

5. Broadcasting và cân bằng tải trong *HD*(m,d,n)*

Việc mở rộng bit trong *HD**, cho phép ta tăng degree và giảm Diameter (chứng minh ở mục 2) cũng như tăng số node lân cận và số cạnh lân cận. Điều này làm cho *HD** tăng khả năng broadcasting.

Cùng với việc tăng *Fault Tolerace*. Như vậy, việc định tuyến node → node sẽ thành công cao hơn. Như vậy khả năng broadcasting đến tất cả các node lân cận trong mạng sẽ được bảo đảm hơn.

Do đặc tính của *HD** là cho nhiều đường đi ngắn nhất. Điều này làm cho mạng có vừa có khả năng định tuyến đường đi ngắn nhất vừa cân bằng tải vừa có khả năng đồng bộ hóa trong những vấn đề xử lý song song và phân tán dữ liệu. Và có thể nói, nó cải tiến rất nhiều trong việc tăng hiệu suất truyền thông đa xử lý.

6. Kết luận

Chúng tôi vừa đề nghị một cách tiếp cận mới trong vấn đề định tuyến giữa 2 node trong mạng. Đó là đưa ra một mô hình graph mới là Hyper-DeBruijn-Aster graph. Và đồng thời chúng tôi cũng đề nghị một giải thuật định tuyến *Shortest Routing cải tiến* áp dụng trên mạng *HD**. Với cách cách tiếp cận trên, chúng tôi đã cải thiện rất nhiều trong vấn đề *Shortest Routing, Fault Tolerance và Broadcasting*. Bên cạnh những ưu điểm đó, *HD** còn có thể góp phần không nhỏ trong vấn đề *cân bằng tải, xử lý đồng bộ* trong những *giải thuật xử lý song song và phân tán dữ liệu* trong mạng truyền thông đa xử lý.

Tài liệu tham khảo

[1]. Samantham, R. Maheswara , and D.K. Pradhan, "*The De Bruijn Multi-processor Network: A Versatile Parallel Processing and Sorting Network for VLSI*," IEEE Trans. on Comp., Vol.38, NO.4, 1989.

[2]. Zhen Liu, Ting-Yi Sung,"*Routing and Transmitting Problem in de Bruijn Networks*" IEEE Trans. On Comp., Vol. 45, Issue 9, Sept. 1996, pp 1056 – 1062.

[3] Elango Ganesan, Dhiraj K Pradhan, "*Wormhole Routing In De Bruijn Networks And Hyper-Debruijn Networks*", Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on Vol :3 , 25-28 May 2003.
[4] J. A. Bondy and U. S. R. Murty, "*Graph Theory and Aplication"*.
[5] Wei Shi and Pradip K Srimani, "*Hyper-Butterfly Network: A Scalable Optimally Fault Tolerant Architecture"*.
[6]. Jyh-Wen Mao and Chang-Biau Yang, "*Shortest path routing and fault tolerant routing on de Bruijn networks*",Journal of Networks,Vol.35, Issue 3, Pages 207-215 2000.
[7] Ngoc Chi Nguyen, Nhat Minh Dinh Vo and Sungyoung Lee, "*Fault Tolerant Routing and Broadcasting in de Bruijn networks"*.
[8]. A.H. Esfahanian, G. Zimmerman, "*A distributed broadcast algorithm for binary De Bruijn networks*", 1988. Conference Proceedings, Seventh Annual International Phoenix Conference on Comp. and Comm., 16-18 March 1988.
[9] Dally, W. J., "*Performance analysis of k-ary n-cube interconnection networks*", IEEE Trans. on Computer*s*, vol. 39, pp. 775-785, Jun 1990.
[10]. S.R.Ohring, D.H.Hondel, "*Optimal Fault-Tolerant Communication Algorithms on Product Networks using Spanning Trees*", IEEE Symposium on Parallel Distributed Processing, Oct 1994.
[11] Pradhan, D. K. and Reddy, S. M., "*A fault-tolerant communication architecture for distributed systems*", IEEE Trans. On Computers, vol. C-31, pp. 863-870, Sep 1982.

Địa chỉ liên lạc

NGUYỄN HỒNG THÁI
Sinh viên khoa Điện - Điện Tử
Trường Đại học Bách Khoa TPHCM
Chuyên ngành: Viễn Thông
Địa chỉ: 241A, Lý Thường Kiệt, P.15, Q.11, TPHCM.
Điện thoại: (084) 08 864 7557
Email: nhthai2005@gmail.com

PHẠM MINH TRÍ
Sinh viên khoa Điện - Điện Tử
Trường Đại học Bách Khoa TPHCM
Chuyên ngành: Viễn Thông
Địa chỉ:
Điện thoại:
Email: minhtriphamlth@yahoo.com

NGUYỄN CHÍ NGỌC
Cán bộ giảng dạy, Khoa Điện - Điện Tử
Trường Đại học Bách Khoa TPHCM
Bộ môn: Viễn Thông
Địa chỉ
Điện thoại:
Email: ncngoc@hcmut.edu.vn