Proceeding, Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008)
Auditorium Universitas Gunadarma, Depok, 20-21 Agustus 2008 ISSN : 1411-6286

ALGORITMA ROUTING UNTUK INDENTIFIKASI JALUR TERPENDEK PADA INCOMPLETE-HYPERCUBE

Ernastuti

Faklutas Ilmu Komputer dan Teknologi Informasi
Universitas Gunadarma
Email : ernas@staff.gunadarma.ac.id

ABSTRAK

Pada tulisan ini dibahas tentang pengembangan suatu algoritma routing untuk menentukan jalur terpendek antara dua prosesor di dalam hypercube yang faulty (incomplete hypercube). Algoritma ini dapat mengidentifikasi kondisi konektifiti jaringan ketika beberapa prosesor pada jalur menjadi faulty. Pada saat kondisi jaringan tetap tehubung, algoritma selalu dapat menunjukkan panjang jalur terpendek antara dua prosesor. Konsep jarak Hamming dan graf kombinatorik himpunan string biner digunakan dalam mendisain Algoritma. Kompleksitas algoritma ini adalah O(N), N adalah jumlah simpul dalam incomplete-hypercube.

Kata kunci : *string biner, incomplete hypercube, jalur terpendek, algoritma routing, jarak Hamming, deadlock.*

1 PENDAHULUAN

Hypercube adalah suatu sistem paralel terdistribusi yang terdiri dari 2^n prosesor dan $n2^{n-1}$ saluran komunikasi (link), di mana setiap unit prosesor memiliki memori sendiri dan terhubung dengan *n* prosesor disebelahnya. Pada paper Saad dan Schultz [7] diperlihatkan bahwa hypercube memiliki struktur homogen yang simetri dan mengandung banyak struktur lain sehingga topologi seperti mesh, ring dan tree dapat ditanamkan. Sistem hypercube diantaranya diaplikasikan pada masalah-masalah pemrosesan sinyal digital, pemrosesan citra, pengenalan pola dan pemrosesan database.

Hypercube dimensi-*n* adalah model jaringan yang terdiri dari dua buah hypercube dimensi *n*-1. Suatu problem pada topologi hypercube adalah bahwa jumlah prosesor di dalam suatu system harus 2^n (power dari 2). Pada prakteknya hal ini sangat membatasi ukuran sistem yang dapat dibangun. Katseff pada papernya tahun 1988 [3] mengatakan bahwa kelemahan ini dapat diatasi dengan menggunakan hypercube yang tidak lengkap (incomplete hypercube), yaitu hypercube yang mengandung prosesor *faulty* (cacat). Tidak seperti hypercube, pada incomplete hypercube dapat dibangun dari sebarang jumlah simpul.

Relibilitas dari pemrosesan dan komunikasi data sangat penting dalam sistem hypercube seperti pada semua sistem paralel lainnya. Kecepatan pemrosesan dan komunikasi serta toleransi peyimpangan dalam sistem akan menurun bila banyak prosesor atau linknya menjadi faulty. Hal ini menjadi permasalahan dalam kinerja sistem. Oleh karena itu, untuk menentukan dan menghindari

prosesor dan link yang faulty dalam komunikasi data, banyak penelitian sampai saat ini mengusulkan metode-metode baru untuk menemukan jalur terpendek diantara prosesor yang menjadi sumber (*source*) dan prosesor yang menjadi target (*destination*). Terdapat tiga jenis cara berkomunikasi antar prosesor, yaitu one-to-one (*routing*), one-to-many (*multicasting*) dan one-to-all (*broadcasting*).

Pada [3] diusulkan algoritma pencarian jalur terpendek di dalam hypercube yang faulty dan tidak faulty dengan pendekatan operasi bitwise exclusive-OR. Pada paper Katseff tersebut diusulkan juga algoritma routing yang bebas deadlock ketika beberapa prosesor menjadi faulty. Liu dan Hsu pada papernya tahun 1992 [4] memanfaatkan algoritma Katsef untuk mendisain algoritma routing dan broadcasting khusus untuk kelas topologi interkoneksi generalized-Fibonaci-cube. Kemudian Qian dan Wu pada tahun 1995 [5] mengusulkan algoritma routing, broadcasting dan multicasting dengan pendekatan jarak Hamming dan sifat Hamiltonicity khusus untuk kelas jaringan interkoneksi enhanced-Fibonacci-cube. Kemudian Allahverdi, Kahramanli dan Erciyes pada tahun 2000 [1] mengusulkan algoritma routing mencari jalur terpendek di dalam hypercube yang faulty dengan pendekatan aljabar kubus.

Pada makalah ini dikembangkan suatu algoritma routing (one-to-one) untuk menentukan jalur terpendek antara prosesor sumber dan prosesor target di dalam hypercube yang faulty. Ide pengembangan algoritma ini terinspirasi dari algoritma yang ada pada paper [3] dan [5]. Algoritma yang diusulkan pada tulisan ini dapat mengidentifikasi kondisi konektifiti jaringan hypercube ketika beberapa prosesor pada jalur menjadi faulty dan dapat menunjukkan panjang jalur terpendek ketika kondisi jaringan tetap terhubung.

2 DEFINISI DAN NOTASI

Berikut diberikan definisi-definisi, notasi-notasi yang mendukung pengembangan algoritma routing pada tulisan ini. Definisi-definisi dan notasi-notasi pada bab ini diambil dari paper Baril dan Vincent [2] dan Roosta [6].

Definisi 1: Bit adalah suatu digit biner 0, 1 atau λ. S*tring biner* panjang-*n* adalah suatu barisan hingga bit dengan panjang-*n*., atau dengan kata lain string biner panjan-*n* adalah suatu barisan $a_n\ a_{n-1} \ldots a_1$ dengan $a_i \in \{0,1\}$.

String λ adalah string yang berisi bit kosong dengan panjang 1. Jikan α adalah string biner dan V adalah himpuna string biner, maka opearasi $\alpha \cdot V$ menyatakan *concatenation* antara string α dan setiap bit-string dalam *V*. Contoh: $01.\{0,1\} = \{010, 011\}$.

Definisi 2: *Jarak Hamming* diantara dua string biner x dan y panjang-*n* adalah jumlah dari posisi bit berbeda diantara keduannya; notasi: *H(x,y)*. Contoh: H(001,100)=2.

Definisi 3: Sebuah graf G=(V,E) terdiri dari himpunan simpul V dan himpunan busur E; sebuah busur adalah suatu pasangan tidak terurut (x,y), di mana $x \neq y \in V_G$. Untuk menghindari pengertian yang ambigu, V dan E dalam graf G masing-masing dinotasikan V_G dan E_G.

Topologi jaringan interkoneksi biasanya dinyatakan sebagai graf, di mana simpul

menyatakan elemen prosesor dan busur menyatakan saluran komunikasi dalam jaringan.

Definisi 4: Sebuah *jalur (path)* pada suatu graf $G=(V_G,E_G)$ adalah barisan terurut $(x_1, x_2, \ldots, x_n)$, $x_i \in V_G$ sedemikian sehingga $(x_1, x_2,)$, (x_2, x_3), …, (x_{n-1}, x_n) adalah busur-busur dalam E_G dimana x_i masing-masing berbeda. Sebuah jalur tertutup $(x_1, x_2, \ldots, x_n)$, $x_1=x_n$, disebut *siklus* (*cycle*).

Jumlah busur dalam dalam suatu jalur disebut panjang (*length*) jalur. *Panjang jalur terpendek* (*short path*) antara x dan y dinotasikan sebagai d(x,y), x,y $\in V_G$.

Definisi 5: Graf G dikatakan *terhubung* jika terdapat paling sedikit satu jalur diantara setiap pasang simpul dalam G.

Definisi 6: Kelas dari objek kombinatorik membentuk suatu graf kombinatorik dengan ketentuan sebagai berikut: simpul adalah objek dan setiap dua simpul akan dihubungkan dengan busur jika jarak Hamming diantara keduanya dibatasi oleh suatu kontanta.

Jika objek kombinatorik adalah himpunan semua string biner dalam $\{0,1\}^n$, $n > 0$, dan untuk setiap dua string binernya terhubung bila jarak Hammingnya sama dengan 1 (satu), maka graf kombinatorik yang terbentuk adalah graf hypercube. Setiap node A mempunyai label atau alamat $a_n\ a_{n-1}\ \ldots\ a_1$ dengan $a_i \in \{0,1\}$, dimana a_i disebut bit ke *i* atau dimensi ke *i* dari alamat.

Hypercube dimensi-n , dinotasikan *Q(n)*, adalah suatu graf tak berarah yang setiap simpulnya diwakili oleh suatu *string* biner panjang-*n* bit. Setiap dua simpul dalam Q(n) akan terhubung jika *string-string* yang mewakilinya berbeda satu posisi bit.

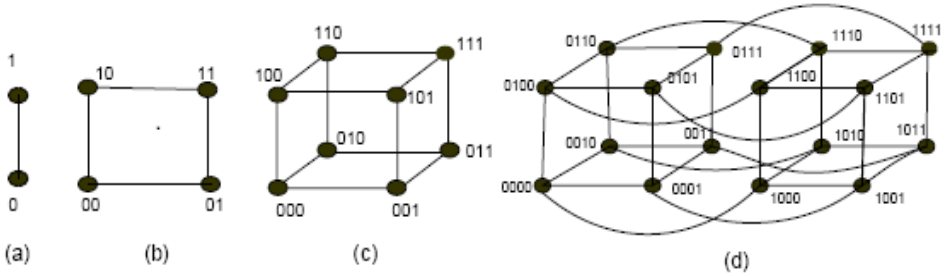
Jumlah simpul dalam *Q(n)* adalah 2^n, dimana setiap simpul mempunyai keterhubungan dengan n simpul lainnya. Derajat setiap simpul adalah *n*. Hypercube dapat didefinisikan secara rekursif sebagai berikut.

Definisi 7: ***(Hypercube***) Untuk $n \geq 0$, $Q(n) = (V_Q(n),E_Q(n))$ dapat didefinisikan : Himpunan simpul $V_Q(n)$ dinyatakan secara rekursif sebagai berikut $V_Q(n) = 0.V_Q(n-1) \cup 1 \cdot V_Q(n-1)$, di mana $V_Q(0) = \{\}$; $V_Q(1) = \{0,1\}$; dua simpul $x,y \in V_Q(n)$ dihubungkan oleh suatu busur $\in E_Q(n)$ jika dan hanya jika jarak Hamming H(x,y)=1.

Contoh: untuk hypercube dimensi-3, Q(3), maka himpunan simpulnya adalah { 000, 001, 010, 100, 101, 110, 011, 111} (lihat Gambar 1.c).

Jarak Hamming antara simpul A dan B dalam hypercube menunjukkan panjang jalur terpendek antara simpul A dan B. Contoh: Panjang jalur terpendek antara 0000 dan 1111 adalah H(0000,1111) = 4; Panjang jalur terpendek antara 1010 dan 1111 adalah H(1010,1111) = 2;

Gambar 1 memperlihatkan graf hypercube Q_n, untuk dimensi n=1,2,3,4.



Gambar 1. Hypercube Q(n) : (a) Q(1), (b) Q(2), (c) Q(3), (d) Q(4)

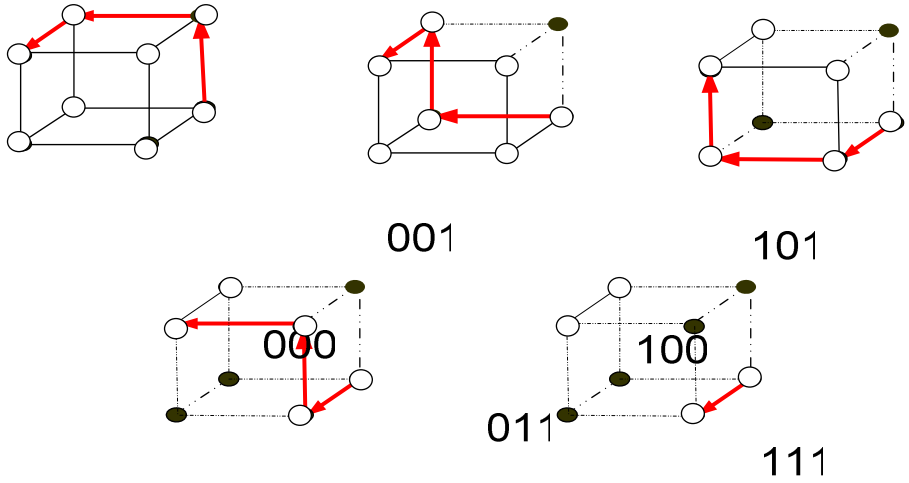
Pada bab-bab selanjutnya kata 'simpul' dan 'prosesor' menunjukkan arti yang sama. Demikian juga dengan kata 'busur' dan 'link' serta kata 'graf' dan 'topologi jaringan'.

3 PROSEDUR DISAIN

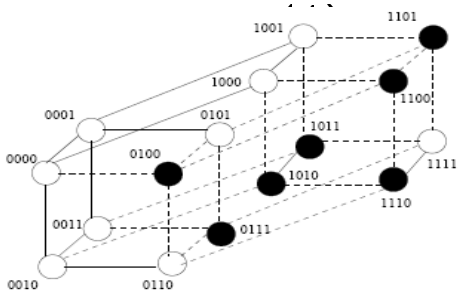
Komunikasi dalam sistem jaringan dicapai dengan membawa pesan/data (*message passing*) melewati busur-busur dalam jalur. Sebuah pesan yang berasal dari suatu prosesor dan berakhir di prosesor lain dapat dilalukan (*routed*) melewati beberapa prosesor perantara (*intermediate processor*) yang ada pada jalur.

Hypercube Q(*n*) mempunyai sifat menarik, yaitu jumlan jalur terpendek dengan panjang *k* antara dua prosesor s_1 dan s_2 berjumlah (*k*-1)! jalur, $k \leq n$. Namun ketika beberapa prosesor menjadi faulty, maka empat kasus berikut mungkin akan terjadi. Kasus pertama, kasus ketika satu atau beberapa prosesor pada jalur antara prosesor s_1 dan s_2 dengan jalur terpendek *k* menjadi faulty, hypercube Q(n) yang sudah menjadi incomplete masih tetap mempunyai jalur terpendek panjang *k* lainnya. Gambar 2.1, 2.2, 2.3 dan 2.4 mengilustrasikan hypercube dimensi-3 Q(3) ketika prosesor satu persatu menjadi faulty. Terlihat bahwa panjang jalur terpendek dari 111 ke 000 masing-masing Gambar masih tetap 3. Kasus kedua (lihat Gambar 2.5) , ketika banyaknya prosesor yang faulty mengakibatkan jalur diantara prosesor 111 dan 000 menjadi putus. Hal ini mengakibatkan jaringan menjadi tak terhubung (*disconnected*). Kasus ketiga (lihat Gambar 3), kemungkinan bisa terdapat jalur antara dua prosesor dengan panjang jalur terpendek ≥ n. Contoh: Jalur antara prosesor 0101 dan 0110 pada hypercube dimensi-4 mempunyai panjang jalur terpendek adalah 4, yaitu melalui barisan prosesor 0101, 0001, 0011,0010, 0110.

Kasus keempat (lihat Gambar 3), kemungkinan terjadi deadlock pada pengiriman pesan, sehingga harus dilakukan backtracking sampai kepada simpul yang mempunyai link alternatif lain menuju simpul target. Contoh ketika simpul sumber adalah 1001 dan simpul target adalah 0110. Simpul-simpul pada jalur yang mungkin adalah 1001, 0001, 0101 , sampai disini routing tidak bisa dilanjutkan karena tidak ada lagi simpul yang bersebelahan dari simpul 0101. Kondisi ini disebut deadlock. Selanjutnya routing melakukan backtracking sampai pada simpul 0001. Kemudian routing dilanjutkan hingga ke simput target dengan melalui jalur 1001, 0001, 0011, 0010, 0110.



Gambar 2. Jalur terpendek dari 111 ke 000 pada incomplete hypercube dimensi-3, Q(3).



Gambar 3. Jalur terpendek dari 0101 ke 0110 pada incomplete hypercube dimensi-4, Q(4).

4 ALGORITMA ROUTING

Perhatikan jalur dari simpul 1001 ke simpul 0110 pada Gambar 3. Jalur tersebut melalui simpul-simpul: 1001, 0001, 0011, 0010, 0110. Simpul **1**001 ke **0**001 dihubungkan dengan sebuah busur dari bit ke 4; Simpul 00**0**1 ke 00**1**1 dihubungkan dengan busur dari bit ke 2; Simpul 001**1** ke 001**0** dihubungkan dengan busur dari bit ke 1; Simpul 0**0**10 ke 0**1**10 dihubungkan dengan busur dari bit ke 3. Sehingga jalur dari 1001 ke 0110 dapat dinyatakan dengan barisan busur berdasarkan dimensinya (4,2,1,3).

Algoritma routing ini didisain berdasarkan keempat kasus di bab tiga. Berikut adalah variabel-variabel yang digunakan pada algoritma routing. Suatu busur mempunyai nomor link *i* jika dan hanya jika busur tersebut menghubungkan dua alamat simpul berbeda tepat pada bit ke *i*. Alamat relatif (*relative address*) antara dua simpul adalah operasi bitwise exclusive-OR, dinotasikan dengan ⊕, antara bit-bit dalam string binernya. Contoh: 1001⊕ 0110 = 1111; 1101⊕1110=0011.

Di langkah awal, prosesor sumber atau setiap prosesor perantara selalu membandingkan alamatnya dengan alamat prosesor target dan memperoleh himpunan dimensi *i* di mana dua alamat saling berbeda di posisi *i*. Himpunan dimensi ini dinyatakan dengan variabel *himp_dim_beda*. Di dalam hypercube, hal ini dilakukan hanya dengan memilih

sebarang dimensi dari himpunan dan mengirim pesan ke tetangga dari dimensi yang dipilih. Namun di dalam incomplete-hypercube ada kemungkinan tetangga dari dimensi yang dipilih tidak ada. Misalkan variabel himpunan dari dimensi-dimensi pada prosesor sumber atau perantara yang mempunyai tetangga dinyatakan dengan *himp_dim_ada_tetangga*. Dengan mengiris himpunan *himp_dim_ada_tetangga* dengan *himp_dim_beda* maka diperoleh himpunan baru yang dinyatakan dengan *himp_dim_ada*. Himpunan *himp_dim_ada* berisi semua dimensi *i* di mana *i* menunjukkan alamat (bit ke *i*) dari simpul yang sedang dikunjungi dengan alamat (bit ke *i*) simpul target yang saling berbeda dan terdapat tetangga dari simpul yang sedang dikunjungi . Kemudian sebuah dimensi dapat dipilih dari *himp_dim_ada*, selanjutnya informasi dikirim dari simpul yang baru dikunjungi ke tetangganya sepanjang dimensi.

Dengan demikian algoritma routing mencari jalur terpendek antara setiap dua prosesor di dalam incomplete-hypercube dapat diperlihatkan sebagai berikut.

Alamat simpul yang sedang dikunjungi dinyatakan dengan variabel *simpul_kunjung* dan alamat simpul target dinyatakan dengan variabel *simpul_target.*

Untuk simpul sumber dan setiap simpul perantara **do** :
himp_dim_beda ← { i | bit ke i dari *simpul_kunjung* ⊕ *simpul_target* = 1 }
If *himp_dim_beda* = ∅
then simpul target dicapai.
else
himp_dim_ada = himp_dim_ada_tetangga ∩ *himp_dim_beda*
If *himp_dim_ada* = ∅ dan *himp_dim_ada_tetangga* = ∅
then lakukan backtacking sampai ke alamat *simpul_kunjung*
yang mengeluarkan busur *i*;
If *himp_dim_ada* = ∅
then pilih sebuah dimensi *i* ∈ *himp_dim_ada_tetangga* , kemudian
kirim pesan ke tetangga dimensi *i;*
else
If | *himp_dim_ada* | = 1 dan i ∈ *himp_dim_ada* menyebabkan deadlock ,
then jaringan menjadi tidak terhubung;
else
pilih sebuah dimensi *i* ∈ *himp_dim_ada* , kemudian
kirim pesan termasuk alamat simpul target ke tetangga sepanjang
dimensi *i* ;

5 UJI COBA ALGORITMA

Perhatikan incomplete-hypercube pada Gambar 3. Misalkan simpul sumber 1001 dan simpul target 0110. *simpul_kunjung*=1001 dan *simpul_target* = 0110. Maka *himp_dim_beda* = {4,3,2,1}; Karena *himp_dim_ada_tetangga* = {4,1}. Maka *himp_dim_ada = himp_dim_ada_tetangga* ∩ *himp_dim_beda* = {4,1}. Kemudian pilih dimasensi 4 ∈ {4,1}. Kirim pesan melalui busur nomor 4.

Sekarang *simpul_kunjung*=0001 dan *simpul_target* = 0110. Maka *himp_dim_beda* = {3,2,1}; Karena

himp_dim_ada_tetangga = {3,2,1}. Maka *himp_dim_ada* = *himp_dim_ada_tetangga* ∩ *himp_dim_beda* = {3,2,1}. Kemudian misal pilih dimensi 3 ∈ {3,2,1}. Maka kirim pesan melalui busur nomor 3.

Sekarang *simpul_kunjung*= 0101 dan *simpul_target* = 0110. Maka *himp_dim_beda* = {2,1}; Namun *himp_dim_ada_tetangga* = ∅. Maka *himp_dim_ada* = *himp_dim_ada_tetangga* ∩ *himp_dim_beda* = ∅. Berarti lakukan backtracking sampai dengan ke *himp_dim_ada* = {3,2,1}. Sekarang pilih dimensi 2 ∈ {3,2,1}. Maka kirim pesan melalui busur nomor 2.

Sekarang *simpul_kunjung* = 0011 dan *simpul_target* = 0110. Maka *himp_dim_beda* = {3,1}; Karena *himp_dim_ada_tetangga* = {1}. Maka *himp_dim_ada* = *himp_dim_ada_tetangga* ∩ *himp_dim_beda* = {1}. Pilih dimensi 1 ∈ {1}. Maka kirim pesan melalui busur nomor 1.

Sekarang *simpul_kunjung* = 0010 dan *simpul_target* = 0110. Maka *himp_dim_beda* = {3}; Karena *himp_dim_ada_tetangga* = {3,2}. Maka *himp_dim_ada* = *himp_dim_ada_tetangga* ∩ *himp_dim_beda* = {3}. Pilih dimensi 3 ∈ {3}. Maka kirim pesan melalui busur nomor 3.

Sekarang *simpul_kunjung* = 0110 dan *simpul_target* = 0110. Maka *himp_dim_beda* = ∅; Karena *himp_dim_beda* = ∅ , maka simpul target tercapai. Dengan kata lain, jalur terpendek dari simpul sumber 1001 ke simpul target 0110 melalui barisan simpul-simpul **1**001, 00**0**1, 001**1**, 0**0**10, 0110 dengan busur nomor 4,2,1,3.

6 KOMPLEKSITAS ALGORITMA

Untuk menghitung kompleksitas waktu algoritma, jumlah dari langkah-langkah routing digunakan sebagai ukuran. Asumsikan bahwa waktu mengambil satu langkah routing untuk mengirim sebuah pesan dari sebuah simpul ke simpul sebelahnya. Kasus pertama, jalur terpendek antara A dan B memerlukan waktu dengan panjang H(A,B). Kasus ini paling lama membutuhkan waktu log N, di mana $N = 2^n$. Kasus kedua, jaringan menjadi putus, memerlukan waktu paling lama N-1-F, di mana F adalah jumlah simpul faulty. Kasus ketiga, kemungkinan bisa terdapat jalur antara dua prosesor dengan panjang jalur terpendek ≥ n. Kasus ini membutuhkan waktu paling lama N-1-F. Kasus keempat, kemungkinan terjadi deadlock pada pengiriman pesan, sehingga harus dilakukan backtracking sampai kepada simpul yang mempunyai link alternatif lain menuju simpul target. Kasus ini membutuhkan waktu paling lama N-1-F. Dengan demikian kompleksitas waktu yang diperlukan algoritma untuk menyelesaikna problem mencari jalur terpendek pada incomplete-hypercube dimensi-*n* pada tulisan ini secara umum adalah O(N), dengan $N=2^n$.

7. DAFTAR PUSTAKA

[1] N.M. Allahverdi, SS Kahramanli dan K. Erciyes. *A Fault Tolerant Routing Algorithm Based On Cube Algebra For Hypercube Systems*. IEEE Trans. Computers (2000).

[2] Jean-Luc Baril, Vincent Vajnovzki. *Minimal change list for Lucas strings and some*

graph theoritic consequences. Elsevier. Theoritical Computer Science (346) 189-199 (2005).

[3] H.P. Katseff. *Incomplete Hypercube.* IEEE Transaction On Computer, vol 37 no.5 , pp. 604-607 (1988)

[4] J. Liu, W.J. Hsu. *Distributed Algorithms for Shortest-path, Deadlock-free Routing and Broadcasting in a Class of Interconnection Topologies.* IEEE Transaction On Computer, 0-186-2672-0. pp. 589-596 (1992)

[5] H. Qian, J. Wu. *Unicast, Multicast, and Broadcast in Enhanced Fibonacci Cubes*. IEEE Transaction On Computer, 0-8186-7180-7/95. pp. 158-161 (1995).

[6] S.H. Roosta. *Parallel Processing and Parallel Algorithms : Theory and Computation* New York : Springer Inc. (2000).

[7] Y. Saad, M. H. Schultz, *Topological properties of hypercubes*, IEEE Trans. Computers, 37, pp. 867-872. (1988)