ICN’s

The n-D hypercube (n-cube) contains $2^n$ nodes (processors).

The nodes are neighbors in the n-cube iff their end bit binary addresses differ in a single bit.

George Antoniou, CS/MSU
Neighbor nodes

Two nodes are neighbors in the n-cube iff their end bit binary addresses differ by a single bit.
Hypercubes

Hypercubes are designed recursively
2 - Cube

$2^2 = 4$ nodes
3 - Cube

$2^3 = 8$ nodes
4 - Cube

$2^4 = 16$ nodes
4 - Cube
Reduced hypercube
Folded hypercube
3-cube routing: data exchange

The nodes are coded as XYZ
Routing by inverting X

The nodes are coded as XYZ
Routing by inverting Y

The nodes are coded as XYZ
Routing by inverting Z

The nodes are coded as XYZ
The shortest path, between nodes, is determined by the Hamming distance.
Hemming distance

XOR
(source and destination)
and count the 1’s
Examples

010 -> 111 =
Examples

010 -> 111 = 010 XOR 111 = 101 = 2 hops
Examples

010 -> 110 =
Examples

010 -> 110 = 010 XOR 110 = 100 = 1 hops
Examples

010 -> 101 = 010 XOR 010 = 111 = 3 hops
Examples

010 -> 101 = 010 XOR 010 = 111 = 3 hops
Hypercube

- The hypercube can emulate (simulate) other ICN's
- The hypercube is a general purpose or universal ICN
Embedding

- How can we embed other ICN's into a hypercube?
Embedding a linear array

- How can we embed other ICN’s into a hypercube?
Algorithm

- Bits must differ by a single bit
- Use the binary Reflected Gray Code
Binary Reflected Code

- RGC(n): n-bit binary Reflected Gray Code

- \[ RGC(n) = [0 \cdot RGC(n-1), 1 \cdot RGC^{-1}(n-1)] \]

- If \( RGC(n-1) = 1 0 \)
  Then \( RGC^{-1}(n-1) = 0 1 \rightarrow\rightarrow\) reverse order

- \( RGC(0) = RGC^{-1}(0) = NULL \)
Linear array; $n = 1$

- RGC(n); n-bit binary Reflected Gray Code

$$\text{RGC}(n) = \begin{bmatrix} 0 & \text{RGC}(n-1) \\ 1 & \text{RGC}^{(-1)}(n-1) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \text{RGC}(1-1) \\ 1 & \text{RGC}^{(-1)}(1-1) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \text{RGC}(0) \\ 1 & \text{RGC}^{(-1)}(0) \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 \end{bmatrix}$$

or
1-cube/2-node array

- **RGC(n):** n-bit binary Reflected Gray Code

\[ RGC(n) = \begin{cases} 0 & RGC(n-1), \\ 1 & RGC^{\{-1\}}(n-1) \end{cases} \]

\[ = \begin{cases} 0 & RGC(1-1), \\ 1 & RGC^{\{-1\}}(1-1) \end{cases} \]

\[ = \begin{cases} 0 & RGC(0), \\ 1 & RGC^{\{-1\}}(0) \end{cases} \]

\[ = \begin{bmatrix} 0 & 1 \end{bmatrix} \]

or

1 - Cube

\[ \begin{array}{c}
0 \\
1
\end{array} \rightarrow 1 \quad \text{Cube} \]
Linear array; \( n = 2 \)

- \( RGC(n); \) n-bit binary Reflected Gray Code

\[
RGC(2) = [ 0 . RGC(2-1), 1 . RGC^{\{-1\}}(2-1) ]
\]

\[
= [ 0 . RGC(1) , 1 . RGC^{\{-1\}}(1) ]
\]

\[
= [ 0 . (0,1) , 1 . (1,0) ]
\]

\[
= [ 00, 01 , 11, 10 ]
\]

or
2-cube/4-node array

- $RGC(n)$: n-bit binary Reflected Gray Code

$$RGC(2) = [0 \cdot RGC(2-1), 1 \cdot RGC^{\{\neg\}}(2-1)]$$
  $$= [0 \cdot RGC(1), 1 \cdot RGC^{\{\neg\}}(1)]$$
  $$= [0 \cdot (0,1), 1 \cdot (1,0)]$$
  $$= [00, 01, 11, 10]$$
Linear array; \( n = 3 \)

- **RGC(n);** \( n \)-bit binary Reflected Gray Code

\[
\text{RGC}(3) = \begin{bmatrix}
0 \cdot \text{RGC}(3-1), & 1 \cdot \text{RGC}^{-1}(3-1)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0 \cdot \text{RGC}(2), & 1 \cdot \text{RGC}^{-1}(2)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0.(00,01,11,10), & 1.(10,11,01,00)
\end{bmatrix}
\]

\[
= \begin{bmatrix}
000,001,011,010,110,111,101,100
\end{bmatrix}
\]

**Addresses**

<table>
<thead>
<tr>
<th>ARRAY</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>HYPERCUBE</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Linear array - Hypercube

HYPERCUBE = 0 1 3 2 6 7 5 4
3-Cube/16-node array

HYPERCUBE = 0 1 3 2 6 7 5 4

3-cube
Embed an 8-leave Binary Tree into a 3-Cube
Embed an 8-leave Binary Tree into a 3-Cube

1. Choose the address of the parent to be the address of the left child.
Embed an 8-leave Binary Tree into a 3-Cube

Choose the address of the parent to be the address of the left child.
Embed an 8-leave Binary Tree into a 3-Cube

Height = n = 4
Nodes = 2^n-1

Choose the address of the parent to be the address of the left child.
Embed an 8-leave Binary Tree into a 3-Cube
Create Tables (connection) for each level

Level- 3

Level- 2

Level- 1

Level- 0
Table: Level -1

Level- 3

Level- 2

Level- 1

Level- 0

001 ->> 000
011 ->> 010
Table: Level -1

Level- 3

Level- 2

Level- 1

Level- 0

001 ->> 000
011 ->> 010
101 ->> 100
111 ->> 110
Table: Level -2

Level- 3

Level- 2

Level- 1

Level- 0

010 ->> 000
110 ->> 100
Table: Level -3

Level- 3

Level- 2

Level- 1

Level- 0

100 -> 000
Draw: Level - 1

- 001 -> 000
- 011 -> 010
- 101 -> 100
- 111 -> 110
Draw: Level - 2

- 001 ->> 000
- 011 ->> 010
- 101 ->> 100
- 111 ->> 110

- 010 ->> 000
- 110 ->> 100
Draw: Level - 3

- 001 -> 000
- 011 -> 010
- 101 -> 100
- 111 -> 110

- 010 -> 000
- 110 -> 100
- 100 -> 000
8-leave Tree/3-cube
8-leave Tree/3-cube