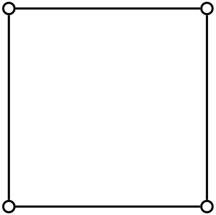
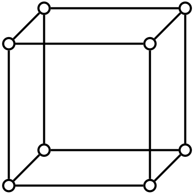
Distributed Memory Systems: Part III

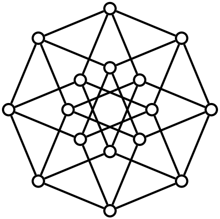
(a) 1D hypercube



(b) 2D hypercube



(c) 3D hypercube



(d) 4D hypercube

Communication Networks (revisited)

Some Remarks on the Hypercube

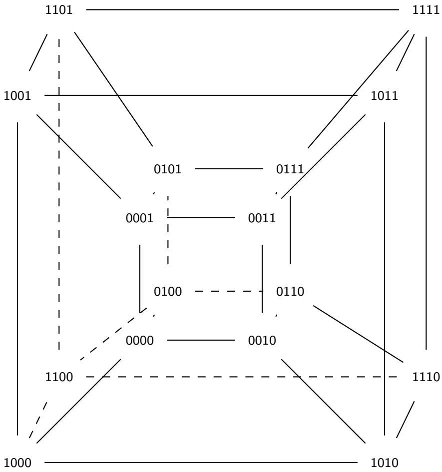


Figure: The hypercube network in 4d

Communication Networks (revisited)

Some Remarks on the Hypercube

We denote the nodes in the d-dimensional hypercube by d-tuples of bits, i.e. we use $n_1, \ldots, n_p \in \{0,1\}^d$. Let $a, b, c \in \{0,1\}^d$ and a_i, b_i, c_i the i-th bit positions. We denote by $\oplus$ the bitwise exclusive `or` operation, i.e.

$$a_1 \ldots a_d \oplus b_1 \ldots b_d = c_1 \ldots c_d$$

with

$$c_i = \begin{cases} 1 & \text{where } a_i \neq b_i, \\ 0 & \text{otherwise} \end{cases} \quad \text{for } 1 \leq i \leq d.$$

Communication Networks (revisited)

Some Remarks on the Hypercube

We denote the nodes in the d-dimensional hypercube by d-tuples of bits, i.e. we use $n_1, \ldots, n_p \in \{0, 1\}^d$. Let $a, b, c \in \{0, 1\}^d$ and a_i, b_i, c_i the i-th bit positions. We denote by $\oplus$ the bitwise exclusive `or` operation, i.e.

$$a_1 \ldots a_d \oplus b_1 \ldots b_d = c_1 \ldots c_d$$

with

$$c_i = \begin{cases} 1 & \text{where } a_i \neq b_i, \\ 0 & \text{otherwise} \end{cases} \quad \text{for } 1 \leq i \leq d.$$

Note that $\forall z \in \{0, 1\}^d$ then

$$00 \ldots 0 \oplus z = z,$$

and if $v, w \in \{0, 1\}^d$ differ in only a single bit, so do $v \oplus z$ and $w \oplus z$.

Properties of the Hypercube graph

- nodes are bit d-tuples,

Communication Networks (revisited)

Some Remarks on the Hypercube

Properties of the Hypercube graph

- nodes are bit d-tuples,
- each node has d links to other nodes

Properties of the Hypercube graph

- nodes are bit d-tuples,
- each node has d links to other nodes
- neighbors differ in a single bit position

Properties of the Hypercube graph

- nodes are bit d-tuples,
- each node has d links to other nodes
- neighbors differ in a single bit position
- the diameter of the graph (i.e., the length of longest path between two nodes) is $d = \log(p)$.

Definition (Spanning tree)

A spanning tree of a graph is a tree that

- picks one node of the graph as its root,
- contains all other nodes as nodes or leaves,
- has only edges that represent valid links in the graph.

Communication Networks (revisited)

Construction of Spanning Trees for Single Broadcasts

Construction Rules for root $00\dots0$

1. all root connections coincide with the links in the graph.

Communication Networks (revisited)

Construction of Spanning Trees for Single Broadcasts

Construction Rules for root $00\dots0$

1. all root connections coincide with the links in the graph.
2. children are generated by inverting a single bit right of the rightmost 1.

Construction Rules for root $00\dots0$

1. all root connections coincide with the links in the graph.
2. children are generated by inverting a single bit right of the rightmost 1.

The rules above imply

Communication Networks (revisited)

Construction of Spanning Trees for Single Broadcasts

Construction Rules for root $00\dots0$

1. all root connections coincide with the links in the graph.
2. children are generated by inverting a single bit right of the rightmost 1.

The rules above imply

- that all leave nodes end on a 1 bit,

Communication Networks (revisited)

Construction of Spanning Trees for Single Broadcasts

Construction Rules for root $00\dots0$

1. all root connections coincide with the links in the graph.
2. children are generated by inverting a single bit right of the rightmost 1.

The rules above imply

- that all leave nodes end on a 1 bit,
- the depth of the tree is $d+1$ since d bits are inverted on the path to the deepest leave $11\dots1$.

Communication Networks (revisited)

Construction of Spanning Trees for Single Broadcasts

Root nodes other than $00\dots0$

Spanning trees for other root nodes v are derived by replacing all nodes w by $w \oplus v$ in the entire tree for root $00\dots0$.

Root nodes other than $00\ldots0$

Spanning trees for other root nodes v are derived by replacing all nodes w by $w \oplus v$ in the entire tree for root $00\ldots0$.

Why is this the case? We noted above the properties of $\oplus$ that

- $00\ldots0$ is the neutral element, and
- v, w differ in only a single bit $\Rightarrow$ $v \oplus z$, $w \oplus z$ do so as well.

Thus, if (v, w) is a hypercube link, then $(v \oplus z, w \oplus z)$ is one as well.

Root nodes other than $00\dots0$

Spanning trees for other root nodes v are derived by replacing all nodes w by $w \oplus v$ in the entire tree for root $00\dots0$.

Why is this the case? We noted above the properties of $\oplus$ that

- $00\dots0$ is the neutral element, and
- v, w differ in only a single bit $\Rightarrow$ $v \oplus z$, $w \oplus z$ do so as well.

Thus, if (v, w) is a hypercube link, then $(v \oplus z, w \oplus z)$ is one as well.

Single Broadcast

The single broadcast can be implemented in $\Theta(\log p) = \Theta(d)$ successively descending through the spanning tree. It can also not be better than that since the diameter of the hypercube is d.

Scatter

A scatter operation needs to send out $p-1$ different messages along the d links of the root node. It can thus not be faster than $\lceil \frac{p-1}{d} \rceil$ time steps.

We will see in the following that this ist the time also needed for a multi broadcast. Since a single scatter can not be slower than that we immediately have that a scatter is $\Theta(\frac{p-1}{\log(p)}) = \Theta(\frac{p-1}{d})$.

Communication Networks (revisited)

Collision Avoiding Spanning Trees for Multi-Broadcast Operations

Problem

The single broadcast spanning trees for the 2^d nodes in the d-dimensional hypercube are not disjoint in the sense that each link is only used by a single operation in each time step if the multi-broadcast is treated as 2^d isolated single broadcasts.

Observation

It is thus mandatory to construct spanning trees such that all sets of edges used in a single time step by the different single broadcasts are disjoint.

Communication Networks (revisited)

Collision Avoiding Spanning Trees for Multi-Broadcast Operations

Definition

- The spanning tree for root node $t \in \{0,1\}^d$ is called T_t, and simply T_0 for $t = 00\ldots0$.
- The set of edges active in time step i for T_t is called $A_i(t)$

Construction

The sets of active edges for root node $t \in \{0,1\}^d$ may be constructed such that for any two edges (x, y) and (x', y') in $A_i(0)$ x, y and x', y' do not differ in the same bit position and the sets for the other root nodes are derived as

$$A_i(t) = \{(x \oplus t, y \oplus t) \mid (x, y) \in A_i\} \qquad \forall 1 \leq i \leq m,$$

where m is the total number of time steps required.

Observation

The set A_i of active edges in the i-th step can have at most d entries, since we only have d bit positions available in the node labels.

Main Idea:

Construct the sets A_i such that $|A_i| = d$ for $1 \leq i < m$ and $|A_m| \leq d$.

Since each of the $p = 2^d$ nodes in the tree has an incoming link, except the root, we have $2^d - 1$ edges in total that are distributed among the A_i, i.e.,

$$\left| \bigcup_{i=1}^{m} A_i \right| = 2^d - 1.$$

Since each of the $p = 2^d$ nodes in the tree has an incoming link, except the root, we have $2^d - 1$ edges in total that are distributed among the A_i, i.e.,

$$\left| \bigcup_{i=1}^{m} A_i \right| = 2^d - 1.$$

Since each of the $p = 2^d$ nodes in the tree has an incoming link, except the root, we have $2^d - 1$ edges in total that are distributed among the A_i, i.e.,

$$\left| \bigcup_{i=1}^{m} A_i \right| = 2^d - 1.$$

This immediately provides a first estimate for m:

$$m = \left\lceil \frac{2^d - 1}{d} \right\rceil$$

Communication Networks (revisited)

Collision Avoiding Spanning Trees for Multi-Broadcast Operations

Since each of the $p = 2^d$ nodes in the tree has an incoming link, except the root, we have $2^d - 1$ edges in total that are distributed among the A_i, i.e.,

$$\left|\bigcup_{i=1}^{m} A_i\right| = 2^d - 1.$$

This immediately provides a first estimate for m:

$$m = \left\lceil \frac{2^d - 1}{d} \right\rceil$$

Note that we can also not get better than that, since each node in the hypercube has to receive $2^d - 1$ messages from the other nodes across its d incoming links.

Definition

We collect some further notation:

- $N_k := \{t \in \{0,1\}^d \mid t \text{ has } k \text{ unit bits ans } d-k \text{ zero bits.}\}$
- These sets have

$$n_k := |N_k| = \begin{pmatrix} d \\ k \end{pmatrix} = \frac{d!}{k!(d-k)!}$$

  elements.
- The N_k are further subdivided into m_k equivalence classes $R_{k1}, \dots, R_{km_k}$ with respect to left rotation. They are ordered by rightmost concentration of the unit bits, i.e., R_{k1} is the class containing $(0^{d-k}1^k)$.

Definition

We collect some further notation:

- $N_k := \{t \in \{0,1\}^d \mid t \text{ has } k \text{ unit bits ans } d-k \text{ zero bits.}\}$
- These sets have

$$n_k := |N_k| = \begin{pmatrix} d \\ k \end{pmatrix} = \frac{d!}{k!(d-k)!}$$

  elements.
- The N_k are further subdivided into m_k equivalence classes $R_{k1}, \dots, R_{km_k}$ with respect to left rotation. They are ordered by rightmost concentration of the unit bits, i.e., R_{k1} is the class containing $(0^{d-k}1^k)$.

Definition

We collect some further notation:

- $N_k := \{t \in \{0,1\}^d \mid t \text{ has } k \text{ unit bits ans } d-k \text{ zero bits.}\}$
- These sets have
$$n_k := |N_k| = \begin{pmatrix} d \\ k \end{pmatrix} = \frac{d!}{k!(d-k)!}$$
elements.
- The N_k are further subdivided into m_k equivalence classes $R_{k1}, \dots, R_{km_k}$ with respect to left rotation. They are ordered by rightmost concentration of the unit bits, i.e., R_{k1} is the class containing $(0^{d-k}1^k)$.

Definition

We collect some further notation:

- The elements in the equivalence classes can be ordered by rightmost concentration of unit bits as well.
- $n(t)$ is the global number of node t in this order.
- $m(t) = 1 + [n(t) - 1 \mod d]$ is t's local number of inside the equivalence class.

Communication Networks (revisited)

Communication Routing on the Hypercube

Definition

We collect some further notation:

- The elements in the equivalence classes can be ordered by rightmost concentration of unit bits as well.
- $n(t)$ is the global number of node t in this order.
- $m(t) = 1 + [n(t) - 1 \mod d]$ is t's local number of inside the equivalence class.

Communication Networks (revisited)

Communication Routing on the Hypercube

Definition

We collect some further notation:

- The elements in the equivalence classes can be ordered by rightmost concentration of unit bits as well.
- $n(t)$ is the global number of node t in this order.
- $m(t) = 1 + [n(t) - 1 \mod d]$ is t's local number of inside the equivalence class.

Let us denote the sets of destination nodes in A_i by E_i. Then we set:

$$
\begin{aligned}
E_0 &= \{00\dots0\} & \\
E_i &= \{t \in \{0,1\}^d \mid (i-1)d+1 \le n(t) \le id\} & 1 \le i < m \\
E_m &= \{t \in \{0,1\}^d \mid (m-1)d+1 \le n(t) \le 2^d-1\} &
\end{aligned}
$$

The set of active edges are then constructed by the rules:

1. connect $t \in E_i$ to start node t' with the $m(t)$th bit inverted,
2. if $t = 11\dots1$ and $m(t) = d$ connect to $t' = 101\dots1$ instead.

Communication Networks (revisited)

Communication Routing on the Hypercube

By construction in each step the tree uses d edges and all sets $A_i(t)$ for the different t are disjoint. Thus, all 2^d single broadcasts can be performed simultaneously and the multi-broadcast can be done in $\Theta(\frac{p-1}{d})$.

Note that although the d-hypercube has only $\frac{d}{2} \cdot 2^d$ edges we can use $d \cdot 2^d$ links in the graph due to the assumption of bidirectional communication.