Time and Bit Optimal Broadcasting on Anonymous Unoriented Hypercubes

Stefan Dobrev[*], Peter Ružička[*], Gerard Tel[†]

Abstract

We consider broadcasting on asynchronous anonymous totally unoriented N-node hypercubes. First we generalize a technique, introduced in [3], for partial broadcasting and orientation. Using this technique we develop a broadcasting algorithm on unoriented hypercubes that uses only linear number of bits and runs in optimal time. This gives a positive answer to the question raised in [7] whether $O(N)$ bits are sufficient for broadcasting on unoriented N-node hypercubes. It is also an improvement over the previous algorithms from [3, 1] both in time and bit complexities.

As an application of broadcasting, we develop an algorithm for computing identities of all nodes in unoriented hypercubes with linear number of messages. (The question was stated in [7]). This allows every subset of nodes (such as covers, independent sets, etc) to be determined in $O(N)$ messages.

1 Introduction

Broadcasting is one of the most fundamental tasks in parallel and distributed computing. One node of the network, called the *source*, has a message which has to be transmitted to all other nodes.

First, consider arbitrary networks. On the one hand, anonymity of the network does not have significant impact on the message complexity of the broadcasting problem, because the symmetry is already broken. Distinct identities of nodes can be computed from the source in asymptotically the same message complexity as the broadcasting in anonymous networks [10]. Thus broadcasting on non–anonymous networks cannot be done asymptotically better than in the anonymous case.

On the other hand, the message complexity of broadcasting strongly depends on the amount of topological information available at nodes. If links of a network are globally consistently labeled, forming *sense of direction* [4, 11], broadcasting is possible using only

[*]Institute of Informatics, Faculty of Mathematics and Physics, Comenius University, Bratislava, Slovakia. E-mail: {dobrev, ruzicka}@dcs.fmph.uniba.sk. Research supported by EU Grant No. INCO-COP 96-0195 "ALTEC-KIT" and VEGA 1/4315/97. Contact author: Peter Ružička

[†]Department of Computer Science, Utrecht University, Utrecht, The Netherlands. E-mail: gerard@cs.uu.nl. Research supported by ESPRIT Long Term Research Project 20244 (Project ALCOM IT: *Algorithms and Complexity in Information Technology.*)

linear number of messages w.r.t. the number of nodes [5]. But if a network is unoriented (i.e. without *sense of direction*), then the lower bound for broadcasting is linear in the number of edges [5]. This lower bound is achievable by the naive broadcasting algorithm, in which a node immediately spreads the message to all neighbours except to the one from which it received it.

While this strategy cannot be improved on general networks, broadcasting algorithms might exploit the knowledge of special topologies to reduce the number of messages. For example, on the complete unoriented network the broadcasting is trivially accomplished by sending a message from the source to all its neighbours. On other topologies the situation is not so simple. It was stated as an open question (see [11], cf. also [6]) whether there exists a broadcasting algorithm on unoriented N–node hypercubes using $O(N)$ messages in the worst case. This question was positively answered only recently in [3] and [1], where two different linear message algorithms for broadcasting on unoriented anonymous hypercubes were presented. However, both algorithms use messages of size $O(\log N)$, bringing the total bit complexity to $O(N \log N)$. A question was raised in [7] whether $O(N)$ is indeed sufficient for bit complexity of broadcasting on unoriented hypercubes. Moreover, these algorithms work in non–optimal time. While the algorithm from [1, 2] needs $O(\log^2 N)$ time units, time complexity of the algorithm from [3] is $O(N/\log^4 N)$.

We will present an algorithm for broadcasting on unoriented anonymous hypercubes with the total bit complexity $O(N)$ in the worst case, thus giving the positive answer to the above question. Our algorithm runs in optimal time $\log N$, thus improving both bit and time complexities of the previous algorithms.

As an application, we design an algorithm for computing identities of all nodes in unoriented hypercubes with linear number of messages, which gives an answer to the question raised in [7]. This also allows every subset of nodes (such as covers, independent sets, etc) to be determined in $O(N)$ messages. And last but not least, this gives a linear message simulation of hypercubes with natural vertex identities on a model of anonymous hypercubes with a leader. Hence, linear message broadcasting algorithm on hypercubes with precomputed identities [8] works also on unoriented anonymous hypercubes in the same complexity.

This paper is organized as follows. In section 2 we give graph theoretic preliminaries concerning the hypercube and present computational model. In section 3 we introduce a technique for efficient partial broadcasting and orientation on unoriented hypercubes. In section 4 we present new broadcasting algorithm, prove its correctness and estimate its time and communication complexities. In section 5 we present a linear message algorithm for computing identities of processors.

2 Preliminaries

2.1 The hypercube

An n-dimensional hypercube is an undirected graph Q_n consisting of $N = 2^n$ vertices and $1/2Nn$ edges.

Vertices are represented by binary strings of length n. For two vertices, their *Hamming*

distance is the number of positions on which they differ. There is an edge between two vertices iff their Hamming distance is 1. Q_n is regular of degree n and has diameter n.

The following notation will be used:

- e_i for $i = 1, \ldots, n$ is a bit vector representing $0^{i-1}10^{n-i}$. Vectors e_i are called *unit vectors.*
- v_i denotes the i–bit of v, where v is n-bit binary string.
- **and**, **or** and $\oplus$ denote bitwise *and*, *or* and *exclusive or* on n bit binary strings, respectively.
- $Q(v; e_{i_1}, \ldots, e_{i_l}), 1 \leq l \leq n$, is a sub-hypercube containing vertices $v \oplus \alpha_1 e_{i_1} \oplus \ldots \oplus \alpha_l e_{i_l}$, where $\alpha_i \in \{0, 1\}$ and e_{i_j} are distinct unit vectors. $e_{i_1}, \ldots, e_{i_l}$ are called *generators* of $Q(v; e_{i_1}, \ldots, e_{i_l})$.
- Let E be any subset of the set of unit vectors and E^C be its complement, then $Q(v; E)$ and $Q(v; E^C)$ are called orthogonal.

Proposition 2.1 *The following properties on Q_n hold:*

1. *Every pair of distinct vertices of Q_n have either 0 or 2 common neighbours.*
2. *Vertices u and v in Q_n have 2 common neighbours iff their Hamming distance is 2. If u and v have the same number of 1's in their representation, these neighbours are u **and** v and u **or** v.*
3. *Each vertex in Q_n is unambiguously determined by any triple of its neighbours.*
4. *For all v in Q_n and all l-tuples $(e_{i_1}, \ldots, e_{i_l}), 1 \leq l \leq n$, of distinct unit vectors there exists a unique isomorphism σ of $Q(v; e_{i_1}, \ldots, e_{i_l})$ to $Q(0^n; e_1, \ldots, e_l)$ such that v is mapped to 0^n and each e_{i_j} is mapped to e_j.*

A *dominating set* of a graph G is a set D of vertices such that every vertex of G belongs to D or is adjacent to a vertex from D. We say that the dominating set D is *perfect* iff each vertex has exactly one neighbour (including itself) in D.

Proposition 2.2 *If $n = 2^k - 1$ then there exists a perfect dominating set of size $2^n/(n+1)$ in Q_n.*

Proof. Let X be the identity of a vertex x written as the binary row matrix $n \times 1$. Take $D_n = \{x | AX = (0, \ldots, 0)^T\}$, where A is the $k \times n$ binary matrix, in which the i-th column is the binary representation of the number i, for $i = 1, \ldots, n$. All computations are in Z_2 (modulo 2).

Let $AX = Y$ for some vertex x. Interpret Y as the binary number y. If $y = 0$, then $x \in D_n$, otherwise consider $x' = x + e_y$. Then x' is the neighbour of x that lies in D_n: $AX' = A(X + E_y) = AX + AE_y = Y + Y = (0, \ldots, 0)^T$. □

Proposition 2.3 *For each n there exists a dominating set D_n of size at most $2^{n+1}/(n+1)$ in Q_n.*

Proof. Take $D_n = D_n^0 \cap D_n^1$, where $D_n^0 = \{x|AX = (0,\ldots,0)^T\}$ and $D_n^1 = \{x|AX = (1,0,\ldots,0)^T\}$. A is the matrix from the previous proposition.

Let $AX = Y$ for some vertex x. Interpret Y as the binary number y. If $y < n$, then follow as in the previous proposition. If $y > n$, then flip the most significant bit of y to get y'. Now $y' < n$, since $y < 2n$. Consider $x' = x + e_{y'}$. Then $AX' = AX + AE_{y'} = Y + Y' = (1,0,\ldots,0)^T$, thus x has the neighbour x' in D_n^1. □

2.2 The model

The computational model is a standard model of asynchronous distributed computing on point–to–point networks [10]. Every message will be delivered in a finite but unbounded time. FIFO requirements on links are not necessary.

The underlying topology of the network is anonymous unoriented hypercube graph Q_n. Anonymity means that processors do not have distinct identities. n - bit strings representing vertices in the previous subsection will be called *identities* of processors. At the beginning, only the source processor knows its identity – 0^n.

Each processor can distinguish its links by uninterpreted labels $e_1,\ldots,e_n$. However, this labeling is arbitrary at each processor and labels are thus without any topological meaning.

We are primarily interested in *bit – complexity*, i.e. the number of bits communicated in the worst case. The worst case refers to the worst messages delays and to adversary decisions concerning choices of yet unused links – if a processor sends a message on an unused link, the actual link (from the set of yet unused links) is chosen by the adversary, as all yet unused links look alike to the sender.

Time complexity is defined as the total execution time of the algorithm in the worst case, assuming that delays on links are bounded by one time unit.

We are considering the problem of *broadcasting*. At the beginning there is a single active processor – source of an information. Other processors will become active only after receiving a message. At the end of the computation we require each processor to have received the information. The cost of transmitting the information is not counted into the bit complexity, we count only the overhead of the broadcasting algorithm. The more appropriate notion for our problem would be *wake up* with single initiator.

3 Partial broadcasting and orientation

Definition 3.1 *Let M be a subset of vertices of Q_n and s be some vertex in M. Define $M_i^s = \{v|v \in M, d(v,s) = i\}$, where $d(x,y)$ is Hamming distance of x and y.*[1] *Let $\mathcal{L}_M = \max\{i|M_i \neq \emptyset\}$.*

We say that M is a computable mask from s iff for all $v \in M$ at least one of the following conditions is true:

[1] We will use M_i instead of M_i^s when s is obvious from the context.

1. $v \in M_0$ *($v = s$),*
2. $v \in M_1$ *(v is a neighbour of s),*
3. $v \in M_k$ *and v has at least three neighbours in M_{k-1},*
4. $v \in M_k$ *and v has two neighbours x and y in M_{k-1}. Let v' be the second common neighbour of x and y. Then $v' \in M_{k-2}$.*

If $s = 0^n$, we say that M is a computable mask.

The following statements follow easily from the previous definition:

- Union of computable masks from s is a computable mask from s.
- $Q(s; E)$ is a computable mask from s for any set of generators E.

3.1 The basic partial broadcasting and orientation

The following algorithm $\mathcal{A}1$ is used to perform partial broadcasting and orientation on a given computable mask M.

Algorithm $\mathcal{A}1$:

Mask M and source vertex s are fixed and known to all vertices.

Messages used:
(*You are:*, x) and (*I'm*, x)
where x is n bit binary string.

Variables used at vertex v:

- $Name_v$: n – bit binary string containing the identity of v. Initial value is *empty*, only s knows its identity.
- $Label_v(h)$: labels of links incident to v, initial value is *empty*. Exceptions are links from s leading to vertices in M, which have their correct labels.

Algorithm in the starting vertex s:

For all links h:
  if $s \oplus Label_s(h) \in M$ then
    Send(*You are:*, $s \oplus Label_s(h)$) on link h;

Algorithm in a vertex v:

Upon receiving a message (*You are:*, x) on link h:

$Name_v := x$;
$Label_v(h) := x \oplus s$;
if $Name_v \in M$ then
  Send(*I'm*, $Name_v$) on all links except h;

Upon receiving a message ($I'm$, x) **on link** h_1**:**

if $Name_v = empty$ then
  Wait for the second message ($I'm$, y); {arrived on link h_2}
  Compute identities w_1 and w_2 of the two common neighbours
    of x and y.
  if $x \in M_{k-1}$ and $y \in M_{k-1}$ and $w_1 \in M_{k-2}$ and $w_2 \in M_k$ then
    $Name_v := w_2$;
  else
    Wait for the third message ($I'm$, z); {arrived on link h_3}
    $Name_v :=$ the unique vertex that has neighbours x, y and z;
  fi

  if $Name_v \in M$ then
    Wait for $I'm$ messages from all your predecessors in M;
    Send($I'm$, $Name_v$) to all neighbours;
  fi
fi

Each vertex that has computed its name computes labels of links within M:

For each link h on which ($I'm$, x) has been received:
  $Label_v(h) := x \oplus Name_v$;

Proposition 3.2 *If M is computable from s and s knows labels of all links leading to its neighbours in M, then the algorithm $\mathcal{A}1$ computes*

1. *identities of all vertices in M,*
2. *link labels for each link between vertices in M.*

Proof.

1. We will prove Proposition 3.2 by induction on the distance from s.

   A vertex v computes its identity in the following cases:

   - After receiving message *You are:* : In this case identity is correctly computed because the source s knows labels of links to vertices in M_1. This is the first step of induction.
   - After receiving two messages ($I'm$, x) and ($I'm$, y) from x and y in the same layer: This case applies only if w_1, the common neighbour of x and y in the previous layer, is also in M. By induction, the computed names of x and y are correct, and because each vertex waits for messages from all its predecessors before announcing its name, w_1 has already computed its name (or $w_1 = s$). It follows that w_2 is the correct name for v.

- After receiving three *I'm* messages : Its identity is unambiguously given in this case. Identities that came in these messages are correct by induction hypothesis.

We have proved that vertices correctly compute their identities. The fact that indeed each vertex $\in M$ computes its identity can be easily proved by induction on the distance from s from the computability of M.

2. This follows from the fact that each vertex in M computes its identity and announces it, together with handling of these messages by the algorithm.

□

Proposition 3.3 *The algorithm $\mathcal{A}1$ uses at most $n|M|$ messages of size $O(n)$ bits.*

Proof. Each vertex in M sends at most n messages of size $O(n)$ bits each. Vertices outside M do not send messages. □

Proposition 3.4 *Time complexity of $\mathcal{A}1$ is $\mathcal{L}_M + 1$.*

Proof. By induction on the distance from the s: Vertex $v \in M_k$ computes its name not later than at the time k. The $+1$ term stands for computing the labels of the last links. □

3.2 Bit–efficient partial broadcasting algorithm

The algorithm $\mathcal{A}1$ does not communicate efficiently, because each vertex v sends its whole identity to all its neighbours. During the computation only some vertices really need the identity of v. The basic scheme is to send just *Hello* messages and only vertices which are really interested in your full identity will ask you for it. A vertex v needs to learn identities of only two or three of its neighbours to be able to compute its identity.

Algorithm $\mathcal{A}2$:
Messages used – *Hello*, *Who are you?* and messages of $\mathcal{A}1$
Variables used at vertex v – as in $\mathcal{A}1$
Algorithm in the starting vertex s is the same as in $\mathcal{A}1$
Algorithm in a vertex v:
Upon receiving a message (*You are:***, x) on link h:**

$Name_v := x$;
if $Name_v \in M$ then
    Send(*Hello*) on all links except h;

Upon receiving the first *Hello* message on link h_1:

Wait for the second *Hello* message; {arrived on link h_2}
Send *Who are you?* on h_1 and h_2;
Wait for (*I'm*, x) and (*I'm*, y) on h_1 and h_2, respectively.
Compute identities w_1 and w_2 of the two common neighbours of x and y.
if $x \in M_{k-1}$ and $y \in M_{k-1}$ and $w_1 \in M_{k-2}$ and $w_2 \in M_k$ then
    $Name_v := w_2$;
else
    Wait for the third *Hello* message ; {arrived on link h_3}
    Send *Who are you?* on h_3;
    Wait for (*I'm*, z) on h_3;
    Compute $Name_v$ as the unique vertex that has neighbours x, y and z;
fi

if $Name_v \in M$ then
    Wait until you receive *Hello* messages from all your predecessors in M;
    (Wait until you have received k *Hello* messages, where k is the number of your predecessors in M.)
    Send(*Hello*) to all neighbours;

Upon receiving *Who are you?* **message on link** h**:**

    Send(*I'm*, $Name_v$) on h

Proposition 3.5 *If M is a computable mask from s and s knows labels of all links leading to its neighbours in M, then the algorithm $\mathcal{A}2$ correctly computes identities of vertices in M.*

Proof. Follows the same line as the proof of $\mathcal{A}1$. The only substantial difference is that $\mathcal{A}2$ at vertex v tests the condition that it received *Hello* messages from all its predecessors in M by counting these messages, while $\mathcal{A}1$ can do this by testing identities. Note that $\mathcal{A}2$ can't be fooled by *Hello* messages from successors of v, because successor must receive a *Hello* from v to proceed and send its *Hello*. □

Proposition 3.6 *The algorithm $\mathcal{A}2$ uses $O(n|M|)$ messages containing totally $O(n(|M|+|M'|))$ bits, where M' is the set of all vertices of $Q_n - M$ that have at least two neighbours in M.*

Proof. We charge constant-bit messages (*Hello* and *Who are you*) to sender and long ((*I'm*, x) and messages (*You are:*, x)) to receiver. Each vertex in $M \cup M'$ is charged $O(n)$ bits, because each vertex in M sends $O(n)$ messages and each vertex in $M \cup M'$ receives at most 3 long messages. □

Note that if $M = Q(v; E)$ then $M' = \emptyset$.

Proposition 3.7 *Time complexity of $\mathcal{A}2$ is $1 + 3(\mathcal{L}_M - 1)$.*

Proof. By induction on the distance from s, similarly as for $\mathcal{A}1$. $\mathcal{A}2$ will proceed with at most 3 time units per one layer, with the exception of the first layer which takes at most one time unit. □

3.3 Applications – optimal computable masks for special target sets

General scheme of broadcasting from the vertex s to some target set of vertices T is the following:

- Choose a set M, $M \supset T$, such that M is computable from s. It is desirable to make M as small as possible.
- Make sure that s knows labels of links leading to its neighbours in M.
- Use $\mathcal{A}1$ ($\mathcal{A}2$) to broadcast on M.

One particular case is to inform a single vertex v at the distance d about its identity. (See e.g. the algorithm FarSend() in [3].) Because of Proposition 2.1 it is sufficient to consider the case $s = 0^n$, $v = 1^d0^{n-d}$.

Lemma 3.8 (FarSend) *There exists a computable mask M of size $1 + d(d+1)/2$ which contains the vertex $v = 1^d0^{n-d}$.*

Proof. It is easy to verify that $M = \{0^i1^k0^{n-i-k} \mid 0 \leq k \leq d,\ i + k \leq d\}$ is such a mask. □

Let $M_{|d}$ denote $M \cap \{0,1\}^d0^{n-d}$. The mask used in the previous lemma is indeed optimal.

Lemma 3.9 *Let M be any computable mask that contains vertex 1^d0^{n-d}, then $|M_{|d}| \geq 1 + d(d+1)/2$.*

Proof. By induction on d. For $d = 1$ the condition trivially holds, because M must contain both the source and the target vertex.

General case: The vertex 1^d0^{n-d} must have at least two predecessors x and y in M_{d-1}. We may assume $x = 1^{d-1}0^{n-d+1}$ and $y = 01^{d-1}0^{n-d}$. From induction hypothesis applied on x we get $|M_{|d-1}| \geq 1 + d(d-1)/2$. It holds $y \notin M_{|d-1}$, because y has 1 at position d. Since identity of a vertex is determined as bitwise **or** of its predecessors (this holds only for the source 0^n), y has a predecessor y' with 1 at position d. We may apply this argument on y' and so on until we get down to $0^{d-1}10^{n-d}$. This means that there is a chain of $d-1$ vertices $y, y', y'', \ldots, 0^{d-1}10^{n-d}$ that are not in $M_{|d-1}$. However, all these vertices must be in $M_{|d}$, otherwise $y \notin M_{|d}$. Hence we get $|M_{|d}| \geq 1 + |M_{|d-1}| + d - 1 \geq 1 + 1 + d(d-1)/2 + d - 1 = 1 + d(d+1)/2$. □

Another useful task is to orient a vertex v consistently with the source s. (See e.g. the algorithm FarOrient() in [3].) While this can be trivially done using a mask of size $O(nd^2)$, the following lemma shows how to do it using a mask of size $O(nd)$.

Lemma 3.10 (FarOrient) *There exists a computable mask M_{Orient} of size $nd+n-2d+2$ containing the target set $S(1^d0^{n-d}; 1)$. Here $S(x; r)$ denotes the sphere with centre x and radius r.*

Proof. Take $M_{Orient} = M' \cup M'' \cup M'''$ where $M' = \{0^i 1^k 0^{n-i-k} |\ 0 \leq i+k \leq d\}$, $M'' = \{1^k 0^{d-k} 0^j 1 0^{n-d-j-1} |\ 0 \leq k \leq d,\ 0 \leq j < n-d\}$ and $M''' = \{1^i 0^j 1^{d-i-j} 0^{n-d} |\ i > 0,\ j > 0,\ i+j < d\}$. □

This mask is close to optimality.

Lemma 3.11 *If M is a computable mask containing $S(1^d 0^{n-d}; 1)$, then $|M| \geq nd + n - d(d+1)/2 + 1$.*

Proof. The proof uses the same ideas as the proof of Lemma 3.9. □

This lower bound differs from the upper bound only by the term $|M'''|$.

Combining previous results with $d \leq n$, we get that any vertex can be reached (or oriented) using $O(n^3)$ messages and $O(n^4)$ bits, which is optimal when using $\mathcal{A}1$.

4 The broadcasting algorithm

We will follow the outline of the algorithm from [1]. The main difference lies in application of $\mathcal{A}1$ and $\mathcal{A}2$ for partial broadcasting on sub–hypercubes instead of "jo–jo" technique from [1, 2].

The broadcasting algorithm $\mathcal{BR}$ works in two stages:

> **STAGE I:** $\mathcal{A}1$ is launched with the mask $M1 = \cup_{v \in D_k} M_{Orient}(v)$ from the source, where D_k is a perfect dominating set of $Q(0^n; e_1, \ldots, e_k)$. k is chosen to be of the form $2^r - 1$ for some integer r, while being as close as possible to $n/2$. Clearly $n/3 \leq k \leq 2n/3$.
>
> **STAGE II:** $\mathcal{A}2$ is launched with the mask $M(v) = \{x | x \in Q_n, x_1 x_2 \ldots x_k = v_1 v_2 \ldots v_k\}$ from each $v \in D_k$.

$\mathcal{A}1$ and $\mathcal{A}2$ were presented using implicit knowledge of M and s. Now we use more invocations, so we must specify what is implicit now and how we use it. Each vertex knows (from knowing n) $M1$, $M = \cup M(v)$ and D_k. Messages of different stages are marked by a stage mark (i.e. by O(1) bits). We cannot afford to mark messages of different invocations of $\mathcal{A}2$ in Stage II, but this is not the problem since these invocations do not interfere.

There are no vertices receiving messages from two different invocations of $\mathcal{A}2$, because there is no vertex in $Q(0^n; e_1, \ldots, e_k)$ with two neighbours in D_k. (D_k is a perfect dominating set.)

Each vertex of $M(v)$ can decide to which invocation of $\mathcal{A}2$ it belongs simply by looking at the first k bits of the first ($I'm$, x) message it received, learning which v and $M(v)$ to use.

Proposition 4.1 *(Correctness) The algorithm $\mathcal{BR}$ is a broadcasting algorithm on unoriented hypercubes.*

Proof. First, note that the mask $M1$ is computable from 0^n and each $M(v)$ is computable from v.

It is sufficient to show that each vertex v has a neighbour in some $M(v)$ – that means that it receives *Hello* message.

Let $v \in Q_n$ be arbitrary vertex. Take $v' = v_1 v_2 \dots v_k 0^{n-k}$. Since $v' \in Q(0^n; e_1, \dots, e_k)$, it has a neighbour $x \in D_k$. Therefore v has a neighbour $x' = x_1 \dots x_k v_{k+1} \dots v_n \in M(x)$. □

Proposition 4.2 *The algorithm $\mathcal{BR}$ broadcasts on unoriented N– vertex hypercubes using $O(N)$ bits.*

Proof. Total bit cost of Stage I can be bounded by $|D_k|$ times the bit cost of orienting one vertex, which is $O(\frac{2^k}{k+1} \cdot n^4) \in o(N)$.

Total bit cost of Stage II can be bounded by the number of invocations of $\mathcal{A}2$ times the cost of one such invocation:

$O(\frac{2^k}{k+1} \cdot n \cdot 2^{n-k}) = O(nN/(k+1)) = O(N)$. □

Time complexity of Stage I can be bounded by $k+1$. Similarly, Stage II works in time $3(n-k)-1$, summing to the total time $3n - 2k < 7/3n$.

4.1 Bit–optimal broadcasting in time n

Note that we can apply $\mathcal{A}1$ instead of $\mathcal{A}2$ to achieve the algorithm that reaches optimal time n, but its bit complexity will be $N \log N$.

However, we could choose $k = n-4$ and use the previous approach. This would result in an algorithm running in time $k + 1 + 3(n-k) - 1 = n + 8$, being still bit optimal. If the root v at Stage II is at the distance $d < n - 12$ from the initiator, all vertices in $M(v)$ and their neighbours will be informed in time $d + 1 + 3(n-k) - 1 = n - 12 + 12 = n$. There are $O(n^{12}.(n+1).2^4) = O(n^{13})$ vertices that would be informed later than in time n. These vertices can be informed directly in time n, using $\mathcal{A}1$ with a mask containing them all, in cost $O(n^4)$ per vertex, with the total added cost $O(n^{17}) \in o(N)$.

The problem is that there may not be perfect dominating set D_{n-4} of Q_{n-4}. But we do not need perfect dominating set as far as we get dominating set which can be partitioned into the constant number of sets $D^0, \dots, D^c$ such that invocations of $\mathcal{A}2$ from vertices of D_i do not interfere. All we need is to mark an invocation of $\mathcal{A}2$ from v by the index of D^i in which v lies.

One such dominating set is the dominating set from the Proposition 2.3.

5 Computing identities of all vertices

Bernard Mans [7] proposed an interesting question whether it is possible to compute identities of all vertices in Q_n using only linear number of messages. This could be of special interest, since the orientation of the hypercube requires $\Omega(N \log N)$ messages (see [9]). The knowledge of identities of vertices allows to compute an orientation locally only where it is needed, thus saving some communication.

It is indeed possible to compute an identity of every vertex using only linear number of messages. For each vertex it is sufficient to learn identities of its three distinct neighbours.

This can be done using three broadcasts with three disjoint dominating sets. Instead of using $\mathcal{A}2$ as in bit-optimal broadcasting, $\mathcal{A}1$ will be used, so each vertex in the mask will inform all neighbours about its identity. If one wants to run it in optimal time n, messages should be appropriately marked by the index of particular broadcasting to avoid collisions. All what is needed is to choose three disjoint perfect dominating sets D_1, D_2 and D_3 of $Q(s; e_1, \ldots, e_k)$. D_2 can be obtained by flipping the first bit of vertices in D_1 and similarly D_3 by flipping the second bit. It is easy to see that D_2 and D_3 are perfect dominating sets of $Q(s; e_1, \ldots, e_k)$ (because D_1 is) and are disjoint. (Because D_1 is perfect dominating set and thus all its vertices are at least three links apart.)

Putting these facts together, we computed identity of each vertex using $O(N)$ messages of size $O(n)$ bits each. Since algorithm $\mathcal{A}1$ was used, the computation time is n.

Total bit complexity is $O(N \log N)$ bits. However, this is optimal for this problem, if we require $O(N)$ number of messages. As the number of messages is bounded by $O(N)$, there must be a constant d such that at least $N/2$ vertices communicate on at most d links. Each of these $N/2$ vertices should compute different identity, based only on the communication history. We get that there must be $N/2$ different communication histories on the constant number of links for each of these $N/2$ vertices, thus bringing up $\Omega(N \log N)$ lower bound.

6 Conclusions

We have shown how to reduce the problem of reaching specified target set T to the problem of finding the smallest computable mask that contains T. This is nontrivial problem and it can challenging for many practical target sets. Another interesting question that arises here is how the structure of T determines the size of the mask M.

We have seen that broadcasting on unoriented hypercubes can be done both in optimal time and optimal bit complexities. It would be interesting to find more classes of graphs (another example is the class of chordal rings with $\log N$ chords leading to closest neighbours in the ring [8]) with asymptotically more edges than vertices that will allow such an efficient broadcasting. This may be helpful in solving more general (and difficult) question: What are the structural properties that the class of graphs must possess to allow linear message broadcasting? What are the properties that enforce non-linear lower bound for broadcasting?

References

[1] Diks, K. – Kranakis, E. – Pelc, A.: *Broadcasting in Unlabeled Networks.* Départment d'Informatique, Université du Québec á Hull, Technical Report, RR 96/12-5, December 1996.

[2] Diks, K. – Dobrev, S. – Kranakis, E. – Pelc, A. – Ružička, P.: *Broadcasting in Unlabeled Hypercubes with Linear Number of Messages.* Information Processing Letters 66, 1998, pp. 181–186.

[3] Dobrev, S. – Ružička, P.: *Linear Broadcasting and $N \log\log N$ Election in Unoriented Hypercubes.* Proc. of the 4th International Colloquium on Structural Information and Communication Complexity (SIROCCO'97), Carleton Scientific, 1997, pp. 53–68.

[4] Flocchini, P. – Mans, B. – Santoro, N.: *Sense of Direction: Formal Definitions and Properties.* Proc. of the 1st International Colloquium on Structural Information and Communication Complexity (SIROCCO'94), Carleton Press, 1995, pp. 9–34.

[5] Flocchini, P. – Mans, B. – Santoro, N.: *On the Impact of Sense of Direction on Communication Complexity.* Information Processing Letters 63 (1), 1997, pp. 23–31.

[6] Mans, B.: *Broadcast, Traversal and Election in Unlabeled Hypercube.* Proc. of the 3rd International Colloquium on Structural Information and Communication Complexity (SIROCCO'96), Carleton Press, Siena, Italy, June 1996, pp. 333–334.

[7] Mans, B.: *Sense of Direction and Applications.* Invited talk at the Research School'97 on "Compact Routing and Sense of Direction", Siena, Italy, June 1997.

[8] Peleg, D.: Personal communication at the Research School'97 on "Compact Routing and Sense of Direction", Siena, Italy, June 1997.

[9] Tel, G.: *Network Orientation.* International Journal of Foundations of Computer Science 5, 1994, pp. 23–57.

[10] Tel, G.: *Introduction to Distributed Algorithms.* Cambridge University Press, Cambridge, 1994.

[11] Tel, G.: *Sense of Direction in Processor Networks.* In: SOFSEM'95, Theory and Practise of Informatics, LNCS 1012, Springer–Verlag, 1995, pp. 50–82.