International Journal of Networking and Computing – `www.ijnc.org`
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 7, Number 2, pages 336–348, July 2017

On the Cost of Waking Up

Stefan Dobrev
Slovak Academy of Sciences
Bratislava, Slovakia

Rastislav Královič
Comenius University
Bratislava, Slovakia.

Nicola Santoro
Carleton University
Ottawa, Canada

Received: February 15, 2017
Accepted: April 5, 2017
Communicated by Susumu Matsumae

Abstract

Often, in a distributed system, a task must be performed in which all entities must be involved; however only some of them are active, while the others are inactive, unaware of the new computation that has to take place. In these situations, all entities must become active, a task known as Wake-Up. It is not difficult to see that Broadcast is just the special case of the Wake-Up problem, when there is only one initially active entity. Both problems can be solved with the same trivial but expensive solution: `Flooding`. More efficient broadcast protocols exist for some classes of dense interconnection networks. The research question we examine is whether also wake-up can be performed significantly better in three classes of regular interconnection networks: hypercubes, complete networks, and regular complete bipartite graphs.

In a d-dimensional *hypercube* network of n nodes, the cost of broadcasting is $\Theta(n)$ even if the edge labeling is arbitrary and the network is asynchronous. We show that, instead, wake-up requires $\Omega(n \log n)$ message transmissions in the worst case, even if the network is synchronous and has sense of direction. Similarly, in a regular *complete bipartite* network $K_{p,p}$ of $n = 2p$ anonymous entities the cost of broadcasting is $\Theta(n)$ even if the edge labeling is arbitrary and the network is asynchronous; instead, we show that wake-up requires $\Omega(n^2)$ message transmissions in the worst case, even if the network is synchronous and has sense of direction.

In a *complete* network K_n of n entities, the cost of broadcasting is minimal: $n-1$ message transmissions suffice even if the entities are anonymous. In this paper we prove that the cost of wake-up is order of magnitude higher. In the case of anonymous entities, $\Omega(n^2)$ message transmissions are needed in the worst case, even if the network is fully synchronous and has sense of direction. In the case of entities with distinct ids, $\Omega(n \log n)$ transmissions need to be performed and the bound is tight. This shows that, when the entities have Ids, Wake-Up is computationally as costly as the apparently more complex Election problem.

Keywords: distributed algorithms, message complexity, sense of direction, complete networks, complete bipartite graphs, hypercubes, wake-up, reset, election

1 Introduction

1.1 Framework

Consider a distributed system composed of a set V of computational entities connected by a set $E \subseteq V \times V$ of communication links; the graph $G = (V, E)$ describes the resulting network topology. The entities communicate by sending and receiving messages over the links, and cooperate to achieve a common goal (e.g., solve a problem, perform a task, etc). In such a system, we are often faced with the following situation: A task has to be performed in which all entities must be involved; however only some of them are aware (because of a spontaneous event, or having locally terminated a previous computation) that the (new) computation must be *started.* A similar situation occurs when, during the execution of a computation, some entities detect the occurrence of an anomalous condition and independently decide that the ongoing computation need to be *restarted.* In these situations, all entities must become aware and active within finite time. Obviously this process can only be started by the entities which are aware already, the *initiators*; notice that, however, these entities do *not* necessarily know which other entities (if any) are initiators, nor their location.

This problem, called Wake-Up, Reset, or Restart, is one of the most basic primitive tasks, employed in a variety of contexts, such as self-stabilization and checkpoint-recovery (e.g., see [3, 4, 5, 9, 10, 11, 16, 19, 20, 21]), and it is related to several other computations. For example, in synchronous systems, it is directly related to the Unison problem requiring the global synchronization of local clocks (e.g., [2, 11, 16]). It is not difficult to see the relationship between Broadcast and Wake-Up: a broadcast is a wake-up with only one initially aware entity; conversely, a wake-up is a broadcast with possibly many initiators (i.e., more than one entity initially has the information). In other words, Broadcast is just a special case of the Wake-Up problem.

The cost of a wake-up is the total number of message transmissions performed during the entire process in the worst case. This cost depends on many factors, first and foremost the topological structure of the network, the number and location of the initiators, but also on a variety of other factors such as the whether the entities are anonymous or have distinct ids, whether the system is synchronous or asynchronous, whether the local edge labelings provide a global sense of direction or is arbitrary, etc.

Interestingly, regardless of any of those factors, both Broadcast and Wake-Up can be solved by the same standard technique, `Flooding`, which consists of two simple rules:

(1) if initiator, send the message to all your neighbours;

(2) if not initiator, receiving the message for the first time, forward it to all the other neighbours. The cost of `Flooding` is $O(m)$ messages, regardless of the number of initiators, where m is the number of links in the network. Although reasonable in sparse networks (i.e., when $m = O(n)$), this cost becomes prohibitive in dense networks. Indeed, much more efficient Broadcast protocols exist for some classes of dense interconnection networks, such as hypercubes, complete bipartite graphs, and complete networks.

The research question we examine is whether also Wake-Up can be performed significantly better in specific classes of networks, and under what conditions. In this paper we examine three classes of interconnection networks: hypercubes, complete bipartite graphs, and complete networks, and establish tight bounds on the complexity of Wake-Up in such networks.

1.2 Contributions

In a d-dimensional *hypercube* network H_d of $n = 2^d$ anonymous entities, the cost of broadcasting is minimal, $O(n)$, even if the edge labeling is arbitrary [7, 8]. We show that, instead, to perform a wake-up requires $n \log n$ message transmissions in the worst case, even if the network is fully synchronous and there is sense of direction. The bound is tight since it can be achieved by just flooding the network.

Similarly, in a regular *complete bipartite* network $K_{p,p}$ of $n = 2p$ anonymous entities, the cost of broadcasting is $O(n)$ even if the edge labeling is arbitrary and the network is asynchronous: the initiator sends the message to all its p neighbours, only one of those messages containing a special marker; the neighbour receiving the marker will then forward the message to all its other $p - 1$

neighbours. We show that, instead, Wake-Up may require $\Omega(n^2)$ message transmissions in the worst case, even if the network is fully synchronous and has sense of direction. The bound is tight since it can be achieved by just flooding the network.

Also in a *complete* network K_n of n entities, the cost of broadcasting is minimal: $n-1$ message transmissions suffice even if the entities are anonymous. In this paper, we show that the cost of Wake-Up is order of magnitude higher. More precisely, in the case of anonymous entities, $\frac{1}{2}(n^2-2n)$ message transmissions may be needed, even if the network is fully synchronous and there is sense of direction. Also in this case the bound is tight since it can be achieved by just flooding the network.

If the entities have unique Ids, then it is well known that a much more complex problem, Election, can be solved in an asynchronous complete graph with $O(n \log n)$ message transmissions [1, 6, 15, 17, 22]. We show that the apparently much simpler Wake-Up problem cannot be solved with a better complexity; in fact, at least $\frac{1}{2} n \log n$ message transmissions need to be performed in the worst case. Since any Election protocol solves also Wake-Up, the lower bound is asymptotically tight, and it indicates that Wake-Up is computationally as expensive as Election.

2 Definitions and Terminology

Consider a distributed system composed of a set V of computational entities connected by a set $E \subseteq V \times V$ of bidirectional edges (or links); the graph $G = (V, E)$, assumed to be simple, describes the communication topology of the system. Given a node $x \in V$, let $E(x) = \{(x, y) \in E\}$ denote the set of edges incident on x. Associated to each node $x \in V$ is a local injective labeling $\lambda_x : E(x) \to \mathcal{L}$, where $\mathcal{L}$ is a set of labels called port numbers; notice that associated to each edge (x, y) are two labels: $\lambda_x(x, y)$ at x, and $\lambda_y(x, y)$ at y. Let Λ be the extension of λ from edges to walks, that is: $\Lambda([(x_0, x_1), (x_1, x_2), \ldots, (x_{m-1}, x_m)]) = [\lambda_{x_0}(x_0, x_1), \ldots, \lambda_{x_{m-1}}(x_{m-1}, x_m)]$.

An important and powerful property is that of *sense of direction* (e.g., [12, 13, 14]); to define it, we need the concept of consistent coding and decoding. A *coding function* f in (G, λ) is a function which maps sequences of labels associated to walks in G to a set $\mathcal{N}$ of names; a coding function f is *consistent* for λ if and only if $\forall x, y, z \in V, \forall \pi_1 \in P[x, z], \pi_2 \in P[x, y]$,

$$f(\Lambda(\pi_1)) = f(\Lambda(\pi_2)) \iff y = z.$$

where $P[x, y]$ denotes the set of walks from x to y. Given a consistent coding function f, a *consistent decoding function* for f is any function $h : \mathcal{L} \times \mathcal{N} \to \mathcal{N}$ such that $\forall x, z \in V, \forall (x, y) \in E(x), \forall \pi \in P[y, z], \pi' \in P[x, z]$,

$$h(\lambda_x(x, y), f(\Lambda(\pi))) = f(\Lambda(\pi')).$$

We say that (G, λ) has *sense of direction* iff there exist a consistent coding function f for λ and a consistent decoding function for f.

Some important examples of labelings that are sense of direction in any networks, in hypercubes, and in complete bipartite graphs, respectively, are described next. A *chordal* labeling of a graph $G = (V, E)$, with $|V| = n$, is defined by fixing a cyclic ordering of the nodes $< v_0, v_1, ..., v_{n-1} >$ and, for each link (v_i, v_j), assigning label $(i - j)\bmod n$ at v_i and label $(j - i)\bmod n$ at v_j; (thus, $\mathcal{L} \subseteq \{1, \ldots, n-1\}$); see Figure 1. With this labeling the network has sense of direction [13]. Consider for example, the set of names $\mathcal{N} = \{0, 1, \ldots, n-1\}$ and the coding function f defined as follows: for any walk $\pi = [(x_0, x_1), (x_1, x_2), \ldots, (x_{m-1}, x_m)]$

$$f(\Lambda(\pi)) = (\sum_{i=0}^{m-1} \lambda_{x_i}(x_i, x_{i+1})) \bmod n$$

It is easy to verify that f is indeed consistent and has a consistent decoding.

The traditional *dimensional* labeling of a d-dimensional hypercube H_d of $n = 2^d$ nodes, where each edge is consistently labeled with one of the hypercube's dimensions (i.e., $\mathcal{L} = \{1, \ldots, d\}$, see Figure 2), creates a sense of direction [13]. For example, the set of names is $\mathcal{N} \subset \mathcal{L}^*$, and computing the coding of the sequence of labels corresponding to a walk consists in applying the following two

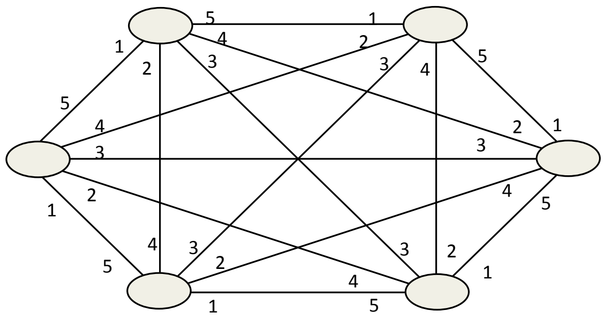

Figure 1: Complete network K_6 with chordal labeling.

steps: (1) remove from the sequence of labels any pairs of identical labels; and (2) lexicographically sort the labels still in the sequence. It is easy to verify that f is indeed consistent and has a consistent decoding.

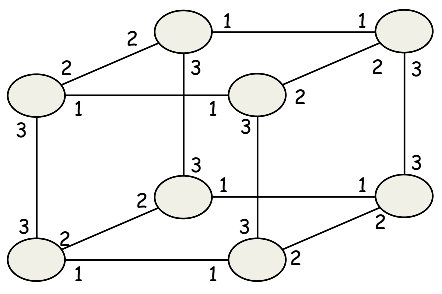

Figure 2: Hypercube H_3 with dimensional labeling.

In a regular complete bipartite graph $K_{p,p}$ of $n = 2p$ nodes $\{a_1, a_2, \ldots, a_p, b_1, b_2, \ldots, b_p\}$, the set of edges is just $\{(a_i, b_j) : 1 \leq i, j \leq p\}$. In the standard *cyclic* labeling of $K_{p,p}$, edge (a_i, b_j) is assigned label $((j - i) \bmod p) + 1$ at a_i, and label $((i - j) \bmod p) + 1$ at b_j (see Figure 3 for an example). Given a walk $\pi = [(x_0, x_1), (x_1, x_2), \ldots, (x_{m-1}, x_m)]$, let $|\pi| = m$ denote its length. Consider now the function

$$g(\Lambda(\pi)) = (\sum_{i=0}^{m-1} (\lambda_{x_i}(x_i, x_{i+1}) - 1)) \bmod p$$

Give two walks π_1, π_2 starting from the same node x, it is not difficult to verify that they end in the same node y if and only if $g(\Lambda(\pi_1)) = g(\Lambda(\pi_2))$ *and* $|\pi_1|$ and $|\pi_2|$ have the same parity (i.e., they are both even or both odd). The corresponding coding function has a consistent decoding, thus, with this labeling, the network has sense of direction.

The entities communicate by sending finite sequence of bits, called messages. The primitive operation is "**send to** l", where l is one of the local port numbers. Once a node x issues such a command, the message will be received in its integrity and in finite time at the neighbouring node y connected to the link (x, y) labeled l at x.

All entities are behaviourally identical in that they obey the same protocol. They are said to be *anonymous* if they do not use distinguished identifiers in their protocols, and *with Ids* if each entity has a distinguished value from a totally ordered set, and it uses it in its protocols. In systems with

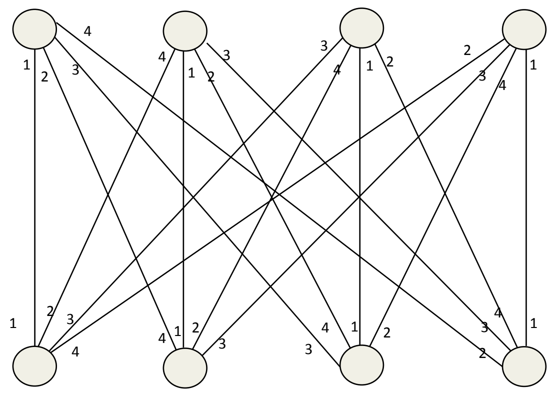
Figure 3: Complete bipartite graph $K_{4,4}$ with cyclic labeling.

Ids, each entity is aware of its identifier, but not necessarily of those of its neighbours; of course, this information can be acquired through communication.

In a *synchronous* system, there is a common unit of time (e.g., all local clocks tick simultaneously), it takes one unit of time for a transmitted message to reach the neighbouring destination, and processing time is negligible (i.e., instantaneous). In an *asynchronous* system, there is no common notion of time, and processing and transmissions delays are (finite but) unpredictable.

Initially, some nodes are *active*, the others *inactive*; the number and locations of the *active* entities is a priori unknown. The `Wake-Up` problem requires all entities to become *active* within finite time.

The cost measure is total number of message transmissions.

3 Distributed Wake-Up in H_d

In a d-dimensional *hypercube* network H_d of $n = 2^d$ anonymous entities, the cost of broadcasting is $\Theta(n)$ even if the edge labeling is arbitrary [7, 8]. By contrast, we show that Wake-Up requires order of magnitude more message transmissions, even if the network is synchronous and has sense of direction.

Theorem 1 *Wake-Up in hypercubes of n anonymous nodes requires the transmission of at least $n \log n$ messages in the worst case. This result holds even if the class of networks is restricted to synchronous hypercubes with sense of direction.*

Proof By contradiction, let $\mathcal{P}$ be a synchronous protocol that correctly solves Wake-Up in every hypercube of n anonymous nodes and does so with less than $n \log n$ message transmissions in hypercubes with sense of direction. Let H be a hypercube of n anonymous entities with the classical *dimensional* labeling; thus H has sense of direction.

Consider the synchronous execution $\mathcal{E}$ of P in H where all entities are initially *active*. Since the entities are anonymous and because of the symmetry of the labeling, at each step every entity will be in the same state, experience the same events (reception of specific messages from specific ports), and perform the same actions (transmission of specific messages through specific ports). In particular, since in the dimensional labeling an edge has the same label at both end nodes, whenever an entity sends a message on an incident edge, it will also receive the same message from that edge. Since, by assumption, $\mathcal{P}$ is correct, at some time t all entities will terminate. Let $L(x)$ be the set of port numbers on which a message was sent to or received from entity x during execution E'. Because of the symmetry, $L(x) = L(y) = L$ for all entities x and y; hence the total number M of message transmissions is $M \geq n|L|$. Since $M < n \log n$ by assumption, then $k = |L| < \log n = d$.

Note that the set of labels L induces 2^{d-k} disjoint subcubes of dimension $k < d$. Let H' be one such subcube.

Consider now the synchronous execution $\mathcal{E}'$ of P in H where only the entities of H' are initially *active*, the others are *inactive*. Clearly, for the entities of H', the two executions are indistinguishable, since they do not interact with entities outside H'. Hence, at time t, they will terminate; however, at time t all the other entities are still *inactive* and they will never become *active*, contradicting the correctness of the protocol.

□

As a consequence, the obvious technique of `Flooding` provides a simple and generic solution that is worst-case *optimal*; furthermore, it works with the same cost also in asynchronous hypercubes with arbitrary labelings.

4 Distributed Wake-Up in $K_{p,p}$

Consider the regular *complete bipartite* graph $K_{p,p}$ of $n = 2p$ entities. In such a network the solution to BROADCAST is rather simple: the sole initiator sends the message to all its p neighbours, only one of those messages containing a special marker; the neighbour receiving the marker will then forward the message (without marker) to all its other $p-1$ neighbours. Regardless of asynchrony and arbitrary edge-labeling, its cost of $2p-1 = n-1$ message transmissions is clearly optimal.

By contrast, we show that WAKE-UP requires orders of magnitude more message transmissions, even if the network is synchronous and has sense of direction.

Theorem 2 *Wake-Up in a regular complete bipartite graph $K_{p,p}$ of $n = 2p$ anonymous entities requires at least $\frac{n^2}{4}$ message transmissions in the worst case. This result holds even if the class of networks is restricted to be synchronous and with sense of direction.*

Proof By contradiction, let $\mathcal{P}$ be a protocol that correctly solves the wake-up problem and does so with less than $\frac{1}{4}n^2$ message transmissions in every regular complete bipartite network of size n with sense of direction.

Let G be a regular complete bipartite network of $n = 2p$ entities, $A \cup B$, where $A = \{a_1, a_2, \ldots, a_p\}$ and $B = \{b_1, b_2, \ldots, b_p\}$, with cyclic sense of direction. Thus, edge (a_i, b_j) is labeled $(j-i)\texttt{mod}p + 1$ at a_i, and $(i-j)\texttt{mod}p + 1$ at b_j; notice that, if an edge is labeled j at one end, it will be labeled $(2-j)$ $\texttt{mod}$ p at the other end.

Consider the fully synchronous execution $\mathcal{E}$ of $\mathcal{P}$ in G in which all entities start the protocol simultaneously and proceeds in synchronous steps. Since the entities are anonymous and because of the symmetry of the labeling, at each step every entity will be in the same state, experience the same events (reception of specific messages from specific ports) at the same time, and perform the same actions (transmission of specific messages through specific ports).

In particular, if x sends a message via port number j at time t, so will everybody else; and all of them will receive it from port number $(2-j)\texttt{mod}p$ at time $t+1$. Since protocol $\mathcal{P}$ is correct by assumption, within a finite number of steps, say at time $\hat{t}$, all the entities will simultaneously terminate. Let $S(x)$ be the set of port numbers on which a message was sent by entity x during this execution. Because of the symmetry, $S(x) = S(y) = S$ for all entities x and y; hence the total number M of message transmissions is $M = n|S|$. Let $R(x)$ be the set of port numbers on which a message was received by entity x during this execution; again, because of the symmetry, $R(x) = R(y) = R$ for all entities x and y; also clearly $|R| = |S|$.

Since, by assumption, in every execution of $\mathcal{P}$ there are fewer than $\frac{n^2}{4}$ message transmissions, then $M = n|S| < \frac{n^2}{4}$; that is, $|S| < \frac{p}{2}$. Thus, for the set $L = S \cup R$ of ports on which a message was sent or received in execution $\mathcal{E}$ we have $|L| = |R \cup S| < p$.

Let $L = \{l_1, ..., l_k\}$, where w.l.g. $l_i < l_{i+1}$ $(1 \le i < k)$, and let $\bar{L} = \{1, ..., p\} \setminus L = \{l_{k+1,...,}l_p\}$. Because of the cyclic labeling and of the symmetry of transmissions/receptions in execution $\mathcal{E}$ (if x sends a message via port number j at time t, so will everybody else; and all of them will receive it from port number $(2-j)$ $\texttt{mod}$ n at time $t+1$), we have that

$$l_i = \begin{cases} (2 - l_{k-i+1})\texttt{mod}p & (1 \le i \le \lfloor k/2 \rfloor) \quad if\ 1 \in \bar{L} \\ (2 - l_{k-i+2})\texttt{mod}p & (2 \le i \le \lceil k/2 \rceil) \quad if\ 1 \in L \end{cases}$$

Let us now construct a regular complete bipartite network G' on the same set of nodes with the edge labeling defined below. Let $A' = \{a_1, ..., a_k\}$, $A'' = \{a_{k+1}, ..., a_p\}$, $B' = \{b_1, ..., b_k\}$, and $B'' = \{b_{k+1}, ..., b_p\}$; since $k < p$, A'' and B'' are not empty. Then for $(a_i, b_j) \in A \times B$, we assign label $\lambda_{a_i}(a_i, b_j)$ at a_i and label $\lambda_{b_j}(a_i, b_j)$ at b_j, where

$$\lambda_{a_i}(a_i, b_j) = \begin{cases} l_{(j-i)\texttt{mod}p+1} & if\ 1 \leq i, j \leq k \\ l_j & if\ 1 \leq i \leq k, \quad k < j \leq p \\ (j-i)\texttt{mod}p + 1 & if\ k < i \leq p \end{cases}$$

and

$$\lambda_{b_j}(a_i, b_j) = \begin{cases} l_{(i-j)\texttt{mod}p+1} & if\ 1 \leq i, j \leq k \\ l_i & if\ 1 \leq j \leq k, \quad k < i \leq p \\ (i-j)\texttt{mod}p + 1 & if\ k < j \leq p \end{cases}$$

It is easy to verify that all the local labeling functions λ_x are injective.

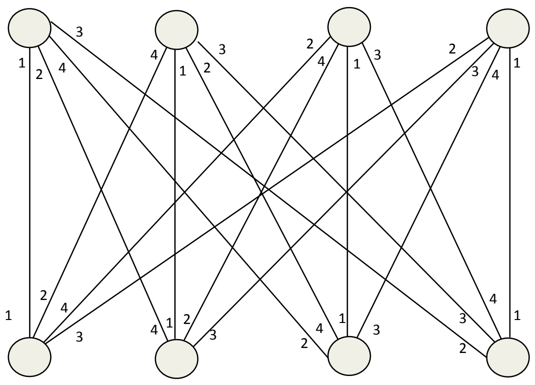

Figure 4: Proof of Theorem 2: G' with $L = \{1, 2, 4\}$ and $\bar{L} = \{3\}$.

Let $V' = A' \cup B'$. Observe that the subgraph $K_{k,k}(V')$ induced by the set of the initiator nodes is a regular complete bipartite graph on $n' = 2k$ nodes; the edges in $K_{k,k}(V')$ are locally labeled with the port numbers in L, while the edges connecting to the other nodes are labeled with the port numbers in $\bar{L}$ (see Figure 4).

Consider now the synchronous execution $\mathcal{E}'$ of $\mathcal{P}$ in G' in which only the entities in $V' = A' \cup B'$ are initially *active*. From the point of view of these initiators, everything in this execution happens exactly as if they were in execution $\mathcal{E}$ in G: messages will be sent to and received exactly from the same ports in L in the same steps in both executions. In particular, none of these entities will send a message outside $K_{k,k}(V')$; hence, none of the entities in $A' \cup B'$ will receive any message and thus none of them will send any message to the initiators; hence no initiator will receive a message from outside $K_{k,k}(V')$. Therefore, the initiators will act as if they are in G and the execution is $\mathcal{E}$; thus, at time $\hat{t}$, the initiators will all terminate the execution of the protocol. However the $n - n' = 2(p - k)$ entities in $A' \cup B'$, initially inactive, will never become *active*, contradicting the correctness of the protocol. □

In other words, any correct wake-up protocol for regular complete bipartite anonymous networks will always require at least $\frac{n^2}{4}$ message transmissions in the worst case, even if the system is synchronous. This also implies that the `Flooding` technique, which provides a solution that uses at most $\frac{n^2}{2}$ messages in such networks, is worst-case *optimal* within a factor of two; furthermore, it works also in asynchronous networks with arbitrary labelings.

5 Distributed Wake-Up in K_n

Consider now a *complete* network K_n of n entities. In such a network, Broadcast is trivial: the sole initiator just sends its message to all neighbours. Regardless of asynchrony, anonymity and arbitrary edge-labeling, its cost of $n-1$ message transmissions is clearly optimal.

In this section we show that the cost of WAKE-UP is order of magnitude higher. Depending on whether the network is anonymous or with Ids, different tight lower-bounds are established.

5.1 Anonymous Entities

Consider the case when the entities are anonymous.

Theorem 3 *Wake-Up in complete networks of n anonymous entities requires at least $\frac{1}{2}(n^2-2n)$ message transmissions in the worst case. This result holds even if the class of networks is restricted to be synchronous and with sense of direction.*

Proof By contradiction, let $\mathcal{P}$ be a protocol that correctly solves the wake-up problem and does so with less than $\frac{1}{2}n^2$ message transmissions in every complete network of size n with sense of direction.

Let G be a complete network of n entities $V=\{x_0,...,x_{n-1}\}$ with chordal sense of direction; thus, the edge (x_i,x_j) is labeled $(j-i)\mathtt{mod}n$ at x_i and $(i-j)\mathtt{mod}n$ at x_j. Consider the fully synchronous execution $\mathcal{E}$ of $\mathcal{P}$ in G in which every entity starts the protocol simultaneously and proceeds in synchronous steps. Since the entities are anonymous and because of the symmetry of the labeling, at each step every entity will be in the same state, experience the same events (reception of specific messages from specific ports), and perform the same actions (transmission of specific messages through specific ports). In fact, if x sends a message via port number j at time t, so will everybody else; and, because of the chordal labeling, all of them will receive it from port number $n-j$ at time $t+1$. Since protocol $\mathcal{P}$ is assumed to be correct, all the entities will simultaneously terminate within a finite number of steps, say at time $\hat{t}$.

Let $S(x)$ be the set of port numbers on which at least one message was sent by entity x during execution $\mathcal{E}$; because of the symmetry, $S(x)=S(y)=S$ for all entities x and y. The total number of messages transmitted by protocol $\mathcal{P}$ in execution $\mathcal{E}$ is thus at least $M\geq n|S|$. Since, by assumption, in every execution of $\mathcal{P}$ in G there are fewer than $\frac{1}{2}(n^2-2n)$ message transmissions, then $n|S|\leq M<\frac{1}{2}(n^2-2n)$; that is

$$|S|<\frac{1}{2}(n-2).$$

Let $R(x)$ be the set of port numbers on which at least one message was received by entity x during execution $\mathcal{E}$; because of the symmetry, $R(x)=R(y)=R$ for all entities x and y. Because the labeling is chordal, $R=\{n-l : l\in S\}$; thus, $|R|=|S|$. Let $L=S\cup R$ and $k=|L|$; then

$$k<n-2.$$

Let $L=\{l_1,...,l_k\}$, where w.l.g. $l_i<l_{i+1}$ $(1\leq i<k)$. Construct now a complete network G' on the same set of nodes with a different edge labeling; the construction differs depending on whether n is odd or even.

(*Case n odd.*) Let $\bar{L}=\mathcal{Z}_n\setminus L$, let $X=\{x_{l_0},x_{l_1},\ldots,x_{l_k}\}$ where $l_0=0$, and let $\bar{X}=V\setminus X$. Notice that, since $k<n-2$, then $|\bar{X}|=n-k>2$. The edge labeling is constructed as follows:

1. For $0\leq i<j\leq k$, label the edge (x_{l_i},x_{l_j}) with label l_{j-i} at x_{l_i} and label l_{k-j+i} at x_{l_j}.
2. Label any other edge (x_i,x_j) with label i at x_j and with label j at x_i.

(*Case n even.*) Let $L^+=L\cup\{n/2\}$ if k is even, $L^+=L$ otherwise; let $\bar{L}^+=\mathcal{Z}_n\setminus L^+$. Let $X=\{x_{l_0},x_{l_1},\ldots,x_{l_{k^+}}\}$, where $l_0=0$ and $k^+=|L^+|\leq k+1$; and let $\bar{X}=V\setminus X$; notice that, since by assumption $k<n-2$, then $|\bar{X}|=n-k^+\geq n-k-1\geq 2$. Let $Y=\{(y_1,z_1),(y_2,z_2),...,(y_p,z_p)\}$, where $p=|\bar{X}|/2$, be a maximal matching of the entities in $\bar{X}$. The edge labeling is constructed as follows (e.g., see Figure 5):

1. For $0\leq i<j\leq k$, label the edge (x_{l_i},x_{l_j}) with label l_{j-i} at x_{l_i} and label l_{k-j+i} at x_{l_j}.

2. Label any edge $(y,z) \in Y$ of the matching with label $n/2$ at both y and z.
3. Label any other edge (x_i, x_j) with label i at x_j and label j at x_i.

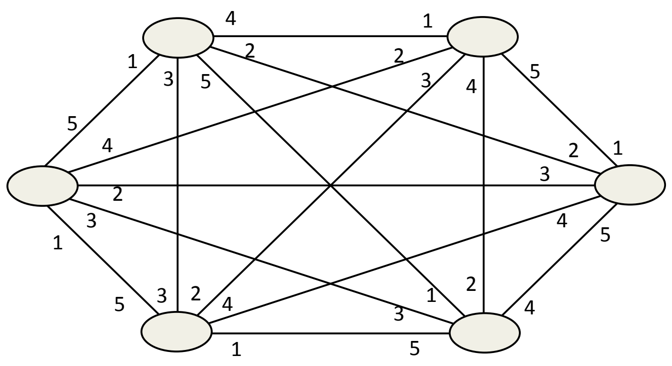

Figure 5: Proof of Theorem 3: G' with $L^+ = \{1,3,5\}$ and $\bar{L}^+ = \{2,4\}$.

It is easy to verify that, in both constructions, all the local labeling functions λ_{x_i} are injective.

Consider now the synchronous execution $\mathcal{E}'$ of $\mathcal{P}$ in G' in which only the entities in X are initially *active*. Observe that the subgraph $K(X)$ induced by X is a clique. From the point of view of these initiators, everything in this execution happens exactly as if they were in execution $\mathcal{E}$ in G: messages will be sent and received exactly from the same ports in the same steps in both executions. In particular, none of these entities will send a message outside $K(X)$; hence, none of the entities in $\bar{X}$ will receive any message and thus none of them will send any message to the initiators; hence no initiator will receive a message from outside $K(X)$. Therefore, the initiators will act as if they are in G and the execution is $\mathcal{E}$; thus, at time $\hat{t}$, the initiators will all terminate the execution of the protocol. However the $n-k \geq 2$ entities in $\bar{X}$ are still *inactive* and they will never become *active*, contradicting the correctness of the protocol. □

In other words, any correct wake-up protocol for complete anonymous networks will always require at least $\Omega(n^2)$ message transmissions in the worst case, even if the system is synchronous. This also implies that the obvious technique of `Flooding` provides a solution that is *asymptotically optimal*, and that works also in asynchronous networks with arbitrary labelings.

5.2 Entities with Unique Ids

Consider now the case when each entity $x \in V$ has a distinguished value $Id(x)$ from a totally ordered set $\mathcal{I}$. The presence of these unique values and the fact that they are drawn from a totally ordered set allows a protocol not only to differenciate topologically identical nodes but also to use them in decisions made by the protocol (e.g., the choice of the port number to use next is made as a function of the value of the id). A protocol that exploits only the properties of the total order of $\mathcal{I}$ is sometimes called comparison-based.

If the entities have unique Ids, then it is well known that a much more complex problem, ELECTION, can be solved asynchronously with $O(n \log n)$ transmissions of messages, each containing a constant number of Ids, by comparison-based protocols ([1, 6, 15, 17, 22]). The apparently much simpler WAKE-UP problem cannot be solved with a better complexity; in fact, the proofs that $\Omega(n \log n)$ messages are needed for solving ELECTION [1, 18] in reality prove that that number of messages are needed for WAKE-UP. Following is a new direct proof of the $\Omega(n \log n)$ worst case complexity of WAKE-UP in the asynchronous case, even for non-comparison based solutions.

Theorem 4 *In asynchronous complete networks with arbitrary labeling, wake-up requires the transmission of at least* $\frac{1}{2}\lfloor \log n \rfloor 2^{\lfloor \log n \rfloor}$ *messages in the worst case, even if the entities are non-anonymous.*

Proof To prove this theorem we consider a game between the protocol $\mathcal{P}$, an arbitrary solution protocol, non necessarily comparison-based, and an *adversary* $\mathcal{A}$ on a complete graph G on n nodes $V = \{x_0, ..., x_{n-1}\}$, each node $x \in V$ with a distinct value $id(x)$. To prove the Theorem, we will show that, with its decisions, the adversary can force an execution $\mathcal{E}$ of $\mathcal{P}$ in G where the number of transmitted messages is as claimed.

The power of the adversary $\mathcal{A}$ is the following : (i) it decides the activation time of each entity; (ii) it decides when a transmitted message arrives (it must be within finite time); (iii) it decides the duration of the execution of each operation at the nodes (must be finite); and, more importantly, (iv) at each node, it decides which incident link is assigned a specific port number, from $\{1, ..., n-1\}$. Since the adversary controls transmission and execution delays, it can enforce that, in the execution of the protocol, each entity sends one message at a time, and only after its previous message has arrived; we will focus on this type of executions.

Given a subset $V' \subseteq V$ of the nodes, let $E'(t) \subseteq V' \times V'$ be the set of edges between the nodes in V' where at least one message has been received (in either direction), from the start of the execution up to and including time t; if the graph $(V', E'(t))$ is connected, then we will say that V' is a *connected component at time t*.

The overall strategy of the adversary $\mathcal{A}$ is composed of two processes: *Creation* and *Control* of connected components, described below.

Process *Component Creation*

1. To construct a new connected component C of *level* 0, $\mathcal{A}$ selects a still inactive node x and makes it *active*; x starts its execution of the protocol, and $C = \{x\}$. If there are no inactive nodes, the adversary creates a virtual empty component; i.e., $C = \emptyset$.

2. To construct a connected component C of *level* $i > 0$:

   (a) $\mathcal{A}$ creates two disjoint connected components A and B of level $i - 1$.

   (b) If both A and B are non-empty, then:

      i. $\mathcal{A}$ decides labels and controls delays, using process *Control*, until there are two nodes, x in A and y in B, that, having communicated (by receiving or sending a message) with all the other nodes in their component, execute a "**send to** l_x" and a "**send to** l_y" operation, respectively, where l_x is a still unused port number of x (i.e., no message was sent on nor received from that port), and l_y is an unused port number of y.

      ii. It assigns to edge (x, y) label l_x at x and label l_y at y, and lets the two messages reach their destinations.

   (c) $\mathcal{A}$ sets $C = A \cup B$.

Process *Component Control*

1. *Existing Internal Link.* When an entity x executes a "**send to** α" operation, where label α has already been assigned by $\mathcal{A}$ to an incident link, $\mathcal{A}$ lets the message reach its destination without delays.

2. *New Internal Link.* If an entity x executes a "**send to** β" operation, where β is an unassigned local label, and x has an unused link (i.e., a link on which no messages have been sent nor received so far) connecting it to a node y in its component, $\mathcal{A}$ assigns label β to that link and lets the message reach its destination without delays.

3. *External Link.* When an entity x executes a "**send to** γ" operation, where γ is an unassigned local label, but all links between x and the nodes of its component have been used, $\mathcal{A}$ forces a more complex process. Let C be the component to which x belongs and let i be its level.

   (a) $\mathcal{A}$ creates a new connected component C' of level i, using process *Creation*.

(b) As soon as an entity y in C', whose links to the nodes of its component have all been used, executes a "**send to** δ" operation, where δ is a still unused port number of y, $\mathcal{A}$ assigns to edge (x, y) label γ at x and label δ at y, and lets the two messages reach their destinations without further delays

(c) When this happens, the connected component $C \cup C'$ of level $i+1$ is formed.

The overall execution $\mathcal{E}$ of $\mathcal{P}$ is started with $\mathcal{A}$ creating a component of level 0 (i.e., activating the first node); it continues with the formation of a larger and larger component; and it terminates as soon as all entities are in the same component. Let t_i be the first time when a connected component of level $i \geq 0$ is formed in execution $\mathcal{E}$; let $size(i)$ be the size (i.e., number of nodes) of that component.

Claim 1 *Let k be the highest level reached by a connected component in execution $\mathcal{E}$.*
(i) $size(k) = n$
(ii) for $0 \leq i < k$, $size(i) = 2^i$
(iii) $k = \lceil \log n \rceil$.

Proof (i) By definition. (ii) By induction. The statement clearly holds for $i = 0, 1$. Let it hold for $1 \leq i < k-1$. Let C_i be the component formed at time t_i; by inductive hypothesis, $size(i) = 2^i$. After time t_i the execution proceeds with the adversary operating according to the Component Control process. Eventually, the External Link operation is invoked by a node of C_i. When this happens, a new component C'_i of level i will be created following the recursive process Component Creation. According to the rules of that process, C'_i will have the same size of C_i provided there are enough inactive nodes when the process starts. At time t_i the number of inactive nodes is $n - |C_i| = n - 2^i$. Hence, if $2^i \leq n/2$, C'_i will contain 2^i nodes; that is, the new component C_{i+1} of level $i+1$, formed when C_i and C'_i merge at time t_{i+1}, will have $|C_i| + |C'_i| = 2^{i+1}$ nodes. In other words, if $2^i \leq n/2$, the statement of the Lemma holds also for $i+1$. Consider now the case $2^i > n/2$; in this case, C_i will be composed of all the $n - 2^i < 2^i$ nodes inactive at time t_i. Thus, when C_i and C'_i merge to form the new component C_{i+1} of level $i+1$ at time t_{i+1}, all nodes become part of the same component; but this means that $i+1 = k$, contradicting the assumption that $i < k-1$. Hence $2^i \leq n/2$ and the statement of the Lemma holds. (iii) It follows from (i) and (ii). □

Hence, we can see the execution of $\mathcal{P}$ as proceeding in *stages*: stage i starting at time t_i and ending at time t_{i+1}, $0 \leq i \leq \lceil \log n \rceil$.

Let us examine how many messages are transmitted in each stage; let $\mu(i)$ denote the minimum number of messages that will be transmitted in stage i. Consider the beginning of stage i, that is when at time t_i there is a single connected component C and its level is i. During this stage, another component of level i is constructed; this is done by first constructing two components of level $i-1$, A and B; merging them into a single component D of level i; and then merging C and D into a single component of level $i+1$. However, regardless of the nature of protocol $\mathcal{P}$, the adversary will not allow two components to merge unless it is forced to. In the case of the merging of A and B, this will happen only when there are two nodes, a and b in A and B, respectively, that, having communicated with all the other nodes in their component, execute a "**send to** l_a" and a "**send to** l_b" operation, respectively, where l_a is a still unused port number of a and l_b is an unused port number of b. Similarly, in the case of the merging of C and D, this will happen only when there are two nodes, c and d in C and D, respectively, that, having communicated with all the other nodes in their component, execute a "**send to** l_c" and a "**send to** l_d" operation, respectively, where l_c is a still unused port number of c and l_d is an unused port number of d. This means that, before the merging of C and D takes place, d must have communicated already with all the nodes in D; if d was in A (resp. B), it has thus sent at least $|B|$ (resp. $|A|$) messages on or after the merge of A and B. In other words, the minimum number $\mu(i)$ of messages transmitted in stage i is: $2\mu(i-1)$ to construct A and B; an additional $|A| = |B| = 2^{i-1}$ to/from d before the merging of C and D; and the two messages between c and d for the merging of C and D, which concludes stage i. Observing that $\mu(0) = 0$, we have

$$\mu(i) = 2\ \mu(i-1) + 2^{i-1} + 2 > i\ 2^{i-1}$$

The total number of stages is $k = \lceil \log n \rceil$; however, if n is not a power of two, one of the two components in the last merge contains only $n - 2^{\lfloor \log n \rfloor}$ nodes. In any case, the adversary can force P to transmit more than

$$\mu(k-1) = \frac{1}{2} \lfloor \log n \rfloor 2^{\lfloor \log n \rfloor}$$

messages, completing the proof of the theorem. □

Note that, as in the proofs of [1, 18] for the Election problem, by appropriately "sliding" the timing of the send and receive events (in the creation and control of the connected components), the proof of Theorem 4 can be extended so that the result holds even when the graph is synchronous.

Since any protocol that solves Election solves also Wake-Up, the existing $O(n \log n)$ protocols for the former problem imply that the above lower bound is asymptotically tight.

6 Concluding Remarks

In this paper we have analyzed the message complexity of the Wake-Up (or Reset) problem in three general classes of interconnection networks: hypercubes, complete graphs, and regular complete bipartite graphs. This basic problem is just the generalization of the Broadcast problem in which there is an arbitrary number of initiators (all with the same message). We have shown that, while anonymous entities can solve Broadcast in $\Theta(n)$ messages in all three classes of networks, order of magnitude more are required in the worst case for Wake-Up: $\Omega(n \log n)$ in hypercubes, and $\Omega(n^2)$ in complete graph and regular complete bipartite graphs, even if the networks are synchronous and with sense of direction. In the case of a complete network with distinct Ids, Wake-Up is computationally as expensive as Election.

The results we have established provide evidence that Wake-Up is a costly process. An important open research problem is to extend our results (and possibly our proof techniques) to other classes of dense networks.

Acknowledgments. Some of these results have been described in [21]. This work has been supported in part by VEGA (Slovakia), and by NSERC (Canada).

References

[1] Y. Afek and E. Gafni. Time and message bounds for election in synchronous and asynchronous complete networks. *SIAM J. Computing*, 20:376–394, 1991.

[2] A. Arora, S. Dolev, and M. Gouda. Maintaining digital clocks in step. *Parallel Processing Letters*, 1(1):11–18, 1991.

[3] A. Arora and M. Gouda. Distributed reset. *IEEE Transactions on Computers*, 43(9):1026–1038, 1994.

[4] B. Awerbuch and R. Ostrovsky. Memory-efficient and self-stabilizing network reset. In *Proceedings of the 13th Symposium on Principles of Distributed Computing (PODC)*, pages 254–263, 1994.

[5] B. Awerbuch, B. Patt-Shami, G. Varghese, and S. Dolev. Self-stabilization by local checking and global reset. In *Proceedings of the 8th International Workshop on Distributed Algorithms (now DISC)*, pages 326–339, 1994.

[6] M.Y. Chan and F.L.Y. Chin. Distributed election in complete networks. *Distributed Computing*, 3(1):19–22, 1988.

[7] K. Diks, S. Dobrev, E. Kranakis, A. Pelc, and P. Ruzicka. Broadcasting in unlabeled hypercubes with a linear number of messages. *Information Processing Letters*, 66(4):181–186, 1998.

[8] S. Dobrev and P. Ruzicka. Linear broadcasting and $o(n \log\log n)$ election in unoriented hypercubes. In *Proceedings of the 4th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 53–68, 1997.

[9] S. Dolev. *Self Stabilization.* MIT Press, 2000.

[10] S. Even and S. Rajsbaum. The use of a synchronizer yields maximum computation rate in distributed networks. In *Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC)*, pages 95–105, 1990.

[11] S. Even and S. Rajsbaum. Unison, canon and sluggish clocks in networks controlled by a synchronizer. *Mathematical System Theory*, 28:421–435, 1995.

[12] P. Flocchini, B. Mans, and N. Santoro. On the impact of sense of direction on message complexity. *Information Processing Letters*, 63(1):23–31, 1997.

[13] P. Flocchini, B. Mans, and N. Santoro. Sense of direction: Definitions, properties and classes. *Networks*, 32(3):65–180, 1998.

[14] P. Flocchini, B. Mans, and N. Santoro. Sense of direction in distributed computing. *Theoretical Computer Science*, 291(1):29–53, 2003.

[15] P.A. Humblet. Selecting a leader in a clique in $O(n \log n)$ messages. In *Proceedings of the 23rd Conference on Decision and Control*, pages 1139–1140, 1984.

[16] A. Israeli, E. Kranakis, D. Krizanc, and N. Santoro. Time-message trade-offs for the weak unison problem. *Nordic Journal of Computing*, 4(4):317–329, 1997.

[17] E. Korach, S. Moran, and S. Zaks. Tight lower and upper bounds for some distributed algorithms for a complete network of processors. In *Proceedings of the 3rd Symposium on Principles of Distributed Computing (PODC)*, pages 199–207, 1984.

[18] E. Korach, S. Moran, and S. Zaks. Optimal lower bounds for some distributed algorithms for a complete network of processors. *Theoretical Computer Science*, 64:125–132, 1989.

[19] S. S. Kulkarni and A. Arora. Multitolerance in distributed reset. *Chicago Journal of Theoretical Computer Science*, 4, 1998.

[20] E.B. Moss. Checkpoint and restart in distributed transaction systems. In *Proceedings of the 3rd Symposium on Reliability in Distributed Software and Database Systems (SRDSDS)*, pages 85–89, 1983.

[21] N. Santoro. *Design and Analysis of Distributed Algorithms.* Wiley, 2007.

[22] G. Singh. Leader election in complete networks. *SIAM Journal on Computing*, 26(3):772–785, 1997.