Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science xx (xxxx) xxx–xxx
Theoretical Computer Science
www.elsevier.com/locate/tcs

Two absolute bounds for distributed bit complexity☆

Yefim Dinitz[a,*], Noam Solomon[b]

[a] *Department of Computer Science, Ben-Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel*
[b] *Department of Mathematics, Ben-Gurion University of the Negev, POB 653, Beer-Sheva 84105, Israel*

Abstract

The concept of distributed communication bit complexity was introduced by Dinitz, Rajsbaum, and Moran. They studied the bit complexity of Consensus and Leader Election, arriving at more or less exact bounds. This paper answers two questions on Leader Election, which remained there open. The first is to close the gap between the known upper and lower bounds, for electing a leader by two linked processors. The second is whether the suggested algorithm, sending $1.5n$ bits while electing a leader in a chain of even length n, is optimal, in the case when n is known to the processors. For both problems, absolutely exact bounds are found. Moreover, the presented lower bound proofs show that there is no optimal algorithm other than the suggested ones.

1. Introduction

The concept of *distributed communication bit complexity* was introduced by Dinitz, Rajsbaum, and Moran [3]; it generalizes the known communication complexity measure (see e.g. [6,2]) to the distributed computing setting. They showed that message complexity is unable to distinguish between the complexities of solving Consensus, Leader Election, and Maximal Id Finding (henceforth, denoted `Consensus`, `Leader`, and `MaxF`) in chains and rings, which contradicts intuition. In contrast, the bit complexity bounds proven there distinguish them successfully, which justifies importance of the suggested complexity measure. In [3] (see its full version in [4]) and the sibling paper [5], several more or less tight pairs of upper and lower bounds for `Consensus`, `Leader`, and `MaxF` in chains, trees, and rings were found.

This paper studies two questions on `Leader` in chains, which remained open in [3]. The first is to close the gap between the known upper and lower bounds, for electing a leader by two linked processors. The second is whether the algorithm [5], sending $1.5n$ bits while electing a leader in a chain of even length n, is optimal, in the case when n is known to the processors. For both cases, absolutely exact bounds are found: for the former question, the recursive formula and explicit expression are given, while for the latter one, optimality of the known algorithm is confirmed. Moreover, in both cases, the family of optimal algorithms is described; the lower bound proof shows that there is no optimal algorithm other than those described.

We hope that our techniques would shed new light on some fine aspects of distributed computing.

☆ Partially supported by the Lynn and William Frankel Center for Computer Science.
* Corresponding author. Tel.: +972 8 647 7867; fax: +972 8 647 7650.
E-mail addresses: dinitz@cs.bgu.ac.il (Ye. Dinitz), noams@cs.bgu.ac.il (N. Solomon).

0304-3975/$ - see front matter © 2007 Published by Elsevier B.V.
doi:10.1016/j.tcs.2007.04.027

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

2. Our model

The model we consider consists of a failure-free, asynchronous message passing distributed system, with arbitrary but finite link delays and negligible local computation time, without shared memory; it is a standard one, for more details see e.g. [1,7]. The network topology concerned is a chain, with n links and $n+1$ processors. We assume that the only input is distinct ids at terminals. Usually, the ids are taken from the set $Z_M = [0 \dots (M-1)] = \{0, 1, \dots, M-1\}$, $|Z_M| = M \geq 2$. All processors are identical, in the sense that they all run the same (deterministic) algorithm, parameterized by n and M. Processors may behave differently, because the processor algorithm has access to its id, if any, and to its number of incident links. Note that in our case of a chain, this information is the same for all internal (non-terminal) processors, so they react to incoming messages in the same way.

Initially, all processors are *asleep* and in the same initial state, except for the id and number of incident links information. When a processor receives a message on an incident link or when it wakes up, spontaneously, before receiving the first message, it is *activated*. Then, the processor algorithm processes its local information, and the processor sends messages on its incident links (it may send zero or more messages on each link), maybe outputs something, and enters the waiting state; in our model, all of this is done immediately.

In this paper, following [3,5], when defining an algorithm, we assume that each message consists of a single bit. In our lower bound proofs, we assume more generally that a processor may send several bits at the same time, but it receives information, sent to it, bit by bit. We also assume that, at any fixed moment, only a single message may be delivered to a processor. As described, there are no sending queues, but there may be queues of bits in-transit, waiting for delivery, at links.

There may be different executions beginning from the same initial state. Thus, the distributed setting implies *tasks*, in contrast to the communication complexity setting, since different scheduling of computation, with a fixed input set, might lead to different (legal) outputs. For a task T, $Bit\,C(T)$ is the number of bits sent needed to solve T, in the worst case.

The `Leader` task requires that each processor should output (decide on) a binary value "leader"/"non-leader", so that there will be exactly one leader. Besides, it is required that any non-leader should learn which of its incident links is in the direction to the leader. The `MaxF` version of `Leader` requires that the terminal with the maximal id must be chosen to be leader; clearly, $Bit\,C(\texttt{MaxF}) \geq Bit\,C(\texttt{Leader})$.

We use the concept of *scheduler* which is a formal device that specifies the order in which processors wake up and messages are delivered. For convenience of analysis, we consider a formal clock: it starts from moment 0, and each processor activation step is done at the next moment $1, 2, \dots$. Also, names A and B are given formally to the terminals. However, those clocks and names are not available to processors, and are in no sense related to the processor algorithm.

Note that the "worst case", for a certain algorithm, is its execution with maximal number of bits sent, over all id pairs and all possible executions. Hence an upper bound, confirmed by showing an algorithm, must be valid for *all* inputs and *all* schedulers, while to obtain a lower bound L, it is sufficient to show that, for any algorithm, there *exists* some input and some scheduler, such that the execution under that scheduler with that input requires at least L bits sent.

We assume, following [3,5], that a scheduler wakes each terminal, eventually, if not awaken by a message previously, and that no internal processor is awaken by the scheduler. Another interesting model case is when a scheduler may wake *any* set of processors, and is meant to wake at least one arbitrary processor (the *minimal waking assumption*). This case is settled for the first problem addressed in this paper, and remains an open question for the second one.

Remark: Notice that the model, where *exactly one* processor is awaken spontaneously, is not interesting. Indeed, the so awaken processor may decide to be leader and just inform all the others of this, by relaying bit sequences in both directions.

We consider the following *Termination Property*: *A distributed algorithm solving some task is said to have the* termination property *(or* is terminating*), if each processor becomes eventually ensured that no more messages concerning this task will be sent by any processor and that there are no messages in transit, except, maybe, messages sent by itself.* This property enables each involved processor to begin eventually communicating with other participating processors on another task. Indeed, the only messages remaining to deliver, if any, are from itself; hence, by the FIFO property, any *new* message sent by it will reach the target processor after it would finish all its activity in

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

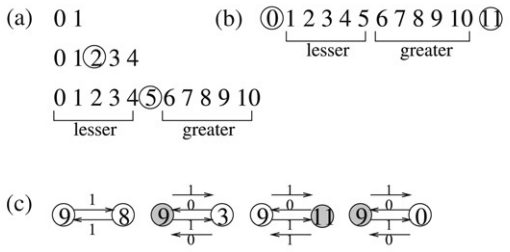

Fig. 1. Optimal algorithms: (a) the terminating case, (b) the non-terminating case, (c) rounds of various types, for the non-terminating case (earlier messages up; the leader is filled grey).

the previous task. For a distributed task T, we denote by $Bit\,C^t(T)$ its bit complexity for the case when the termination property is required; clearly, $Bit\,C^t(T) \geq Bit\,C(T)$.

Suppose that there is an algorithm solving task T, for the case when ids are taken from the set $Z_M = [0 \ldots (M-1)]$. Notice that its slight variation solves it, with the same complexity, if the id range is $[s \ldots (s+M-1)]$. Indeed, receiving the parameter value s, it suffices to simulate the original algorithm, relating to each id x as to $x - s$. In what follows, we identify such an algorithm version with the original algorithm.

For simplicity of presentation, in this paper, we identify a terminal with its id; we say that two ids are paired if they are given to the terminals.

3. `Leader` and `MaxF` for two processors

In this section, we consider the network consisting of two linked processors only. We find the bit complexity for `Leader` and `MaxF` and describe all optimal algorithms.

Theorem 3.1. *For the two processor network,* $Bit\,C(\texttt{Leader}) = Bit\,C(\texttt{MaxF}) = 2\lceil \log_2((M+2)/3.5)\rceil$ *and* $Bit\,C^t(\texttt{Leader}) = Bit\,C^t(\texttt{MaxF}) = 2\lceil \log_2((M+1)/3)\rceil$.

The theorem is implied by the following algorithms and lower bounds. Since `MaxF` is a special variant of `Leader`, we present our upper bounds for `MaxF` only and lower bounds for `Leader` only.

3.1. Algorithms

Proposition 3.2. *For the two processor network, there exists a terminating algorithm solving* `MaxF` *with* $2\lceil \log_2((M+1)/3)\rceil$ *bits sent.*

Proof. We suggest a recursive algorithm; its execution is divided into rounds sending two bits each. For the minimum value 2 of M, there is a single degenerate round, requiring no communication: each processor decides leader or non-leader according to the value of its id 1 or 0, respectively. For any $M \geq 3$, let us divide the id range into the *median* id $\lfloor M/2 \rfloor$, the sub-range of *lesser* ids $[0 \ldots (\lfloor M/2 \rfloor - 1)]$, and that of *greater* ids $[(\lfloor M/2 \rfloor + 1) \ldots (M-1)]$. Note that, for any M, the longest one of the sub-ranges is of length exactly $\lfloor M/2 \rfloor$.

In the algorithm description, mnemonic meaning of operations is given in square brackets, for easier understanding.

Round Description (for illustration see Fig. 1(a))

1. Each lesser id sends 0 [[Am I the non-leader?]], each greater id sends 1 [[Am I the leader?]], while the median id sends nothing and waits.
2. If a lesser (resp., greater) id receives 0 (resp., 1), then [[it realizes that both ids are lesser (resp., greater) ones, and]] the algorithm continues to the next round, for solving the problem in the corresponding sub-range (whose length is at least 2, by the case assumption).

   When the median id receives bit 0 or 1, it decides that it is the leader or non-leader, respectively, and responds by the opposite bit [[I agree with your suggestion]].

   If a lesser id receives 1, then it decides non-leader; if a greater id receives 0, then it decides leader.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

Example: Let M be 11, so that $Z_M = [0..10]$, and the id pair be 10 and 8. *Round* 1: [[id 5 is the middle one]] Both 10 and 8 send 1 and receive 1. *Round* 2: [[id 8 is the middle one]] Id 10 sends 1; when 8 receives that 1, it decides non-leader and replies by 0; when 10 receives that reply 0, it decides leader.

The correctness of the proposed algorithm is immediate. The algorithm is terminating, since at any stage, each processor knows for sure whether it waits for the next bit to be delivered, or the algorithm finishes.

Let us analyze now the number of bits sent. We define the following recurrent relation:

$$m^t_{2(r+1)} = 2m^t_{2r} + 1, \quad r \geq 0;\ m^t_0 = 2.$$

Lemma 3.3. *If M is at most m^t_{2r}, at most $2r$ bits are sent by the above algorithm.*

Proof. We prove by induction on r. In the base case $r = 0$, the statement is trivially correct. Assume it is correct for $r = k$, $k \geq 0$, and consider the case $r = k + 1$. For any $M \leq m^t_{2(k+1)}$, either there is a single round, with two bits sent, or the algorithm continues to the same problem in some id sub-range. In the latter case, the length of both id sub-ranges is at most m^t_{2k}. Hence, the algorithm finishes the problem solution with sending at most $2k$ bits, by the induction assumption. Thus, at most $2k + 2$ bits are sent totally, as required. ■

The solution to the above recurrence relation is:

$$m^t_{2r} = 3 \cdot 2^r - 1.$$

By inverting it, we obtain the upper bound $2\lceil \log_2((M+1)/3)\rceil$ for the number of bits sent, as required. (Note that, in comparison with the bound $2\lceil \log_2 M\rceil - 2$ of the algorithm [3,4], this is less by approximately $2 \cdot \log_2 1.5 \approx 1$ bit, as the asymptotic averages.) ■

Proposition 3.4. *For the two processor network, there exists an algorithm solving* `MaxF` *in* $2\lceil \log_2((M+2)/3.5)\rceil$ *bits sent.*

Proof. The algorithm is similar, in structure, to the terminating algorithm as above. Moreover, it coincides with it for any M up to 5. Beginning from $M = 6$, the algorithm is as given below. The id range is divided into the *minimum* id 0, the *maximum* id $M - 1$, the *lesser* ids from 1 to $\lfloor M/2\rfloor$, and the *greater* ids from $\lfloor M/2\rfloor + 1$ to $M - 2$. Each round sends 2 bits, except for the finishing round considering the id sub-range of length at least 6, which sends 4 bits.

Round Description, M $\geq$ 6 (for illustration see Fig. 1(b), (c)

1. Each lesser id sends 0 [[Am I the non-leader?]], each greater id sends 1 [[Am I the leader?]], while the minimum and maximum ids decide non-leader and leader, respectively, send nothing, and wait.
2. If a lesser (resp., greater) id receives 0 (resp., 1), then [[it realizes that both ids are lesser (resp., greater) ones, and]] the algorithm continues to the next round, solving the problem in the corresponding sub-range.

   When the minimum or maximum id receives any bit, it responds by the opposite bit [[I am not the same as you; let us finish]].

   If a lesser (resp., greater) id receives 1 (resp., 0), then it repeats by sending once more 0 (resp., 1) [[confirms its intention]].
3. After receiving the second bit, the minimum id always responds by 0 [[I am the non-leader]], and the maximum one by 1 [[I am the leader]].

   If a lesser or greater id receives the second bit 0, it decides leader, and if 1 non-leader.

Let us prove that the algorithm solves `MaxF`. The only non-trivial case concerns the decision of a lesser (resp., greater) id which gets the first bit 1 (resp., 0). Let us consider such a lesser id x (the other case is similar). If the other id is a greater or the maximum one, it sends the second bit 1 to x, and x correctly decides non-leader, while if the other id is the minimum one, it sends 0 to x, and x correctly decides leader.

The algorithm is *non-terminating* in the case when, in some round, the minimum or maximum id, w.r.t. the range of that round, decides and waits. Indeed, let that id be x. If the other id is the maximum or minimum one, the execution finishes, and no bit will ever be delivered to x. In all other cases, some bit should be delivered to x, eventually. Note that there are no means, for x, to distinguish between these two situations.

For the analysis of the number of bits sent, we define the following recurrence relation:

$$m^{nt}_{2r+2} = 2m^{nt}_{2r} + 2, \quad r \geq 1;\ m^{nt}_2 = 5,\ m^{nt}_0 = 2.$$

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

Lemma 3.5. *If M is at most m^{nt}_{2r}, at most $2r$ bits are sent by the above algorithm.*

Proof. The proof is similar to that of Lemma 3.3, so only differences from that proof are mentioned. The (trivial) base cases of induction are $r = 0$ and $r = 1$. At the induction step, the analysis is similar to that in the proof of Lemma 3.3, except for the finishing round. By the round description, the number of bits sent in any finishing round is exactly four; this suffices, since for $M \geq 6$, it holds that $2r \geq 4$. ■

The solution to the above recurrent relation is:

$$m^{nt}_{2r} = 3.5 \cdot 2^r - 2, \quad r \geq 1.$$

By inverting it, we obtain the upper bound $2\lceil \log_2((M+2)/3.5)\rceil$ for the number of bits sent. Also for the basic case $M = 2$, this expression gives the right value zero, as required. (Note that the above bound is less by approximately $2 \log_2 1.75 \approx 1.5$ bits, as the asymptotic averages, than the number of bits $2\lceil \log_2 M\rceil - 2$ of the algorithm [3,4].) ■

3.2. Lower bounds

Now, we pass to the lower bounds. We prove the lower bounds for executions under the *symmetric scheduler* [3,4], defined as follows:

- At step 0, both processors are awaken;
- At each following step, the first bit in every queue, if any, is delivered.

Proposition 3.6. *Consider an arbitrary algorithm $\mathcal{A}$ solving* `Leader` *in the two processor network, when ids are chosen from an arbitrary integer set Z. For any $r \geq 1$, if $|Z| \geq m^{nt}_{2(r-1)} + 1$, then there exists an input pair, such that in the execution of $\mathcal{A}$ under the symmetric scheduler at least $2r$ bits are sent. If $\mathcal{A}$ is terminating, the same holds even if $|Z| \geq m^{t}_{2(r-1)} + 1$.*

Proof. Let us call an id *passive*, w.r.t. $\mathcal{A}$, if it sends no bit upon its spontaneous wake-up.

Lemma 3.7. 1. *There exist at most two passive ids.*
2. *If there exist two passive ids, then each one of them decides immediately upon its wake-up, while the decision depends on the id only and the decisions for those two passive ids are different.*
3. *If $\mathcal{A}$ is terminating and M exceeds 2, then there exists at most one passive id.*

Proof. If there are at least two passive ids, each of them must decide immediately. Indeed, let us give any two of them to the processors, and if the algorithm does not finish immediately, there is a deadlock, under the symmetric scheduler. The notion of "decision of a passive id" is well defined, since information on its id only is available to a processor upon its wake-up. Moreover, the decisions of any two passive ids must be different: leader and non-leader, for legality of pairing of those ids. This implies 2. Also, 1 is straightforward, since the existence of three passive ids contradicts the above observation.

Now we pass to item 3. Suppose that for a terminating algorithm, there are two passive ids, say x and y; since M is at least 3, there is at least one non-passive id, say z. If x and y are given to the processors, the algorithm finishes, and x will never get any bit. If x and z are given to them, under the symmetric scheduler, x will get some bit sent by z, eventually. Notice that there are no means for x, upon its (immediate) decision, to distinguish between these two situations. Hence, x can never be ensured that the algorithm has or has not finished, a contradiction to the termination property. ■

Let us, first, assume that $\mathcal{A}$ is terminating. Let us denote $|Z| = M$. We prove the statement of the Proposition by induction on r. Basic case $r = 1$: Since M is at least $m^t_0 + 1 = 3$, by Lemma 3.7(3), there are at least two *non-passive* ids. Let us give them to the processors. Since each one of them sends at least one bit, under the symmetric scheduler, at least two bits are sent.

Assume now correctness of the statement for $r = k$, $k \geq 1$, and let us prove it for the case $r = k + 1$. Among at least $m^t_{2k} + 1$ ids in Z, at most one is passive. Let us divide the set of other ids, which is of cardinality at least $m^t_{2k} = 2m^t_{2(k-1)} + 1$, into two groups Z_0 and Z_1, according to their first bit sent, either 0 or 1, respectively. The largest one of them is of cardinality at least $m^t_{2(k-1)} + 1$; we denote it by Z'.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

Let us consider the continuation of $\mathcal{A}$, from its step 1 under the symmetric scheduler an on, for all choices of ids from Z'. In fact, the only information available to each processor after step 0, is its own id and the fact that the other id belongs to Z' as well. Therefore, the simulation of $\mathcal{A}$ from step 1 under the symmetric scheduler and on is an algorithm solving `Leader`, when ids are chosen from Z'. Note that if an id sends k bits, $k > 1$, upon its wake-up by $\mathcal{A}$, then it sends the last $k-1$ bits out of them, upon its wake-up in this simulation. By the induction assumption, there exist a pair of ids from Z', such that the simulation sends at least $2k$ bits, under the symmetric scheduler. Totally, $\mathcal{A}$ working for that id pair, under the symmetric scheduler, sends at least $2k+2$ bits, as required.

Let us turn now to the case when the termination property is not required.

Basic case $r = 1$ *and* $M \geq 3$: The situation when there exist two non-passive ids is considered previously. Otherwise, there are passive ids x and y, deciding differently, and a single non-passive id, z. Since z may be paired with any one of x and y, it cannot decide immediately after its wake-up. Hence, it has to be activated by a message, before deciding. Therefore, if z is paired with x, there is a message from z and a message to z, summing in at least two bits sent.

Basic case $r = 2$ *and* $M \geq 6$, as well as the *inductive step* are considered similarly as the inductive step for the terminating case, except for the case $r = 2$ and $M = 6$. In this remaining case, the sub-case $\max\{|Z_0|, |Z_1|\} \geq 3$ may also be considered in the same way. Hence the only remaining sub-case is when $|Z_0| = |Z_1| = 2$ and thus there are two passive ids. Recall that our aim is now the lower bound four bits sent.

1. Suppose that some passive id *does not* reply to the first bit received. Then, any non-passive id, if paired with it, must send at least two bits immediately upon its wake-up in order to be activated once more. Then, if two non-passive ids are paired, they send 4 bits already at step 0.

2. Assume that some passive id, x, replies by the *same* bit, w.l.o.g. bit 1, to the first bit received by it. Consider, as above, the continuation of $\mathcal{A}$ after step 0, when ids are chosen from $Z_1 \cup \{x\}$. Its simulation is an algorithm solving `Leader`, while there are three possible ids, with the additional *restriction* that x decides as it decides in $\mathcal{A}$. As shown above, this needs at least two bits sent, in the worst case, even without any restriction, all the more with it. Thus, $\mathcal{A}$ needs at least four bits sent, in the worst case.

3. Consider the remaining case when every passive id replies by the *opposite* bit to the first bit received. If an id from Z_0 receives 1, it cannot decide, since this 1 may come from the id already decided leader or that already decided non-leader. The same concerns any id from Z_1. Hence, if we pair an id from Z_0 and an id from Z_1, both of them have to receive two bits. This suffices. ■

3.3. On a family of optimal algorithms

The proof of Proposition 3.6 implies that the structure of any optimal algorithm must be such as that of the algorithms from Section 3.1. Let us describe possible variations of those algorithms; validity and fullness of the following description may be established easily.

Let us, first, describe variations for an optimal algorithm solving `MaxF`. They are minimal in the case of a border value of M, i.e. that equal to m^t_{2r} or m_{2r}: then, only roles of bit values 0 and 1 may be exchanged, at each round. Note that the value of the reply of the middle, minimal, or maximal id should be opposite to the value of the bit received by it, by the reasons as in item 2 in the proof of Proposition 3.6.

If M is less at least by one than m^t_{2r}, for a terminating algorithm, then the id $\lfloor M/2 \rfloor$ may join the set of lesser ids. Also, if M is less at least by one than m_{2r}, for a non-terminating algorithm, then either id 0 may be set to be lesser, or id $M-1$ to be greater, or id $\lfloor M/2 \rfloor$ may play the role of *middle* id, instead of both of them, as in the terminating algorithm. If M is at most $m_{2r} - 2$, both minimal and maximal id roles may be cancelled similarly. Besides, if the summary number of lesser and greater ids is strictly less than $2m^t_{2(r-1)}$ or $2m_{2(r-1)}$, then the lengths of sub-ranges may vary arbitrarily, restricted from above by $m^t_{2(r-1)}$ or $m_{2(r-1)}$ each one. Similar variations may be done at *each round*, provided the current sub-range length is strictly less than the corresponding border value.

Summarizing, we see that there is the same generic structure of the sub-range tree, while the lengths of sub-ranges may vary, restricted by values of $m^t_{\cdot}$ or $m_{\cdot}$. Also, at some rounds, middle, minimal, and maximal ids may be cancelled, or the middle id may be introduced instead of the pair of minimal and maximal ones. These are all possible variations, for `MaxF`.

Consider now additional variations for optimal `Leader` algorithms (of course, the variations as for `MaxF` may be applied). Now, there is no need for lesser ids to have values lesser than those for greater ids, and similarly for middle,

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

minimal and maximal ids. Also, the sets of lesser and maximal ids must not be continuous integer intervals. Hence, values of middle, minimal, and greater ids may be chosen arbitrarily, and not only the lengths, but also the composition of the sets of lesser ids may be arbitrary, at each algorithm round.

3.4. Case of minimal waking assumption

Let us consider now the model variation, where just a single processor, out of the two, may be awaken by the scheduler (the *minimal waking assumption*). Let us use notion of $Bit\,C^m(T)$ and $Bit\,C^{mt}(T)$, for this model variation. We assume that there are at least three possible ids.

The main observation is that there is no passive id, at the beginning of any algorithm, working under this assumption. Indeed, assume to the contrary that there exists a passive id. If it is paired with any other id, and awaken alone, by the scheduler, then the execution finishes immediately. Hence, all ids are meant to have a built-in decision. However, pairing two of them with the same decision leads to a non-legal decision pair.

Theorem 3.8. *For the two processors network, under the minimal waking assumption,* $Bit\,C^m(\texttt{Leader}) = Bit\,C^m(\texttt{MaxF}) = 2\lceil\log_2((\lceil M/2\rceil+2)/3.5)\rceil+2$ *and* $Bit\,C^{mt}(\texttt{Leader}) = Bit\,C^{mt}(\texttt{MaxF}) = 2\lceil\log_2((\lceil M/2\rceil+1)/3)\rceil+2$.

Proof. In terms of Section 3.1, the sense of the above observation is that, at moment 0, the lesser and greater id sets, as well as Z_0 and Z_1, partition the entire id set.

Correspondingly, descriptions of the relevant algorithms are similar to those in Section 3.1, differing at the *first* round only, and is as follows: For both algorithms, the lesser ids are $[0\ldots\lfloor M/2\rfloor]$, the greater ids are $[(\lfloor M/2\rfloor+1)\ldots(M-1)]$, and there are no middle, minimal, and maximal ids. Algorithm validity is straightforward, as in Section 3.1.

We denote by $m^m_{\cdot}$ and $m^{mt}_{\cdot}$ analogues of $m_{\cdot}$ and $m^t_{\cdot}$, respectively. Clearly, $m^m_{2r} = 2\cdot m_{2(r-1)}$ and $m^{mt}_{2r} = 2\cdot m^t_{2(r-1)}$. This implies the algorithm running time bounds given by the expressions in the theorem statement. Also, the matching lower bound may be proved similarly to the proof of Proposition 3.6, taking into account the above observation. ■

4. `Leader` in a chain of even length

Consider the `Leader` problem in a chain of even length n. Let P_0 denote the middle processor of the chain. We call *A-half* and *B-half* the chain intervals $[A\ldots P_0]$ and $[P_0\ldots B]$, respectively. All algorithms presented in this section are terminating, so we do not distinguish between $Bit\,C(\cdot)$ and $Bit\,C^t(\cdot)$.

4.1. Algorithms

Let us begin with the description of a few algorithms (for illustration see Fig. 2). In any execution, we call a *b-processor* an internal processor awaken by the bit b, $b\in\{0,1\}$.

Algorithm 2 (works for $n=2$): The terminals decide "non-leader", while the single non-terminal has the built-in decision "leader". No bit is sent.

Algorithm 4 (works for $n=4$): Terminals decide "non-leader" and send 1. Any 1-processor decides "non-leader" and sends 0 forward. The single 0-processor P_0 does not send anything. It decides "leader", upon its wake-up or when it gets one more 0. Totally, 4 bits are sent.

Flipping the bit values 0 and 1 results in another variant of Algorithm 4.

Algorithm 6 (works for $n=6$): Each terminal decides "non-leader" and sends 1. Any 1-processor decides "non-leader", and sends 00 forward. Any 0-processor, upon its wake-up, does nothing and waits for an additional bit. In what follows, when activated by the second bit from the *same* neighbor, it decides "non-leader" and sends 0 forward. If it receives the second bit from the *other* neighbor, it decides "leader" and sends nothing (this happens for P_0 only). Totally, 8 bits are sent.

A degenerate, in a sense, version of Algorithm 6 arises by setting arbitrarily the behavior of a 0-processor after receiving 1 after 0, which *never happens*. Other algorithm variants arise by changing 00 to 01. The rearrangement between first two links to carry 1 and $0b$, $b\in\{0,1\}$, defines another variants of Algorithm 6, with sub-variants where

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

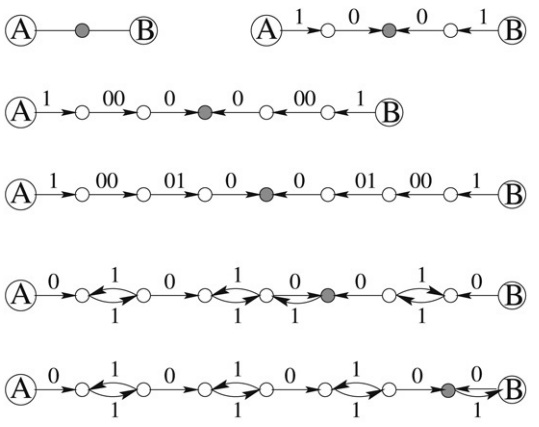

Fig. 2. Executions of Algorithm 2, of the basic variant of each one of Algorithm 4, Algorithm 6, and Algorithm 8, and two sample executions of the Main Algorithm, for $n = 8$. (Bits sent at a link are ordered from left to right. The leader is filled grey.)

some ids send 00, while others send 01. Even more variants are the result of flipping the bit values 0 and 1, as in Algorithm 4.

Algorithm 8 (works for $n = 8$): It differs from Algorithm 6 in the following: Instead of 1, 00, and 0 sent on the three consequent links of each half-chain, there are 1, 00, 01, and 0 sent on its four links. Totally, 12 bits are sent.

The variants of Algorithm 8 are obtained by arbitrary rearrangements of 1, 00, and 01, and by flipping the bit values.

Main Algorithm [5] (works for any $n \geq 8$): Each terminal decides "non-leader" and sends 0. Any 0-processor sends 1 forward. Any 1-processor decides "non-leader", sets that the leader is farther, and sends 0 forward and 1 backward. Any 0-processor, upon receiving 1 from its following processor, decides "non-leader", sets that the leader is farther, and sends nothing. Upon receiving 0 from its following processor, a 0-processor decides "leader", and sends nothing. When a 1-processor receives a bit, except for that waking it, it does not react. Totally, $1.5n$ bits are sent.

Subtle variations of the Main Algorithm arise if a single id sends nothing upon its wake-up, and when awaken by a message (necessarily by bit 1), either decides "leader" and replies by 1, or decides "non-leader" and replies by 0; instead, the same decision may be made, by that exceptional id, immediately upon its wake-up. The other variant of the above mentioned versions of the Main Algorithm arises by flipping the bit values.

Proposition 4.1. *Algorithms* 2, 4, 6, 8, *and the Main Algorithm solve Leader in* 0, 4, 8, 12, *and* $1.5n$ *bits sent, respectively.*

Proof. The validity of Algorithms 2, 4, 6, and 8 is obvious. The validity of the Main Algorithm is proved in [5]; we prove it, for completeness. It is easy to check that, since n is even, the "waves" propagating from the terminals meet *always* at a link between a 0-processor, sending 1, and either a 1-processor or a terminal, sending 0. Therefore, (i) exactly one 0-processor (that incident to the wave meeting link) gets reply 0, which causes it to decide "leader", and (ii) the number of bits sent at the links looks *always* as two sequences $1, 2, \ldots 1, 2$, beginning from the two chain ends (one may be empty), which results totally in $1.5n$ bits sent. The only exception, for (i), is in one of the cases, mentioned as "subtle variations", where *all* 0-processors receive the reply 1, while the terminal (with the exceptional id) decides "leader". Also in this case, there is the bit sequence $1, 2, \ldots 1, 2$, as above. ■

4.2. Lower bound

Let us consider the family of schedulers which wake A first, allow B to be awaken by the first message sent to it, if it exists, and otherwise wake B after all processors have become quiescent. Let us call the execution prefix before waking B, under any such scheduler, a *full A-wave*; a *full B-wave* is defined similarly. A full wave is called *halting* if it halts before reaching the other terminal. Clearly, any wave depends only on the id given to its initiating terminal, which we denote by the wave *origin*, and on the scheduler. Any prefix of a full wave is called a *wave*.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

It is easy to see that any execution begins from *interleaved independent A- and B-waves*. We say that *waves meet*, when a message, sent at a link, e, in one of the waves, reaches a processor, P, activated by the other wave. We denote the wave meeting moment, if any, by t_m, and say that the waves *meet at* e and *at* P. Assume that full A- and B-waves, executed separately, cover overlapping chain areas. Clearly, by fine tuning the wave interleaving by the scheduler, we are able to make the waves meet at any processor in the overlapped area, and at any one of its incident links.

It may also happen that some two waves do not meet at all, but both halt before a meeting. In wave analysis, we use words like "forward", " backward", "farther", "next", etc., w.r.t. the wave origin.

Theorem 4.2. *For a chain of even length n and a set of possible ids of cardinality $M \geq 6$, Bit C*(`Leader`) *is* 0 *if* $n = 2$, 4 *if* $n = 4$, 8 *if* $n = 6$, $1.5n$ *if* $n \geq 8$. *Besides, if n is at least* 10*, there is no optimal algorithm, except for the Main Algorithm.*

The statement holds even if non-leaders are not required to know the direction to the leader.

The rest of Section 4.2 is devoted to the proof of this theorem. In what follows, we analyze an arbitrary *optimal* `Leader` algorithm $\mathcal{A}$; by Proposition 4.1, it sends no more bits than stated in Theorem 4.2, in the worst case. In what follows, we prove the required lower bounds and uniqueness, without using the assumption that any non-leader is required to know the direction to the leader.

Obviously, Algorithm 2 is optimal. Henceforth, we assume n be at least 4.

4.2.1. Auxiliary statements

Lemma 4.3. *In the case when $n \geq 4$ and $M \geq 3$, for at most one id, there exists a wave initiated by it which halts before P_0.*

Proof. Suppose, to the contrary, that there exist two distinct ids, x, y, such that for any one of them, there exist a wave, initiated by it, which does not reach P_0. If we interleave those two waves, arbitrarily, we obtain some legal execution, E, where P_0 is not awaken. Then, P_0 has some built-in decision. However, since the algorithms for all internal processors are the same, all (at least three) internal processors have the same built-in decision. Therefore, the built-in decision of all inner processors, if any, must be "non-leader"; thus, the leader is always chosen among the terminals. Let us show that this cannot happen.

Observe that x and y decide differently, in E: one "leader", and the other "non-leader", based on their halting waves only.

Let us pair some other id, z, with either x or y, in the following manner: we first simulate the x/y-wave of E, and then start a wave from z. Observe, first, that z must decide differently, in those two executions, since all inner processors had decided "non-leader", and terminals given x or y had have different decisions before the wave from z was initiated. Different decisions may happen only if the z-wave meets either the x-wave or y-wave (or both), and information propagates from some wave meeting back to z; assume, w.l.o.g., the x-wave.

Let us analyze the number of bits sent in the above (x, z)-execution. Up to the wave meeting, there were at least n messages sent on all the links of the chain. By our assumption, the wave meeting processor is farther than P_0, from z. In other words, at least $0.5n + 1$ bits propagated from the wave meeting back to z. This results in more than $1.5n$ bits, totally—a contradiction to the choice of $\mathcal{A}$. ■

We denote the exceptional id, as in Lemma 4.3, by $ex\text{-}id_1$, if exists. For a wave from any other id, we call its part up to waking P_0 the *half-wave*. Notice that if any two ids other than $ex\text{-}id_1$ are paired, the two half-waves should meet at P_0, if their finishing moments are synchronized to be close by.

Let us say that a non-terminal processor is a *one-entry*, w.r.t. some execution, if it receives just a single bit (waking it), during that execution.

Observation 4.4. *Assume we are using ids other than $ex\text{-}id_1$. In any execution of any algorithm with N bits sent, there are at least $2n - 2 - N$ one-entries. Therefore, in any execution of $\mathcal{A}$, there are at least two one-entries, if $n = 4, 6, 8$, and at least three if $n \geq 10$.*

Indeed, suppose that there are k one-entries, each receiving one bit, while the $n - 1 - k$ other internal processors receive at least two bits each. Then, N is at least $k + 2(n - 1 - k) = 2n - 2 - k$, and the first part of the statement follows. The second part is implied by Proposition 4.1.

We say that a scheduler is *Tail-Preference Balanced* (or *TPB*) if:

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

1. In each half-chain, it delivers always messages in-transit at links *closest* to the wave origin;
2. It delivers no messages to P_0, as far as possible;
3. Immediately after delivering the first message to P_0, the first message on the other incident link is delivered to it, if any.

Let us fix an arbitrary TPB scheduler, $\mathcal{S}$, and analyze executions of $\mathcal{A}$ under it. By the TPB-rules and Lemma 4.3, if ids are other than $ex\text{-}id_1$, the two waves meet at P_0. Besides, at t_m, all processors are quiescent, except for P_0, and all the message queues at links of the chain are empty, except for those to and from P_0. Hence after t_m, information to processors other than P_0 propagates *from* P_0 *only*. Notice, besides, that up to t_m, the two waves are completely independent of one another, except for maybe the messages sent by P_0 (but not delivered).

Lemma 4.5. *Consider an algorithm sending at most N bits. Then, for any id, x, given to w.l.o.g. A, except for* $ex\text{-}id_1$ *and at most one more id, some other id,* $y \neq ex\text{-}id_1$*, may be given to B, so that in the execution under* $\mathcal{S}$*, there are at least* $n - 1 - \lfloor N/2 \rfloor$ *one-entries in the A-half-chain.*

Proof. Suppose the contrary. Let us say that an id is *lacking one-entries*, if when we give that id to w.l.o.g. A, and any other id, except for $ex\text{-}id_1$, to B, in the execution under $\mathcal{S}$, there are less than $n - 1 - \lfloor N/2 \rfloor$ one-entries in the A-half-chain.

Suppose that, except for $ex\text{-}id_1$, there are more than one id lacking one-entries. Let us pair two such ids in an execution under $\mathcal{S}$. There would be less than $n - 1 - \lfloor N/2 \rfloor$ one-entries in each half chain, that is less than $2(n - 1 - \lfloor N/2 \rfloor) \leq 2n - 2 - N$ one-entries in both of them, while P_0 is not a one-entry—a contradiction to Observation 4.4. ■

Let us denote the second exceptional id, as in Lemma 4.5, if exists, by $ex\text{-}id_2$. Notice that for $\mathcal{A}$ and any n, the lower bound for the number of one-entries $n - 1 - \lfloor N/2 \rfloor$ is strictly positive, and is at least 2 if n is at least 10, by the choice of $\mathcal{A}$ and by Proposition 4.1.

Let us assign to each id x, as in Lemma 4.5, some execution, E_x, as in that lemma. Let P_x denote the farthest one-entry in A-half-chain, w.r.t. E_x.

Corollary 4.6. *For any id, x, given to w.l.o.g. A, except for* $ex\text{-}id_1$*,* $ex\text{-}id_2$ *and at most one more id, in the execution of* $\mathcal{A}$ *under* $\mathcal{S}$*, all processors in* $[A \ldots P_x]$ *decide "non-leader" (in particular, the terminal decides so), independently of the id given to the other terminal.*

Proof. By the definition of a one-entry, in E_x, the propagation from P_0 after t_m to the direction of x stops before reaching P_x. Hence, by TPB-rules, the interval $[A \ldots P_x]$ has finished all its activity in E_x, including processor decisions, before the A-wave propagated beyond P_x, that is, independently of the information propagated from B. Recall that the A-half-wave from x under $\mathcal{S}$ is the same in all executions where x is given to A, that is, it *coincides* with that of E_x, including processor decisions.

Assume to the contrary that there are two ids, each one of them causing at least one processor, in the interval as above, to decide "leader". Then, we may pair these ids, under $\mathcal{S}$, and thus arrive at an execution with two processors decided "leader"—a contradiction. ■

Let us denote the exceptional id as in Corollary 4.6, if it exists, by $ex\text{-}id_3$. Notice that, by Corollary 4.6, for any pair x, y of ids distinct from $ex\text{-}id_{1,2,3}$, the only region, where the leader may be chosen, is strictly between P_x and P_y.

4.2.2. Case analysis

Clearly, the reaction of all internal processors to the bit waking it—deciding and bit sending—is completely defined by that bit, since the initial internal processor state is unique. Let us classify algorithms by satisfying the following properties $\mathcal{I}(b)$, $b = 0, 1$, of their executions under $\mathcal{S}$. $\mathcal{I}(b)$*: Any b-processor decides and relays a bit forward immediately upon its wake-up*. In the following optimality proofs, we assume that *no id is* $ex\text{-}id_1$, which implies that, in any execution, the two waves meet.

Case 1: *Neither* $\mathcal{I}(0)$*, nor* $\mathcal{I}(1)$ *are satisfied.*

Let us see that this case is impossible, for $\mathcal{A}$. Indeed, let us consider an execution of $\mathcal{A}$ under $\mathcal{S}$. The middle processor P_0 should receive a message at its two adjacent links, by Lemma 4.3. Each other internal processor should

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

receive at least two messages, either for deciding, or for propagating the wave to the direction of P_0. Thus, at least $2n-2$ messages should be received totally, which is too much, for $\mathcal{A}$, for any n.

Case 2: *Both* $\mathcal{I}(0)$ *and* $\mathcal{I}(1)$ *are satisfied.*

Let us see that this case is impossible as well, for $\mathcal{A}$. For this, we consider executions under $\mathcal{S}$.

If both 0- and 1-processors decide "leader", then in any execution, all internal processors (at least three) would be leaders—a contradiction. Hence, we may assume w.l.o.g. that any 1-processor decides "non-leader".

Observe that each 0-processor must decide "leader". Indeed, otherwise, in any execution, all internal processors would decide "non-leader"; then, by Corollary 4.6, in any execution under $\mathcal{S}$ of two ids other than $ex\text{-}id_{1,2,3}$, no leader would be chosen.

Assume that any 1-processor relays bit 0 farther. Obviously, no two ids send 0, since otherwise two leaders would be chosen. Let us pair two ids, other than $ex\text{-}id_1$, sending 1 upon the wake-up. If $n \geq 6$, the two processors at distance 2 from the chain ends would decide "leader"—a contradiction. If $n = 4$, the two half-waves, together with the message sent by P_0, upon its wake-up, contain at least 5 bits sent, which is more than Algorithm 4 sends—a contradiction to the optimality of $\mathcal{A}$.

The only remaining sub-case is when any 0-processor decides "leader", while any 1-processor decides "non-leader" and relays 1 farther. Recall that no two ids cause the terminal to send 0. Hence, in the case $M \geq 6$, we may choose an execution E such that both terminals send 1, upon their wake-up. Then, in E, there would be no 0-processors, i.e. no internal leaders. Applying Corollary 4.6, we deduce that no leader would be elected in E—a contradiction.

Summarizing, by cancelling Cases 1 and 2, for $\mathcal{A}$, we have shown that the only possible case is:

Case 3: *W.l.o.g.,* $\mathcal{I}(1)$ *is satisfied, while* $\mathcal{I}(0)$ *is not.*

Notice that any one-entry is a 1-processor, since any one-entry must decide and relay a message forward immediately. By Observation 4.4, there exists an execution of $\mathcal{A}$ under $\mathcal{S}$ with at least two one-entries, which are 1-processors. So, the decision of each 1-processor must be "non-leader", since otherwise at least two leaders would be chosen, at that execution.

Assume, first, that any 1-processor relays 1 farther, upon its wake-up. By Corollary 4.6, for any execution of an id pair (x, y), $x, y \neq ex\text{-}id_{1,2,3}$, under $\mathcal{S}$, the only region, where the leader may be chosen, is strictly between P_x and P_y. However, both P_x and P_y are 1-processors, which implies that all processors in that interval are 1-processors. Thus, no leader will be elected—a contradiction.

Now, we are left with the single possibility, used in the algorithms mentioned in the Theorem: *Main Sub-Case* (*MSC*) : *"Any 1-processor relays* 0 *farther"*.

Observe that *in any execution, with ids other than* $ex\text{-}id_1$*, any 0-processor receives at least two bits*. Indeed, any 0-processor, except maybe that of the wave meeting, must be activated at least twice, since upon its wake-up, it either does not decide, or does not propagate the wave (or both); the processor where the waves meet, by definition receives at least two bits.

At first, let us concentrate on the case $n \geq 10$. Recall the execution E_x, defined after Lemma 4.5, for any id x other than $ex\text{-}id_{1,2}$. By Lemma 4.5, the half-wave from x, in it, validates the following property, for our case $n \geq 10$.

Property 4.7. *There exists a wave of* $\mathcal{A}$*, from any id* x *other than* $ex\text{-}id_{1,2}$*, containing at least two processors, which had been awaken by the same bit, w.l.o.g.* 1*, and since then were never activated.*

Let us consider a wave, W, as in Property 4.7, w.l.o.g. from A, and let P and Q denote the first two processors~~,~~ ~~as there~~. Let us denote the distance between P and Q by k. Let us consider W, while giving to B any id other than $ex\text{-}id_1$, and change it to the following (well defined) execution prefix, E_1, which:

- begins with waking A and B,
- does not deliver the first message sent by B,
- coincides with W (except for waking B), until waking Q,
- continues after that moment, up to reaching B, by delivering messages to processors farther than Q only, exactly as for those farther than P, periodically, with period k links.

Note that, by construction, any processor at a period border has the same history as P; in particular, it is a 1-processor and was activated only once. We claim that the prefix E_1 is legal. This is so, since the situation in each period after Q is, by construction, exactly the same as in that between P and Q, at the legal wave W. Note that

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

nothing can prevent continuation of the prefix E_1, until B is reached; in other words, this prefix is a non-halting wave from x. Also, nothing can prevent E_1 to continue to be completed to a full execution, E'. For illustration of the following analysis, the bottom sample in Fig. 2 may be useful.

Claim 4.8. *In MSC, in any execution beginning from a wave from A reaching B and a bit sent by B, at least* $1.5n$ *bits are sent.*

Proof. Let us divide the processors, beginning from that after A and finishing at B, into $n/2$ consequent pairs, and show that each pair receives at least three bits, which will suffice. At any pair consisting of two internal processors, if the first one is a 0-processor, then it receives at least two bits, which, together with the bit waking the next processor, sums into three bits. Otherwise, the first one is a 1-processor, which relays 0, causing the next one be a 0-processor; the latter one receives at least two bits, which suffices similarly. At the last pair, B receives a bit, while the first processor receives the bit waking it and the bit sent by B, which also suffices. ■

Notice that thus we have already proved that the Main Algorithm is optimal. The rest of the analysis, for the case $n \geq 10$, concerns its uniqueness as the optimal one.

Recall that the total number of bits sent by $\mathcal{A}$ cannot exceed $1.5n$. Therefore, the bits sent in E' and counted in the proof of Claim 4.8 are *all the bits sent*. In particular, the terminal A, given any id other than $ex\text{-}id_{1,2}$, never receives any bit; hence, such ids decide immediately upon waking up. An essential corollary, from the fullness of counting, is that at any pair of consecutive internal processors, exactly one is a one-entry. Hence, by the choice of P and Q (those defining the prefix E_1), the period, at E_1, is of length 2; it consists of a 1-processor and a 0-processor.

Lemma 4.9. *For* $n \geq 10$*, internal processors behave at* $\mathcal{A}$ *exactly as in the Main Algorithm.*

Proof. A consequence of the above analysis is that 0- and 1-processors must alternate, at any wave; thus *any 0-processor relays* 1. Recall that, by Property 4.7, a full wave E_1 as above may be built from any id, except for $ex\text{-}id_{1,2}$. Recall also that terminals decide immediately. Since such a decision may be "leader" for at most one id, and since M is at least 6, there exist at least three ids deciding immediately "non-leader".

Let us consider the full wave E_1 from A given one of those ids, x, while B is given another one of them. Recall that any 1-processor decides "non-leader", so *the leader must be chosen among 0-processors*. By the periodicity of each wave, the only place for the leader may be at the link of wave meeting, that is, the leader must be the 0-processor, R, next to B. An easy parity reason implies that A sent 0. Since x was chosen arbitrarily, all ids that decide immediately "non-leader" send 0; thus B sent 0 as well. Recall that there may not be any bits sent, except for those counted in the proof of Claim 4.8. Hence, R should decide immediately upon receiving 0 from B. Therefore, we arrive at a necessary condition that *whenever a 0-processor receives 0 from the next processor, it decides "leader"*.

Observe that no 0-processor receives a bit from the previous processor, except for the bit waking it. Indeed, otherwise, by the periodicity of E_1, all of them should receive such a bit; in particular, R would receive three bits, in a contradiction with fullness of counting in the proof of Claim 4.8. As a consequence: *when a processor receives first bit* 0, *it sends* 1 *forward immediately*. Thus, the second bit received by any 0-processor comes always from the next (1-)processor; note that it should be the same bit, for all of them. Since n is at least 10, there are at least two 0-processors not incident to the link of wave meeting, so the bit as above must be 1. Hence, *when a processor receives first bit* 1, *it immediately sends* 1 *backwards and* 0 *forward*. Summarizing, we arrive at all the rules of the Main Algorithm concerning internal processors. ■

Lemma 4.10. *For* $n \geq 10$*, terminals behave at* $\mathcal{A}$ *exactly as in the Main Algorithm.*

Proof. Recall that any id, other than $ex\text{-}id_1$, sends some bit upon its wake-up. By the rules for internal processors (shown to be unique, in Lemma 4.9), any wave propagating from such an id is non-halting. Let us show that no wave may begin from sending bit 1. Indeed, then the full wave would contain $2 + 1 + \cdots + 2 + 1 = 1.5n$ bits sent, up to waking the other terminal. However, if the other terminal is not $ex\text{-}id_1$, it sends some bit as well, resulting in at least $1.5n + 1$ bits sent totally—a contradiction. So, *any id, other than* $ex\text{-}id_1$*, sends bit 0, upon its wake-up*. When we pair two ids, other than $ex\text{-}id_1$, under $\mathcal{S}$, two waves beginning from bit 0 meet at P_0. Hence, either P_0 or one of its adjacent 0-processors (depending on the parity of $n/2$) would be elected as leader. This means that any such id must decide "non-leader" immediately upon its wake-up.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

So, the only case remaining to consider is when $ex\text{-}id_1$, w.l.o.g. at B, does not send anything upon its wake-up. It may either decide, or not decide immediately. When activated by the coming full wave, it has to decide "leader" or "non-leader", if not decided upon the wake-up, and send a bit, in order to inform its neighbor R of its decision. Then, R will decide accordingly. Notice that this variation of the Main Algorithm is as given in its description. ■

This is the end of analysis of the case $n \geq 10$.

For the cases $n = 4, 6, 8$, let us assume, to the contrary, that $\mathcal{A}$ sends *strictly less* than 4, 8, 12 bits (respectively), in the worst case. Then, the inequality $n - 1 - \lfloor N/2 \rfloor \geq 2$, which was crucial in the case $n \geq 10$, for the existence of two one-entries in half-waves (in the sense made precise in Lemma 4.5), holds as well, and thus establishes Property 4.7. As a consequence, we are able to establish Claim 4.8, in the same way as in the above case. We ~~thus show~~ that at least $1.5n$ bits are sent, arriving at a contradiction to our assumption. This is the end of the proof of Theorem 4.2.

4.3. *Proof of uniqueness for* $n = 2, 4, 6, 8$

In this section, we extend the uniqueness result of Theorem 4.2 to the case $n \leq 8$. In our analysis of this case, we restrict attention to the sub-case *MSC*, from the case analysis of Theorem 4.2, since the part of this analysis before considering MSC is for general n. We often mean using scheduler $\mathcal{S}$, without mentioning this.

Theorem 4.11. *In the case* $n \leq 8$ *and* $M \geq 6$*, there is no optimal algorithm, except for Algorithms* 2*,* 4*,* 6*,* 8*, and the Main Algorithm for* $n = 8$.

Proof. Consider, first, the case $n = 2$, when no bit is sent by $\mathcal{A}$. Obviously, Algorithm 2 is optimal, so the only goal is to prove that $\mathcal{A}$ cannot be distinct from it. If the built-in decision of P_0 is "leader", Algorithm 2 is clearly unique. Assume to the contrary that P_0 decides "non-leader". At a terminal, id only defines the decision. It is easy to see that there exist two ids causing the terminal to decide in the same way. Pairing such two ids, we arrive at a non-legal execution. Therefore, Algorithm 2 is the unique optimal algorithm, for the case $n = 2$.

Passing to the case $4 \leq n \leq 8$, let us summarize what remains to be proved. Assume first, that algorithm $\mathcal{A}$, competing with Algorithms 4, 6, and 8, admits *at least one* non-halting wave, say W from id z. By Observation 4.4 and the definition of MSC, W should contain at least two processors, each received only one bit 1. That is, the statement of Property 4.7 holds for z; let us show that it holds for all ids required. Notice that the wave W may be changed to become periodic, in the way as given in the proof of Theorem 4.2. Let us consider any id, y, other than z and $ex\text{-}id_{1,2}$. By Lemma 4.5, there exists a half-wave, generated by y, which contains at least one 1-processor, P. Now, if we follow that wave from y up to waking P, and after that change it to become periodic, with the same period as that of the z-wave, we end up with a (legal w.r.t. $\mathcal{A}$) periodic y-wave. This establishes Property 4.7. Now, we may arrive at the Main Algorithm, in the same way as given in the proof of Theorem 4.2. Note that the Main Algorithm is optimal for $n = 8$, but is not optimal for $n = 4, 6$.

The following Proposition settles the case when the above assumption does not hold, thus finishing the proof of Theorem 4.11. ■

Proposition 4.12. *In the case* $n = 4, 6, 8$*, if* $\mathcal{A}$ *generates halting waves only, then it either coincides with one of Algorithms* 4*,* 6*, and* 8*, or sends more bits, than those algorithms, in the worst case.*

Proof. In further main constructions, we restrict ourselves by using only ids other than $ex\text{-}id_1$; that is, *no wave halts before* P_0. The behavior of $ex\text{-}id_1$ is considered separately. Note that a wave may halt at a 0-processor only. Recall, besides, that at any execution, *any 0-processor receives at least two bits*, as was shown in the proof of Theorem 4.2.

Let us prove the Proposition for the case $n = 4$. Since $\mathcal{A}$ sends four bits, it follows that there is exactly one bit at each link. Hence, the two waves halt at P_0, which receives 0 on both sides. By Observation 4.4, there must be at least two one-entries, which should be the two other internal processors, both 1-processors. Any terminal decides based on its id only; the decision should be "non-leader" for almost all ids. Thus, P_0 should decide "leader", after receiving one or two 0's. Thus we arrive at the unique optimal algorithm, except for possible specifics for $ex\text{-}id_1$. It should send something, otherwise the next processor would not decide. It cannot send 0, since then the next processor and P_0 would finish in the same state, so that there would be either two undecided processors or two leaders. So, $ex\text{-}id_1$ must send 1, and thus has no specifics.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

Lemma 4.13. *In MSC, for $n = 6, 8$, any 0-processor waits for* exactly *one more bit, before it sends bits forward.*

Proof. By definition of MSC, any 1-processor relays 0 forward. So, there are 0-processors other than P_0, in any half-wave. Assuming to the contrary that at some execution, a 0-processor waits for at least three bits, before propagating the wave, let us count a lower bound for the number of bits received, totally. Let n be 6: 3 (that processor) + 2 (another 0-processor) $+2(P_0) + 2 * 1$ (one-entries) $= 9$ bits—a contradiction to optimality.

For $n = 8$: No half-chain may contain two one-entries, since then we would be able to show a periodic, and hence a non-halting wave, contradicting the assumption of the proposition. So, there are at most two one-entries (which are 1-processors) in the entire execution, implying at least $3+3\cdot 2+2+2*1 = 13$ bits sent—a contradiction to optimality of $\mathcal{A}$. ■

In fact, the proof of this Lemma shows that the bit count, for $\mathcal{A}$, is very tight, and thus the algorithm should be very definite:

- any wave propagates forward only;
- there are exactly two 1-processors, one in each half-chain, and any one of them receives just the single bit 1 waking it;
- any 0-processor, except for P_0, receives exactly two consecutive bits, b_1b_2, from the previous processor, and only then sends something; it is never activated, after that (we call it b_1b_2-processor);
- P_0 receives exactly two bits, both 0's, from both sides, without sending anything, and then decides.

Moreover, similarly to the case of $n = 4$, the decision of P_0 must be "leader", and thus decisions of the terminals must be "non-leader", including $ex\text{-}id_1$, if any. Notice that any wave must reach P_0, since otherwise the latter would not decide. In other words, $ex\text{-}id_1$ does not exist, according to its definition.

Also, one-directional wave propagation implies that no two processors, in the same half-chain, receive the same bits. Indeed, if so, we would be able to build a legal periodic, and thus non-halting wave, similarly to the proof of Theorem 4.2, thus arriving at a contradiction to the assumption of the proposition. In other words, the only freedom, for $\mathcal{A}$ in any half-chain, is:

- to choose dispersion of *distinct* sendings, out of 1, 00, and 01, on the links, and
- to assign these sendings to the processor types: 1-, 00-, and 01-processor, and a terminal given a certain id.

We now finish to prove Proposition by case analysis. In what follows, * means any bit value, b denotes a certain bit value, while $\bar{b}$ the other value.

Let n be 6. First, assume that there exists some id, such that a terminal given it sends 1. Hence, any 1-processor sends forward $0b$. This implies that no id sends 0* forward; indeed, then the next processor should send 1 and the 1-processor after it $0b$, not a single 0, as needed. In other words, all terminals send 1 forward. Therefore, any (1-)processor adjacent to a terminal sends $0b$, and the next processor sends a single 0 forward. The result is one variation of Algorithm 6.

The remaining case is that all terminals send 0*, depending on their id. Then, a ~~0-processor~~ relays 1, while a 1-processor sends 0 forward. We arrived at the two possible variations of Algorithm 6.

Let now n be 8. We begin once more with the assumption that some id sends 1. Then, any 1-processor sends $0b$, any processor receiving $0b$ relays $0\bar{b}$, and any processor receiving $0\bar{b}$ relays 0. This once more implies that no id sends 0* forward; indeed, then the wave from that id would halt before P_0, by the rule as above. So, all terminals send 1 forward, implying a variation of Algorithm 8.

Otherwise, let us consider an id sending $0b$. Then, either receiving $0b$ causes sending $0\bar{b}$, $0\bar{b}$ causes 1, and 1 causes 0, or $0\bar{b}$ and 1 exchange, in such a sequence. In any case, no id may send $0\bar{b}$, since then the wave from it would not reach P_0. In other words, all ids create the same wave of the type as above, and we arrive at other variations of Algorithm 8. ■

4.4. *Small values of M, with $n \geq 4$*

Results of this section are less important, and are presented as a sketch.

For the case $M \geq 4$, we conjecture that the result of Theorem 4.2 holds. We have a draft of its proof; however, it is technical and does not add much to our understanding.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

For the case $M = 3$, the following algorithm $\mathcal{A}_{M=3}$ is, clearly, correct: One of the ids decides "leader", propagates 1, 1, ... wave and halts. Some other id decides "non-leader", propagates 0, 0, ... wave and halts. The third id is silent and waiting; when receiving a bit, it decides according to its value. Internal processors have built-in decision "non-leader". They propagate 1's and 0's, deciding on the direction to the leader accordingly; they do not react to the second bit received. Clearly, the number of bits sent is $n + 1$. The bit values 0 and 1 may be flipped.

Proposition 4.14. *For $M = 3$ and $n = 4$, Bit C equals 4, and the unique optimal algorithm is Algorithm 4.*
For $M = 3$ and $n \geq 6$, Bit C equals $n + 1$, and the unique optimal algorithm is $\mathcal{A}_{M=3}$.

Proof. The case $M = 3$ and $n = 4$: Optimality is clear ~~from~~ Lemma 4.3. ~~For making~~ the proof of uniqueness of Algorithm 4, given in the proof of Proposition 4.12, work here, we should release it from the assumption of MSC. In other words, from the assumption that a 1-processor relays a bit immediately, while a 0-processor halts. Notice that pairing two distinct ids, other than $ex\text{-}id_1$, in an execution under $\mathcal{S}$, proves that either a 0-processor or a 1-processor halts. Indeed, otherwise, P_0 would not halt, and more than 4 bits would be sent totally. So, we may assume, up to symmetry of 0 and 1, that a 0-processor waits for another bit. ~~By the bit counting,~~ a 1-processor must propagate the wave immediately, as required. Therefore, the proof of uniqueness given there, is valid in our case as well.

The case $M = 3$ and $n \geq 6$: Let us prove optimality of $\mathcal{A}_{M=3}$. Let us consider some id, x, different from $ex\text{-}id_{1,2}$. By Lemma 4.5, we can pair it with some other id, different from $ex\text{-}id_1$, say y, in an execution E, under $\mathcal{S}$, so that E contains at least two one-entries in the x-half-wave. This implies that the x-wave may be continued to reach the other terminal, by the following reasons. Either those two one-entries are of the same type, implying the existence of a periodic wave in the "usual" way, given in the proof of Theorem 4.2. Or there are two one-entries, such that one is a 1-processor and the other a 0-processor. This implies that both 1-processors and 0-processors do not halt, and so we may continue the wave to be non-halting, giving preference to its propagation. If the scheduler begins with waking both x and y, then y sends a bit, since it is distinct from $ex\text{-}id_1$. If the scheduler then gives preference to the x-wave, it reaches y, sending at least n bits. Totally, there are at least $n + 1$ bits.

Let us now prove the uniqueness of $\mathcal{A}_{M=3}$. The above analysis presents the full bit count. Hence, the x-wave is one of the following two waves: either the alternating 0–1 wave, or the constant 1- or 0-wave. The former option may be disregarded, since in E, both waves reach P_0, and there is no way, for internal processors, to decide the direction to the leader.

The latter option is indeed ~~that~~ used in $\mathcal{A}_{M=3}$. By symmetry, it is sufficient to set the wave from x be the constant 1-wave. Consider now the y-wave in E, which as we know, does not halt before P_0. If y sends 1, it is also the constant 1-wave, and again internal processors would not know the direction to the leader. Hence, y must send the bit 0, to its adjacent processor. If a 0-processor waits for more bits, then more than $n+1$ bits would be sent in the entire execution E: the two half-waves ~~(remembering~~ that the processor adjacent to y receives at least two bits) plus the bit 1 relayed by P_0 after being awaken by the x-wave. So, a 0-processor sends immediately something. If it sends 1 forward, then its neighbor would be a one-entry and a 1-processor (again from the tight counting of bits). But also, on the other side of the chain, we have such processors, and so, they will not be able to determine the direction to the leader. So, the only option is that any 0-processor immediately sends 0 forward, and hence y generates the constant 0-wave (by tightness of bit counting), which is again the option given in the description of $\mathcal{A}_{M=3}$. The "decision properties" of internal processors and terminals are obviously immediate, again ~~from~~ the tightness of the above bit counting. Hence, the only freedom is the assignment between "leader/non-leader" decisions and constant 0/1-waves. ■

For the case M equals 2, the optimal algorithm is as follows: One of the ids decides "leader", while the other decides "non-leader". All internal processors are non-leaders. For setting the direction to the leader, the leader (another version: the non-leader) propagates either 0, 0, ..., or 1, 1, ... The bit complexity is n. If the direction to the leader is not required, there is no such wave, and the bit complexity is zero.

References

[1] Hagit Attiya, Jennifer Welch, Distributed Computing: Fundamentals, Simulations and Advanced Topics, McGraw-Hill, England, 1998.
[2] Martin Dietzfelbinger, The linear-array problem in communication complexity resolved, in: Proceedings of the 29th ACM Symposium on Theory of Computing, 1997, pp. 373–382.
[3] Ye. Dinitz, S. Moran, S. Rajsbaum, Bit complexity of breaking and achieving symmetry in paths and rings, in: Proc. of the 31th Symposium on Theory of Computing, STOC'99, pp. 265–274.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027

[4] Ye. Dinitz, S. Moran, S. Rajsbaum, Bit complexity of breaking and achieving symmetry in chains and rings, Technical Report #CS-2004-11, Dept. of Comp. Sci., Technion, August 2004, 29p.
[5] Ye. Dinitz, S. Moran, S. Rajsbaum, Exact communication costs for consensus and leader in a tree, J. Discrete Algorithms 1 (2003) 167–183.
[6] Eyal Kushilevitz, Noam Nisan, Communication Complexity, Cambridge University Press, 1997.
[7] Nancy A. Lynch, Distributed Algorithms, Morgan Kaufmann Publishers, Inc., 1996.

Please cite this article in press as: Ye. Dinitz, N. Solomon, Two absolute bounds for distributed bit complexity, Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.04.027