Efficient Communication in Unknown Networks

Luisa Gargano, Andrzej Pelc, Stéphane Pérennes, Ugo Vaccaro

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Efficient Communication in Unknown Networks

Luisa Gargano — Andrzej Pelc — Stéphane Pérennes

— Ugo Vaccaro

N° 3609

janvier 1999

THÈME 1

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

Efficient Communication in Unknown Networks

Luisa Gargano [*] , Andrzej Pelc [†] , Stéphane Pérennes
, Ugo Vaccaro [‡]

Thème 1 — Réseaux et systèmes
Projet sloop

Rapport de recherche n° 3609 — janvier 1999 — 22 pages

Abstract: We consider the problem of disseminating messages in networks whose topology and size are not known to nodes. Three communication tasks of increasing difficulty are studied. In *blind broadcasting* (BB) the goal is to communicate the source message to all nodes. In *acknowledged blind broadcasting* (ABB) the goal is to achieve BB and inform the source about it. Finally, in *full synchronization* (FS) all nodes must simultaneously enter the state *terminated* after receiving the source message. We show that BB is achieved in time at most $2n$ in any n-node network and show networks in which time $2n - o(n)$ is needed. For ABB we show algorithms working in time $(2 + \epsilon)n$, for any fixed positive constant ϵ and sufficiently large n. Using large messages, the source can additionally learn the entire topology of the network and time can be reduced to $2n$. Finally, we show a simple algorithm for FS working in time $3n$, and indicate how this time can be slightly reduced using large messages. The optimal time of full synchronization remains an open problem. This is the first paper devoted to the study of deterministic communication under network ignorance scenario.

Key-words: Broadcasting, anonymous networks, imcomplete information protocols

Research done during the authors's visit at INRIA Sophia Antipolis

[*] Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84081 Baronissi(SA), Italy. (lg@dia.unisa.it)

[†] Département d'Informatique, Université du Québec à Hull, Hull, Québec J8X 3X7, Canada. Supported in part by NSERC grant OGP 0008136. (pelc@uqah.uquebec.ca)

[‡] Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84081 Baronissi(SA), Italy (uv@dia.unisa.it)

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 —Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Communications efficaces dans les réseaux inconnus

Résumé : Nous considérons le problème de diffusion dans les réseaux dont la topologie et la grandeur sont inconnues aux noeuds. Nous étudions trois tâches de communication de difficulté croissante. Le but de la *diffusion aveugle* est de communiquer le message de la source à tous les noeuds. Le but de la *diffusion aveugle certifiée* est d'accomplir la diffusion aveugle et d'en informer la source. Finalement, pour accomplir la *synchronisation complète* tous les noeuds doivent simultanément entrer en l'état *terminé* après avoir obtenu le message de la source. Nous montrons que la diffusion aveugle est accomplie en temps au plus $2n$ dans chaque réseau avec n noeuds et nous montrons des réseaux dans lesquels le temps $2n - o(n)$ est necessaire. Pour la diffusion aveugle certifiée nous proposons des algorithmes qui travaillent en temps $(2+\epsilon)n$, pour n'importe quelle constante ϵ positive fixée et n suffisament grand. En employant des grand messages la source peut en plus apprendre la topologie du réseau entier et on peut réduire le temps à $2n$. Finalement nous montrons un algorithme simple pour la synchronisation complète, qui travaille en temps $3n$ et nous indiquons comment ce temps peut être légèrement réduit en employant des grands messages. Le temps optimal de la synchronisation complète reste un problème ouvert.

Mots-clés : Diffusion, Réseaux anonumes, Protocoles distribués

1 Introduction

Coping with incomplete information in the design of network algorithms is an important problem whose many variations have been extensively studied. One of such aspects are computations in anonymous networks (cf. [3, 4, 8, 17, 18, 19]), in which processors do not have distinct identities and execute identical algorithms. The impossibility of distinguishing processors yields symmetry in computations and restricts the computational power of the network.

A different restriction on the amount of knowledge available to processors is the lack of *sense of direction* (cf. [21, 13]), when there is no "consistent" labeling of links. For rings, such a consistent labeling is equivalent to orientation and its computational aspects were studied, e.g., in [3].

One of the fundamental tasks in distributed computing is *broadcasting*. (See, e.g., surveys [16, 14] for references to the literature on broadcasting.) One node of the network, called the *source*, has a message which has to be transmitted to all other nodes. Recently a few papers [9, 10, 11] have been devoted to the study of broadcasting in networks in which neither links nor nodes have *a priori* assigned labels, and thus in networks both anonymous and devoid of sense of direction. In this setting a node cannot distinguish between yet unused adjacent links, and the decision on which of them the message should be sent belongs to the adversary, as the worst case cost of broadcasting is studied. A similar adversarial approach to broadcasting was adopted in [1, 15] where *messy* broadcasting was considered. However, in all these papers it was assumed that nodes know the topology of the network, although they lack some information necessary to orient it, and thus to carry out broadcasting in the most efficient way. This knowledge of topology is sometimes crucial, e.g., knowing that the network is a clique, the source is sure that broadcasting is completed as soon as it informs all of its neighbors.

The question arises of how communication tasks can be accomplished in networks whose nodes have no knowledge of the network whatsoever, they do not know its topology or even its size. Such complete *a priori* ignorance was assumed in the context of graph exploration [2, 5, 7, 20], when a robot is placed in a totally unknown graph and has to explore it, i.e., traverse all of its edges, minimizing the total number of traversals.

In this paper we study the problem of broadcasting, assuming that nodes are completely ignorant of the network. The *a priori* knowledge of each node is limited to its own degree. Such a situation can arise, e.g., when the network consists of mobile agents who move frequently and are capable of communicating only with those who are within the reach of their send/receive device. In this situation the topology of

the resulting communication network changes frequently and is unknown to each agent at any given moment. A similar lack of knowledge occurs when the network is a portion of the internet to which a site wants to transmit its message. In this case a node is usually aware only of the names of adjacent nodes or at most of those situated within a small distance.

While randomized protocols have been studied under assumption that nodes do not know the topology of the network (cf., e.g., [6, 12]), to the best of our knowledge, ours is the first paper devoted to the study of deterministic communication under network ignorance scenario.

We assume that communication is executed in *rounds* controlled by a global clock. In each round every informed node can send a message to at most one neighbor, and this message becomes available at the neighbor in the next round. Simultaneously a node can receive messages from many neighbors. Messages can contain additional control information apart from that being disseminated by the source. We are interested in the number of rounds necessary to carry out a given communication task and call it the *time* needed for this task.

Since nodes are ignorant of network topology, they cannot distinguish between yet unused adjacent links and consequently the decision on which of them the message should be sent belongs to the adversary, similarly as in [9, 10, 11]. We consider the worst case time, i.e., the time resulting from the most detrimental behavior of the adversary.

In the classic setting, when all necessary information about the network is available to nodes, time of broadcasting (which will be called *classic broadcasting* in this case) can be known in advance and thus all nodes can be aware of the termination of broadcasting as soon as it is completed. A different situation occurs in our setting. Since even the size of the network is unknown, broadcast can well be finished but no node may be aware of this fact. Consequently, we study three communication tasks of increasing difficulty. In *blind broadcasting* (BB) the goal is simply to communicate the source message to all nodes. In *acknowledged blind broadcasting* (ABB) the goal is to achieve BB and inform the source about it. This may be crucial, e.g., when the source has several messages to disseminate and nobody should learn the next message until everybody learns the previous one. Finally, in *full synchronization* (FS) all nodes must simultaneously enter the state *terminated* after receiving the source message. In particular, this implies common knowledge about completion of broadcasting (all nodes know that everybody got the message, they know that everybody knows that everybody got the message, etc.). Full synchronization is the first time

when a concerted action of all nodes can be undertaken after learning the source message.

We show that BB is achieved in time at most $2n$ in any n-node network and show networks in which time $2n - o(n)$ is needed. For ABB we show algorithms achieving this goal in any n-node network in time $(2 + \epsilon)n$, for any fixed positive constant ϵ and sufficiently large n. These algorithms use messages of logarithmic size. Using large messages, the source can additionally learn the entire topology of the network and time can be reduced to $2n$. Finally, we show a simple algorithm for FS working in time $3n$ for any n-node network. This algorithm also uses messages of logarithmic size. We indicate, how to reduce this time, e.g., to $2.9n$, using large messages, and simultaneously inform all nodes about the topology of the network. The optimal time of full synchronization remains an open problem.

We use the following terminology. A network is modeled by a simple connected undirected graph with a distinguished node called the *source*. The *distance* between two nodes is the length of the shortest path between them. The *depth* of the network is the largest distance between the source and any other node. The ith *level* of a network is the set of all nodes at distance i from the source. Clearly a node in level i can be adjacent only to nodes in levels $i - 1$, i and $i + 1$. Every node in level i is adjacent to a node in level $i - 1$.

The paper is organized as follows. Sections 2,3 and 4 deal with blind broadcasting, acknowledged blind broadcasting and full synchronization, respectively. Section 5 contains conclusions.

2 Blind broadcasting

In this section we give asymptotically tight upper and lower bounds on the time of blind broadcasting. Call an edge *saturated* if the source message has been transmitted on it in at least one way. For BB there is essentially one optimal algorithm: in every round every informed node sends the source message on (any) yet unsaturated edge. (No control messages are necessary, the only transmitted message is the original one disseminated by the source.) Transmitting on a saturated edge or refraining from any transmission is a loss of time and the choice among unsaturated edges belongs to the adversary. It should be noticed that accomplishing BB is equivalent to saturating all edges because a protocol leaving an unsaturated edge would not acccomplish BB if an additional node of degree 2 were added in the middle of this edge, which could not be noticed prior to a transmission along this edge.

Hence establishing an upper bound for BB is equivalent to showing that all edges of any network will be saturated in the given time, regardless of the behavior of the adversary, while establishing a lower bound is equivalent to constructing a network and showing an adversary strategy for it which prevents saturation of all edges before the given time.

Theorem 2.1 *For any n-node network of depth d, blind broadcasting is completed in worst case time $2n - d - 2$.*

Proof: Let s be the source and L_i, for $i = 1, ..., d$, be the ith level of the network. Denote by a_i the size of L_i. After a_1 rounds since the beginning of broadcasting, all edges between the source and L_1 are saturated. After at most $a_1 - 1 + a_2$ further rounds, all internal edges in L_1 and all edges between L_1 and L_2 are saturated. After at most $a_2 - 1 + a_3$ further rounds, all internal edges in L_2 and all edges between L_2 and L_3 are saturated. Proceeding in this way we conclude that after at most $a_1 + (a_1 - 1 + a_2) + (a_2 - 1 + a_3) + \cdots + (a_{d-1} - 1 + a_d)$ rounds, all edges of the network, except possibly the internal edges in L_d, are saturated. Further $a_d - 1$ rounds are sufficient to saturate all these edges. Thus total time of BB is at most $2(a_1 + \cdots + a_d) - d = 2(n-1) - d = 2n - d - 2$. □

The next result provides an asymptotically matching lower bound.

Theorem 2.2 *For any positive integer n, there exists an n-node network and an adversary strategy for which blind broadcasting requires time $2n - o(n)$.*

Proof: We show the argument for n of the form $4k^2$. Modifications in the general case are easy. First we define the network. The set of all n nodes is divided into $m = 2k$ disjoint sets A_1,...,A_m of size m. Edges join every pair of nodes in A_i, for all $i = 1, ..., m$, and every pair u, v of nodes, such that $u \in A_i$ and $v \in A_{i+1}$, for all $i = 1, ..., m-1$. Let $A_1 = \{x_0, ..., x_{m-1}\}$. The source is x_0.

Next we define the adversary strategy. In the ith round, $i = 1, ..., m-1$, all nodes $x_0, ..., x_{i-1}$ send a message to x_i. In the mth round every node $x_i \in A_1$ sends a message to a different node $y_i \in A_2$. In the next $m - 1$ rounds the adversary strategy is as follows. In round $m + j$, $j = 1, ..., m-1$, nodes x_i and $y_{i+j(\bmod m)}$, $i = 0, ..., m-1$, send messages to each other. Hence, after $2m - 1$ rounds since the beginning of broadcasting, saturated edges are exactly those between pairs of nodes in A_1 and between A_1 and A_2. Moreover, informed nodes with incident unsaturated edges are exactly those in A_2.

It is well known that the set of edges of a complete graph of even size m can be partitioned into $m-1$ pairwise disjoint perfect matchings. Let M_1,..., M_{m-1} be these matchings for the clique on A_2. In the further $m-1$ rounds, nodes in A_2 send messages to each other along edges from M_i in round $(2m-1)+i$. The next m rounds are spent sending messages between A_2 and A_3, and so on. In general, saturating all internal edges of A_i takes $m-1$ rounds, and saturating all edges between A_i and A_{i+1} takes m rounds. Hence the total number of rounds is $m(m-1)+(m-1)m = 2m^2-2m = 2n-2\sqrt{n}$. □

Remark. An adversary strategy yielding lower bound $2n-O(\sqrt{n})$ can be also given for the complete network on n nodes.

3 Acknowledged blind broadcasting

In this section we describe a family of protocols achieving acknowledged blind broadcasting. Their aim is to broadcast the source message to all other nodes of the network, and inform the source that all nodes got the message. For any *a priori* given positive constant ϵ, we show a protocol working in worst case time $(2+\epsilon)n$, for n-node networks, where n is sufficiently large. The protocol uses messages of size at most $\log n$ and additionally informs the source of the number of nodes in the network.

Let ϵ be a positive constant fixed throughout this section. The protocol consists of two phases, the first of which is modified blind broadcasting and the second is the acknowledgement phase. The aim of the first phase is to broadcast the source message to all nodes, and simultaneously create a spanning tree of the network, of height equal to the depth of the network. Then information about completion of broadcasting and the number of nodes is efficiently sent to the source bottom-up using this tree.

The tree is constructed dynamically, and the depth of each node in it decreases during the execution of the protocol, to finally reach the level number of the node in the network. This guarantees efficient execution of the acknowledgement phase but requires updating labels which indicate the current depth, and updating current parents. The above actions must be carefully scheduled to avoid long delays in spreading the source message. We show how this can be done so that the additional delay caused by these actions be at most ϵn.

Another problem is avoiding confusion among various versions of the constructed tree. Nodes do not know when the tree is final and hence send information about

completion of broadcasting and about size of parts of the network already using earlier versions of the tree. Thus messages concerning various versions of the tree circulate simultaneously and there is an *a priori* danger of forgetting a node or counting it more than once. This could happen if information from different rounds were mixed: a node earlier counted in one branch could subsequently change parents and be also counted in a different branch. For the same reason a node might not be counted at all. In order to avoid this confusion we use "time stamps" indicating which version of the tree the message refers to. At the end, the source makes sure to consider a single version of the tree.

Protocol ABB(ϵ)

Phase 1.

Let c be an integer larger than $6/\epsilon$. c is appended to the original source message and forms the message M which is disseminated. The source assigns itself label 0 and sends it together with M on consecutive yet unsaturated edges. When a node gets M for the first time from a node with label i, it assigns itself label $i+1$ and sends it together with M on consecutive incident edges.

The crucial idea is label updating. Whenever a node v gets a new label, it sends it (together with M), to all neighbors u from which it didn't hear, or from which it heard that u's label is higher than v's current label. Such neighbors u will be called *compulsory* with respect to the current label of v. If a node has label i and hears from a node with label $j < i-1$, it updates its own label to $j+1$. Thus labels of each node are decreased every time they change, and the smallest label a node can get is the number of its level.

Label updating is combined with dynamic construction of a spanning tree. At every round every informed node has a single *parent* which can change several times in the course of the protocol. The current parent of node v is its neighbor whose message caused v to modify its label most recently (or whose message woke up v, in case when v still has its first label).

A node v acknowledges its current parent by sending to it a message "you became my parent" and cancels the old parent by sending to it a message "you stopped being my parent". This is combined with label updating in the following way.

After getting a new label l, a node v starts process Spread(l) described as follows. In the first $c-1$ rounds after getting label l, v sends this label (together with M) to consecutive compulsory neighbors. In the next two rounds, it first cancels the old parent, then acknowledges the new parent. In further rounds it continues sending its label to other compulsory neighbors. If during process Spread(l), v gets a new label l', Spread(l) is halted and v starts process Spread(l') from the beginning.

A node involved in some process Spread(l) is called *busy*. Otherwise it is *idle*. Clearly, an idle informed node has all its incident edges saturated.

A node considers as its *current children* all nodes from which it got an acknowledgement "you became my parent" which has not been cancelled afterwards.

A node v that becomes idle at some round t', starts a *waiting period* of $c+1$ rounds. If it does not change its label before the end of the waiting period, v becomes *confirmed* in round $t = t' + c + 1$. If at this round v does not have current children, v considers itself a *leaf* of the tree corresponding to round t, and enters Phase 2. (Notice that a confirmed node can subsequently loose its status by getting a new label. In this case it becomes busy, then idle, and then confirmed again.)

Phase 2.

Messages in phase 2 consist of a *time stamp* and of a *content* of the form "my subtree has size x". Every confirmed node keeps a variable *con* whose value is the round number in which it became (most recently) confirmed. Such a node also keeps lists of all messages obtained from all current children.

Phase 2 is started by confirmed nodes which consider themselves leaves of the tree corresponding to round t. In every round $i > t$ such a node v sends a message "my subtree has size 1", with time stamp $i-1$, to its current parent. This is done as long as v remains confirmed.

In every round every confirmed node v that has current children looks at all lists of messages obtained from all of them. If there exists a common time stamp t' in all these lists, larger than the current value of v's variable *con*, call the corresponding messages t'-*healthy*. Let t be the maximum of such t'. Node v takes all contents "my subtree has size x_i" corresponding to all t-healthy messages from all its current children c_i, $i = 1, ...r$, and then sends the message "my subtree has size $x_1 + \cdots + x_r + 1$ " with time stamp t to its parent.

As soon as the source detects t-healthy messages from all its children u_i, $i = 1, ..., k$, for some time stamp t, it knows that all nodes of the network got the source message, and that the total number of nodes is $n = y_1 + \cdots + y_k + 1$, where "my subtree has size y_i" are the corresponding contents.

Theorem 3.1 *Protocol ABB(ϵ) correctly accomplishes acknowledged blind broadcasting and works in worst case time at most $(2+\epsilon)n$, in any n-node network, for sufficiently large n.*

Proof: In order to prove the correctness of Protocol ABB(ϵ), it is enough to show that detecting t-healthy messages by the source, for some time stamp t, implies

that in round t broadcasting has been completed and this information reached the source bottom-up using the spanning tree frozen at time t.

Indeed, according to the description of process Spread(l), a node may be acknowledged as a parent at most $c+1$ rounds after becoming it. Hence, the waiting period imposed on a node v before it becomes confirmed, guarantees that every node w whose parent is v, is already counted by v among its current children. In particular, nodes considering themselves leaves of the tree corresponding to round t are indeed leaves of this tree.

When a confirmed non-leaf v gets t-healthy messages "my subtree has size x_i", it knows that all of its children corresponding to round t reported the sizes of their subtrees corresponding to this round, and consequently v's subtree in the tree corresponding to round t had size $x_1 + \cdots + x_r + 1$. Node v reports this information (still with time stamp t) to its parent, that did not change since round t. Consequently, the rule of conveying information by non-leaves guarantees that only snapshots corresponding to the same round are relayed up the tree. Hence the total number of nodes computed by the source is indeed the number of nodes in the spanning tree corresponding to some round t, i.e., the number of nodes in the network.

It should be noted that between round t and the round when t-healthy messages are detected by the source, the tree could change many times. Nevertheless the source gets a correct and complete snapshot of the situation occurring in round t.

It remains to show that the source will detect t-healthy messages for some $t \leqslant (2+\epsilon)n$.

Call a node *large* if its degree is at least c, and *small* otherwise. According to the description of process Spread(l), the actions of cancelling and acknowledging parents do not delay blind broadcasting in small nodes, and each such action delays it by at most one round in large nodes. Let a_i be the size of the i-th level of the network. Define the round

$$T_i = a_1 + (a_1 - 1 + a_2) + ... + (a_{i-1} - 1 + a_i) + 2j_{i-1},$$

where j_{i-1} is the number of levels before the i-th that contain large nodes.

By round T_i all nodes in the i-th level may have changed their labels several times but they will certainly get the message from a neighbor in level $i-1$, containing its final label $i-1$. Hence such a node, in case when it is large, may only loose two further rounds to cancel the old parent and acknowledge the new one (which from now on will never change), and then it will pass its final (freshly updated) label i further.

A computation similar to that in the proof of Theorem 2.1 shows that all edges will become saturated in round at most $2n - d + 2x$, where x is the number of levels

containing large nodes. Clearly, $x \leqslant 3n/c$. The waiting period is $c+1$. Hence after the round $2n - d + 2x + c + 1$, phase 2 must start, and at this point all labels are final, the spanning tree has height d (equal to the depth of the network) and never changes again. This implies that by round $2n - d + 2x + c + 1 + d$ the source will detect t-healthy messages, for $t = 2n - d + 2x + c + 1$, if it has not detected such messages for smaller t at some earlier round. We conclude that Protocol ABB(ϵ) works in worst case time at most $2n - d + 2x + c + 1 + d \leqslant 2n + 6n/c + c + 1$, which does not exceed $(2+\epsilon)n$, for sufficiently large n. □

It should be noticed that at the time when the source learns that all nodes got the message in Protocol ABB(ϵ), other nodes still send messages with new time stamps at each round. As mentioned in the introduction, ABB should be considered as one cycle in an ongoing process of broadcasting several messages, in which a new message should be disseminated only after everybody learned the previous one. When the source sends the next message, it simultaneously orders to stop all messages concerning the previous cycle.

We conclude this section by describing a protocol working even faster than Protocol ABB(ϵ) (in time at most $2n$ for an n-node network) and accomplishing much more. Upon its completion, all nodes get the source message, and the source learns not only that and the number of nodes, as in the previous case, but in fact learns the topology of the entire network. The drawback of this protocol is that the circulating messages are very large (of maximum size $O(n^3 \log n)$, where n is the size of the network, as opposed to logarithmic size in Protocol ABB(ϵ)). However, large messages are hard to avoid if the goal is to learn fast the entire topology.

We first present the protocol assuming that, when a node is informed, it gets a name, all names being distinct. We then show how this naming procedure can be added to the protocol.

The protocol works in two phases. Similarly as before, the aim of the first phase is to broadcast the source message to all nodes and simultaneously create a spanning tree of the network, of height equal to the depth of the network. This is done by a mechanism of label updating, as in Protocol ABB(ϵ), with the exception that current parents are not acknowledged (this permits to save ϵn time). In phase 2, nodes send all topological information they learned to date, up the tree, toward the source.

Protocol Learn-All

Phase 1.

The source assigns itself label 0 and sends it together with the original message on consecutive yet unsaturated edges. Nodes get and update labels in the same way as in Protocol ABB(ϵ). When a node v gets a new label, it sends it to all neighbors

compulsory with respect to this label (i.e., to all neighbors u from which it didn't hear or from which it heard that u's label is higher than v's current label).

The current parent of a node is defined as before. Any node that got all of its edges saturated and sent its current label to all compulsory neighbors, enters phase 2. (As oppposed to Protocol ABB(ϵ), phase 2 is started by every node independently, not only by the leaves of the current tree.)

Phase 2.

Every node v keeps a record $CLK(v)$ called *current local knowledge*. It consists of the name v of the node, of its degree $\deg(v)$, and of a list of neighbors of v, initialized as empty. In the beginning of phase 2, every node v sends $CLK(v)$ to its current parent. Whenever v gets a phase 2 message from a neighbor u, it updates $CLK(v)$ by adding u to its list of neighbors and then sends its own $CLK(v)$ together with all $CLK(w)$ that it previously learned, to its current parent. In fact, nodes keep track of what they have sent to the current parent. If the parent did not change and v obtained no new information, it does not send anything in the given round.

It should be noted that messages from phase 1 can still be sent after a node started phase 2. If v changes its label, it first sends it to all compulsory neighbors and then sends all available $CLK(w)$ (together with its own) to the new parent.

The source continuously updates all incoming records $CLK(v)$ by adding new nodes to the respective neighbors lists, and applies the following *termination rule*. At each round, for any v for which it got a record $CLK(v)$, the source counts the number $N(v)$ of nodes u such that either u is in the neighbors list in $CLK(v)$ or v is in the neighbors list in $CLK(u)$. As soon as $N(v) = \deg(v)$ for all these v, the source concludes that all nodes in the network got the message and reconstructs the entire network as follows. The nodes of the network are precisely those nodes v for which it got records $CLK(v)$, and nodes u and v are adjacent, iff either u is in the neighbors list in $CLK(v)$ or v is in the neighbors list in $CLK(u)$.

It remains to show how distinct names can be assigned to all nodes. This can be done in phase 1. Names are binary sequences. The source gets name (0). Let d be the degree of the source. Then the source selects d binary sequences s_1,..., s_d, of length $\lceil \log d \rceil$, and assigns sequences $(0) \star s_i$ to consecutive neighbors, by appending these sequences to the messages sent according to Protocol Learn-All. ($\star$ denotes concatenation.) Every other node v, upon getting a message for the first time, considers the assigned sequence s as its name. It then selects $d(v) - 1$ binary sequences r_1,..., $r_{d(v)-1}$, of length $\lceil \log d(v) \rceil$, where $d(v)$ is its degree, and assigns sequences $s \star r_1$,..., $s \star r_{d(v)-1}$ to all neighbors other than the one from which it got

sequence s. Whenever a node that already has a name gets a new sequence assigned to it, it ignores this new assignment.

Using this method, distinct names of length at most $n \log n$ can be assigned to all nodes. Since Protocol Learn-All requires sending messages containing $O(n^2)$ names, this yields messages of size $O(n^3 \log n)$.

Theorem 3.2 *Upon completion of Protocol Learn-All, the source knows that all nodes got the original message and has a correct knowledge of the topology of the network. The protocol works in worst case time at most $2n$, for any n-node network.*

Proof: It is clear that using the termination rule, the source learns a subgraph of the network. Suppose that there is a node u in the network, unnoticed by the source, in spite of the use of this rule. Since the network is a connected graph, there exists a node w unnoticed by the source and adjacent to a node v for which the source has record $CLK(v)$. Thus the number of nodes x known to the source, such that either x is in the neighbors list in $CLK(v)$ or v is in the neighbors list in $CLK(x)$ must be smaller than $\deg(v)$, which contradicts the rule. A similar argument works for unnoticed edges.

It remains to show that the condition of the termination rule is satisfied in round $2n$ at the latest. A reasoning similar to the one in the proof of Theorem 3.1 shows that all edges are saturated and all labels are final before round $2n - d$, where d is the depth of the network. The current tree at this round is final and the entire topological information is locally available, i.e., for any edge uv in the network, either u is in the neighbors list in $CLK(v)$ or v is in the neighbors list in $CLK(u)$. All this information reaches the source in at most d further rounds, i.e., in round $2n$ at the latest. At this point the condition of the termination rule is satisfied. It is also clear that by this round all nodes stop sending any messages. □

4 Full synchronization

In full synchronization (FS) all nodes must simultaneously enter the state *terminated* after receiving the source message. We first present a simple protocol for FS working in worst case time at most $3n$, for any n-node network. The advantage of this protocol is that it uses only messages of size logarithmic in n.

The idea of the protocol is the following. In the first phase, blind broadcasting is accomplished and simultaneously a spanning tree (of uncontrolled depth and shape) is constructed. In the second phase, gossiping (i.e., all-to-all broadcasting) is done

in this tree, thus informing all nodes that everybody got the source message, and permitting them to simultaneously enter the state *terminated*.

Protocol Simple-FS

Phase 1.

This phase is a simplified version of phase 1 of protocol ABB(ϵ). Nodes of the network perform blind broadcasting. When a node v gets the source message for the first time, and the sender of the message is neighbor u, v sends the message "you are my parent" to u in the next round, and then relays the source message on consecutive yet unsaturated edges, as usual. (As opposed to protocol ABB(ϵ), parents never change.) If node v gets the source message for the first time simultaneously from many neighbors, it chooses as its parent one of these nodes, arbitrarily. After getting the acknowledgement, u considers v as its child. The parent and all children of a node are called its *tree neighbors*. They are indeed its neighbors in the spanning tree which is being constructed.

Phase 2.

Phase 2 is started by nodes that got all their incident edges saturated and did not get the message "you are my parent" in the next round. In the special case when the source has degree 1, phase 2 is also started by the source, after it has send the message on its unique edge. The above nodes are *leaves* in the (unoriented) spanning tree constructed in phase 1. They start phase 2 by sending the message "my subtree has size 1" to their unique tree neighbor. All other nodes v whose all incident edges are saturated, proceed as follows. Suppose that the tree neighbors of v are w, v_1,..., v_k, and v got messages "my subtree has size x_i" from all nodes v_i, $i = 1, ..., k$. Then v sends the message "my subtree has size $x_1 + \cdots + x_k + 1$" to node w. Any node that got messages "my subtree has size y_i" from all its tree neighbors u_i, $i = 1, ..., k$, knows that all nodes have already got the source message and that the total number of nodes in the network is $n = y_1 + \cdots + y_k + 1$. It then waits until round $3n$ and enters the state *terminated*.

Notice that the size of messages should only permit sending numbers smaller than n, thus size logarithmic in n suffices.

Theorem 4.1 *Protocol Simple-FS correctly accomplishes full synchronization and works in worst case time at most* $3n$*, for any* n*-node network.*

Proof: Acknowledging the parent in phase 1 delays blind broadcasting by one round in every node, except the source. Hence, using the computation of the upper bound for BB from the proof of Theorem 2.1, we conclude that phase 1 of Protocol Simple-FS will be completed at the latest in round $(2n - d - 2) + d < 2n$. Leaves

need to wait at most 1 more round to learn that they are indeed leaves. Hence phase 2 is initiated by all leaves before round $2n$.

Phase 2 is completed in at most n further rounds because its duration is limited by the longest travel time between any pair of nodes. Hence before round $3n$ every node knows that all nodes have already got the source message and knows the size n of the network. Hence all nodes compute $3n$ before round $3n$, and consequently are able to enter the state *terminated* simultaneously in round $3n$. □

Our second protocol for full synchronization is slightly faster than Simple-FS but it uses large messages, similarly as Protocol Learn-All. Its additional advantage is that it informs all nodes about the entire topology of the network. In order to describe and analyze the protocol we need the following facts about time of classic broadcasting in relation to network connectivity.

A *2-edge-connected* (respectively, *2-node-connected*) component of a network is a maximal subnetwork for which there are at least two edge disjoint (respectively node disjoint) paths between any pair of distinct nodes.

Proposition 4.1 *Classic broadcasting in a 2-node-connected n-node network can be completed in time at most $\lceil n/2 \rceil$, for any $n \geqslant 3$.*

Proof: Any 2-node-connected network can be obtained from a cycle by successive applications of the following operations:

1. adding an edge,
2. splitting an edge (replacing an edge (u, v) by two edges (u, x), (x, v), where x is a new node).

We will prove the following property by induction on the number of operations as above, used to obtain a given 2-node-connected network from a cycle:

- For any 2-node-connected n-node network there exists a classic broadcasting protocol which informs all nodes in time $\lceil n/2 \rceil$, and all nodes except at most one in time $\lfloor n/2 \rfloor$.

The property is true for the cycle. Adding an edge does not increase broadcasting time. Suppose that the property is true for a 2-node-connected n-node network G and let P be the respective protocol. Without loss of generality assume that no node is informed by more than one node. Replace edge (u, v) by edges (u, x), (x, v). If n is even, P informs all nodes of G in time $n/2$ and x can be informed in time $n/2 + 1$.

If n is odd, P informs all nodes of G in time $(n+1)/2$, and at most one node of G gets the message in round $(n+1)/2$. If both u and v get the message before round $(n+1)/2$, x can be informed in round $(n+1)/2$. Otherwise, suppose that v gets the message in round $(n+1)/2$. Let z be a neighbor of v in G different from u (in a 2-node-connected network all nodes have degree at least 2). If v is not informed by u according to protocol P, u can inform x in round $(n+1)/2$. Otherwise z is idle in round $(n+1)/2$ according to P, and protocol P can be changed to P' replacing the call from u to v by the call from z to v. Now the call from u to x informing x in round $(n+1)/2$ can be added to P'. □

Any 2-edge-connected network can be viewed in the following way. Let T be any tree. Replace all nodes of T by any 2-node-connected networks in such a way that networks corresponding to nonadjacent nodes of the tree have disjoint sets of nodes, and networks corresponding to adjacent nodes of the tree have exactly one common node. The 2-node-connected networks replacing nodes of T are called *pieces* of the 2-edge-connected network.

Proposition 4.2 *Classic broadcasting in a 2-edge-connected n-node network can be completed in time at most $2n/3$, for any $n \geqslant 3$.*

Proof: We prove the following stronger property by induction on the number k of pieces in the n-node 2-edge-connected network:

For any fixed node v of the network, there exists a classic broadcasting algorithm which informs node v in time less than $2n/3$, and all other nodes in time at most $2n/3$.

The property is true for $k = 1$, in view of Proposition 4.1. Suppose that it is true for k, and let G' be a $(k+1)$-piece 2-edge-connected network. We can view G' as a k-piece 2-edge-connected network G with n nodes, to whose node w the $(k+1)$st piece of G' is attached. Call this piece P and suppose it has p nodes (including node w). Hence G' has $n+p-1$ nodes.

Consider a classic broadcasting algorithm for G which informs node w in time less than $2n/3$, and all other nodes of G in time at most $2n/3$.

If $p = 4$ or $p \geqslant 6$, the property for G' follows from Proposition 4.1 because in this case $\lceil \frac{p}{2} \rceil \leqslant \frac{2}{3}(p-1)$. Thus it remains to consider cases $p = 3$ and $p = 5$. An easy case-by-case analysis (considering values of $n \equiv 0, 1, 2 \pmod 3$) shows that the property holds for G', for these sizes of P as well. □

Any network can be viewed in the following way. Let T be any tree. Replace all nodes of T by any 2-edge-connected networks G_i with pairwise disjoint sets of nodes,

and join any pair of networks G_j, G_k corresponding to adjacent nodes of the tree by a simple chain whose extremities are in G_j, G_k, respectively, and all internal nodes are outside of all networks G_i and of all other chains.

We will describe Protocol Faster-FS whose execution time is slightly shorter than that of Simple-FS: It runs in time at most $2.9n$. Moreover, apart from achieving full synchronization, it informs all nodes about the entire topology of the network. However, since it is based on Protocol Learn-All, it uses large messages. Similarly as in case of Learn-All, we may assume in our description that nodes get distinct identities, upon receiving the source message.

The basic idea of the protocol is to inform the source about the topology of the network using Protocol Learn-All, and then use optimal classic broadcasting from the source to spread this knowledge among all nodes. In the classic broadcasting phase, nodes learn the topology and hence can continue broadcasting in an optimal way. The only problem is how a node identifies which of its links joins it to which neighbor. This can be done as follows. In the Learn-All phase, apart from conveying neighborhood information, every node marks each of its incident links by the round number during which this link was first used by it (to send or to receive). All of this information reaches the source which then adds these marks to the picture of the network, indicating that link uv is the one marked t by u and by v. Hence, in the classic broadcasting phase, nodes learn not only the topology of the network but in fact acquire sense of direction, i.e., learn a consistent labeling of links, and thus can continu optimal classic broadcasting.

Since Protocol Learn-All works in time $2n$, and classic broadcasting can be completed in time $\frac{2}{3}n$ in 2-edge-connected n-node networks (Proposition 4.2), the above idea yields time $\frac{8}{3}n$ for these networks. However, for general networks, the time of this protocol becomes longer. In fact, in the extreme case of the line with the source at the endpoint, classic broadcasting takes time n, which yields total time $3n$, similarly as for Simple-FS. This is due to the fact that for networks "similar" to the line, gathering information in the source and then spreading it back is not efficient. In fact, the other end of the line can learn the entire topology much faster. We will modify the Learn-All + Classic Broadcasting idea to take advantage of it.

A *bridge* in the network is a link ab whose deletion splits the network into two connected components induced by sets of nodes A and $\overline{A}$. If the source s is situated in A, we say that ab yields an $(A, \overline{A})$ split. If $|A| = \alpha$ and a is situated in A, we say that ab yields an $(\alpha, n-\alpha)$ *cut*, and that a is the *head* and b is the *tail* of the bridge. A network containing a bridge that yields an $(\alpha, n-\alpha)$ cut, with $\beta \leqslant \alpha \leqslant 1-\beta$, is called β-*balanced*. Otherwise, it is β-*unbalanced*. We say that a node u *detects* a

bridge ab, if u knows that ab is a bridge yielding an $(A, \overline{A})$ split, and additionally u knows the topology of A. Using Protocol Learn-All, the source can detect a bridge yielding an $(\alpha, n-\alpha)$ cut, in time at most 2α.

We now indicate how Protocol Learn-All should be modified to obtain Protocol Faster-FS.

Protocol Faster-FS

- When the source detects a bridge ab yielding an $(A, \overline{A})$ split, it broadcasts a *step* message in A, using an optimal classic broadcasting tree, and becomes *dormant*. The *step* message contains the entire topological information about A and a special flag for the bridge head a. After receiving it, a sends to bridge tail b a message "you are the acting source".

- After becoming an acting source, a tail b of a bridge yielding an $(A, \overline{A})$ split, considers nodes in A as *left* and nodes in $\overline{A}$ as *right*. It then acts as a source with respect to right nodes. When it detects a bridge $a'b'$ yielding a $(B, \overline{B})$ split in $\overline{A}$, it broadcasts a *step* message in B, using an optimal classic broadcasting tree, and becomes *dormant*.

- Conflicts can arise between messages of Protocol Learn-All and the *step* message. (A given node may be required to send these messages to two different neighbors at the same round.) In this case the *step* message has priority.

- When an acting source or a dormant node (that was previously an acting source) receives complete topological information about the part of the network induced by its right nodes, it knows the entire topology of the network. It then broadcasts this information to the right nodes and to the left nodes using optimal classic broadcasting trees in both cases. Messages involved in this final broadcast get *priority 1*. They have priority over all previously mentioned messages.

- When the original source s receives complete topological information about the entire network, it decides if the network is $\frac{2}{7}$-balanced or not. In the former case the source does nothing. In the latter case, the source initiates optimal classic broadcasting. Messages involved in it get *priority 0*. They have priority over all previously mentioned messages.

- After learning the entire topology of the network (and thus knowing n), every node waits till round $2.9n$ and enters state *terminated*.

Theorem 4.2 *Protocol Faster-FS correctly accomplishes full synchronization, informs all nodes about the topology of the n-node network, and works in worst case time at most* $2.9n$*, for sufficiently large* n*.*

Proof: We have to show that all nodes learn the topology of the network before round $2.9n$. Since, after learning the topology of the network, the source decides that *priority 0* messages are sent iff the network is $\frac{2}{7}$-unbalanced, the proof can be split into two cases, for $\frac{2}{7}$-balanced networks with *priority 0* messages, and for $\frac{2}{7}$-unbalanced networks without them.

Case 1. $\frac{2}{7}$-balanced networks.

Let ab be a bridge yielding an $(\alpha, n-\alpha)$ cut, with $\frac{2}{7} \leqslant \alpha \leqslant \frac{5}{7}$. Let $(A, \overline{A})$ be the corresponding split. Suppose that bridges detected before ab during the execution of Protocol Faster-FS are a_1b_1,..., a_kb_k, where a_i denote heads and b_i denote tails. Let a_1b_1 yield the split $(A_1, V(G) \setminus A_1)$, a_2b_2 the split $(A_1 \cup A_2, V(G) \setminus (A_1 \cup A_2))$,..., and a_ib_i the split $(A_1 \cup A_2 \cup \cdots \cup A_i, V(G) \setminus (A_1 \cup A_2 \cup \cdots \cup A_i))$. Denote $\alpha_i = |A_i|$. Node b_1 becomes the acting source in round at most $3\alpha_1$. It learns the topology of A_2 in round at most $2(\alpha_1+\alpha_2)$. Hence b_2 becomes the acting source in round at most $\alpha_2 + max(2(\alpha_1+\alpha_2), 3\alpha_1) \leqslant 3(\alpha_1+\alpha_2)$. It follows by induction that b_i becomes the acting source in round at most $3(\alpha_1 + \cdots + \alpha_i)$. Hence the tail b of bridge ab knows the topology of A in round at most 3α. Moreover, it knows the topology of $\overline{A}$ in round at most $2n$. It follows that b starts sending *priority 1* messages in round at most $max(2n, 3\alpha)+1$. The number of further rounds to complete the protocol is at most $max(\alpha, n-\alpha)$. Hence all nodes learn the topology of the network by round $max(2n, 3\alpha) + max(\alpha, n-\alpha)$, which is smaller than $2.9n$, since $\frac{2}{7} \leqslant \alpha \leqslant \frac{5}{7}$.

Case 2. $\frac{2}{7}$-unbalanced networks.

In this case there exists a 2-edge-connected component X of the network, with the following property. For every bridge ab incident to a node a of X, the connected component of the network resulting from removing ab, and containing b, has size at most $\frac{2}{7}n$. Let C_1,..., C_r be all these components. Denote by y the maximum of $|C_i|$ and by x the size of X. Suppose that the source is in some C_j (the argument is similar when the source is in X). Let $z = |C_j|$. Thus $y \leqslant \frac{2}{7}n$ and $z \leqslant \frac{2}{7}n$. By definition, $n \geqslant x + y + z + r - 2$.

The source knows the topology of the network in round at most $2z+2x+2y+r+2$. At this time it starts sending *priority 0* messages. Some node of X gets such a message in round at most $(2z + 2x + 2y + r + 2) + z + 1$. All nodes of X get such a message in round at most $(2z + 2x + 2y + r + 2) + (z + 1) + \frac{2}{3}x$, by Proposition 4.2.

It follows that all nodes of the network get such a message in round at most

$$(2z + 2x + 2y + r + 2) + (z + 1) + \frac{2}{3}x + (r - 1 + y) = 3(z + y) + \frac{8}{3}x + 2r + 2 \leqslant$$

$$3(z + y) + \frac{8}{3}(n - (z + y)) + 8 = \frac{8}{3}n + \frac{1}{3}(z + y) + 8 \leqslant \frac{8}{3}n + \frac{4}{21}n + 8,$$

which is smaller than $2.9n$, for sufficiently large n. □

The optimal time of full synchronization remains an open problem, both using logarithmic messages, and when arbitrary messages are allowed. However, it should be noted that if the source knows *a priori* the size n of the network, full synchronization can be easily achieved in time at most $2n$, using only messages of logarithmic size. Indeed, it is sufficient to perform blind broadcasting, adding the starting time and the value n to the original message. In view of Theorem 2.1 all nodes get the message before round $2n$. Then, knowing n, they wait until round $2n$ and enter state *terminated*.

5 Conclusion

We considered the number of rounds required for three communication tasks of increasing difficulty, performed by nodes entirely ignorant of the network. In blind broadcasting (BB) the goal is to communicate the source message to all nodes. In acknowledged blind broadcasting (ABB) the goal is to achieve BB and inform the source about it. In full synchronization (FS) all nodes must simultaneously enter the state *terminated* after receiving the source message. We showed algorithms yielding upper bounds $2n$, $(2 + \epsilon)n$, and $3n$, for these tasks, respectively, where n is the number of nodes. These algorithms use messages of size $O(\log n)$. We showed that using large messages, completion times for ABB and FS can be slightly decreased and, moreover, the topology of the network can be learned. We also proved that in some networks these tasks cannot be achieved faster than in $2n - o(n)$ rounds. While this gives very tight estimates in case of BB and ABB, the optimal time of full synchronization remains an open problem.

It also seems interesting to study other communication problems, such as gossiping (i.e., all-to-all broadcasting) or multicasting, under the network ignorance scenario. Other important questions concern the impact of partial knowledge of the network (e.g., nodes knowing only such parameters as the size, the diameter, or the maximum degree of the network) on the efficiency of communication protocols.

References

[1] R. Ahlswede, H.S. Haroutunian, L.H. Khachatrian, Messy Broadcasting in Networks, in: Communications and Cryptography, eds. R.E. Blahut, D.J. Costello, Jr., U. Maurer, T. Mittelholzer (Kluwer, Boston/Dordrecht/London, 1994), 13-24.

[2] S. Albers and M. R. Henzinger, Exploring unknown environments, Proc. 29th Symp. on Theory of Computing (1997), 416-425.

[3] H. Attiya, M. Snir and M. Warmuth, Computing on an Anonymous Ring, Journal of the ACM, 35 (1988), 845-875.

[4] H. Attiya and M. Snir, Better Computing on the Anonymous Ring, Journal of Algorithms 12 (1991), 204-238.

[5] B. Awerbuch, M. Betke, R. Rivest and M. Singh, Piecemeal graph learning by a mobile robot, Proc. 8th Conf. on Comput. Learning Theory (1995), 321-328.

[6] R. Bar-Yehuda, O. Goldreich, and A. Itai, On the time complexity of broadcast in radio networks: An exponential gap between determinism and randomization, Proc. 6th ACM Symposium on Principles of Distributed Computing (1987), 98 - 108.

[7] X. Deng and C. H. Papadimitriou, Exploring an unknown graph, Proc. 31st Symp. on Foundations of Computer Science (1990), 356-361.

[8] K. Diks, E. Kranakis A. Malinowski and A. Pelc, Anonymous wireless rings, Theoretical Computer Science 145 (1995), 95-109.

[9] K. Diks, E. Kranakis and A. Pelc, Broadcasting in unlabeled tori, Parallel Processing Letters 8 (1998), 177-188.

[10] K. Diks, E. Kranakis and A. Pelc, Perfect broadcasting in unlabeled networks, Discrete Applied Mathematics 87 (1999), 33-47.

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

[12] U. Feige, D. Peleg, P. Raghavan and E. Upfal, Randomized broadcast in networks, Random Structures and Algorithms 1 (1990), 447-460.

[13] P. Flocchini, B. Mans and N. Santoro, Sense of Direction: Formal Definitions and Properties, Proc. 1st Int. Conf. on Structural Information and Communication Complexity (1994), 9-34, Carleton University Press, 1995.

[14] P. Fraigniaud and E. Lazard, Methods and problems of communication in usual networks, Disc. Appl. Math. 53 (1994), 79-133.

[15] H.A. Harutyunyan and A.L. Liestman, Messy broadcasting, Parallel Processing Letters 8 (1998), 149-160.

[16] S.M. Hedetniemi, S.T. Hedetniemi and A.L. Liestman, A survey of Gossiping and Broadcasting in Communication Networks, Networks 18 (1988), 319-349.

[17] E. Kranakis, Symmetry and Computability in Anonymous Networks: A Brief Survey, Proc. 3rd Int. Conf. on Structural Information and Communication Complexity (SIROCCO'96), Carleton University Press, 1997, 1-16.

[18] E. Kranakis and D. Krizanc, Distributed Computing on Anonymous Hypercube Networks, Journal of Algorithms 23 (1997), 32-50.

[19] E. Kranakis, D. Krizanc and J. van der Berg, Computing Boolean Functions on Anonymous Networks, Information and Computation, 114 (1994), 214-236.

[20] P. Panaite and A. Pelc, Exploring unknown undirected graphs, Proc. 9th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA'98), 316-322.

[21] N. Santoro, Sense of Direction, Topological Awareness, and Communication Complexity, ACM SIGACT News 16 (1984), 50-56.

Contents

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399