Proceedings of the IASTED International Conference
PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS
November 6-9, 2000, Las Vegas, Nevada, USA

Gossiping with Multicasting Communication Primitives

TEOFILO F. GONZALEZ

Department of Computer Science
University of California
Santa Barbara, CA 93106, USA
teo@cs.ucsb.edu

Abstract

The gossiping problem consists of an n processor communication network, N, in which every processor has to broadcast a single message. We present an efficient algorithm to generate a communication schedule with total communication time at most $n + 3r - 1$, where r is the radius of the network. Our algorithm begins by constructing a spanning tree (or tree network T) with least possible radius. In the second step all the communications are carried out in the tree network as follows: Each processors waits its turn to transmit consecutively to its parent and children all the messages in its subtree. Before and after these communications, each processor must transmit to its children all the messages emanating elsewhere in the network. We briefly discuss an algorithm that generates a communication schedule with total communication time at most $n + r + 1$.

Key Words: Approximation Algorithms, Gossiping, Multicasting, Scheduling.

1 Introduction

1.1 The Problem

Let N be any n processor (or node or vertex) communication network (or graph). The *broadcasting* problem defined over N consists of sending a message from one processor in the network to all the remaining processors. The *gossiping* problem over N consists of broadcasting n messages each originating at a different processor. Gossiping problems have been studied under many different objective functions and communication models. Our communication model allows each processor to multicast one message to any subset of its adjacent processors, but no processor may receive more than one message at a time. Our objective is to determine when each of these messages is to be transmitted so that all the communications can be carried in the least total amount of time. As we shall see later on, multicasting is a powerful communication primitive that allows for the communications to be performed much faster than when restricting to the telephone (or unicasting) communication model (a processor may transmit a message to just one of its adjacent processors at a time) or broadcasting communication model (a processor may transmit a message to all the adjacent processors) communication primitives.

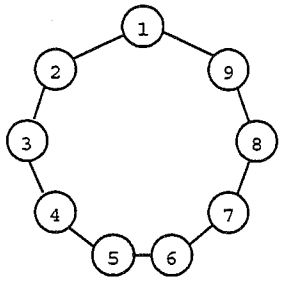

Figure 1: Network (N_1) with a Hamiltonian circuit.

Example 1. There are nine processors ($n = 9$) in network N_1 (Figure 1). The first communication in an optimal schedule for this problem instance is for each processor to send to its right hand side neighbor the message it holds, and then it the next 7 iteration every processor transmits to its right hand side neighbor the message it just received on its left hand side link (edge). In this case it is simple to verify that all the communications can be carried out in $n - 1$ steps, which is best possible under our communication model.

Let us formally define our problem. Initially each processor P_i *holds* one message in its hold set h_i and *needs* to receive the remaining $n - 1$ messages. The communications allowed in our network must satisfy the following two restrictions.

1.- During each time unit each processor P_i may transmit one of the messages it holds (i.e., a message in its hold set h_i at the beginning of the time unit), but such message can be multicasted to a set of processors adjacent to P_i. The message will also remain in the hold set h_i.

2.- During each time unit each processor may receive at most one message provided such message was sent

during the previous time unit. The message that processor P_i receives (if any) is added to its hold set h_i at the beginning of the time unit when it was received.

The communication process ends when each processor has the n messages. The above communication rules define a communication mode (or step) for a communication schedule as follows. A *communication mode* C is a set of tuples of the form (m, l, D), where l is a processor index ($1 \leq l \leq n$), and message $m \in h_l$ is to be multicasted from processor P_l to the set of processors with indices in D. In addition the set of tuples in a communication mode C must obey the following communications rules imposed by our network:

1.- All the indices l in C are distinct, i.e., each processor sends at most one message; and

2.- Every pair of D sets in C are disjoint, i.e., every processor receives at most one message.

A *communication schedule* S for a problem instance I is a sequence of communication modes such that after performing all of these communications every processor will hold the n messages. The *total communication time* is the number of communication modes in schedule S, which is identical to the latest time there is a communication. Our problem consists of constructing a communication schedule with least total communication time. From the communication rules we know that in every problem instance every processor needs to receive $n-1$ messages and since no processor may receive two or more messages simultaneously, it follows the $n-1$ is a trivial lower bound on the total communication time. Therefore the schedule we constructed for network N_1 in Example 1 is an optimal one.

In this paper we are mainly concerned with the off-line gossiping problem, i.e., the schedule is constructed by a processor that knows all the information about the problem ahead of time.

Example 1 suggests a method for solving the gossiping problem. The idea is to first construct a Hamiltonian circuit and then use that circuit as in Example 1 to transmit all the messages in $n-1$ time units. As it is well know, the Hamiltonian circuit problem is an NP-complete problem and it is conjecture that there is no efficient algorithm for its solution. Fortunately, it is not sufficient for a network to have a Hamiltonian circuit in order for the gossiping problem be solvable in $n-1$ steps. There are networks that do not have a Hamiltonian circuit, but in which gossiping can be performed in $n-1$ communication steps even under the telephone communication model. Example 2 gives a network that does not have a Hamiltonian circuit, but in which gossiping can be performed in $n-1$ communication steps under the multicasting communication model but not under the telephone communication model. Since the telephone communication model is a restricted version of the multicasting communication model, the example establishes that multicasting is much more efficient way to communicate.

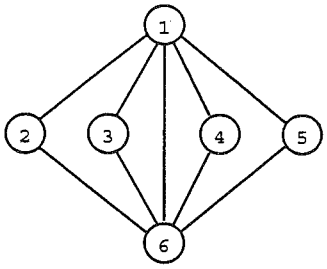

Figure 2: Network (N_2).

Example 2. There are six processors ($n = 6$) in network N_2 (Figure 3). A communication schedule with total communication time 5 is given in Table 1. Every communication schedule with total communication time $n-1$ has at each time unit each of the 5 processors receiving a message. But in network N_2 processor 2, 3, 4 and 5 can only send a message to processors 1 and 6. Therefore under the telephone communication model it is impossible to send 5 messages to five destinations at time 0 and there does not exist a communication schedule with total communication time $n-1=5$ for N_2 under the telephone communication model.

Table 1: Optimal Gossiping (with multicasting).

Time	Message	From Processor $\rightarrow$ To Proc.
T_0	M_1	$P_1 \rightarrow \{P_2, P_3, P_4, P_5\}$
	M_2	$P_2 \rightarrow \{P_6\}$
	M_6	$P_6 \rightarrow \{P_1\}$
T_1	M_6	$P_1 \rightarrow \{P_2, P_3, P_4, P_5\}$
	M_3	$P_3 \rightarrow \{P_6\}$
	M_2	$P_6 \rightarrow \{P_1\}$
T_2	M_2	$P_1 \rightarrow \{P_3, P_4, P_5\}$
	M_4	$P_4 \rightarrow \{P_6\}$
	M_3	$P_6 \rightarrow \{P_1, P_2\}$
T_3	M_3	$P_1 \rightarrow \{P_4, P_5\}$
	M_5	$P_5 \rightarrow \{P_6\}$
	M_4	$P_6 \rightarrow \{P_1, P_2, P_3\}$
T_4	M_1	$P_2 \rightarrow \{P_6\}$
	M_4	$P_1 \rightarrow \{P_5\}$
	M_5	$P_6 \rightarrow \{P_1, P_2, P_3, P_4\}$

The above examples suggest that it is always possible to perform gossiping in our communication model in $n-1$ steps. However, that is not the case. Consider the straight line network. In this line network it is impossible to deliver a new message to each end of the line during each time period, though it is possible to deliver it to only one of its end points.

1.2 Previous Work, New Results, and Applications

The broadcasting and gossiping problems are not new, these problems have been studied for the past three decades [11]. However, most of the work is for the telephone type of communication, i.e., at every step each processor may send at most one message to at most one processor and no processor may receive more than one message at a time. Also, most of the previous work allows for up to n messages to be transmitted over a single link at a time. In other words, the transmission packets must be of size $\Omega(n)$. This implies that such algorithms are not scalable. Under the traditional communication model these problems are computationally difficult, i.,e., NP-hard. But there are efficient algorithms to construct optimal communication schedules for restricted networks under some communication models [3, 6, 15]. Up to now there is no known polynomial time approximation algorithm with fixed approximation ratio for the broadcasting problem defined over arbitrary graphs, i.e., there is no known efficient approximation algorithm A such that $f(\hat{I})/f^*(I) \leq c$ for every problem instance I, where $f(\hat{I})$ is the total communication time for the schedule constructed by algorithm A for problem instance I, $f^*(I)$ is the total communication time of an optimal schedule for problem instance I, and c is a constant. Determining whether or not such algorithm exists has been an intriguing open problem for more than two decades. The best known approximation algorithms appear in [12, 15], and a randomized algorithm is presented in [4].

Broadcasting under our communication model is trivial to solve. At time zero, the processor that has the message broadcasts it to all its neighbors. Then at each iteration, each processor that just received a message will plan to multicast it to all its neighbors that do not have the message. But, if there are two or more processors currently planning to send a processor the message, then only one of them will actually send it. Once the round of communications is completed, we start with the next iteration. It is simple to see that every processor i receives a message at time j if, and only if, the shortest path (remember that all edges have weight one) from the broadcasting node to vertex i has j edges. Clearly, the total communication time in the communication schedule generated by the above procedure is equal to the maximum length of a shortest path from the broadcasting node to any vertex in the graph. The above algorithm is clearly off-line.

A variation of the gossiping problem in which there are costs associated with the edges and there is a bound on the maximum number of packets that can be transmitted through a link at each time unit has been studied in [5]. Approximation algorithms for several versions of this problem are given in [5, 1, 7].

Routing under the multicasting communication model has been considered in [14, 8, 10, 9]. But they study the multimessage multicasting problem. In this problem each processor needs to transmit a set of messages, but each message is to be received by its own subset of processors. Shen [14] studied the problem for hypercube connected processors, and Gonzalez [8, 10, 9] considered the problem for fully connected processors and also for processors interconnected via a multistage interconnection network that satisfies some simple properties (e.g. the MEIKO CS-2 parallel computer system).

In this paper we study the gossiping problem under the multicasting communication model. Our main motivation is that this communication model has been available for many years and allow us to generate solutions with fewer communication steps than the telephone communication model. Gossiping arises in many application [2, 13], that include sorting, matrix multiplication, Discrete Fourier Transform, solving linear equations, etc.

2 Algorithms

We discuss in this section our algorithm to generate a communication schedule with total communication time at most $n + 3r - 1$, where r is the network radius. The *radius* of a network is the least integer r such that every vertex v in the network has a path from v to each vertex in the graph with at most r edges. Our procedure consists of two steps. First we build a special tree network (subsection 2.1) and then we perform all the communications in that tree network (subsection 2.2).

2.1 Constructing the Tree Network

As we mention in the previous section the first step of the algorithm is to construct a spanning tree of minimum radius. To do this we begin by finding the length of the shortest path between all pairs of vertices. Then we select a processor in the network such that the maximum length of a shortest path for it to all vertices in the network is least possible and construct a tree rooted at that node in which all the paths to the other vertices are shortest paths in the original network. Then we perform all the communications in that tree network.

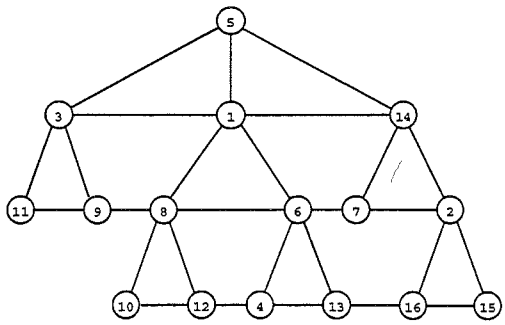

Figure 3: Network.

Applying this procedure to the network in Figure 3 re-

sults in the network given in Figure 4. In the next subsection we present several algorithms for gossiping in trees.

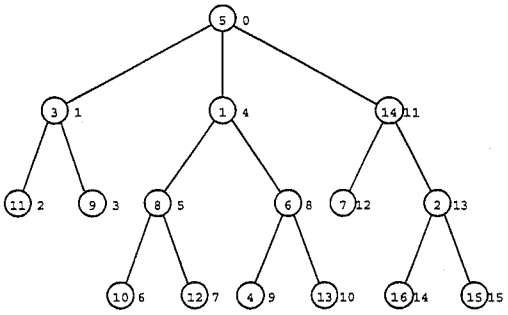

Figure 4: Tree Network generated from the Graph in Figure 3.

2.2 Gossiping in Tree Networks

The problem of gossiping in an arbitrary network has been reduced to gossiping in a tree with height r. Our algorithm for gossiping in trees is a little bit complex, so before we discuss it we introduce increasingly more complex procedures. Lets begin by defining some terms. The topmost vertex is called the *root* of the tree. The *level* of every vertex in the tree network is defined as follows: the level of root is zero, the level of the children of the root is equal to one, the grandchildren are at level 2, and so forth. For every vertex we sort its subtrees from left to right. Our algorithm proceeds by labeling the nodes in the tree as follows.

```
Initially count =0.
Call to Label-it(root);
...
Label-it(t)
 label vertex t with count
 count++
 for every child r of t
     from left to right do
   Label-it(r);
 endfor
End Label-it
```

Applying the algorithm to the problem tree in Figure 4 we obtain the labels that appear to the right of the vertices.

We begin by discussing the first algorithm (`Simple`) to perform the gossiping in $2n + r$ time units. This procedure has been used to solve other message routing problems. The idea is to send up all the messages to the root first so that message i is received by the root at time i. The message labeled i at level k is transmitted to its parent (if any) at time $i - k$, to its grandparent (if any) form its parent at time $i - k + 1$, and so on. Clearly, there are no conflicts and at time n all the messages are received by the root. Clearly this process takes n communication steps. Now all the messages need to be propagated downwards. At time n message 1 is sent from the root to all its children, at time $n + 1$ message 2 is sent from the root to all its children and so on. Note that during all this process when a non-root vertex receives a message from its parent it immediately sends it to all its children. It is simple to verify that by time $2n + r$ all nodes will receive all the messages. The main advantage of procedure `Simple` is that it is quite simple, but on the other hand the total communication time is not so small.

Theorem 2.1: The communication schedule generated by procedure `Simple` has total communication time $2n + r$ for any tree with n nodes and height r.

Proof. The proof follows from the above discussion. □

Our next procedure (`UpDown`), is more complex, but the communication schedule it generates has smaller total communication time. The approach is similar to procedure `Simple`, except at the same time the algorithm sends messages up and down throughout the tree. The procedure consists of two phases. In the first phase, like in the algorithm `Simple`, all the messages are propagated to the root, but at the same time it begins the process of propagating messages to other parts of the tree. In the second phase the algorithm just propagates down some messages that got stuck in the network. The first and second phase take $n + r$ and $2(r - 1) + 1$ steps, respectively.

In the first phase we propagate all messages to the root and at the same time we propagate most of them throughout the network. Let us now introduce additional notation. Consider vertex v at level $k \leq 1$. Vertex v has message i initially and the subtree rooted at v includes messages i up to message j initially. The parent of vertex v, which we refer to as v', has message i' initially and the subtree rooted at vertex v' includes messages i' up to message j' initially. The level of vertex v' is $k' = k - 1$. To simplify the notation we say that the root of the tree has a (virtual) parent called v' with $i' = 0$ and $j' = n$.

The messages in every non-root vertex v are labeled as follows:

- messages $1, 2, ..., i - 1$ are called the *front* messages or *f-messages*.
- messages $i, i + 1, ..., j$ are called the *body* messages or *b-messages*. The b-messages are partitioned with respect to vertex v as follows:
  - message i is called the *original* message or *o-message*.
  - message $i + 1$, if $i + 1 \leq j$, is called the *lookahead* messages or *l-message*.
  - messages $i + 2, ..., j$ (if any) are called the *remaining* messages or *r-message*.

  The b-messages are also partitioned with respect to vertex v', (the parent of vertex v) as follows:

- message i, if $i = i' + 1$, is the *lookahead in parent (lip)* message or *lip-message*.
- messages $max\{i, i'+2\}, ..., j$, if any, are the *remaining in parent (rip)* messages or *rip-message*.

- messages $j+1, ..., n$ are the *end* messages or *e-message*.

- At each vertex no more than 2 f-messages (or the o-message in the root vertex) will also be labeled delayed-messages (d-messages). Note that if an f-message is labeled as a d-message in a vertex, then in another sibling vertex it might not be labeled as a d-message.

Note that the root of the tree ends up labeled as follows: message $i = 0$ is the o-message and all the messages $1..n$ are called r-messages. There are no l-messages and message 0 is a d-message.

First we establish that a set of messages will be sent to the root as specified by algorithm Up and then we show that algorithm Down propagates all messages to all the vertices. Both of these algorithms will end up operating concurrently. Algorithm Up will guarantee that all the b-messages of each vertex v will be available by time $j - k$ at v. This implies that the root of the tree will receive all the messages by time n. When all the communications of Algorithm Down finish every vertex v will have all the messages except for all the d-messages in the predecessors of v. To complete the dissemination of information one needs to transmit all the d-messages at every vertex to all its descendants. In order for these two algorithms to deliver all the messages to all the vertices quickly, the algorithms are interleaved.

`Algorithm Up (v)`

1. {Time 1} At time 1 vertex v receives from a child its l-message (if any). Specifically, if $i + 1 \leq j$ (v is not a leaf vertex), then vertex v receives message $i + 1$ at time 1.

2. {Time $i - k + 2..j - k$} Starting at time $i - k + 2$ vertex v receives sequentially from its children all its r-messages. This is equivalent to saying, if message $i + \alpha$ is an r-message, it will be received by v at time $i+\alpha-k$, simply because the first r-message, if any, is message $i+2$ and it is received at time $i-k+2$, then the remaining r-messages are received sequentially in order.

3. {Time 0} If v is not the root of the tree, then at time 0 vertex v sends to its parent its lip-message.

4. {Time $i-k+w..j-k$} If v is not the root of the tree, then starting at time $i - k + w$ send sequentially to its parent all its rip-messages, where w is the number of lip-messages at v. This is equivalent to saying that if message $i' + \alpha$ is a rip-message at v then it will be sent at time $i' + \alpha - k$ to its parent, simply because each of these messages is sent k units ahead of time. The first of these messages is labeled $i + w$ and it is sent at time $i-k+w$ and the remaining rip-messages will be sent sequentially in order.

`End of Algorithm Propagate-Up`

Lemma 2.1: Algorithm Propagate-Up is feasible, i.e., every vertex v in the tree receives the messages in steps (1) - (2) as specified, and every non-root vertex v in the tree sends the messages in steps (3) - (4) as specified.

Proof. The proof is by induction on the height of the subtree rooted at v. For brevity the proof is omitted. □

`Algorithm Propagate-Down (v)`

If vertex v is the root of the tree then perform (1), else perform operations (2) - (5) and if v is not a leaf-node then also perform operations (6) - (10).

1. {Time $1..n$} The root of the tree propagates its information down as follows: for time $t = 1, 2, ..., n$ send to all its children message i (except for the child that already has the message). Message 0 will be labeled as d-message.

2. {Time $k + 1..i - k + 1$} Starting at time $k + 1$ vertex v receives from its parent all its f-messages except for no more than $2(k - 1) + 1$ of them which have been labeled d-messages at predecessor vertices of v These messages are not necessarily received one after the other.

3. {Time $j + k + 2..n + k$} Starting at time $j + k + 2$ vertex v receives from its parent all the e-messages. These messages are not necessarily received one after the other.

4. {Time $k + 1..i - k - 1$} Starting at time $k + 1$ vertex v sends sequentially to its children the f-messages that are not d-messages in predecessors of v. The f-messages received at time $i - k - 2$ and $i - k - 1$ (if any) will be called d-messages at d-messages for v.

5. {Time $i - k..j - k$} Starting at time $i - k$ vertex v sends sequentially to its children (except for the children that already have them) all its b-messages.

6. {Time $j+k+2..n+k$} Starting at time $j+k+2$ vertex v sends the e-messages it receives to all its children.

`End of Algorithm Propagate-Down`

Lemma 2.2: If Algorithm Propagate-Down is feasible then Algorithm Propagate-Up is feasible, i.e., the root of the tree propagates the messages as in (1) and for the remaining vertices the messages are received as specified by steps (2) - (3), and the messages in steps (4) - (6) will be sent as indicated.

Proof. The proof is by induction on the level of v. For brevity the proof is omitted. □

The second phase in procedure UpDown is simple. We just propagate downwards all the d-messages. There are at most two d-messages in each vertex v that need to be propagated to all descendants of v. This operation is performed by each processor sending to all its children (if any) starting at time 0 the d-message it holds which are originally at that vertex or which have been received at this phase. Clearly this process takes $2(r-1)+1$ communication steps since each vertex has at most two d-messages and the root of the tree has only one such message.

Theorem 2.2: The communication schedule generated by procedure UpDown takes $n + 3r - 1$ for any tree with n nodes and height r.

Proof. The proof is based on the above discussion. □

The third algorithm, FastUpDown, is the most complex one and it is based on the observation that all the operations can be carried out in phase two of the previous algorithm except for the propagation of the message labeled zero. The total communication time for this algorithm is $n + r + 1$ steps. We will present this new algorithm in a subsequent paper.

3 Conclusion

We have presented algorithms to construct communication schedules with total communication time at most $n+3r-1$, where r is the radius of the graph, and discussed a way to decrease this bound by $2r - 2$. The algorithms are efficient and generate near optimal solutions. With a little bit of preprocessing our algorithm can be made on-line provided there is a general synchronization process. For brevity we cannot elaborate on this extension. The most time consuming part of the algorithm is solving the all pair shortest paths problem. This information is needed to compute the radius of the network and then building a tree network with least height. All the other steps of the algorithm take $O(n)$ time.

References

[1] J. C. Bermond, L. Gargano, C. C. Rescigno and U. Vaccaro, "Fast Gossiping by Short Messages," *SIAM Journal on Computing* Vol. 27, No 4, (1998), pp. 917 – 941.

[2] D. P. Bertsekas and J. N. Tsitsiklis, "Parallel and Distributed Computation: Numerical Methods," Prentice-Hall, Englewood Cliffs, NJ, 1989.

[3] S. Even and B. Monien, "On the Number of Rounds Necessary to disseminate Information," Proc. 1st ACM Symp. on Parallel Algorithms and Architectures, Santa Fe, NM, 1989, pp. 318 – 327.

[4] U. Feige, D. Peleg, P Raghavan and E. Upfal, "Randomized Broadcast in Networks," International Symposium SIGAL '90, Lecture Notes in Computer Science, Springer-Verlang, Berlin, 1990, pp. 128 – 137.

[5] P. Fraigniaud and S. Vial, "Approximation Algorithms for Broadcasting and Gossiping," *Journal of Parallel and Distributed Computing,* Vol. 43, (1997), pp. 47 – 55.

[6] S. Fujita and M. Yamashita, "Optimal Group Gossiping in Hypercube under Circuit Switching Model," *SIAM Journal on Computing* Vol. 25, No 5, (1996), pp. 1045 – 1060.

[7] L. Gargano, A. A. Rescigno and U. Vaccaro, "Communication Complexity of Gossiping by Packets," *Journal of Parallel and Distributed Computing,* Vol. 45, (1997), pp. 73 – 81.

[8] T. F. Gonzalez, "Complexity and Approximations for MultiMessage Multicasting," *Journal of Parallel and Distributed Computing,* 55, (1998), 215 – 235.

[9] T. F. Gonzalez, " Simple Algorithms for MultiMessage Multicasting with Forwarding," *Algorithmica,* to appear.

[10] T. F. Gonzalez, "Improved Approximation Algorithms for MultiMessage Multicasting," *Nordic Journal on Computing,* Vol. 5, 1998, 196 – 213.

[11] S. Hedetniemi, S. Hedetniemi and Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks," *NETWORKS,* 18 (1988), pp. 129 – 134.

[12] J. Hromkovic, R. Klasing, B. Monien and R. Peine, "Dissemination of Information in Interconnection Networks (Broadcasting and Gossiping)," In D. Z. Du and D. F. Hsu (Eds.). Kluwer Academic, 1995, pp. 273 – 282.

[13] D. W. Krumme, K. N. Venkataraman and G. Cybenko, "Gossiping in Minimal Time," *SIAM Journal on Computing* Vol. 21, No 2, (1992), pp. 111 – 139.

[14] H. Shen, "Efficient Multiple Multicasting in Hypercubes," *Journal of Systems Architecture,* Vol. 43, No. 9, Aug. 1997.

[15] R. Ravi, "Rapid Rumor Ramification," Proc. 35th Annual Symp. on Foundations of Computer Science, 1994, pp. 202 – 213.