Gossiping in the Multicasting Communication Environment

Teofilo F. Gonzalez
Department of Computer Science
University of California
Santa Barbara, CA, 93106
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 $n + r$, where r is the radius of the network, when message multicasting is allowed. Our algorithm begins by constructing a spanning tree (or tree network T) with least possible radius. Then all the communications are carried out in the tree network as follows. Each processor 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.

1 Introduction

1.1 The Problem

Let N be any $n \geq 3$ 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* (at each time unit a processor may transmit a message to just one of its adjacent processors and receive at most one message) or *broadcasting communication model* (at each time a processor may transmit a message to all the adjacent processors, but receive at most one message) communication primitives.

Example 1: There are nine processors ($n = 9$) in a ring network N_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 in the next seven iterations 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 *multicasting communication model* 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 transmitted simultaneously 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

0-7695-0990-8/01/$10.00 (C) 2001 IEEE

$(1 \leq l \leq n)$, 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 that $n-1$ is a trivial lower bound on the total communication time. Therefore the schedule we constructed for network N_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 known, that the Hamiltonian circuit problem is an NP-complete problem and it is conjectured 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.

Figure 1 gives network N_2 that does not have a Hamiltonian circuit 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 above example establishes that multicasting is much more efficient way to communicate.

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.

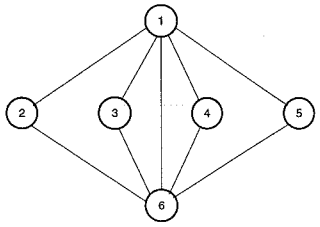

Figure 1. Network N_2.

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 [12]. 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 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 [13, 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. This is trivial to solve because we are dealing with the off-line scheduling problem. 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 [16, 8, 9, 11]. 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 [16] studied the problem for hypercube connected processors, and Gonzalez [8, 9, 11] 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, 14], 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 $n + r$, where r is the network radius. The *radius* of a network is the least integer r such that for some vertex v_0 there is a path from every vertex v in the graph to vertex v_0 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 above, the first step of the algorithm constructs a minimum radius spanning tree. 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.

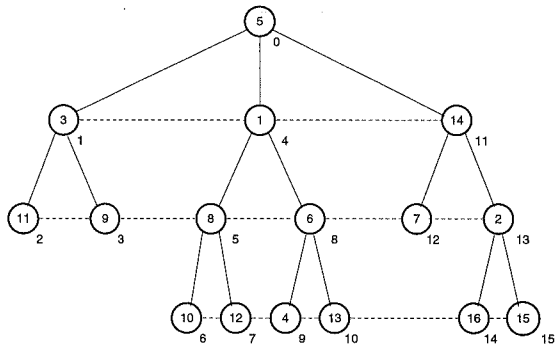

Figure 2. Network N_3 (solid and dashed lines) and Tree Network T (solid lines).

Applying this procedure to the network N_3 in Figure 2 (solid and dotted edges) results in the tree network T given in Figure 2 (solid edges). In the next subsection we discuss several algorithms for gossiping in trees.

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. Let us 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 vertices in in depth-first search order starting at the root (label 0) and ending at some leaf (label $n - 1$). Applying the algorithm to the tree network T in Figure 2 (solid edges) we obtain the labels that appear to the right and just below the vertices.

We begin by discussing the first algorithm (`Simple`) to perform the gossiping in $2n + r - 3$ 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 $i \geq 1$ 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 $n - 1$ all the messages are received by the root. Clearly this process takes $n - 1$ communication steps. Now all the messages need to be propagated downwards. At time $n - 2$ message 0 is sent from the root to all its children, at

time $n-1$ message 1 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 $2n+r-3$ 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.

Lemma 1: The communication schedule generated by procedure `Simple` has total communication time $2n+r-3$ for any tree with n nodes and height r.

Proof: The proof follows from the above discussion.

Procedure `UpDown` developed by Gonzalez [10] 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 $n+r-1$ and $2(r-1)+1$ steps, respectively.

In this paper we introduce algorithm `ConcurrentUpDown` which is the most complex of our procedures, but generates the best schedules with respect to the total communication time. It is based on the observation that all the operations can be carried out in a single stage by maximizing the concurrent operations performed at each step. The total communication time of the schedules generated by this algorithm is just $n+r$.

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 non-root 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 $j'=n$. Remember that vertices are labeled from 0 to $n-1$.

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*. The f-messages are partitioned with respect to vertex v as follows:
  - messages $1, 2, ..., i'-1$ are called the *front in parent* messages or *fip-messages*.
  - messages $i', i'+1, ..., i-1$ are called the *new front* messages or *nf-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* message or *l-message* .
  - messages $i+2, ..., j$ (if any) are called the *remaining* messages or *r-messages*.

  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-messages*.
- messages $j+1, ..., n-1$ are the *end* messages or *e-message*. The e-messages are partitioned with respect to vertex v as follows:
  - messages $j+1, j+2, ..., j'$ are called the *new end* messages or *ne-messages*.
  - messages $j'+1, ..., n-1$ are called the *new end in parent* messages or *eip-messages*.
- message 0 is called the *really last* message or *rl-message*.

Note that the root of the tree ends up labeled as follows: message $i=0$ is the o-message, message 1 is the l-message, messages $2..n-1$ are called r-messages and all messages are rip-messages. There are no lip-messages and message $i=0$ is also called an rl-message.

First we establish that a set of messages will be sent to the root as specified by algorithm Propagate-Up and then we show that algorithm Propagate-Down propagates all messages to all the vertices. Both of these algorithms will end up operating concurrently. Algorithm Propagate-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 contain all the messages by time $n-1$. When all the communications of Algorithm Propagate-Down finish every vertex will have all the messages. In order for these two algorithms to deliver all the messages to all the vertices quickly, the algorithms are interleaved.

`Algorithm Propagate-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 in order. 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. 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: 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 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) - (6) and if v is not a leaf-node then also perform operations (7) - (10).

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

2. {Time $k+1..i'-k+1$} Starting at time $k+1$ vertex v receives from its parent all its fip-messages except for no more than $2(k-1)$ of them which will be received later on by v. These messages are not necessarily received one after the other or in order.

3. {Time $i'-k+2..i-k+1$} Starting at time $i'-k+2$ vertex v receives from its parent all its nf-messages. These messages are received one after the other and in order.

4. {Time $j-k+3..j'-k+2$} Starting at time $j-k+3$ vertex v receives from its parent all its ne-messages $j+1..j'-1$. These messages are received one after the other and in order.

5. {Time $j'-k+3..n+k$} Starting at time $j'-k+3$ vertex v receives from its parent all its eip-messages plus the fip-messages (no more than $2(k-1)$) which were not received earlier. These messages are not necessarily received one after the other or in order.

6. {Time $n+k$} At time $n+k$ vertex v receives the rf-message.

7. {Time $k+1..i-k-1$} Starting at time $k+1$ vertex v sends sequentially to its children the f-messages it received in (2) and (3) except possibly for the ones received at time $i-k$ and $i-k+1$ (if any), which will be sent later on.

8. {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.

9. {Time $j-k+1$ and $j-k+2$} At time $j-k+1$ and $j-k+2$ vertex v sends to its children the f-messages received at time $i-k$ and $i-k+1$ (if any).

10. {Time $j-k+3..n+k$} Starting at time $j-k+3$ vertex v sends the e-messages it receives to all its children. These messages are the e-messages, the rf-message and the f-messages that were postpones at previous levels.

`End of Algorithm Propagate-Down`

Lemma 3: 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) - (5), and the messages in steps (6) - (10) will be sent as indicated.

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

The schedule is given below for the vertex labeled 0, 1, 4 and 8 in the tree network T Figure 2. The schedule for the vertex labeled 0 is straight forward. Message i is received at time i and it is sent at time i for $1 \leq i \leq 15$, and message 0 is sent at time 16. The schedule for the vertex labeled 1 is simple since it receives the b-messages and then the e-messages.

The schedule for vertex labeled 4 is more complex since it includes messages 2 and 3 that are delayed. In the vertex labeled 8 it is more complex since messages 2, 3, 6, and 7 are the ones delayed.

Procedure `ConcurrentUpDown` is just algorithms `Propagate-Up` and `Propagate-Down` executed concurrently.

Table 1. Schedule for the Vertex Labeled 1.

Time	0	1	2	3	4	5	6	7	8
Rec.	-	2	3	-	-	4	5	6	7
Snd.	1	2	3	-	-	4	5	6	7
Time	9	10	11	12	13	14	15	16	17
Rec.	8	9	10	11	12	13	14	15	0
Snd.	8	9	10	11	12	13	14	15	0

Table 2. Schedule for the Vertex Labeled 4.

Time	0	1	2	3	4	5	6	7	8
Rec.	-	5	1	2	3	6	7	8	9
Snd.	-	-	1	4	5	6	7	8	9
Time	9	10	11	12	13	14	15	16	17
Rec.	10	-	-	11	12	13	14	15	0
Snd.	10	2	3	11	12	13	14	15	0

Theorem 1: The communication schedules generated by procedure `ConcurrentUpDown` has total communication time $n + r$ for any n node network with radius r.

Proof: The idea is to establish that there are no conflicts at the vertices when messages are being transmitted. This is accomplished by identifying time periods where the different type of messages are being transmitted. For brevity the proof is omitted.

3 Discussion

We have presented an algorithm to construct communication schedules with total communication time at most $n + r$, where r is the radius of the graph. The algorithms are efficient and generate near optimal solutions. 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 to construct the schedule take $O(n)$ time.

With a little bit of preprocessing and storing additional information at each vertex, our algorithm can be made online provided there is a general synchronization process. For brevity we cannot elaborate on this extension. Our algorithm can be easily adapted to the weighted gossiping problem where each processor has at least one message to transmit.

References

[1] J. C. Bermond, L. Gargano, A. A. Rescigno, and U. Vaccaro. Fast gossiping by short messages. *SIAM Journal on Computing*, 27(4):917–941, 1998.

Table 3. Schedule for the Vertex Labeled 8.

Time	0	1	2	3	4	5	6	7	8	9
Rec.	-	9	-	1	4	5	6	7	10	-
Snd.	-	-	-	1	4	5	8	9	10	6
Time	10	11	12	13	14	15	16	17	18	19
Rec.	-	2	3	11	12	13	14	15	0	-
Snd.	7	2	3	11	12	13	14	15	0	-

[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. In *Proc. 1st ACM Symp. on Parallel Algorithms and Architectures*, pages 318–327, 1989.

[4] U. Feige, D. Peleg, R. P, and U. E. Randomized broadcast in networks. In *International Symposium SIGAL '90, Lecture Notes in Computer Science*, pages 128–137, Berlin, 1990.

[5] P. Fraigniaud and S. Vial. Approximation algorithms for broadcasting and gossiping. *Journal of Parallel and Distributed Computing*, 43:47–55, 1997.

[6] S. Fujita and M. Yamashita. Optimal group gossiping in hypercube under circuit switching model. *SIAM Journal on Computing*, 25(5):1045–1060, 1996.

[7] L. Gargano, A. A. Rescigno, and U. Vaccaro. Communication complexity of gossiping by packets. *Journal of Parallel and Distributed Computing*, 45:73–81, 1997.

[8] T. F. Gonzalez. Complexity and approximations for multimessage multicasting. *Journal of Parallel and Distributed Computing*, 55:215–235, 1998.

[9] T. F. Gonzalez. Improved approximation algorithms for multimessage multicasting. *Nordic Journal on Computing*, 5:196–213, 1998.

[10] T. F. Gonzalez. Gossiping with multicasting communication primitives. In *Proceedings of the 2000 Parallel and Distributed Computing Systems Conference, PDCS'2000*, volume II, pages 768–773. IASETED, 2000.

[11] T. F. Gonzalez. Simple algorithms for multimessage multicasting with forwarding. *Algorithmica*, to appear.

[12] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication network. *NETWORKS*, 18:129–134, 1988.

[13] J. Hromkovic, R. Klasing, B. Monien, and R. Peine. *Dissemination of Information in Interconnection Networks (Broadcasting and Gossiping)*, pages 273–282. In Combinatorial Network Thoey, D. Z. Du and D. F. Hsu (Eds.), Kluwer Academic, 1996.

[14] D. W. Krumme, K. N. Venkataraman, and G. Cybenko. Gossiping in minimal time. *SIAM Journal on Computing*, 21(2):111–139, 1992.

[15] R. Ravi. Rapid rumor ramification. In *Proc. 35th Annual Symp. on Foundations of Computer Science*, pages 202–213, 1994.

[16] H. Shen. Efficient multiple multicasting in hypercubes. *Journal of Systems Architecture*, 43(9), August 1997.