An Optimal Scheme for Disseminating Information

Yijie Han and Raphael Finkel

Computer Science Department
Patterson Office Tower
University of Kentucky
Lexington, KY 40506-0027

Abstract — We present an optimal communication pattern that achieves complete dissemination of information among n machines in $\lceil \log n \rceil$ time by repeated use of point-to-point messages. This scheme improves previous results in this field by removing restrictions on n (it need not be a power of 2) and by introducing reorganization to tolerate single-machine faults.

1 Introduction

Data broadcasting is a very important operation in parallel and distributed systems[1,2,3,4,5]. Alon *et al.*[1] have characterized the data-dissemination process as one in which each machine repeatedly receives, modifies, and forwards messages. All machines act in synchony; in each round, each machine sends and receives at most one message. Alon *et al.* have shown that broadcasting can be accomplished by data dissemination without physical broadcast.

The fundamental mechnism of Drezner[2] and Alon *et al.*[1] disseminates information by repeatedly doubling the number of machines that receive the information during each round. Information originally in machine i_1 can be sent in one round to machine i_2. During the second round both i_1 and i_2 become senders, and two other machines i_3 and i_4 receive the information. Information can therefore be broadcast to all machines in $\lceil \log n \rceil$ rounds. Broadcasting by data dissemination requires at least $\lceil \log n \rceil$ rounds. Simultaneous broadcasts can piggyback data onto the same messages.

The situation becomes a more complicated when the following conditions must be met.

(a) No matter which machine starts broadcasting, the information should reach all n machines in $\lceil \log n \rceil$ rounds.

(b) No matter at what round a machine starts broadcasting, the information should reach all n machines in $\lceil \log n \rceil$ rounds.

(c) The scheme should be able to detect and/or tolerate certain faults.

(d) The system should be able to dynamically reorganize after certain faults.

The scheme proposed by Alon *et al.*[1] meets conditions (a) and (b) when n is a power of 2 or one of a family of primes. For general n, they adopt a quotient emulation [6] that

requires $2\lceil \log n \rceil$ rounds. Their scheme can detect any single-machine failure in n rounds. (A failed machine no longer sends messages.) They propose analytic solutions for faulty systems as an area worthy of further research.

In this paper we present an optimal scheme for disseminating information. Our scheme meets conditions (a) and (b) for any n. In addition, our scheme tolerates single-machine failure if the number of rounds used for broadcasting information is $\lceil \log n \rceil + 2$. Our scheme allows the system to be reorganized in $\lceil \log n + 2 \rceil$ rounds to excise a failed machine.

The underlying model that we will use assumes that any permutation of messages can be realized in one round. We consider such a permutation of messages a basic communication event. The token ring network and the ethernet are particularly suitable as implementations for our model. However, a restricted network with $\lceil \log n \rceil$ connecting links for each machine is sufficient for our data-dissemination procedure; after each $\lceil \log n \rceil$ rounds, our scheme repeats the destination pattern.

The rest of the paper is organized as follows. Section 2 presents our scheme for data dissemination. Section 3 shows how our scheme can be modified to tolerate single faults. Section 4 discusses reorganizing the system after failure. We summarize these results and draw some conclusions in Section 5.

2 Broadcast

Our scheme for disseminating information is for each machine i, $0 \leq i < n$, to execute the following procedure.

Procedure Dissiminate
loop
  – each iteration is one run
  for round := 0 **to** $\lceil \log n \rceil - 1$ **do**
    – each iteration is one round
    message := new + old broadcast data
    send message to machine $(i + 2^{\text{round}}) \bmod n$
  end – for
end – loop

The information sent by machine i during any time t is a combination of

- New information that i has decided to broadcast starting at time t.
- Information received by i during the previous $\lceil \log n \rceil - 1$ rounds that must still be transmitted to other machines.

We will not be concerned with how messages are packaged and how i maintains its data structures to show which information needs to be sent on any round. We will assume that messages are long enough to hold all the information that must be sent. If a basic unit of

information uses b bytes, and each machine may only originate one broadcast per round, then the longest message needed will be $2bn$ bytes. However, many applications can reduce this length by combining information from several simultaneous broadcasts. To produce an average, for example, a broadcast message need only include the sum of the values provided by all the broadcast sources.

We will call a set of $\lceil \log n \rceil$ rounds a **run**. The following chart shows the destinations for the 3 rounds that constitute a run when $n = 6$.

round	source					
	0	1	2	3	4	5
0	1	2	3	4	5	0
1	2	3	4	5	0	1
2	4	5	0	1	2	3

If machine 2 wishes to disseminate some information at round 0, it will send it to machine 3. At round 1, machines $\{2, 3\}$ will send the information to $\{4, 5\}$. At round 2, machines $\{2, 3, 4, 5\}$ will send the information to $\{0, 1, 2, 3\}$, at which point all 6 machines will have the desired information. In fact, machines $\{2, 3\}$ will have seen it twice. If this is a problem (and it would be for computing global statistics such as averages), the messages can include a unique broadcast serial number (in this case, picked by machine 2) that can be checked for duplicates.

All machines must agree on the current round within a run. Either they can refer to a global clock, or they can agree that each machine will send exactly one message per round even if no broadcasts are current.

Theorem 1: Procedure *Disseminate* will broadcast information from any source to all destinations within any consecutive $\lceil \log n \rceil$ rounds.

Proof: Any broadcast started by machine i arrives at machine j in round s if and only if a broadcast started by machine 0 arrives at machine $(j - i) \bmod n$ in round s, so without loss of generality, we may assume the source is machine 0. After $\lceil \log n \rceil$ rounds, the broadcast has reached machines $(0[+2^1][+2^2]...[+2^{\lceil \log n \rceil - 1}]) \bmod n$, where square brackets indicate that the term is optional. The order in which these machines are reached depends on which round in a run is current when the broadcast starts, but the set of reached machines is the same in any case. Clearly, this set includes all machines $0 \leq i < n$. □

In procedure *Disseminate*, we can view all machines as arranged in a cycle, as shown in Figure 1. The (asymmetric) distance from machine p to machine q is $(q - p) \bmod n$. If broadcast starts at round 0, then after round i, machines within distance 2^i from p will receive information from p. The pattern is not so clear for broadcasts that start at different rounds. Alon *et al.*[1] realized that procedure Disseminate will finish broadcasting if it is started at round 0, whereas Theorem 1 proves a stronger fact.

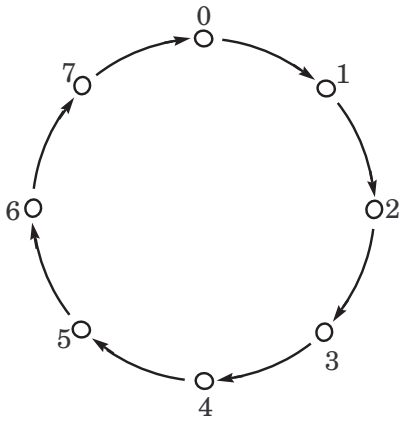

Figure 1: Conceptual arrangement of machines.

3 Fault tolerance

We assume that machine failures prevent messages from being sent from the failed machine, but that no other indication of failure is given. This failure model is intermediate between immediately globally visible failures, which are trivial to deal with, and Byzantine failures [7], which are quite difficult.

The disseminating process of *Disseminate* assumes a tree shape, as shown in Figure 2. We will call this tree the **broadcasting tree** of machine 0 with depth $\lceil \log n \rceil$ starting at round 0. A failed machine will cast a shadow in this tree, preventing its descendants from getting information originating from the root of the broadcasting tree. In order to transfer information from the root to all nodes, we have to use more than $\lceil \log n \rceil$ rounds. Extra rounds can circumvent the failed machine. In the following theorem we show that if exactly one machine fails, a message from any other machine can reach all working machines within any consecutive $\lceil \log n \rceil + 2$ rounds of *Disseminate*.

Theorem 2: Procedure *Disseminate* will broadcast information from any source to all destinations within any consecutive $\lceil \log n \rceil + 2$ rounds if exactly one machine has failed.

Proof: Without loss of generality, machine 0 is the source of the broadcast. Suppose machine 0 starts broadcast at round i, and the information first reaches the failed machine f at round j, where $0 \leq i, j < \lceil \log n \rceil$. Consider the following cases:

Case 1: $0 \leq i \leq j < \lceil \log n \rceil$. After rounds $i, i+1, ...j, ..., \lceil \log n \rceil - 1, 0, 1, ..., i-1$, machines that have not yet received information from machine 0 are in $S = \{f \leq m | m = f[+2^{j+1}]...[+2^{\lceil \log n \rceil - 1}][+2^0]...[+2^{i-1}]\}$. No pair of machines in S is separated by exactly 2^i. In the next round, round i, every machine in S will receive information from a source 2^i places earlier in the cycle, and that source is not in S. Therefore, that source already has the

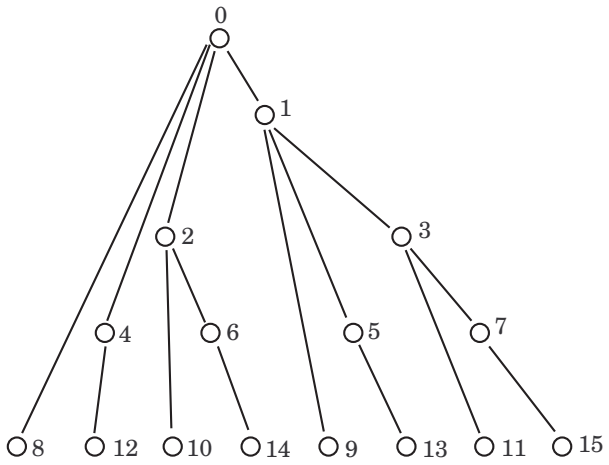

Figure 2: Broadcasting tree of machine 0 with 4 rounds starting at round 0.

information derived from machine 0. In this case, $\lceil \log n \rceil + 1$ rounds suffice for broadcasting information.

Case 2: $0 \leq j < i < \lceil \log n \rceil$. After rounds $i, i+1, \lceil \log n \rceil - 1, 0, 1, ..., j, ..., i-1$, the machines that have not yet received information from machine 0 are in $S = \{m | m = f[+2^{j+1}]...[+2^{i-1}]\}$. When $i < \lceil \log n \rceil - 1$, no two machines in S are separated by exactly 2^i, so broadcasting can be accomplished with one more round. When $i = \lceil \log n \rceil - 1$, it is possible that two machines in S are separated by exactly 2^i (see the example following this proof). In this situation, two more rounds will finish the broadcast. The reason is that when $i = \lceil \log n \rceil - 1$, the second broadcast round is round 0. A broadcast that starts at round 0 would require $\lceil \log n \rceil + 1$ rounds to complete in the face of a single failure, as shown in Case 1. □

Here is an example that demonstrates the necessity of $\lceil \log n \rceil + 2$ rounds. Let $n = 10$, $i = 3$ (the round at which broadcast starts), $f = 1$ (the failed machine). Broadcast is initiated by machine 0. The following chart shows that 6 rounds are required for broadcast to complete. A mark (•) indicates that the given machine has received the broadcast.

round	machine									
	0	1	2	3	4	5	6	7	8	9
3	•	f							•	
0	•	f							•	•
1	•	f	•						•	•
2	•	f	•	•	•		•		•	•
3	•	f	•	•	•		•	•	•	•
0	•	f	•	•	•	•	•	•	•	•

The analysis presented in this section shows that by using two more rounds for broadcasting our scheme can tolerate single faults.

4 Reorganization

After a machine f fails, it can be advantageous to reorganize and degrade an n-machine system to an $(n-1)$-machine system. Reorganization can be accomplished as follows. If $f = n-1$, the rest of the machines simply assume that there are only $n-1$ machines left. If $f \neq n-1$, machine $n-1$ will assume the identity of f. Depending on the communication medium used, destination tables might need to be modified on every machine. (We require that failures never partition the network.)

It can happen that $n = 2^j + 1$ for some j, in which case the reduction in machines leads to a new value for $\lceil \log n \rceil$. The current round number may be invalid for this new value. Therefore, we will reset the round to 0 after reorganization.

We must resolve two technical details to accomplish reorganization. The first is achieving simultaneous reorganization, and the second is protecting active broadcasts during reorganization.

Simultaneous reorganization

Reorganization must occur simultaneously on all machines or chaos will result. The following method achieves this aim. If a single machine f should fail, one machine, say i, will notice that fact during the next round as an expected message fails to arrive. A different machine, say j, will notice the failure during the subsequent round as its expected message fails to arrive. Meanwhile, i can report the failure to a third machine k during the same period. Although both i and j notice that f has failed, only i gets the correct failure time. The failure time of f noticed by j is later than that noticed by i. During later rounds, if machine m receives word of the failure from both i and j, m can discard the notice from j because the failure time reported by i is earlier than that from j. By Theorem 2, a failure notice from i will be received by every functioning machine in $\lceil \log n \rceil + 2$ rounds. Every such machine can reorganize its tables effective $\lceil \log n \rceil + 3$ rounds after the failure occured. This scheme is expressed by the following procedure.

Procedure Reorganize;
– reorganization during broadcast
var
i : integer := whatever;
– identifier of this machine
n : integer := whatever;
– number of machines
clock : integer := 0;
– monotone increasing every round
f : integer; – which machine failed

```
    failtime : integer := ∞;
      – when it failed: based on clock
    reorganize : integer := -1;
      – time reorganization scheduled
  loop – each iteration is one run
    for round := 0 to ⌈log n⌉ − 1 do
      if reorganize = clock then
        – time to reorganize
        if i = n − 1 then
          i := f
        endif;
        failtime := ∞;
        n := n − 1;
      endif;
      message := traffic, f, failtime;
      send message to machine (i + 2^round) mod n
      accept message from (i − 2^round) mod n
      if no message arrives then
        – a failure has occurred
        newfail := (i − 2^round) mod n
        newfailtime := clock - 1;
      else
        traffic, newfail, newfailtime := message;
      endif;
      if newfailtime < failtime then
        – discard old value
        failtime := newfailtime;
        f := newfail;
      endif;
      if failtime < ∞ then
        reorganize := failtime + ⌈log n⌉ + 3;
      endif;
      clock := clock + 1;
    end – for loop; one round
  end – loop; one run
```

Based upon the above analysis we have:

Theorem 3: The system can be reorganized in $\lceil \log n \rceil + 3$ rounds after a single failure.

Proof: If machine f fails at round i, then at round $i+1$, some machine notices the failure and the failure time. By Theorem 2, this information will be known to all machines in an additional $\lceil \log n \rceil + 2$ rounds. Therefore at the end of round $\lceil \log n \rceil + 3$, every machine is ready for reorganization. □

The first machine p that observes the failure starts a failure broadcast. The relation between this p and f enables us to avoid the $(\lceil \log n \rceil + 2)$-round case (Case 2 of the proof of Theorem 2). We can prove the following stronger result.

Theorem 4: The system can be reorganized in $\lceil \log n \rceil + 2$ rounds after a single failure.

Proof: The only situation where $\lceil \log n \rceil + 2$ rounds are required for broadcasting with a single failure occurs when a machine starts broadcast at round $\lceil \log n \rceil - 1$ (see the proof of Theorem 2). This happens when f fails at round $\lceil \log n \rceil - 3$, machine p discovers the failure at round $\lceil \log n \rceil - 2$, and starts broadcast at round $\lceil \log n \rceil - 1$.

In this case the distance from f to p is $2^{\lceil \log n \rceil - 2}$ and the distance from p to f is greater than $2^{\lceil \log n \rceil - 2}$ (recall that the distance is asymmetric). Ignore the first round in broadcast, that is, round $\lceil \log n \rceil - 1$. Machine f is a leaf in the broadcasting tree of machine p starting at round 0 and therefore will not cast shade on any machine in this tree. Thus in $\lceil \log n \rceil + 2$ rounds after failure every machine has the correct failure information. □

Active broadcast

The other detail to be worked out is the fate of broadcasts active during failure and reorganization. Reorganization is likely to introduce chaos into any broadcasts that are active. For this reason, machines should not start new broadcasts if they know that a reorganization is scheduled within the next $\lceil \log n \rceil + 2$ time units. (If a reorganization is scheduled, there must be a failed machine, in which case broadcast can take that long to finish.) Any broadcast that was inadvertently started within this window should be restarted.

A failure that occurs at time t could damage broadcasts started as early as time $t - \lceil \log n \rceil + 1$. Two strategies can cope with these damaged broadcasts. The first strategy restarts such broadcasts after the error has been repaired. Broadcasts may therefore require as long as $3\lceil \log n \rceil$ time to complete. The first $\lceil \log n \rceil$ is for the original broadcast that fails near the end because of a broken machine. The second $\lceil \log n \rceil$ is needed to inform the world about the failure, and the third $\lceil \log n \rceil$ is to run the broadcast again.

The second strategy uses $\lceil \log n \rceil + 2$ rounds for each broadcast to tolerate single faults so that broadcast need not be restarted when a machine fails. The first strategy is more efficient on the average if failures are rare, but can suffer long delays when failures occur. The second strategy uses $\lceil \log n \rceil + 2$ rounds for every broadcast, but it is a useful strategy for real-time systems.

5 Conclusions

We have presented an optimal data-dissemination algorithm for machines capable of sending messages to each other. It improves previous results in that it does not restrict the number of machines and accomodates single-machine failure and reorganization.

Although our data-dissemination algorithms are described for point-to-point networks, they are also suited to multi-machine organizations with broadcast media such at token-ring

network and ethernets [8]. In these networks, each message can potnetially reach all machines, although usually only the intended destination machine actually reads any message. Ordinary broadcast by n processors requires n messages, but each machine must receive $n-1$ messages, leading to $O(n^2)$ total work. In comparison, dissemination requires only $O(n \log n)$ total work. The reduction is due to piggybacking messages. In some applications, such as finding average load, piggybacking does not require increasing the message size.

Several open questions remain. Multiple failures could prevent broadcasts from reaching from some sources to some destinations altogether. We know that one failure never has this property. What is the minimum number of failures that does?

Our reorganization scheme is synchronous. Is there an asynchronous reorganization scheme that preserves broadcasts that are underway?

Acknowledgements

We would like to thank Debra Hensgen for helpful discussions.

References

1. N. Alon, A.Barak, and U. Manber, "On disseminating information reliably without broadcasting," *Proceedings of the 7th International Conference on Distributed Computing Systems*, pp. 74-81 (September 1987).

2. Z. Drezner and A. Barak, "A probabilistic algorithm for scattering information in a multiprocessor system," Technical report CRL-TR-15-84, University of Michigan Computing Research Laboratory (March 1984).

3. C.E. Leiserson and J.B. Saxe, "Optimal synchronous systems," *Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science*, pp. 23-36(1981).

4. M. Livny and U. Manber, "Distributed computation via active messages," *IEEE Trans. on Comput.* **C-34**(12) pp. 1185-1190 (December 1985).

5. D. Nassimi and S. Sahni, "Data broadcasting in SIMD computers," *IEEE Trans. Comput.* **C-27**(2) pp. 2-7 (1979).

6. J.A. Fishburn and R.A. Finkel, "Quotient networks," *IEEE Trans. on Comput.* **C-31**(4) pp. 288-295(April 1982).

7. R. Cristian, H. Aghili, and R. Strong, "Atomic broadcast from simple message diffusion to byzantine agreement," Technical report RJ 4540, IBM San Jose Research Center (December 1984).

8. A.S. Tanenbaum, *Computer networks*, Prentice-Hall (1981).