Reliable Broadcasts Through Partial Broadcasts*

Özalp Babaoğlu
Pat Stephenson†

TR 85-720
December 1985

Department of Computer Science
Cornell University
Ithaca, New York 14853

*Partial support for this work was provided by the National Science Foundation under Grant MCS 82-10356.

†Supported by the Defense Advanced Research Projects Agency (DoD) under ARPA order 5378, Contract MDA903-85-C-0124. The views, opinions and findings contained in this report are solely those of the authors and should not be construed as an official Department of Defense position, policy, or decision.

Reliable Broadcasts Through Partial Broadcasts*

Özalp Babaoğlu
Pat Stephenson †

Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.

ABSTRACT

In the *Reliable Broadcast Problem*, a processor disseminates a value to all other processors in a distributed system where both processors and communication components are subject to failures. We prove lower bounds for the execution time of any reliable broadcast protocol in distributed systems with arbitrary communication networks. Our results apply to common distributed system architectures consisting of multiple broadcast network-based clusters of processors. In light of these lower bounds, our earlier protocols are shown to be optimal with respect to execution time.

November 29, 1985

---

* Partial support for this work was provided by the National Science Foundation under Grant MCS 82-10356.

† Supported by the Defense Advanced Research Projects Agency (DoD) under ARPA order 5378, Contract MDA903-85-C-0124. The views, opinions and findings contained in this report are solely those of the authors and should not be construed as an official Department of Defense position, policy, or decision.

1. Introduction

In a distributed computing system, a protocol implements Reliable Broadcast (RB) if it guarantees the following two conditions:

RBA: *(Agreement)* In response to a broadcast, all correct processors accept the same message.

RBV: *(Validity)* If the broadcasting processor, called the *sender*, is correct, then all correct processors accept the message that was broadcast.

In [BDa], we described RB protocols that exploited communication architectures other than point-to-point networks. In this paper, we prove lower bounds for the execution time of any RB protocol in systems with arbitrary communication networks and show that the protocols in [BDa] are optimal.

2. Broadcast Networks

A common architecture for a distributed system consists of several processor clusters on a common network where the intra-cluster communication takes place over shared, multiple-access media that support broadcasts [Sta]. Figure 1 illustrates such an architecture with three clusters of n_1, n_2 and n_3 processors. This broadcast network-based architecture encompasses a wide range of designs. For example, the clusters in Figure 1 could represent geographically distant local area networks connected through gateways (such as the Xerox Internet comprising a large number of Ethernets [MB]). Alternatively, each cluster could be a single, tightly-coupled multiprocessor with an

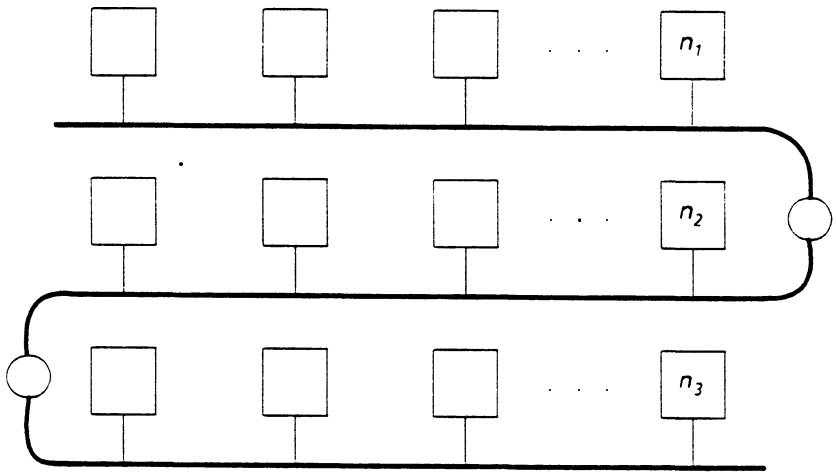

Fig. 1. *A broadcast network-based distributed system with three clusters*

internal bus interconnect and inter-cluster links implemented as bus adaptors.

Regardless of the physical realization of the architecture, we can abstract the behavior of communication in such systems as follows:

BNP: *(Broadcast Network Property)* In response to a **broadcast**, all processors that receive a message receive the same message.

BNP ensures that for all possible failures, a processor cannot send conflicting messages to other processors in a single broadcast. For a given broadcast network, the set of processors that receive the (same) message in response to a broadcast is called the *receiving set.* The *broadcast degree* of a network is defined to be a lower bound on the size of the receiving sets for all broadcasts in the network. The receiving sets may vary from one broadcast message to another as well as from one sender to another. For a net-

work to have broadcast degree b, all we require is that each of these sets contain at least b processors. We assume that every processor receives its own broadcasts regardless of failures. Note that communication failures manifest themselves in defining a particular broadcast degree for the network. For the example depicted in Figure 1, if we assume that communication failures can occur only in the gateways (or whatever physical component the two circles are modeling), it is easy to see that the broadcast degree of the network is $b = \min(n_1, n_2, n_3)$.

3. System Assumptions

Informally, a processor is said to be *correct* if its behavior conforms to an abstract specification (usually in the form of an algorithm). Each time that the behavior of a processor differs from its specification, a *failure* is said to occur; the processor is called *faulty* after the first failure. We make these definitions precise in the next section. In this paper, we consider systems subject to *omission* failures [Had], where a faulty processor may fail to send some of the messages prescribed by its protocol, but messages it does send are always correct. Failures in the communication components cause messages to be lost; all messages that are delivered are delivered unchanged. Simple network protocols that approximate this behavior of the communication media are well known [Tan].

We make the following additional assumptions about the system:

NA: *(Network Assumption)* Despite processor and communication component failures, the network remains connected and guarantees a bounded delivery time for all messages.

CA: *(Clock Assumption)* Correct processors have local clocks that differ by at most ε units and have bounded drift with respect to (unobservable) absolute time.

Protocols that achieve CA in the presence of faulty components are known [HSS, LMS, ST]. For clarity, we present our protocols as synchronous distributed computations executing in lockstep units called *rounds* (i.e., assuming $\varepsilon = 0$). Informally, a round is the minimum length of time required for each processor to send messages to all other processors, receive any messages destined for it and perform a given amount of local computing. Note that it is easy to transform such a round-based protocol into an equivalent one that works in a system where $\varepsilon > 0$ as shown in [BDa].

In general, satisfying the NA requires replication of the network components. A *redundant broadcast network* interconnects processors through a sufficient number of broadcast networks, each exhibiting the Broadcast Network Property, such that NA is satisfied. The *broadcast degree* of the entire redundant network is defined to be the minimum of the broadcast degrees of its constituents. We assume that the **broadcast** primitive, when invoked by a correct processor, functions by broadcasting the message over all of the

network replicas which the processor is connected to. However, no assumptions are made regarding the order in which networks are selected or the possibility of omitting messages from certain networks by faulty processors. Multiple rings, Ethernets or buses can be used to construct redundant broadcast networks. In the rest of the paper, we will consider only redundant broadcast networks and refer to them simply as broadcast networks. We will also make the assumption that a **broadcast** performed by a correct processor is received by all other processors in the same round. We will relax this assumption in Section 6.

In a distributed system with n processors, the case $b=n$ corresponds to a broadcast network where no communication failures occur and RB can be trivially solved by a single **broadcast.** On the other hand, a system where $b=2$ is logically equivalent to a point-to-point communication network and all of the result for RB in such an environment apply [FL, DS, Had]. The case where $t+1 \leq b < n$ was covered in [BDb] where we presented two-round protocols to implement RB under a variety of failure models and proved their optimality. In [BDa] we exhibited a family of RB protocols for networks with broadcast degrees in the range $2 < b < t+1$. In this paper, we prove lower bounds for the execution time of RB protocols in such networks that are said to support a *partial broadcast* communication facility.

4. Model of Execution

The following model, similar to that used in [DS], is intended to capture the execution of any deterministic synchronous protocol in an environment that supports partial broadcasts. The execution of an m-round protocol is modeled as a directed graph, called a *history*. A history H consists of a number of *levels*. Each level corresponds to one round in the execution of the protocol. For convenience, we introduce an additional level numbered 0 to indicate the sender's value. In the subsequent discussion, level and round will be used interchangeably. The total number of levels in the history is called the *depth*. In all the histories we consider, the level will be $m+1$. Each round consists of n *nodes*, one per processor in the system. The node representing processor p in round i is denoted p_i. The sending processor will be denoted by s.

The *subhistory of history H* according to any processor p, denoted as pH, is the subgraph obtained from H by deleting all edges except those that are directed towards p. To model a reliable broadcast protocol in this system, we define two functions on subhistories according to each processor p. To represent a processor's action at round i in the protocol we define the (deterministic) *action function* A that maps the subhistory pH up to round $i-1$ into the labeled outedges of p_i. Since we are dealing with a broadcast model, all possible action functions will either generate an edge to every other node, all labeled identically, or no edges. The label of an outedge from

node p_i may be from the message domain only if there is an inedge to one of $p_0 \cdots p_{i-1}$ with this label. We say that node p_i is *correct* if it has exactly the outedges given by A for round i. A processor p is *correct* if p_i is correct in all rounds i, where $1 \leq i \leq m$. A processor is *correct after round* k if all of $p_{k+1} \cdots p_m$ are correct. A processor has *failed by round* k if any of $p_1 \cdots p_k$ are not correct. A processor *fails in round* k if $p_0 \cdots p_{k-1}$ are correct but p_k is not.

Consider a history corresponding to the execution of an m-round RB protocol where there may be at most t faulty processors. Communication between processors through a network with broadcast degree b imposes the following structure on this history:

(1) In round 0, s_0 has an inedge labeled with the value v to be broadcast. No other processor has inedges in round 0.

(2) In round i, $1 \leq i \leq m$, each node p_i either has:

(2a) Outedges to all nodes in round i (node p_i is correct).

(2b) Outedges to k nodes, $b \leq k < n$, in round i, including node p_i (processor p has failed by round i).

(2c) No outedges (either node p_i is correct or processor p has failed by round i).

(3) No more than t processors may have associated nodes that are not correct during the entire history.

(4) Since we consider only omission failures, all outedges from a particular node must have the same label. An outedge may only be labeled with a value from the message domain if the processor has an inedge in some previous round labeled with this value. Otherwise we do not restrict labels. This allows us to admit protocols that may exchange control information.

Each processor also has a *decision function* F that maps subhistories pH to the domain of possible message values or δ. The decision function must return δ if p has no inedges labeled with a value from the message domain in any round.

In this model, the conditions for reliable broadcast may be restated as:

RBA: *(Agreement)* In all histories H, for any two correct processors p and p', $FpH = Fp'H$.

RBV: *(Validity)* In all histories H, if s_0 is correct and broadcasts the value v then $FpH = v$ for all correct processors p.

Two histories H and H' are said to be *equivalent* if $FpH = FqH'$ for any p correct in H and any q correct in H'. Every history must be equivalent to itself. Note that this equivalence definition requires every history to satisfy the agreement condition for the reliable broadcast protocol.

5. Lower Bound Proof for RB Protocols with Partial Broadcasts

In this section we give a lower bound for the number of rounds required for the RB problem in networks with partial broadcasts. Our proofs are generalizations of those found in [DS, Had] that establish similar results for other communication and failure models.

Intuitively, to establish a lower bound, we should consider histories where the value transmitted by the sender is kept from correct processors for as long as possible. We now formalize the class of such histories. We consider all histories of depth $t-b+3$, satisfying RBA and RBV with which we can associate a sequence of sets of processors $S(1) \cdots S(t-b+2)$, called the *possible failure sequence*, with the following properties:

(1)

$$|S(r)| = \begin{cases} 1 & \text{if } r = 1 \\ b+r-2 & \text{if } r > 1 \end{cases}$$

(2) If processor p has failed by round r, then $p \in S(r)$.

(3) $S(r) \subset S(r+1)$, for all r such that $1 \leq r < t-b+2$.

We call all such histories the class C. Note that we are not restricting the behavior of any reliable broadcast protocol; only the class of failures we are considering. If a particular history H is a member of C, it may have many associated possible failure sequences. Finally, note that an execution in which no processors fail is a member of C, in which case we may associate any sequence S satisfying conditions (1) and (3) with this history. Recall

that in a correct execution all processors must accept the value v broadcast by the sender s. Therefore, for some history $H \in C$, $FpH = v$ for all correct p.

We will now outline our proof. We have defined an equivalence relation on histories in C, and we have noted that the history where no processors fail is a member of C. We will show below that a large class of histories in C are equivalent. This class will include a history with no failures, and also a history where the sender is faulty and no correct processor ever receives the value it broadcasts. However these histories cannot be equivalent, since they should accept different values. Thus we will conclude that the class C does not exist.

A processor p is *hidden* in round r in a history $H \in C$ as follows:

(1) In round $t-b+2$ all faulty processors that only have outedges to processors in $S(t-b+2)$ are hidden.

(2) A processor is hidden in round r, where $0 \leq r < t-b+2$ iff all outedges from the processor in round r are directed to processors in $S(r+1)$ that are hidden in round $r+1$.

Intuitively, a processor is hidden in a round if the information it gained in the previous round may not disseminate to all other processors before the termination of the protocol.

The following lemma shows that in any history in C, we may modify inedges of hidden processors without affecting the result of a reliable

broadcast.

Lemma 1. For any history $H \in C$, and any set of processors P in H such that all $p \in P$ are hidden in round r, there exists a history H', equivalent to H, with any single inedge e_p to each $p \in P$ added, removed or with a different label. However an edge may only be removed if its source has at least b remaining outedges. H' is identical to H in rounds 1 through r, apart from the edges e_p.

Proof: We construct a downward induction on r, the number of rounds. For the base case, we may add, remove or change the label of any single edge to each of the hidden processors in the last round without changing the subhistory seen by any of the correct processors. Thus this is the required equivalent history H'. Now assume we can do this for all rounds $r+1$ through $t-b+2$. We modify the edges one at a time. If we alter, add or remove an inedge in round r to a hidden processor, we replace its old outedges either with the same edges with a new label or with no edges at all. However in either case we are affecting inedges of hidden processors, so we apply the induction hypothesis. Thus the final history H' will be equivalent to H. □

The following lemma formally shows the effect of a hidden sender.

Lemma 2: For any history $H \in C'$ satisfying RBA and RBV in which the sender is hidden in round 1, $FpH = \delta$ for all correct p.

Proof: Since the sender is hidden, only processors in $S(r)$ may have inedges labeled with values from the message domain. Any processor not in $S(r)$ is correct, and thus must have $FpH = \delta$. By RBA, all correct processors have $FpH = \delta$. □

The following lemma is the core of the proof. It shows how we may transform any history in C to correct or hide any processors, and preserve equivalence.

Lemma 3: For every history $H \in C$, for any processor p at any round r:

(1) There exists a history $H' \in C$, equivalent to H, such that:

(1a) H' is identical to H in rounds 0 through $r-1$.

(1b) H' is identical to H in round r except possibly for edges from p_r.

(1c) The processor p is correct in H'.

(1d) All processors are correct after round r in H'.

(2) There exists a history $H' \in C$, equivalent to H, such that for any legal assignment to $S(r)$:

(2a) H' is identical to H in rounds 0 through $r-1$.

(2b) H' is identical to H in round r except possibly for edges from the processors in $S(r)$.

(2c) All processors in $S(r)$ are hidden at round r in H'.

Proof: We prove the lemma by downward induction on r, the number of rounds.

Base Case: Let $r = t-b+2$. For part (1), consider a faulty processor in the last round, missing some edges specified by the action function A. If we add these edges, one at a time, at each step there will be correct processors that see the same subhistory afterwards as before. Therefore the final history H' will be equivalent to H. For part (2), consider any legal assignment to $S(t-b+2)$. We wish to hide all processors in this set. We consider each processor, one at a time, and remove outedges, one at a time. At each point, there will be correct processors which see the same subhistory afterwards as before. Therefore the final history H' is equivalent to H.

Induction case: Given the hypothesis for all rounds greater than k, we show how to construct the required H' for conditions 1 and 2 as follows:

(1) Here we are trying to correct an arbitrary processor p in round r.

(a) Correct all processors in rounds $r+1$ through $t-b+2$. We can do this by (1) of the induction hypothesis.

(b) There are now a maximum of $S(r)$ incorrect processors in the history, since any processor that became incorrect after round r is corrected. Since $|S(r)| < |S(r+1)|$, there is at least one correct processor q in $S(r+1)$.

(c) For an incorrect processor p_r proceed as follows:

While there is an outedge of p_r missing:

(c.1) Select a missing outedge e (with respect to the function A) of p_r directed to p'_r.

(c.2) Let $S'(r+1) = (S(r+1)-\{q\}) \cup \{p'\}$.

(c.3) Hide p' at round $r+1$, by (2) of the induction hypothesis.

(c.4) Insert the edge e into the graph for round r, by Lemma 1.

(c.5) Correct all processors at round $r+1$, by (1) of the induction hypothesis. The resulting history is a member of C, since we can use $S'(r+1)$ instead of $S(r+1)$. Note that by a similar argument to (b) above, there is a correct processor q still in $S'(r+1)$. Thus we can continue this process.

(d) Processor p is correct in the resulting history since we have added all missing edges, and all processors are correct after round r. It thus satisfies part (1) of the induction hypothesis.

(2) Here we want to hide all the processors in $S(r)$ at round r.

(a) Correct all processors at round $r+1$, by part (1) of the induction hypothesis.

(b) Hide all processors in $S(r+1)$ at round $r+1$, by part (2) of the induction hypothesis.

(c) Add any missing edges from every $p \in S(r)$ to any $p' \in S(r+1)$, by Lemma 1.

(d) Correct all processors at round $r+1$, by part (1) of the induction hypothesis.

(e) Now all processors in $S(r)$ have at least b edges to processors in $S(r+1)$ and all processors are correct after round r. The following transformations remove all edges from all $p \in S(r)$ to any $p' \notin S(r+1)$. We generate a series of equivalent histories $H_0 H_1 \cdots H_j$ with associated failure sequences $S_0 S_1 \cdots S_j$, where $H_0 = H$ and $S_0 = S$. We note that since we do not change the failure behavior of any history before round r, and only of processors in $S_0(r)$, we have $S_0(r) = S_1(r) = \cdots = S_j(r)$.

(f) For each $p \in S_0(r)$

For each edge e from p directed to $p' \notin S_0(r+1)$

(f.1) Since all processors are correct after round r, there is some $q_i \in S_i(r+1)$ that is correct.

(f.2) Let $S_{i+1}(r+1) = (S_i(r+1) - \{q_i\}) \cup \{p'\}$.

(f.3) Hide $S_i(r+1)$ at round $r+1$, by the induction hypothesis. Now p' is hidden.

(f.4) Since e is now directed at a hidden processor, we may remove it by Lemma 1.

(f.5) Correct all processors in $S_{i+1}(r+1)$ by the induction hypothesis. We now have H_{i+1}.

(g) Let $S_j'(r+1) = S_j(r) \cup \{q_0\}$. This is clearly a possible value for H_j's failure sequence since all processors are correct in round $r+1$.

(h) Hide all processors in $S_j'(r+1)$ at round $r+1$, by the induction hypothesis.

The resulting history satisfies part (2) of the induction hypothesis. All processors in $S(r)$ now only communicate with processors in $S'(r+1)$ that are hidden. Thus all processors in $S(r)$ are hidden. □

Theorem 1. In a network with broadcast degree b, there exists no protocol that achieves reliable broadcast in fewer than $t-b+3$ rounds, where t is the maximum number of faulty processors.

Proof. Given any history $H \in C$, we may apply Lemma 3 repeatedly to transform it to an equivalent history in which the sender is hidden. In this history, by Lemma 2, the value accepted by the correct processors must be δ. However, the history in which there are no failures is a member of C. In this history, the value accepted by the correct processors must be v, the value broadcast by the sender. Thus we cannot achieve the agreement requirement in C, where processors are only permitted $t-b+2$ rounds of communication. □

6. Coping with Networks of Arbitrary Diameter

Recall that within a round, each processor can communicate with all other processors. We define a *phase* to be the interval of time necessary for all processors to communicate only with their immediate neighbors in the current system connectivity graph. Note that during the execution of the protocol, the original system graph will dynamically evolve as processor and communication failures occur. Consequently, the set of immediate neighbors of a processor can change from phase to phase.

In this section we consider systems where a single phase is no longer sufficient to implement a round. In such systems, let the *survivor diameter*, denoted as d, be the maximum number of phases required to achieve a round under all permissible failure scenarios [BDF]. Our earlier assumption that a **broadcast** performed by a correct processor is received by all other processors in the same round is equivalent to requiring that the survivor diameter of the system graph be one. We can now relax this assumption and consider systems with arbitrary survivor diameters.

Theorem 2. In a network with broadcast degree b and survivor diameter d, there exists no protocol that achieves reliable broadcast in fewer than $t-b+d+2$ phases, where t is the maximum number of faulty processors.

Proof Outline: Consider an execution of a RB protocol such that all of the processors in some set Π that includes the sender remain immediate neighbors of each other during the first $t-b+3$ phases. In other words, the subsystem consisting of processors in Π is a system with broadcast degree b and survivor diameter one to which Theorem 1 applies.

Let Π be such that it includes all of the t faulty processors and at least one correct processor p. Let q be a correct processor not in this subsystem such that it is distance $d-1$ from p and otherwise not connected to any other correct processor in Π. Thus, the system $\Pi \cup \{q\}$ has survivor diameter d and broadcast degree b, consistent with the hypothesis of the theorem. Consider a history where processor p receives the broadcast value for the first time in the last phase of a protocol that implements RB for the subsystem Π. Since q is distance $d-1$ away from p, at least these many additional phases are required for q to receive its first message. By Theorem 1, $t-b+3$ phases is a lower bound for the execution of any protocol to implement RB in the subsystem Π. Therefore, $t-b+d+2$ phases represents the minimum number of phases for a protocol to implement RB in the original system. □

7. Conclusions

We have studied the reliable broadcast problem in distributed systems where the communication network is characterized by its broadcast

degree and survivor diameter. Conventional network topologies such as fully connected point-to-point graphs, simple linear chains and rings correspond to special cases of this characterization. The class of networks that are generated for arbitrary b and d correspond to common architectures consisting of a collection of (possibly non-homogeneous) broadcast networks that are interconnected through gateways where any of the network components may fail.

We proved lower bounds for the execution time of any reliable broadcast protocol in such systems. The minimum number of rounds necessary to implement RB in these systems increases linearly as the broadcast degree decreases or as the system graph survivor diameter increases. The protocols we exhibited in [BDa] are optimal with respect to execution time in that they coincide with the lower bounds.

References

[BDa] Babaoğlu, Ö. and Drummond, R. Time-communication tradeoffs for reliable broadcast protocols. Tech. Report No. TR 85-687, Cornell University, Ithaca, New York, June 1985.

[BDb] Babaoğlu, Ö. and Drummond. R. Streets of Byzantium: Network architectures for fast reliable broadcast. *IEEE Trans. Software Eng.*, vol. SE-11 (June 1985), 546-554.

[BDF] Broder, A., Dolev, D., Fischer, M., Simons, B. Efficient fault tolerant routings in networks. In *Proceedings of 16th ACM Symposium on Theory of Computing* (1984), 536-541.

[DS] Dolev, D. and Strong, H.R. Authenticated algorithms for Byzantine agreement, *SIAM J. on Comput. 12*, 4 (November 1983), 656-666.

[FL] Fischer, M. and Lynch, N. A lower bound for the time to assure interactive consistency. *Inform. Proc. Letters 14*, 4 (April 1982), 183-186.

[Had] Hadzilacos, V. Issues of fault tolerance in concurrent computations. Ph.D Thesis, Tech. Rep. TR-11-84, Aiken Computation Laboratory, Harvard University, Cambridge, Mass., June, 1984.

[HSS] Halpern, J.Y., Simons, B., Strong, H.R. and Dolev, D. Fault-tolerant clock synchronization. In *Proceedings of 3rd ACM Symposium on Principles of Distributed Computing*, Vancouver, B.C., Canada (August 1984), 89-102.

[LMS] Lamport, L. and Melliar-Smith, P.M. Byzantine clock synchronization. In *Proceedings of 3rd ACM Symposium on Principles of Distributed Computing*, Vancouver, B.C., Canada (August 1984), 68-74.

[MB] Metcalfe, R. and Boggs, D.R. Ethernet: Distributed packet switching for local computer networks. *Commun. ACM 19*, 7 (July 1976), 396-403.

[ST] Srikanth, T.K. and Toueg, S. Optimal Clock Synchronization, *Proc. 4th Symposium on the Principles of Distributed Computing*, Minaki, Canada (August 1985).

[Sta] Stallings, W. Local networks, *ACM Computing Surveys*, vol. 16, no. 1, (March 1984), 3-41.

[Tan] Tanenbaum, A. *Computer Networks*. Prentice Hall, Englewood Cliffs, N.J., 1981.