A Reliable Broadcast Protocol

ADRIAN SEGALL, SENIOR MEMBER, IEEE, AND BARUCH AWERBUCH, STUDENT MEMBER, IEEE

Abstract—**Broadcast in a communication network is the delivery of copies of messages to all nodes. A broadcast protocol is reliable if all messages reach all nodes in finite time, in the correct order and with no duplicates. The present paper presents an efficient reliable broadcast protocol.**

I. INTRODUCTION

BROADCAST multipoint communication is the delivery of copies of messages to all nodes in a communication network. In a network with mobile subscribers, for example, the location and connectivity to the network of such subscribers may change frequently and this information must be broadcast to all nodes in the network, so that the corresponding directory list entry can be updated. Broadcast messages are used in many other situations, like locating subscribers or services whose current location is unknown (possibly because of security reasons), updating distributed databases, or transmitting information and commands to all users connected to the communication network.

There are certain basic properties that a good broadcast algorithm must have and the most important are a) reliability, b) low communication cost, c) low delay, and d) low memory requirements. *Reliability* means that every message must indeed reach each node, duplicates should be recognizable upon arrival at a node and only one copy accepted, and messages should arrive in the same order as transmitted. *Communication cost* is the amount of communication necessary to achieve the broadcast and consists of, first, the number of messages carried by the network per broadcast message (*broadcast communication cost*), second, the number of control messages necessary to establish the broadcast paths (*control communication cost*), and third, the overhead carried by each message (*overhead cost*). Low *delay* and a small *buffer memory* are basic requirements for any communication algorithm, and broadcasts are no exception.

The definition of reliability indicated above requires some discussion, because in some applications not all the requirements are necessary. For example, broadcast of topological information in the new ARPANET routing algorithm [4] does not require order preservation and does allow duplicates. On the other hand, these properties are important when the information to be broadcast may be packetized and needs reassembly at the receiving nodes, as well as in applications where the broadcast consists of series of commands whose order and nonduplication is important. In the present paper we achieve reliability in the sense defined above.

The broadcast communication cost is minimized if the algorithm uses spanning trees, but normally there is need for a large control communication cost in order to establish and maintain these trees. However, the control cost can be reduced considerably provided that the routing mechanism in the network constructs routing paths that form directed trees towards each destination, in which case these trees can be used in the reverse direction for broadcast purposes. This general idea is presented in [1], but the authors show that the proposed algorithms, named *reverse path forwarding* and *extended reverse path forwarding*, are not reliable when the routing algorithm is dynamic, since in this case nodes may never receive certain messages, duplicates may be received and accepted at nodes, and the order of arriving messages may not be preserved. As said before, in order to be efficient, the abovementioned algorithm requires that the routing paths to each destination are directed trees. An adaptive routing algorithm that maintains at all times spanning directed trees rooted at the destination has been proposed in [2] and throughout the present paper we assume that the protocol of [2] is the underlying routing algorithm in the network. However, for the reasons stated before, namely the fact that the routing paths are dynamic, the broadcast algorithm of [1] is unreliable even if applied to the routing procedures of [2].

The purpose of the present paper is to propose and validate an algorithm whose main property is that the broadcast propagating on the tree provided by the routing protocol of [2] is reliable. It is convenient for the purpose of our discussion to separate the property of reliability into two parts: *completeness* means that each node accepts broadcast messages in the order released by their origin node, without duplicates or messages missing, while *finiteness* is the property that each broadcast message is indeed accepted at each node in finite time after its release. As shown by the authors, the algorithms of [1] are neither complete nor finite. In the algorithm of the present paper, completeness is achieved by requiring nodes to store broadcast messages in the memory for a given period of time and by introducing counter numbers at the nodes. Finiteness is obtained by attaching a certain impeding mechanism to the routing protocol. We may mention

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication after presentation at INFOCOM '83, San Diego, CA, April 1983. Manuscript received November 5, 1981; revised February 12, 1983. This work is based in part on an M.Sc. thesis at the Technion, Haifa, Israel, by B. Awerbuch, and supervised by A. Segall. The work of A. Segall was performed as part of a consulting agreement with the Laboratory for Information and Decision Systems at M.I.T., Cambridge, MA, and was supported in part by the Office of Naval Research under Contract ONR/N00014-77-C-0532 and in part by the Defense Advanced Research Project Agency under Contract N00014-75-C-1183.

A. Segall is with the Department of Electrical Engineering, Technion–Israel Institute of Technology, Haifa, Israel, on leave at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.

B. Awerbuch is with the Department of Electrical Engineering, Technion–Israel Institute of Technology, Haifa, Israel.

here that a broadcast algorithm can be easily made reliable if one allows infinite memory, unbounded counter numbers, and infinite overhead in the broadcast messages. The properties that make our algorithm tractable are bounded memory, bounded counter numbers, no overhead carried by broadcast messages (in form of counter numbers or any other kind) and the fact that the impeding mechanism is not activated most of the time.

In the rest of the paper we proceed as follows: Section II-A contains a brief description of the routing protocol of [2]; Sections II-B and II-C build the reliable broadcast protocol step by step, while its final form and main properties are given in Section III. The proofs of the main theorems are included in the Appendix.

II. THE BROADCAST PROTOCOL

A. *The Routing Protocol*

The underlying routing protocol considered in this paper is *the basic protocol* of [2]. In summary, this protocol proceeds in updating cycles triggered and terminating at the destination node named SINK. An updating cycle consists of two phases: a) control messages propagate uptree from SINK to the leaves of the current tree and each node i performs this phase whenever it receives a control message MSG from its current preferred neighbor p_i; b) control messages propagate downtree, while new preferred neighbors are selected and this phase is performed at node i upon detecting receipt of MSG from all neighbors. Our basic assumption is that all messages sent on a link arrive in arbitrary but finite time after their transmission, with no errors and in the correct order (FIFO). Observe that this does not preclude channel errors, provided there is an acknowledgment and retransmission protocol on the link. Under these conditions, the routing protocol maintains at all times a directed spanning tree rooted at SINK.

Before specifying the routing protocol we indicate several notations used in all algorithms in this paper. Subscript i indicates variables at node i and corresponding variables without subscript indicate variables in the received message. Whenever a message arrives from a certain neighbor it is first stamped with the identity of that neighbor, so that a control message, e.g., MSG($\cdot$), received at i from neighbor l, will be seen by the processor at i as MSG(l, $\cdot$). All variables and control messages of the algorithm are indexed by SINK and the protocol is performed independently for every possible source of broadcast messages in the network; in order to simplify notation, we suppress the index SINK. We also write in short *For MSG* (l, $\cdot$) instead of "Whenever MSG($\cdot$) is received from neighbor l, perform" and denote by G_i the set of neighbors of i.

We next indicate the algorithm at each node that implements the routing protocol of [2]. In the following sections we shall need to identify the updating cycles and it is convenient to attach already at this point a counter number α to each cycle. The cycle number will be incremented by the SINK when it starts a new cycle and will be carried by the control messages MSG belonging to the given cycle. For the time being the counter number will be unbounded, but later we shall show that a binary variable is sufficient.

Variables at Node i:

- α_i — counter number of the current updating cycle as known by i (values 0, 1, 2, $\cdots$) (not used in this and the next algorithms, but introduced here for later convenience)
- p_i — current preferred neighbor at node i
- $N_i(l)$ — 1 if MSG corresponding to current cycle has been already received from neighbor l, = 0 otherwise; $\forall l \in G_i$; (initialized to 0).

Routing Algorithm for Node i (RA):

1) For MSG(l, α)
2) $\quad N_i(l) \leftarrow 1$
3) $\quad$ *if* $l = p_i$ *then*: $\alpha_i \leftarrow \alpha$; send MSG ($\alpha_i$) to all $l \in G_i$, except p_i
4) $\quad$ *if* $N_i(l') = 1$ holds $\forall l' \in G_i$ *then*: send MSG(α_i) to p_i; select new p_i; set $N_i(l') = 0$ $\forall l' \in G_i$.

We have deliberately suppressed from the algorithm of [2] all variables that are not directly relevant to the broadcast and have not explicitly indicated the procedure for selecting the new p_i because it is not important for our purpose, except for the property that it maintains at all times a directed spanning tree rooted at SINK. For simplicity, p_i will be called the *father* of i. The algorithm is indicated for a given SINK that is not specified explicitly and that becomes the source of the broadcast messages. The SINK performs the following algorithm (lines are numbered to match equivalent instructions to the routing algorithm):

3) Start new cycle by $\alpha_{SINK} \leftarrow \alpha_{SINK} + 1$, send MSG($\alpha_{SINK}$) to all $l \in G_{SINK}$. (*Note:*[1] <3> can be performed only after <4> of the previous cycle has been performed.)
1) For MSG(α)
2) $\quad N_{SINK}(l) \leftarrow 1$
4) $\quad$ *if* $N_{SINK}(l') = 1$ holds $\forall l' \in G_{SINK}$ *then:* cycle α completed; set $N_{SINK}(l') = 0$ $\forall l' \in G_{SINK}$.

In principle, the routing tree can be used for broadcast purposes as follows: a node i accepts only broadcast messages received from its father p_i and forwards them to all nodes k whose father is i. Observe that we distinguish between receiving a broadcast message and accepting it. In general, a broadcast message received at a node may be either accepted or rejected, depending on the specific algorithm.

The first problem that one encounters with the above procedure is that in the routing algorithm a node i knows only its father p_i, but does not know the nodes k for which $p_k = i$. Consequently, we need an addition to the routing algorithm, so that whenever a node i changes its father p_i (line <4> in the routing algorithm), it sends two special messages: DCL (declare) to the new father and CNCL (cancel) to the old father. Each node i will have a binary variable $z_i(k)$ for each neighbor k that will take on the value 1 if i

[1] A specific line in an algorithm will be indicated in angular brackets ⟨ ⟩. The algorithm we refer to will be either clear from the context or indicated explicitly.

thinks that $p_k = i$ and 0 otherwise. Receipt of DCL at node k from i shows that at the time DCL was sent, node i selected k as p_i, so that $z_k(i)$ is set to 1. The nodes i for which $z_k(i) = 1$ are called *sons* of k. Observe that because of link delays, if i is a son of k, it does not mean that at the same time k is the father of i. We can now write in our notation the combination of the above routing algorithm and the extended reverse path forwarding (ERPF) broadcast algorithm of [1], where B denotes a broadcast message.

Variables at Node i: Same as in RA, and in addition $z_i(l) = 1$ if l is son of i, $= 0$ otherwise; $\forall l \in G_i$; (initialized to 0).

ERPF Broadcast Algorithm:

1) For MSG (l, α)
2) $N_i(l) \leftarrow 1$
3) *if* $l = p_i$ *then:* $\alpha_i \leftarrow \alpha$; send MSG to all $l \in G_i$ except p_i
4) *if* $N_i(l') = 1$ holds $\forall l' \in G_i$ *then:*
4a) select new p_i;
4b) *if* new $p_i \neq$ old p_i *then:* send DCL(α) to new p_i and CNCL to old p_i;
4c) send MSG(α) to old p_i; set $N_i(l') \leftarrow 0 \forall l' \in G_i$
5) For CNCL(l) set $z_i(l) \leftarrow 0$
6) For DCL(l, α) set $z_i(l) \leftarrow 1$
7) For $B(l)$
8) *if* $l = p_i$ *then:* accept B; send copy of B to all l' for which $z_i(l') = 1$.

Notes: Line <8> means that if $l = p_i$, then B is accepted, otherwise it is rejected. Recall that α is not used in the algorithm, but is included in MSG and DCL for later convenience.

B. Completeness

The above broadcast protocol is noncomplete and nonfinite. The purpose of this section is to show that completeness can be achieved by using memory and counter numbers at the nodes. We achieve our goal without requiring broadcast messages to carry any counter numbers, so that the algorithm has no overhead cost. For purposes of illustration, it is best for the time being to impose no bounds on the memory or on the counters, and also to describe the protocols as if completeness was already proved. After indicating the formal algorithm we shall show that it is indeed complete, and in the following sections we shall introduce features that will make the memory and the counters finite.

We require each node i to have a LIST_i where every accepted broadcast message is stored in the received order and also to keep a counter IC_i, counting the accepted messages (recall that all variables are indexed by SINK). Completeness of the broadcast protocol means that for any value of IC_i, the list LIST_i contains all messages sent by the source SINK up to and including counter number IC_i, with no duplicates and in the correct order. In other words, if IC_i^B denotes the value of IC_i after broadcast message B has been accepted at node i, we have $\text{IC}_i^B = \text{IC}_{\text{SINK}}{}^B$ for all B and all i. In the algorithm we also require that every DCL message sent by node k will have the format DCL(α, IC) where IC $= \text{IC}_k$ at the time DCL is sent. In this way, when a node i receives DCL from k, it will have updated information about the "state of knowledge," denoted by $\text{IC}_i(k)$, of its new son k. Only broadcast messages B with $\text{IC}_i^B > \text{IC}_i(k)$ need to be sent by i to k.

The formal algorithm is now as follows.

Variables and Buffers at Node i: Same as in ERPF, and in addition

- LIST_i buffer in which all accepted broadcast messages are stored in the received order (infinite storage) (initially empty)
- IC_i counts accepted broadcast messages (values $0, 1, 2, \cdots$) (initialized to 0)
- $\text{IC}_i(l)$ values of IC_l as presently known by i, $\forall l \in G_i$ (values $0, 1, 2, \cdots$) (initialized to 0).

The Complete Routing-Broadcast Algorithm (CRB) for Node i:

1) For MSG(l, α)
2) $N_i(l) \leftarrow 1$
3) *if* $l = p_i$ *then:* $\alpha_i \leftarrow \alpha$; send MSG($\alpha_i$) to all $l' \in G_i$ except p_i
4) *if* $N_i(l') = 1$ holds $\forall l' \in G_i$ *then:*
4a) select new p_i
4b) *if* new $p_i \neq$ old p_i *then* send DCL(α_i, IC_i) to new p_i and CNCL to old p_i
4c) send MSG(α_i) to old p_i; set $N_i(l') \leftarrow 0 \ \forall l' \in G_i$
5) For CNCL(l) set $z_i(l) \leftarrow 0$
6) For DCL(l, α, IC) set $z_i(l) \leftarrow 1$
6a) *if* IC $< \text{IC}_i$ *then:* send to l contents of LIST_i from IC + 1 to IC_i while incrementing $\text{IC}_i(l)$ up to IC_i
6b) *else* $\text{IC}_i(l) \leftarrow$ IC
7) For $B(l)$
7a) *if* $l = p_i$ *then:* $\text{IC}_i \leftarrow \text{IC}_i + 1$; include B in LIST_i;
7b) $\forall j \in G_i$ for which $z_i(j) = 1$ and $\text{IC}_i(j) < \text{IC}_i$ *do*
7c) send B to j; $\text{IC}_i(j) \leftarrow \text{IC}_i(j) + 1$.

The proof that the CRB protocol is indeed complete appears in the Appendix. Here we only mention that the important property leading to completeness is the statement of Lemma A1, which will be called the *session property*. Broadcast protocols associated with other routing algorithms can be made to have this property, but several additions to the algorithm are necessary. It is a special feature of the routing protocol of [2] that the session condition holds with no extra instructions. Also observe that, as will be seen in Lemma A2 and Theorem A1, completeness is achieved without requiring messages to carry their counter number.

C. Finiteness

Completeness means that broadcast messages are accepted at nodes in the correct order and with no duplicates or messages missing. However, it does not ensure that all messages

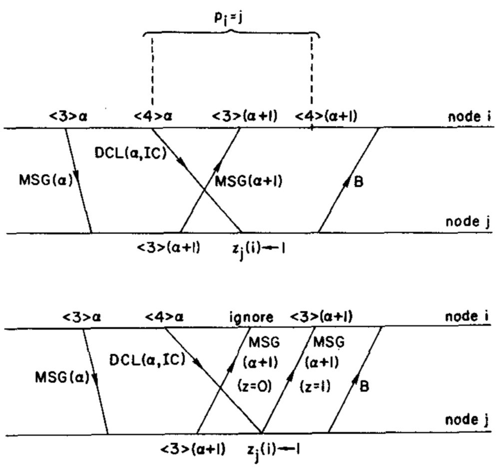

Fig. 1. Timing diagram for Section II-C. (a) CRB. (b) RRB.

are indeed accepted at all nodes. The following scenario shows that, since we allow arbitrary propagation time for messages on each link, there may be a situation in the CRB algorithm where a node i accepts no messages from a certain time on.

Consider Fig. 1(a), where <3> α denotes execution of line <3> of cycle α in CRB. Suppose that $p_i = j$ between <4> α and <4> $(\alpha + 1)$, while $p_i \neq j$ holds outside this interval. Then upon execution of <3> α and <4> α, node i sends MSG(α) and DCL(α, IC), respectively, to j. If the propagation time of DCL(α, IC) is long enough, it may happen that node j performs <4> α, cycle α is completed at SINK, and node j performs <3> $(\alpha + 1)$ before receipt of DCL(α, IC) at j. In this case, node i may perform <3> $(\alpha + 1)$ and <4> $(\alpha + 1)$ before the time it receives a broadcast message B and then $p_i \neq j$ so that B is not accepted. This scenario can be repeated indefinitely, so that B and the broadcast messages following it keep arriving at node i but are never accepted.

In order to correct the situation and achieve finiteness, we introduce an "*impeding mechanism*" in the CRB algorithm. Control messages MSG(α) sent from j to i will carry an additional variable $z = z_j(i)$. Any control message MSG(α, z) with $\alpha = \alpha_i + 1$, $z = 0$ received from $j = p_i$ will be ignored [see Fig. 1(b)]. If node j receives DCL(α, IC) with $\alpha < \alpha_j$, in which case by Lemma A3 we have $\alpha = \alpha_j - 1$, node j retransmits MSG(α_j, z) with $z = 1$. Thus, node i postpones execution of <3> until it receives acknowledgment from $j = p_i$, in the form of MSG(α_j, $z = 1$), that the last DCL message has been received at j. In this way we at least guarantee that all broadcast messages sent at the time of receipt of DCL(α, IC) (line <6a> in CRB or <6a> <6b> in RRB) will be accepted at node i.

For each broadcast message accepted at a node i, it is convenient at this point to indicate explicitly the cycle during which it was accepted. To do so we replace LIST$_i$ by a set of buffers LIST$_i(\alpha)$, $\alpha = 1, 2, \cdots$ (for the meantime an infinite number of unbounded buffers) and all broadcast messages accepted while i was in cycle α are stored in LIST$_i(\alpha)$. By definition, a node i is in cycle α if $\alpha_i = \alpha$. Also, counters $C_i(\alpha)$ are used, counting messages accepted during cycle α. Out of the messages corresponding to cycle α, those that have been accepted at neighbor l as fas as i knows are counted in $C_i(l)(\alpha)$. Consequently, the counter IC is redefined as the pair IC $= (\alpha, C(\alpha))$, where IC$' <$ IC$''$ means that either $\alpha' < \alpha''$ holds or both $\alpha' = \alpha''$ and $C'(\alpha') < C''(\alpha'')$ hold.

The resulting algorithm is given below and the proof that it is complete and finite appears in the Appendix.

Variables and Buffers at Node i: Same as in ERPF and in addition

LIST$_i(\alpha)$	buffers in which all broadcast messages accepted while i is in cycle α are stored $(\alpha = 0, 1, 2, \cdots)$
$C_i(\alpha)$	counts broadcast messages accepted during cycle α $(\alpha = 0, 1, 2, \cdots)$ $(C_i(\alpha) = 0, 1, 2, \cdots)$
$C_i(l)(\alpha)$	value of $C_i(\alpha)$ as presently known by i, $\forall l \in G_i$.

The Reliable Routing-Broadcast Algorithm (RRB) for Node i:

1) For MSG(l, α, z)
2) *if* $l \neq p_i$ *then*: $N_i(l) \leftarrow 1$
3) *if* $l = p_i$ and $z = 1$ *then*: $N_i(l) \leftarrow 1$; $\alpha_i \leftarrow \alpha_1 + 1$; send MSG($\alpha_i$, $z_i(l)$) to all $l' \in G_i$ except p_i
4) *if* $N_i(l') = 1$ holds $\forall l' \in G_i$ *then*:
4a) select new p_i
4b) *if* new $p_i \neq$ old p_i *then*: send DCL(α_i, $C_i(\alpha_i)$) to new p_i and CNCL to old p_i
4c) send MSG(α_i) to old p_i; set $N_i(l') \leftarrow 0 \forall l' \in G_i$,
5) For CNCL(l, α, C) set $z_i(l) \leftarrow 0$
6) For DCL(l, α, C) set $z_i(l) \leftarrow 1$
6a) *if* $C < C_i(\alpha)$ *then*: send to l contents of LIST$_i(\alpha)$ from C to $C_i(\alpha)$ while incrementing $C_i(l)(\alpha)$ to $C_i(\alpha)$
6b) *if* $\alpha = \alpha_i - 1$ *then*: send MSG(α_i, $z_i(l)$) to l; send to l contents of LIST$_i(\alpha_i)$ from 1 to $C_i(\alpha_i)$ while incrementing $C_i(l)$ to $C_i(\alpha)$
6c) *else, if* $C \geqslant C_i(\alpha)$ *then*: $C_i(l)(\alpha) \leftarrow C$
7) For $B(l)$
7a) *if* $l = p_i$ *then*: $C_i(\alpha_i) \leftarrow C_i(\alpha_i) + 1$; include B in LIST$_i(\alpha_i)$;
7b) $\forall j \in G_i$ for which $z_i(j) = 1$ and $C_i(\alpha_i)(j) < C_i(\alpha_i)$ *do* send B to j; $C_i(\alpha_i)(j) \leftarrow C_i(\alpha_i)(j) + 1$.

Before proceeding, we note here that the impeding mechanism slows down the routing algorithm, but only in extreme situations. This is because the impeding mechanism is in fact activated only in the case when DCL(α, C) sent by a node i to j arrives there after node j has performed <3> of cycle $(\alpha + 1)$. Since such a DCL message is sent by i when it performs <4> of cycle α, this means that propagation of DCL on link (i, j) takes more time than propagation of the routing cycle α from i all the way to SINK plus propagation of cycle $(\alpha + 1)$ all the way from SINK to node j. This may indeed happen if we allow arbitrary delays on links, but the chances are small.

III. THE RELIABLE BROADCAST PROTOCOL

The final form of the broadcast protocol will be obtained from the RRB algorithm after making several observations.

1) The broadcast messages accepted by node i while it is in cycle α are exactly those broadcast messages released by SINK while it is in cycle α (follows from Corollary A1).

2) If node i is in cycle α, it will never be required to send to neighbors messages accepted prior to cycle $(\alpha - 1)$, and therefore it needs to store only messages accepted during the present and the previous cycles.

From 1) and 2) it follows that we can make significant simplifications in RRB. The variables α, α_i can be binary; only two lists $\text{LIST}_i(0)$ and $\text{LIST}_i(1)$ need to be stored; if SINK is allowed to send no more than M broadcast messages per cycle, those LIST's can have finite size M; only counters $C_i(0)$, $C_i(l)(0)$, $C_i(1)$, $C_i(l)(1)$ are needed and all those are bounded by M; control messages MSG need not carry the variable α. The resulting broadcast algorithm has the following properties.

Properties of RRB (Network Has N Nodes and E Links):

1) Reliability.

2) Finite memory and counters.

3) No overhead cost.

4) Control communication cost: the routing protocol requires $2E$ messages MSG per cycle whether broadcast is operating in the network or not. Broadcast requires no new MSG messages, except in the peculiar situation described at the end of Section II-C. In addition, we need at most N DCL messages and N CNCL messages per cycle.

5) Broadcast communication cost: most of the time broadcast messages propagate on spanning trees. The only situation when two copies of the same message arrive at a node (and one is ignored) is when a broadcast message "crosses paths" with a CNCL message. This means that CNCL is sent by i to j and the broadcast message is sent by j before CNCL has arrived and is received by i after CNCL was sent. The worst case gives $2(N - 1)$ messages in the net per broadcast message, but in most cases this situation will not occur, especially if the propagation time of CNCL is small, so that the average is very close to $(N - 1)$ copies per message, which is the minimal broadcast communication cost.

6) Delay: the routing algorithm tends to find paths with small total weight (sum of link weights from nodes to SINK). The delay of broadcast messages will be small if the weights are link delays and the traffic is symmetric on links or if the weights of link (i, j) contain a measure of the delay on link (j, i).

APPENDIX A

Here we prove that the CRB protocol of Section II-B is complete and that the RRB protocol of Section II-C is complete and finite. First we recall several properties of the routing protocol (RA) of [2] indicated in Section II-A and introduce additional definitions.

1) In each cycle α, the routing protocol requires each node i to send exactly one MSG(α) to each neighbor.

2) Cycle α starts when SINK sends MSG(α) to all its neighbors (<3> in the algorithm for SINK) and ends when SINK receives MSG(α) for all neighbors (line <4>).

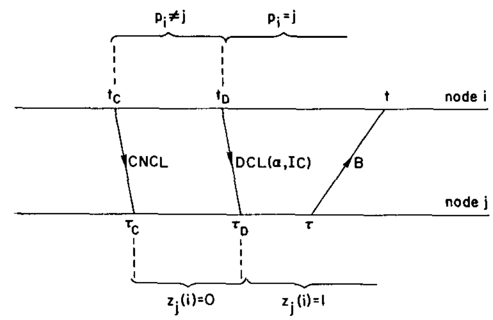

Fig. 2. Timing diagram for Lemma A1.

3) A node i is said to be in cycle α while $\alpha_i = \alpha$, i.e., from the time it performs <3> with $\alpha_i \leftarrow \alpha$ and until it performs <3> with $\alpha_i \leftarrow \alpha + 1$.

4) At the time just before node i performs <3> we have $\alpha = \alpha_i + 1$, so that α_i always increases by 1.

5) Whenever we need to indicate the value of a variable, say p_i, at a certain time t we shall write $p_i[t]$.

Lemma A1 (Session Property): Consider the CRB protocol of Section II-B. If a broadcast message B is received at time t at node i from j and *it is accepted*, then B was sent by j after receiving the last DCL message sent by i until time t.

Proof: Let $\tau < t$ be the time B was sent by j. Since broadcast messages are accepted only from fathers (see <9> of CRB) and sent only to sons (see <7> and <10>), we have $p_i[t] = j$ and $z_j(i)[\tau] = 1$. Thus, the last DCL message sent by i before time t (at time t_D, say) was indeed sent to j and we want to show that it was received by j (at time τ_D, say) *before time* τ, or in other words, i is the son of j at time τ as a result of this last DCL and not of some previous DCL's. This is exactly the session property. The timing diagram is given in Fig. 2. Consider also the last CNCL sent by i before t to j and let t_C, τ_C, α be, respectively, the time it was sent, the time it was received, and the cycle number of i at time t_C. Clearly, $t_C < t_D$ and by FIFO we also have $\tau_C < \tau_D$. In order to prove the lemma we need to show that $\tau_D < \tau$. Observe now that $z_j(i) = 0$ between τ_C and τ_D and since $z_j(i)[\tau] = 1$, time τ cannot be between τ_C and τ_D. It is therefore sufficient to show that $\tau_C < \tau$. Observe that <4b> shows that CNCL is sent after receiving MSG(α) from all neighbors, in particular j, and before sending MSG(α) to j and therefore $\alpha_j[\tau_C] = \alpha$, where α_j is the cycle number of node j. Suppose now that $\tau_C > \tau$. Then $\alpha_j[\tau] \leqslant \alpha$ and B was sent (and received, by FIFO) from j to i before MSG($\alpha + 1$), so that i could not have performed <4> of cycle $\alpha + 1$ before t. Since p_i changes only in <4>, it follows that $p_i[t] = p_i[t_C +] \neq j$ which is a contradiction. This proves the session property of the routing-broadcast protocol of Section II-B. Observe that the proof relies heavily on the properties of the routing protocol of [2].

Lemma A2: If broadcast message B is received at node i from j and is accepted, then $\text{IC}_i^B = \text{IC}_j^B$. (Recall that IC_i^B denotes the value of the counter IC_i just after node i has accepted B.)

Proof: Consider the notations of Lemma A1 and of Fig. 2. From line <4b> in the CRB algorithm it follows that the DCL(α, IC) message carries the counter number $\text{IC} = \text{IC}_i[t_D]$. Since $p_i = j$ on the interval $(t_D, t]$, node i accepts during this time broadcast messages only from j, and by the session property, those are sent only after time τ_D at which j performs <6>, <7>. Now it is easy to check (see <7>, <9>-<11> for node j) that in both cases, $\text{IC} < \text{IC}_j[\tau_D-]$ and $\text{IC} \geqslant \text{IC}_j[\tau_D-]$, node j will consecutively send to i after τ_D the broadcast messages corresponding to counter number IC + 1, IC + 2, etc. When they will be received and accepted at i, the counter IC_i will be increased, respectively, to IC + 1, IC + 2, etc.

Theorem A1: The CRB algorithm of Section II-B is complete, i.e., $\text{IC}_i^B = \text{IC}_{\text{SINK}}{}^B$ holds for every node i and every broadcast message B.

Proof: If the above relation does not hold, let i and B be the node and broadcast message for which it is violated for the first time throughout the network, and let t be the time B was accepted at i. If B was received from j, then Lemma A2 implies $\text{IC}_j^B = \text{IC}_i^B$ so that $\text{IC}_j^B \neq \text{IC}_{\text{SINK}}{}^B$. But B was accepted at j before being accepted at i, violating the fact that the statement of the theorem held throughout the network until time t.

For future reference we need the following lemma.

Lemma A3: If DCL(α, IC) arrives at node j, then $\alpha = \alpha_j$ or $\alpha_j - 1$.

Proof: Consider the notations of Lemma A1 and of Fig. 2. Then $\alpha_i[t_D] = \alpha$, and therefore MSG($\alpha + 1$) will be sent from i to j after the DCL message. Consequently, <4> of cycle ($\alpha + 1$) can be performed at j only after τ_D; hence, $\alpha_j[\tau_D] \leqslant \alpha + 1$. On the other hand, t_D is the time i performs <4> of cycle α, and hence MSG(α) has been received at i from j before or at t_D, so that $\alpha_j[\tau_D] \geqslant \alpha$.

We next proceed to the proof that the RRB protocol of Section II-C is complete and finite.

Lemma A4: In the RRB protocol, if a MSG(α', $z = 0$) arrives at i from $j = p_i$ (and by <2>, <3> is ignored), then MSG(α', $z + 1$) will arrive at i finite time from j, and then j will still be the father p_i of i.

Proof: With the notations of Fig. 1, where B is replaced by MSG(α', $z = 0$), it holds that $\tau < \tau_D$ (since $z = 0$) and $t > t_D$ (since $p_i = j$). Now $\alpha_j[\tau_D] \geqslant \alpha_j[\tau] = \alpha' = \alpha_i[t] + 1 \geqslant \alpha_i[t_D] + 1 = \alpha + 1$, where the second equality follows from property 4) at the beginning of the Appendix. From Lemma A3 it follows that $\alpha_j[\tau_D] = \alpha + 1$, and hence j will send to i at time τ_D control message MSG(α', $z = 1$) according to line <6b> in RRB.

Definition: A control message MSG(α, $z = 1$) is said to be "accepted" at node i if it triggers execution of <3> in RRB at node i. Also, define the counter number associated with an accepted message MSG(α, $z = 1$) as IC_i(MSG(α, $z = 1$)) = (α, $C_i(\alpha) = 0$).

Lemma A5: With the above definitions, control messages with $z = 1$ propagate in RRB as if they were regular broadcast messages.

Proof: Broadcast messages are accepted at i only if they arrive from p_i and are sent to sons, either when they are accepted or in response to DCL with $\text{IC} < \text{IC}_i$. Control messages MSG(α, $z = 1$) are accepted only if they arrive from p_i and are sent to sons, either when they are accepted (<3> in RRB) or in response to DCL with $\text{IC} < \text{IC}_i$ (<6b> in RRB). Moreover, MSG(α, $z = 1$) is accepted at i before all broadcast messages B with $\text{IC}_i^B = (\alpha, C_i(\alpha))$, since node i enters cycle α as a result of accepting MSG(α, $z = 1$) from p_i, and broadcast messages with IC_i^B as above are all accepted while i is in cycle α. Now, MSG(α, $z = 1$) is sent to any node before all such broadcast messages (see <3> and <6b>), so that the order is preserved as well; hence the statement of the Lemma.

Corollary A1: The combination of broadcast messages and control messages with $z = 1$ performs a jointly complete algorithm, i.e., all such messages are accepted in the order released by the source node SINK, with no duplicates and no messages missing.

Theorem A2: The RRB protocol is complete and finite.

Proof: From Lemma A4 and the fact that every routing cycle of the algorithm of [2] propagates in finite time, it follows that the propagation of control messages with $z = 1$ is finite, namely, every node enters every cycle in finite time. By Corollary A1, all broadcast messages released by SINK while SINK is in cycle α are accepted at each node while the node is in cycle α, and since each node enters cycle ($\alpha + 1$) in finite time, all such broadcast messages are accepted at each node in finite time.

REFERENCES

[1] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *Commun. Ass. Comput. Mach.*, vol. 21, pp. 1040–1048, Dec. 1978.

[2] P. M. Merlin and A. Segall, "A failsafe distributed routing protocol," *IEEE Trans. Commun.*, vol. COM-27, pp. 1280–1287, Sept. 1979.

[3] Y. K. Dalal, "Broadcast protocols in packet switched computer networks," Ph.D. dissertation, Stanford Univ., Stanford, CA, DSL Tech. Rep. 128, Apr. 1977.

[4] J. M. McQuillan, I. Richer, and E. C. Rosen, "The new routing algorithm for the ARPANET," *IEEE Trans. Commun.*, vol. COM-28, pp. 711–719, May 1980.

★

Adrian Segall (S'71–M'74–SM'79) was born in Bucharest, Rumania, on March 15, 1944. He received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion—Israel Institute of Technology, Haifa, in 1965 and 1971, respectively, and the Ph.D. degree in electrical engineering with a minor in statistics from Stanford University, Stanford, CA, in 1973.

After serving active duty in the Israel Defence Army, he joined in 1968 the Scientific Department of Israel's Ministry of Defence. From 1973 to 1974 he was a Research Engineer at System Control, Inc., Palo Alto, CA, and a Lecturer at Stanford University. From 1974 to 1976 he was an Assistant Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology, Cambridge, and in 1976 he joined the faculty of the Technion. Since July 1982 he has been on leave with the IBM T. J. Watson Research Center, Yorktown Heights, NY. His current research interest is in the area of computer communication networks.

Dr. Segall was selected as a delegate to the First IEEE-USSR Information Theory Workshop, and was the 1981 recipient of the Miriam and Ray Klein Award for outstanding research.

★

Baruch Awerbuch (S'83) was born on November 7, 1958. He received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion—Israel Institute of Technology, Haifa, in 1978 and 1982, respectively, and is currently a Ph.D. student in the Department of Electrical Engineering at the Technion.

From 1978 to 1981 he served in the Israel Defence Army. From 1981 to 1982 he was a Teaching Assistant at the Technion in communication, control, and optimization courses. Currently he is with the IBM Scientific Center, Haifa. His research interests lie mainly in the areas of distributed and parallel computation, computer networks, and algorithmic complexity.