AN EFFICIENT BROADCAST PROTOCOL EMBEDDED IN MULTI-HOP RADIO NETWORKS

by

T. Chlamtac and S. Kutten

Technical Report #328

July 1984
ABSTRACT

In this paper we consider the problem of obtaining an efficient broadcast protocol in multi-hop radio networks. The proposed protocol is designed to provide an access method with high broadcast capacity and bounded delay while using only a small amount of network bandwidth and node memory in the broadcasting process. Thus during an ongoing broadcast activity the nodes can continue to use the channel for other non broadcast related transmissions using an existing data link multiaccess protocol. As a result, we obtain a combination of a "deterministic" type of service for the broadcast activity embedded in a multiaccess data link protocol designed for random demands. This integration of the differing services, concurrently using the same network resources, is comparable in concept to hybrid circuit/packet switching. Thus applications with different service criteria can be interleaved in the same radio network.
1. INTRODUCTION

A number of packet radio networks have been introduced in recent years into the civilian and military sectors \[9, 10, 18, 21, 22, 24, 25\]. In many practical implementations not all network nodes directly communicate with each other so that multi-hop radio network topologies are obtained. Communication in these networks must take into account the broadcasting nature of the radio channel whereby each transmission arrives at all the nodes which are in line of sight and within range of the transmitting node. Multiple transmissions arriving concurrently at a given node will cause "collisions" and subsequent loss of information. These network characteristics effect the design and evaluation of both the data link and network layer protocols \[5, 17\].

At the network level the broadcasting protocols play an important role in information dissemination \[2, 11, 13, 14, 25\]. Several such protocols have been proposed for radio networks \[6\]. These protocols use a random access data link protocol which provides variable quality service. When alternatively the data link layer is dedicated to the broadcast function then broadcast protocols developed for point to point networks can be considered to obtain a predictable service.

In \[3\] it was shown that a straightforward implementation of "point to point" protocols to optimize the radio network performance is not practical but that heuristics with predictable service can be developed when the data link access is controlled solely by the broadcast protocol. In this paper we do not require that data link protocol be completely dedicated to the network layer (broadcast) activities. However, by time multiplexing the data link and broadcast activities we obtain efficient broadcasting with guaranteed level of service while "bursty" data arrivals can still be serviced by a random access mechanism. The only requirement made is that nodes participating in the broadcasting activity execute an identical broadcasting protocol with prior-
ity higher than that of the data link protocol layer.

In the following discussion we show how to implement the protocol in networks using any time division based allocation of the channel. Similar protocols can be had for networks using frequency division based data link protocols [3]. We shall here assume that nodes are synchronized and can correctly receive transmissions emanating from their neighbors subject to absence of collisions. A large number of existing network architectures is consistent with these data link layer requirements [15,19,21,22,24].

The basic approach in the protocol design is the construction of broadcasting trees over which the broadcast can flow with high capacity and bounded delay. The tree construction process can use existing data link protocols producing a time division based allocation of transmission rights for the broadcast activity while reducing the amount of network resources dedicated to it. In other words the number of time slots during which each node is dominated by the broadcast protocol oriented allocation and the amount of node memory required are relatively low. Thus, the total effect of using the proposed protocol is the propagation of broadcasts over a reliable broadcasting tree constructed on top of possibly unreliable data link mechanism using limited network resources to obtain guaranteed service broadcast activity.

In Sections 2, 3 and 4 we propose a formal model for multi-hop radio networks, we present relevant performance measures and analyze the properties of optimal and "near optimal" broadcasting.

Section 5 presents verifiable, linear time, centralized and distributed algorithms for building the broadcasting trees.

Section 6 gives a broadcast protocol and provides bounds on protocol performance. The discussion in Section 7 closes the paper.

2. MODEL DESCRIPTION

For construction and analysis of subsequent algorithms we construct the following network model. We present a multi-hop radio network by a
directed-graph \( G(V, E) \). Vertices in \( V \) represent the network nodes, \( |V| = N \), and an undirected edge \( e \in E \) is interpreted as two anti parallel directed edges. A directed edge \( u \rightarrow v \) signifies that \( v \) receives every signal transmitted by \( w \); \( w \) is defined as an incoming neighbor of \( v \). A correct (collision free) reception by \( v \) is conditioned by:

a. \( v \) does not transmit in the same slot \( w \) does (i.e., \( v \) cannot transmit and receive simultaneously).

b. \( v \) is the only incoming neighbor of \( v \) transmitting in the same slot (i.e., \( v \) cannot receive two simultaneously incoming transmissions).

Conditions a. and b. define the two cases of packet collisions in the radio network. For networks discussed here we shall assume that transmissions originated in different slots do not collide. Finally, we assume that a broadcast can be delivered to all nodes from the broadcast-source node. We define a network for which this condition holds to be "source connected". Notice that although in this discussion time interpretation of slots is assumed in a general model a slot can also represent a frequency subband [3].

3. MEASURES FOR EVALUATING THE EFFICIENCY OF A BROADCAST PROTOCOL

We propose a set of measures to evaluate a broadcast protocol in terms of its efficiency by observing the broadcast delay and capacity, and by evaluating the amount of resources consumed by the broadcasting activity in terms of node memory requirements and bandwidth unavailable for other network activities.

We define:

Maximum broadcasting delay, \( D_{\text{max}} \): the maximum number of times slots needed for the broadcasted message to reach all nodes.

Average maximum broadcasting throughput (i.e. average capacity) \( T_{\text{max}} \): the average (over the time slots) number of the maximum number of first correct recep-
tions of a broadcasted message in the basic time slot.

Average memory requirements, \( M_{-}\text{avg} \): (assuming each message requires one buffer) the average number of time slots a node is required to store the broadcasted message.

Average broadcast bandwidth consumption, \( C_{-}\text{avg} \): the average number of time slots during which a node cannot receive other but the broadcasted message.

Recall that during a broadcast a node cannot use the channel not only due to its own broadcast transmission and reception but also to avoid disturbing the broadcast activities of its neighbors. Thus the measure of bandwidth consumption reflects the efficiency of the chosen broadcast protocol: the protocol with the minimal consumption will use up the least amount of bandwidth (over time) to perform the broadcast, leaving the nodes free for non-broadcast related communication. In the optimal case when the broadcast can be performed in a single time slot (e.g., in a fully connected network) we have \( C_{-}\text{avg} = 1 \), i.e., the lower bound is obtained. On the other hand, in the worst case where only one additional node receives the broadcast message in each time slot we have, \( C_{-}\text{avg} = N \). Thus, obtaining for a given broadcasting protocol, \( C_{-}\text{avg} > N \) implies that there exist time slots in which the protocol performs no useful work, that is no new nodes are covered by the broadcast. Thus, for a broadcast protocol, \( 1 \leq C_{-}\text{avg} \leq N \) should be obtained.

4. THE COST OF CONSTRUCTING BROADCASTING PROTOCOLS

Correct design of a broadcasting protocol should lead to the optimization of the defined measures. In [3] it was shown that minimizing broadcast delay defined in the previous section involves high complexity algorithms. Here we show that constructing a protocol which allocates transmission rights in a way which optimizes bandwidth consumption or memory requirements is not practical. Similar conclusions are obtained with regard to the optimizations of the number of broadcast transmissions and the number of
transmitted copies. The proofs of the following theorems are given in Appendix A.

4.1 The Complexity of Optimal Solutions

Notice that minimization of bandwidth consumption reflects a policy which minimizes the cost of utilizing the channel by the broadcast, while minimization of memory requirements reflects the store and forward delay and buffering.

**Theorem 1:** The problem of $C$-avg minimization is NP-Hard.

**Theorem 2:** The problem of $M$-avg minimization is NP-Hard.

4.2 Heuristics for Bandwidth Consumption Minimization

Examination of the preceding theorems and their proofs shows that the high complexity of optimal solutions results from the need to consider the mutual dependencies of all nodes before making an "optimal" transmission decision. To obtain a good heuristic we try to reduce this set of dependencies by limiting this optimization to the transmission instances. Specifically, we notice that broadcast consumption is induced in those time slots when the node, or any neighbor of that node, transmits the broadcasted message. In the three following heuristics we concentrate only on the actions of the neighboring nodes and progressively reduce the optimality requirements in order to reduce the complexity.

**Definition:** Neighbors'-induced-consumption, $C$-neighbors: the number of time slots in which at least one of the node's neighbors is transmitting the broadcasted message.

**Heuristic H1:** Perform the broadcast minimizing the average neighbors'-
induced-consumption.

Theorem 3: The problem of C-neighbors minimization is NP-Hard.

For neighbor transmissions optimization heuristics H1 must, of course, recognize simultaneous transmissions to avoid counting such time slots more than once. To simplify this counting procedure one can only observe the number of copies sent, i.e., the number of arrivals of the message at the network nodes, regardless of the time slots they occupy. Intuitively, the complexity of the resulting optimization problems will thus be reduced. (Notice also that this heuristic favors broadcasting in the network through areas of low connectivity.)

Heuristic H2: Perform the broadcast minimizing the total number of transmitted copies.

Theorem 4: The problem of minimizing the total number of copies in the broadcast is (still) NP-Hard.

The final simplification of the bandwidth consumption minimization problem is obtained by counting only the number of transmissions of the broadcasted message. Recalling that bandwidth is consumed by transmissions of the node's neighbors and the node itself, the heuristic derived by minimizing the total number of transmissions can be also viewed as an approach complementing heuristics H1 and H2.

Heuristic H3: Perform the broadcast minimizing the total number of transmissions.

Theorem 5: The problem of minimizing the total number of transmissions is (still) NP-Hard.
It is worth pointing out that regardless of their relation to the original problem (see Theorem 1), the objectives of heuristics H2 and H3 are independently important measures in evaluating a broadcast protocol [2,17]. In other words, Theorems 4 and 5 show that in radio networks the construction of protocols which minimize the number of transmissions or the number of copies is not practical.

4.3 A Heuristic for Minimizing the Memory Requirements

In the broadcasting process a node must keep a copy of the message in its memory from the time slot it was received and until the message is successfully sent to all those neighbors of \( i \) which depend on node \( i \) for the message reception. To cover all these neighbors \( i \) may be required to transmit the message in several time slots, since the reception of \( i \)'s transmission depends also on the activities of the neighbors and their neighbors in turn. To minimize the time a node is required to store a copy of the message, all subsequent transmissions must therefore be coordinated. As shown in Theorem 2, this yields an NPH problem. We simplify the requirement for coordination as follows: enable each node to cover all of its uncovered neighbors (nodes which have not received the message so far) by a single transmission. This transmission will not necessarily occur in the slot immediately following the slot in which the message was received. To satisfy this requirement we request that no two nodes, with a common uncovered neighbor, transmit in the same time slot. This requirement guarantees that if a node transmits a message to its neighbor, no simultaneous transmissions of that neighbor's neighbor will interfere with it! By introducing this restriction we intuitively reduce the memory requirements while obtaining a policy which requires the inspection of a more limited number of potential transmission sequences.

Heuristic \( H4 \): Perform the broadcast minimizing \( M \)-avg under the restriction that no two nodes, having a common uncovered neighbor, may transmit in
Theorem 6: The problem of minimizing M-avg. under the restriction that no two nodes, having a common uncovered neighbor, may transmit in the same time slot is NP-Hard.

5. LINEAR TIME ALGORITHMS FOR CONSTRUCTING A BROADCASTING TREE IN MULTI-HOP NETWORKS

The following algorithms construct broadcasting trees rooted at the source of the message and produce an associated time based division of transmission rights. The broadcast protocol given in the following section utilizes this information to perform the broadcast efficiently in terms of the measures presented earlier.

5.1 Centralized Broadcast Tree Construction

We assume that the source node knows the network topology and all nodes have distinct identities. The algorithm assigns timeslots to all nodes so that each node can receive the broadcast message correctly from its predecessor (father in the tree) and no node interferes with broadcast transmissions directed to its neighbors. (Centralized control is used in several radio networks [18,20,22,24].)

Definitions:
Root - the source node, s, from which tree construction is initiated.

K - the maximum number of neighbors of any network node.

Covered node - we say that the source node is covered by the broadcast at the broadcast initiation and a node \( v \) becomes covered after a covered neighbor of \( v \) has transmitted without collision occurring in \( v \).

Covered network - a network is covered by the broadcast when all its nodes become covered.

Numbered node - a node which has already been assigned the number of time slot
Transmission hearing relations:

(1) A node \( v \) hears (the message) from node \( w \) in timeslot \( i \) iff a) \( w \) is \( v \)'s only neighbor which has been assigned the slot number \( i \) and b) slot \( i \) has not been assigned to \( v \).

(2) A node other than the source hears iff there is at least one timeslot in which (1) holds for that node. (The source hears by initiating the broadcast.)

(3) Father \( \rightarrow \) son relation: \( u \) is defined as the father of all nodes which hear (i.e., start hearing) by the virtue of the numbering of \( u \).

The Centralized Tree Algorithm

BEGIN
Assign timeslot number 1 to the source
WHILE there is a node which hears but is not numbered DO
BEGIN
FOR \( i = 1 \) TO \( \text{max\_number\_of\_slots\_assigned\_so\_far} \) DO
WHILE there is a set \( V \) of unnumbered nodes which already hear in slot number \( i \) - with \( v \in V \) the first node which became hearing in set \( V \) - DO
BEGIN
Assign \( v \) the lowest sequential slot number \( j \) under the conditions:
\( c1: j \) is not the timeslot of \( v \)'s father
\( c2: j \) has not been assigned to a father of \( v \)'s neighbor
END;
END;
END.

Comments:

1. The timeslot numbers are assigned from the set of positive integers.

2. The "For" statement of the algorithm uses a dynamically changing value "\( \text{max\_number\_of\_slots\_assigned\_so\_far} \)"; i.e., its value may change in every execution of the "assign.....slot" statement.
Theorem 7: The following properties hold for the centralized tree algorithm.

a. The total number of distinct timeslots is bounded by \( \min\{K+1,1+2\sqrt{|E|}\} \).

b. The algorithm always stops successfully and is linear in the number of edges, \( 0|E| \).

c. The algorithm constructs a spanning tree routed at the source.

d. If every node transmits in its assigned timeslot then no collisions occur during the broadcast in the father to sons direction.

e. The number of timeslots \( |t| \) a node must participate in the broadcast is less or equal to the number of node's neighbors plus one.

f. We define a cycle as the total number of distinct timeslots assigned by the algorithm. Then, all broadcast related timeslots \( |t| \) at any node are concentrated in at most two consecutive cycles.
Proof of Theorem 7

Due to space limitations we do not bring the proofs of properties a to d here. Similar proofs can be found in [3].

The proof of e is immediate from conditions c1 and c2. For the proof of f we need the following lemma:

Lemma: Let A be a sequence of assignments made by the algorithm. Let B be a sequence defined by the (partial) order of nodes receiving the broadcast message based on the assignments in A. Then, A equals B up to (but not including) the (inner) order between nodes which hear in the same timeslot.

Proof: Observe that the execution of the algorithm is a simulation of the performance of the broadcast:
(1) in the simulation the value of i (the FOR loop parameter) is the slot number in a cycle of the current timeslot;
(2) the number of the iteration of the external while-loop is the number of cycles from the initiation of the broadcast.

Clearly, this observation is true for the first timeslot (counting from the initiation of the broadcast) where the assignment is to the root. Let us assume that the observation is true until and including the j-th timeslot counting from initiation of the broadcast. Assume, (in addition to 1 and 2) that (3) all transmission assignments already done (in the simulation) are assignments to nodes that have already received the broadcast. Let us partially order those nodes according to their assigned slot - the ordering being done cyclically starting with i + 1. Knowing that every such node will transmit in the nearest timeslot allowed by its assignment, it is clear that the first nodes to receive the broadcast are the sons of the nodes corresponding to the first class of the partial order defined above. Clearly, those same nodes are also the first to be chosen for assignment by the simulation. This follows from
assumption (1) and from the fact that the FOR loop is located in the external WHILE loop. It is easy to see that following these slot assignments the induction assumption remains valid.

Q.E.D.

Proof of f: The timeslot in which \( f(v) \) (father of \( v \)) precedes the timeslots of \( v \)'s transmission by less than one cycle-length. Likewise, \( f(f(v)) \)'s transmission slot precedes \( f(v) \)'s transmission by less than one cycle-length. The claim follows by showing that the fathers of \( v \)'s neighbors do not transmit before \( f(f(v)) \) and not after \( v \). For a neighbor which is a son, or a father, or a brother (has the same father) of \( v \) the claim is trivial. Let \( u \) be \( v \)'s neighbor, which is not \( v \)'s son or brother or father. Had \( f(u) \) transmitted before \( f(f(v)) \), \( u \) would have become hearing before \( v \), and by the preceding lemma would have been assigned a slot before \( v \). Thus, \( v \)'s father would be the node \( u \) and not \( f(v) \). Likewise had \( f(u) \) transmitted after \( v \), \( v \) would have been \( u \)'s father, both cases leading to contradiction.

Q.E.D.

5.2. Distributed Broadcast Tree Construction

The following two algorithms are distributed. That is we assume that a node only knows the identity of its neighbors and every node participates in the tree construction. These assumptions are consistent with existing procedures in radio networks \([19, 20, 24, 26]\). The serial-distributed tree algorithm discussed first is distributed in the sense that the focus of the tree building activity rests each time with another node which uses only local information. The essence of the algorithm is the assignment of timeslots in a way consistent with conditions c1 and c2 given in the Centralized Tree Algorithm using an integrated breadth first and depth first search approach \([7]\). In this algorithm a token is generated at the source. A node \( v \) receives the token
from its father and chooses a timeslot (consistent with c1 and c2) considering in this process only those neighbors which had already been designated as sons of some father in the tree. Node \( u \) distributes its timeslot choice information to all its sons (all neighbors not designated so far as sons to other fathers). These in turn distribute their father’s slot assignment to all their neighbors. Once this distribution is accomplished, node \( u \) passes the token to its next son ("from the left") that has not been granted the token before. The procedure of distributing the timeslot information, combined with the BFS/DFS prescribed order of father-son adoption process, guarantees that every node can always choose a timeslot consistent with conditions c1 and c2 above.

In the distributed tree constructing algorithms we make the following assumptions and definitions:

**Assumptions:**

A1. Nodes are not synchronized with respect to the order of control message arrivals. This order is determined by the existing data link mechanism. However, two messages sent by node \( v \) to node \( u \) do arrive at \( u \) in the correct order. Messages are handled serially. (In previous studies control messages were assumed to be transmitted over a separate reliable channel \([19,20]\).)

A2. The arrival of a control message causes an interrupt which comes into effect when the tree construction algorithm at the node enters, or is in the wait state.

A3. The delivery of control messages generated by the tree-construction algorithm is the responsibility of the underlying data link mechanism. It is assumed however that a control message will arrive in undefined but finite time.

**Definitions:**

1. Each node can be of type: FATHER, SON or NEIGHBOR depending on the token location.

2. Control messages are of type: TOKEN, TOKEN_RETURNED;
ADOPTION_NOTICE (a father node adopts a son),
NEIGHBORS_NOTIFIED (of father's timeslots),
UNAVAILABLE_FOR_ADOPTION (a node informs its neighbors of not
being a candidate for adoption any longer);
SLOT_DELETED (from the set of assignable timeslots).

3. Each node maintains the following information:
   NODE record of (1) IDENTITY: (a unique value for each node)
   (2) TRANSMISSION_SLOT;
   (3) SET OF ASSIGNABLE_SLOTS;
   (4) FATHERS_TRANSMISSION_SLOT;
   (5) SET OF SONS;
   (6) SET OF NEIGHBORS;
   (7) ARRAY OF NEIGHBORS_FATHERS_SLOTS (the
       transmission slots of neighbors' fathers);

4. initialization of all fields in NODE is to undefined values excluding IDENTITY, SET OF
   NEIGHBORS and SET OF ASSIGNABLE SLOTS (positive integers).

5. The algorithm at each node terminates when the token is released (i.e., returned to
   the node's FATHER). At this time fields (4), (2), and (7) hold all information
   needed for broadcasting and the node can start participating again in another
   tree construction process.

The Serial-Distributed Tree Algorithm

<1> BEGIN {main program};

<2> WAIT; {for a control message arrival}

<3> END

{execute the following "interrupt procedures" on control
message arrival:}

<4> ON {arrival of} TOKEN {at NODE}

BEGIN WITH NODE DO BEGIN

<5> TRANSMISSION_SLOT = MIN [ASSIGNABLE_SLOT],
SONS := \{ v | (v \in NEIGHBORS) \land (NEIGHBORS' FATHERS_SLOTS[v] = \text{undefined}) \}.

For each v in SONS do concurrently

1. NODE adopts v as its son in the broadcasting tree.

2. BEGIN NEIGHBORS' FATHERS_SLOTS[v] := TRANSMISSION_SLOT;

3. Send message
   ADOPTION_NOTICE(ENTITY, TRANSMISSION_SLOT)
   to v;

4. wait for message NEIGHBORS NOTIFIED from v;

5. END;

6. For each v in SONS do serially
   NODE passes the token to all its sons who return it upon
   completion of their slot-assignment process;

7. BEGIN send message TOKEN to v;

8. wait for message TOKEN_RETURNED from v;

9. END;

10. send message TOKEN_RETURNED to FATHER;

11. END; { of WITH }

12. END; { of TOKEN arrival at NODE }

13. ON arrival of ADOPTION_NOTICE (adopting_father,
   adopting_father_transmission-slot) { at NODE }

   BEGIN WITH NODE DO BEGIN

14. [ASSIGNABLE_SLOTS] := [ASSIGNABLE_SLOTS] -
   father_transmission_slot;

15. FATHER := adopting_father;

16. FATHER_TRANSMISSION_SLOT := adopting_father_transmission_slot;

17. FOR each v in NEIGHBORS do concurrently
   notifying all nodes' neighbors of its father
   transmission_slot;
BEGIN

send message UNAVAILABLE_FOR_ADOPTION

(identity,father_transmission_slot)
to v;

wait {for message} SLOT_DELETED from v;

END;

send message NEIGHBORS_NOTIFIED to FATHER;

END {of WITH}

END {of ADOPTION_NOTICE arrival at SON}

ON {arrival of} UNAVAILABLE_FOR_ADOPTION (neighbor,
neighbor's_father_transmission_slot) {at NODE}

{receive information for constructing the set of sons and transmission slot choice}

BEGIN WITH NODE DO BEGIN

NEIGHBORS_FATHERS_SLOTS[neighbor]:=
neighbor's_father_transmission_slot;

[ASSIGNABLE_SLOTS]:=[ASSIGNABLE_SLOTS]-
neighbor's_father_transmission_slot;

send message SLOT_DELETED to
UNAVAILABLE_FOR_ADOPTION.neighbor;

END; {of UNAVAILABLE_FOR_ADOPTION arrival at NODE}

Comments:

1: Clearly each node is able to determine the cycle number of its transmission slot given its father's transmission slot and cycle;

2: Notice that the algorithm execution at the source node (tree root) is degenerate with token arriving from and returned to the broadcast initiating "higher level" protocol.
Theorem 8: The following properties hold for the Serial Distributed Tree Algorithm.

a. The total number of distinct time slots, i.e., the cycle length is bounded by 
   \[ \min\{k+1, 2+\sqrt{|E|}\} \].

b. The number of control messages sent in the tree construction process does not exceed \( 6|E| \).

c. The number of computation steps in the tree construction process is \( O(|E|) \).

d. The algorithm always stops at the source.

e. The algorithm constructs a spanning tree rooted at the source.

f. If every node transmits in the timeslot assigned to it (by the algorithm) then the broadcast covers the network by transmissions directed from fathers to sons with no collisions occurring in the sons.

g. The number of timeslots a node must participate in the broadcast does not exceed the number of its neighbors plus one.

Proof of Theorem 8: The following lemmas are constructive in the proof:

Lemma 1: At any given time there is only a single node in possession of the token; i.e., a single node which received a message TOKEN without sending back the message TOKEN_RETURNED (to its father) and which has received messages TOKEN_RETURNED from all nodes to which it sent the message TOKEN.

Lemma 2: On inclusion of a node \( u \) in the set SONS by \( v \), the following hold: 1) \( u \) is not included in the set SONS of any other node; 2) the values of FATHER'S_TRANSMISSION_SLOT at node \( u \) and the values of FATHER'S_TRANSMISSION_SLOT at all \( u \)'s neighbors are undefined.

Lemma 3: At the time a node \( v \) chooses a timeslot, its father's timeslot is known to \( v \)
**Lemma 4:** At the time a node \( v \) chooses a timeslot, all its neighbor's father's timeslots are known to \( v \).

**Lemma 5:** Every node receives the TOKEN exactly once.

**Lemma 6:** Every node (except the source) receives an ADOPTION_NOTICE exactly once.

**Lemma 7:** Every node sends at most one UNAVAILABLE_FOR_ADOPTION message to each neighbor.

For proof of Lemmas 1-7 see Appendix B.

**Proof of Theorem 8**

a. Notice that timeslot assignment is performed one node at a time (Lemma 1) taking each time the minimal allowable value under conditions \( c1 \) and \( c2 \) (Lemmas 3 and 4). These, however, are also the key elements used in proving property a. in Theorem 7, and thus property a. holds.

b. By counting the number of messages a node \( v \) can send:

b1: The number of TOKEN and TOKEN_RETURNED messages (does not exceed the number of \( v \)'s neighbors), since \( v \) sends one TOKEN to each son and one TOKEN_RETURNED to its father (statements <12> to <15> following <4> and Lemma 6) and since \( v \)'s father cannot become \( v \)'s son.

b2: \( v \) sends one UNAVAILABLE_FOR_ADOPTION message (from Lemma 6 and <21> to <25>) to each neighbor (excluding its father) and one NEIGHBOR_NOTIFIED to its father (from Lemma 1 and <25>).

b3: \( v \) sends no more than one SLOT_DELETED message to each neighbor (Lemma 7).

In summary therefore, no more than three control messages will traverse each
edge in each direction and the total number of control messages is bounded by $6|E|$.

c. Only a constant number of steps is executed in every node for every control message reception. In the case of loops each loop iteration is associated with the reception of a distinct control message. Therefore, and together with property f, we obtain the bound on the number of computational steps:

d. By virtue of property (c) the algorithm must terminate. Since all but the source node conclude their participation in the algorithm by sending a TOKEN-RETURNED message (Lemmas 5 to 7 and step <16>) the termination must be at the source (root of the tree).

e. The first token reception at each node is from its father following its father's token reception. Thus and from Lemma 4 no loops exist in the directed subgraph defined by the father $\rightarrow$ son relation.

Spanning Tree: Follows from Lemma 5 and the fact that nodes receive the message TOKEN from their fathers (<12> and <13>).

Rooted at the source: Follows from the argument given in the proof of "Tree" property above and from the fact that the source is the first node to execute the algorithm.

f. The proof is had by induction on the progress of the broadcast on the spanning tree (property e). Similar proof can be found in [3].

g. Since a node must refrain from transmission when one of its neighbors, or the node itself, are receiving the broadcast message (Lemmas 3 and 4 and step <5>), the property follows immediately.

Q.E.D.

The Parallel Distributed Broadcast Tree Algorithm

While the serial algorithm allows for partial concurrency in the "father adopts sons" process only one father remains in possession of the token at
any time. If the number of concurrent tokens in the algorithm is increased the tree construction can be executed in parallel in several directions starting at the root. Intuitively one can argue that this can shorten the total tree construction time. The actual number of steps will of course also depend on the network topology.

The parallel algorithm exercises the parallel construction using data structures similar to the serial algorithm. The main difference in the two approaches stems from the way conditions c1 and c2 are guaranteed. In our case the construction is initiated by the source and continued concurrently by all its sons. Noticing that interference in broadcast can occur between a node, node's father and the fathers of its neighbors the algorithm passes information about the neighbors of each node's neighbors thus covering the two hop radius critical for collision prevention. Since the construction flows in parallel a node may receive more than one adoption notice and deadlocks may occur. For deadlock resolution the highest identity node among the potential fathers is the chosen father and an additional blocking mechanism is introduced. Due to space limitations we do not bring the detailed description of the parallel distributed approach.

6. THE BROADCAST PROTOCOL

The following protocol when executed at each node provides the broadcasting mechanism using the broadcast tree defined in the preceding section.

Given: A cycle of \([C]\) timeslots and for each node:

a. The cycle number and timeslot number of its father's transmission - \(f\).

b. The cycle and timeslot numbers of its neighbors' fathers \([n]\).

c. The node's transmission timeslot number.
At node $i$ do:

1. on reception of broadcast from father transmit the message in the next transmission timeslot;
2. do not transmit (any message) in timeslots $i$ and $n$.

**Broadcast Protocol Properties**

From Theorems 7 and 8 we immediately have that the protocol guarantees that broadcast transmissions are received without collision along the broadcasting tree. Furthermore, the protocol guarantees bounds on performance of any network regarding maximum delay, throughput, memory, bandwidth consumption, the number of transmissions and the number of copies.

**Upper Bound on Maximum Delay:** Let $H$ be the longest path on the broadcast tree (starting at the source) then.

$$D_{\text{max}} \leq H \cdot |C|, \text{ (Clearly } H \leq N).$$

**Lower Bound on Maximum Throughput (or broadcast capacity):**

The following throughput is always obtained, (assuming the source is continuously generating messages for broadcast)

$$T_{\text{max}} \geq \frac{N}{|C|}.$$

**Upper Bound on Memory Requirements in the Network:**

Defining the network memory requirements as the total number of buffers at all nodes during a broadcast of a single message with the message forwarded by each node at the first timeslot assigned at that node for broadcasting we obtain:
Bound on Average Broadcast Bandwidth Consumption

With \( d \) - the maximum number of node's neighbors, we obtain:

\[
C_{\text{avg}} \leq \frac{|C|-1}{|C|}
\]

In contrast to point to point networks, in the case of radio, the number of transmissions of the broadcasted message will not generally be equal to the number of received copies. It is neither clear whether both measures can be minimized simultaneously. For a given constructed tree the number of transmissions, \( T_r \), is simple given by the number of nodes excluding leaves, while the number of received copies is clearly bounded by \( 2|E| \).

The sample network given in Figure 1 has a general topology with a limited number of nodes. The performance of the broadcast protocol is given in Table 1 and it shows that all actual measures are better than predicted by the bounds.

In regular topologies (i.e., the degrees of all nodes except leaves are equal), the various measures can be derived also for networks with large number of nodes.

In regular networks the average and the maximum number of neighbors equals \( K, K \approx 2|E|/\pi \). We thus obtain from properties 7.a and 8.a that

\[
|C| (\leq \min\{K+1, 2+2\sqrt{E}\}) = K+1
\]

Therefore, we see that in regular networks, even though \( n, |E| \to \infty \) the cycle length remains constant. It can be shown that for linear networks and trees this cycle length is optimal.

DISCUSSION

This paper presented the problem of constructing an efficient broadcasting protocol for multi-hop radio networks. The performance measures
for evaluating such protocols were given for evaluating both the broadcast transmission efficiency, and to reflect the amount of network resources consumed by the broadcast.

It has been shown that construction of algorithms which optimize these measures is not practical for a general topology network due to their high complexity and consequently linear heuristics have been presented. These were proven to guarantee bounds on broadcast delay and throughput and shown to place low requirements on the network bandwidth and memory.

Specifically, with these protocols every node dedicates its attention to the broadcast for only a limited number of timeslots and can participate in other transmissions in the remaining time.

Several observations deserve additional research. In this paper a given network topology was assumed. A number of civilian and military networks are in fact stationary in nature by having stationary users [23,25,27] or by virtue of constructing a stationary network of communication stations [18,21,22]. When this is the case or when in a mobile network the topology changes are not frequent the simplest way to update the tree is by reconstruction.

In the centralized approach the whole tree is reconstructed as proposed in previous studies [19,20]. Since in this case the broadcast passes each node in at most two slot cycles the effect of change or failure on the ongoing broadcast remains localized. In other words, if the broadcast process is not disrupted at the node or its vicinity for the period of two cycle length the broadcast is guaranteed to have been received successfully by all the node's neighbors.

In the case of distributed algorithms and, particularly the parallel distributed tree construction topology changes should be dealt with locally. To minimize the effect of a single update on the network it is desirable to limit the tree reconstruction to the smallest subset of nodes possible. Intuitively,
a tradeoff situation between the total number of assigned timeslots and the extent of tree reconstruction exists. The question of an efficient update procedure requires further research.

Secondly, we have shown that the proposed protocol can be embedded in an existing network in a way which does not limit the network in the choice of the channel access policy during timeslots not dedicated to broadcast. Thus, for example, a random access protocol can be used for bursty data traffic ("datagrams") while the broadcast allocation provides deterministic type of service ("virtual circuits"). The sharing of the channel by applications requiring different services resembles hybrid packet and circuit switching methodologies. These are suitable e.g. for the integration of voice and data a requirement often emphasized in the radio environment [13,25,26]. In this respect the question of circuit construction which covers only a subtree or a path (i.e., partial broadcast) deserves further attention.

Lastly, while throughout the current presentation a time division channel allocation has been assumed similar broadcast protocols can be obtained for networks using frequency division multiplexing [3].

APPENDIX A

Proofs of Theorems 1 to 6

Proof of Theorem 1: By reduction of the 3XC problem known as NPC [4]. We show that \( C_{\text{avg}} \) minimization is NP-Hard by proving that the following problem is NP-Complete: can a broadcast cover the network so that \( C_{\text{avg}} \) is less or equal to \( I \) timeslots:

**Input:** An undirected graph \( G(V,E) \) representing the network, a vertex \( s \) the source in \( V \) and an integer \( I \). We assume the network is source connected.

In the 3 exact cover (3XC) problem we have the following input: a collection \( C \) of sets with three members in each, \( U \) is the union of the sets.
Question: Does there exist a subcollection C' included in C, s.t. every member u in U is a member of exactly one set c in C'?

Given an input to the above problem we transform it into the problem of Theorem 1 as follows:

We construct a network graph G(V,E) s.t. V includes 1) a node c' for each set c in C (we shall term them 'C-nodes'); 2) a node u', for each member u in U (we shall term them 'U-nodes'); 3) an additional node s - the source. E includes an edge between s and each C-node, an edge between each C-node c' and each U-node u' which is a member in c, and edge between every two C-nodes. It follows that if there is a 3XC, then the nodes are occupied twice by transmissions, i.e., they cannot receive messages except the broadcast message in two slots (u nodes are consumed also when all the c-nodes are not allowed to transmit). Thus we can use \( I = \frac{2 + 2 |C| + 2 |U|}{1 + |C| + |U|} \). If no such cover exists then their C-nodes and s will be occupied by transmissions more than twice, and part of the U-nodes will be consumed more than twice - that is because all the C-nodes must be silent when one of them receives, and because some of C-nodes has to transmit in different slots in order to resolve or avoid collisions.

Q.E.D.

Proof of Theorem 2: We show that M-avg minimization is NP-Hard by showing that the following problem is NP-Complete: can a broadcast cover the whole network storing the message for time less or equal to \( \ell \) timeslots?

Question: Can a broadcast cover the network using M-avg which is less or equal to \( \ell \) timeslots? We again prove by reduction to the 3XC problem.

We shall use the construction of Theorem 1 but exclude the edges between every two C-nodes. If there is an exact cover, then the buffer in the source node is needed for one timeslot, and there is a need for one buffer timeslot in exactly \( \frac{1}{3} \cdot \text{SIZE}(U) \) C-nodes. Otherwise the number of C-nodes
which have to provide a buffer for the broadcast message would be larger or the source node would have to keep the message longer. Thus we can use

\[ I = \frac{1 + \frac{1}{3} |U|}{1 + |C| + |U|} \]

Q.E.D.

**Proof of Theorem 3:** Proof by reduction to the 3XC problem.

Question. Can a broadcast cover the network s.t. the average neighbors induced consumption, C-neighbors is less than or equal to 1 time slots at each node?

With the construction of Theorem 2 it is easy to see that if a solution to the 3XC problems exists then the source node s can transmit in the first slot and C-nodes can transmit in the second. Consequently, the average C-neighbors of every node equals one. If no solution exists then some of the C-nodes will have to transmit in different timeslots and the transmission of source's neighbors will occupy more than one slot, increasing the C-neighbors at the source. Similar argument can be made for U-nodes and

\[ I = \frac{1 + C + |U|}{1 + |C| + |U|} = 1 \]

**Proof of Theorem 4:** Input as in the proof of Theorem 2.

Question: Can a broadcast cover the network s.t. the number of copies sent is less than or equal to 1?

Using the construction of Theorem 2 we obtain:

1. given a solution to 3XC only a single copy will be sent to every U-node and \(|U|/3\) copies will be sent to the source s.

2. without such solution more than one copy will be sent to at least one U-node and more than \(|U|/3\) copies will be sent by the C-nodes to s. Thus, for

\[ I = |U| + C + \frac{|U|}{3} \] 1 copies of the message will be sent iff an exact cover exists.
Proof of Theorem 5: Input as in the proof of Theorem 2.

Question: Can a broadcast cover the network with the number of transmission less or equal to 1?

We use the construction of Theorem 2. If there is an exact cover (from the 3XC problem) then the source will transmit once, and there will be exactly $\text{SIZE}(U)/3$ transmissions by the C-nodes. Otherwise, the number of transmissions by the C-nodes must be larger. Thus, we can use $I = 1 + \frac{|U|}{3}$.

Q.E.D.

Proof of Theorem 6: Input as in the proof of Theorem 2.

Question: Can a broadcast cover the network using $M$-avg which is less or equal to 1, under the restriction that no two nodes, having a common uncovered node, may transmit in the same timeslot?

Proof is immediate by using the construction and argument of Theorem 2.

APPENDIX B

Proof of Lemmas for Theorem 8

Lemma 1: The proof is obtained by induction on the order of events associated with the passing of tokens and by noticing that the sending of token (TOKEN, TOKEN_RETURNED) and receiving of token (TOKEN, TOKEN_RETURNED) always come in pairs.

Lemma 2: We prove property 1, property 2, can be obtained similarly. We assume the converse. Suppose $u$ is included in the set SONS of its neighbor, say $w$, $w \neq v$. The inclusion of $u$ in this set can take place only when the single token (Lemma 1) is held by $w$ (starting from token reception, step <4> and until token sending, step <13> or <16>). Similarly, $v$ can include $u$ in its set SONS while in possession of the token passed to it by $w$ or an intermediate node (in <13> or <16>) prior to which $w$ executed step <7> to <11>.
and thus also step <8> to <10> for u. Notice further that the execution of step <10> by w must occur after execution of step <25> by u and thus of, of course after u has executed step <22> and <23> for each one of its neighbors (including v) see step <21>. However, the execution by u of <23> for v must have followed the execution of step <29> and prior to it of <27> by node v. Therefore, we have that following the execution of <27> by v the value of NEIGHBOR(u). FatherPhase cannot become undefined again preventing v from including w in its set SONS (in <6>) contrary to the assumption.

Lemma 3: The proof follows the technique given in the proof of Lemma 2.

Lemma 4: Given u a neighbor of v we distinguish two cases:

a. u has no father yet: it follows u has not yet received an ADOPTION_NOTICE (at <7> following <6>) and thus U could not have sent an UNAVAILABLE_FOR_ADOPTION message (at <22> following <17>). Thus since the UNAVAILABLE_FOR_ADOPTION message from u has not been received by v its NEIGHBOR(u) father phase is still undefined. (This may change only at <8> or <27> at which point u becomes v’s son at <6>.)

b. u has a father: in this case the value of Neighbor(u) father phase cannot be undefined since this would cause u to become v’s son (at <6>) contrary to Lemma 2. Since, however, at protocol initialization the value of NEIGHBORS’_FATHERS_SLOTS was undefined v must have executed <27> following the reception of UNAVAILABLE_FOR_ADOPTION message sent to it by u (on execution of <22>). In sending this message u used the father phase received (in <17>) from its father upon the execution of step <9> by u’s father (see step <7> and Lemma 2).
Thus, for both cases Lemma 4 is true.

**Lemma 5:** The proof of this lemma is similar to the proof of Depth First Search algorithm in [8].

**Lemma 6:** The lemma follows from Lemma 5 and from the fact that the node receives the ADOPITION_NOTICE from its father only when its father is in the possession of token (i.e., noticing that steps <7> to <11> are always preceded by step <4>).

**Lemma 7:** The proof is immediate using Lemma 6 and from the sequence of steps <25> to <29>. 
Figure 1 - A Sample Network

- father, son relation
- \( x \) - the node's slot assignment
- \( y \) - the order of tree construction
- \( 1 \) - the message source
\[ |V| = n = 26 \; ; \; |E| = 51 \; ; \; K = 9 \; ; \]

<table>
<thead>
<tr>
<th>Measures</th>
<th>Theoretical bounds</th>
<th>Actual values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>(cycle length) [</td>
<td>C</td>
<td>_]</td>
<td>(\min(10,16) = 10)</td>
</tr>
<tr>
<td>(consumption) C-avg</td>
<td>26</td>
<td>(\frac{75}{26})</td>
<td></td>
</tr>
<tr>
<td>(memory) M-max</td>
<td>2</td>
<td>1</td>
<td>not including reception and transmission slots</td>
</tr>
<tr>
<td>(capacity) T-max</td>
<td>(\frac{26}{10})</td>
<td>(\frac{26}{4})</td>
<td></td>
</tr>
<tr>
<td>(delay) D-max</td>
<td>(4 \cdot 5 = 20)</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Number of copies</td>
<td>(\leq 2</td>
<td>E</td>
<td>= 118)</td>
</tr>
<tr>
<td>Number of transmissions</td>
<td>(n-1 = 25)</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 1 - Performance of the Sample Network**
REFERENCES


March 1979.


