BROADCASTING FINITE STATE MACHINES: FOR MODELING LAN PROTOCOLS

M. G. Gouda*
Department of Computer Sciences
University of Texas at Austin
and
A. R. K. Sastry
Science Center
Rockwell International Corporation

ABSTRACT

We present a simple language, called *broadcasting finite state machines*, that can be used to model a variety of LAN protocols. In this language, a protocol is modeled as a network of finite state machines that communicate by broadcasting messages over a connecting bus. A novel feature of this language is the ability of a machine to detect idleness (e.g. no machine is broadcasting), transmission (e.g. exactly one machine is broadcasting), or collision (e.g. two or more machines are broadcasting) over the bus. We also discuss how to verify the safety and liveness properties of a protocol using an *aggregate reachability graph* of the modeling network. As an example, a CSMA/CD access protocol is modeled and verified using this approach.

1. INTRODUCTION

Interest in Local Area Networks (LANs), and their communication protocols is growing rapidly. A good deal of effort has gone into the development of standard protocols for LANs, the most notable among them being the adoption of the IEEE/ANSI standards for CSMA/CD [1], Token passing bus [2], and Token passing ring [3]. While descriptions of these protocols are detailed and exhaustive, they remain largely narrative and do not incorporate a good degree of formalism. We believe that a major reason for this phenomenon is that the technical community has so far ignored the development of communication primitives that are based on "broadcasting," which is the basic primitive in most LAN protocols. Indeed most of the proposed primitives in the literature are "point-to-point," e.g. CSP [4], CCS [5], and CFSM [6,7,11]. In the absence of broadcast communication primitives, the exercise of modeling and verifying a bus-oriented LAN protocol becomes cumbersome.

In this paper, we introduce a minimal language for defining networks of autonomous finite state machines that communicate exclusively by broadcasting messages. We also discuss, via some examples, how to use this language to model LAN protocols. We hope that by presenting only a minimal language for defining LAN protocols, many potential users are encouraged to learn it, then use it in modeling and verifying their LAN protocols.

We also discuss a technique to verify the safety and liveness properties of networks of broadcasting machines. The technique consists of constructing a small directed graph, called an aggregate reachability graph, to represent the large state space of the network being verified, then checking some conditions to establish its safety and liveness properties. Aggregate reachability graphs, proposed earlier [10] to verify the safety and liveness properties of networks of communicating finite state machines, will thus be extended to be applicable to networks of broadcasting machines.

The paper is organized as follows. In Section 2, we present the syntax of our simple language for networks of broadcasting machines. The semantics of the language is presented in Section 3. In Section 4, we define aggregate reachability graphs, and discuss their characteristics. Then in Section 5, we discuss how to use aggregate reachability graphs to verify the safety and liveness properties of broadcasting networks. In Section 6, we use our language and verification technique to model and verify a CSMA/CD protocol [1].

2. NETWORKS OF BROADCASTING FINITE STATE MACHINES

In this section we define a simple language, called *Broadcasting Finite State Machines*, that can be used in modeling LAN protocols. The simplicity of this language is intended to make it easy to learn, but still allow it to be used conveniently in modeling a variety of LAN protocols. On the other hand, one should be prepared to resort to *simulation* when modeling protocol features that don't have closely corresponding features in the language. This will become apparent as we discuss how to use the language to model some protocol examples in later sections. The language consists of two basic concepts: networks and machines; they are discussed in detail next.

*This work was supported in part by the contract B6C3007 from Rockwell International Corporation.

CH2284-8/86/0000/0058 $1.00 © 1986 IEEE

A *network* is a tuple $[M_1,...,M_r]$ of r broadcasting finite state machines, to be defined shortly. For all practical purposes, we assume that each network has at least two machines, i.e. $r>1$.

A *broadcasting finite state machine* M has a finite number of *states* and state transitions. One of the states of M is identified as its *initial state*. Each *transition* t of M is defined as a triple

[m,c,m']

where m is a state of M called the *present state* of t,
m' is a state of M called the *next state* of t, and
c is called the *action* of t
and has one of the following formats

-g where "-" is a reserved symbol and g is a message type. In this case, the transition is called *sending*, and the action reads send a message of type g (over the bus).

+g where "+" is a reserved symbol and g is a message type. In this case, the transition is called *receiving*, and the action reads receive a message of type g (from over the bus).

I, T, or C where I, T, and C are reserved symbols. In these cases the transition is called *idleness-*, *transmission-*, or *collision-detection*, and the action reads detect idleness, transmission, or collision (over the bus), respectively.

A transition t of M is said to be *ingoing* into some state m of M iff the next state of t is m. Transition t is said to be *outgoing* from state m iff the present state of t is m.

In general, we assume that M is *nondeterministic*, i.e. outgoing transitions from the same state of M can have identical actions.

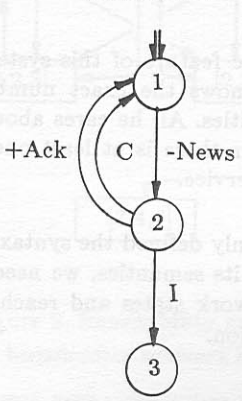
States of M are distinguished into three types: sending, receiving and terminating. A *sending state* has at least one outgoing transition, and each of its outgoing transitions is sending. A *receiving state* has at least one outgoing transition, and none of its outgoing transitions is sending. A *terminating state* has no outgoing transitions.

It is convenient to represent each broadcasting finite state machine M as a directed graph whose nodes represent the machine states and whose edges represent its transitions. More specifically, each state m of M is represented as a node, also called m for convenience, and each transition [m,c,m'] of M is represented as a directed edge, labeled c, from node m to node m'.

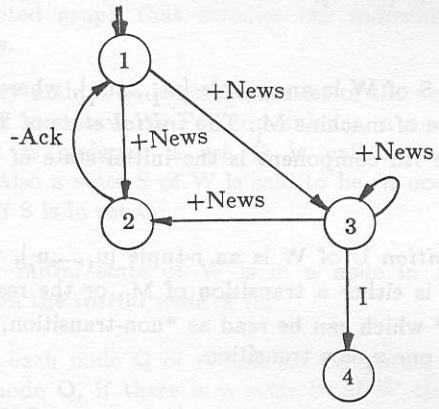
Example 1:

Consider a system of one broadcaster and a number of listeners. The broadcaster broadcasts news items to the listeners so long as one or more of them are showing their appreciation by acknowledging each broadcast. If one broadcast goes unacknowledged, the broadcaster terminates the service. This system can be represented as a network $[B,L_1,...,L_r]$, where B models the broadcaster and L_i models a listener.

Machine B is shown in Figure 1a. Starting at its initial sending state, state 1, B sends one news item, then proceeds to a receiving state, state 2, to wait for a response. If it receives an acknowledgement or if it detects a collision (which means that two or more listeners are sending their acknowledgements simultaneously), then it returns to state 1 and the cycle



(a) Broadcaster B



(b) Listner L_i (i=1,...,r)

Figure 1. A broadcaster network $[B,L_1,...,L_r]$.

repeats. On the other hand, if it detects an idleness (which means that no listener is sending an acknowledgement), it proceeds to its terminating state, state 3.

Machine L_i is shown in Figure 1b. At its initial state, L_i waits to receive a news item, then decides (supposedly depending on whether the received item is of interest or not) whether to proceed to state 2 or state 3. At state 2, L_i sends back an acknowledgement, returns to state 1 to receive the next item, and the cycle repeats. At state 3, L_i does not acknowledge the received news items (supposedly hoping that they will be stopped soon). However if it receives an interesting item at state 3, it can proceed to state 2, return an acknowledgement, then wait for the next item. Finally if at state 3 L_i detects idleness, it proceeds to its terminating state, state 4.

One characteristic feature of this system is that the broadcaster never knows the exact number of his listeners or their identities. All he cares about to continue his service is whether there is at least one listener who is interested in the service. □

So far we have only defined the syntax of our simple language. To define its semantics, we need to introduce the concepts of network states and reachability graphs in the following section.

3. NETWORK STATES AND REACHABILITY GRAPHS

Let $W=[M_1,...,M_r]$ be a network of r broadcasting machines.

A *state* S of W is an r-tuple $[m_1,...,m_r]$, where each m_i is a state of machine M_i. The *initial state* of W is a state whose ith component is the initial state of M_i in W.

A *transition* U of W is an r-tuple $[u_1,...,u_r]$, where (a) each u_i is either a transition of M_i, or the reserved symbol "*" which can be read as "non-transition," and (b) at least one u_i is a transition.

Transition U is said to be *enabled at* state S iff each transition u_i in U is outgoing from state m_i in S, and one of the following three conditions is satisfied:

i. No state in S is sending, and each transition in U is idleness-detection. Moreover for each non-transition u_i in U, state m_i has no outgoing idleness-detection transition.

ii. Only one state in S is sending, and one transition in U is sending labeled -g, and each other transition in U is either receiving labeled +g or transmission-detection whose present state has no outgoing receiving transition labeled +g. Moreover for each non-transition u_i in U, state m_i has no outgoing transition that is either receiving labeled +g or transmission-detection.

iii. Two or more states in S are sending, and each transition in U is either sending or collision-detection. Moreover for each non-transition u_i in U, state m_i has no outgoing sending or collision-detection transition.

The three conditions, i, ii, and iii in the above definition need some explanation. Condition i means that all the machines in the network that are capable of detecting idleness at some network state will indeed do so iff no machine in the network is capable of sending at this state. Condition ii means that a machine that is capable of sending at some network state will do so *without collision* iff no other machine is capable of sending at the same state. Moreover, as this machine sends its message, every other machine will (a) receive the message, (b) detect the transmission, or (c) do nothing if it is capable of neither at this state. If a machine is capable of both receiving the message and detecting the transmission at this state, it will receive the message. Condition iii means that all machines that are capable of sending at some network state will do so *with collision* iff there are two or more of them. Each other machine that is capable of detecting a collision at this state will do so.

Notice that these conditions imply that at any network state all the network machines that are capable of executing transitions, will do so at the same instant. Thus their execution is *synchronous*.

Let $S=[m_1,...,m_r]$ and $S'=[n_1,...,n_r]$ be two states of W, and let $U=[u_1,...,u_r]$ be a transition of W. S' is said to *follow* S *over* U iff (a) U is enabled at S, (b) for every transition u_i in U, u_i is outgoing from state m_i and ingoing to state n_i, and (c) for every nontransition u_i in U, m_i and n_i are the same state in M_i.

A state S of W is said to be *reachable* iff there is a sequence of network states $S_1,...,S_k$ such that (a) S_1 is the initial state of W, (b) for i=1,...,k-1, S_{i+1} follows S_i over some network transition, and (c) $S=S_k$.

A *communication sequence* of W is an infinite sequence of reachable states $S_1, S_2, \ldots$ such that (a) S_1 is the initial state of W, and (b) for i=1,2,..., S_{i+1} follows S_i.

A *reachability graph* G of W is a directed graph where (a) for each reachable state S of W there is a node, also called S for convenience, in G, and (b) for each pair of reachable states S and S' of W, if S' follows S over some network transition, then there is a directed edge from node S to node S' in G.

The reachability graph of a network can be used for reasoning about the communication properties of the network as illustrated by the following example.

Example 2:

Consider the broadcaster network discussed earlier. Figure 2 shows the reachability graph of this network in the special case where there are only two listeners, i.e. network $[B,L_1,L_2]$. This graph shows the eight reachable states of the network and the network transitions between them. Examining this reachability graph shows that the network satisfies the following three desirable properties:

i. *Service continues only if needed:* Whenever B is at state 1 (ready to send the next news item), at least one L_i is at state 1 (willing to receive the next news item).

ii. *Service terminates only if not needed:* Whenever B is at its terminating state, every L_i is at its terminating state.

iii. *Proper termination:* Whenever all the L_i's are at their terminating states, B is also at its terminating state. □

In general the number of reachable states of a network is finite. However this number is usually very large, and can increase very rapidly as the number of machines in the network increases; see example 3 below. This suggests that reachability graphs are usually very large, and so are not convenient to be used for verifying the communication properties of networks with large numbers of machines. For verifying such networks, we use "Aggregate Reachability Graphs" which are discussed in the next section.

Example 3:

The reachability graph in Figure 2 has only eight nodes or network states; but recall that its network has only two listeners. The reachability graph for a network with 10 listeners has at least 1024 nodes or states. In general, the reachability graph for a broadcaster network with r listeners has at least 2^r nodes (since there are that many reachable network states where the broadcaster B is at state 2 and each listener L_i is at state 2 or state 3). Therefore, the size of the reachability graph increases exponentially with the number of listeners in the network, and the reachability graph is not convenient to be used in proving the communication properties of a network with a large number of listeners. □

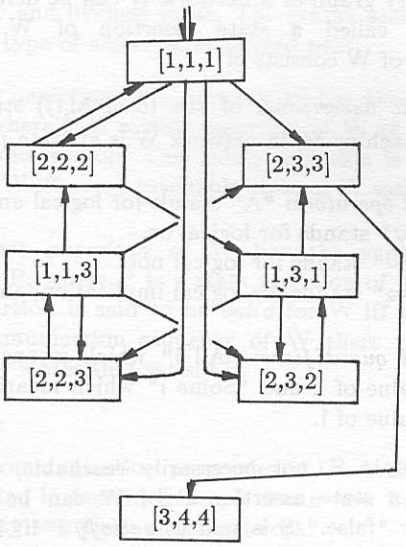


Figure 2. Reachability graph of the broadcaster network $[B,L_1,L_2]$

4. AGGREGATE REACHABILITY GRAPHS

An *aggregate reachability graph* A of a network W is a directed graph that satisfies the following three conditions.

i. Every node in A is assigned a set of the network states of W. For convenience, a node that is assigned a set Q is called *node* Q. Also a state S of W is said to be *in* node Q iff S is in set Q.

ii. The initial state of W is in a node in A, called the *initial node* of A.

iii. For each node Q of A and for each state S in node Q, if there is a state S' of W that follows S over some network transition, then there is a node Q' in A such that (a) S' is in node Q', and (b) there is a directed edge from node Q to node Q' in A.

iv. For each node Q that has at least one outgoing edge in A, and for each state S in node Q, there exists a state of W that follows S over some network transition.

Condition iv implies that no "deadlock" or "terminating" state is in a node that has outgoing edges in an aggregate reachability graph. Such states can only be in nodes without outgoing edges. As will become apparent shortly, condition vi is included to allow us to state and verify Theorem 2 below.

The set of states assigned to a node in an aggregate reachability graph of a network W can be defined by an assertion, called a state assertion of W. A *state assertion* of W consists of

i. *atomic assertions*: of the form $M_i(j)$ stating that machine M_i in network W is at state j.

ii. *logical operators*: "$\wedge$" stands for logical and,
"$\vee$" stands for logical or,
"$\sim$" stands for logical not,
"$\Rightarrow$" stands for logical implication, and

iii. *logical quantifiers*: "All i" which means for every value of i, and "Some i" which means for some value of i.

At any state S, not necessarily reachable, of W, the value of a state assertion P of W can be computed "true" or "false." S is said to *satisfy* P iff P=true at S.

Example 4:

Figure 3 shows an aggregate reachability graph A of the broadcaster network $W=[B,L_1,...L_r]$ discussed earlier. The set of states assigned to any node in A is defined by a state assertion written inside that node. For example the set of the states assigned to the initial node of A is defined by the assertion:

$B(1)\wedge[\text{Some i:i=1,...,r:}L_i(1)]\wedge[\text{All i:i=1,...,r:}\sim L_i(2)]$,

where B(1) states that machine B is at state 1,
$L_i(j)$ states that machine L_i is at state j,
"$\wedge$" is the logical "and" operator, and
"$\sim$" is the logical "not" operator.

This assertion consists of three parts. The first part states that the broadcaster is at state 1, the second part states that at least one of the listeners is at state 1, and the third part states that none of them is at state 2. Notice that the initial state of W satisfies this assertion; this explains why its node is the initial node of A. Similarly, the state assertion $B(3)\wedge[\text{All i:i=1,...,r:}L_i(4)]$ is written inside the node that has no outgoing edges in A to indicate that the singleton set {[3,4,...,4]} is assigned to this node. □

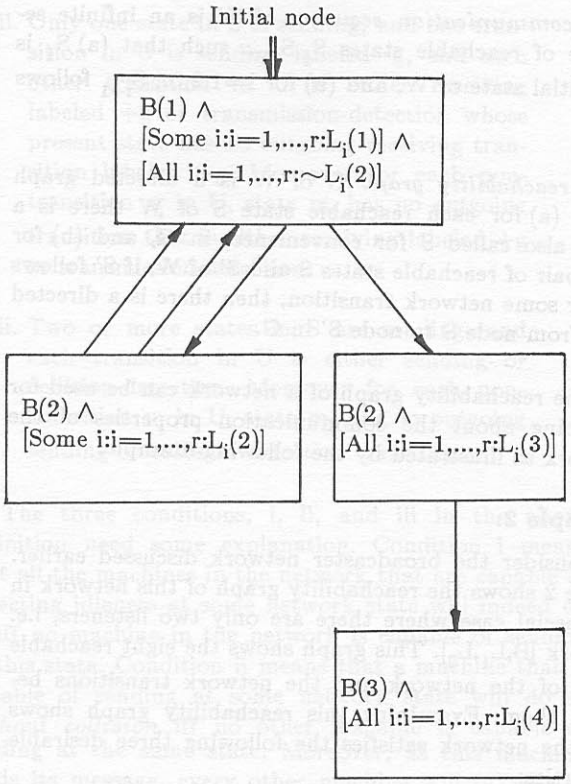


Figure 3. An aggregate reachability graph for the broadcaster network.

Some comments concerning aggregate reachability graphs are in order. First, one can view the reachability graph of a network as a special aggregate reachability graph, where the set of states assigned to each node is a singleton. Second, a network can in general have many aggregate reachability graphs not necessarily isomorphic to one another. For example, Figure 4 shows another aggregate reachability graph, for the broadcaster network in Figure 1, that is not isomorphic to its aggregate reachability graph in Figure 3. Third, the next two theorems state two properties of aggregate reachability graphs that make them, as discussed in the next section, useful for verifying the communication properties of their networks.

Theorem 1:

Let A be an aggregate reachability graph of a network W. Every reachable state of W is in some node in A. The converse is not necessarily true, i.e. not every state that is in some node in A is necessarily reachable.

Proof:

Let S be a reachable state of W. Then, there is a sequence $S_1,...,S_k$ of the states of W such that (a) S_1 is the initial state of W, (b) $S=S_k$, and (c) for i=1,...,k-1, S_{i+1} follows S_i over some network transition. From the

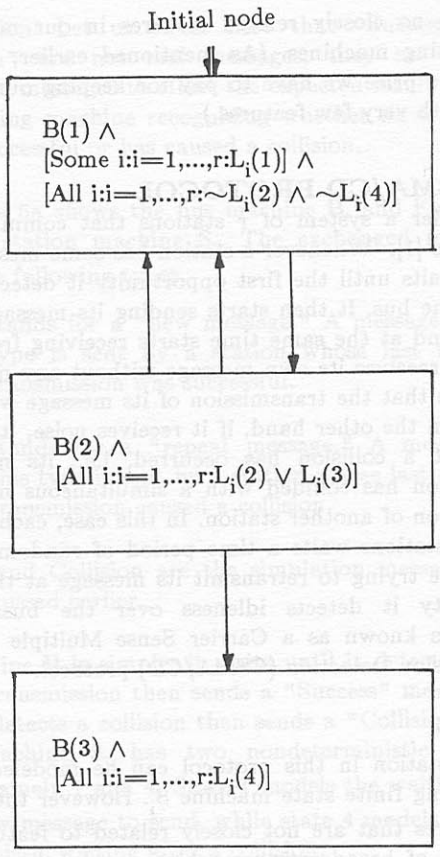


Figure 4. Another aggregate reachability graph for the broadcaster network.

second condition of an aggregate reachability graph, the initial state S_1 must be in the initial node of A. From the third condition of an aggregate reachability graph, S_2 must be in a node that follows the initial node of A. This argument can be repeated k-2 times yielding that S_k or S must be in some node in A.

To show that the converse is not necessarily true, it is sufficient to give a counterexample. The state [1,1,...,1,4] of network [B,L_1,...,L_r] is in the initial node of the aggregate reachability graph in Figure 1d, even though it is not reachable. □

Theorem 2:

Let A be an aggregate reachability graph of a network W. For every communication sequence $S_1,S_2,...$ of W, there exists an infinite directed path with nodes $Q_1,Q_2,...$ in A such that Q_1 is the initial node of A, and for i=1,2,... S_i is in Q_i. The converse is not necessarily true.

Proof: is similar to that of Theorem 1 using condition iv in the definition of aggregate reachability graphs. □

In the next section we discuss how to use aggregate reachability graphs to verify the communication properties of their networks.

5. NETWORK PROPERTIES AND THEIR VERIFICATION

The communication properties of a network can be defined by assertions. There are two types of assertions, called safety and liveness in [9]. The syntax and semantics of each type of assertion is defined next.

A *safety assertion* of a network W has the syntax: Always P, where P is a state assertion of W as defined in the previous section. This safety assertion is said to be *valid* for W iff every reachable state of W satisfies P.

A *liveness assertion* of W has the syntax: Infinitelyoften P, where P is a state assertion of W. This liveness assertion is said to be *valid* for W iff in every infinite communication sequence of W there exist infinitely many states that satisfy P.

Example 5:

All the communication properties of the broadcaster network stated in text in Example 2 can now be stated as valid safety assertions of the network.

i. *Service continues only if needed:*
F_1::Always [B(1)⇒[Some i:i=1,...,r:L_i(1)]]

ii. *Service terminates only if not needed:*
F_2::Always [B(3)⇒[All i:i=1,...,r:L_i(4)]]

iii. *Proper termination:*
F_3::Always [[All i:i=1,...,r:L_i(4)]⇒ B(3)]

Similarly, a liveness property of this network can be stated as the following valid liveness assertion of the network.

V :: Infinitelyoften [B(1)∧[Some i:i=1,...,r:L_i(1)]] □

The following theorem follows directly from Theorems 1 and 2 and the definition of "an assertion being valid for a network;" it shows how to use an aggregate reachability graph of a network to verify that a safety or liveness assertion of the network is valid for the network.

Theorem 3:

Let A be an aggregate reachability graph of a network W, and let P be a state assertion of W. Assume that A has k nodes $Q_1,...,Q_k$ with state assertions $P_1,...,P_k$, respectively.

i. If for i=1,...,k, $P_i \Rightarrow P$
then the safety assertion "Always P" is valid for W.

ii. If every directed cycle in A has a node Q_i whose state assertion P_i implies P (i.e. $P_i \Rightarrow P$) then the liveness assertion "Infinitelyoften P" is valid for W. □

Example 6:

It is straightforward to use Theorem 3 and the aggregate reachability graph A in Figure 3 to establish that the four assertions F_1, F_2, F_3, and V in Example 5 are valid for the broadcaster network in Figure 1.

For example, since each of the state assertions of the nodes in A implies the state assertion

$$P :: [B(1) \Rightarrow [\text{Some } i{:}i=1,...,r{:}L_i(1)]]$$

and since $F_1 :: \text{Always } P$

then by Theorem 3 part i, F_1 is valid for the network.

Similarly, since each directed cycle in A contains the initial node of A whose state assertion implies

$$P' :: [B(1) \wedge [\text{Some } i{:}i=1,...,r{:}L_i(1)]]$$

and since $V :: \text{Infinitelyoften } P'$

then by Theorem 3 part ii, V is valid for the network. □

Not every valid assertion of a network W can be proved valid using Theorem 3 and any aggregate reachability graph of W. This is better demonstrated by an example.

Example 7:

The safety assertion

Always $[B(1) \Rightarrow [\text{All } i{:}i=1,...,r{:} \sim L_i(4)]]$

for the broadcaster network in Figure 1 states that whenever the broadcaster is at state 1 then no listener can be at state 4. Using Theorem 3, one cannot establish the validity of this assertion from the aggregate reachability graph in Figure 3; however one can establish its validity from the aggregate reachability graph in Figure 4. □

In the next section, we discuss how to model a practical LAN protocol example for CSMA/CD as a network of broadcasting machines. The objective of this exercise is to demonstrate the effectiveness of *simulation* in modeling protocol features for which there are no closely related features in our model of broadcasting machines. (As mentioned earlier, simulation is the price we have to pay for keeping our model simple with very few features.)

6. A CSMA/CD PROTOCOL

Consider a system of r stations that communicate over a bus [1]. Whenever a station has some message to send it waits until the first opportunity it detects idleness on the bus. It then starts sending its message over the bus and at the same time starts receiving from the bus. If it receives its own message without any noise, it recognizes that the transmission of its message was successful. On the other hand, if it receives noise, it recognizes that a collision has occurred, i.e., its message transmission has collided with a simultaneous message transmission of another station. In this case, each of the colliding stations waits a time period of random duration before trying to retransmit its message at the first opportunity it detects idleness over the bus. This protocol is known as a Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol.

Modeling

Each station in this protocol can be modeled as a broadcasting finite state machine S_i. However there are two features that are not closely related to features in our model of broadcasting machines, and so they need to be simulated. They are:

i. the ability of each machine to wait a random time period before trying to retransmit its message at the earliest opportunity it detects idleness, and

ii. the ability of each machine to send and receive simultaneously.

The first feature can be simulated using nondeterminism. In particular, the waiting state of the machine is provided with two outgoing idleness-detection transmissions. One of them returns to the waiting state itself (i.e. it forms a self-loop in graphical notation), and the other leads to a retransmission state. Thus at this waiting state the machine can detect idleness an arbitrary number of times before deciding to retransmit. The reason that this waiting construct is a mere simulation is that in the real situation the station does not try to detect idleness while it waits; idleness detection resumes only after the random waiting period has expired.

The second feature can be simulated by defining one broadcasting finite state machine to model the bus. This machine observes whether each message transmission is successful or has caused a collision, then broadcasts a message of type "Success" or "Collision" to inform the transmitting station(s) of the outcome of the

transmission. Let us stress here that "Success" and "Collision" are not real messages; they are merely simulation messages to effect the expected result of each transmitting machine recognizing whether its transmission is successful or has caused a collision.

Figure 5a shows the bus machine B, and Figure 5b shows a station machine S_i. The exchanged messages are of the following types.

Nmsg stands for a "new message." A message of this type is sent by a station whose last message transmission was successful.

Rmsg stands for a "repeat message." A message of this type is sent by a station whose last message transmission caused a collision.

Success and Collision are the simulation messages discussed earlier.

Machine B is simple; it waits until it detects a successful transmission then sends a "Success" message, or until it detects a collision then sends a "Collision" message. Machine S_i has two nondeterministic waiting states, namely 1 and 4. State 1 models the waiting of S_i for a new message to send, while state 4 models its nondeterministic waiting after a collision.

Verification

Figure 5c shows an aggregate reachability graph for the network $[B,S_1,...,S_r]$. It has five nodes whose state assertions are P_1 to P_5 defined as follows:

P_1 :: $B(1) \wedge [\text{All } i{:}i=1,...,r{:}S_i(1) \vee S_i(4)]$

P_2 :: $B(1) \wedge [\text{Some } i{:}i=1,...,r{:}[S_i(2) \vee S_i(5)] \wedge [\text{All } j{:}j=1,...,r{:}j \neq i \Rightarrow S_j(1) \vee S_j(4)]]$

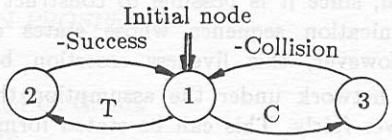
P_3 :: $B(2) \wedge [\text{Some } i{:}i=1,...,r{:}[S_i(3)] \wedge [\text{All } j{:}j=1,...,r{:}j \neq i \Rightarrow S_i(1) \vee S_i(4)]]$

P_4 :: $B(1) \wedge [\text{Some } i,j{:}i=1,...,r,j=1,...,r{:}[i \neq j] \wedge [S_i(2) \vee S_i(5)] \wedge [S_j(2) \vee S_j(5)] \wedge [\text{All } k{:}k=1,...,r{:}S_k(1) \vee S_k(2) \vee S_k(4) \vee S_k(5)]]$

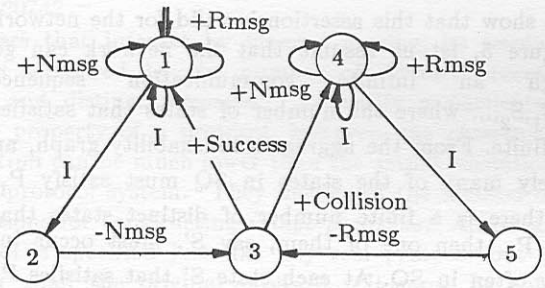
P_5 :: $B(3) \wedge [\text{Some } i,j{:}i=1,...,r,j=1,...,r{:}[i \neq j] \wedge [S_i(3)] \wedge [S_j(3)] \wedge [\text{All } k{:}k=1,...,r{:}S_k(1) \vee S_k(4) \vee S_k(3)]]$

Using this aggregate reachability graph and Theorem 3, it is straightforward to verify that the following safety assertions are valid for the network.

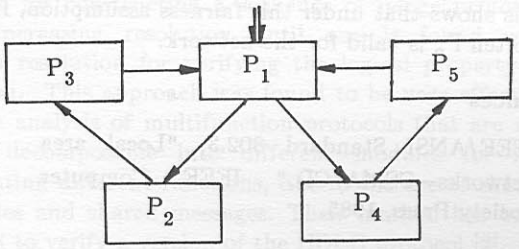
i. *Machine B never causes a collision:*
Always $[B(2) \vee B(3) \Rightarrow [\text{All } i{:}i=1,...,r{:} \sim S_i(2) \wedge \sim S_i(5)]]$



(a) Bus B



(b) Station S_i (i=1,...,r)



(c) Aggregate reachability graph

Figure 5. A CSMA/CD access protocol.

ii. *Whenever B is ready to send "Success," exactly one station has just sent a message:*
Always $[B(2) \Rightarrow [\text{Some } i{:}i=1,...,r{:}[S_i(3) \wedge [\text{All } j{:}j=1,...,r{:}j \neq i \Rightarrow S_j(1) \vee S_j(4)]]]]$

iii. *Whenever B is ready to send "Collision," two or more stations have just sent messages:*
Always $[B(3) \Rightarrow [\text{Some } i,j{:}i=1,...,r,\ j=1,...,r{:}i \neq j \wedge S_i(3) \wedge S_j(3)]]$

To establish the liveness of the network, we need to prove that infinitely often one station gets to send successfully, i.e. verify that the liveness assertion Infinitelyoften P_2 is valid for the network. This is not

true in general, since it is possible to construct an infinite communication sequence whose states do not satisfy P_2. However, this liveness assertion becomes valid for the network under the assumption that the network behaves fairly. This can be stated formally as follows

If every network transition that is enabled infinitely often is executed infinitely often
then Infinitelyoften P_2.

To show that this assertion is valid for the network in Figure 5, let us assume that the network can go through an infinite communication sequence $SQ=S_1,S_2,...$ where the number of states that satisfies P_2 is finite. From the aggregate reachability graph, an infinitely many of the states in SQ must satisfy P_1. Since there is a finite number of distinct states that satisfy P_1, then one of them, say S', must occur infinitely often in SQ. At each state S' that satisfies P_1 there is an enabled network transition U' such that S" follows S' over U' and S" satisfies P_2. Let U be such a transition for state S. Thus, U is enabled infinitely often along SQ but is executed a finite number of times, i.e. SQ does not satisfy the fairness assumption stated earlier. This shows that under this fairness assumption, Infinitelyoften P_2 is valid for the network.

References

[1] IEEE/ANSI Standard 802.3, "Local area networks–CSMA/CD," IEEE Computer Society Press, 1985.

[2] IEEE/ANSI Standard 802.4, "Local area networks/token passing bus," IEEE Computer Society Press, 1985.

[3] IEEE/ANSI Standard 802.5, "Local area networks/token ring access method," IEEE Computer Society Press, 1985.

[4] C.A.R. Hoare, "Communicating sequential processes," *Prentice-Hall*, 1985.

[5] Robin Milner, "A calculus of communicating systems," *Lecture Notes in Computer Science* 92, Springer-Verlag, Berlin, 1980.

[6] G. V. Bochmann, "Finite state description of communication protocols," *Computer Networks*, Vol. 2, No. 4/5, Oct. 1978.

[7] D. Brand, and P. Zafiropulo, "On communicating finite-state machines," *JACM*, Vol. 30, No.2, April 1983, pp. 323-342.

[8] M. G. Gouda, "Closed covers: to verify progress for communicating finite state machines," *IEEE Trans. on Software Engineering*, Vol. SE-10, No. 6, Nov. 1984, pp. 846-855.

[9] S. Owicki, and L. Lamport, "Proving liveness properties of concurrent programs," *ACM Trans. on Programming Languages and Systems*, Vol. 4, No. 3, July 1982.

[10] N. Multari, M. G. Gouda, and S. S. Lam, "Add-on protocols: their modeling and verification," in preparation, 1985.

[11] K. Sabnani and M. Schwartz, "Verification of a multidestination protocol using temporal logic," *Proc. of the Second International Workshop on Protocol Specification, Testing, and Verification*, C. Sunshine (editor), North-Holland, 1982.