MODELING AND ANALYSIS OF LAN PROTOCOLS USING LABELED PETRINETS

Mohamed G. Gouda and Michael K. Molloy

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

TR-84-15 September 1984

ABSTRACT

Modeling and analysis of a Local Area Network (LAN) protocol can be "tricky". It requires one to define and reason about the communications among n identical but autonomous stations in the LAN, where n is a parameter that can assume any positive integer value. In this paper, we demonstrate that the labeled Petrinets of Keller [5] are useful for this task. In particular, we use these Petrinets to model and analyze the LAN protocols, for a token ring, nonpersistent CSMA, virtual time CSMA, P-persistent CSMA and Capetanakis' collision resolution algorithm. In each case, the resulting model of the protocol is clear and concise, and the analysis of its safety and liveness properties is straightforward.

Table of Contents

1. INTRODUCTION

A Local Area Network (LAN) consists of n autonomous *stations* that communicate by exchanging messages over a single *communication medium*, as shown in Figure 1. If at some instant, exactly one station starts to transmit a message over the medium, and if no other station tries to transmit, then this transmission will complete *successfully*. If two or more stations start to transmit over the medium, then a *collision* will occur. A collision occurrence can be detected by all the stations in the LAN, and will cause the colliding transmissions to be tried later by their stations until they complete successfully. A *LAN protocol* is a set of rules that should be observed by each station in the LAN to *prevent*, *avoid*, or *resolve* collisions as the stations try to transmit over the medium.

Modeling and analysis of a LAN protocol involving n stations can be a "tricky" task, especially since the protocol's analysis needs to be carried out for any positive integer value of n. The straightforward approach to solve this problem is as follows: First, the LAN protocol under consideration is modeled by defining the local states and state transitions of a typical station in it. Second, the protocol's model is analyzed by using the local states and state transitions of individual stations to reason about the global states and state transitions of the total protocol. The problem with this approach is that the number of global states and state transitions of any reasonable protocol is "huge"; this complicates the analysis task by a large degree. Sasha, Penueli, and Ewald [11] followed this approach to model and analyze some CSMA protocols using temporal logic. However they assumed that all stations progress at equal speeds, thus reducing the total number of reachable global states and greatly simplifying the analysis task. The LAN protocol example in Gouda and Chang [4] has followed this same approach. But the complexity of the analysis was contained by limiting the analysis to a specific value for the parameter n (the number of stations in the LAN), rather than carry the analysis for any value of n.

In this paper, we propose a new approach to solve the problem. The LAN protocol under consideration is modeled by one labeled Petrinet, as defined by Keller [5], with a *fixed* number of nodes; i.e. the number of places and transitions in the modeling Petrinet does not depend on the number of stations in the protocol. The fact that there are n stations in the protocol is modeled by having n tokens in the Petrinet. The modeling Petrinet is relatively small, and so its safety and liveness properties can be easily established. We demonstrate the simplicity of this approach by using it to model and analyze some real LAN protocols. (For a complete exposition to the modeling and verification of communication protocols using Petrinet-based models, we refer the reader to the excellent survey by Diaz [3].)

The paper is organized as follows. The labeled Petrinets of Keller [5] are presented briefly in Section 2. The general approach of using these Petrinets to model and analyze LAN protocols is discussed in Section 3. In Section 4, we demonstrate this technique by

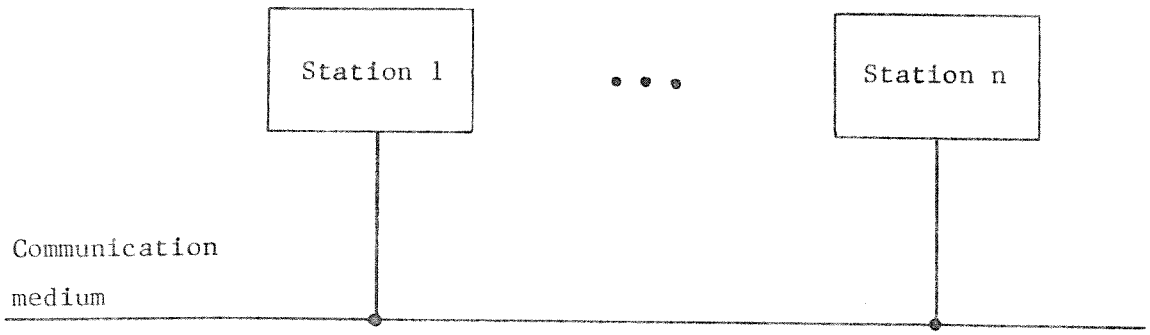


Figure 1 Architecture of a LAN

modeling and analyzing five LAN protocols: a token ring, a nonpersistent CSMA, a virtual-time CSMA, a P-persistent CSMA, and a collision resolution protocol. Concluding remarks concerning the advantages and disadvantages of this technique are in Section 5.

2. LABELED PETRINETS

A *labeled Petrinet* of Keller [5] is a pair (X,G), where

X is a vector of variables, and

G is a directed labeled graph with two types of nodes called *places* and *transitions*. Each arc in G is either from a place to a transition, or from a transition to a place. Each transition t in G is labeled with an expression of the form

when $P_t(X)$ **do** $X := F_t(X)$

where P_t is a predicate, and F_t is a function over the vector X. P_t is called the *enabling predicate* of t, and F_t is called the *action function* of t.

For the sake of this paper, each transition in G is restricted to having one ingoing arc and one outgoing arc. If there is an arc from place p to transition t then p is called the *input place* of t. If there is an arc from transition t to place p, then p is called the *output place* of t. We further restrict the model to the case where the input and output places of any transition t in G are distinct.

Let (X,G) be a labeled Petrinet. Associated with each place p in G is an implicit variable n_p, called the *place variable* of p, whose value ranges over the natural numbers. The place variables of G are distinct from the variables in vector X. (Referring to ordinary Petrinets [10], the value of a place variable n_p indicates at any instant the number of tokens at its place p at that instant.)

Let (X,G) be a labeled Petrinet, and let V be a vector of all the place variables of G. A *state* of (X,G) is a pair (x,v) where x and v are values for vectors X and V respectively. As suggested by Keller [5], it is possible to think of x as the "state of data" and of v as the "state of control" for the labeled Petrinet.

Let (x,v) and (x',v') be two states of a labeled Petrinet (X,G). State (x',v') is said to *follow* (x,v) *over* some transition t, denoted by $(x,v)-t\rightarrow(x',v')$, iff the following five conditions are satisfied: (Let n_i and n_o be the values of the place variables for the input and output places of transition t in v, and let n_i' and n_o' be their respective values in v'):

i. $n_i \geq 1$

ii. $n_i' = n_i - 1$

iii. $n_o' = n_o + 1$

iv. $P_t(x) = \text{true}$

v. $x' = F_t(x)$

Recall that P_t and F_t denote the enabling predicate and the action function of transition t.

Let (x,v) and (x',v') be two states of a labeled Petrinet (X,G), and let $t_1,t_2,...,t_k$ be transitions in G. State (x',v') is said to be reachable *from* (x,v) *over* the transition sequence $t_1t_2...t_k$, denoted by $(x,v)-t_1t_2...t_k\rightarrow(x',v')$, iff there exists a sequence of states $(x_1,v_1),...,(x_{k+1},v_{k+1})$ of (X,G) such that

i. $(x,v) = (x_1,v_1)$,

ii. $(x',v') = (x_{k+1},v_{k+1})$, and

iii. for $i=1,...,k$, $(x_i,v_i)-t_i\rightarrow(x_{i+1},v_{i+1})$.

A state (x',v') is said to be *reachable from* a state (x,v) iff either (x',v') = (x,v) or there exists a sequence of transitions $t_1t_2...t_k$ such that $(x,v)-t_1...t_k\rightarrow(x',v')$.

A state of a labeled Petrinet (X,G) is identified as its *initial state*, and any state of (X,G) is called *reachable* iff it is reachable from the initial state of (X,G).

In the next section, we discuss how to use labeled Petrinets to model and verify LAN protocols in general; specific examples are discussed in the following section.

3. MODELING AND ANALYSIS OF LAN PROTOCOLS

It is sometimes helpful to "visualize" that zero or more tokens are assigned to each place in the directed graph of a labeled Petrinet (as in the case of ordinary Petrinets [10]). At each instant, the number of tokens assigned to any place p equals the value of the place variable n_p of p at the same instant. In this view, the state changes of a labeled Petrinet can be considered as "moving" tokens between different places in the Petrinet. We use this view to explain how to model a LAN protocol between n identical stations using one labeled Petrinet (X,G):

i. Each state s of a typical station in the LAN is modeled as one place in G, also called s for convenience.

ii. Each state transition from a state s_1 to a state s_2 of a typical station in the LAN is modeled in G as one transition with one ingoing arc from place s_1, and one outgoing arc to place s_2.

iii. Each station is modeled by a single token. The fact that a station is at a state s at some instant is modeled by assigning the station's token to place s at the same instant. The fact that a station changes its state from s_1 to s_2 at some instant is modeled by moving the station's token from place s_1 to place s_2 in G.

iv. The variables in X and the enabling conditions and action functions for the transitions in G are then chosen to control the token movements in accordance with the modeled protocol.

Some examples of this modeling technique are discussed in the next section. But first, we discuss the analysis of LAN protocols modeled as labeled Petrinets. There are two types of properties that need to be proved for any LAN protocol:

i. **Safety Properties:** These properties assert that nothing bad happens during the protocol's execution. For example, they can assert *mutual exclusion* which can be stated as follows: Whenever one station is transmitting successfully over the communication medium, no other station can be transmitting successfully over the same medium. In our model, a safety property can be defined by an *invariant*, i.e. an always valid assertion, based on the variables in vector X and on the place variables. To show that such an assertion is indeed an invariant (i.e. always valid), it is sufficient to use the *induction principle* of Keller, and prove the following two properties [5]:

   (a) The assertion is valid at the initial state of the Petrinet.

   (b) If the assertion is valid at some state s and if state s' follows s over some transition, then the assertion is also valid at s'.

ii. **Liveness Properties:** These properties assert that at any instant during the protocol's execution, something good (i.e., a successful transmission by one station over the medium) can still happen at a subsequent instant. Although this notion of liveness is rather "weak" in general (since it does not guarantee that something good will ever happen), it seems adequate for most LAN protocols where indeed there is no guarantee that a successful transmission by one station will ever occur. To establish liveness in our model, it is sufficient to show that every transition t in the labeled Petrinet is "live" at any reachable state s_r of the Petrinet. In other words, show that for any transition t and any reachable state s_r, there exist two states s and s' reach-

able from s_r such that $s-t\rightarrow s'$. As suggested by Keller [5], this can be achieved by showing the following two properties:

(a) Each transition t in the labeled Petrinet is "live" at the initial state (i.e. show that for each transition t, there exist two reachable states s and s' such that $s-t\rightarrow s'$).

(b) From any reachable state s_r, the labeled Petrinet can still return to its initial state s_o (i.e. show that for any reachable state s_r, there exist a sequence of transitions $t_1,...,t_k$ such that $s_r-t_1...t_k\rightarrow s_o$).

Some LAN protocol examples are discussed next.

4. EXAMPLES OF LAN PROTOCOLS

In this section, we model and analyze five LAN protocols using labeled Petrinets. They are a token ring protocol, a nonpersistent CSMA protocol, a virtual time CSMA protocol, a P-persistent CSMA protocol and a collision resolution protocol. For these examples, we adopt the following notation: Places are named with upper case letters A,B,C,..., and each transition is named t_{ij} where i (j) is the name of its input (output) place.

4.1 A Token Ring Protocol

In this LAN protocol [1], stations circulate a special message (called the "token") in a round-robin fashion. Whenever a station receives the token, it can send at most one message to another station; it then sends the token to its next neighbor in the round-robin order, and so on.

A. Modeling

This protocol can be defined by the labeled Petrinet (X,G) in Figure 2. Vector X consists of three variables; s of type 0..n, and α and β of type boolean. (Recall that n is the number of stations in the protocol.) The directed graph G has three places A, B, and C that correspond to the three states of a station, namely "waiting for the token", "transmitting", and "waiting for the next round". G also has three transitions t_{AB}, t_{BC}, and t_{CA}. The initial state of this Petrinet is defined as follows:

$s=1$, $\alpha=$*false*, $\beta=$*false*, $n_A=n-1$, $n_B=1$, and $n_C=0$,

where n_p is the place variable of place p.

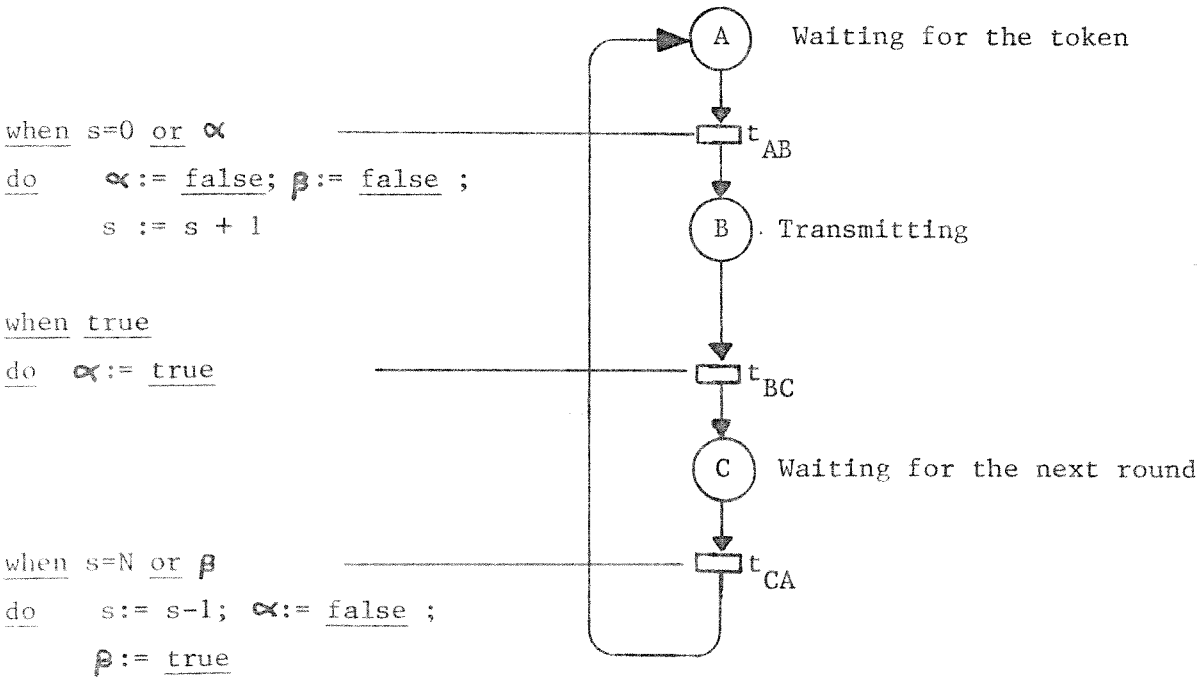


Figure 2 A labeled Petrinet that models
a token ring protocol

B. Analysis

a. Proving Safety Properties: It is straightforward to prove that the following assertions are invariants ($\Rightarrow$ denotes logical implication):

i. $n_A + n_B + n_C = n$ (Proof is by induction)

ii. $n_A + s = n$ (Proof is by induction)

iii. $n_B + n_C = s$ (Proof follows from i and ii)

iv. $n_B = 1 \Rightarrow s \geq 1$ (Proof follows from iii)

v. $n_B = 1 \Rightarrow \alpha = \mathit{false}$ (Proof is by induction making use of iv)

vi. $n_B \leq 1$ (Proof is by induction making use of iv and v)

vii. $\alpha = \mathit{false} \wedge \beta = \mathit{false} \Rightarrow n_B \geq 1$ (Proof is by induction)

Invariant vi establishes the mutual exclusion property of this protocol; invariant vii is needed in the proof of liveness, discussed next.

b. Proving Liveness Properties: The state of this Petrinet is defined as the tuple

$[s,\alpha,\beta,n_A,n_B,n_C]$,

and its initial state is s_o = [1, *false*, *false*, n-1, 1, 0]. It is straightforward to show that

$s_o - t_{BC}(t_{AB}t_{BC})^{n-1}\ t_{CA} \rightarrow s_r$,

where s_r is some reachable state, and $t_{BC}(t_{AB}t_{BC})^{n-1}\ t_{CA}$ is the sequence of transitions consisting of one occurrence of t_{BC}, followed by n-1 occurrences of $t_{AB}t_{BC}$, then followed by t_{CA}. Since this sequence of transitions contains at least one occurrence of each transition, each transition is live at the initial state s_o. It remains now to show that the initial state is reachable from any reachable state s_r.

Let $s_r = [s,\alpha,\beta,n_A,n_B,n_C]$ be any reachable state. There are two cases to consider:

Case 1 (n_B=0): From invariant vii, either α=*true* or β=*true*.
If α = *true*,
then $s_r - (t_{AB}t_{BC})^n A\ (t_{CA})^n (t_{AB}) \rightarrow s_o$ (the initial state).
If β = *true*,
then $s_r - (t_{CA})^n C\ (t_{AB}) \rightarrow s_o$.

Case 2 (n_B=1): In this case, $s_r - (t_{BC}t_{AB})^n A\ t_{BC}\ (t_{CA})^n (t_{AB}) \rightarrow s_o$.

This completes the proof that the initial state s_o is reachable from s_r.

As discussed earlier, this proof of liveness merely establishes that any transition in G can be made to fire starting from any reachable state s_r. However, this fact coupled with the fact that G is a simple cycle, guarantee that each transition in G will indeed fire infinitely often.

4.2 A Nonpersistent CSMA Protocol

Whenever a station in this protocol [12] has a message to send, it changes its state from "idle" to "trying", and checks the current state of the communication medium:

i. If the medium is *idle*, indicating that no other station is currently sending over the medium, then the station starts transmitting its message over the medium. This leads either to a successful transmission of the message, or to a collision if other stations start transmitting over the medium. In the case of a successful transmission, the station returns to an "idle" state. In the case of a collision, the station moves to a "waiting to try" state until at some later time when the medium becomes idle, the station returns to a "trying" state, and the cycle repeats.

ii. If the medium is *busy*, the station moves to a "waiting to try" state until at some later time when the medium becomes idle, the station returns to a "trying" state, and the cycle repeats.

A. Modeling

This protocol can be defined by the labeled Petrinet (X,G) in Figure 3. Vector X has four variables (r,t,s,c), each is of type 0..n, where n is the number of stations in the protocol. G has six places and eight transitions. Each place corresponds to a different state of a station:

Place I corresponds to the Idle state.
Place R corresponds to the Trying state.
Place T corresponds to the Transmitting state.
Place S corresponds to the Success state.
Place C corresponds to the Collision state.
Place W corresponds to the Waiting to try state.

As shown later, the values of variables r, t, s, and c at any instant indicate the numbers of tokens at places R, T, S, and C (respectively) at the same instant. The initial state of this Petrinet is as follows:

$$r = t = s = c = n_R = n_T = n_S = n_C = n_W = 0 \text{ and } n_I = n$$

This implies that initially all the stations are idle.

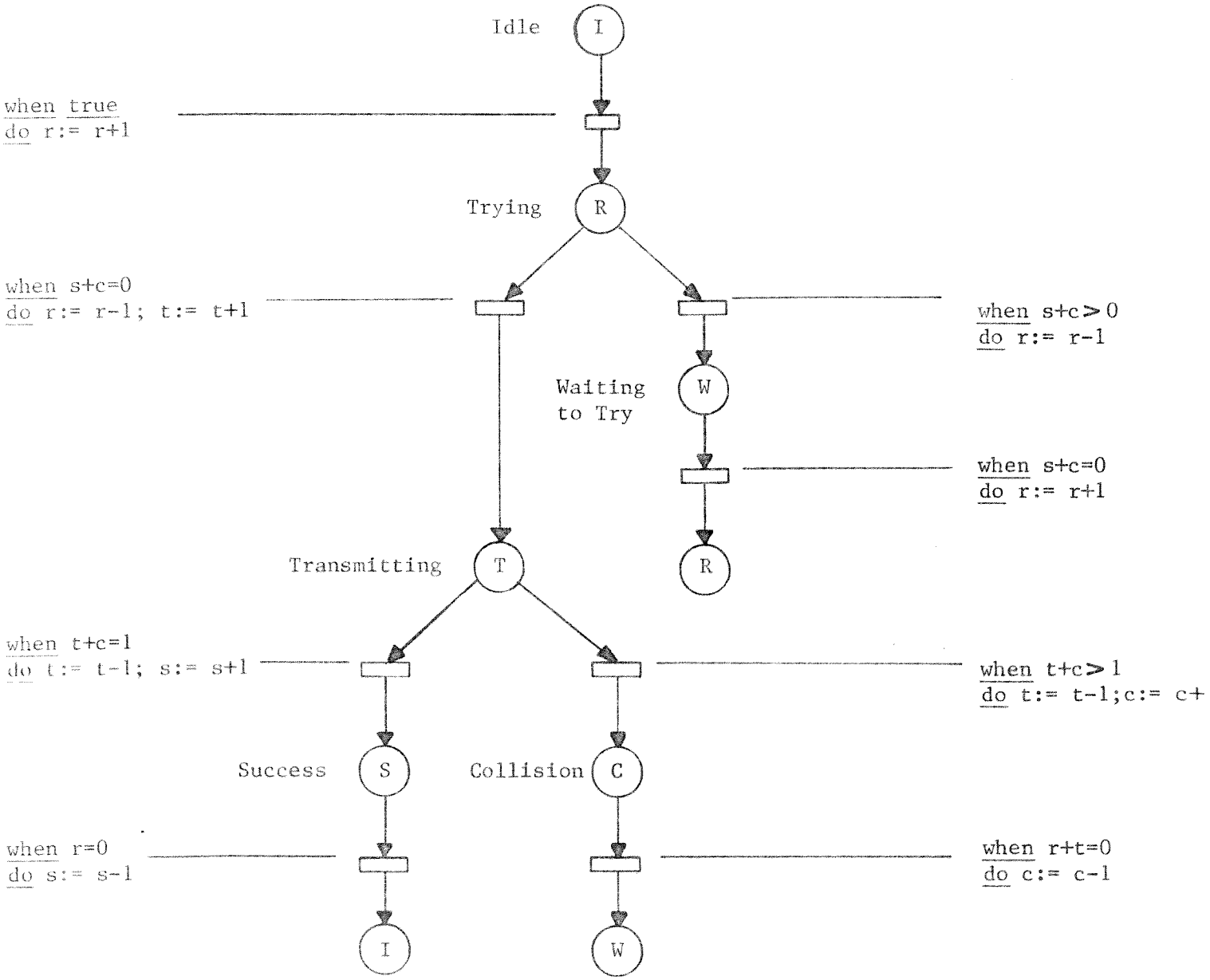


Figure 3 A labeled Petrinet that models
the nonpersistent CSMA protocol

B. Analysis

a. Proving Safety Properties: It is straightforward to prove that the following assertions are invariants:

i. $n_I + n_R + n_T + n_S + n_C + n_W = n$ (Proof is by induction)

ii. $r = n_R \wedge t = n_T \wedge s = n_S \wedge c = n_C$ (Proof is by induction)

iii. $s \geq 1 \Rightarrow t + c = 0$ (Proof is by induction)

iv. $s \leq 1 \wedge s.c = 0$ (Proof follows from iii)

Invariants iii and iv establish the mutual exclusion property of this protocol. Whenever one station is successfully transmitting, then at the same time no other station can be transmitting or colliding in its transmission (invariant iii), or even successfully transmitting (invariant iv).

b. Proving Liveness Properties: Because of invariant ii, the state of this Petrinet can be defined as:

$[r,t,s,c,n_I,n_W]$,

and its initial state is $s_0=[0,0,0,0,n,0]$. It is straightforward to show that

$$s_0 - t_{IR}\, t_{RT}\, t_{TS}\, t_{SI}\, (t_{IR})^3\, (t_{RT})^2\, (t_{TC})^2\, (t_{RW})\, (t_{CW})^2\, t_{WR} \rightarrow s_r,$$

where s_r is some reachable state. Since this sequence of transitions contains at least one occurrence of each transition, each transition is live at the initial state s_0. It remains now to show that the initial state is reachable from any reachable state.

Let $s_r=[r,t,s,c,n_I,n_W]$ be any reachable state. There are five cases to consider:

Case 1 (s=0 and t=0 and c=0):

$$s_r - t_{RT}\, t_{TS}\, (t_{RW})^{r-1}\, t_{SI}\, (t_{WR}\, t_{RT}\, t_{TS}\, t_{SI})^{r+n_W-1} \rightarrow s_0$$

Case 2 (s=0 and t=1 and c=0):

$$s_r - t_{TS}\, (t_{RW})^{r}\, t_{SI}\, (t_{WR}\, t_{RT}\, t_{TS}\, t_{SI})^{r+n_W} \rightarrow s_0$$

Case 3 (s=0 and t=0 and c=1):

$$s_r - (t_{RW})^{r}\, t_{CW}\, (t_{WR}\, t_{RT}\, t_{TS}\, t_{SI})^{r+n_W+1} \rightarrow s_0$$

Case 4 (s=0 and t + c > 1):

$$s_r - (t_{TC})^{t}\, (t_{RW})^{r}\, (t_{CW})^{t+c}\, (t_{WR}\, t_{RT}\, t_{TS}\, t_{SI})^{r+t+c+n_W} \rightarrow s_0$$

Case 5 (s=1): In this case, t=0 and c=0 from invariant iii. Hence,

$s_r - (t_{RW})^r\ (t_{SI})\ (t_{WR}\ t_{RT}\ t_{TS}\ t_{SI})^{r+n_W} \rightarrow s_0$

This completes the liveness proof for this protocol.

4.3 A Virtual Time CSMA Protocol

Each station in this protocol [8] has two clocks, one for real time, and one for virtual time. The virtual time clock progresses only when the communication medium is idle; therefore it is always lagging behind the real time clock. Whenever a station has a message to send, it changes its state from "idle" to "trying" and assigns the current value of its real time clock to the message. Whenever the value of the virtual time clock reaches the value assigned to the message, the station starts transmitting the message over the medium. (Notice that this can happen only when the medium is idle, since only then can the virtual time clock progress.) This message transmission leads to either a successful transmission or a collision. In case of a successful transmission the station returns to its "idle" state. In case of a collision, the station returns to its "trying" state and assigns the current value of its real time clock to the message and the cycle repeats.

A. Modeling

Figure 4 shows a labeled Petrinet (X,G) that models this protocol. Vector X has four variables (r,t,s,c), each of them is of type 0..n, where n is the number of stations in the protocol. G has five places I,R,T,S,C and six transitions. The initial state of this Petrinet is as follows:

$$r=t=s=c=n_R=n_T=n_S=n_C=0 \text{ and } n_I=n.$$

B. Analysis

a. Proving Safety Properties: It is straightforward to prove that the following assertions are invariants:

i. $n_I+n_R+n_T+n_S+n_C=n$ (Proof is by induction)

ii. $r=n_R \wedge t=n_T \wedge s=n_S \wedge c=n_C$ (Proof is by induction)

iii. $s \geq 1 \Rightarrow t+c=0$ (Proof is by induction)

iv. $s \leq 1 \wedge s.c=0$ (Proof follows from iii)

The mutual exclusion property of this protocol is established by invariants iii and iv.

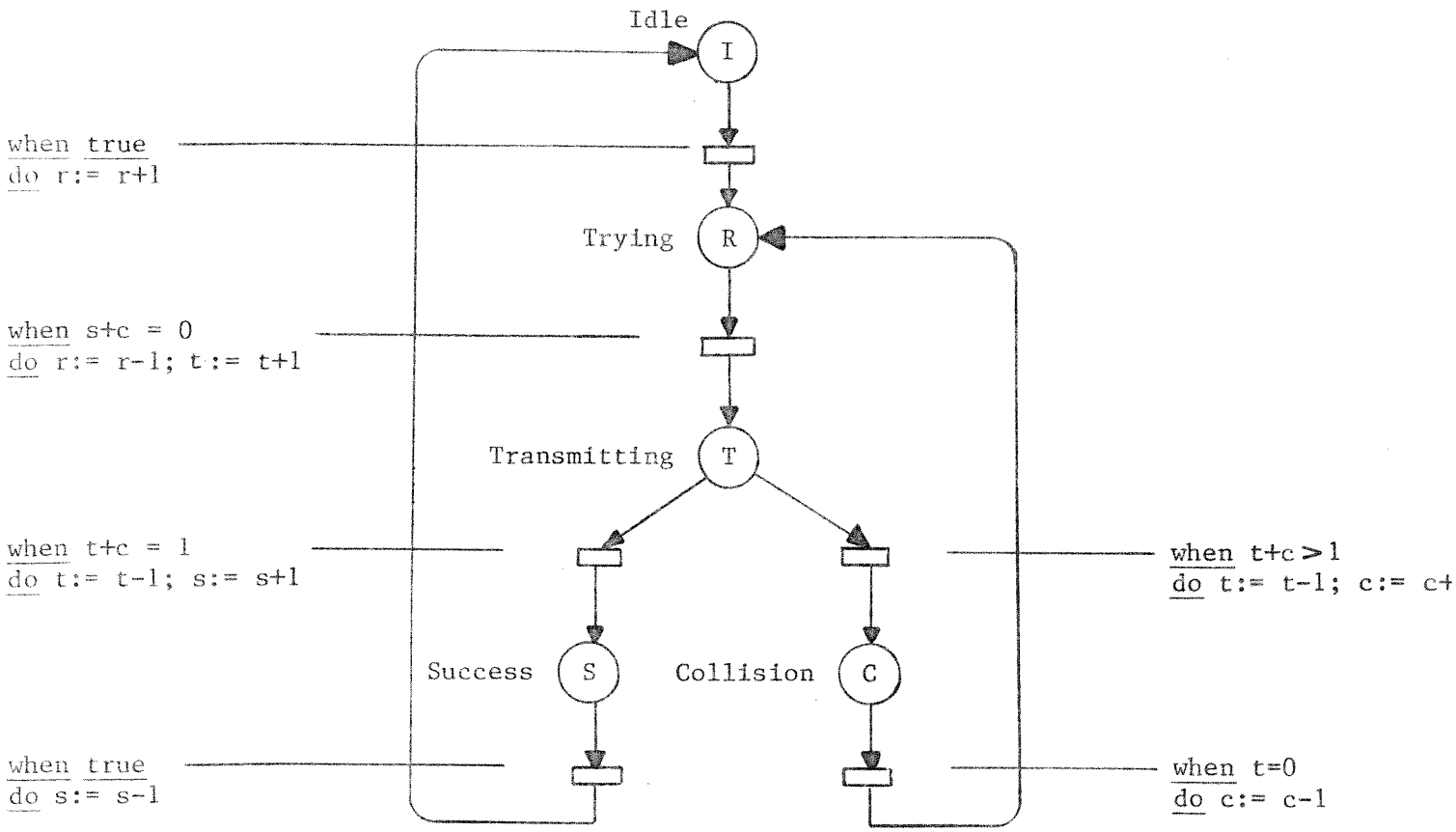


Figure 4 A labeled Petrinet that models the virtual time CSMA protocol

b. **Proving Liveness Properties:** This proof is similar to previous liveness proofs, and so we omit it.

4.4 A P-Persistent CSMA Protocol

In this protocol [7,12], whenever a station has a message to send and the medium is idle, the station changes its state from "not trying" to "flipping a coin." In the state of "flipping a coin," the station decides with probability P to transmit its message, and with probability 1-P to defer its transmission. If the station decides to defer and discovers shortly that the medium has remained idle, it reconsiders its decision with the same probability distribution. This continues until one of two outcomes occurs. (a) The station while in a "defer" state sees the medium become busy. (b) The station moves to a "transmitting" state. In the first case, the station enters a "waiting to try" state until at some later time when the medium becomes idle, it returns to the "flipping a coin" state. The second case leads either to a successful transmission by the station (in which case the station returns to its "not trying" state), or to a collision with other stations (in which case the station enters the "waiting to try" state until at some later time it returns to the "flipping a coin" state.)

A. Modeling

This protocol can be defined by the labeled Petrinet (X,G) in Figure 5. Vector X has six variables (f,t,s,c,d,r), each is of type 0..n, where n is the number of stations in the protocol. G has eight places named N,F,T,S,C,D,W, and R, and eleven transitions. As shown later, the values of f,t,s,c,d, and r at any instant indicate the numbers of tokens in places F,T,S,C,D and R (respectively) at the same instant. The initial state of this Petrinet is as follows:

$$f=t=s=c=d=r=n_F=n_T=n_S=n_C=n_D=n_W=n_R=0 \text{ and } n_N=n.$$

B. Analysis

a. Proving Safety Properties: It is straightforward to prove that the following assertions are invariants:

i. $n_N+n_F+n_T+n_S+n_C+n_D+n_W+n_R=n$ (Proof is by induction)

ii. $f=n_F \wedge t=n_T \wedge s=n_S \wedge c=n_C \wedge d=n_D \wedge r=n_R$ (Proof is by induction)

iii. $s \geq 1 \Rightarrow t+c=0$ (Proof is by induction)

iv. $s \leq 1 \wedge s.c=0$ (Proof follows from iii)

v. $r \geq 1 \Rightarrow t=0$ (Proof is by induction)

As in the previous protocol, invariants iii and iv establish the mutual exclusion

property of this protocol. Invariant v establishes the fact that a station can be in a "retry" state only if no other station is in a transmitting state.

b. Proving Liveness Properties: The proof is similar to that in previous protocols and so we omit it.

4.5 A Collision Resolution Protocol

In this protocol [2], each station remains in a "not-trying" state until (a) it has a message to send over the medium, and (b) all activities over the medium (i.e. successful transmissions and/or collisions) cease. When these two conditions are satisfied for some station, the station changes its state from "not-trying" to "transmitting", and starts to transmit its message over the medium. This leads either to a successful transmission (in which case the station returns to a "not-trying" state, and the cycle repeats), or to a collision (in which case all the stations involved in the collision take part in a resolution phase that ultimately leads each of them to transmit its message successfully, and return to a "non-trying" state).

In a resolution phase, each of the stations taking part in the resolution "flips a coin"; there are two possible outcomes:

i. *A station gets a "head":* In this case, the station transmits its message over the medium. This leads either to a successful transmission (in which case, the station returns to a "not-trying" state, and the cycle repeats), or to a collision (in which case each of the colliding stations takes part in a resolution phase by flipping a coin, and the cycle repeats).

ii. *A station gets a "tail":* In this case, the station assigns itself a "rank" of value one, and continues to observe the medium:

   a. If it observes a collision, it increments the value of its rank by one, and continues to observe the medium.

   b. If it observes no activity for some time, or a successful transmission, it decrements the value of its rank by one (provided that its value is already greater than one), and continues to observe the medium.

   This continues until the rank of the station becomes zero, in which case the station transmits its message over the medium. This leads either to a successful transmission (in which case, the station returns to a "not-trying" state, and the cycle repeats), or to a collision (in which case, each of the colliding stations takes part in a new resolution phase, by flipping a coin, and the cycle repeats).

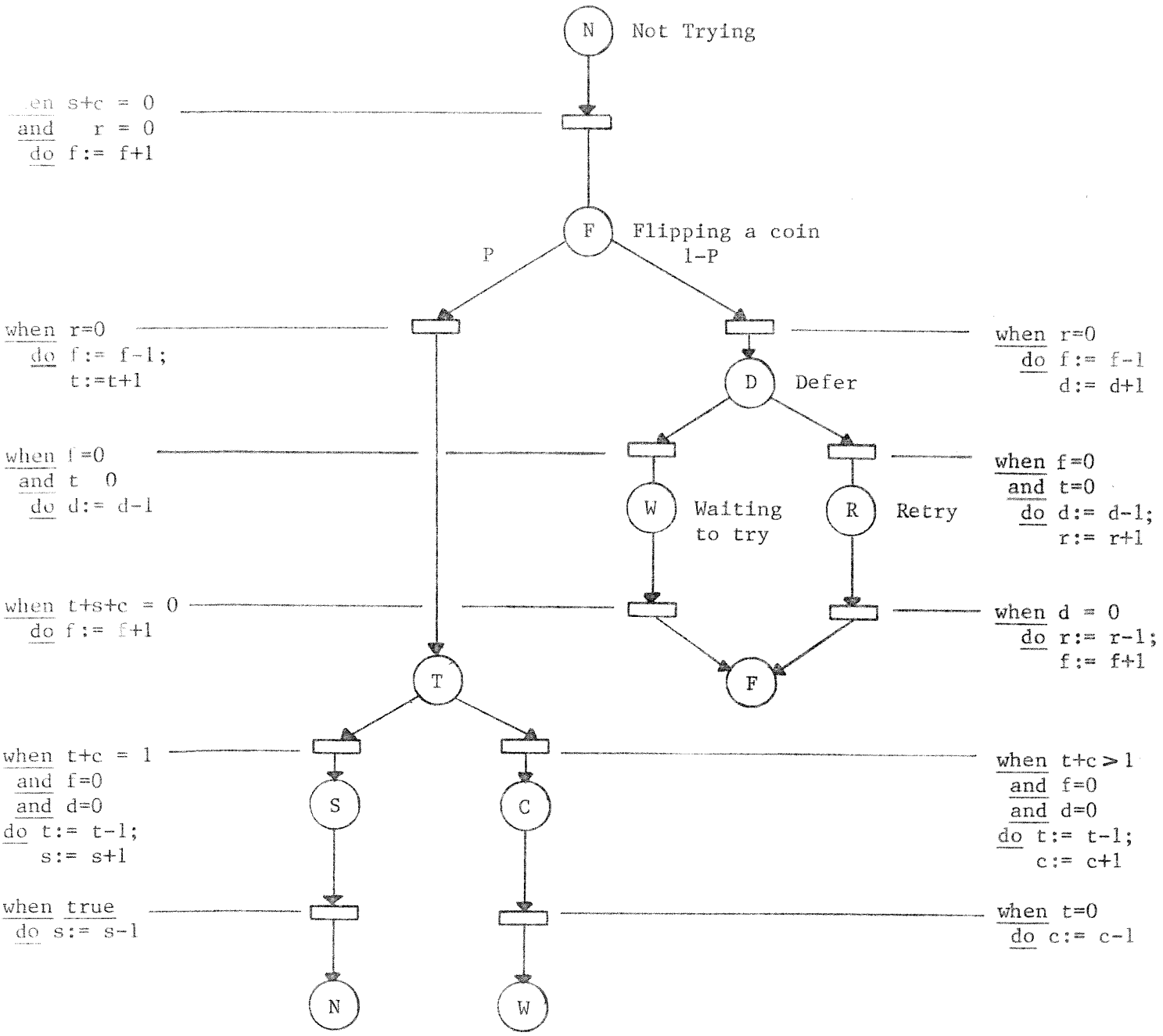


Figure 5 A labeled Petrinet that models the P-persistent CSMA protocol. (For one-persistent CSMA, the enabling predicate of t_{FD} is modified from r=0 to false.)

A. Modeling

This protocol can be defined by the labeled Petrinet (X,G) in Figure 6. Vector X consists of fourteen variables:

i. Each of the ten variables t, s, c, f, d, p, r, l, i, z is of type 0..n, where n is the number of stations in this protocol. (As shown later, the values of these variables at any instant indicate the numbers of tokens in the corresponding places in G at the same instant.)

ii. Each of three variables α, β, and γ is of type *boolean*.

iii. Variable "rank" is of type *stack* whose elements are taken from the set {e,s}, where "e" denotes a *regular element*, and "s" denotes a *separator element*. There are four functions to operate on stack rank:

"rank.push(e)" adds an e element at the top of rank.
"rank.push(s)" adds an s element at the top of rank.
"rank.pop" removes the top element of rank.
"rank.top" refers to the top element of rank.

Stack rank is used to order the stations that have gotten "tail" in their last flipping of coin with respect to their rank values, according to the following four rules:

a. Each element in the stack corresponds to one station that has gotten a "tail" in its last flipping of coin.

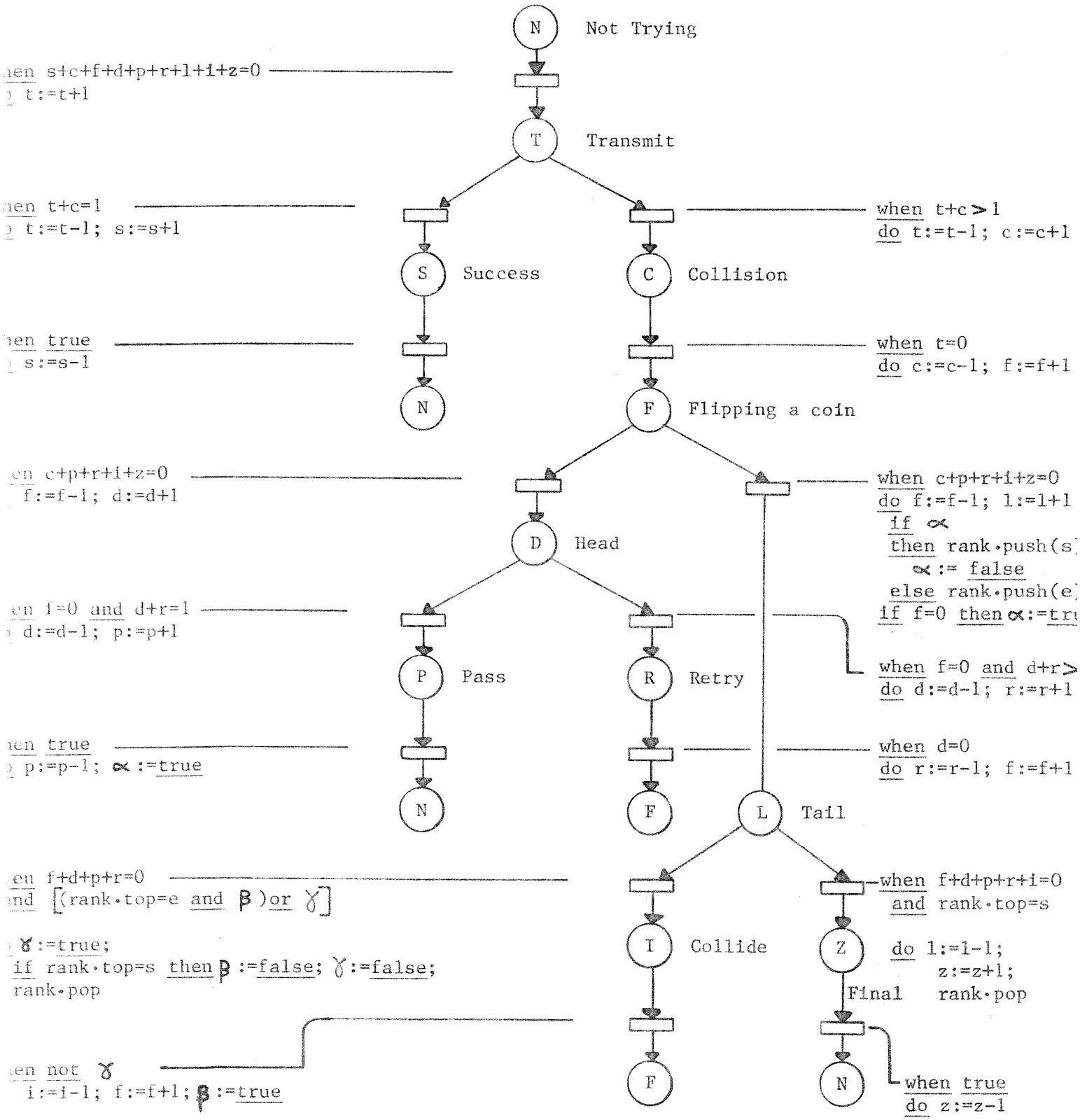
b. The top element in the stack correspond to a station whose rank value is minimum.

c. Two consecutive elements in the stack, where the top element is e correspond to two stations with identical rank values.

d. Two consecutive elements in the stack, where the top element is s correspond to two stations such that the rank of the station corresponding to the bottom element is higher than that of the station corresponding to the top element.

The initial state of this Petrinet is as follows: $t = s = c = f = d = p = r = l = i = z = n_T = n_S = n_C = n_F = n_D = n_P = n_R = n_L = n_I = n_Z = 0$, $n_N = n$, $\alpha = \beta =$ true, $\gamma =$ false, and stack rank is empty.

Figure 6 A labeled Petrinet that models
a collision resolution protocol.



B. Analysis

a. Proving Safety Properties: It is straightforward to show that the following assertions are invariants:

i. $t + s + c + f + d + l + p + r + i + z = n$ (Proof is by induction)

ii. $t = n_T \wedge s = n_S \wedge c = n_C \wedge f = n_F \wedge d = n_D \wedge l = n_L \wedge p = n_P \wedge r = n_R \wedge i = n_I \wedge z = n_Z$ (Proof is by induction)

iii. $s \geq 1 \Rightarrow t + c = 0$ (Proof is by induction)

iv. $s \leq 1 \wedge s.c = 0$ (Proof follows from iii)

v. $f + d + l + p + r + i + z \geq 1 \Rightarrow t + s = 0$ (Proof is by induction)

vi. $s.p = 0 \wedge s.r = 0 \wedge s.i = 0 \wedge s.z = 0$ (Proof follows from v)

vii. $d + l + p + r + i + z \geq 1 \Rightarrow t + c = 0$ (Proof is by induction)

viii. $p.c = 0 \wedge i.c = 0 \wedge z.c = 0$ (Proof follows from vii)

ix. $p \geq 1 \Rightarrow d + r + i + z = 0$ (Proof is by induction)

x. $p \leq 1 \wedge p.i = 0 \wedge p.z = 0$ (Proof follows from ix)

xi. $z \geq 1 \Rightarrow l + i = 0$ (Proof is by induction)

xii. $z \leq 1 \wedge z.i = 0$ (Proof follows from xi)

xiii. The number of elements in stack rank $= n_L$ (Proof is by induction)

Invariants iv, vi, vii, x and xii establish the mutual exclusion property of this protocol.

b. Proving Liveness Properties: The proof is tedious, but follows the same pattern as previous liveness proofs, and so is omitted.

5. CONCLUDING REMARKS

The above approach to modeling and verification of LAN protocols has the following features:

i. **Concise Modeling:** Each LAN protocol is completely defined by one labeled Petrinet whose size (i.e. number of its places, transitions, and edges)

is *fixed*, and does not depend on the size (i.e. number of stations) of the LAN protocol under consideration. This conciseness makes the modeling Petrinets easier to construct, understand, and explain. In fact, the hardest part for us in preparing each of the above five examples was to understand the informal description of the protocol. In each instance, once a protocol was understood, constructing its modeling Petrinet took a couple of hours.

ii. **Straightforward Analysis:** The safety and liveness properties of a LAN protocol are mapped directly to the safety and liveness properties of its modeling Petrinet. The latter can be established in a straightforward fashion using the techniques discussed by Keller in [5]. We hope that the simplicity of these techniques is demonstrated by the above five examples.

iii. **Disadvantage (Abstract Modeling):** The main drawback of this technique is that synchronization mechanisms in some LAN protocols are not modeled explicitly; only their effects are modeled. Consider for example, the labeled Petrinet in Figure 2 that models a token ring protocol. The special message called token in this protocol is not modeled explicitly in the Petrinet. Instead, only its effect of ensuring that the stations transmit one at a time without any collisions is modeled. Another more subtle example of this phenomenon is in the labeled Petrinet of Figure 6. According to this Petrinet, all stations in the "not-trying" state will remain in this state until all activities over the medium (including collision resolutions) cease. The Petrinet, however, does not explain the mechanism by which a station in the "not-trying" state knows that a current collision resolution phase is over. In other words, the Petrinet does not model this mechanism explicitly; it merely models its effect. (For a curious reader, the mechanism is simple [6]: Each station keeps track of the "highest" existing rank in the LAN by continuously observing the sequence of successful transmissions, collisions, and inactive periods over the medium. It then recognizes that the current collision resolution phase is over when it computes that the highest existing rank is zero.) From these examples, a synchronization mechanism may be modeled, using our technique, in an abstract fashion, showing only its effects rather than its details. This seems a "reasonable" price to pay for the technique's simplicity and elegance.

The analysis of LAN protocols discussed in this paper does not address the performance issues, such as response time and throughput, of these protocols. Thus, an interesting problem that deserves further research is to investigate how to extend the analysis of labeled Petrinets to cover such performance issues. (An example of such analysis for regular, unlabeled Petrinets is discussed in Molloy [9].)

REFERENCES

[1] Bux, W., "Local area subnetworks: a performance comparison," *Proceedings of the IFIP W.G. 6.4 Local Area Networks Zurich Workshop*, North-Holland Press, August 1980, pp. 171-196.

[2] Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Transactions on Information Theory*, Vol. IT-25, Sept. 1979, pp. 505-515.

[3] Diaz, M., "Modelling and analysis of communication and cooperation protocols using Petri net based models," *Proceedings of the IFIP W.G. 6.1 Second International Workshop on Protocol Specification, Testing and Verification*, North-Holland Press, 1982, pp. 465-510.

[4] Gouda, M. and C. K. Chang, "Proving liveness for networks of communicating finite state machines," Tech. Rep. TR-84-04, Dept. of Comp. Sc., Univ. of Texas at Austin, Feb. 1984. Submitted for journal publication.

[5] Keller, R. M., "Formal verification of parallel programs," *Communications of the ACM*, Vol. 19, No. 7, July 1976, pp. 371-384.

[6] Massey, J., "Collision resolution algorithms and random-access communication," UCLA Tech. Report UCLA-ENG-8016, 1980.

[7] Metcalfe, R. and D. Boggs, "Ethernet: distributed packet switching for local networks," *Communications of the ACM*, Vol. 19, No. 7, July 1976, pp. 395-404.

[8] Molle, M., "Unification and extensions of the multiple access communications problem," UCLA Tech. Report CSD-81073, June 1981.

[9] Molloy, M., "Performance analysis using stochastic Petri nets," *IEEE Transactions on Computers* Vol. C-31, No. 9, Sept. 1982, pp. 913-917.

[10] Peterson, J., *Petri net theory and the modeling of systems*, Prentice-Hall, 1981.

[11] Sasha, D. E., A. Pnueli, and W. Ewald, "Temporal verification of carrier-sense local area network protocols," *Proceedings of Principles of Programming Language* 1984, pp. 54-65.

[12] Tobagi, F., "Multiaccess protocols in packet communication systems," *IEEE Transactions on Communications*, Vol. COM-28, No. 4, April 1980, pp. 468-488.