

# Petri Nets Theory for the Correctness of Protocols

GÉRARD BERTHELOT AND RICHARD TERRAT

**Abstract**—After a brief introduction to the theory of Petri nets, the ECMA transport protocol is presented. Then a model of the connection and disconnection phases is developed. Properties of correctness are demonstrated, using Petri net theory results, namely, reductions and linear invariants techniques. Predicate/transition nets are introduced and the underlying network service is modeled. Then a model of the data transfer phase is described. It allows correction of the errors signaled by the network level, by using a window mechanism and control frames for acknowledgments and rejections. The correctness of the data transfer is demonstrated using invariants. The service provided to the upper level is thus validated.

## I. INTRODUCTION

IN recent years considerable effort has been given to develop models for specification and validation of protocols (see, e.g., Sunshine [19]). Most of these models fall in two categories: the finite state automaton approach and the programming language approach. Among the models of the former, Petri nets, which have been designed for the purpose of communications between automata, seem a quite natural tool for modeling the protocols and this was suggested by Merlin [14]. However, it was rapidly and commonly recognized that Petri nets suffer from two severe drawbacks: first, modeling a complicated protocol leads to a very intricate and quite unreadable graph, and second, a state exploration analysis of the obtained model is impractical because of the combinatorial explosion.

In order to deal with the first drawback, Danthine [6] was led to use evaluation nets (which are Petri nets with some graphical augmentations) defined by Nutt [15]. Although representation was improved, this model offers no direct solution to the second drawback.

The verification of models by the programming language approach, being made by assertions techniques, does not suffer from the combinatorial explosion, and Bochmann [4] proposed a mixed model based on Keller's model [12] in which control aspects are represented by guarded commands. In the authors' opinion, however, this last model presents two disadvantages. The first one is that it mixes two semantics, the semantic of Petri nets and the semantic of guarded commands. It follows that the interrelations between these two semantics must be extremely carefully defined if we want to avoid contradictions or underspecifications and also to derive rigorous proofs of correctness. The second disadvantage is that there is no clear frontier between those aspects of protocols which must be represented by Petri nets and the

others. This becomes a serious problem when a multilevel protocol is considered. The aim of this paper is to show that progress in the Petri net theory now provides a large number of notions and tools to avoid a brute force state analysis and to obtain useable representations by an integrated model which does not have the disadvantages mentioned above.

In order to illustrate this, we have considered the classes of transport protocols proposed for the ISO reference model by the ECMA [8]. In the first part of the paper, until Section VI, we use invariants methods and reductions methods to prove the correctness of connection and disconnection phases. In the second part, the transmission medium and the transfer phase are modeled by using the predicate/transition nets model. Predicate/transition nets were defined by Genrich and Lautenbach [9] to provide a unique formal basis for various proposed extensions of Petri nets. They differ from the evaluation nets in the sense that under weak restrictions, they can be considered as abbreviations of Petri nets and, thus, can benefit from all the notions and results of the latter. Then, in order to demonstrate that this model can also easily support proofs by assertions techniques, we prove that the data transmission is correct.

## II. BASIC NOTIONS OF PETRI NETS

We give the usual basic notions and notations of the Petri net theory which we shall need later. Further details can be found in [5] and other notions of interest for protocol studies in [7].

A Petri net is a 5-tuple  $PN = \langle P, T; \text{Pre}, \text{Post}; M_0 \rangle$  where  $P$  is a set of places (represented by circles),  $T$  is a set of transitions (represented by bars),  $\text{Pre}: P \times T \rightarrow \{0, 1\}$  is the forward incidence function (represented by an arc from place  $p$  to transition  $t$  if  $\text{Pre}(p, t) = 1$ ),  $\text{Post}: P \times T \rightarrow \{0, 1\}$  is the backward incidence function (represented by an arc from transition  $t$  to place  $p$  if  $\text{Post}(p, t) = 1$ ),  $M_0$  is the initial marking (markings, denoted by  $M$ , are mappings of  $P$  on the set of natural numbers  $\mathbb{N}$ , and  $M(p) = n$  is represented by  $n$  tokens inside the place  $p$ ).

A transition  $t$  is fireable for a marking  $M$  if  $M(p) \geq \text{Pre}(p, t)$  for every place and the firing of  $t$  [denoted by  $M(t > M')$ ] results in a new marking  $M'$  such that  $M'(p) = M(p) - \text{Pre}(p, t) + \text{Post}(p, t)$  for every place.

A sequence  $\sigma = t_1, t_2, \dots, t_n$  of transitions is a firing sequence for a marking  $M_1$  if there exists a sequence of markings  $M_2, M_3, \dots, M_{n+1}$  such that:  $\forall i, 1 \leq i \leq n, M_i(t_i > M_{i+1})$  [this is denoted by  $M_1(\sigma > M_{n+1})$ ].

The set of reachable markings  $[M]$  of a marking  $M$  of  $PN$  is the set of all markings (including  $M$ ) reachable from  $M$  by a firing sequence, i.e.,  $[M] = \{M' / M(\sigma > M'), \sigma \in T^*\}$ .

Manuscript received March 10, 1982; revised July 16, 1982.

G. Berthelot is with the Laboratoire d'Informatique Théorique et Pratique (LITP), C.N.R.S., Paris, France.

R. Terrat is with the Institut de Programmation, Université Pierre et Marie-Curie—Paris VI, 75230 Paris Cedex 05, France.

The language generated by  $PN$  at  $M$ ,  $L(M)$ , is the set of all firing sequences fireable from  $M$ .

A place  $p$  of  $PN$  is *bounded* if this place does not contain more than a fixed number  $b$  of tokens for every reachable marking (i.e.,  $\exists b \in \mathbb{N} \forall M \in [M_0], M(p) \leq b$ ). This place is *safe* if  $b = 1$ .  $PN$  is bounded (safe) if every place is bounded (safe).

A transition  $t$  is *live for a marking  $M$*  if, no matter what marking has been reached from  $M$ , it is always possible to fire  $t$  by firing some further sequence of transitions.  $PN$  is *live* if every transition is live for the initial marking.

A marking  $M$  is a *home-state* if no matter what  $M'$  has been reached from  $M_0$ ,  $M$  is reachable from  $M'$ , i.e.,  $\forall M' \in [M_0], M \in [M']$ .

To provide a matrix representation of  $PN$ , places and transitions can be indexed. Then, a marking is represented by a vector  $M = (m_i)$  where a component  $m_i$  is the marking of place  $p_i$ . The incidence matrix  $C = (c_{ij})$  with  $c_{ij} = \text{Post}(p_i, t_j) - \text{Pre}(p_i, t_j)$  defines completely a Petri net unless it contains some *impurity* (i.e., some place  $p$  and transition  $t$  such that  $\text{Pre}(p, t) = \text{Post}(p, t) = 1$ ). However, it is always possible, in such a case, to add dummy places and transitions to eliminate the impurities.

The incidence matrix representation plays a central role in the Petri net theory; first, because it can be mechanized very easily, and second, because it provides the basis for calculating invariants. These invariants are of two kinds: place-invariants and transition-invariants.

A mapping  $\phi: P \rightarrow \mathbb{N}$  is a transition-invariant if  $\phi$ , considered as a vector, is a solution of  $\phi \cdot C = 0$ . It provides a linear invariant over markings through firing sequences. As a matter of fact, for a given marking  $M$  and a given firing sequence  $\phi$  for  $M$ , if we call  $\bar{\sigma}$  the *characteristic vector* of  $\sigma$  (a component  $\bar{\sigma}_i$  of  $\bar{\sigma}$  is the number of occurrences of transition  $t_i$  in the sequence  $\sigma$ ), then  $\sigma$  leads from  $M$  to a marking  $M' = M + C \cdot \bar{\sigma}$ . Then, by left multiplying the left-hand and right-hand sides of this equation by  $\phi$ , we obtain  $\phi \cdot M = \phi \cdot M'$ . Such invariants have been shown to be very useful in proving important properties as liveness, safety, and mutual exclusion by Lautenbach [13], and considerable work has been carried out to find efficient algorithms to find them [17], [20].

A mapping  $\Omega: P \rightarrow \mathbb{N}$  is a transition-invariant if  $\Omega$ , considered as a vector, is a solution of  $C \cdot \Omega = 0$ . If a firing sequence  $\sigma$  has  $\Omega$  as characteristic vector, it is easily seen by using the equation  $M' = M + C \cdot \bar{\sigma}$  again that  $M = M'$  and, thus, this firing sequence (such firing sequences are called *neutral sequences*) has no effect on the marking. Transition-invariants are very useful to find languages generated by live and safe Petri nets.

### III. ECMA TRANSPORT PROTOCOL

Although Petri nets have been used very often for the representation of rather simple protocols (e.g., for the alternating bit protocol), only very few of the results of this theory were used for the validation. To show that the Petri net theory provides tools to deal with all the aspects (losses and replications of messages, flow control, time-out, and rejections) of a real life protocol, we have chosen the European

Computer Manufacturer Association (ECMA) transport protocol [8]. It is an end-to-end protocol standing between the session and the network layers of the ISO reference model [11].

Four classes of protocols are defined allowing network layer errors of increasing severity (see [8] for details). Each class contains three sets of functions: establishment of a transport connection (connection), data transfer, and termination of a transport connection (disconnection).

We shall turn our attention to the connection and disconnection phases for classes 1 and 2. For these classes there are no mechanisms for recovery from errors, and when an error occurs, each end returns to the initial disconnected state where all but connect request messages are ignored. Using only ordinary Petri nets, we shall model these phases and establish properties of correctness in the absence of errors. Coping with errors leads, obviously, to more complex nets, but the same principles can be used.

Then we shall turn our attention to the data transfer phase. Models for classes 1 and 2 are trivial. For class 3 some problems arise and a model of the transmission medium is needed. To do this we introduce predicate/transition nets and, then, establish properties of correctness for data transfer. Class 4 needs an hypothesis about maximum lifetime of a message in the network transmission medium. It has not been studied here, models and proofs being under investigation.

### IV. CONNECTION AND DISCONNECTION MODEL (CLASSES 1 AND 2)

For session users, the transport layer provides the means to communicate by establishing a transport connection between two transport entities. Transport entities are managed locally in such a way that a transport entity can be involved in a transport connection only if it is disconnected, i.e., has not tried to establish or has not established a transport connection. Hence, simultaneous attempts for connection result in the creation of two transport connections, and there is always an initiating entity and an accepting entity in a transport connection.

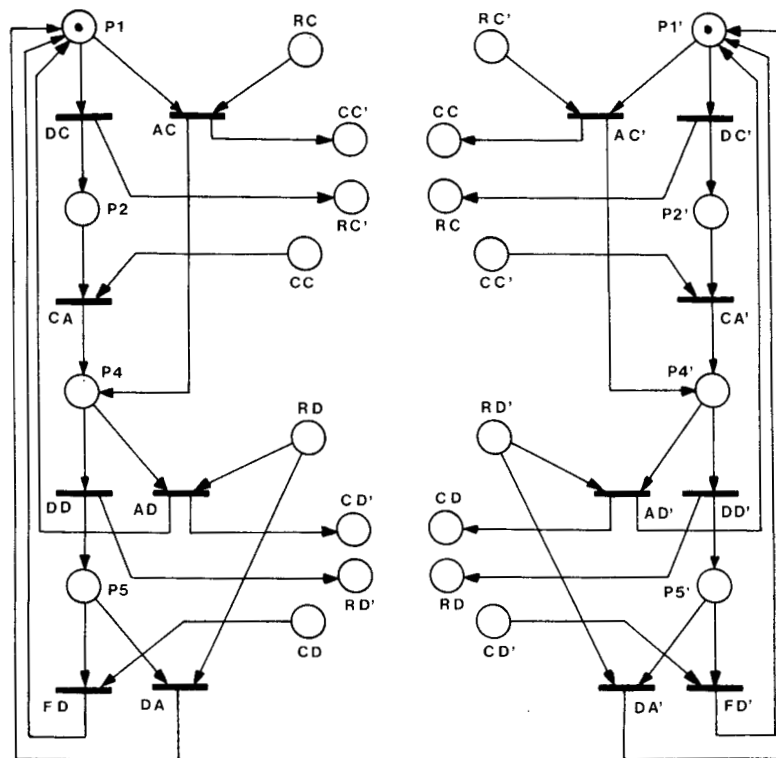
A transport entity is initially disconnected. Its behavior from this state is described by one of the nets of Fig. 1 in which we have not represented interactions with the session level.

In the initial state (place  $P1$ ) and on receipt of a request from the session level, the entity sends (transition  $DC$ ) a connect request (place  $RC'$ ) and passes to the "waiting for a connect confirm" state (place  $P2$ ). When this confirm message arrives (place  $CC$ ), the entity passes to the "data transfer" state (place  $P4$ ) and can send data (not represented here).

Conversely, on receipt of the connect request (place  $RC'$ ), a remote entity is created (with place  $P1'$  marked); it sends (transition  $AC'$ ) a connect confirm (place  $CC$ ), and passes directly to the "data transfer" state.

Once in the data transfer state, each entity may end a transport connection by sending (transition  $DD$ ) a disconnect request (place  $RD'$ ), and passing to the "waiting disconnect confirm" state (place  $P5$ ).

In this state data messages are discarded. On receipt of a



PLACES

States of one entity :

- P1 : idle
- P2 : waiting CC
- P4 : data transfer
- P5 : waiting CD

Control frames :

- RC : connection request
- CC : connection confirm
- RD : disconnection request
- CD : disconnection confirm

TRANSITIONS

- AC : connection accepted
- DC : request for connection
- DD : request for disconnection
- CA : connection established
- AD : disconnection accepted
- DA : disconnection accepted
- FD : end of disconnection

Fig. 1. Nets of transport entities (connection-disconnection phases).

disconnect request, the remote entity returns to the initial state either by sending (transition  $AD'$ ) a disconnect confirm (place  $CD$ ) if it is was in the "data transfer" state, or not (transition  $DA'$ ) if a disconnect collision has occurred and it was in the waiting for disconnect confirm state. The role of each entity can eventually be switched and that is why we have two symmetrical nets. In every state an error signal causes both entities to return to the initial state (transitions not shown for simplicity) where all but connect request messages are ignored (also not shown).

V. VALIDATION OF CONNECTION AND DISCONNECTION (CLASSES 1 AND 2)

In these classes, handling of errors is so drastic that it is quite clear that no problem can occur at this level. Thus, we shall prove properties establishing correctness in the absence of errors. These proofs do not require state exploration analysis but are based on two specific methods of the Petri net theory: the reduction method and the linear invariants method. The reduction method was defined for UCLA graphs and used for protocol verification by Postel [16], and was also developed and refined for Petri nets. Although they have not been used until now for protocol validation, they are

commonly employed in the Petri net field and have even been automated [17].

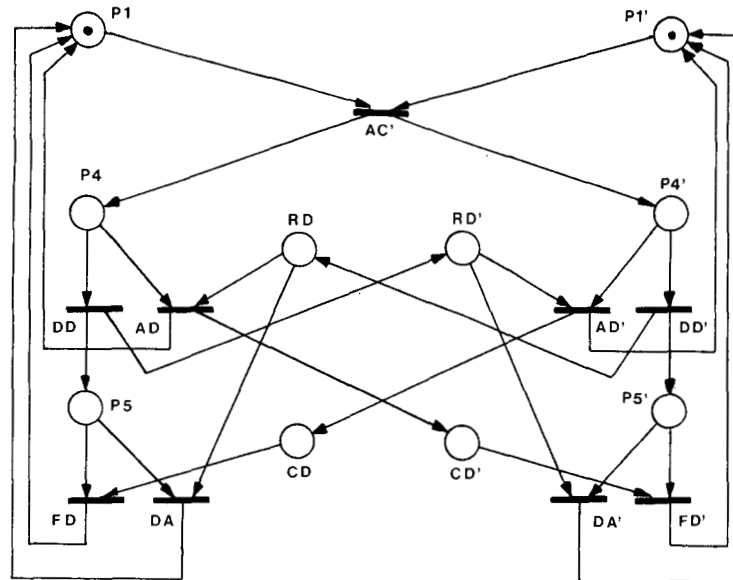
To obtain a global model of the protocol it is sufficient to join the two nets of Fig. 1 by collapsing places with the same names. However, due to the distinction between initiator and acceptor mentioned above, we have to remove transitions  $AC$ , places  $RC$  and  $CC'$  for the left one, transitions  $DC'$  and  $CA'$ , and places  $RC$ ,  $CC'$ , and  $P2'$  from the right one.

This net is then simplified by using reduction rules which are detailed in [1]. These reduction rules preserve properties of interest such as boundedness, safeness, liveness, home-states, and linear invariants. These reductions are the following ones.

- First, there is one token in  $P2$  iff there is one token in  $RC'$  or  $CC$ ; thus,  $P2$  is redundant and may be removed.
- Second, transitions  $DC$ ,  $AC'$ , and  $CA$  are serially fired; thus, they can be gathered on  $AC'$ , and so, places  $RC'$  and  $CC$  disappear.

The reduced net, its incidence matrix, and the solutions of  $\phi \cdot C = 0$  are shown in Fig. 2.

Taking into account the initial marking, these four solu-



(a)

	AC'	DD	AD	FD	DA	DD'	AD'	FD'	DA'
P1	-1		1	1	1				
P4	1	-1	-1						
P5		1		-1	-1				
RD			-1	-1	1				
CD				-1	-1	1			
P1'	-1					1	1	1	1
P4'	1					-1	-1		
P5'						1	-1	-1	
RD'		1				-1	-1	-1	
CD'			1				-1	-1	

I1	I2	I3	I4
1		1	
1			1
1			
		1	
	1		1
	1	1	
	1		
			1
			1

Ω1	1		1			1		1	
Ω2	1	1		1			1		1
Ω3	1	1			1	1			1

(b)

Fig. 2. (a) Reduced net of connected entities. (b) Incidence matrix.

tions lead to the four following place-invariants:

- (I1)  $M(P1) + M(P4) + M(P5) = 1$ .
- (I2)  $M(P1') + M(P4') + M(P5') = 1$ .
- (I3)  $M(P1) + M(P4') + M(RD) + M(CD) = 1$ .
- (I4)  $M(P1') + M(P4) + M(RD') + M(CD') = 1$ .

Property 1: The net is safe.

Proof: Immediate from I1, I2, I3, and I4. No place can hold more than one token. As a result, no message can occur at an end of a transmission more than once before being processed.

Property 2: The initial state is a home-state.

Proof: We use the following theorem from Keller [12]:  $M_a \in [M_0]$  is a home-state if there exists a mapping  $\nu: [M_0] \rightarrow \mathbb{N}$  such that 1) and 2) are satisfied:

- 1)  $\nu(M) = 0 \Leftrightarrow M = M_a$ .
- 2)  $\forall M \in [M_0], \nu(M) \neq 0, \exists \sigma,$   
 $M(\sigma > M' \wedge \nu(M') < \nu(M))$ .

We take  $\nu(M) = 2 \cdot M(P4) + 2 \cdot M(P4') + M(P5) + M(P5')$  which obviously satisfies 1). Then, we prove that if a marking is compatible with the invariants (which is the case of every marking of  $[M_0]$ ), there is at least one transition which decreases  $\nu$ . We have five cases to check:

- 1)  $M(P1) = 1 \wedge M(P1') = 0, \{I2, I3\} \Rightarrow M(P5') = 1$   
 $\{I1, I2, I4\} \Rightarrow$  either  $M(CD') = 1$  and  $FD'$  decreases  $\nu$   
 or  $M(RD') = 1$  and  $DA'$  decreases  $\nu$ .
- 2)  $M(P4) = 1$   $DD$  decreases  $\nu$ .
- 3)  $M(P5) = 1 \wedge M(P1') = 1$  case symmetrical to 1.
- 4)  $M(P5) = 1 \wedge M(P4') = 1$  case symmetrical to 2.
- 5)  $M(P5) = 1 \wedge M(P5') = 1, \{I1, I2, I3\} \Rightarrow M(RD)$   
 $+ M(CD) = 1$

and either  $FD$  or  $DA$  decreases  $\nu$ .

Thus, 2) is also satisfied; hence, the initial state can always be reached again.

*Property 3:* The net is live.

1) It can easily be seen that from the initial state, there exists a firing sequence which will fire any transition.

2) The initial state is a home-state.

As a result, no transition is dead and the net is deadlock-free.

*Property 4:* The resolution of the equation  $C \cdot \Omega = 0$  gives a basis of three characteristic vectors  $\Omega_1$ ,  $\Omega_2$ , and  $\Omega_3$  such that

$$\Omega_1(AC') = \Omega_1(AD) = \Omega_1(DD') = \Omega_1(FD') = 1$$

$$\text{and } \Omega_1(t) = 0 \text{ otherwise}$$

$$\Omega_2(AC') = \Omega_2(DD) = \Omega_2(FD) = \Omega_2(AD') = 1$$

$$\text{and } \Omega_2(t) = 0, \text{ otherwise}$$

$$\Omega_3(AC') = \Omega_3(DD) = \Omega_3(DA) = \Omega_3(DD') = \Omega_3(DA') = 1$$

$$\text{and } \Omega_3(t) = 0 \text{ otherwise.}$$

For the initial marking of the net they correspond to the three following neutral firing sequences:

$$1 = AC'; DD'; AD; FD'$$

$$2 = AC'; DD; AD'; FD$$

$$3 = AC'; (DD//DD'); (DA//DA')$$

where  $t$ ;  $t'$  means term  $t$  followed by term  $t'$ ,  $t//t'$  means term  $t$  and term  $t'$  in any order, and  $t, t'$  means either term  $t$  or term  $t'$ .

Moreover, since this Petri net is safe and live, and since the initial state is a home state, it is possible to show that its generated language from the initial marking to the initial marking is

$$L(M_0, M_0) = (AC'; ((DD'; AD; FD'), (DD; AD'; FD), ((DD//DD'); (DA//DA'))))*.$$

From the properties of reductions, we deduce that the similar language for the nonreduced Petri net is

$$L'(M_0, M_0) = (DC; AC'; CA; ((DD'; AD; FD'), (DD; AD'; FD), ((DD//DD'); (DA//DA'))))*.$$

In the authors' opinion, generated languages are the basis for comparisons between different Petri nets and are of great interest for the definition of the service provided. This area will be investigated in a further paper. This completes our analysis of connection establishment for classes 1 and 2.

Since the data transfer is quite simple for classes 1 and 2, there is no difficulty in formally establishing correctness for these classes. However, if we try to do the same for class 3, two problems arise.

1) Recovery from error needs identification of data messages.

2) Since a lot of control and data messages can be simultaneously transferred on the medium, the order of emission

and the order of reception are of fundamental importance for the validation. So we need a model of the underlying transmission medium.

These two problems cannot be easily solved with ordinary Petri nets and require the use of predicate/transition nets.

## VI. PREDICATE/TRANSITION NETS

Predicate/transition nets have been defined by Genrich and Lautenbach [9] to provide a unified formal basis for various extensions proposed for Petri nets. They can be viewed as a concise abbreviation for large ordinary Petri nets: places stand for sets of ordinary places and can contain tokens having  $n$ -tuples of parameters specified by constants or variables; similarly, transitions stand for sets of ordinary transitions, arcs are labeled by  $n$ -tuples of constants or variables defining the pattern of tokens produced or consumed by a transition in a place, and finally for every transition, a predicate (possibly the ever true predicate) specifies relations between values of different tokens which are needed or produced by its firing.

To illustrate this informal definition, we shall model one direction of the transmission medium which is provided to the class 3 transport protocol by the network layer. This medium does not misorder or duplicate messages, and if messages are lost or corrupted, then this is signaled at each end. For the sake of clarity, we shall assume that in this case a message is replaced by a message of "error." Finally, let us assume also that this medium has a bounded capacity of  $n$  messages and that these messages are of a finite number  $m$  of different types. One direction of such a medium is in fact a FIFO queue with  $n$  elements and is represented by the net of Fig. 3.

Place  $p_f$  contains tokens which represent occupied (full) elements. These tokens have two attributes, the first is the number of a FIFO element and the second is the type of the message that this element contains. Place  $p_e$  contains tokens with only one attribute which represents the number of an empty element. Thus,  $p_e$  contains numbers of empty elements.

Transitions  $t_s$  and  $t_r$  constitute the interface between the network and transport layers (not represented here) and, thus, are only partly specified here. This is why they have no predicate.

Transition  $t_s$  stands for the sending of a message: if the first element is empty ( $p_e$  contains the token with attribute 1), it can put a token with attributes  $(1, i)$  in the first element for a message where type  $i$  is specified by the transport layer. Conversely, transition  $t_r$  stands for the receipt of a message of type  $j$  which was contained by the last ( $n$ th) element. Thus, this element becomes free and transition  $t_r$  puts a token with number  $n$  in  $p_e$ .

Transition  $t_p$  stands for the progression of messages from one element to the following: if the element  $q$  contains a message of type  $k$ , if the element  $q'$  is empty and if  $q' = q + 1$ , then a message of type  $k$  is put in the element  $q'$  and the element  $q$  becomes empty. This can be also simplified by writing  $q + 1$  instead of  $q'$  and removing the predicate. Finally, transition  $t_e$  stands for detected loss or corruption of message: a token of type  $e$  (error) is substituted for a message of type  $k$  in any element  $q$ .

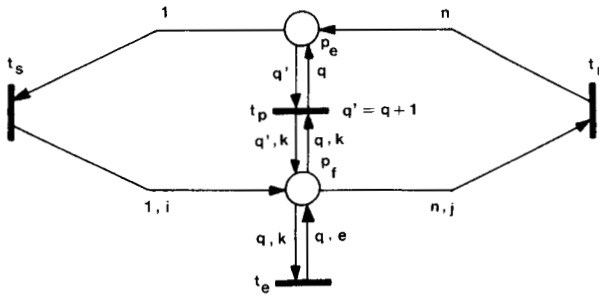


Fig. 3. Predicate/transition net of one direction of the medium.

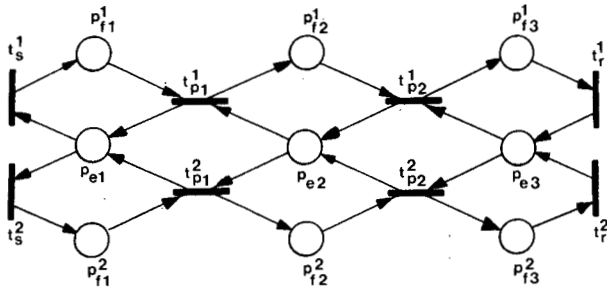


Fig. 4. Unfolded net.

*Remark:* It must be noticed that labels of arcs are only related to the transition they are connected with and have no global significance.

As mentioned above, such predicate/transition nets can be expanded to obtain ordinary Petri nets. This is done by having for  $p_f$  and  $p_e$  a place for every value of the attributes that tokens can carry. This leads to the net of Fig. 4 in which  $m = 2, n = 3$ , and transitions for errors are not represented.

## VII. DATA TRANSFER PHASE (CLASS 3)

We shall now focus our attention on another critical but less studied aspect of protocols: correct data transfer. Connection and disconnection can be proved with the same method we shall use for data transfer. First, we define the model, and then, in the following section, correctness properties are established.

### A. Data Transfer Principles

Control flow is performed by numbering data messages modulo  $N$  inside the transport layer and by a window mechanism with size  $f$  ( $f$  must be less than  $N$  for obvious reasons). That is, the sender is authorized to send  $f$  messages from the last acknowledged message. At any time the receiver may send an acknowledgment with the number of the next message it is awaiting, and by so doing, it acknowledges all messages until this number. When an error is signaled, the receiver sends a reject frame with the number of the next expected message.

### B. Data Transfer Model

The data transfer model is represented in Fig. 5. The sender at the left-hand side and the receiver at the right-hand side are connected together by the network transmission medium of Section VI.  $M$  represents a FIFO queue for messages, while  $A$  represents a FIFO queue for acknowledgments and rejects.

For the sake of clarity, interrelations between  $M$  and  $A$  for error signaling are not represented. Transitions  $t_2, t_3, t_4, t_5, t_6, t_7$ , and  $t_8$  are the interface with the network layer.

Transitions  $t_1$  and  $t_5$  are the interface with the session layer. In order to prove the correctness of the transmission, two counters (places  $P$  and  $Q$ ) belonging to the session layer are introduced. They have the following meaning.

$P$ : Session sequence number of the next message to be transmitted to the transport layer. Its current value, denoted by  $p$ , is incremented each time a message is transferred (transition  $t_1$  of the sender).

$Q$ : Number of messages received by the session layer. Its current value, denoted by  $q$ , is incremented each time a message is transferred to the session layer (transition  $t_5$  of the receiver).

Messages transmitted on  $M$  carry two numbers, the first denoted by  $m$  is the transport sequence number and the second denoted by  $x$  is the session sequence number. The session number is set by  $t_1$  to the current value of  $P$ , but it is invisible and unused in the transport layer.

*Notations:*  $\oplus$  and  $\ominus$  stand for addition and subtraction modulo  $N$ .  $[x, y]$  denotes the sequence of number from  $x$  to  $y$ . The appropriate bracket is toward the outside if the lower or upper bound is excluded from the sequence, i.e.,  $[1, 4[$  denotes 1, 2, 3.

*Sender:* The sender consists of a buffer and three counters.

$B$ : Buffer for messages. In the initial state,  $B$  contains dummy messages with transport sequence numbers from  $0 \ominus f$  to  $0 \ominus 1$ . These messages cannot be sent since  $s = 0$ . It will be shown that in any state,  $B$  contains all messages with transport sequence numbers from  $z \ominus f$  to  $z \ominus 1$ . Messages from  $z \ominus f$  to  $r \ominus 1$  are acknowledged and cannot be sent. Messages from  $r$  to  $z \ominus 1$  can be sent by  $t_2$ .

$R$ : (current value denoted by  $r$ ). Transport sequence number of the first unacknowledged message. When receiving an acknowledge frame ( $i, a$ ) or a reject frame ( $j, a$ ) (transitions  $t_3$  and  $t_4$ ), its value is updated to the sequence number  $a$  contained in the frame.

$S$ : (current value denoted by  $s$ ). Transport sequence number of the next message to be sent. This message can be sent from  $B$  (transition  $t_2$ ) only if its number is inside the window (i.e.,  $s \in [r, r \oplus f[$ ). After that,  $S$  is incremented modulo  $N$ . When a reject frame ( $j, a$ ) or an acknowledge frame ( $i, a$ ) with a sequence number outside the expected window is received (transition  $t_3$ ),  $S$  is returned to the value  $a$  contained in the frame.

$Z$ : (current value denoted by  $z$ ). Transport sequence number of the next message to be received from the upper layer. Such a message can be received (transition  $t_1$ ) only if the message  $z \ominus f$  has been acknowledged, i.e.,  $z \in [r, r \oplus f[$ . This message is stored in  $B$ .

Error signals from the network layer received by the sender from  $A$  are discarded (not represented here).

*Receiver:* The receiver consists of one counter  $W$ . Its current value denoted by  $w$  is the sequence number of the next expected message. Its use is described below.

When receiving the expected message (transition  $t_5$ ),  $W$  is incremented modulo  $N$ .

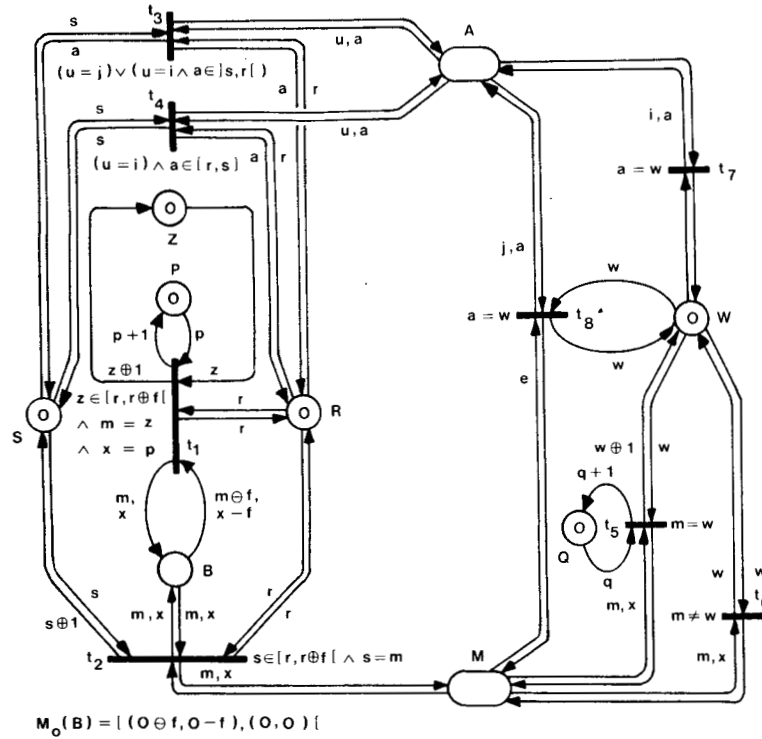


Fig. 5. Transport protocol.

Other messages are discarded (transition  $t_6$ ) and  $w$  is unchanged.

At any time the receiver can send (transition  $t_7$ ) an acknowledgment frame with value  $w$ .

On receipt of an error signal (transition  $t_8$ ), a reject frame with value  $w$  is sent. It must be noticed that if this reject frame is lost, another error will be signaled and, thus, another reject frame will be sent.

For several protocols, it is possible for the receiver to accept a data message not only if it carries the next expected number, but also if this number is inside the window  $[w, w \oplus f[$ . This would lead to a more complicated model but, conversely, proof of correct data transfer would be simplified because it was demonstrated by Stenning [18] that this proof can be exhibited only if the window size is less or equal to  $N/2$ . In this case the medium cannot contain two different messages with the same modulo number and we would not need the notion of increasing subsequence (see below).

### VIII. VALIDATION OF DATA TRANSFER

We shall show that data are correctly transmitted without losses, duplications or misordering. This property is established with the help of invariant assertions. Absence of deadlock is also established according to the same method used in Section V (i.e., a transition can be fired from a home-state).

*Notation:* The counter  $W$  can progress only when receiving messages constituting a sequence of strictly increasing numbers without miss. Such a sequence is formalized by the notion of increasing subsequence.

An *increasing subsequence*  $ISS(w)$  beginning in  $w$  is a subsequence of messages of  $M$  such that the first one carries



Fig. 6. Diagram for assertions  $AS_0-AS_5$ .

the transport sequence number  $w$ , and the  $i$ th carries the number  $w \oplus i \ominus 1$ .

*Example:* ( $N = 4, f = 3, w = 3$ )

receiver ← sender

content of  $M = 0, 1, 2, 3, e, 1, 2, 3, 0, 1$ .

$ISS(3) = 3, 0, 1$ .

Such an increasing subsequence can be modified or even emptied if messages are lost or corrupted.

*Assertions on Transport Sequence Number:* The following assertions are true in the initial state and remain true through the firing of every transition (see Fig. 6).

- $AS_0$   $B = [z \ominus f, z[$  (all transport sequence numbers from  $z \ominus f$  to  $z \ominus 1$  are in  $B$ ).
- $AS_1$   $z \in [w, r \oplus f[$ .
- $AS_2$   $w \in [r, z]$ .
- $AS_3$   $ISS(w) \subset [w, z[$  (every transport number of  $ISS(w)$  is in  $[w, z[$  and  $ISS(w)$  is initially empty).
- $AS_4$   $a \in [r, w]$  (every acknowledgment or reject in  $A$  has a number in  $[r, w]$ ).
- $AS_5$   $s \in [r, w] \vee s \in ]w, z] \vee (ISS(w) = w, w \oplus 1, \dots, s \oplus 1, \dots \vee e \in M \vee j \in A)$   
 $(e \in M$ : there exists an error frame in  $M)$   
 $(j \in A$ : there exists a reject frame in  $A)$ .

*Sketch of the Proof:* (complete proofs can be found in [3]) we prove the theorem by induction on transition firings.

*Basis:* For the initial state  $M_0$ , assertions are trivially true.

*Firings:* We assume that  $M_0(\sigma > M)(t' > M')$  and we prove that  $M'$  fullfills the assertions under the assumptions that  $M$  does.

*Property 5:* The protocol is deadlock-free.

The two lemmas below prove that the initial state is reachable from every reachable state.

*Notation:* The state of the protocol is represented by a 6-tuple  $\langle z, s, r, w, \bar{m}, \bar{a} \rangle$  where  $z, s, r, w, \bar{m}, \bar{a}$  are, respectively, the markings of  $Z, S, R, W, M, A$ . The marking of  $B$  can be deduced from  $AS_0$ . " $\cdot$ " denotes the empty marking.

*Lemma 1:*  $\langle z, z, z, z, \cdot, \cdot \rangle$  is a home-state for  $\langle z, x, y, t, \bar{m}, \bar{a} \rangle$ .

*Proof:* The protocol can progress according to the following steps.

*Step 1:*

$$\langle z, x, y, t, \bar{m}, \bar{a} \rangle \xrightarrow{\tau_1} \langle z, x, y, u, \cdot, \bar{a} \rangle$$

clearing out  $M$  (either  $t_5, t_6$ , or  $t_8$  is fireable).

*Step 2:*

$$\langle z, x, y, u, \cdot, \bar{a} \rangle \xrightarrow{\tau_2} \langle z, b, c, u, \cdot, \cdot \rangle$$

clearing out  $A$  (either  $t_3$  or  $t_4$  is fireable).

*Step 3:*

$$\langle z, b, c, u, \cdot, \cdot \rangle \xrightarrow{\tau_3} \langle z, u, u, u, \cdot, \cdot \rangle$$

sending an acknowledge frame ( $t_7$ ); if the frame is received by  $t_4$ , it follows from  $AS_5$  that  $s \in [r, w]$ , thus,  $s = u$ .

*Step 4:* While  $u \neq z$ ,  $t_2$  is fireable (from  $AS_0$ )

$$\langle z, u, u, u, \cdot, \cdot \rangle \xrightarrow{\tau_4} \langle z, z, z, z, \cdot, \cdot \rangle$$

where  $\tau_4 = (t_2 t_5 t_7 t_4)^{z \ominus u}$ .

*Lemma 2:*  $\langle 0, 0, 0, 0, \cdot, \cdot \rangle$  is a home-state for  $\langle z, z, z, z, \cdot, \cdot \rangle$ .

*Proof:*

$$\langle z, z, z, z, \cdot, \cdot \rangle \xrightarrow{\tau_5} \langle 0, 0, 0, 0, \cdot, \cdot \rangle$$

where  $\tau_5 = (t_1 t_2 t_5 t_7 t_4)^{0 \ominus z}$ .

*Lemma 3:*  $t_2$  can be fired from the home-state  $\langle 0, 0, 0, 0, \cdot, \cdot \rangle$ .

This completes the proof of Property 5.

*Assertions on Session Sequence Numbers:* The following assertions are true in the initial state and remain true through firing of every transition:

$$AS_6 \quad z = p \text{ modulo } N.$$

$$AS_7 \quad m = x \text{ modulo } N.$$

$$AS_8 \quad w = q \text{ modulo } N.$$

$$AS_9 \quad p \in [q, q + f].$$

$$AS_{10} \quad x \in [q, p[ \text{ for every message belonging to ISS } (w).$$

$$AS_{11} \quad B \subset [p - f, p[ \text{ (session sequence numbers of messages contained in } B \text{ are between } p - f \text{ and } p - 1).$$

*Proof:* For  $AS_6$  and  $AS_8$  it is evident since the two counters are incremented simultaneously but the first is incremented modulo  $N$ .  $AS_7$  derives directly from  $AS_6$ . Proof of the rest is accomplished similarly to  $AS_0$ - $AS_5$  and can also be found in [3].

*Property 6:* Messages are transmitted without loss, duplication or misordering.

*Proof:* To show this property it is necessary and sufficient to show that every message transferred to the session layer carries a session sequence number equal to the total number of messages received by this layer (since the first message carries the number zero). Thus, we shall prove that  $x = q$  every time transition  $t_5$  fires. Let us assume that transition  $t_5$  is fireable with a message such that  $m = w$ . From  $AS_7$ ,  $m = x$  modulo  $N$  and from  $AS_8$ ,  $w = q$  modulo  $N$ ; thus,  $m = w$  implies  $x = q + k \cdot N$ . From  $AS_{10}$  and  $AS_9$ ,  $q \leq x < q + f$ ; hence,  $q \leq q + k \cdot N < q + f$  or  $0 \leq k \cdot N < f$ . But,  $f < N$  implies  $k = 0$  and finally we have  $x = q$ .

With Property 5 (deadlock-freeness) and Property 6 (correct message transmission), we have proved that the net of Fig. 5 has the same properties as a net similar to the one of Fig. 3, but of length  $f$  and without transition  $t_e$  (i.e., a FIFO queue of  $f$  elements without losses of messages). So, in our opinion, such a net defines the service provided to the session layer by the transport layer. That means that the study of a session level protocol can be done by using a model of the service provided and not by the complete model of the underlying transport protocol.

## IX. CONCLUSION

We have shown that Petri net theory provides a large number of notions and tools for the study of even complicated protocols. Predicate/transition nets, which use the simple semantics of ordinary Petri nets, allow complete but clear graphical representation even for algorithmic aspects such as the numbering of messages. For correctness proofs we did not employ the usual states analysis, but both specific Petri nets methods (reductions, linear invariants) on ordinary Petri nets and assertion methods on predicate/transition nets; it must be pointed out, however, that it has been shown in [9] that invariants can be used directly on predicate/transition nets; and practical tools for this method and also for the reduction method are currently under development. Even when they can only be obtained by assertions techniques, proofs could probably be automated or aided in such verification systems as AFFIRM [10], since these proofs take advantage, in a natural way, of predicates of transitions which give Pre and Post assertions for changes.

Finally, we think that this theory provides a formal basis for comparison between models, namely by comparison of the generated languages. An important issue about this is the proof that a protocol provides to the upper layer the service required.

## ACKNOWLEDGMENT

The authors are grateful to C. Sunshine for his numerous corrections and helpful remarks on the manuscript.

REFERENCES

[1] G. Berthelot, G. Roucairol, and R. Valk, "Reductions of nets and parallel programs," in *Proc. Advanced Course General Net Theory Processes Syst.*, Hamburg, Germany, 1979 (Lecture Notes in Computer Science, vol. 84), W. Brauer, Ed. New York: Springer-Verlag, 1980.

[2] G. Berthelot and R. Terrat, "Utilisation de reseaux de Petri a predicats pour la modelisation et la preuve de protocoles de transmission de type HDLC," Presented at the AFCET Cong., Sept. 1981.

[3] —, "Petri nets theory for correctness of protocols," in *Proc. 2nd Int. Workshop Protocol Specification, Testing, Verification*, May 1982. New York: North-Holland, 1982.

[4] G. V. Bochmann and J. Gecsei, "An unified method for the specification and verification of protocols," in *Proc. IFIP Cong.*, Toronto, Ont., Canada, Aug. 1977. New York: North-Holland, 1977.

[5] W. Brauer, *Proc. Advanced Course General Net Theory Processes Syst.*, Hamburg, Germany, 1979 (Lecture Notes in Computer Science, vol. 84), W. Brauer, Ed. New York: Springer-Verlag, 1980.

[6] A. Danthine, "Petri nets for protocol modelling and verification," in *Proc. Comput. Networks Teleprocessing Symp.*, vol. II, Budapest, Hungary, Oct. 1977.

[7] M. Diaz, "Modeling and analysis of communication and cooperation protocols using Petri net based models," in *Proc. 2nd Int. Workshop Protocol Specification, Testing, Verification*, May 1982. New York: North-Holland, 1982; also *Comput. Networks*, to be published.

[8] ECMA, 3rd Draft of Transport Protocol, ECMA/TC24/80/16, Feb. 1980.

[9] H. J. Genrich and K. Lautenbach, "The analysis of distributed systems by means of predicate/transition nets," in *Semantics of Concurrent Computation* (Lecture Notes in Computer Science, vol. 70), G. Kahn, Ed. New York: Springer-Verlag, 1978.

[10] S. L. Gerhart, "An overview of AFFIRM: A specification and verification system," in *Proc. IFIP Cong.*, Melbourne, Australia, Oct. 1980. New York: North-Holland, 1980.

[11] "Data processing open system interconnection basic reference model," ISO 97/16/DP 7498, *Comput. Networks*, vol. 5, no. 2, 1981.

[12] R. M. Keller, "Formal verification of parallel programs," *Commun. Ass. Comput. Mach.*, vol. 19, July 1976.

[13] K. Lautenbach and H. Schmid, "Use of Petri nets for proving correctness of concurrent processes systems," in *Proc. IFIP Cong.* New York: North-Holland, 1974.

[14] P. M. Merlin, "A methodology for the design and implementation of communication protocols," *IEEE Trans. Commun.*, vol. COM-24, 1976.

[15] G. F. Nutt, "Evaluation nets for computer system performance analysis," in *Proc. AFIPS Fall Joint Comput. Conf.*, vol. 41. Montvale, NJ: AFIPS Press, 1972.

[16] J. B. Postel and D. Farber, "Graph modelling of computer communications protocols," in *Proc. 5th Texas Conf. Comput. Syst.*, Austin, TX, Oct. 1976.

[17] B. Pradin, "Un outil interactif pour la vérification des systèmes à évolution parallèle décrits par réseaux de Petri," these de Docteur-Ingénieur, Toulouse, France, Dec. 1979.

[18] N. V. Stenning, "A data transfer protocol," *Comput. Networks*, vol. 1, no. 2, 1976.

[19] C. A. Sunshine, "Formal modeling of communication protocols," Inform. Sci. Inst., U.S.C., Los Angeles, CA, RR-81-89, Mar. 1981. Also in *Computer Networks and Simulation*, S. Schoemaker, Ed. New York: North-Holland, 1982.

[20] J. M. Toudic, "Algorithmes d'analyse structurelle des réseaux de Petri," thèse de 3ème cycle, Institut de Programmation, Univ. Pierre et Marie Curie, Paris, France, Oct. 1981.



**Gérard Berthelot** was born in Paris, France, in 1951. He received the Doctorat de Troisième Cycle degree in computer science in 1978 from the University of Paris VI, Paris, where he has been Assistant Professor of computer science since 1976.

In 1978 he joined the Laboratoire d'Informatique Théorique et Pratique (LITP) of C.N.R.S., Paris, where he is now involved in the new Concurrency Communication and Cooperation project. His current research interests include description and verification of concurrent processes, mainly with Petri nets and other related models, but also with abstract data types and temporal logic.



**Richard Terrat** received the engineering degree from the Ecole Nationale Supérieure des Arts et Métiers, France, in 1967 and the Doctorat de Troisième Cycle degree in 1970.

From 1970 to 1972, he was professor at the University of Montreal, Montreal, P.Q., Canada, before joining the University of Pierre et Marie Curie, Paris, France. From 1974 to 1980 he worked in the development of simulation tools for operating systems. He is Director of the Department of Computer Science of the University Pierre et Marie Curie. His current research interests include the modeling of concurrency in distributed systems in order to prove correctness of some important properties, especially about communications protocols and distributed algorithms.