computer network protocols, A. Danthine, editor.
© université de liège, 1978.

FINITE STATE DESCRIPTION OF COMMUNICATION PROTOCOLS*

Gregor V. Bochmann[+]
Département de Mathématiques
Ecole Polytechnique Fédérale de Lausanne
Lausanne, Suisse

SUMMARY

This paper gives an overview of a finite state model for the specification and validation of communication protocols. The concept of "direct coupling" between interacting finite state components is used to describe a hierarchical structure of protocol layers. This paper discusses different aspects of protocol validation, some verification tools based on the finite state formalism, and the basic limitations of the finite state modelling of protocols. An "empty medium abstraction" is proposed for reducing the complexity of the overall system description. The concept of "adjoint states" can be useful for summarizing the relative synchronization between the communicating system components. These concepts are applied to the analysis of a simple alternating bit protocol, and to the X.25 call set-up and clearing procedures. The analysis of X.25 shows that the procedures are stable in respect to intermittant perturbations in the synchronization of the interface introduced for different reasons, including occasional packet loss. However, on very rare occasions, an undesirable cyclic behavior could be encountered.

1. INTRODUCTION

Formalized methods for the specification and verification of communication protocols are developed for simplifying the problems of design, validation and implementation. Two basically different approaches have been used for this purpose : modelling by finite state machines, and specifications using high-level programming languages.

The purpose of this paper is two-fold. The paper gives an overview of a modelling approach with finite state machines and points out its advantages and limitations. It also suggests the use of the "empty medium abstraction" for reducing the complexity of the state space of the system, and the concept of "adjoint states" for summarizing the relative synchronization between the communicating entities. The second purpose of the paper is the detailed presentation of an analysis of the X.25 call set-up and clearing procedures. Most of the ideas and results in this paper have already been described in an unpublished report (1).

2. FINITE STATE DESCRIPTION OF PROTOCOLS

The basic approach of the description method consists of subdividing the system into a number of communicating components, such that each component is a finite state machine. For the i-th component of the system, we use the following notation : The possible states of the component are written s_i . A transition between two states s_i and s_i' is identified by these states : $s_i \longrightarrow s_i'$. The transitions of the component are partitioned into a number of transition types. We write $s_i \xrightarrow{t} s_i'$ to indicate that the transition $s_i \longrightarrow s_i'$ is of type t .

2.1 Direct coupling

A possible basic communication mecanism between the different components of the system is what we call "direct coupling". For a given component, certain types of transitions are directly coupled with transition types in other components. Such a transition can only be executed in parallel with a corresponding transition in another component. All non-coupled transitions of a component can be executed independently from the other components. (In the examples given in this paper, most transition types turn out to be coupled).

For each pair of communicating components i and j , the coupling is specified by a list of pairs of directly coupled transition types $\{t_i \| t_j\}$. We write $t_i \| t_j$ to indicate that transition type t_i of component i is directly coupled with transition type t_j of component j . For example, the direct communication of a *sender* and *receiver* component in the form of message transmission can be modelled by the direct coupling

[+] On leave from the Université de Montréal, Canada.

* Work supported in part by the National Research Council of Canada and the Ministère de l'Education du Québec.

$\underline{m}_{sender} || m_{receiver}$, which means that a transition of type $\underline{m}$ of the *sender* can only be executed with a simultaneous transition of type m of the receiver. (We use the notation where sending transition types are underlined, and receiving transition types are non-underlined).

The concept of direct coupling can be used to describe the interaction between the two stations of a communications protocol through a medium (see section 3), between a station and the medium (see section 2.2 below), and between different components within the same station (2). One way of realizing direct coupling is by identifying certain output symbols of the automaton of one component with certain input symbols of the other component, without any intermediate buffering (3,4).

2.2. Modelling communication protocols

In the following, we consider the modelling of the communication between two stations. In the case of a hierarchical protocol structure in several layers, we consider the specification and validation of one layer, and its relationship to the layers above and below.

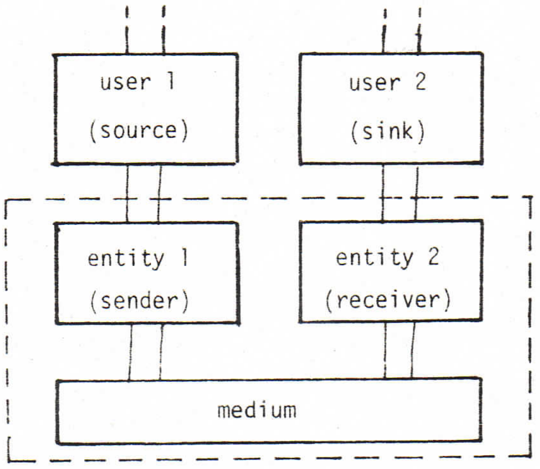


Figure 1 : Architectural structure of a communication system.

Figure 1 shows the component structure of the two stations, from the point of view of the protocol layer implemented in two particular system components, called *entity 1* and *entity 2*. These components use facilities of the component called *medium* for the exchange of information (at the lowest protocol level, this would be the physical transmission medium between the two stations). In turn, they provide a communication facility for the components, called *user 1* and *user 2*, of the next higher protocol layer. Therefore, together, the components *entity 1*, *entity 2*, and *medium* can be considered the communication medium to be used by this next higher layer.

Each component, including the *medium*, is modelled by a finite state machine. For example, a medium providing unreliable transmission of messages between a *sender* and *receiver* component can be modelled by the state diagram of figure 2.

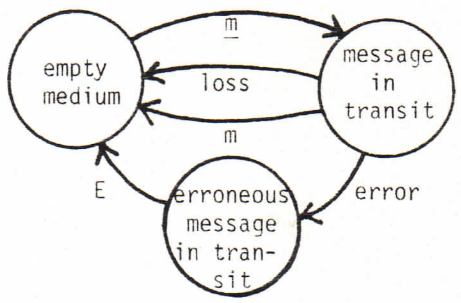


Figure 2 : Finite state model of a simple transmission medium.

The interface between the components, in our model, are modelled by direct couplings. In the above example, the interface between the *sender* and the *medium* would be characterized by the direct coupling $\underline{m}_{sender} || \underline{m}_{medium}$, and the interface between the *receiver* and the *medium* by $m_{receiver} || m_{medium}$, and $E_{receiver} || E_{medium}$ (the latter transitions indicating the reception of an erroneous message). The transition type *loss* and *error* of the *medium* would be uncoupled (spontaneous).

A similar formalism for the modelling of protocols has been developed independently by Rusbridge and Langsford (3), who also give a characterization of different kinds of simple transmission media.

2.3 Protocol validation

The validation of protocols specified in terms of interacting finite state components can be automated to a large extend (5,6). Compared to the approach of specifying protocols in terms of a programming language, where this does not seem feasable, this is an advantage of the finite state approach; although it sometimes seems to be a necessity because of the state space explosion of the considered problems.

It is difficult to define what protocol validation means. The following points, each describes a certain aspect of the operation of a protocol. Protocol validation can be considered as the analysis of these different aspects, and the comparison of the results obtained with the operational requirements (7). These principles are the same for protocols and other systems of parallel processes (8).

2.3.1 Reachability analysis

The basis for all subsequent validation aspects is an analysis of the possible transitions of the overall system, the state space of which is the Cartesian product of the state spaces of the interacting components. The reachability analysis yields the transition diagram of the overall system.

2.3.2 Deadlocks

A deadlock is characterized by a state of the overall system, reachable from the initial state of the system, for which no further transition is possible. Deadlocks must be avoided, since once the system has arrived in a deadlock state, it is blocked forever.

2.3.3 Liveness

A state (or transition) is live if it can be reached from all states of the overall system that are reachable from the initial state. Usually a protocol contains a so-called "home state", or "ready state", which is live and from which all pertinent operations of the protocol can be reached.

2.3.4 Loops

Each protocol with a home state contains a loop in the transition diagram of the overall system starting at the home state and leading back to it. Usually, there are other loops necessary for the operation of the protocol. In addition, mostly during the design phase of a new protocol, other loops can possibly be found in the transition diagram. They are undesirable since their execution does not advance the useful processing of the protocol. Because they could be followed by the system an unlimited number of times, depending on the relative speed of different operations, they have been called "tempo-blockings" (6).

It seems difficult to automatically distinguish between loops that are useful and necessary, and loops that are undesirable. To make this distinction, it is important that the transition diagram of the overall system be comprehensible to the (human) designer. (For the other points of the validation, complete automation seems to be possible).

2.3.5 Self-synchronization and stability

A system is self-synchronizing if, started up in any possible state of the overall system, it always returns, after some finite number of transitions, into the normal cycle of operation including the home state. This property is important for error recovery in an unreliable environment where, for example, the transmission medium does not always function properly, one station does not follow the prescribed protocol due to a software or hardware bug, or the protocol is not properly initialized.

The self-synchronizing property implies protocol stability, because it ensures that the protocol reverts directly to its normal mode of operation after any initial or intermittent perturbation in the synchronization of the two communicating subsystems, introduced for whatever reason.

2.3.6 Characterization of the operation by action sequences

To verify that the actions executed by the protocol correspond to the operations to be performed, it may be useful to characterize the possible action sequences of the overall system by regular expressions (9). Since the system usually contains loops, the possibility of infinite action sequences must be foreseen. This requires an extension (9,10) of the classical formalism of regular expressions.

2.4 Abstraction

We consider again the hierarchical aspect of the protocol model shown in figure 1. As explained above, the operation of the three components in the broken box is described by an overall transition diagram which is the product of the three interacting components. From the point of view of the next higher protocol layer, i.e. as seen from the components *user 1* and *user 2*, these three components together form a transmission medium. From this point of view the broken box should be described by a finite state diagram containing essentially the transitions that are directly coupled with transitions of the higher level entities *user 1* and *user 2*. Clearly, this diagram may be much simpler than the overall transition diagram of the box, and is therefore an abstraction of the latter.

A possible method for obtaining this abstraction is the following : Derive an extended regular expression for the possible action sequences of the protocol (see section 2.3.6), including as relevant actions only those transitions that are directly coupled with the next higher protocol layer. Then construct a (simple) finite state machine which realizes exactly the action sequences specified by the regular expression. A simple example is given in section 4.

2.5 Limitations of the finite state approach

The main limitation of the finite state approach to the modelling of protocols is the "state space explosion". A very large number of states are needed to describe a component of a protocol which involves counters or more complex data structures. The number of states is multiplied when the transition diagram of the overall system is constructed (see section 2.3.1). To a certain degree, these problems can be overcome by automated validation tools (5,6).

Another limitation is inherent in the finite state modelling of the transmission medium. This limitation restricts the analysis to situations with only a small number of messages in transit at any given time. In the case of a datagram service or a full duplex link, these conditions are not always satisfied. As an extension to the finite state model described in this paper, regular expressions may be used to model message queues (1) without imposing a limit on the number of messages in transit.

The approach using a high-level programming language with assertions on program variables for the specification and validation of protocols seems to be especially useful in cases where the finite state approach encounters problems. Therefore a unified method combining both approaches has been proposed (11).

3. THE "EMPTY MEDIUM" ABSTRACTION

We call "empty medium abstraction" a simplified view of the overall transition diagram of the protocol, i.e. of the broken box in figure 1, which consists of considering only those states in the product state space of the overall system for which the *medium* component is empty, i.e. no messages are in transit. This abstraction is particularly useful when the protocol is such that only a small number of messages are in transit at any given time, such as

(a) protocols using a two-way alternate mode of transmission (HDX),

(b) protocols using a two-way simultaneous mode of transmission (FDX)

in a conversational mode, i.e. only a small number of messages are outstanding at any time in each direction. Most initialization protocols are of this nature, and in particular the example treated in section 6.

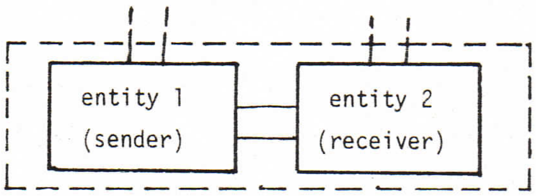


Figure 3 : Architectural structure of a system with direct coupling between the communicating entities.

Since the medium is in the empty state for all states of the protocol considered, the *medium* component is no longer needed for the description of the protocol in the empty medium abstraction, as indicated in figure 3.

Instead, the communicating entities 1 and 2 are directly coupled. The nature of this coupling takes into account the internal structure of the medium. Taking the example of section 2.2 and figure 2, we arrive at a direct coupling between the components *sender* and *receiver* which is characterized by the following pairs of coupled transition types :

(i) $\underline{m}_{sender} || m_{receiver}$ (correct message transmission, transition path in the *medium* : $\underline{m}$.m)

(ii) $\underline{m}_{sender} || E_{receiver}$ (message transmission with error, transition path in the *medium* : $\underline{m}$.error.E)

(iii) $\underline{m}_{sender} || I_{receiver}$ (message loss, transition path in the *medium* : $\underline{m}$.loss).

We note that I denotes the identity transition, i.e. no transition.

In general, the direct coupling between the two entities is characterized by the set of all paths of transitions within the medium component that lead from the empty state back to this state. Let $\sigma = t_m^{(k_1)}.t_m^{(k_2)}...t_m^{(k_n)}$ be the sequence of transition types corresponding to such a path. (In the above example, we may for instance consider $\sigma = \underline{m}$.error.E). Then the direct coupling between the two entities contains all pairs of coupled transition type sequences $\sigma_1 || \sigma_2$, where $\sigma_1 = t_1^{(i_1)}.t_1^{(i_2)}...t_1^{(i_{n'})}$ and $\sigma_2 = t_2^{(j_1)}.t_2^{(j_2)}...t_2^{(j_{n''})}$ are sequences of transition types of the components *entity 1* and *entity 2*, respectively, such that the transition types of the *medium* directly coupled with the *entity 1* or *entity 2* components occur in σ in an order compatible with the order of the corresponding transition types in σ_1 and σ_2 . Without loss of generality, we may assume that the sequences σ_1 and σ_2 begin and end with a transition type coupled to a transition type of σ . (In the above example, we obtain $\sigma_1 = \underline{m}$ and $\sigma_2 = E$) . In general, there may be several pairs $\sigma_1 || \sigma_2$ that correspond to one given σ , because the $\sigma_i (i=1,2)$ may contain intermediate uncoupled transitions.

We note that only a few of all the possible pairs $\sigma_1 || \sigma_2$ are relevant to the communication between the two entities since most of them represent transition sequences which cannot be realized either by *entity 1* or *entity 2*, or both. Examples are given in sections 4 and 6.

4. A SIMPLE EXAMPLE : THE ALTERNATING BIT PROTOCOL

We consider the alternating bit protocol of Bartlett et al. (12), and use our own notation (11). The protocol uses the transmission medium usually in a two-way alternate mode. Therefore the empty medium abstraction is suitable for describing the overall transition diagram of the protocol, as explained below.

4.1 Normal operation

If we consider a reliable transmission medium with neither transmission errors nor losses, we obtain a direct coupling between the *sender* and *receiver* components characterized by the pairs

$$\underline{D_0} || D_0 \quad , \quad \underline{D_1} || D_1 \quad , \quad A_0 || \underline{A_0} \quad , \quad A_1 || \underline{A_1} \quad .$$

Based on the transition diagrams of the *sender* and *receiver* shown in figure 4 and assuming initialization in the states 1_0 , one obtains the transition diagram of the overall system shown in figure 5 (fat transitions only).

If we allow for transmission errors, then the following pairs of transition types must be added to the direct coupling :

$$\underline{D_0} || E \quad , \quad \underline{D_1} || E \quad , \quad E || \underline{A_0} \quad , \quad E || \underline{A_1}$$

The corresponding transition diagram is shown in figure 5 (fat and thin transitions).

If, in addition, we allow for message loss, then the following pairs must be added :

$$\underline{D_0} || I \quad , \quad \underline{D_1} || I \quad , \quad I || \underline{A_0} \quad , \quad I || \underline{A_1}$$

The corresponding transition diagram is shown in figure 5 (including the dashed transitions). The message loss transitions lead to deadlock states. To recover from these situations, a time-out transition is provided in the *sender* component, which is supposed to trigger only after a D or A message has been lost, as indicated in figure 5.

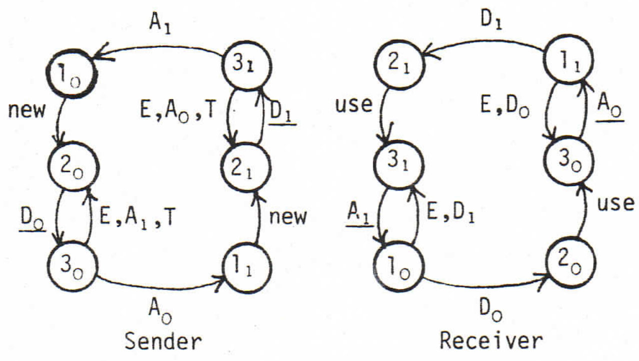


Figure 4 : State diagram of *sender* and *receiver* components

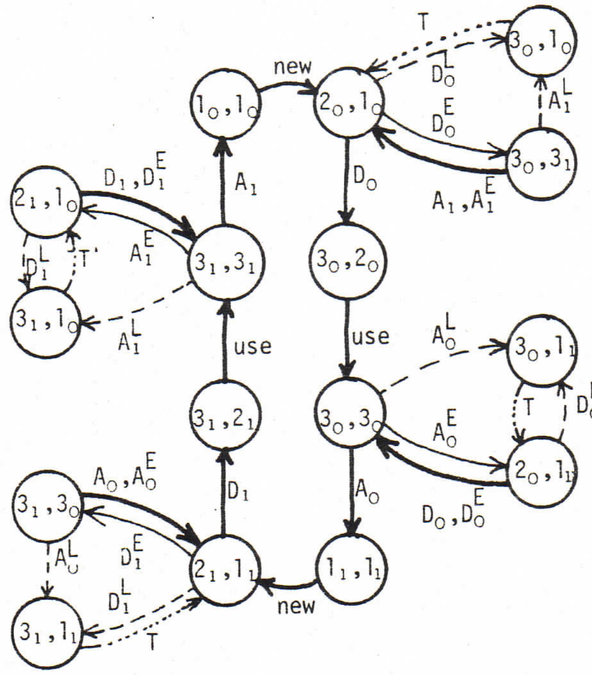


Figure 5 : Transion diagram of the overall system. (Notation: D_1 stands for $\underline{D_1} \| D_1$, D_1^E for $\underline{D_1} \| E$, D_1^L for $\underline{D_1} \| I$, etc.)

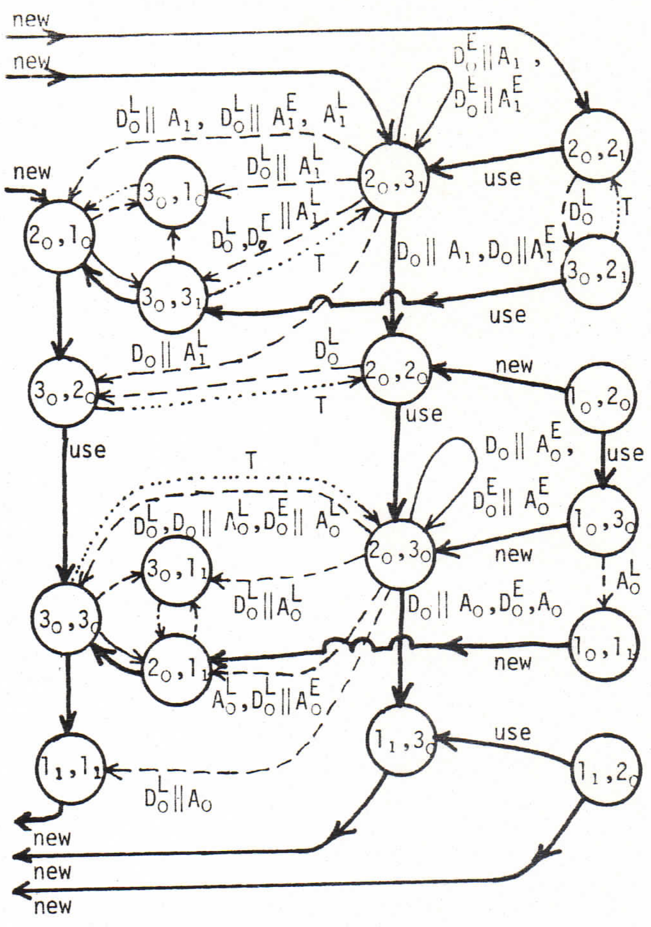


Figure 6 : Complete transition diagram of the overall system (only the part corresponding to the right part of figure 5 is included). Notation : same as in figure 5, and $D_0 \| A_0$ stands for $\underline{D_0}A_0 \| \underline{A_0}D_0$, $D_0 \| A_0^E$ for $\underline{D_0}E \| \underline{A_0}D_0$, etc.

4.2 Self-synchronization

To check the self-synchronization of the protocol, the overall transition diagram has to be constructed for arbitrary initial states, as shown in figure 6. One sees by inspection that the protocol is self-synchronizing. It is interesting to note that, during reliable transmission, a second cycle of operation exists. This cycle (through the states $< 2_0,3_1 >$, etc.) is characterized by simultaneous message transmission through the medium in both directions. In practice this cycle would not be very stable, since message loss could easily occur due to a small mismatch of the processing speeds of the communication components, and would lead back to the normal cycle of operation.

To describe the transitions involving simultaneous message transmission in both directions, as shown in figure 6, we have to introduce the following additional pairs of transition types for the direct coupling of the *sender* and *receiver* components :

$\underline{D}_x A_y \| \underline{A}_y D_x$ (correct transmission) , $\underline{D}_x E \| \underline{A}_y D_x$, $\underline{D}_x A_y \| \underline{A}_y E$, $\underline{D}_x E \| \underline{A}_y E$ (transmission errors) , $\underline{D}_x \| \underline{A}_y D_x$, $\underline{D}_x A_y \| \underline{A}_y$, $\underline{D}_x \| \underline{A}_y$ (loss) , $\underline{D}_x E \| \underline{A}_y$, $\underline{D}_x \| \underline{A}_y E$ (transmission error and loss), where x and y take the values 0 and 1.

If the time-out value in the *sender* is not properly adjusted, or no maximum response time can be established, it is possible that the time-out transition of the *sender* will occur before the expected response of the *receiver* actually arrives. This may lead the system from the normal operation cycle onto the second cycle mentioned above. The dotted transitions in figure 6 are those transitions of the overall system which are introduced by assuming no real-time relation between the time-out transition of the *sender* and the other transitions of the system. The transitions shown are those involving at most one outstanding message in each direction. We note that a very short time-out period may lead to cyclic retransmission and several messages in transit (if this is supported by the medium). Although not shown in the figure, these complex transitions do not invalidate the conclusions below.

4.3 Abstraction

From the point of view of the user of the protocol (the next higher protocol layer) only the transitions *new* and *use* are of interest. By inspecting figure 6, it is clear that, after proper initialization of the system, the only possible action sequence involving the *new* and *use* transitions only is of the form $(new.use)^\infty$, i.e. a regular alternation between *new.use.new.use*...etc. Therefore the transition diagram of figure 7 is a suitable abstraction of the protocol for the next higher protocol layer.

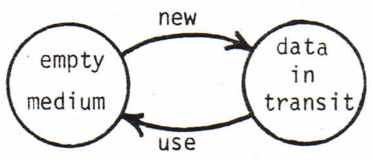


Figure 7 : Finite state model of the service provided to the higher level by the alternating bit protocol (abstraction of figure 5).

To prove that the transition *use* actually submits the data that has been obtained during the transition *new*, it is sufficient to show that between a transition *new* and the consecutive transition *use* of the overall system, there is a correct reception of a D message at the *receiver*. This is obvious from figure 6. We note that the analysis of action sequences (see section 2.3.6) provides a more general proof method.

It is also clear from inspection of figure 6 that the protocol may initially introduce up to two spurious data packets or loose one if it is not properly initialized.

5. ADJOINT STATES

Considering the protocol model of figure 1, we define for each state s_1 of the component *entity 1* the adjoint states of s_1 to be all those states s_2 of the component *entity 2* for which there is a state $< s_1, s_m, s_2 >$ of the overall system (where s_m is some state of the *medium*) which is reachable from the initial state by arbitrarily long paths, i.e. for any given L , there is a path longer then L which leads from the initial state to $< s_1, s_m, s_2 >$.

Knowledge of the adjoint states gives information about the relative synchronization between the components *entity 1* and *entity 2*. When the component *entity 1* is in a given state s_1 , all it knows about the state of *entity 2* is that the latter is in one of the adjoint states of s_1 (9). Clearly, the adjoint states may be obtained from the overall transition diagram of the protocol.

In the case of the "empty medium" abstraction or, more generally, in the case of two directly coupled components, the adjoint states satisfy the following property which allows their determination without the construction of the overall transition diagram. Writing $adj(s_1)$ for the set of adjoint states of s_1 , the property may be stated as follows :

$s_2' \varepsilon\ adj(s_1')$ iff

exist
$s_1, s_2, t_1 \| t_2$ such that $s_2 \varepsilon\ adj(s_1)$, $s_1 \xrightarrow{t_1} s_1'$
and $s_2 \xrightarrow{t_2} s_2'$

or exist
s_1, t_1' such that $s_2' \varepsilon\ adj(s_1)$ and $s_1 \xrightarrow{t_1'} s_1'$

or exist
s_2, t_2' such that $s_2 \varepsilon\ adj(s_1')$ and $s_2 \xrightarrow{t_2'} s_2'$,

where s_i and s_i' are states of component $i(i=1,2), t_1 \| t_2$ is a pair of directly coupled transition types, and t_i' are uncoupled transition types of component $i(i=1,2)$.

This property may be transformed into a set of equations of the form

$$adj(s_1^{(i)}) = \bigcup_j F_{ij}(adj(s_1^{(j)}))$$

where
$$F_{ij}(X) = \bigcup_{t_1 \| t_2, s_2} \left\{ s_2' \middle| s_1^{(j)} \xrightarrow{t_1} s_1^{(i)} \wedge s_2 \varepsilon X \wedge s_2 \xrightarrow{t_2} s_2' \right\}$$
$$\cup \bigcup_{t_2', s_2} \left\{ s_2' \middle| s_2 \xrightarrow{t_2'} s_2' \wedge s_2 \varepsilon X \right\}$$
$$\cup\ X \text{ if exists } t' \text{ such that } s_1^{(j)} \xrightarrow{t'} s_1^{(i)}$$

and $s_1^{(i)}$ is the i-th state of component 1. Such a system of equations may be solved by recursive substitution. Using as initial values $adj(s_1^{(i)}) = s_2^{(k)}$ and $adj(s_1^{(j)}) = \emptyset$ for all $j \neq i$ yields as the solution, the adjoint states for the case where the components are initialized in states $s_1^{(i)}$ and $s_2^{(k)}$, respectively. Using as initial values $adj(s_1^{(i)}) = \{all\ s_2\}$ for all i yields as the solution the adjoint states for the case in which nothing can be assumed about the initialization. Examples are given elsewhere (1).

6. AN ANALYSIS OF THE X.25 CALL SET-UP AND CLEARING PROCEDURES

Some results of an analysis of the X.25 call set-up and clearing procedures have been presented previously (13) . This section gives more details about this analysis, removes some restrictions of the previous work, and shows how the principles outlined in the above sections apply to the analysis of the X.25 procedures. It is also shown that the procedures are stable enough for working in an environment where packets are occasionally lost.

6.1 The specification of the protocol

The following analysis is based on the protocol defined by figure 8. Contrary to the official X.25 document (14) which uses a single diagram for specifying the procedures, we use two distinct transition diagrams for the two communicating components, one for the DTE and one for the DCE. The advantage of our approach has been pointed out previously (13).

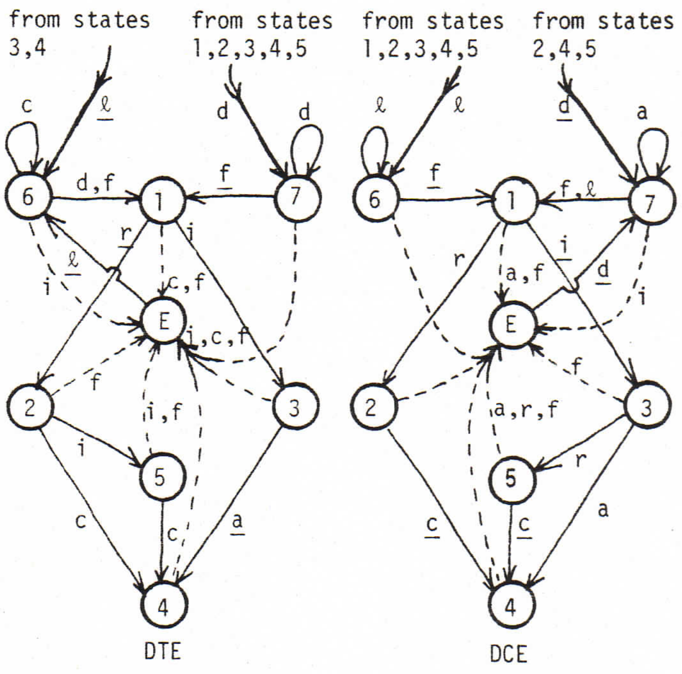


Figure 8 : State diagram of the X.25 call set-up and clearing procedures. Notation :

r	call request	i	incoming call
a	call accepted	c	call connected
ℓ	clear request	d	clear indication
f	clear confirmation		

The protocol of figure 8 corresponds to the specifications of X.25, with the following restrictions :

a) A *clear request* and a *clear indication* may not be sent from the states 1, 2, 5 and 1, 3 respectively. There are two reasons for this restriction : (i) it simplifies the analysis (excluding the repeated *clear request* and *clear indication* transitions in the states 6 and 7, respectively, makes the protocol conversational, in the sense of section 3), and (ii) it avoids certain protocol instabilities discussed by Belsnes and Lynning (15).

b) The handling of packets received in the error state is not specified in the standard (14). On the contrary to our previous work (13), we assume that packets may be received when the component is in the error state. These packets are ignored.

6.2 The analysis

The protocol occasionally uses the *medium* in both directions simultaneously. A typical example is given in figure 9 which shows the possible transitions starting in the *ready* state < 1,1 > . The example shows the possibility of call collision, and abnormal system states due to packet loss. The figure also demonstrates the simplification obtained by the use of the "empty medium" abstraction.

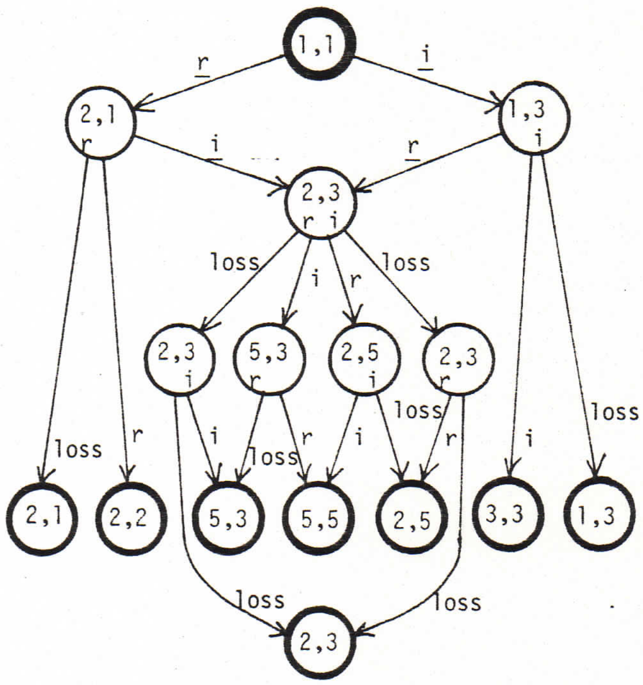


Figure 9 : Partial transition diagram of the X.25 interface, including states with packets in transit (for instance, $\overset{2,1}{r}$ is the state where the DTE is in state 2, the DCE in state 1, and a *call request* is in transit).

The main difficulty of the "empty medium" abstraction is to decide which pairs of directly coupled transition types must be included in the analysis for completeness, and which pairs are redundant for the analysis. For example, in order to take care of the possible two-way simultaneous operations, the pairs $\underline{m}.m' \| \underline{m}'.m$ for $m \,\varepsilon\, \{r, a, l, f\}$ and $\overline{m}' \varepsilon\, \{\bar{i}, c, d, f\}$ must be included in addition to the primitive pairs $\underline{m} \| m$ and $m' \| \overline{m}'$. But pairs such as $\underline{m}^{(1)}.m'.\underline{m}^{(2)} \| \underline{m}'.m^{(1)}.m^{(2)}$ need not be included because they are equivalent to the successive execution of the transition pairs $\underline{m}^{(1)}.m' \| \underline{m}'.m^{(1)}$ and $\underline{m}^{(2)} \| m^{(2)}$.

The protocol also allows for two consecutive packets to be in transit in one direction. To deal with the possibility of simultaneous packets in the other direction, the transition pairs $\underline{m}^{(1)}.\underline{m}^{(2)}.m' \| \underline{m}'.m^{(1)}.m^{(2)}$ with $m^{(1)}.m^{(2)} = a.l$ or $f.r$, and $\underline{m}.m'^{(1)}.m'^{(2)} \| \underline{m}'^{(1)}.\underline{m}'^{(2)}.m$ with $m'^{(1)}.m'^{(2)} = c.d$ or $f.i$ must be included in the analysis.

In order to take care of possible packet loss, additional pairs of transitions must be considered which are derived from the above by replacing one or more reception transition types by the identity transition (no transition). The annex contains a list of all possible transitions of the overall system in the empty medium abstraction.

6.3 The results

The complete transition diagram of the overall system is too complex to be shown. The adjoint states are given in the table below for the following cases :

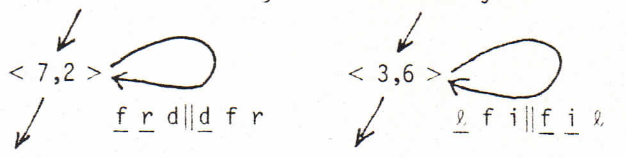
(a) no packet loss
- (a1) with initial synchronization
- (a2) for arbitrary initial states

(b) with possible packet loss, and arbitrary initial states
- (b1) protocol of figure 8
- (b2) protocol of figure 8 with time delay after the transmission of a clear confirmation packet.

Table of adjoint states of the DCE for a given state of the DTE.

	state of DTE 1	2	3	4	5	6	7	E
a1	1	2	3	4	5	6	7	
a2	1	2	3,6	4	5	6	7,2	
b1	1,2, 3,4, 6,7, E	all	1,3, 4,6, 7	1,2, 3,4, 6,7, E	3,4, 5,7	all	1,2, 3,4, 7,E	all
b2	1,3, 7,E	1,2, 3,4, 5,7, E	3	3,4, 7,E	3,4, 5,7	1,3, 4,6, 7,E	7,E	1,3, 4,6, 7

The two unexpected adjoint state in case (a2) are due to the following two undesired cycles



These potentially infinite cycles may in practice "not persist long given the intrinsic variable time delays and any intelligent processing of the clear by either the DTE or DCE" (16). These cycles could be completely avoided by introducing, after the transmission of a clear confirmation packet, a time delay longer than the transmission time of the medium, i.e. the X.25 link access procedure (case (b2)).

We have assumed that a similar time delay is introduced in the error state E before the *clear request*, or *indication* respectively, is sent. If this delay is not present, a station in the error state may send the *clear* before it has received all packets in transit (the other station may have sent two consecutive packets, of which the first has led into the error state). In this way an infinite cycle may occur during which the medium would never be empty, although this is not probable.

The adjoint states for the case (a2) indicate that the protocol is stable (except for the cycles mentioned above). This means that after an initial or intermittent perturbation of the synchronization between the two communicating components, introduced for whatever reason, the protocol recovers into its normal mode of operation.

The adjoint states for the case (b2) indicate that this is also true when the cycles mentioned above are avoided, and packets may be lost. In this case the DTE knows less about the state of the DCE (for most states of the DTE, there are several adjoint states of the DCE), since after one station has sent a given packet, the other may or may not receive it. It is interesting to note that when the DTE is in state 3, the DCE is necessarily in the same state.

The overall states < 2,3 > and < 6,7 > (pairs of adjoint states) are deadlock states. In these states, both components wait for a response, which will never be sent. This situation can possibly be recovered by retransmission after a time-out period, as in the case of the alternating bit protocol discussed in section 4. The same time-out mecanism could also recover from the packet loss leading into the states < 2,1 > , < 1,3 > , etc. We have not including possible retransmissions in the analysis of the protocol.

These results indicate that the X.25 procedures for call set-up and clearing may be used with an underlying link level protocol that simply detects transmission errors, ignores erroneous frames, but does not recover from losses.

Acknowledgements

I would like to thank Jan Gecsei, Bill Armstrong and Mark Gold for many interesting discussions on the subject of this paper, and Don Weir for pointing out an error in an earlier version.