# IMPACT OF QUEUED INTERACTION ON PROTOCOL SPECIFICATION AND VERIFICATION

par

Gregor v. Bochmann* et Alain Finkel**

**DOCUMENT DE TRAVAIL #650**

* Université de Montréal, Département d'informatique et de recherche opérationnelle

** Centre de recherche informatique de Montréal et Université de Montréal

DEPARTEMENT D'INFORMATIQUE ET DE RECHERCHE OPERATIONNELLE

Université de Montréal

JUIN 1988

# Impact of queued interaction on protocol specification and verification

**Gregor v. Bochmann* and Alain Finkel****

* Université de Montréal, Département d'informatique et de recherche opérationnelle.

** Centre de Recherche Informatique de Montréal et Université de Montréal
(Permanent address: University of Paris-Sud, L.R.I., Bat. 490, 91405 Orsay Cedex, France.)

## Abstract:

The paper shows how a newly developed FSM-oriented specification model can be used in the design process of a communication protocol. The impact of queued interactions between different specified processes is the main question addressed in the paper. First, the specification model provides decidable algorithms for analysing most questions of interest for the validation of a protocol, even in the presence of unbounded queues. Second, the impact of queues on the specified system is analysed for a non-trivial example consisting of a simplified Transport protocol. In particular, the impact of queues at the service interface is discussed. It is argued that a service interface is better specified with rendezvous interactions between the service user and provider. The specification model used in the paper allows the combination of queued and rendezvous interfaces within the same system.

## 1. Introduction

For the design and implementation of communication protocols, the establishment of the protocol specification is an important step. The choice of an appropiate specifiation language is an important question, since it has an impact on the ease of understanding of the specification as well as on availability of automated tools for the validation of the specification and the development of implementations.

One of the major reasons why people model communication protocols using automata theoretical techniques is to provide a formal framework in which to analyze the systems. Generally it is the case that detection of design errors with respect to communication protocols corresponds directly to solving one or more of the classical analysis problems with respect to the automata theoretic model. Systems of Communicating Finite State Machines (CFSMs) are very popular among researchers concerned with communication protocols: CFSMs are often used for the modelling [Sunshine 81], analysis [Bochmann 78], [Brand...83] [Gouda...87] and synthesis [Zafiropoulo...80] of communication protocols.

When the reachability set of a system of CFSMs is finite we can, at least theoretically, verify the traditional properties such as deadlock freedom, quasi-liveness, finite termination ... But if the reachability set is infinite, verification of these properties are in general impossible. It is reasonably easy to show that simple systems of CFSMs can simulate Turing machines; thus the analysis problems are in general undecidable [Brand...83]. If we consider CFSMs interacting with each other using CSP-type input/output operations (rendezvous), there exist algorithms to verify many properties we want. As a matter of fact CFSMs whose channels are bounded FIFO or rendezvous channels are essentially equal to finite automata: but the counterpart is the difficulty to simulate unbounded FIFO queues or the "test à zéro". Recall that there can be no algorithm to analysis problems even for the class of CFSMs whose FIFO alphabets are of size at most two with one unbounded FIFO queue [Brand...83] [Rosier...86]. CFSMs whose FIFO queue alphabets are of size one are easily simulated by a Petri net hence almost analysis problems are decidable [Brauer 86]. But for restricted models of CFSMs such as linear systems [Gouda...87] [Choquet...87] it is possible to decide some of these properties. The reduction of the number of states is achieved with the help of a coverability tree [Karp...69] [Finkel 87].

One issue which usually arises in the design and implementation of communication protocols has to do with collision of messages at the same process concerning different requests, usually related to cross-over of messages within the communication medium. Most protocols include certain features to resolve such conflict situations. Sometimes cross-over situations also occur at local interfaces between processes due to the introduction of queued process communication. It is therefore sometimes proposed to use rendezvous communication for local interactions, which avoids such cross-over situations. Since certain FSM-based specification models are based on queued interactions, their adoption has an impact on the cross-over issue.

This paper presents a newly developed theoretical model [Gouda...87] [Choquet...87] which provides decidable algorithms for many interesting analysis problems and applies it to the design and verification of a non-trivial protocol example, namely the OSI Transport protocol with multiplexing. The protocol specification developed here does not cover the data transfer phase in detail, but concentrates on the connection establishment and clearing phases. It is influenced by a more complete specification given in [Boch 87k]. Another verification of a simplified Transport protocol was given in [Bert 82] using the Petri net formalism. The formalisme used here is different in several respects. This paper concentrates on the aspects of multiplexing of several connections and on questions related to the design of local service interfaces, in particular the avoidance or handling of cross-over situations.

Section 2 gives an introduction to the theoretical model. The discussion of the Transport example is covered in three sections: Section 3 gives an overview of the connection establishment and clearing phases and Section 4 deals with the interface with the Transport service user. Each section also includes the discussion of the verification of the described protocol aspects.

## 2. Specification model: interacting finite state machines

### 2.1. System of FSMs communicating by FIFO or rendezvous channels

**Definition 2.1**: A **finite state machine** (FSM) is a quadruplet $M = \langle S,T,h,e_0 \rangle$ where $S$ is the finite set of states, $T$ is a finite set of transitions, $h$ is a partial function from $S \times T$ into $S$ and $e_0$ is the initial state.

**Definition 2.2**: A **labelled FSM** is a triple $(M, X, l)$ where $M$ is a FSM, $X$ is a finite alphabet and $l$ is a morphism from $T$ into $X^*$.

In a system of CFSMs, the finite state machines communicate exclusively by exchanging messages via connecting channels. There are generally two one-directional FIFO or Rendezvous channels between each pair of machines in the system. Each state transition rule is accompanied by either sending or receiving one message to or from one of the output or input channels of the machine. There also can be internal transition without sending or receiving operations. More formally, we have:

**Definition 2.3**: A **Communicating FSM** (CFSM) is a labelled FSM with the alphabet $X = \{+a_1, ..., + a_n\} \cup \{-b_1, ..., -b_p\} \cup \{\lambda\}$

A **global state** $s$ of a system $S$ of CFSMs with FIFO channel is

$$s = (s_1, s_2, ..., s_n, ..., w_{ij}, ...)$$

where $s_i$ is the current state of machine $M_i$ and $w_{ij}$ is the content in the channel, $C_{ij}$, from machine $M_i$ to $M_j$.

**Definition 2.4**: A **global state** $s$ of a system of CFSMs with rendezvous channels is $s = (s_1, ..., s_n)$ where $s_i$ is the current state of machine $M_i$.

A transition $t \in T_i$ of a communicating finite state machine $M_i = (S_i, T_i, h_i, s_{i0}, l_i)$ such that $M_i$ belongs to a system of CFSMs with FIFO channels is **fireable from the global state** $s = (s_i , s_2, ..., s_n , ..., w_{ij}, \cdots )$ in the following three cases:

1. $l_i(t) = \lambda$

2. $l_i(t) = -a$

3. $l_i(t) = +a$ and the input FIFO channel $c_{ji}$ contains a word $w_{ji}$ which begins with a (i,e., $w_{ji} = a \,.\, w'_{ji}$)

The global state $s'$ is then **reached from** $s$ by firing $t$. This new state is defined in the three cases as follows:

1. $s' = (s_i, ..., s_{i-1}, h_i\,(s_{i,} \; t), s_{i+1},..., s_n, ..., w_{ij}, ...)$

2. $s' = (s_1, ..., S_{i-1}, h_i\,(s_i, t), s_{i_1}, ..., s_n, ..., w_{ij}.a, ...)$

3. $s'(s_i, ..., s_{i-1}, h_i(s_i, t), s_{i+1}, ..., s_n, ..., w'_{ji}, ...)$ with $w_{ji} = a.w'_{ji}$

In systems of CFSMs with rendezvous channels, the firing rules are the following:

**Definition 2.5**: A transition $t \in T_i$ of a CFSM $M_i = (s_i, T_i, h_i, s_{io}, l_i)$ such that $M_i$ belongs to a system of CFSMs with rendezvous channel is **fireable from the** global state $s = (s_1, ..., s_n)$ in the following cases

1. $l_i(t) = \lambda$

2. $l_i(t) = -a$ and there exists a machine $M_k$ and a channel between $M_i$ and $M_k$ such that $M_k$ has a transition $t'$ for which $h_k(s_k, t')$ is defined and $l_k(t') = +a$ .

3. $l_i(t) = +a$ and similarly $h_k(s_k, t')$ is defined and $l_k(t') = -a$

The global state $s'$ which is reached from $s$ by firing $t$ in case (1), or $t$ and $t'$ in parallel in cases (2), and (3) is defined as follows:

1. $s' = (s_1, ..., s_{i-1}, h_i(s_i,t), s_{i+1}, ...)$

2,3. $s' = (s_1, ..., s_{k-1}, h_k(s_k, t'), s_{k+1}, \cdots s_{i-1}, h_i(s_i, t), s_{i+1}, \cdots s_n)$

**Definition 2.6**: The **reachability set**, RS(S), from the initial state, is the set of all states which are reachable from the initial state. We associate to a system $S$ of CFSMs a **reachability tree** RT(S). The **language** of a system of CFSMs is defined by $L(S) = \{x \in T^* \,/\, x$ fireable from $e_o\}$; $L(S)$ is the set of words labelling a branch in the reachability tree of $S$. The **input language** of a channel is the set of sequences of messages that may pass through a channel. For CFSMs with FIFO channels, we define bounded channels in the following way. A FIFO channel $C_{ij}$ is **bounded** iff there exists an integer K such that for all global states $s = (s_1, ..., s_n, ..., w_{ij}, ...)$ in the reachability set, the length of $w_{ij}$ is always inferior to K .

**Interesting properties for systems of CFSMs.**

1. The finite termination problem (FTP). Is $L(S)$ finite?

2. The deadlock problem (DP): Is there a finite path in $RT(S)$ that cannot be extended?

3. The finite reachability problem or the boundedness problem (FRP or BP): Is $RS(S)$ finite or is a channel not bounded?

4. The reachability problem (RP): Given a state $s$, is $s \in RS(S)$?

5. The partial deadlock problem (PDP): Is there a finite path to a global state such that the state of one process does not change for all possible extensions.

Note: The CFSM model used in this paper has the property of "blocking receptions", that is, a machine for which an input is present, but no input transition is specified for its present state remains leaves the input in the queue and waits for other transitions to be enabled (the reception is blocked [Zhao 86]). This is in contrast to the model where an "unspecified reception" is considered an error [Zafiropulo...80].

## 2.2. Power of the model

There is a price to be paid for choosing systems of CFSMs with FIFO channels as a modelling tool:

**Theorem 2.1** [Brand ...83][Finkel 86] Systems of CFSMs with FIFO channels have the power of Turing machine when there is at least one unbounded fifo channel involving a messages alphabet of at least two messages.

**Colorraly 2.2** There does not exist an algorithm for CFSMs with respect to any of the aforementioned problems.

**Theorem 2.3** For CFSMs interacting with each other using rendezvous, there exist algorithms to verify the four problems

As a matter of fact, such systems are equivalent to finite automata.

One wants restricted classes of these systems, that leave the FIFO mechanism intact (in some important way), where most if not all of the analysis problems are decidable. The goal is to obtain better tools for the analysis of systems that contain queues. As Pachl observes: "It makes sense to search for a class of CFSMs protocols that is large enough to contain as many common protocols as possible, but small enough to allow automatic verification of deadlock-freedom (and other reachability properties)" ([Pachl 86, p.208]) Typical restrictions that have been considered involve restricting the allowable sequences of messages that can pass through a fifo channel [Rosier...84] [Choquet...87] [Finkel...88] [Pachl 86] [Finkel 88]. Recall that there can be no algorithm for any of the four problems even for the class of CFSMs whose FIFO queue alphabets are of size at most two. CFSMs whose FIFO queue alphabets are of size one are easily simulated by a Petri net for which the four problems are decidable. CFSMs whose FIFO queues are bounded or which communicate by rendezvous are essentially equal to finite automata. Again all four problems admit algorihms with respect to this class. Algorithms tend not to exist for classes that allow one unbounded FIFO queue over a two letter alphabet.

In this paper, we are using a special class of CFSMs called linear CFSMs. They have been first defined in [Gouda...87] and some complementary results are given in [Choquet...87] in the more general framework of Fifo nets [Memmi...85].

**Definition 2.7** [Choquet...87] A system of CFSMs is **linear** iff the input language of each FIFO channel is included in a linear language $a^*_i.b^*_i \cdots z^*_i$ where $a_i, b_i, ..., z_i \in A$ and are different from each other.

**Remark:** In [Gouda...87], it is allowed to have $a_i = c_i$ (for example), but this implies fewer decidable results.

In the general case, there is no algorithm which allows to decide the linearity property. But there exist a decidable sufficiant condition to be sure that a FIFO channel is linear.

**Proposition 2.4.** Let $M$ be a finite state machine which puts messages into a FIFO channel $f$. If the projection, of the regular language of $M$ on the input alphabet of $f$, is a linear language then f is linear. Moreover, this condition is decidable.

**Sketch of proof:** Let us denote by $IA_f$ the input language of $f$. The sufficiant condition is straighforward. Let us show its decidability.

There exist a finite set of linear languages (on a finite alphabet) in which the input language of $f$ must be included if it is linear. If $IA_f = \{a_1, ..., a_p\}$ there exist $p!$ linear languages

$$\{a^*_{i_1}, ..., a^*_{i_p} \,/\, \{i_1, \cdots, i_p\} = \{1, ...,p\}\}$$

Hence, for each linear language $L_j$, $j$=1, ..., $p$ we check whether

$$proj_{IA_f}(L(M)) \subseteq L_j$$

where $proj_{IA_f}$ is the projection on the alphabet $IA_f$.

It is decidable because the inclusion of two regular languages is decidable. Decision problems with respect to linear CFSMs were considered in [Choquet...87], [Choquet 87], where the finite termination problem, the boundedness problem, the deadlock problem and the reachability problem were all shown to be decidable.

**Theorem 2.5** [Choquet...87] The FTP, the DP, the BP and the RP are decidable for linear CFSMs

**Idea of the proof:** shows that the language of every linear CFSMs is equal to the language of some labelled Petri net and given a linear CFSMs illustrates how to effectively construct the corresponding Petri net. The general idea is for the corresponding Petri net to simulate a fifo queue over $a^*_1 a^*_2 \cdots a^*_k$ by $2 * k$ places; $k$ places are used for synchronisation and $k$ places are used to keep track of the number of $a_i s$ , $1 \leq i \leq k$, in the current state. A structured set of terminal markings is then constructed that forces the Petri net to faithfully simulate the linear CFSMs system. The construction is such that the FTP, the DP, the BP and the RP with respect to linear CFSMs can easily be decided. In this paper, we are going to use a generalization of linear CFSMs.

**Definition 2.8**: A system of CFSMs is **semi-linear** iff the input language of each FIFO channel is included in a finite union of linear languages.

**Theorem 2.6** The FTP, the DP, the BP and the RP are decidable for semi-linear CFSMs.

**Idea of the proof:** The language of a semi-linear CFSMs system $S$ is equal to the finite union of the linear CFSMs systems languages which are included in $S$. And linear systems can be analysed with the help of an associated Petri net. Hence by this way the five decidability results can be generalized to semi-linear CFSMs systems.

## 3. Example: Simple Transport protocol

As an example, we consider throughout this paper the connection and disconnection phase of a simplified OSI Transport protocol (class 2), similar to the one of [Boch 87**]. The data transfer phase is further simplified, ignoring the aspects of flow control and acknowledgements. Figure 3.1 shows the structure of communicating FSM's in the simplest case where two user modules establish a Transport connection between one another.

The queues $f_u$ and $f_{u'}$ represent the service interface between a user and its respective AP module. The following service interactions are exchanged:

Queue $f_u$ (from user to AP):

TCONreq: request for a new connection

TCONresp: positive response to TCONind

TDATAreq: data sent by user

TDISreq: request for a disconnection

Queue $f_{u'}$ (from AP to user):

TCONind: indication of a new connection

TCONconf: confirmation of a new connection

TDATAind: data sent to the user

TDISind: indication of disconnection

TDISconf: confirmation of disconnection

The following protocol messages (PDU's) are exchanged between the two peer AP modules (through the queues $f_a$ , $f_{a'}$ shown in Figure 3.1):

CR : connection request

CC : connect confirm

DT : data PDU

DR : disconnect request

DC : disconnect confirm

A specification of the AP modules is given in Figure 3.2. The notation used in this and the following figures is as follows: A single arrow in the figure stands for one or more of the simple transitions defined in Section 2, each of which involve either an input or an output, but not both. The transition labeled "TCONreq/CR", for instance, stands for the succession of two simple transitions, where the first has "TCONreq" as input and the second produces a "CR" PDU as output. The symbol "/" means that the following interaction is output. For example, the transition from state 4 to state 6 in Figure 3.2 consists of four simple transitions, the first with input "DR" and the following producing three outputs "TDISind", "DC" and "TDISconf".

The user module is assumed to exchange interactions with the AP module in a manner consistent with the AP specification. We assume here that the user and AP module communicate through rendezvous. In this case, the specification of the most general user behavior, compatible with the AP module, can be obtained from the AP specification by projection on the interactions at the user interface, as shown in Figure 3.3. A more detailed discussion of the user interface, in particular in the presence of queues, is given in Section 5.

To make automatic verifications of this first protocol, we need algorithms to check deadlock freedom or unboundedness. We are going to show that the six FIFO queues in Figure 3.1. are semi-linear. As $f_a$ has the same input language than $f'_a$, it is sufficient to prove that $f_u, f'_u$ and $f_a$ are semi-linear. In order to determine in which order messages are sent through the queues $f'_u$ and $f_a$, we make projections of the AP specification with respect to the interactions sent over these queues respectively.

For the queue $f_a$, this projection is shown in Figure 3.4(a) which is the projection of the AP module on the alphabet $\{CC,DR,CC,DT\}$. This automaton can be reduced to the equivalent specification of Figure 3.4(b). One easily sees that the possible sequences of messages sent through this queue is described by the following regular expression

CR* [ ( {CC + CR} . DT* . {DC + DR} ) + DR ]

This expression shows that the queue $f_a$ is semi-linear.

For the queue $f'_u$, the expression

(TDISind) * . [ ( (TCONconf + TCONind) . (TDATAind) * . (TDISind TDISconf) + (TCONind . TDISconf) ]

is obtained by the projected state machine (on {TCONconf, TCONind, TDATAind, TDISind, TDISconf}) of Figure 3.5. It is also linear. Concerning the queue $f_u$ from the user, we consider the projection on {TCONreq, TCONresp, TDATAreq, TDISreq} of the user module shown in Figure 3.6. The input language of $f_u$ is equal to

(TCONreq) * . [{TCONresp + $\lambda$ } . (TDATAreq) * ] . TDISreq + (TCONreq)*

Hence $f_u$ is semi-linear.

**Proposition 3.1.** The six FIFO queues are semi-linear.

Given this analysis which shows that the system is semi-linear, we can therefore apply a reachability analysis as described in [Choquet...87]. With the Theorem 2.6, we can analyse this system. The beginning of the reachability tree is shown in Figure 3.7.

It is straightforward to see that a deadlock is attained in the case of a collision of two connect requests initiated by both users simultaneously. The two AP modules are in state 2 (wait for CC). There is a deadlock because each of them is waiting for a message CC from the other. In general, such deadlocks in semi-linear systems can be detected by the coverability tree of the associated Petri net.

In general, the case of collision between two requests initiated by different modules may be resolved in one of the following approaches:

(a) One module aborts is own request and accepts the request from the other module.

(b) The other module aborts its request and accepts the request of the first module.

(c) Both sides abort their proposed action and start again, possibly after a random timeout to avoid races.

(d) If the two requests can be satisfied by some kind of compromise, this compromise is accepted by both sides. Note that approaches (a) or (b) are realized by the protocol design method described in [Gouda 84].

For the connect request collision in our example, the approach (d) would correspond to the establishment of a single connection between the two user modules in response to their requests. This approach is, however, in contradiction to the OSI protocol specification which foresees the establishment of two independent connections in this case. The consideration of multiple connections goes beyond the scope of this article. The handling of this problem and the avoidance of the deadlock mentioned above, is described in [Boch 88] using a formalism related to the one used in this paper.

The approaches (a) or (b) to the above collision problem would mean that the two AP modules do not have the same behavior, thus leading to a non-symmetric protocol. The asymmetry could for instance be based on the different Network addresses of the two AP modules, however, this would again be in contradiction to the OSI Transport protocol specification and is therefore not further explored.

Approach (c) is the least efficient among the possibilities mentioned above. It is therefore usually not taken.

## 4. Modelling the communication service interface

In this section we discuss issues related to the local interface between two state machines. We consider the interface between the protocol entity and the service user, in particular. One of the issues is whether queued interactions or rendezvous is more appropriate. Using the same example discussed above, it is argued that a rendezvous interface is more appropriate for describing the interface at a high level of abstraction. The introduction of queued interactions requires the specification of additional design choices for the case of cross-over of interactions at the interface, which seem to be implementation oriented.

### 4.1. Interface with rendezvous interactions

Figure 3.3 shows a FSM for the user module which is obtained from the specification of the AP module given in Figure 3.2 by projection [Merl 83] on the set of interactions taking place at the service interface with the user. This FSM represents the image of the protocol entity as seen by the user through the interface if interactions proceed through general, however, a queued interface allows for other event sequences were interactions remain in the queue while other interactions are sent. In certain cases, even an unlimited number of interactions may be stored in a queue; this is for instance the case for TDATA interactions between the AP and user modules.

The queued interaction at an interface may also give rise to the possibility of collisions, as already discussed in Section 3 for AP-to-AP interactions. In the case of our example, collisions of interactions at the interface may occur for the connection phase when a connect request from the user and a connection indication from the entity occur "simultaneously". The two users send the TCONreq message; then the $AP_1$ module receives TCONreq and sends CR; the $AP_2$ module receives the CR message, sends a TCONind message to user$_2$ and it goes into state 3, called "wait for TCONresp". The user$_2$ is in state 2 and waits for a TCONconf; the TCONind interaction cannot be received by user$_2$ which remains blocked. Also the $AP_2$ module is waiting indefinitely for the TCONresp interaction from user$_2$. We have a deadlock. Corresponding to the four approaches mentioned in Section 3, the following scenarios may be followed for handling the situation:

(a) The user aborts its request and accepts the connect indication from the server.

(b) The server aborts the connection for which the indication was given and accepts the new request from the user.

(c) Both sides abort their proposed action and start again, possibly after a random timeout to avoid races.

The approach (d) is not feasable, since in general the parameters of the two colliding connections are incompatible (e.g. different remote addresses).

Figure 4.1 shows the added transitions (as dotted arrows) in the specifications of the user and transport entity for the case that the approach (a) is taken. The approach (b) could have been chosen as well. The authors consider that the choice of one of these approaches represents an implementation choice. Therefore such a choice should not be made in the protocol specification.

Similar collision cases occur for the disconnection phase when the user and the protocol entity "simultaneously" initiate a disconnection. Additional inputs that must be expected for these cases are indicated (as dashed arrows) in Figure 4.1. In this case, the approach (d) above can be taken, since the colliding requests imply the same action. Therefore the colliding request can be simply ignored, as indicated in the Figure.

## 4.3. Reinitialization

The specification discussed so far has a "home state" which is a state that can be reached from every reachable state. From a theoretical point of view, the home state problem seems to be decidable because it has been proved decidable for Petri nets [Frutos 87].

The home state of our example is the global state were all queues are empty and all AP modules are in state 6 ("closed"). In practice, however, we are interested in a specification were the intial state is a home state such that an unlimited number of Transport connections can be established successively. Such a specification can be obtained by creating a new module for each new connection to be established (note that the Transport specification in [TP FD 85] uses this approach). Remaining in a context were the structure of the communicating FSM's is determined statically, as discussed in this paper, an equivalent specification can be obtained by identifying the final state 6 (closed) of the AP module with its initial state 1 (closed). One reason why this approach was not taken in the example of Figure 2.2 is the fact that with such an identification the example does not satisfy the semi-linearity conditions (see Definition 2.8) which are the basis for the analysis discussed above.

Looking at the specification of Figure 4.1 from this point of view, we notice that the queues between the AP module and its user module are not necessarily empty when the modules are in their final state 6, since a TDISreq may remain in the queue in the case of the disconnect collision case discussed above. If the final and initial states of the AP module are identified, this interaction may be interpreted by the AP module as a response to a new TCONind which would be a mistake. The following approaches may be used to solve this problem:

(a) Model extension: It is possible to extend the specification model by introducing an additional RESET operation on the queues which can be invoked by the two FSM's associated with the queue and having the effect of eliminating all interactions in the queue, leaving it in the empty state [Ansa 84]. This extension gives the power of Turing machines to the model of semi-linear systems.

(b) Redesign service interactions: It is noted that a similar problem does not occur between two AP modules for the disconnection interactions at the PDU level (a DR received after the sending of a DR is interpreted as a DC). A similar approach can be used for the disconnect interactions at the service level. For instance, the disconnect interactions could be defined to follow the OSI convention of "confirmed" service elements [], which means that a TDISreq is followed by a TDISconf (as in Figure 4.2) and a TDISind is followed by a TDISresp sent by the user module. In the case of disconnect collision, the TDISreq and TDISind received could be interpreted as TDISresp and TDISconf, respectively, by the AP and user modules.

This redesign of the service interface (point (b) above) is not perfect, however. After the sending of a TDISresp, the user module may immediately invoke a new TCONreq which may arrive at the AP module before the DC PDU is received from the peer side. The processing of the new TCONreq should wait until the old connection has been completely closed. This problem, again, can be solved by different approaches:

(i) Using a FSM model with blocked receptions (see Section 2.1). In this case the TCONreq cannot be received in the "wait-for-DC" state (5) and remains in the queue until the AP module reaches the initial "closed" state. This is the model assumed in this paper; it is also used in Estelle [Este 87] and can be modelled in SDL [SDL 87] by be SAVE construct.

(ii) Redesigning the service interface in order to introduce explicit flow control on new TCONreq interactions. This is the approach which originally lead to the use of a TCONconf after a TDISind in Figure 3.2 (see also [Boch 87K]). Combined with the redesign above, this would lead to an interface as defined in Figure 4.2 except that after a TDISind, an additional TDISresp should be returned by the user before the AP terminates by a TDISconf. With this approach a new TCONreq is only received by an AP module, after it has sent a TDISconf.

## 6. Bibliography

[Aho...79] A.V. Aho, J.D. Ullman and M.Yannakakis "Modelling communications protocols by automata" in 20th Annual Symp. on Found. of Comp. Sci. (1979).

[Ansa 84] J.P. Ansard, R. Castanet, P. Guitton and O. Rafiq, "Some operational tools in an OSI study environment. Proc. ACM SIGCOMM Symposium. 1984.

[Berthelot...82] G. Berthelot and R. Terrat "Petri nets theory for correctness of protocols" IEEE Trans. on Comm. COM 30 (12), (December 1982).

[Bochmann 78] G. Bochmann "Finite state description of communication protocols" Proc. Comp. Network Protocols Symp., pp. 361-371, Liege, Belgium (February 1978).

[Boch 87k] G.v. Bochmann, "Specifications of a simplified Transport protocol using different formal description techniques", submitted to Computer Networks and ISDN Systems.

[Boch 88 ] G.v. Bochmann and A. Finkel, "Protocol specification and verification using FIFO nets", Technical report in preparation.

[Brand...83] D. Brand and P. Zafiropulo "On communicating finite-state machines" J.A.C.M., Vol. No. 2, pp. 323-342 (April 1983).

[Brauer...86] W. Brauer, W. Reisig and G. Rozenberg (Eds) "Petri Nets: Central models and their properties" Advances in Petri Nets 1986, Part 1, Proceedings of an Advanced Course Bad Honnef, LNCS 255, Springer Verlag (September 1986).

[Choquet...87] A. Choquet and A. Finkel "Simulation of linear fifo nets by Petri nets having a structured set of terminal markings" in 8th European Workshop on applications and theory of Petri nets, Zaragoza, Spain (June 1987).

[Choquet 87] A. Choquet "Analyse et propriétés des processus communiquant par files fifo: réseaux à files à choix libre topologique et réseaux à files linéaires" Thèse de 3-ième cycle, University Paris 11 (September 1987).

[Chow...85] C-H. Chow, M.G. Gouda and S.S. Lam "A discipline for constructing multiphase communication protocols" ACM Trans. on Comp. Syst., Vol. 3, No. 4, (November

1985).

[Courtiat...84] J.P. Courtiat, J.M. Ayache and B. Algayres "Petri nets are good for protocols" ACM 0-89791-136-9/84/006/0066 (1984).

[Este 87] ISO DIS9074 (1987) "Estelle: A formal description technique based on an extended state transition model".

[Finkel 86] A. Finkel "Structuration des systèmes de transitions: applications au contrôle du parallélisme par files fifo" Thèse d'Etat, University of Paris 11, (June 1986).

[Finkel 87] A. Finkel "A generalization of the procedure of Karp and Miller to Well Structured Transition Systems" 14th ICALP, Karlsruhe, RFA (July 1987).

[Finkel...87] A. Finkel and L. Rosier "A survey on fifo nets" Report of the University of Montréal, No. 630, submitted. (October 1987).

[Finkel...88] A. Finkel and A. Choquet "Fifo nets without order deadlock", Acta Informatica 25 (1988).

[Frutos 87] D. de Frutos Escrig "Decidability of the home state property for Petri nets" (1987).

[Gouda 84] M. Gouda and Y. T. Yu, "Synthesis of communicating Finite State Machines..." IEEE Trans. on Com-32, No. 7, (July 1984).

[Gouda...87] M. Gouda, E. Gurari, T. Lai and L. Rosier "On deadlock detection in systems of communicating finite state machines" Computers and Artificial Intelligence, Vol. 6, No. 3., pp. 209-228. (1987).

[Gouda...87] M. Gouda, E. Gurari, t. Lai and L. Rosier "On deadlock detection in systems of communicating finite state machines" Computers and Artificial Intelligence, Vol. 6, No. 3, pp. 209-228. (1987).

[Jurgensen...84] W. Jurgensen and S. T. Vuong "Formal specification and validation of ISO transport protocol components, using Petri nets" ACM 0-89791-136-9/84/006/0075 (1984).

[Lotos 87] ISO DIS8807 (1987), "LOTOS: a formal description technique".

[Memmi...85] G. Memmi and A. Finkel "An introduction to Fifo nets - monogeneous nets: a subclass of Fifo nets" T.C.S. 35 (1985)

[Memmi...86] G. Memmi and J. Vautherin "Analysing nets by the invariant method" L.N.C.S. 254, pp. 300-336.

[Pachl 86] J. Pachl "Protocol description and analysis based on a state transition model with channel expressions" 6th Intern. Workshop on Protocol Specification, Testing, and Verification, Montréal, Québec, IFIP 86, North Holland (June 1986).

[Rosier...86] L. Rosier and H. Yen "Boundedness, empty channel detection, and synchronisation for communicating finite automata" T.C.S. 44 pp. 69-105 (1986).

[Sarikaya...86] B. Sarikaya and G. Bochmann (Editors) "Protocol Specification, Testing, and Verification, 6", IFIP 86, North Holland (1987).

[SDL 87] CCITT SG XI, REcommendation Z.100 (1987)

[Sunshine 81] C. Sunshine "Formal modelling of communication protocols" Computer Networks and Simulation 2, North Holland, Schoemaker (Ed.), (1982).

[TP FD 85] ISO TC97/SC16/SC6, "Formal specification of the Transport protocol", (April 1985).

[Zafiropulo...80] P. Zafiropulo and AL "Towards analyzing and synthesizing protocols" IEEE Trans. on Comm., Vol. COM-28, No. 4, pp. 651-661, (April 1980).

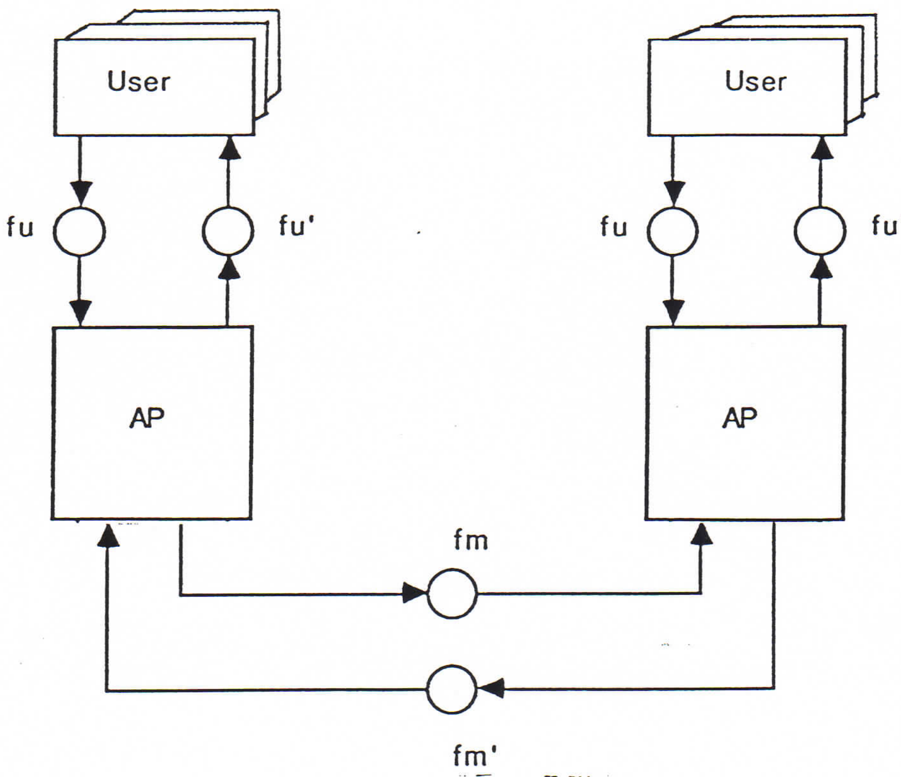

Figure 3.1

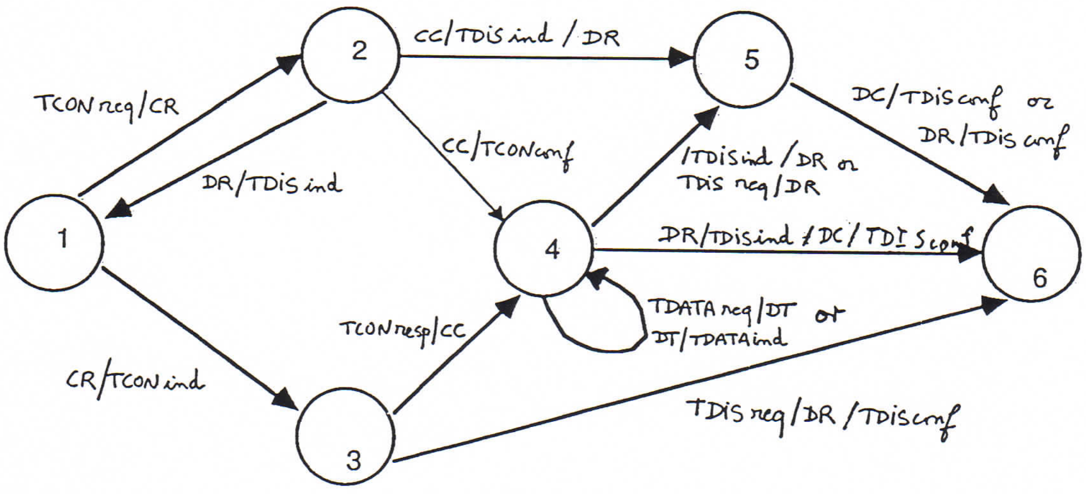

Figure 3.2 *AP module*

1: closed
2: wait for cc
3: wait for TCON resp
4: open
5: wait for DC
6: closed'

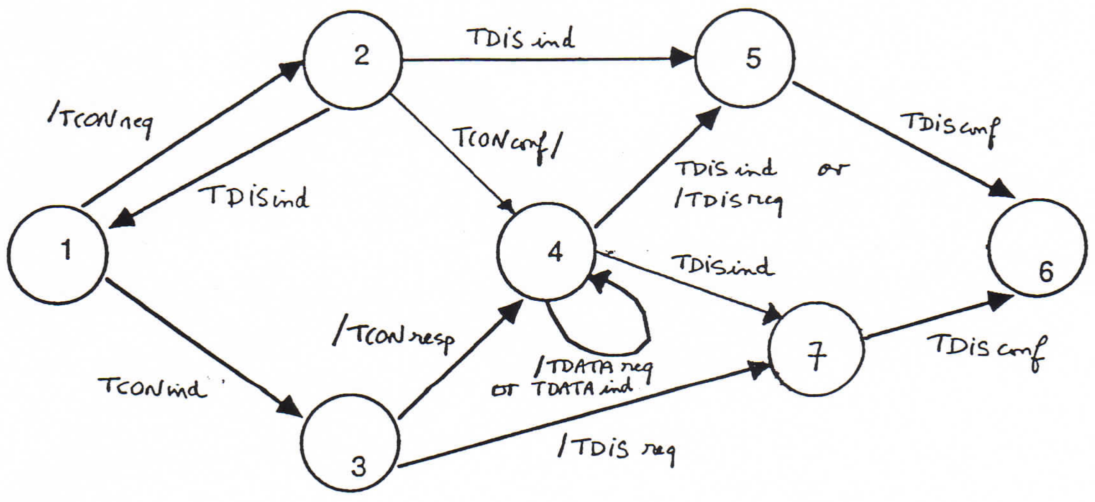

Figure 3.3

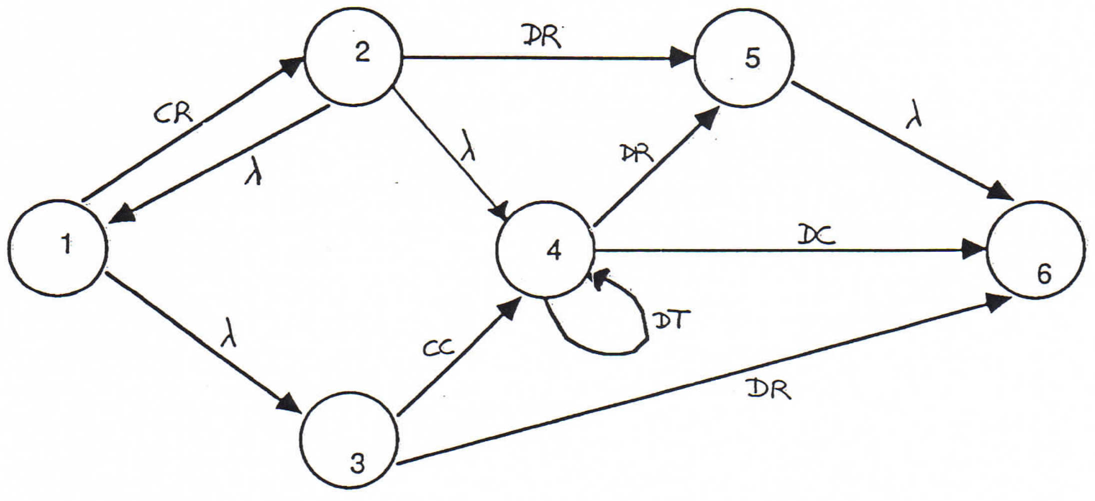

Figure 3.4 (a)

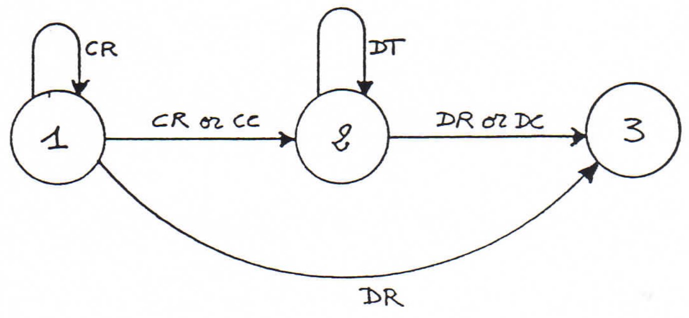

Figure 3.4 (b)

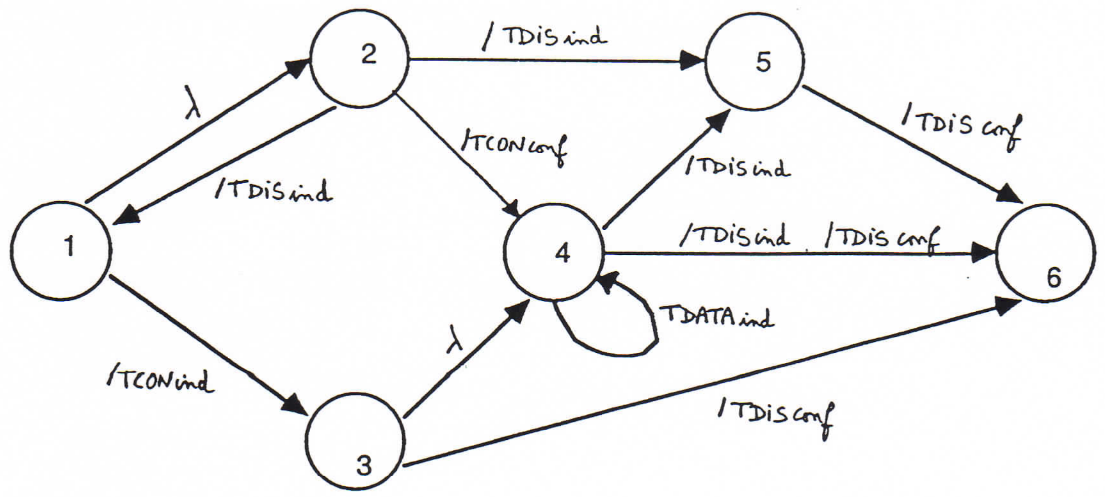

Figure 3.5

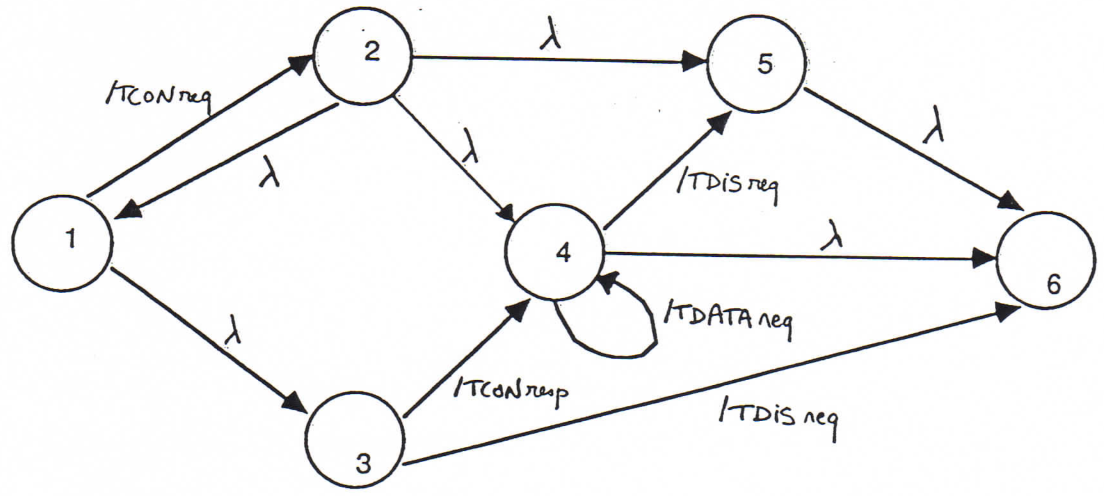

Figure 3.6

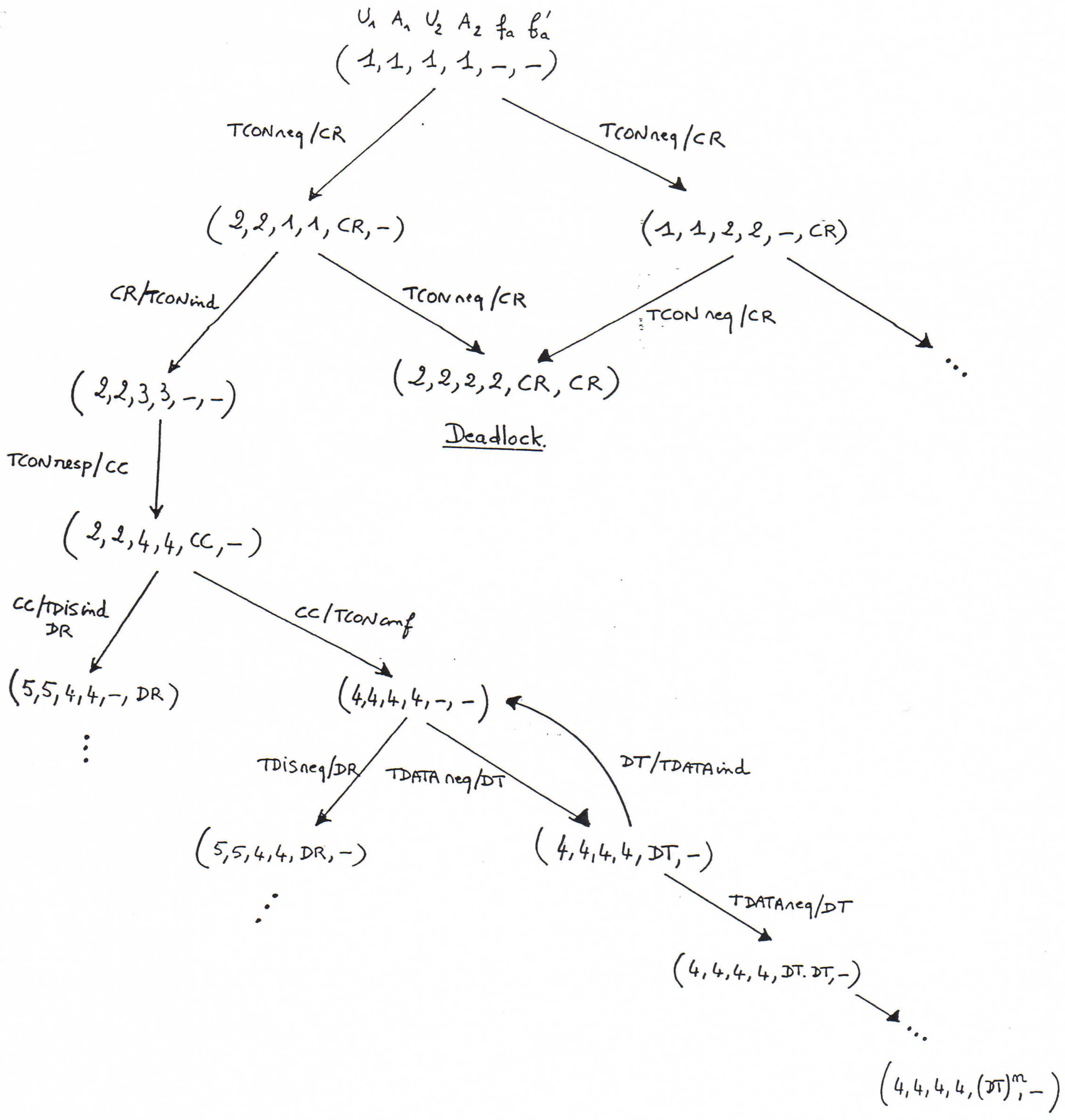

Figure 3.7 A part of the reachability tree (User and AP are synchronized by Rendez-vous)

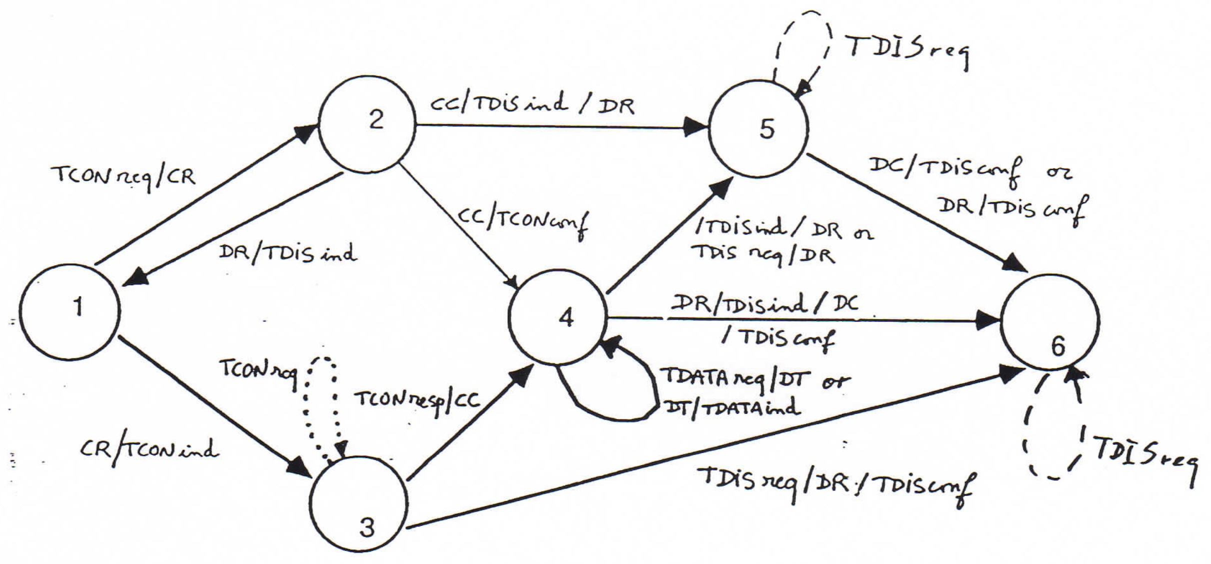

Figure 4.1 (a) AP module

1: closed
2: wait for CC
3: wait for TCONresp
4: open
5: wait for DC
6: closed'

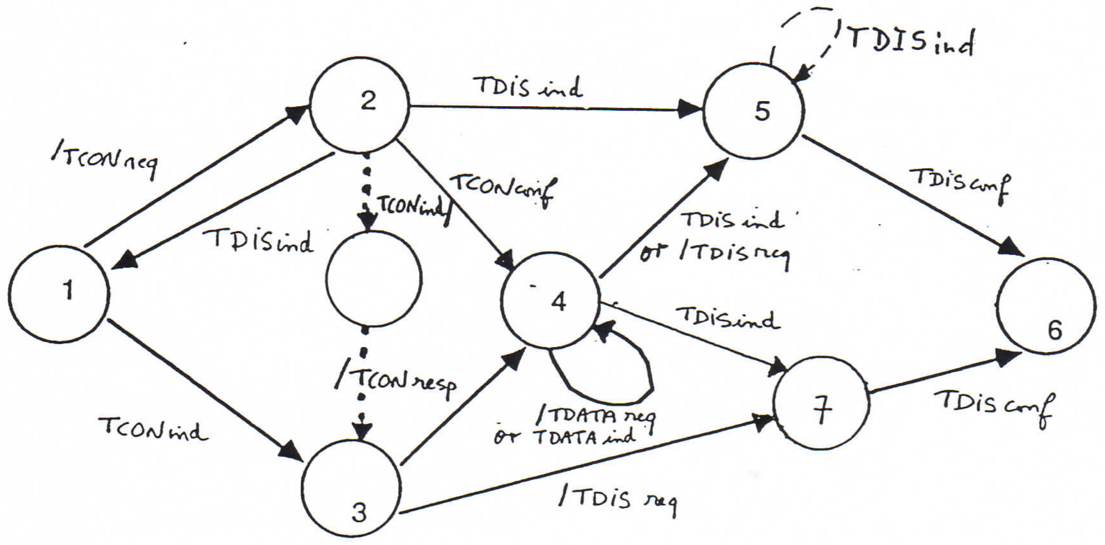

Figure 4.1 (b) user module

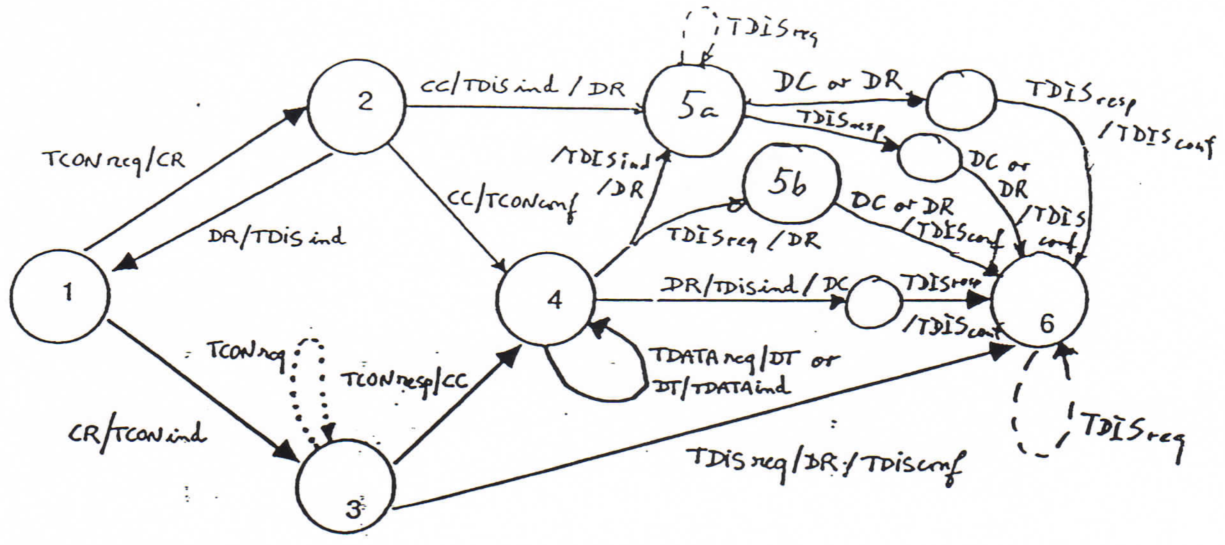

Figure 4.2 (a) AP module

1: closed
2: wait for cc
3: wait for TCON resp
4: open
5: wait for DC
6: closed'

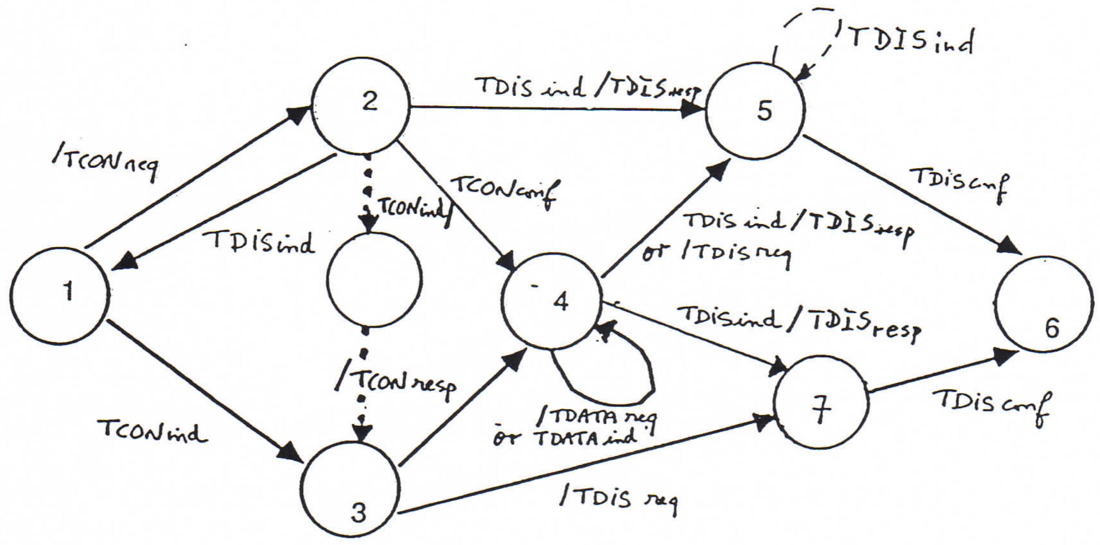

Figure 4.2 (b) user module