On Controlling Distributed Communicating Systems

A. Khoumsi* G.v. Bochmann R. Dssouli

Université de Montréal
Faculté des arts et des sciences
Département d'informatique et
de recherche opérationnelle
C.P. 6128, Succursale Centre-Ville
Montréal, (Québec) H3C 3J7

April 1994

Abstract

In this paper, we propose a new approach for controlling a distributed communicating system (DCS). The aim of the control is to restrict the behaviour of the system for obtaining a desirable behaviour. Traditionally, the control consists in forbidding, when desired, the occurrences of some events. In our study, we consider the fact that a distributed system is necessarily composed by several subsystems which, besides interacting with the environment, communicate with each other via a medium of communication. Therefore, the task the local controllers is not only to prevent the occurrences of some events, but also to exchange a private information via the medium of communication. This new approach is applied to a particular structure of distributed sequential communicating systems.

1 Introduction

A discrete event system (DES) is a dynamic system in which events occur instantaneously, causing a discrete change of the state of the system. In this paper, we consider the case where the sequences of events constitute a regular language. Thus, a DES can be modeled by a FSM. Since DESs need in general to be controlled in such a way to avoid several undesirable sequences of events, a control theory was initiated by Ramadge and Wonham ([12, 8]). This theory has subsequently been extended to encompass other aspects, such as decentralized control which interests us in this study ([2, 9]). This extension arises to consider distributed DESs, while the initial theory failed to resolve problems for networks of communicating processes, which can be modeled as distributed DESs

*Supported by FCAR-NSERC-BNR grant

Among the most interesting works about controlling distributed discrete event systems, there are those presented in [2, 9]. Since the results in [9] are the most general, we consider only this reference. Traditionally, the task of the local controllers is to prevent, when desired, the occurrences of some events. In our study, we generalize the task of the local controllers. For that, we use the fact that a distributed system is composed by several local centralized subsystems which interact with the environment and communicate with each other via a medium supposed reliable. Consequently, in our case the task of the local controllers is to :

- Prevent, when desired, the occurrences of some events (traditional task);
- Exchange a private information with each other via the reliable medium (new task).

The remaining of this paper is organized as follows. Section 2 reviews the part of supervisory control framework needed for our study. Section 3 introduces the structure of the distributed communicating systems considered in our study. This structure is based on service and protocol concepts. In Section 4, the new approach of control is detailed and applied to the structure presented in Section 3. A simple example is given in Section 5. Finally, in Section 6, we conclude and propose some possible extensions. Let's notice that the terms "controller" and "supervisor" will be used as synonyms.

2 Control of discrete event systems

2.1 Centralized control

For a given DES noted M_1 and specified by a FSM S_1, the aim of the control is the following. From a desirable behaviour M_0 specified by a FSM S_0, we have to synthesize systematically the controller noted M_2 such that $M_1 \parallel M_2$ —i.e., M_1 working in parallel and in interaction with M_2— behaves as desired. To achieve the desired behaviour, i.e., to influence the evolution of M_1, the controller has to ([5, 8, 12]):

- *track* the evolution of M_1, by observing occurrences of its events;
- *disable* —i.e., prevent occurrences of— some events when desired.

We consider here only the case where all events of M_1 are observable by M_2, i.e., the supervisor detects occurrences of all events of M_1. Before continuing, let's give the following definitions.

Definition 1 A FSM is defined by $S = (Q, K, \delta, q_0)$ where : **(a)** Q is the set of states; **(b)** K is the alphabet, i.e., the set of events; **(c)** δ defines the transitions, i.e., $\delta : Q \times K \longrightarrow Q$, and $\delta(q, \sigma)!$ means that $\delta(q, \sigma)$ is defined; **(d)** q_0 is the initial state.
Let's notice that δ is also defined for a sequence s of events, i.e., $\delta(q, s)$ is the state reached from the state q after the occurrence of the sequence s, and $\delta(q, s)!$ means that $\delta(q, s)$ is defined.

Definition 2 Let $A = (Q_A, V, \delta_A, q_{A0})$ and $B = (Q_B, V, \delta_B, q_{B0})$ be two FSMs defined over a same alphabet V. A is smaller than B (or B is bigger than A), if all sequences of events executable in A from q_{A0}, are executable in B from q_{B0}. This is noted $A \leq B$. In other words, A is smaller than B if and only if the language L_A accepted by A is included in the language L_B accepted by B.

Definition 3 Let A and B be two FSMs over a same alphabet K, respectively accepting the regular languages L_A and L_B. The sum of A and B, noted $A + B$, is the minimal FSM which accepts the language $L_A \cup L_B$.

Since M_2 has to disable some events of M_1, it is natural to partition the set K of events into *controllable* and *uncontrollable* events: $K = K_{co} \cup K_{uc}$ (Figure 1). A controllable event σ ($\sigma \in K_{co}$) is an event whose occurrence can be prevented by M_2. On the contrary, an uncontrollable event γ ($\gamma \in K_{uc}$) is always enabled by M_2. The DES M_1 to be controlled is then modeled by a FSM $S_1 = (Q_1, K, \delta_1, q_{10})$, and the desired behaviour M_0 is specified by a FSM $S_0 = (Q_0, K, \delta_0, q_{00})$, with $K = K_{co} \cup K_{uc}$.

The desired behaviour specified by S_0 is realizable if and only if $S_0 \leq S_1$ (Def. 2) and S_0 does not necessitate to disable uncontrollable events from S_1. In this case, such behaviour is said *controllable*, w.r.t. S_1. Otherwise, it is said *uncontrollable*, w.r.t. S_1. Let then $Cont_{S_1}(S_0)$ be the set of FSMs which specify the controllable behaviours, w.r.t. S_1, which are smaller than or equal to S_0. Since $Cont_{S_1}(S_0)$ is closed under FSM sum (Def. 3, [12]), we can define the supremal element of $Cont_{S_1}(S_0)$, noted $sup(Cont_{S_1}(S_0))$, which is the sum of all elements of $Cont_{S_1}(S_0)$. Algorithms for computing $sup(Cont_{S_1}(S_0))$ can be found in [5, 12]. Let's notice that if S_0 is controllable, w.r.t. S_1, then $S_0 = sup(Cont_{S_1}(S_0))$.

For obtaining the supremal behaviour specified by $sup(Cont_{S_1}(S_0))$, the supervisor M_2 has to observe the evolution of M_1 and to update the set of allowed events. Therefore, M_2 is specified by $C = (S_2, \Psi)$, where :

- $S_2 = sup(Cont_{S_1}(S_0)) = (Q_2, K, \delta_2, q_{2,0})$.
- $\Psi : Q_2 \longrightarrow 2^K$. For each state q of Q_2, $\Psi(q)$ is the set of events of K, which are enabled by M_2 when M_1 has executed a sequence s from its initial state $q_{1,0}$, such that $q = \delta_2(q_{2,0}, s)$. $\Psi(q)$ contains necessarily K_{uc}, and S_2 is smaller than both S_1 and S_0.

In our case where all events are observable—and then the alphabets of S_1 and S_2 are equal—, computing Ψ from S_2 is self-evident. In fact, if $S_2 = (Q_2, K, \delta_2, q_{2,0})$ then Ψ is formally defined by ([5]) :

$$\forall q \in Q_2 : \Psi(q) = K_{uc} \cup \{\sigma | (\sigma \in K_{co}) \wedge (\delta_2(q, \sigma)!)\}.$$

Example 1 The alphabet of S_1 and S_0 of Figures 2.(a) and 2.(b) is $K = K_{co} \cup K_{uc}$, where $K_{co} = \{a, b, c, d, e\}$ and $K_{uc} = \{\alpha, \beta\}$. S_0 is uncontrollable since it necessitates to disable the uncontrollable events α and β, respectively at states 3 and 4 of S_1. By using the algorithm proposed in [5], we compute the FSM $S_2 = sup(Cont_{S_1}(S_0))$ represented on Figure 2.(c). The behaviour of the controlled M_1 is modeled by S_2 if the supervisor disables the event b at state 1, and the event d at state 2. Therefore : $\Psi(1) = K \setminus \{b\} = \{a, c, d, e\} \cup \{\alpha, \beta\}$, $\Psi(2) = K \setminus \{d\} = \{a, b, c, e\} \cup \{\alpha, \beta\}$, and the supervisor is specified by $C_2 = (S_2, \Psi)$.

2.2 Decentralized control

A distributed communicating system (DCS) is a DES whose events may occur in different sites. The set K of events is then partitionned into $K_1, K_2, \ldots K_n$, where n is the number of sites, and K_i is the set of events occurring in site i. We consider here only the case where :

$$K = \cup_{i=1}^{n} K_i, \text{ and } \forall i, j \leq n : i \neq j \Rightarrow K_i \cap K_j = \emptyset.$$

Intuitively, this means that the different sites are disjoint (Fig. 3). Each element of the alphabet K is defined by e_i, where e is the name of an action and i identifies the site where e occurs. Let's notice that the events of K correspond to both: **(a)** interactions between the DCS and the environment; **(b)** communication between the sites. This fact is detailed in Section 3.

For controlling a DCS M_1, n local supervisors $M_{2,1}, \ldots, M_{2,n}$ may be necessary. Each $M_{2,i}$ can observe all and only the events of K_i and can disable only controllable events of K_i. Therefore, each K_i is partitionned into $K_{i,co}$ and $K_{i,uc}$, i.e., $K_i = K_{i,co} \cup K_{i,uc}$, which respectively represent controllable and uncontrollable events on site i. We also define $K_{co} = \cup_{i=1}^{n} K_{i,co}$, $K_{uc} = \cup_{i=1}^{n} K_{i,uc}$, and then $K = K_{co} \cup K_{uc}$. For a DCS M_1 modeled by a FSM S_1, the aim of a decentralized control is then the following one. From a global desirable behaviour M_0 specified by a FSM S_0, we have to synthesize systematically the local controllers $M_{2,i}$, for $i = 1, \ldots, n$, such that M_1 behaves as desired when it is in interaction with the n local supervisors. Before continuing, let's define different kinds of projections.

Definition 4 Projections

Def. 4.1: (*Projection of an event*). Let σ be an event of the alphabet K. The projection of σ on an alphabet K_i is noted $P_i(\sigma)$ and is defined by :

$$P_i(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in K_i \\ \epsilon & \text{otherwise} \end{cases}$$

where ϵ is an empty sequence (without event).

Def. 4.2: (*Projection of a sequence*). Let s be a finite sequence of events, and let σ be an event. The projection of a sequence on an alphabet K_i is defined recursively by:

$$\begin{cases} P_i(\epsilon) = \epsilon \\ P_i(s\sigma) = P_i(s)P_i(\sigma) \end{cases}$$

Example: if $K = \{a, b, c\}$, $K_1 = \{a, b\}$ and $s = acbabcbcca$, then $P_1(s) = ababba$.

Def. 4.3: (*Projection of a regular language*). If L is a regular language, the projection of L on K_i is defined by: $P_i(L) = \{P_i(s) \mid s \in L\}$.

Def. 4.4: (*Projection of a FSM*). Each FSM A accepts a regular language noted L_A. The projection of A on an alphabet K_i is the minimal FSM noted $P_i(A)$ which accepts the language $P_i(L_A)$. In other words : $L_{P_i(A)} = P_i(L_A)$.

For obtaining a desired and realizable behaviour specified by a FSM S_0, each supervisor $M_{2,i}$ has to :**(a)** observe the local evolution of M_1, i.e., the occurrences of events of M_1 in

site i; (b) update the set of local allowed events, i.e., the set of events which may occur in site i. Therefore, each $M_{2,i}$ is specified by $C_i = (S_{2,i}, \Psi_i)$, where :

- $S_{2,i} = P_i(S_0) = (Q_{2,i}, K_i, \delta_{2,i}, q_{2,i,0})$;
- $\Psi_i : Q_{2,i} \longrightarrow 2^{K_i}$ is defined by $\Psi_i(q) = K_{i,uc} \cup \{\sigma \mid (\sigma \in K_{i,co}) \wedge (\delta_{2,i}(q,\sigma)!)\}$.

Intuitively, when M_1 executes the sequence s and is then in state $q_1 = \delta_1(q_{1,0}, s)$, each supervisor $M_{2,i}$, for $i = 1, \ldots, n$, is in state $q_2 = \delta_{2,i}(q_{2,i,0}, P_i(s))$ and enables events in $\Psi_i(q_2)$.

In [9], it is proven that a desired behaviour S_0 is realizable if and only if S_0 is controllable and n-observable, w.r.t. S_1. The controllability is defined in Section 2.1, with $K_{co} = \cup_{i=1}^{n} K_{i,co}$ and $K_{uc} = \cup_{i=1}^{n} K_{i,uc}$. A formal definition of n-observability is given in [9], and here we give only an intuitive idea. A behaviour S_0 is n-observable if it necessitates that the local supervisors have to make decisions which depends only on what they observe. In other words, every local supervisor $M_{2,i}$ takes a same decision after the executions, from the initial state of S_1, of two sequences s and t such that $P_i(s) = P_i(t)$. If n=2, the n-observability is called coobservability ([9]). If there is only one local supervisor, the n-observability is equivalent to the observability ([5, 8]).

Example 2 The alphabet of S_1 and S_0 of Figures 4.(a) and 4.(b) is $K = K_1 \cup K_2$ with $K_1 = K_{1,co} = \{a_1, b_1\}$, $K_2 = K_{2,co} \cup K_{2,uc}$, $K_{2,co} = \{c_2\}$ and $K_{2,uc} = \{\delta_2\}$. S_0 is not controllable because it necessitates to disable the uncontrollable event δ_2 from state 3. S_0 is not coobservable because $M_{2,2}$ (supervisor in site 2) must enable δ_2 after the occurrence of event a_1 (State 2), and disable the same event δ_2 after the occurrence of sequence $a_1 b_1$ (State 3). This is not possible because $M_{2,2}$ cannot know if M_1 is at state 1,2 or 3, since $P_2(a_1) = P_2(a_1 b_1) = \epsilon$. The supremal controllable behaviour $S_2 = sup(Cont_{S_1}(S_0))$ is represented on Figure 4.(c). In this example, S_2 is realizable because it is controllable and *coobservable*. The local supervisor $M_{2,1}$ has to disable b_1, while $M_{2,2}$ has to enable c_2 and δ_2 (Fig. 4.(f)). $M_{2,1}$ and $M_{2,2}$ are respectively specified by $C_1 = (S_{2,1}, \Psi_1)$ and $C_2 = (S_{2,2}, \Psi_2)$. $S_{2,1}$ and $S_{2,2}$ are represented on Figures 4.(d) and 4.(e), $\Psi_1(1) = \{a_1\}$ and $\Psi_2(1) = \{c_2, \delta_2\}$. In this example, $M_{2,2}$ is not really necessary since it disables no event. Therefore, in general if we obtain a local supervisor $M_{2,i}$ which enables all local events of K_i, then $M_{2,i}$ is not necessary.

In the next section, we introduce the structure of the distributed communicating systems considered in our study. Such structure is based on service and protocol concepts. Afterwards in Section 4, we propose a procedure for controlling a DCS having the structure considered. With this procedure, the local supervisors $M_{2,i}$, not only prevent occurrences of some events, but they also exchange some private information with each other. In this case, the controllable behaviours, which are not realizable by only preventing occurrences of events, become realizable.

3 Distributed communicating structure

3.1 Introduction

A distributed communicating system is a system shared on different sites which can communicate :

- with the user (environment) via servive access points (SAP)
- with each other via a medium.

The medium is supposed reliable since it is not just a physical link, but it also contains all software and hardware tools necessary to hide the unreliability of the physical link. With the 7-layer OSI architecture, the medium provides at least a service of the transport layer ([11]). Therefore, a message sent from a site i to a site j, reaches its destination without being corrupted. Let's notice that the term "user" is used in a general case, i.e., the user represents the environment which interacts with the DCS.

3.2 Service and protocol concepts

Each site i contains a module, called protocol entity and noted PE_i, which : **(a)** interacts with the user of the site i; **(b)** communicates with the medium, to exchange messages with other protocol entities. In the user's viewpoint, the distributed system is globally a black box, where interactions with the medium are invisible (Fig. 5.(a)). Therefore, the specification of the service provided to the user (called *service specification*) defines the ordering of the interactions visible by the user. These interactions are called service primitives. Informally, such specification defines the service provided to (or desired by) the user of the DCS.

In the designer's viewpoint , it is necessary to generate the local specifications of the n protocol entities (Fig. 5.(b)), $PE_1, \ldots, PE_n$ (called *protocol specifications*) from the specification of a desired service. Informally, each local specification of PE_i specifies "what is implemented in site i". An approach which directly generates protocol specifications is called synthesis ([1, 3, 4, 6, 7, 10]). In our present study, we use Finite automata as a formalism of specification, and we consider only sequential systems.

Definition 5 A sequential service is described by a FSM $SS = (Q_s, K_s, \delta_s, q_{s,0})$ which specifies the global ordering of service primitives (SP) observed by the user at the different sites. Events of the alphabet K_s of SS are noted e_i, where e is the name of a service primitive (SP), and i identifies the site where the SP is executed. The occurrence of an event e_i means that the service primitive e is executed in the site i by the protocol entity PE_i.

Example 3 Let the distributed system constituted by two sites and schematized on Figure 6.(a). A formal specification of the service is represented on Figure 6.(b). Four service primitives a, b, c and d are defined, and the alphabet is $K_s = \{a_1, b_1, c_1, d_1, c_2, d_2\}$,

Definition 6 A protocol entity PE_i is described by a FSM $PS_i = (Q_i, K_i, \delta_i, q_{i,0})$ which specifies the ordering of the local interactions with the user and with the medium on site i. The elements of the alphabet K_i, i.e., the events which occur in site i, are of three types.

1. Execution of a service primitive e on site i (interaction with the local user). This event is noted e_i.
2. Sending a message with a parameter p from PE_i to a protocol entity PE_j. This event is noted $s_i^j(p)$.
3. Reception by PE_i of a message with a parameter p coming from a protocol entity PE_j. This event is noted $r_i^j(p)$.

The use of the parameter p contained in a message is explained in Section 3.3. An example of protocol specifications is given on Figure 8 (See Example 4 in Section 3.3).

3.3 Synthesizing the protocol from the service

Before introducing the principle of the procedure for synthesizing the protocol, let's define the global protocol specification.

Definition 7 The global protocol specification is a FSM $GPS = (Q, K, \delta, q_0)$ which specifies the ordering of all events (interactions with the user and with the medium) which occur in the distributed system. Therefore : $K_s \subset K = \cup_{i=1}^{n} K_i$. The service and protocol specifications SS and PS_i, for $i = 1, \ldots, n$, can be obtained from GPS by projections (Def. 4 on Section 2). Let P_s and P_i, for $i = 1, \ldots, n$, be respectively the projections on the alphabets K_s and K_i, for $i = 1, \ldots, n$. Then $SS = P_s(GPS)$, and $PS_i = P_i(GPS)$ (Example 4). Informally, GPS specifies the global structure of the distributed communicating system. An example of GPS is given on Figure 7 (See Example 4).

The aim of synthesis is the generation of the protocol specifications PS_i, for $i = 1, \ldots, n$, from the desired service specification SS. Intuitively, we have to generate what must be implemented in each site from what the user desires. The basic principle used for synthesizing the different PS_i is the following ([1, 3, 4, 6, 7, 10]).

- If, in SS, two consecutive service primitives A and B are executed by two different protocol entities PE_i and PE_j, i.e., if in SS two transitions A_i and B_j, with $i \neq j$, are consecutive, then :
  1. PE_i executes A and sends a message to PE_j. This message is parameterized by the identifier p of the state of SS reached after the execution of A.
  2. When PE_j receives the message parameterized by p, it executes B.
- If after a transition A_i, there is a choice between m transitions Bk_{jk} executed by the protocol entities PE_{jk}, for $k = 1, \ldots, m$, then :
  1. PE_i executes the primitive A and selects one of the protocol entities PE_{jk} which executes one of the primitives Bk.

2. PE_i sends a message to the selected PE_{jk}. This message is parameterized by the identifier p of the state of SS reached after the execution of A.
3. When PE_{jk} receives the message parameterized by p, it may execute Bk.

The parameter p contained in a message is necessary to avoid any ambiguity when a protocol entity receives a message. From this basic principle, several systematic methods of synthesis are developped in the literature, and we propose the one used in [7]. The latter generates, in a first step, the global specification GPS from the specification SS of the desired service. In a second step, PS_i ,for $i = 1, \ldots, n$, are obtained by projecting GPS on the alphabets K_i.

Example 4 From the service specification of Figure 6.(b), the obtained GPS is represented on Figure 7. From GPS, we deduce the two alphabets K_1, K_2 and K as follows :
$K_1 = \{a_1, b_1, c_1, d_1, s_1^2(4), r_1^2(2)\}$; $K_2 = \{c_2, d_2, s_2^1(2), r_2^1(4)\}$; $K = K_1 \cup K_2$. The projections of GPS on alphabets K_i, for $i = 1, 2$, give the two protocol specifications PS_i of Figure 8.

4 Controlling a sequential DCS

4.1 Introduction of the problem

Let a sequential DCS (SDCS) M_1 modeled by a FSM GPS_1, over the alphabet K which contains all interactions with the user and the medium. In the user's viewpoint, M_1 can be modeled by a FSM SS_1 which specifies the service provided to the user. The alphabet K_s of SS_1 contains only interactions with the user, and SS_1 is then such that $SS_1 = P_s(GPS_1)$, where P_s is the projection on the alphabet K_s. As an example, let the SDCS M_1 modeled by GPS_1 and SS_1 respectively represented on Figures 7 and 6.(b).

The problem is then the following. From a specification SS_0 which models a desired service,over the alphabet K_s, the aim is to control the SDCS M_1 in such a way that it provides the biggest realizable service which is smaller than SS_0. The entries of the problem are then : GPS_1, SS_1 and SS_0.

4.2 Approach for the problem

A desired behaviour of a distributed system is realizable, by only preventing occurrences of some events, if and only if it is controllable and n-observable ([9],Section 2.2). Intuitively, a behaviour is not n-observable if the decisions of at least one local controller $M_{2,i}$ do not depend only on what $M_{2,i}$ observes locally, but also on previous decisions of local controllers in other sites. In other words, at least one local supervisor needs additional information for making its decisions. As an example, the behaviour specified by the FSM of Figure 4.(b) is neither controllable nor n-observable (see also Example 3 in Section 2.2).

Since there is a possibility for exchanging information between the sites, via a medium of communication, we propose that the different local controllers, besides forbidding some events, exchange information in such a way that every local supervisor receives all the

necessary information for making its decisions. In this case, a controllable behaviour, which is initially *not* n-observable, can be made realizable by adding to it an exchange of information between the local supervisors.

Let's propose a way for exchanging information between the different local supervisors $M_{2,i}$, for $i = 1, \ldots, n$, in order to control a SDCS M_1. Each $M_{2,i}$ interacts with the corresponding protocol entity PE_i of site i by forbidding some local events, and exchanges some private information with other local supervisors. The private information is transmitted by filtering the messages exchanged between the protocol entities. The message filtering is realized as follows (Figure 9).

Let a protocol entity PE_i which has to send to PE_j a message parameterized by p (event $s_i^j(p)$). The message is intercepted by $M_{2,i}$ which adds a private information m in it, before sending the message to PE_j. Such operation is called *Filtering of a transmitted message*, and is noted $s_i^j(p, +m)$.

When the message reaches the site j, it is intercepted by $M_{2,j}$ which removes the private information m, before giving the message to PE_j. Such operation is called *Filtering of a received message*, and is noted $r_j^i(p, -m)$.

4.3 Synthesizing local supervisors

Before proposing a procedure which generates systematically the formal specifications of the local supervisors $M_{2,i}$, let's define the operator $\otimes$.

Definition 8 Let A and B be two FSMs respectively over the alphabets V_A and V_B, and accepting respectively the languages L_A and L_B. Let $A \times B$ be the synchronized product of A and B. If for instance $V_A = V_B$, then $A \times B$ accepts the language $L_A \cap L_B$.
$A \otimes B$ is the supremal (biggest) FSM which is smaller than $A \times B$ and is free of deadlock.

The proposed procedure for synthesizing the local supervisors is noted *Synt_Loc_Sup* and is composed of the five following steps. The entries of *Synt_Loc_Sup* are the FSMs GPS_1, SS_1 and SS_0 (Sect. 4.1). GPS_1 is defined over the alphabet $K = K_s \cup K_m$, where K_s is the alphabet of SS_1 and SS_0, and K_m contains events corresponding to the communication between the sites.

Step 1. If SS_0 is not smaller than SS_1 or contains deadlocks, then SS_0 is replaced by $SS_0 \otimes SS_1$ (Def. 8).

Step 2. The FSM $GPS_{1,0} = GPS_1 \otimes SS_0$ is computed. $GPS_{1,0}$ models the supremal behaviour of M_1 which provides the desired service (because $SS_0 = P_s(GPS_{1,0})$) and is free of deadlocks.

Step 3. $GPS_{1,0}$ is possibly not controllable, and then not realizable. Therefore, the supremal controllable GPS_c, w.r.t. GPS_1, which is smaller than GPS_0 is computed. This is noted $GPS_c = sup(Cont_{GPS_1}(GPS_{1,0}))$ (Section 2). Let's notice that for computing GPS_c, all the events $r_i^j(p)$ are considered uncontrollable.

Step 4. GPS_c is possibly not n-observable (Sections 2.2 and 4.2), and then not realizable. In this case, it may be impossible to provide the service $SS_c = P_s(GPS_c)$. Therefore, GPS_c is transformed into GPS with the following rules. If $E_{i,j}(p)$ is the sequence composed of event $s_i^j(p)$ followed by $r_j^i(p)$ (Figure 10.(a)), then for any i, j and p, such that $E_{i,j}(p)$ is defined in GPS_c :

Case 1. If $E_{i,j}(p)$ is single in GPS_c then it remains unchanged.

Case 2. If all sequences $E_{i,j}(p)$ lead to a same state of GPS_c (Figure 10.(b)), then they remain unchanged.

Case 3. Otherwise, each $E_{i,j}(p)$ leading to a state identified by m (Figure 10.(c)), is replaced by the sequence of $s_i^j(p,+m)$ followed by $r_j^i(p,-m)$ (Figure 10.(d)), where $s_i^j(p,+m)$ and $r_j^i(p,-m)$ are defined in Section 4.2.

This transformation is noted $Filt$, i.e., $GPS = Filt(GPS_c)$, and is such that $P_s(Filt(GPS_c)) = P_s(GPS_c)$. Informally, the transformation $Filt$ does not change the provided service. Contrary to GPS_c, the behaviour specified by $Filt(GPS_c)$ is always realizable. Intuitively, the private information exchanged between the local controllers $M_{2,i}$ —$s_i^j(p,+m)$ followed by $r_j^i(p,-m)$— eliminates the reazon why GPS_c is not n-observable. In fact, every $M_{2,i}$ will have all necessary information for deciding which local controllable events are allowed.

Step 5. The specifications of the local supervisors are computed by projecting GPS in the alphabets of the different sites. In other words, for each site i, $M_{2,i}$ is specified by $(S_{2,i}, \Psi_i)$ where :

- $S_{2,i} = P_i(GPS) = (Q_{2,i}, K_{2,i}, \delta_{2,i}, q_{2,i,0})$, P_i being the projection on the set K_i of events occurring in site i.
- $\Psi_i : Q_{2,i} \longrightarrow 2^{K_i}$, where $\Psi_i(q)$ is the set of local events allowed when $M_{2,i}$ is in state q. Formally : $\Psi_i(q) = K_{i,uc} \cup \{\sigma \mid (\sigma \in K_{i,co}) \wedge (\delta_{2,i}(q,\sigma)!)\}$.
  In the definition of $\Psi_i(q)$, $\delta_{2,i}(q, s_i^j(p))!$ (resp. $\delta_{2,i}(q, r_i^j(p))!$) means that the event $s_i^j(p)$ or an event $s_i^j(p,+m)$ (resp. $r_i^j(p)$ or $r_i^j(p,-m)$) is executable from the state q of $S_{2,i}$.

5 Example

Let the SDCS to be controlled modeled by GPS_1 of Figure 7, and then providing the service modeled by SS_1 of Figure 6.(b). The desired service is specified by SS_0 represented on Figure 11.
Procedure *Synt_Loc_Sup*

Step 1. SS_0 is smaller than SS_1 and is free of deadlocks, then $SS_1 \otimes SS_0 = SS_0$

Step 2. The obtained $GPS_{1,0}$ is represented on Figure 12 and is not controllable, because it necessitates to forbid the uncontrollable event b_1 at state 3.

Step 3. The supremal controllable GPS_c, w.r.t. GPS_1, which is smaller than $GPS_{1,0}$, i.e., $GPS_c = sup(Cont_{GPS_1}(GPS_{1,0}))$, is represented on Figure 13. In this example, the controllable events are a_1, c_1, d_1 and all events $s_i^j(p)$. Thus, the uncontrollable events are b_1, c_2, d_2 and all events $r_i^j(p)$.
Intuitively, since the uncontrollable event b_1 is forbidden from state 3 of GPS, then the state 3 must be avoided by forbidding the controllable event d_1 from state 2.

Step 4. GPS_c is not coobservable, w.r.t. GPS_1, and then not realizable (Section 2.2). Intuitively, when the message with parameter 2 is received in site 1 (by $r_1^2(2)$), the local controller $M_{2,1}$ cannot know if it corresponds to the event which leads to state 2 or to state 12 (Figure 13). Therefore, $M_{2,1}$ cannot decide to forbid the controllable event c_1.
If we apply the transformation $Filt$ to GPS_c, we obtain $GPS = Filt(GPS_c)$ of Figure 14. Intuitively, $M_{2,1}$ can decide to forbid the event c_1 because the message coming from site 2 contains an information (2 or 12) which did not exist in GPS_c.

Step 5. Each $M_{2,i}$, for $i = 1, 2$, is specified by $(S_{2,i}, \Psi_i)$. The FSMs $S_{2,1}$ and $S_{2,2}$ are represented on Figure 15. Let $K_{i,co}$ and $K_{i,uc}$ be respectively the sets of controllable and uncontrollable events in site i. Therefore, $K_{1,co} = \{a_1, c_1, d_1, s_1^2(4)\}$, $K_{1,uc} = \{b_1, r_1^2(2)\}$, $K_{2,co} = \{s_2^1(2)\}$ and $K_{2,uc} = \{c_2, d_2, r_2^1(4)\}$. The functions Ψ_1 and Ψ_2 are as follows :

$\Psi_1(1) = K_{1,uc} \cup \{a_1\}$, $\Psi_1(2) = K_{1,uc}$, $\Psi_1(3) = K_{1,uc} \cup \{s_1^2(4)\}$, $\Psi_1(4) = K_{1,uc}$, $\Psi_1(5) = K_{1,uc} \cup \{c_1\}$.

$\Psi_2(1) = K_{2,uc}$, $\Psi_2(2) = K_{2,uc}$, $\Psi_2(3) = K_{2,uc}$, $\Psi_2(4) = K_{2,uc} \cup \{s_2^1(2)\}$, $\Psi_2(5) = K_{2,uc} \cup \{s_2^1(2)\}$.

Let's notice that, in this example, the task of $M_{2,2}$ is only to inform $M_{2,1}$ if d_2 has occurred before c_2. If yes, $M_{2,2}$ adds the parameter 12 in the sent message $(s_2^1(2, +12))$. Otherwise, $M_{2,2}$ adds the parameter 2 in the sent message $(s_2^1(2, +2))$.

6 Conclusion

In this study, a new approach is proposed for controlling a distributed communicating system (DCS). Besides preventing occurrences of some controllable events, the task of the local controllers is also to exchange a private information with each other. With this approach, the controllable behaviours of a DCS which are unrealizable when the local controllers do not communicate with other, become realizable. A procedure is proposed for generating automatically the specifications of the local controllers when the DCS to be controlled is sequential. The procedure is applied to a simple example.

For future work, we intend to extend our study by considering the control of distributed systems with *timing requirements*. We also intend to study the control of *concurrent* distributed systems.

References

[1] G.v. Bochmann and R. Gotzhein. Deriving protocols specifications from service specifications. In *Proceedings of the ACM SIGCOMM Symposium*, USA, 1986.

[2] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260, March 1988.

[3] C. Kant, T. Higashino, and G.v. Bochmann. Deriving protocol specifications from service specifications written in lotos+. Technical Report 805, Université de Montréal. Département d'informatique et de recherche opérationnelle, C.P. 6128, Succursale Centre-Ville, January 1992.

[4] F. Khendek, G.v. Bochmann, and C. Kant. New results on deriving protocol specifications from services specifications. In *Proceedings of the ACM SIGCOMM Symposium*, pages 136–145, 1989.

[5] A. Khoumsi, G.v. Bochmann, and R. Dssouli. Contrôle et extension des systèmes à événements discrets totalement et partiellement observables. In *Proceedings of The Third Maghrebian Conference on Software Engineering and Artificial Intelligence*, Rabat, Morocco, April 1994.

[6] A. Khoumsi, G.v. Bochmann, and R. Dssouli. Dérivation de spécifications de protocole à partir de spécifications de service avec des contraintes temps-réel. *Réseaux et Informatique Répartie*, 1994.

[7] A. Khoumsi, G. v. Bochmann, R. Dssouli, and A. Ghedamsi. A systematic and optimized method for designing protocols for real-time applications. Technical Report 900, Université de Montréal. Département d'informatique et de recherche opérationnelle, C.P. 6128, Succursale Centre-Ville, 1994.

[8] P.J. Ramadge and W.M. Wonham. The control of discrete event systems. In *Proceedings of the IEEE*, volume 77, pages 81–98, January 1989.

[9] K. Rudie and W.M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, November 1992.

[10] K. Saleh and R. Probert. A service based method for the synthesis of communicating protocols. *International Journal of Mini and Microcomputer*, 12(3), 1992.

[11] A. Tanenbaum. *Réseaux: Architectures, protocoles, applications.* InterEditions, Paris, 1990.

[12] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J.Control and Optimization*, 25(3):637–659, May 1987.

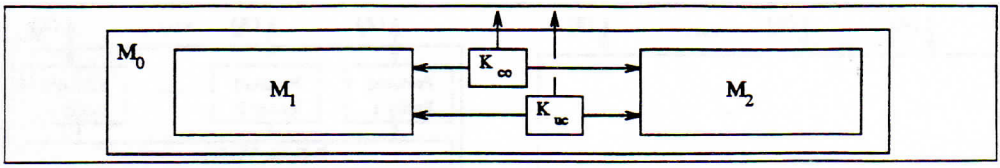

Figure 1: Symbolic representation of controllable and uncontrollable events.

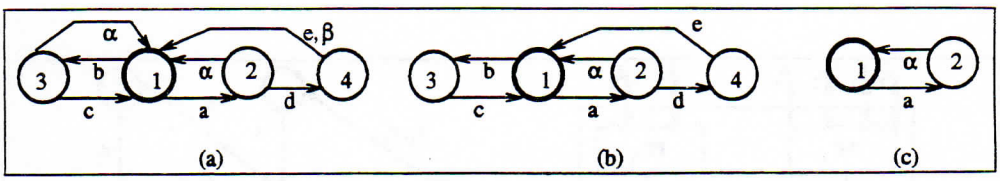

Figure 2: Example of a centralized control. (a) S_1. (b) S_0. (c) $S_2 = sup(Cont_{S_1}(S_0))$.

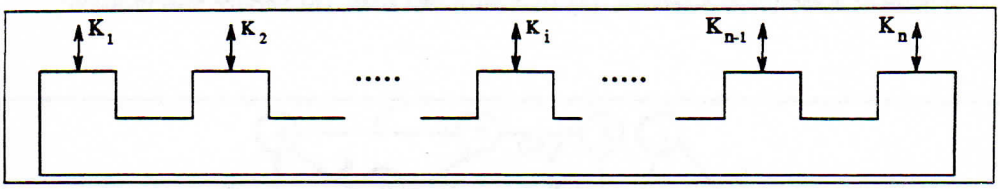

Figure 3: Symbolic representation of a distributed system.

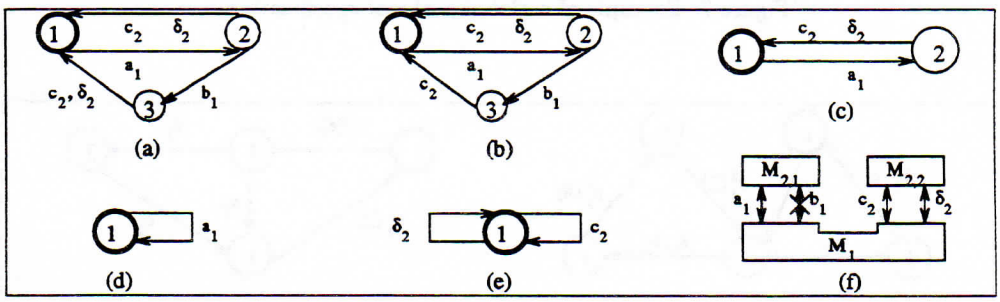

Figure 4: Example of a decentralized control problem. (a) S_1. (b) S_0. (c) $S_2 = sup(Cont_{S_1}(S_0))$. (d) $S_{2,1}$. (e) $S_{2,2}$. (f) Symbolic representation of local supervisors.

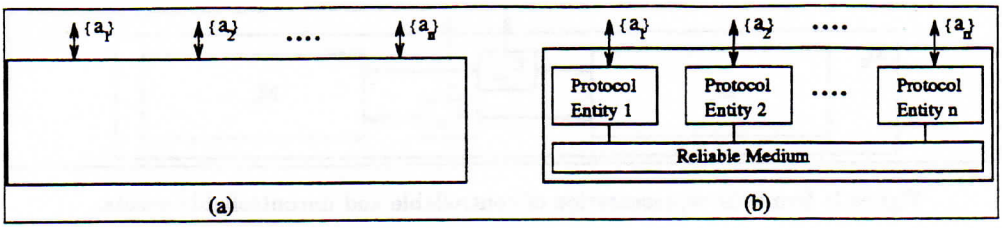

Figure 5: Service and protocol concepts. (a) Service. (b) Protocol.

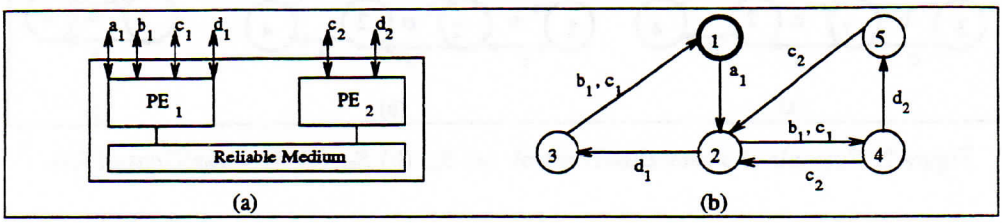

Figure 6: Example of a service. (a) DCS with two sites. (b) Service Specification.

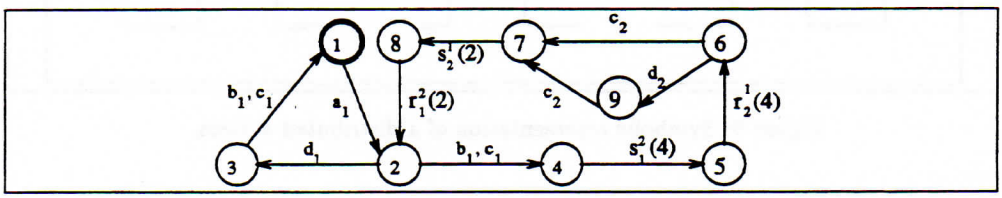

Figure 7: Example of a global protocol specification.

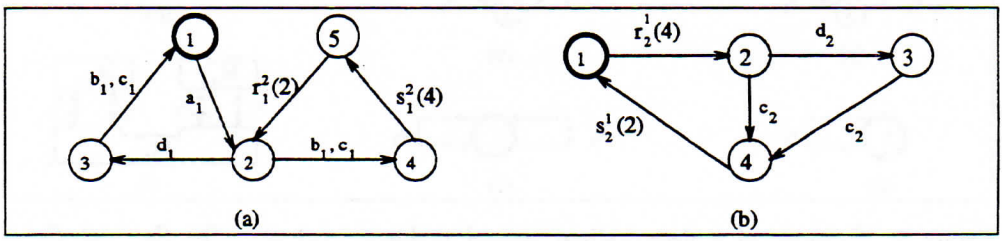

Figure 8: Synthesized protocol specifications. (a) PS_1. (b) PS_2.

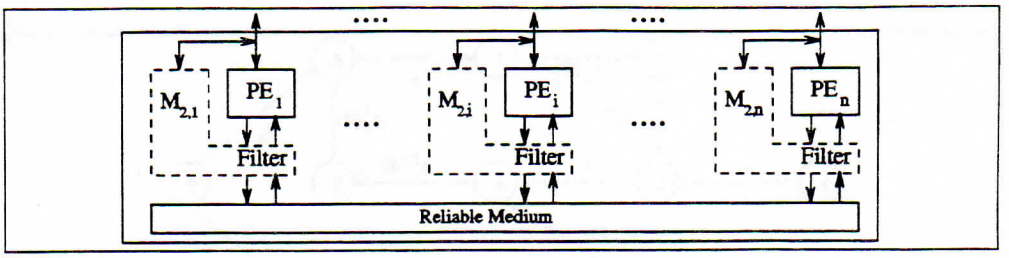

Figure 9: Local controllers.

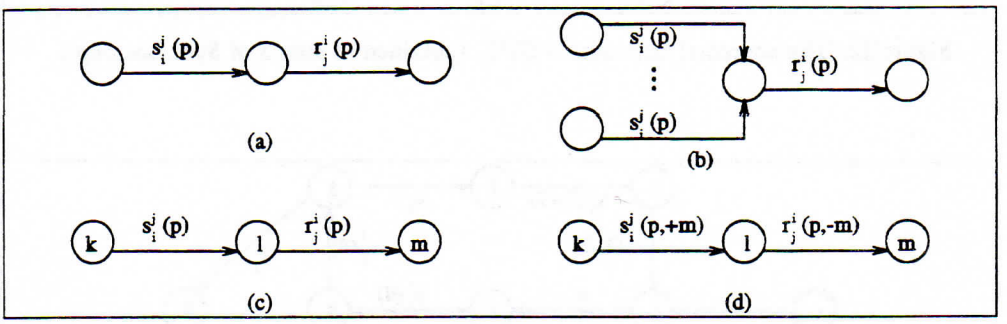

Figure 10: Rules of filtring.

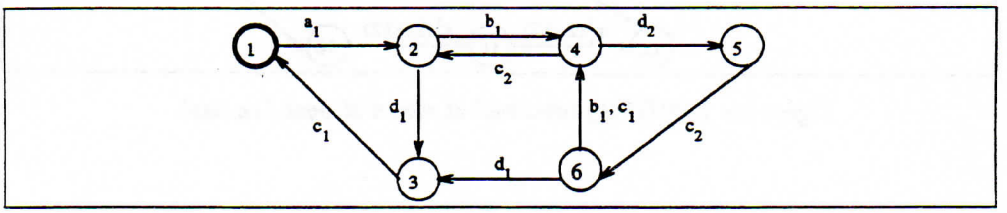

Figure 11: Example of a desired service specification SS_0.

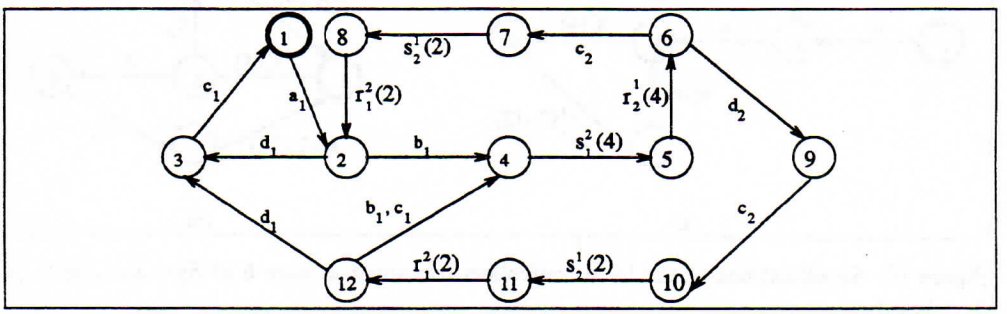

Figure 12: $GPS_{1,0}$ (obtained at step 2 of *Synt_Loc_Sup*).

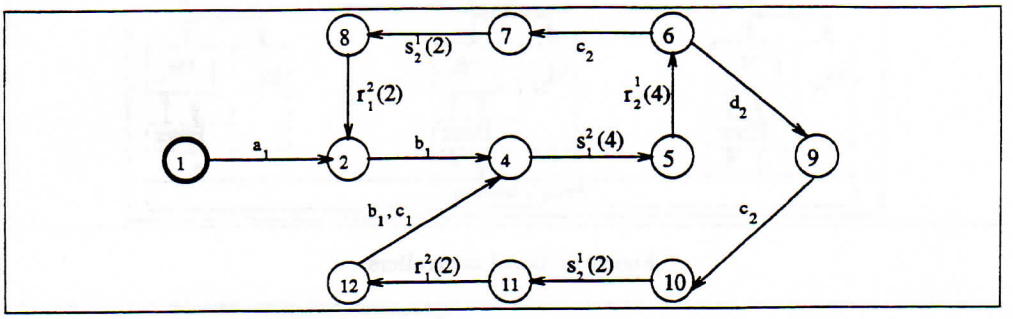

Figure 13: The supremal controllable GPS_c (obtained at step 3 of *Synt_Loc_Sup*).

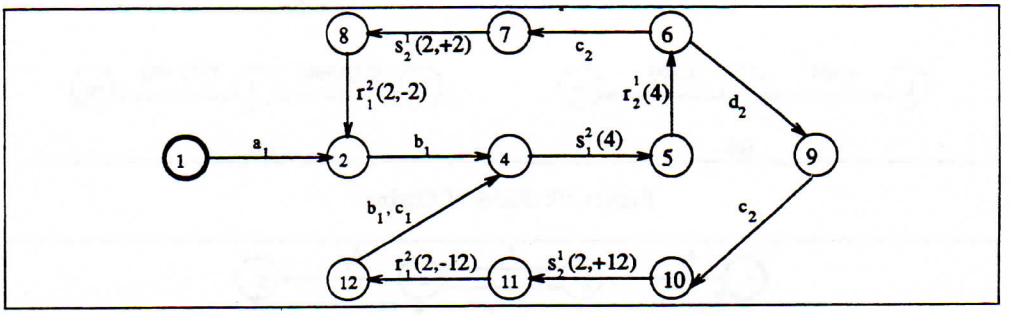

Figure 14: $Filt(GPS_c)$ (obtained at step 4 of *Synt_Loc_Sup*).

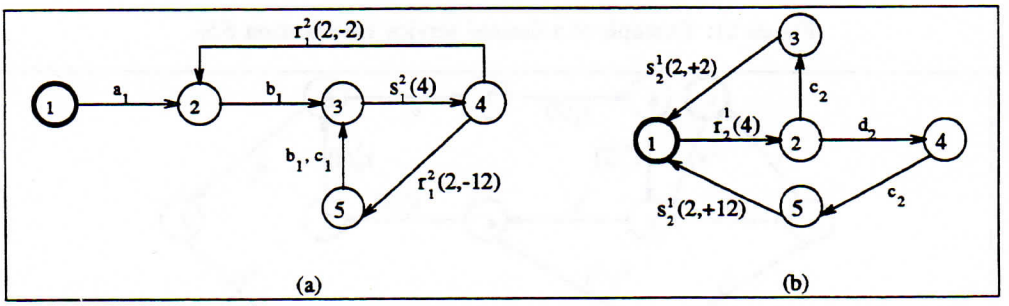

Figure 15: Specifications of the local controllers (obtained at step 5 of *Synt_Loc_Sup*). (a) $S_{2,1}$. (b) $S_{2,2}$.