A Protocol for WorkGroup in Industrial Environments Based on MMS.

Raúl Jacinto-M., Saúl Pomares Hdez., Felix F. Ramos C., Roberto Gómez C..

CINVESTAV-Gdl Prol. López Mateos Sur 590
Guadalajara Jal. 45090
México
rjacinto, saul, framos @gdl.cinvestav.mx

ABSTRACT. *MMS is a widely used protocol which provides the basic services to share information between the components of a given industrial application. The MMS services are the product of a long evolution which has not yet stopped. Considering the cooperation explicitly in the design of distributed real time systems covering the SCADA systems spectrum is a research area to explore. The purpose of this paper is to incorporate some techniques of group communication into industrial environments which lead to a natural extension of the MMS services actually considered.*

KEY WORDS: *MMS, Real Time Distributed Systems, Protocols, Group Communication, AMOEBA, Cooperation, WorkGroup.*

1 Introduction

MMS [ISO9506]is a widely used protocol which provides the basic services to share information between the components of a given industrial application. The offered services by MMS are independents of the application or the developer, giving the basis for a truly open environment.

The basic abstraction in the design of MMS is the VMD (Virtual Manufacturing Driver) this abstraction captures a wide range of commonalties founded in the industrial devices as robots, PLC's, industrial PC's. This abstraction is in the basis for the generality in the offered services.

MMS is rigorous enough to minimize the differences between applications performing similar or complimentary functions while still being generic enough for many different kinds of applications and devices[MAKIE94].

The MMS services are the product of a long evolution started since 1986 with the first version of this standard (in fact the MMFS standard)[MAKIE94][ISO9506]. This evolution is far from being stopped, it is necessary to keep the door open for the new technologies emerging from computer sciences and from the electronic equipment.

The actual trend in computer systems is the design of collaborative applications (CSCW). Considering the cooperation explicitly in the design of distributed real time systems covering the SCADA systems spectrum is a research area to explore. The purpose of this job is to incorporate some techniques of group communication into industrial environments. This lead to a natural extension of the MMS services actually considered. The reasons to base this job on MMS Standard are quite obvious: the great body of knowledge constructed around MMS can not be dismissed and is a valuable guide to assure an ordered and useful evolution of MMS based systems.

The proposed extensions mainly consider the needed primitives to group management. Simultaneously the actual primitives are studied to determine the kind of communication best suited for a group work.

In order to be able to support the proposed extensions, a two sub-layer architecture is proposed. The two sublayers fit onto the application layer of the OSI model. The two sublayers architecture allows to cleanly separate responsibilities, the lower sublayer works to assure the kind of communications needed, (peer-to-peer, causal and FIFO broadcast capability and most important fault recovery). The higher sublayer provides the extension to the MMS services and the MMS services themselves, masking the choice of the kind of required communication.

This job is structured in the following manner: in the next section presents the framework, the third section discuss about the related jobs, the fourth section presents the model and the architecture where the protocol is intended to run the analysis of the service offered and the communication needed to support it and finally some conclusions and future work are presented. . The annex includes the main primitives of the proposed extensions.

2 FRAMEWORK

2.1 Real Time Considerations

The goal of this job is to furnish the needed support for the cooperative work in industrial environments. With this purpose real time restrictions and a high degree of reliability have to be considered. To met these restrictions, flexible and reliable mechanisms are needed.

It is important to keep in mind that the goal is to provide a support for applications, not restraint the application design to a specific style of work. This support needs to be not so specific that constraint the design of the intended applications but it has to be simple and flexible enough to do not compromise the real time behavior.

Real Time Systems RTS A real time system is one with a time dependant behavior [MULL95]. Stanckovic[STAN92] states " a real time system is one with a predictable behavior".

Both definitions stresses the fact that the proper function of a real time system is not only dependant of its logical computational results but also from the time those are produced.

Real Time Communication a set of rules which determine the worst case of communication between two process using a data transmission network.

To get a real time communication two things are needed: a real time network and a real time protocol. A real time network is one which exhibits a limited and known propagation delay and a reliable behavior (no data losses caused by the network). A real time protocol is one which guarantees that there is a deterministic and fair access for all the sites connected to the network.

A real time communication exhibits some of the following functional attributes [MULL95]:

- Bounded and known propagation delay.
- Deterministic behavior in the presence of distortion factors(overload, failures) .
- Connectivity.

For this job only LAN´s are considered, and particularly those with a real time communication behaviour, by instance the fieldbus familiy of networks (FIP, PROFIBUS, FieldBus) or the IEEE802.4 (token bus) . These networks are based on the model of the figure 1 which considers just the layers needed for communication on a LAN[PIM89].

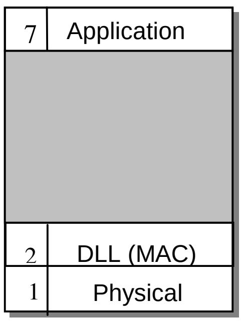

Fig. 2.1 OSI Collapsed Model

It is assumed that the MAC[1] function of the DLL [2] is designed to assure a real time communication behavior defined in terms of broadcast communications:

- Broadcast: All sites in a network receiving a message receives the
same message.
- Error detection: The sites of a network are capable of detecting an error in a message caused by the network.
- Network Order. In a single segment of a LAN two messages received by two different sites are received in the same order.
- Full Duplex: The transmitter can be designated also as recipient of a message.
- Tightness: a correct message received by two differents sites are received into a known interval
- Bounded Transmission Delay: Every message stored in a site is transmitted within a limited delay $T_{td}+T_{ina}$ (T_{td} is the maximum transmission bounded delay, T_{ina} is the maximum inaccessibility time).
- Bounded Omission Degree. In a bounded known interval T_{rd}, the number of transmission omission can not exceed of *k* transmissions, where *k* is fixed and known.
- Bounded Inaccessibility : In a known interval T_{rd}, the network only can become inaccessible at most *i* times with a maximal total duration T_{ina} .

2.2 GROUPS

As long as the purpose of this job is to provide a support for cooperative work in industrial environments some related definitions are needed. These definitions cover the main aspects of work group[FOX81]:

Group: A group is defined as a set of agents working together to achieve a common goal. In a distributed system this definition implies the need for a communication network. In this job, an agent for site is considered and both terms (agent and site) are used interchangeability.

Group organization: A group needs an organization, defined as the assignment of roles to agents members of the group. This assignment goes with the responsibilities and the authority inherent to each role.

The most popular kind of organizations are:

- Hierarchical: with a single site acting as a global responsible of the work group. This site commands all the actions of the group and is responsible of the synchronization of the global application. This approach is the one commonly

[1] Medium Access Control

[2] Data Link Layer

used in actual applications, is very simple to understand and implement but it has the drawback of the dependence of the coordinator site.

- Peer to peer: in this kind of organization all the agents have the same degree of authority (equipolency) and some voting mechanisms are needed to achieve consensus between all the agents when a decision has to be taken.

In between these two extremes a lot of organizations can be found.

Group Structure: A group also need an structure, defined as the setting of the relationships between the agents members of the group.

Membership A group has a membership, which defines if a site belongs to a group.

Dynamic membership A membership is said dynamic if the set of members of the group is allowed to change in time.

Closed groups: A group is said to be closed if the only communication allowed is inside the group. An external agent to the group can not address the members of the group.

3 RELATED JOBS

In a recent paper Kopetz[KOP98], prones for a design of distributed control applications based on components, where the components can be assimilated to groups of VMD's separated by TimeWalls. A TimeWall is a synchronization mechanism by means of which, a set of agents arrive to coordinate its job. Using this concept, an application can be constructed using components with a well defined interface.

A group can be seen as a component just like in the Kopetz approach, however the difference between the component and the groups is the mechanism to achieve the separation: for a component the mechanism is the TimeWall, for a group the mechanism is the group address which mask the inner structure of the group. Discussing which mechanism is better is out of the scope of this paper, what it is important is to stress the need of a basic element (a group or a component) to structure complex applications.

In [FuHMAN94] a fault tolerant application is designed using ISIS, however the emphasis is made on the improvement in fault tolerance for a particular application more than in the integration of MMS in a group environment.

Designing Distributed Systems is a difficult task which is largely a matter of achieving a certain degree of synchrony in the global application. The synchronization problem has been widely addressed and solutions as the Logical Clocks [LAM78] or the Causal Vectors [HT93] provide the basis to solve it. A closely related issue is the fault tolerance which can be achieved of using these mechanisms. These mechanism are communication style related and particularly with group communication.

3.1 Broadcast Communication Definitions

The following definitions and the kind and characteristics of communications needed to support group communications are taken from[HT93]. Causality is a key factor to organize an application. If it can be assured that every participant sees the global application as preserving the causality relation (an event *x* causes an event *y*) then the application is correct [HT93]. Causality is defined as follows:

An event *e* causally preceds an event *f* ($e \rightarrow f$) iff:
1. a process executes e, f in that order.
2. *e* is the transmission event of the *m* message and *f* is the reception event *f*
3. it exists *h* such that ($e \rightarrow h$) and ($h \rightarrow f$).

Causal broadcast: a causal broadcast is a reliable transmission such that it respect the Causal Order. If the transmission of the *m* message precedes causally the transmission of a message *m'*, then no correct process will deliver *m'* unless it has previously received *m*.

Atomic Broadcast: an atomic broadcast is a reliable transmission such that if any two pair of correct process receive the *m* and *m'* messages the p delivers m before m' iff q delivers m before m'.

FIFO Atomic Broadcast: a reliable transmission satisfying both the FIFO order and the total order in messages delivery.

Atomic Causal Broadcast a reliable broadcast satisfying the causal order and the total order in the delivery of the messages.

Group communication is based on these kind of communications, which are the basis to design reliable and efficient distributed systems.

3.2 ISIS and AMOEBA

ISIS and AMOEBA are two application examples of the above theory, and its worth to let their main characteristics established.

3.2.1 ISIS

The ISIS Toolkit [BIR93] offers services for group management (closed and dynamics) and Atomic Broadcast and Causal Broadcast, its Virtual Synchrony concept is a natural mechanism to achieve a great degree of fault tolerance. The ISIS Toolkit does not assume any particular communication network and this fact gives it a great flexibility to cover a vast field of applications. This characteristic implies that it is not optimized to work in a real time environment. However ISIS is a good example of the kind of communication needed for work group, and its theoretical basis are a cornerstone for the proposed protocol.

3.2.2 The AMOEBA Architecture

AMOEBA is Distributed Operating System [KT92]designed for operation in a LAN environment, its main goal is to provide the user with a clean and transparent interface to reliably access the network available resources.

AMOEBA offers also group management and fault tolerance based on a LAN. Its architecture is the one depicted in figure 3.1. AMOEBA assumes that the communications are acknowledged, and the synchrony between all sites is achieved by relying in a designated site called the SEQUENCER, any site can be the sequencer but there is only one active at a time. AMOEBA offers the Atomic Broadcast service, and because the broadcast capabilities of the subjacent network a FIFO Atomic Broadcast is achieved.

The basic mechanism of AMOEBA is as follows: whenever a site desires to send a message, first it sends the message *M* using its local kernel and blocking. The kernel sends *M* to the sequencer, which assigns a sequence number *s* to the message, broadcasts the message *M+s* to all sites, the blocked originator site can now proceed. The sequencer is able to maintain the global history of the global application, the history is perceived as a total order for all participants.

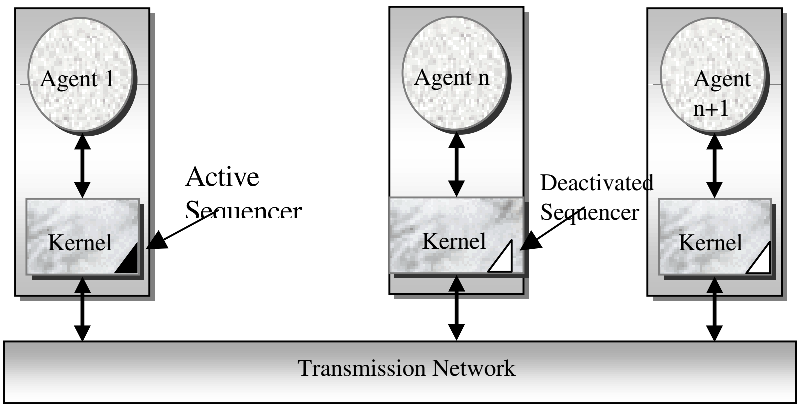

Fig. 3.1 AMOEBA System Architecture

AMOEBA uses the total order of the history to offer a fault tolerant service It is possible to design a new sequencer in case of failure of the initial one. A failed site at recovery time will ask the sequencer for the application history, based on this history the site will be able to synchronize its operation.

The AMOEBA atomic broadcast service also relies on the sequencer operation: when it is required that a message will be received for all its intended receivers, the message is sent to the sequencer which in turn (and after assigning a sequence number) will broadcast the message waiting for acknowledge messages from the receivers. Once the acknowledges received, the sequencer will broadcast the deliver order which allow to all sites to deliver the message to their respective process [3].

4 THE PROPOSED PROTOCOL

In order to be able to support the proposed extensions, a two sub-layer architecture is proposed. The two sublayers fit onto the application layer of the OSI model. The design of the lower layer is based on AMOEBA considering some modifications to consider a real time behavior. The upper layer adapts the MMS protocol to use the service provided by the lower layer. A real time MAC is assumed from the DLL. What is proposed here are the needed tools (the primitives) to structure complexes applications.

4.1 The Model

The protocol model is based on groups which works on a LAN's segment with broadcast capabilities. Particularly a bus topology is considered. It is supposed that there exists a sequencer for group which is defined at design time. It is possible to interconnect groups, but they will interact by means of the sequencers of each group. The sequencers in fact will form another group.

The model considers a high reliability network, the failures are considered to happen mostly on sites which only can fail on fail-stop mode without disturbing the operation of the other sites. It is also assumed that every site has enough memory to hold appropriately the incoming messages when needed. The likelihood of loosing a message is considered to be low enough to do not need to be a major concern.

---

[3] It is easy to see that the above behavior can be modified to accommodate a *k*-resilient degree, just for waiting *k* acknowledges in the sequencer.

Addressing and group structure: the groups are addressed by a unique direction, a process can address the group without worrying about its inner structure. It is assumed that the groups are closed and their membership can be dynamic. It is a sequencer responsibility to keep the group membership updated. If an outside communication is received, this is catched by the sequencer and broadcasted if needed into the group.

Reliability: the proposed protocol considers the possibility of communication failures. Lost of a message because of buffer overflow or hardware error it is possible to get a copy of the lost message.
The sequencer based architecture allows also the detection of a failed site. If a site remains silent for a long period the sequencer will ask directly the suspected site, in case of no response the site will declared death and kept out of the group.

Ordering: Two kinds of order are considered: inside a group (and in consequence in a segment of a LAN) a fifo atomic broadcast is guaranteed. Between groups a Causal order is considered, this order is achieved by means of Causal Vectors.

Delivery semantics: an "exactly once" semantics is considered, all the messages have a sequence number to avoid duplicities, the sequence number is constituted by a sequence number and the site ID.

Answer Semantics: the protocol is based on a negative acknowledgement scheme. A process only sends a negative acknowledge when it has lost a message, in a group this can be detected thanks to the sequence number and the network capabilities of broadcast.

A negative acknowledge is appropriated because the great reliability of the network, acknowledging every message by each site will need a number of messages equal to the number of sites wasting time and bandwidth.

4.1.1 Associations vs Connections

A connection is an attribute of the underlying network layers that represents a virtual circuit between two nodes. An application association is an agreement between two networked applications governing their communications. In analogy with a telephonic conversation, a connection is just the telephonic circuit used to exchange information, whereas the association is the conventions used between the speakers to exchange information (by instance not speak about football) over the telephone[PIM89]. Application associations exist independent of any underlying network connections.

The natural way of structuring applications is by means of associations, where the association can exist not just between pairs of process but in groups of process.

Where a certain degree of privacy or synchrony between a pair of process is needed a connection is mandatory.

4.2 THE ARCHITECTURE

The protocol is intended to fit in the application level of the OSI model[PIM89]. The architecture of each site considers a two sublayer organization. The lower sublayer is oriented to provide the communication support needed for the MMS primitives. This sublayer offers the services of causal broadcast, atomic broadcast, normal broadcast and point to point communication. The upper layer provides the MMS services mapping them to the appropriated communication support (Fig. 4.1).

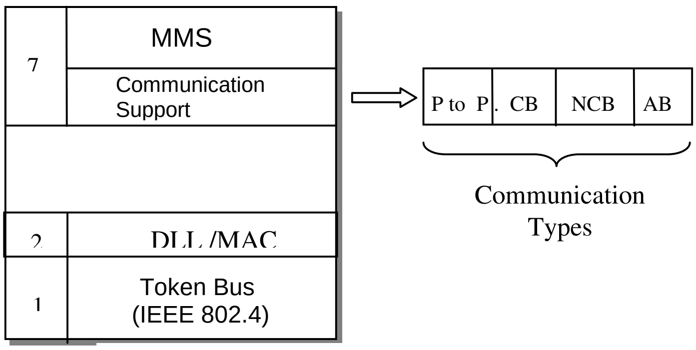

Figura 4.1 The proposed architecture

In order to provide the expected support, the organization of the lower sublayer is based on AMOEBA. There exists a site which is responsible of the following functions:

- Work orders transmission.
- Synchronization signals.
- Intergroup communications.
- Membership update.
- Retransmission of lost messages.
- Registration of not delivered message

As in AMOEBA the sequencer can be any of the sites, however unlike AMOEBA the designation of a new sequencer in case of failure of the original is solved using a predetermined assignation, wasting the minimum of time to determine the next sequencer.

The communication types supported for the lower sublayer are point to point, causal broadcast, atomic broadcast and non causal broadcast. The causal broadcast is mainly used to achieve intergroup communication.

4.3 THE SERVICE

The main element of MMS is the Virtual Manufacturing Device (VMD) model, based on the client/server concept. The VMD specifies how MMS devices or servers, behave as viewed from an external MMS client application point of view. MMS allows any application or device to provide both client and server functions simultaneously[ISO9506]. In general the VMD model defines the objects (e.g. variables) contained in the server, the services needed for a client to manipulate the objects and the behavior of the server upon receipt of these service requests from clients.

The main difference between the traditional view in MMS and the one proposed here is the fact that the VMD model is the client/server and the model assumed here is one based on collaborative work. In a collaboration the participates cooperate according their capabilities, so it can be argued that a collaborative model subsumes the client-server relationship.

The important point is the one of structuring applications, the hypothesis of this job is that work group is needed to cope with complexes tasks. The advent of techniques from the Distributed Artificial Intelligence and Distributed Control and CSCW fields will need a transition tool to cover the traditional applications and the new ones.
The MMS services cover the following functional units[ISO9506][MAKIE94]:

- Variable Access: services oriented to allow the access of remote variables either consulting or updating.
- Resources Sharing: mutual exclusion services to assure consistency in resource access, declaration of global resources. This services are based on binary semaphores, the services do not include deadlock detection or prevention.
- Events Management: services to notification and acknowledge of important events of the system.
- Programs Management: services to remote control of program execution, by instance load programs, initiate execution, and so on.
- Message passing: services mainly oriented to provide files exchange or information related to the context of an particular application.
- Other services: services such as connections management, operator communications, etc..

This job proposal is to add a new functional unit, one related to group management. This functional unit will cover all relative to group management. Typical services of this unit are: group creation, membership updating (joining the group and leaving the group), group deletion, detection of failed sites.

To design this functional unit, it has been considered that a previous work to design the application exists, i.e. that a participant in a group knows previously the existence of its group. Also it is assumed that at least there is one site which has to start the work of the group, it is called the sequencer.

These hypothesis come from the kind of applications which uses MMS, these are industrial applications in closed worlds where the environment can be controlled in some degree, by instance a cloth factory will do all kinds of clothes, but no will factory bread.

Because of the extension of the work, in this paper only the group management primitives proposed are presented and discussed in some extent. Other functional units are reported and work is still in progress to cover all the MMS primitives in a work group context.

4.3.1 Group Management Functional Unit.

The group management primitives functional unit is composed of the primitives listed in the table 1 of the annex. The table shows the proposed name for the primitive, the intended semantics the kind of communication support required and the arguments for the primitive.

It is clear that covering in depth all the aspects of the extended MMS protocol is a great job, which has to be clearly detailed to be useful. The experience gained in protocol specification for the OSI model shows that a formal specification is needed. An ESTELLE[ISO] specification is in progress and now it covers completely the lower sublayer services and the group management primitives. Also those related with the functional units : variable access, resource sharing and event management.

4.3.2 Variable Access Functional Unit

This functional units covers the related functions to variable management, but also considers the need for publishing results. Publishing results is a powerful technique which can be used with coordination and synchronization purposes with no special complexity. Publishing can be used to inform the group about the actual state of computation.

Traditional services are kept, allowing a more synchronous operation between pairs of process. The primitives composing this functional unit are listed on the table 2 of the annex.

4.3.3 Resource Sharing Functional Unit

Resource sharing is a key issue in distributed applications. Guaranteeing mutual exclusion in resource access is a main concern where consistency in resource management is mandatory. Mechanisms for mutual exclusion can be implemented easily using group communication. Defining the best mechanism remains open in this work, in fact it is assumed that the resource owner is also responsible to handle the associated semaphore, but the responsibility can now be distributed. The primitives intended to constitute this functional unit are listed on the table 3 of the annex.

4.3.4 Event Management Functional Unit

Event management is likely to be a great concern in designing a distributed control system. An event is the manifestation of an operation which need to receive special attention, and is a key feature in industrial applications. The events need to be detected and signaled to the whole group in order to receive the appropriate treatment. The detection, signaling and treatment need to be in real time, where the communication support needs to be as efficient as possible.

MMS associates to each event **conditions** and **actions**. A **condition event** is defined by a set of rules and value ranges for variables which determine if an event has occurred. **Action events** are the defined planning to recover the system to a safe state, either previous, equivalent, or just safe. The primitives intended to constitute this functional unit are listed on the table 3 of the annex.

5 CONCLUSIONS AND FUTURE WORK.

This job is based on the hypothesis that work group is the natural evolution of industrial MMS based systems. The group abstraction can lead to an easier structuring of any given application. A group is identified by an address which masks the inner structure of the group.

Cooperation needs some communication tools to assure the coordination needed. The extensions to MMS proposed here are oriented first to provide the needed support for these communication tools, and second to redefine the MMS primitives to adapt to workgroup.

5.1 Actual State and Future Work

This job is far from finish, actually an Estelle specification[DSB88] is ongoing. This specification covers the communication support and the group management, variable access, resources management and the event management. A job to complete and fully verify this specification is in cours. However even if there is a great lot of job in work group applications, a validation job for the industrial environments is needed and a project designing a prototype SCADA system based on MMS-WorkGroup to control a power generation plant is now started.

6 Bibliography

[BIR93] Kenneth P. Birman. THE ISIS PROCESS GROUP APROACH TO RELIABLE DISTRIBUTED COMPUTING. Communications of the ACM, December 1993.

[DSB88] P. Dembiski, S. Budkowski, SPEFICICATION LANGUAGE ESTELLE, North Holland 1988

[FOX81] Mark S. Fox AN ORGANIZATIONAL VIEW OF DISTRIBUTED SYSTEMS IEEE Transactions on Systems, Man and Cybernetics, SMC 11:70-80, 1981.

[FuHM94] C. Fuhrman, S. Messina, "Fault Tolerance Techniques Integrating MMS and ISIS", Proc. of ETFA'94, Tokyo, Japan, Nov. 6-10, 1994.

[HT93] V. Hadzilacos, and S. Toueg. FAULT-TOLERANT BROADCAST AND RELATED PROBLEMS, in Distributed Systems. 2nd. ed. S. Mullender, Addison-Wesley, Readings, MA, 1995.

[ISO 9506] ISO/IEC 9506-(1,2,3,4,5,6) Industrial Automation Systems - Manufacturing Message Specification - International Organization for Standarization.

[KT92] M. Frans Kaashoek, and Andrew S. Tanenbaum. EFFICIENT RELIABLE GROUP COMMUNICATION FOR DISTRIBUTED SYSTEMS. Vrije Universiteit,1992.

[KOP98] Herman Kopetz COMPONENT BASED DESIGN OF LARGE DISTRIBUTED REAL TIME SYSTEMS Control Engineering Practice IFAC, Vol. 6 No. 1. 1998.

[LAM78] Leslie Lamport. TIME, CLOCKS, AND THE ORDERING OF EVENTS IN A DISTRIBUTED SYSTEM. Communications of the ACM, July 1978. COORDINATING MULTIPLE DISTRIBUTED TASKS. Computer, May 1994.

[MAKIE94] Ralph Mackiewicz, AN OVERVIEW TO THE MANUFACTURING MESSAGE SPECIFICATION, Copyright 1994. In http://litpc13.epfl.ch/MMS/mms_IntroSISCO.txt

[MULL95] Sape Mullender. DISTRIBUTED SYSTEMS 2nd. ed. S. Mullender, Addison-Wesley, Readings, MA, 1995.

[PIM89] Juan R. Pimentel. COMMUNICATIONS NETWORKS FOR MANUFACTURING SYSTEMS, Prentice Hall, 1989.

[STAN92] John A. Stankovic. Misconceptions about Real Time Computation, Computer Magazine, Vol 3 No. 6 July 1988.

7 Annex A: Service for MMS extensions.

Notation

In the following the next notation will be used:

TCP : reliable and ordered peer to peer communication
UDP.- unreliable and unordered peer to peer communication
CB.- Causal broadcast
NCB.- Non Causal Broadcast or just broadcast
AB.- Atomic broadcast
Sense of the communication

↘	Sequencer to Participant	Point to Point messages
↗	Participant to Sequencer	
⇗	Participant to Sequencer	
⇘	Sequencer to Participant	
→	Sequencer to Group	Broadcast messages
⇒	Participant to Group	

Participant is either a site wishing to belong to a group or a group member. Sequencer is the controller of the group communication as in AMOEBA.

PRIMITIVE	Semantics	Direction	Required Communication
Create_Group (gd, id_process)	Creation of the **gd** group by the process **id_process**	→	AB
Join_Group (gd, id_process)	Request for the process **id_process** to enter to the **gd** group.	↗	TCP
Leave_Group (gd, id_process)	Request of the **id_process** process to leave the **gd** group .	↗	TCP
Join_Refused (gd, id_process, why)	Join Group **gd** Request denied to the **id_process** process.	↘	TCP
Join_Accepted (gd, id_process)	Join Group **gd** Request accepted to the **id_process** process. .	→	NCB
Leave_Refused (gd, id_process, why)	Leave Request the **gd** group denied to the **id_process** process..	↘	TCP
Leave_Accepted (gd, id_process)	Leave Request the **gd** group Accepted to the **id_process** process..	→	NCB
Erase_Process (gd, id_process)	Notification that the process **id_process** has been deleted from the **gd** group.	→	NCB
Erase_group(gd)	Notification that the **gd** group has been ceased to exist.	→	NCB
Alive_Request (gd, id_process)	Request for an Alive Answer message to determine if the **id_process** of the **gd** is still alive.	↘	UDP
Alive_Answer (gd,id_process)	Alive answer message to the alive request for the **id_process** process of the group **gd.**	↗	UDP
New_Sequencer (gd, id_process)	New **id_process** sequencer notification in case of failure of the primary sequencer of the **gd** group	→	AB

Table 1. Group Management Primitives

Primitive	**Semantics**	**DIRECTION**	**Required Communication**
Create_Resource (gd, id_process, Resource, attributes type)	Declares a resource **Resource**of type **type** with the attributes **attributes** inside the **gd** group, managed by the **id_process**.	⇒	NCB
Delete_Resource (gd, id_process, Resource)	The process **id_process** erase the resource **Resource** from the group **gd**. It is assumed that the process owns the resource.	⇒	NCB
Modify_Resource (gd, id_process, Resource)	The process **id_process** modifies the attributes **attributes** of the resource **Resource** from the group **gd**.	⇒	NCB
Take_Resource (gd, id_process, Resource, priority)	Request for the process **id_process** with priority **priority** to gain access to the resource **Resource** in the group **gd**.	⇒	CB
Take_Refused (gd, id_process, Resource, why)	Refusal to the request issued by the process **id_process** to access the **Resource** of the group **gd** by the reason listed in **why**.	⇘	TCP
Take_Accepted (gd, id_process, Resource)	Affirmative anser to the request issued by the process **id_process** to access the **Resource** of the group **gd**	⇒	NCB
Take_Suspended (id_process, Resource gd,)	Notification to the process **id_process** of the temporal suspension of its access to the resource **Resource** of the **gd** group.	⇘	TCP
Liberate_Resource (gd, id_process, Resource)	Release the resource **Resource** of the group **gd** from the process **id_process**	⇒	NCB
Take_Continue (gd, id_process, Resource)	Notification to the process **id_process** of the renewal of its access to the resource **Resource** of the **gd.**	⇘	TCP
Take_Canceled (gd, id_process, Resource)	Notification to the process **id_process**, of the definitive suspension of its rights to access the resource **Resource.**	⇒	NCB

Table2 Resource Management Functional Unit

PRIMITIVE	DESCRIPTION	DIRECTION	Required Communication
Define_Var (attrib, type, var_name, gd, id_process,list)	Definition by the process **id_process** of the variable **var_name** in the list **list**, with the attributes **attrib** y and type **type** . The variable is defined just inside the group **gd.** A variable belongs to the process which defines it.	⇨	NCB
Delete_Var (var_name, gd, id_process,list)	Erase the variable **var_name,** in the list **list** del grupo **gd**.	⇨	NCB
Read_Var (gd, id_process, var_name)	Request by the process **id_process** for the value of the variable **var_name** of the **gd** group.	⇗	TCP
Send_Read (gd, var_name, val_var)	Sends the value **al_var** of the variable **name_var** defined inside the group **gd.** This is the answer to the request for value issued with the *Read_Var* primitive.	⇨	CB
Write_Var (gd, id_process, var_name, val_var)	Request for writing the value **val_var** for the **var_name** variable belonging to the group **gd** .	⇗	TCP
Write_Refused (gd, id_process, var_name, why)	Refusing of the request of the process **id_process** to write over the variable **var_name** belonging to the group **gd,** the reason of the refusal is given in **why .**	⇘	TCP
Read_Refused(gd, id_process, var_name, why)	Refusing of the request of the process **id_process** to read the **var_name** variable belonging to the group **gd,** the reason of the refusal is given in **why .**	⇘	TCP
Send_Var (gd, id_process, var_name, val_var)	Sends the value **val_var** of the variable **name_var** belonging to the process **id_process** inside the group **gd**. This primitive does not need any other to be executed.	⇨	CB
Send_Var_List (gd, id_process, val_list, list)	Sends the value list **val_list** of all the variables of the list **list,** belonging to the process **id_process** inside the group **gd**. This primitive is the answer to the *Read_Var_List* primitive.	⇨	NCB
Read_Var_List (gd,id_process, list)	Request by the process **id_process** of the values of the variables defined on the list **list** of the **gd** group**.**	⇨	CB
Delete_List (gd, list)	Request by the process **id_process** to erase the variable list **list** of the group **gd.**	⇨	CB

Table 3 Resource Management Functional Unit

.

PRIMITIVE	SEMANTICS	DIRECTION	Required Communication
Define_Event (gd, id_process, id_event, type, status, description)	Define the event **id_event** with its type **type**, description **description** and status **status.** The event is declared to be handled by the group **gd.**	⇨	NCB
Modify_Event (gd, id_process, id_event, type, status, description)	Request of the process **id_process** to modify the type **type** and description **description** to the event **id_event** of the group **gd** . This modification is in fact a notification of the owner of changes in the attributes of event	⇨	NCB
Delete_Event (gd, id_process, id_event)	Request of the process **id_process** to erase the event **id_event** of the group **gd**	⇨	NCB
Event_Notification (gd, id_process, id_event)	Notification to the group **gd** of the occurency of the event **id_event**.	⇨	CB
Trigger_Event (gd, id_process, id_event)	Request for activation of the event **id_event** by the process **id_process** of the **gd** group.	⇨	CB
Event_Status (gd, id_process, id_event)	Request to the process **id_process** for the status of the event **id_event** of the group **gd** .	⇗	TCP
Repport_Status (gd, id_event, status)	Report the status of the event **id_event** del grupo **gd**.	⇘	CB

Table 4 Event Management Functional Unit