AMP: A HIGHLY PARALLEL ATOMIC MULTICASTING DATA LINK PROTOCOL*

Paulo Veríssimo, Luís Rodrigues, Mário Baptista

Abstract

The present paper deals with the problem of reliable group communication for dependable applications, in the context of the Reliable Broadcast class of protocols. An atomic multicast protocol for token passing Lans is presented. The actual implementation is on an 8802/4 Token-bus, although it is applicable to 8802/5 Token-rings and the FDDI Fibre-Optic network.

Traditional approaches to Byzantine Agreement protocols make no assumptions about the faulty behaviour of components, being however very costly in time and traffic. The simplicity and efficiency of reliable broadcast protocols may be considerably improved, if the system fault model is restricted or convenient architectures are used.

The principles of using fail-controlled communication components to build efficient reliable broadcasting Lan Data Link protocols, discussed in another paper, are used here to build a reliable multicast protocol on top of the exposed MAC interface of a VLSI Lan controller. The resulting MAC provides the user with multicasting primitives, in addition to the standard ones.

The architecture is built on standard Lans, in view of taking advantage of the availability of communications hardware and of the possibility of coexistence with standard stations, in the same network. The service offered allows restricted group communication (multicast), which is essential to give efficient support for high performance distributed computing systems.

*A version of this report was published in the Proceedings of the SIGCOM'89 Symposium, Austin (TX), USA, 1989. This work was been supported in part by the EEC, through ESPRIT Project 1226.

1 Introduction

The present paper deals with the problem of reliable group communication for dependable applications, in the context of the Reliable Broadcast class of protocols. An atomic multicast protocol for token passing Lans is presented. The actual implementation is on an 8802/4 Token-bus, although it is applicable to 8802/5 Token-rings [MCS88]; an FDDI implementation is also in progress.

Traditional approaches to Byzantine Agreement protocols [Lamport82] make no assumptions about the faulty behaviour of components. Taking the whole universe of faults into account avoids the problem of fault coverage, i.e. the probability of the system failing differently from an assumed failure mode, which could lead to a catastrophic failure.

However, since these protocols are very costly in time and traffic, it seems reasonable to look for cheaper solutions, acceptable in all but very high safety applications.

It has been shown that the simplicity and efficiency of reliable broadcast protocols may be considerably improved, if the system fault model is restricted [Hadzilacos84, Dolev83, Schneider84b, Chang84] or if convenient architectures are used [Babaoglu85a].

The principles of using *fail-controlled* [Laprie86] communication components to build efficient reliable broadcasting Lan Data Link protocols, are discussed in detail in [Verissimo87g]. The communication servers exhibit what we call the *Fail-Silent* property: a kind of halt-on-failure behaviour, enforced through self-checking hardware.

The architecture is built on standard Lans, in view of taking advantage of the availability of communications hardware and of the possibility of coexistence with standard stations, in the same network. The service offered allows restricted group communication (multicast), which is essential to give efficient support for high performance distributed computing systems.

One of the implementation alternatives [Verissimo87d] consists of embedding the reliable multicast functionality in the Lan MAC (Medium Access Control) sublayer, by intervention in its state machines. While requiring modified MAC VLSI, it is by far the most efficient, through use of *token holding control* and *low level synchronization*. The machines can be implemented so that an enhanced MAC is obtained, retaining its conformance to the original MAC standard, plus the additional reliable multicast capability.

This paper addresses an alternative implementation, which avoids the significant investment of the first approach, while still displaying an interesting performance. It consists of building the reliable multicast protocol on top of the exposed MAC interface of a VLSI Lan controller. The particular implementation is for a Token-bus Lan.

The resulting MAC provides the user with multicasting primitives, in addition to the standard ones.

2 Communication System

The communication system is based on the principle of communication servers (Fig. 1) providing a reliable multicast communication service, both decoupling communications (NAC) from processing (Host) and presenting an error containment domain, due to the Fail-Silent property of the NACs (Network Attachment Controllers), which restricts their faulty behaviour to stopping and omission faults.

The Channel must have some provision to maintain *network availability*. In the token-bus, a dual-medium Channel is used. Hardware implemented management procedures and fault treatment operations ensure continued service provision and a controlled degree of Channel omissions (lost frames) [Verissimo88b].

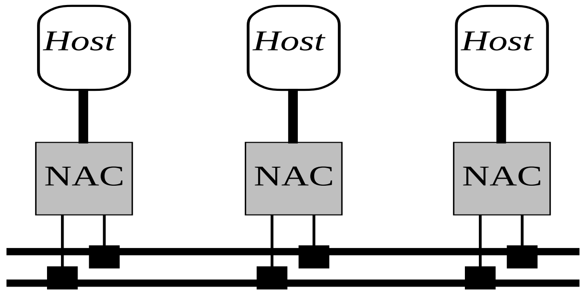

Figure 1: The communication system

To end with, this architecture allows using an approach inspired on the centralized two-phase commit protocol [Gray78], to build a powerful yet lightweight communication primitive, that we call *Atomic Multicast Protocol (AMp).*

3 Atomic Multicasting

The *Atomic Multicast Protocol (AMp)* presented in this paper provides highly parallel reliable group communication primitives, which are useful in a number of applications, such as client/server or coordinator/cohort interactions, distributed transactional and database systems, fault-tolerant mechanisms based on replicated computations.

The facilities of broadcast Lans are used to improve the performance of the protocol. Namely, location independent multipoint multicasting is achieved by using MAC logical addressing.

Communication takes place inside groups of processes, which map onto communication objects, called gates [Powell88]. All gates in a multicast gate group (MGateGroup) are identified by the same logical address. These groups form restricted universes of communication objects, and parallelism is achieved, externally by running as many instantiations of AMp as there are different MGateGroups, and internally by letting the group members transmit competitively.

The AMp provides a service which in short, ensures that all correct recipients in a group accept the same messages in the same order, within a known bounded time [Verissimo87d]. The protocol is resilient to omission and stopping faults during its execution. A bounded omission degree [1] is tolerated, but any number of nodes may fail. In case of emitter failure, a termination protocol [Skeen85] is ran by a monitor, to ensure completion of the transmission in course.

Node failures and group membership changes are indicated to the gate users, consistently ordered relatively to the information messages.

Message acceptance is atomic: it is either the message emitted, or a "null" message (reject) if at least one of the recipients is not accessible [Verissimo87d].

Accessibility is, to the authors' knowledge, a novel concept in reliable broadcasting. It expresses the capability of accessing the communication facilities to accomplish a broadcast transfer. It is of paramount importance in non-replicated reliable broadcast architectures, once it allows modelling certain temporary impairments, or accessibility constraints, such as buffer overflow in remote NACs, or network unaccessibility due to token loss/recovery.

Each component subject to unaccessibility has an Accessibility Structure, controlled by the Layer Management Entity (Lme), where the causes (accessibility constraints) and timing (duration, rate) are defined. For example, a Gate (or the whole NAC) that gives an overflow response too often, or for too long, in order that the conditions of its accessibility structure are violated, will be considered failed and fault treatment will be performed by the Lme, so that the Gate is closed, or the NAC is shutdown.

Accessibility, by defining what is an acceptable temporary degradation, establishes the frontier between temporary fault and failure. It thus provides a formal basis for defining timeliness in otherwise rather unpredictable single-Lan architectures. In consequence, an upper time bound to reach agreement can be defined. The protocol achieves loosely synchronous agreement, as detailed in section 3.3.1.

[1](Omission Degree (Od) is the number of consecutive omission errors that a correct NAC can do. It is bounded by design (fail-silent assumption), so that a NAC exceeding it is forced to shut down.

Message delivery order is maintained consistent inside each group, i.e. partial orderings, as opposed to global ordering, are ensured for the several MGateGroups in an implicit way, based on the sequence of message passing in the network: the Receive Queue order. In subsequent references in the text, order means partial order inside an MGateGroup (MGG).

3.1 Multicast Groups

Group communication is balanced multipoint, i.e. any member may transmit to all, and is inclusive, i.e. the sender receiving itself the message emitted, fulfilling the order property. As shown in Fig. 2, any group member may initiate a transmission, and for example m_2 will arrive everywhere after m_1 and before m_3, including in station 2. Balanced inclusive atomic multicast eases the programming of replicated and cooperating applications, like for example distributed state machines [Schneider88, Powell88].

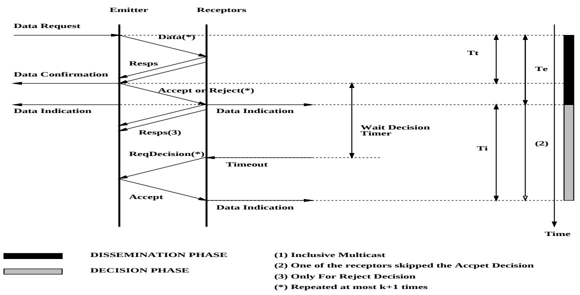

Figure 2: Protocol timing

The abstract StationIdentifier (SI), used by the AMp, is the unique StationPhysicalAddress or MAC address, which is the frame source address. The destination address is the group address. A station may belong to any number of MGateGroups and the number of MGGs in the Lan is only bounded by implementation limits.

Multicasting is transparent, in the sense that only one message is sent and there is not a previous knowledge by the user, of the whereabouts or number of destinations. However, group privacy is ensured, by making sending or receiving authorization depend on the knowledge of the group LogicalId, acquired prior to joining the group.

Processes (mapped on gates) join and leave a group at any station, through local gate opening or closing operations. Due to the nature of the communication architecture, we avoid the sophistication of truly dynamic join/leaves such as found in [Cristian86, Schneider84b]. This is namely because omission faults may occur, and the protocol is not tightly synchronous, allowing transient differences in the local AMp machine states of group members. Instead, those actions are performed in a privileged state, which ensures that the group views in all MGateGroup members are updated consistently, in relation to the ordered flow of information (section 3.4). The operation is done in a way such that it is perceived as being dynamic in most situations. [Joseph88] describes a similar pseudo-dynamic effect in his group join/leave protocols.

3.2 Multicast Communication Entities

This section gives an overview of the entities, related to multicast communication, which must be created whenever a gate is opened in a station (Fig. 3): the Local Emitter Machine (LEM) and the Local Receiver Machine (LRM)(together they form the Local Group Communicator (LGC)), the Local Group Monitor (LGM), the Group View (GV) and the Group Receive Queue (GRQ).

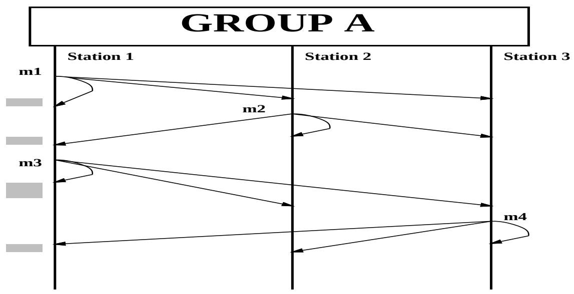

Figure 3: balanced and Inclusive Multicast with Loose synchrony

The LEM and the LRM provide detection of failed group members, in order that the protocol may terminate with the remaining correct members, or be terminated under monitor control, in case of emitter failure. Information is provided, to be used by the Local Group Monitor (LGM) in deferred error recovery and fault treatment procedures.

The Local Group Monitor is the entity that ensures coherence of the Group Views among the several group members, executing the procedures of insertion and removal of group members and the recovery from error situations. The LGM will be the subject of section 3.4.

3.2.1 Group View

Error detection is done on a transfer-by-transfer basis, and relies on consistency of the group view by each member. The minimum information needed is the ConciseGroupView (CGV), which contains only the number of group members.

The CGV is used by the Local Group Communicator or by the Local Group Monitor to detect station failures or undelivered frames (omission errors): for instance, if an emitter requests acknowledgement to a frame, it can compare the number of acks received with the CGV in order to detect the presence of an error.

However, while the Concise Group View is enough for station failure detection, recovery and thus maintenance of group views, requires the identification of the failed stations, through an investigation procedure.

An appealing method to allow fast identification of failed stations is the permanent maintenance of a complete list of member identifiers: we call it ExtendedGroupView (EGV). This approach, being very efficient, may also be very expensive in terms of storage since a station identifier uses six bytes and many MGateGroups, having several members, may coexist in the same station. In consequence, a compaction technique must be used in order to make effective use of the ExtendedGroupView [2].

3.3 The Protocol

The AMp relies on an atomic commitment philosophy [Gray78]. The *Two-phase Accept* principle relies on the *Fail-Silent* property of the attachment controllers: the halt-on-failure attribute allows

[2] A description of the actual implementation can be found in [Verissimo88d]

centralizing protocol execution, taking advantage of the efficiency of commit protocols compared to byzantine agreement protocols.

The protocol, sketched in Fig.4, starts with the emitter sending an information frame, implicitly querying the recipients if they can accept it (DISSEMINATION phase). This phase is implemented with transmission- with-response rounds (TxwResp), and ends after reception of all expected responses. In worst case, it may take as many rounds as the allowed omission degree plus one (w+1).

Upon analyses of the responses, the emitter issues its decision, either ACCEPT or REJECT (DECISION phase). Accept is only issued when unanimity can be reached. For example, when one of the NACs is unaccessible, the frame is rejected (by all). Worst case duration of accept decision is w+1 rounds.

3.3.1 Termination

In a previous paper [Verissimo87d] where a VLSI oriented protocol was presented, TxwResp was also used for the *Decision* phase. This version however, can use some optimization, to lighten up the software cost of each transfer.

Using the convenient assumptions, we can build a robust quasi-synchronous [3] atomic multicast protocol, optimized for no-fault operation, using explicit Decision, with acknowledged Reject, and negatively acknowledged Accept (Fig.4).

[3] The protocol displays loosely-synchronous termination: T_i may be non-zero (Fig.5), but worst case T_e is known and bounded. We term quasi-synchronous, a group protocol that terminates simultaneously at all sites, with a high probability (i.e. in absence of faults). The difference to tightly-synchronous protocols is that the notion of lock-step is not embedded in the communication system, and so global clock synchronization is not required. Quasi-synchrony is enforced with a low granularity in single segment Lans (that of the round-trip delay), by timer based during-transfer synchronization. The difference in granularity between the MAC and user level timings makes the simultaneity assumption reasonable. The user processes themselves may run in lock-steps using this primitive, provided they synchronize themselves to a global clock, and progress in worst case T_e increments. Quasi-synchronous protocols are however more efficient for applications using asynchronous or bursty invocations, but requiring fast group response.

In absence of faults, the protocol terminates in two rounds. If faults occur, the execution time grows accordingly to their number and severity, up to the worst case figures. The protocol performance is evaluated by the following times:

- *Transmission Cycle Time* (T_t): The time the emitter spends sending a frame.
- *Execution Time* (T_e): The time between the SEND_DATA.req primitive and the issue of the last DATA_REC.ind for that frame.
- *Inconsistency Time* (T_i): The time between the first and the last DATA_REC.ind for the same data frame at any two different receivers.

These times vary for different station failure and omission error scenarios. A detailed study of best and worst T_t, T_e and T_i times is omitted for space reasons [4].

3.3.2 Assumptions

We will present below the set of fundamental assumptions that guarantee safety of operation of our two-phase-accept protocol implementation, namely the optimized termination approach.

- *ASSUMPTION 1*: At any time, at most one AMp LocalEmitterMachine (LEM) is running at each node.

A LocalEmitterMachine, once started, executes atomically, i.e. it is not preempted by other emitting actions, for example, from the Active-Monitor.

- *ASSUMPTION 2*: Each node runs as many AMp LocalReceiverMachines (LRM) as the currently active atomic multicast transmissions it takes part in, managing the state of the relevant GroupReceiveQueues.

[4] This study is presented in [Verissimo88d]

Assumption 2 is the source of external parallelism in AMp: unlike other approaches proving global order [Chang84, Cristian85], AMp enforces partial orderings inside groups of users, like in [Birman87], and in consequence several parallel executions for different groups run simultaneously. On the other hand, internal parallelism, meaning "inside the group", is ensured in a light way: group members just run their LEMs competitively, in a fully decentralized fashion. Several transmissions from different LEMs may be initiated simultaneously; order and agreement are achieved by implicit network passing sequencing and frame numbering, based on Assumption 4 below. This results simpler and more reliable than other approaches, like the publishing approach strategy in [Chang84], where requests pass through a central token holder, or causal ordering [Lamport78a, Birman87], involving significant context exchange to enforce logical ordering of messages.

As a matter of fact, causal broadcasts in the sense of [Birman87] are virtually ordered with our method, at no cost. We will use Fig.2 to demonstrate this fact. By Assumption 1, m_1 will always arrive everywhere before m_2. On the other hand, supposing that delivery of m_1 at node 2 would trigger the send request for m_2, they are causally related ($m_1 \rightarrow m_2$), so m_2 must arrive everywhere after m_1. It is easy to see that by Assumption 4 below, messages are ordered by their passing in the network. So the only order that may be subverted in delivery is the order of sending by different nodes to the same group, if omissions occur. If m_2 is sent as a result of delivery of m_1 at some node, m_1 must have been in that node's GroupReceiveQueue before m_2. Thence, the same is true for all the group receive queues, because the two-phase-accept protocol ensures that the emitter only proceeds to an accept decision after all the (remaining) correct recipients have the message in their queues. Messages m_1 and m_3 are also causally related, because they come from the same emitter ($m_1 \rightarrow m_3$). As a consequence of Assumption 1 [5] , they are delivered everywhere in the order of sending.

[5] In fact, Assumption 1 is stronger than necessary to achieve this. It is enough that LEMs run atomically. The actual phrasing of A.1 reflects our restrictive approach to parallelism, discussed in this section.

Speaking of parallelism, please note that there is a subtle restriction to it, materialized by Assumption 1. As a matter of fact, Assumption 1 implies that if a number of requests to different MGateGroups are issued simultaneously (or almost) at one node, they will effectively be sequenced, request i+1 being served only after request i transmission completion (Fig.6a). This is opposed to letting one node run several transmissions to different MGateGroups (Fig.6b). The restriction is only imposed by implementation reasons; as a matter of fact, if we wish to impose no restriction to the number of groups a node may belong to, significant context must kept to support full parallelism, meaning both space and processing overhead. But conceptually, the protocol fundamentals remain unchanged in either approach.

- *ASSUMPTION 3*: If Decision=Reject, the emitter positively confirms that it is received by all nodes belonging to its GroupView.

Assumption 3 together with Assumption 1, allow safe use of an error recovery algorithm detailed later in the text, which uses no context about previous transmissions. In consequence, we avoid maintenance of histories, or lists, etc. like it is found in other approaches.

- *ASSUMPTION 4*: Messages are delivered to a Gate(i) user in the order they are in GroupReceiveQueue(i).

In absence of faults, Assumption 4 alone would ensure consistent order of delivery in all recipients. Section 3.3.5 details the procedure for assisting Assumption 4 to maintain order, in the presence of omission faults. This order is the order of sending, *only* for the frames coming from the same emitter.

3.3.3 Multicast Frames

This section introduces most of the basic functionality of the protocol, while presenting the format of the frames used. Details about the protocol mechanisms will be given in the next section.

All multicast frames possess the basic structure:

MFC	DATA

MFC is the Multicast Frame Control Field, containing relevant information for protocol execution. The data field may contain either information or control data depending on the frame type. The structure of the MFC is :

GI	SA	FT	AT	MTN	SL	MDN

Where :

GI the group identification, is a destination address. All the stations belonging to the group will receive the frame.

SA is the station physical address of the emitter (source address). The emitter must also belong to the destination MGG (an exception, as will be seen in section 4, occurs during the insertion of new members in the MGG).

FT is the Frame Type field. The frame type specifies the function of the frame and indirectly, the structure of the frame-s data field.

MTN is the Multicast Transmit Number. Each emitter numbers its outgoing frames with increasing values of MTN. The pair SA/MTN is then a unique identifier of a frame and is called the Multicast Frame Identifier (MFI).

MDN is the Multicast Data Number and is only present in frames carrying the Order attribute (see Attribute List). As the MTN, it always assumes increasing values, however, retransmissions of the same data frame have the same MDN. The MDN is used to identify different retransmissions of a frame so the pair SA/MDN is called Multicast Data Frame Identifier (MDI).

SL is the suspension level and is only present in frames which carry the Suspension attribute (see Attribute List and 3.4.3).

AT is an Attribute List. Each attribute specifies a certain action to be performed by the frame recipient. Attributes may be combined to produce high level behavior patterns which make protocol implementation versatile and evolvable, if compared to fixed type frames, found in 802.x protocols. For example a frame may use Acknowledge to implement a transmission with response round, or be combined with Flush, to inhibit transmission pipelining.

The available attributes are:

Acknowledge (Ack) The acknowledge attribute, or simply Ack, states when responses to the frame are expected or not. If this attribute is used, a number of responses given by the emitter Concise Group View will be expected after the frame emission. Frames that do not carry this attribute are not acknowledged by their recipients.

SuspendTraffic The SuspendTraffic attribute is used to stop the group multicast traffic. This attribute is always associated with a Suspension Level. The emitter of the frame with this attribute is called Active Monitor. After the reception of a frame with this attribute the recipient memorizes the identification of the active monitor and the frame Suspension Level. No further frames are accepted unless emitted by the current Active Monitor, or having a higher Suspension Level. In the latter situation, the emitter of the new frame becomes the new Active Monitor.

Control frames, such as response and decision frames, are exceptions to the previous rule and are always accepted regardless of the suspension state.

Flush The flush attribute states that the frame should only be analyzed when it gets to the head of the GRQ. After its arrival to the receive queue the frame waits until all the previous frames are either accepted or rejected. If the frame also carries the Ack attribute this implies that the acknowledge is not sent immediately after its reception but only when it reaches the head of the queue.

WaitDecision This attribute specifies that the frame will be followed by a Decision frame, which will condition its delivery. A Decision frame is either accept, in which case the frame is delivered, or reject, implying that the frame be discarded. The recipient triggers a timer (WaitDecisionTimer), and on its eventual expiry, the occurrence of an omission error is assumed.

Resume This attribute is used to restart a suspended traffic. The emitter of the frame suspending the multicast traffic can send any number of frames, before resuming the traffic with a frame carrying this attribute. For instance, the emitter may want to win the exclusive control of group communication, then send one or more multicast frames, the first of which carries the suspension attribute, and finally resuming it with the correspondent decision carrying the resume attribute. This is a useful feature to privilege traffic in real-time communication.

Order This attribute specifies that the frame must be ordered relatively to other frames carrying this attribute. The order is that of arrival to the GroupReceiveQueue. This means that a deliverable (accepted) frame may stay in the queue if a previous frame is still waiting for a decision.

3.3.4 Multicast Machine

We have already seen that communication is performed inside an MGateGroup(i) in a balanced way, in that everyone may send to all, including itself, with consistent order of delivery inside the group.

Each station creates an instantiation of AMp for each new locally open gate, whose components were detailed in section 3.2 (Fig.3). These instantiations run in parallel, sharing the medium, and the very little synchronization that exists is concerned by enforcing the assumptions made in section 3.3.2.

Only one LocalEmitterMachine can be sending at a time. So atomic multicast transfer requests queue at the AMp SAP. As we explained earlier in the text, we believe that this does not present a major drawback, being instead a natural source based flow control mechanism, fed back by current network occupation and destinations processing capability.

A GroupReceiveQueue (GRQ) may have several pending frames, though at most one frame from each group member. This means that all members of a group may be sending simultaneously (*internal parallelism*) which represents an efficient support for replicated client/server and competitive clients communication.

Note that there are two mechanisms by which an emitter machine can be activated. The first is the arrival of a send request, at the AMp S-AP, performed by a service user. The second is the detection, by a Local ReceiverMachine, of the failure of an emitter, which activates a prioritary recovery procedure, at group level.

To ensure that only one LocalEmitterMachine is active at a time, it is then necessary the presence of a scheduler which queues all requests, activating the EmitterMachines sequencialy, giving priority to the activation of GroupMonitors (Fig.7).

3.3.5 Multicast Transfer

From the last sections, it became evident that attributes of the transmitted frames are combined in order to obtain the desired protocol response. A multicast transfer is initiated by the protocol coordinator, the Emitter(E), by sending a multicast frame, with Ack and WaitDecision attributes, and if order is desired, with Order attribute.

The Dissemination phase then proceeds as follows:

After transmission, E will expect a number of responses indicated by its Group View, with a predefined response time (RespT). When all responses arrive or RespT has expired, they are analysed and if some recipient is not accessible (cannot accept the frame), a Decision= Reject is propagated.

If all received responses are of "can accept" type and (only) if all recipients responded, according to the sender GroupView, a Decision= Accept is propagated. If there are responses missing, the data frame is retransmitted. The retransmission will carry the same MDN of the original data frame, but a higher MTN. Then, Assumption 4 needs to be complemented, so that the ordered delivery property is still fulfilled. A distributed algorithm is used, to recover from omissions in the Dissemination phase:

a) A recipient must discard any old data frame pending in its GroupReceiveQueue which has the same MDN of a freshly received data frame.

b) Retransmission restarts the protocol, in terms of response control: the dissemination phase is ended when either there is a TxwResp round where all responses are received, or a retry limit is exceeded. The method respects ordering inside the group, due to Assumption 4, because only frames which were totally disseminated are retained.

This procedure is called recursively until a retry limit is reached. If any station does not answer within the retry mechanism it is considered failed. This allows timely termination: an accept Decision may be sent if all the remaining stations are accessible. After the Decision phase, fault treatment is done: the Local Group Monitor is invoked to reestablish group coherence. Stations considered as failed are removed from the Group View.

The Decision phase is implemented in the following way:

Reject frames always carry the Ack attribute (Assumption 3). So, the Emitter ensures that Reject is received by all group members, retransmitting it when omission errors occur. As above, a station that does not answer within the retry limit, is considered failed.

The Accept frame is sent without attributes, only once, in the present imqplementation. A timeout mechanism, at the recipients, covers the omission faults : after the reception of a multicast frame carrying the WaitDecision attribute, a timer is started with a predefined WaitDecisionTime.

If no decision is received within this time, the recipient requests a decision to the emitter (Fig.5). Note that in case of Reject, an emitter only starts a new transmission after assuring that all the group members received the Reject (Assumption 3). So, when an emitter receives such a decision request it can answer with an Accept without any knowledge of the past, or proceed, if it was still transmitting the frame. The recipients will retransmit the decision request, until the retry limit is exceeded. When that happens, the emitter is considered failed and the Group Monitor is called upon, to reestablish group coherence.

The Accept Decision being the most frequent completion of the protocol, we chose to make it negatively acknowledged, which optimizes transmission rate, due to the pipelining effect, in absence of faults. However, the detection of omission faults in a negatively acknowledged transmission is slower than its positively acknowledged counterpart, since a recipient must wait a worst case transmission time, before issuing a decision request frame. However, a performance improving consequence of Assumptions 1 and 3 is that a recipient may accept a pending frame if it receives a new frame from the same emitter. This is expected to avoid the transmission of decision request frames, in situations of fair to high traffic, maintaining the pipelining effect. In consequence, efficiency, in the presence of Decision omission faults, will be kept very near the no-fault situation.

To end with, it is worthwhile mentioning that with the underlying philosophy of the AMp, of specifying a transfer by transfer quality of service based on attributes, a modified primitive may readily be created, requesting fully acknowledged Decision, by sending both Accept and Reject with the Ack attribute. Conversely, relaxed forms of the primitive can be readily obtained, by not using certain attributes, as a tradeoff for better performance.

3.4 Distributed Group Monitor

The Distributed Group Monitor (DGM) is an entity which executes, under a privileged state, critical activities relevant to correct operation of the protocol. Namely, it maintains consistency of the Group View, recovering from station failures. Additionally, it runs the termination protocol in case of emitter failure.

The DGM is composed of several Local Group Monitors (LGM), executing locally at every group member, which closely cooperate. Each MGate-Group possesses is own distributed group monitor, which executes with total independency of the monitors of the other groups. Within the context of a MGG all the LGM are usually in the Standby state. When an LMG wants to actively interact with the other LGMs of the same group it tries to assume the Active State (see 3.4.2). In a MGG only one LGM can be, at a given time, in the Active State, all the others must be in the Standby state.

When an LGM is in the active state, it is executing an Active Monitor Action (AMA). Since during the AMA the coherence of the group view cannot be ensured, the multicast traffic is suspended in the group by the DGM, being resumed at the end of an AMA when the group coherence is reestablished. If an Active Monitor fails, it is be replaced by another LGM who transits, after the detection of the failure, from Standby to Active state.

The DGM also controls the group membership : joins and leaves from the Multicast Gate Group require activation of the Distributed Group Monitor so that all Group Views change consistently.

3.4.1 Group Monitor Activation

The DGM is activated whenever an incoherence is detected or when an action which may introduce incoherences in the group view is to be performed. Incoherences are detected by the Local Group Communicator, (for ex., a station failure). Opening and closing of gates, which may introduce temporary incoherences in the Group View, are requested by the Layer Management.

The active monitor procedures to open and close gates will be explained in detail in section 4. The active monitor actions which are executed in order to recover from failures will be detailed in the following paragraphs.

Note that if the failure is detected by the Local Emitter Machine, the monitor is activated immediately after the detection of the failure. However, if the failure is detected by a Local Receive Machine, the activation of the group monitor may be delayed since other Local Emitter Machine may be active at the time, as explained in section 3.3.4.

3.4.2 Active Monitor Election

The need for the DGM intervention can be detected simultaneously at two or more group members so it is expected that several Local Group Monitors (LGM) will compete for the activity. To ensure that only one monitor will become active, the first frame sent by a candidate carries the Suspension attribute. The first active monitor will then suspend the multicast traffic on the MGateGroup. Other LGMs will find the traffic suspended and will return to the Standby State.

However, this simple solution must be enhanced to avoid deadlock and contention. Deadlock occurs when an Active Monitor fails, leaving the traffic suspended. Contention can occur in the presence of an omission fault during the competition for the activity if two different monitors lock different subsets of the recipients.

These two problems are solved with a mechanism based on the association of an integer value, the Suspension Level, to the suspension attribute. When a receiver is suspended it stores the Suspension Level associated with the suspending frame, this value is the Current Suspension Level. If another frame, with the Suspension Attribute, is received during the suspended state, this frame is rejected unless it carries a suspension level higher than the current, in which case its emitter will become the new Active Monitor (and the current suspension level is actualized). With this mechanism, a faulty Active Monitor cannot suspend indefinitlly the traffic since it can be preempted by a correct monitor.

To solve the deadlock problem a timer is started when the traffic is suspended. Whenever the Active Monitor sends a new frame the timer is restarted. If the AM remains silent for a long time, the timer will expire and the failure of the AM is assumed. The Standby Monitor who first detects the failure becomes active incrementing the Suspension Level. The contention problem is also easily bypassed, when one monitor detects the presence of a contention (receiving some responses reporting the traffic suspension and others acknowledging its frame) it retransmits the frame incrementing the SL. In both situations, the new frame with a higher suspension level than the current SL will be accepted by all the suspended recipients establishing a new Group Monitor.

3.4.3 Active Monitor and Failed Stations

Whenever a station fails, the DGM is called to reestablish the group coherence. The LGM which wins the activity must, if needed, finish the transmission interrupted by the failure and disseminate a new group view. In order to accomplish this objective the monitor must execute two rounds.

These rounds include the identification of failed stations, to search for the presence of pending frames from the failed emitters, the decision to accept or reject those frames and finally disseminate the new group view.

The decision process for the frames pending from failed emitters is the most difficult step of all monitor actions. The Active Monitor must ensure that the pending transmissions are finished correctly. This means that the AM must investigate if the frame had been accepted by any member of the MGG and, if so, all the other members must also accept the frame. If none of the MGG members had accepted the frame, it can be rejected. This information can be only collected if the recipients keep some knowledge of the past transmissions since many other accepts/ rejects can be received (from other emitters) prior to the detection of the failure.

Each station keeps a table with the MDNs of the last frames accepted, one for each emitter in the LAN. The active monitor reads the contents of the MDNs List (called StationMdnTable), on all group members, collecting information about the entries corresponding to the failed station. After this information is received, the active monitor chooses the highest value and disseminates it during the next phase.

The value of the highest MDN collected in the previous phase is disseminated trough all the MGG members. These search for the presence of a frame in the receive queue emitted by one of the failed stations listed in the Exception Decision Frame. If such frame exists its MDN (rxMdn) is compared with the received MDN (decisionMdn) and a decision is taken using the following function:

Exception decision : if (decisionMdn $\geq$ rxMdn) then Accept else Reject.

The first round covers the identification of the failed stations, the search for pending frames and, finally, the investigation of the group StationMdnTables. The investigation frame carries the identification of the failed stations. The responses will carry a list of triplets containing the id of the failed station, the content of the GroupMdnTable for that station and a bit stating if there is any pending frame in the receive queue sent by that station.

After this a second round is performed including the dissemination of the decision for the pending frames and the dissemination of the group view. The active monitor sends a frame where the new group view is disseminated and if needed, a list of exception decisions is included. An exception decision is simply a pair stationId/ Mdn where the Mdn is the highest Mdn received in the first round. The recipients will use this value to finish the pending transmissions.

The first frame sent always carries the suspend attribute and the remaining round is only performed if the monitor wins the activity. The last frame carries the resume attribute. If more stations fail during this procedure it must be restarted increasing the number of frames between the traffic suspension and resumption.

4 Joining And Leaving The Group

The gate pdus are frames sent to change the group membership, inserting or removing a station from the MGG. These frames called respectively OpenGate and CloseGate are sent by request of the Layer management entity.

Since these two frames change the Group View they interfere with the other transmissions on course the traffic is flushed and suspended prior the processing of the gate pdus. This ensures that no frame is affected by the gate pdus.

Note that the Flush of the multicast traffic is only possible in the absence of failures. If an emitter fails, the decision will never be sent and the frame rests in the receive queue until an exception decision is sent by an Active Monitor. Due to this fact, if a recipient detects a failure while the traffic is being flushed it informs the sender of the flushing frame, who must solve the incoherence. This means that the recovering phases described in the previous chapter can be performed by a monitor who became active to perform a gate pdu transfer.

The CloseGate action is very simple. A frame with the identification of the station to be removed is sent with the Flush and Suspend attributes. If the traffic is not already suspended by another group member, the end of the current transmissions is waited before an Accept frame, resuming the traffic is sent. When the Accept frame is received all the group members change their group views and restart the multicast traffic.

The OpenGate action needs a extra step since the station who desires to enter the MGG must, prior to the emission of the gate pdu, obtain a view of the group. The OpenGate is then started with an GetView investigation which is sent with the Suspend and Flush attributes as above. This ensures the correctness of the View obtained.

If no response is received to the GetView investigation, the frame is retransmitted. If the silence persists after the maximum number of retries, the new station assumes to be the first member of the MGG and initializes the group view.

Note that the local group receive machine is activated at the reception of the own GetView frame. This means that if another GetView frame is emitted during the OpenGate procedure, it will be rejected by the new suspended receive queue, and the late station will give up. This assures that only one station can initialize the group, even if more than one station start the open procedure at the same time.

If the GetView investigation obtains response it is followed by the pdu which is retransmitted if needed. When the pdu is acknowledged by all the group members an Accept is sent, changing the Group View, inserting the new station in the MGG and the traffic is resumed [6].

[6]The cost of these actions is presented in [Verissimo88d]

5 Conclusions

We have presented a protocol which provides atomic multicasting at the Data Link layer. The interest of this low level approach, from an architectural point of view, is that it takes advantage from the inherent facilities of the underlying Lans, from which we name broadcast capability, multicast addressing, token-based access control, priority scheduling, low-level synchronization.

Either by using a software shell on top of the exposed interface of VLSI controllers, or/and hardware implementations [Verissimo87d, MCS88], one of our aims was efficiency at the low cost of a non-replicated Lan.

The present software implementation has been thoroughly verified, and performance tests are now being run. The protocol was optimized for the no-fault situation, i.e. by using negatively acknowledged accept and implicit accept. The implementation tried to restrict as little as possible the high parallelism achievable with the independent AMp(GateGroup) instantiation concept. The protocol is easily evolvable: this versatility is given, among other factors, by the attribute based quality of service and GroupView type choices at service invocation time.

The communication primitive provided was proven to be of great help in assisting the implementation of distributed computations. A report soon to be issued describes the programming of the Amaze game [Berglund84] as a replicated state machine [Schneider88]. The state machine uses directly the AMp, with no further "glue", for its interactions. In the Delta-4 system [Powell88], AMp is used as the low-level primitive in an OSI-like stack, which has fault-tolerance mechanisms based on replication and voting, implemented at the Session layer. The resulting communication architecture provides support for a state-machine based computation system.

Acknowledgements

- The authors wish to thank José Alves Marques for his suggestions throughout the elaboration of this work. Jerome Hammel (Bull)

contributed the compaction technique for storing Extended Group Views, which allowed us to currently use EGVs, at a very low space cost.

References

[Babaoglu85a] O.Babaoglu, R.Drummond, *Streets of Byzantium: Network Architectures for fast Reliable Broadcasts*, IEEE Transactions on Software Engineeering, nr 6, June 1985

[Berglund84] E.J.Berglund, D.Cheriton, *Amaze: A Distributed Multi-Player Game Program using the Distributed V Kernel*

[Birman87] K.Birman, T.Joseph, *Reliable Communication in the Presence of Failures*, ACM Tran. on Computer Systems, Vol. 5, nr 1, February 1987

[Chang84] J.Chang, N.Maxemchuk, *Reliable Broadcast Protocols*, ACM Transactions on Computing Systems, Vol12, nr 3, August 1984

[Cheriton85a] D.Cherinton, W.Zwaenepoel, *Distributed Process Groups in the V-Kernel*, ACM tran. on Computer Systems, V.3, nr 2, May 1985

[Cristian85] F.Cristian, H.Aghili, R.Strong, D.Dolev, *Atomic Broadcast: From Simple message diffusion to Byzantine Agreement*, procs. FTCS15, Ann Arbor-USA, June 1985

[Cristian86] F.Cristian, H.Aghili, R.Strong, *Clock Synchronization in the Presence of Omission and Performance Faults, and Processor Joins*, proc. FTCS16, Viena-Austria, July 1986

[Dolev83] D.Dolev, R.Strong, *Authenticated Algorithms for Byzantine Agreement*, SIAM J. on Comput. 12, November 83

[Gray78] J.Gray, *Notes On Database Operating Systems*, in Lectures Notes in Cumputer Science, Springer Verlac, 1978

[Hadzilacos84] V.Hadzilacos, *Issues of Fault-Tolerance in Concurrent Computations*, Ph. D. Thesis - TR11-84, Harvard Univ., June 84

[Joseph88] T.Joseph, k.Birman, *Reliable Broadcast Protocols*, Artic88, An Advanced Course on Operanting Systems, Trömso, Norway, July 1988

[Lamport78a] L.Lamport, *Time, Clocks, and the Ordering of Events in a Distributed System*, Comm. of the ACM, Vol. 21, nr 7, July 1978

[Lamport82] L.Lamport, R.Shostak, M.Pease, *The Byzantine Generals Problem*, ACM Transactions on Prog. Lang. and Systems, Vol. 4, nr. 3, July 1982

[Laprie86] J.C.Laprie, *Dependability: A Unifying Concept for Reliable Computing and Fault-Tolerance*, LAAS-France, TR 86357, December 1986

[MCS88] , *MCS Functional Specifications - Token Ring Access Method and Physical Layer*, DELTA-4 Tech. Rep, BULL, August 1988

[Powell88] D.Powell, D.Seaton, G.Bonn, P.Veríssimo, F.Waeselynk, *The Delta-4 Approach to Dependability in Open Distributed Computing Systems*, Delta4 Proj., procs. of 18th FTCS, June88, Tokyo-Japan

[Schneider84b] F.Schneider, D.Gries, R.Schlichting, *Fault-Tolerant Broadcasts*, North-Holland, science Computing Programming 4, 1984

[Schneider88] F.Schneider, *The State machine Approach: A Tutorial*, Springer Verlac, Proc of workshop on Fault-Tolerant Distributed Computing, New York USA, 1988 (to appear)

[Skeen85] D.Skeen, *Dtermining the Last Process to Fail*, ACM Trans. on Computer Systems, V.3, nr 1, February 1985

[Verissimo87d] P.Veríssimo, L.Rodrigues, J.Marques, *Atomic Multicast Extensions for 802.4 Token-Bus*, proc. FOC/LAN 87 Conference, Anaheim-USA, October 1987

[Verissimo87g] P.Veríssimo, J.Marques, *Reliable Broadcast on Standard Lans*, INESC Tech. Report, Nov. 1987, rev. Sep. 1988

[Verissimo88b] P.Veríssimo, *Redundant Media Mechanisms for Dependable Communication in Token-Bus Lans*, proc. 13th Local Computer Network Confer., Minneapolis-USA, October88

[Verissimo88d] P.Veríssimo, L.Rodrigues, M.Baptista, *Atomic Multicast Communication on a Token-Bus Lan*, INESC Tech. Report, Sept. 1988