Deterministic multiple access protocols for real-time local area networks

Gerard Le Lann

INRIA
ENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.(3) 954 90 20

Rapports de Recherche

N° 246

DETERMINISTIC MULTIPLE ACCESS PROTOCOLS FOR REAL-TIME LOCAL AREA NETWORKS

Gérard LE LANN

Octobre 1983

DETERMINISTIC MULTIPLE ACCESS PROTOCOLS FOR REAL-TIME LOCAL AREA NETWORKS

Gérard LE LANN
Projet SCORE
September 1983

ABSTRACT

Real-time local area networks are needed in a large number of automated applications where promptness, robustness and flexibility requirements are stringent.

Examples of such application areas are flexible manufacturing systems and airborne/shipborne/spaceborne systems.

In this report, two deterministic multiple access protocols intended for local area networks which use passive broadcast busses are presented.

RESUME

Les réseaux locaux temps réel sont destinés aux applications automatisées pour lesquelles les contraintes de promptitude, robustesse et flexibilité sont particulièrement sévères. Parmi ces applications, on trouve les systèmes de production flexible et les systèmes embarqués (air/mer/espace). Ce rapport décrit deux protocoles déterministes de gestion des accès multiples destinés à des réseaux locaux construits à partir de bus à diffusion passifs.

CONTENTS

1. INTRODUCTION

2. PRESENTATION OF THE PROBLEM

3. A DETERMINISTIC PROTOCOL BASED ON A UNILATERAL INTERFERENCE DETECTION CHANNEL

4. A DETERMINISTIC PROTOCOL BASED ON MULTIPLE CONTENTION CHANNELS

5. CONCLUSION

Notice : This report is based on two documents which were published as Projet SCORE internal reports in January 1983. The protocols described are still under investigation. This report is issued for early peer distribution.

1. INTRODUCTION

Real-time local area networks (LAN's) are bound to meet the requirements identified for real-time distributed computing systems in general [LEL83]. Robustness and promptness requirements are usually so stringent in real-time applications (e.g. automated factories, airborne/seaborne systems) that existing LAN's cannot be used without introducing problems of their own. Among the major problems, one finds :

- limited robustness : usually, only one, sometimes two, separate physical channels can be accomodated

- inadequate promptness : multiple access protocols currently employed (token-passing or contention based) do not guarantee appropriate upper bounds for channel access delays when message scheduling is time-dependent ; furthermore, these multiple access protocols do not adequately handle "emergency" messages.

Special purpose LAN's based on static time division multiplexing techniques do not usually meet robustness and flexibility requirements.

For a number of reasons which will not be presented here, we believe that Carrier Sense Multiple Access protocols are well suited to handle real-time traffic over the types of LAN's that we consider, which are briefly described in the next section.

2. PRESENTATION OF THE PROBLEM

2.1. Type of traffic

Messages to be transmitted may be short (e.g. control and monitoring messages) or long (e.g. computer files). The sizes considered range between a few bytes to several millions of bits. Messages may also be submitted periodically or aperiodically. With periodic messages, there is of course a lower bound for period values that can be accomodated.

2.2. Message types

The real-time LAN's we consider must be able to handle two types of messages, namely **regular** messages and **emergency** messages. Emergency messages should always have priority over regular messages. Submission of emergency messages is considered to be a very rare event. Over a large time interval, the probability that an emergency message has to be transmitted is very much negligible. However, it might be the case that over a short time interval, several emergency messages are submitted more or less simultaneously. When such an "avalanche" occurs, the objective is to transmit emergency messages first, possibly to the detriment of some regular messages.

2.3. Message scheduling

Intuition and queueing theory indicate that promptness requirements cannot be met in general when scheduling algorithms or servicing policies are based on static priorities. In a real-time computing system, one finds many reasons to make explicit use of physical time. Consequently, physical clocks are to be taken into account in the early design stages. Message scheduling will be conducted according to time oriented priorities. More precisely, every message will be carrying a physical timestamp indicating its transmission deadline. These deadlines will be used to perform deadline dependent scheduling and time dependent scheduling. It is also necessary to check at various points before transmission whether a message has missed its deadline or not. This is what we call promptness control. The servicing policies that are part of the multiple access protocols presented will make explicit use of deadlines to enforce promptness properties, which indicates that they will almost never be equivalent to LIFO ou FIFO strategies. These policies, to be efficient, require hardware implementation of such primitives as those selecting a customer satisfying a criterion in a waiting queue and such tools as timers. Recent hardware products, e.g. the Intel iAPX 432 microprocessor, suggest that such implementations are well within the possibilities of current technology.

We will then assume that every message m is associated a deadline T(m) which indicates at which latest time transmission should succeed. In order to perform some "optimal" system-wide message scheduling, each individual station that has messages waiting for transmission will select the message with the closest deadline T(m) and compute the instant T at which transmission of the corresponding message will be attempted, as follows :

$$T := \text{current time} + K * | T(m) - \text{current time} |$$

where K is either a constant or a variable such that $0 =< K =< 1$. Parameter K could be assigned a unique constant value or could be assigned different values at different stations in order to adapt to changes in local traffic conditions.

2.4. Local area networks architectures

The LAN's we consider are equivalent to passive broadcast buses. Their general characteristics are as follows :

- user requirements yield the need for operating n separate channels in parallel

- one of these channels is selected to be the "control channel" (CC), others being the "data channels" (DC's)

- every station is connected to each channel

- every message is multi-copied, all copies being sent simultaneously on all data channels

- stations have static names, ranging from 1 to S

- p, the maximum propagation delay on the channels is known ; p is small compared to the average transmission time of a message but large compared to the duration of a single bit

- stations know the value of the channel slot time a , with a > 2 p

- a collision is detected at most a time units after transmission initiation

- stations can detect EOC and EOJ signals

- stations can read headers of messages circulating on the channels.

Furthermore, the multiple access protocols needed are designed so as to operate efficiently over channels whose nominal bandwidth (B) ranges between 1 and 100 Mbits/s and whose length (L) ranges between 10 m and 1 km.

For these CSMA oriented protocols to work, the following inequation must be verified :

message length (bits) > 100 * B * L * F,

where L is in km, B in Mbits/s and where F (F > = 1) is a safety factor.

Let us briefly mention that for powerful LAN configurations, e.g. 100 Mbits/s and 1 km long channels, efficient channel utilization can still be achieved even when small messages are to be transmitted. Explaining why and how would be out of the scope of this report. Like some other proposals, (e.g. | CHL80 |, | GOL83 |, | MOL81 |, | POW81 |, | ROM81 |), the protocols presented are based on detection of collision, detection of end-of-carrier signal and detection of end-of-jamming signal.

One of the protocols makes use of the channel slot time, whose value is referred to as a. The only timing computations using slot time a that we allow ourselves are based on conventional features, as those used in the Ethernet-compatible Intel 82586 chip.

3. A DETERMINISTIC PROTOCOL BASED ON A UNILATERAL INTERFERENCE DETECTION CHANNEL

A unilateral interference detection (UID) channel performs a bit-by-bit logical exclusive OR operation on the signals that are concurrently present on the channel. Good output states of this (atomic) operation are either true or false. Only those message sources which read "true" are allowed to pursue transmission | POW81 |.

Let us now assume that a preamble is computed by sources for each message ahead of each of the source SQ's. A preamble is a fixed-length bit pattern that is sent ahead of a message. If preambles sent "simultaneously" are all different, a unilateral interference detection channel elects only one preamble to be

completely transmitted, thus allowing only one message to be sent on the channel, independently of the number of collisions. Our solution assumes high-order bits of preambles transmitted first and "0" winning over "1".

The control channel (CC) will operate as a UID channel while data channels (DC's) will be used as normal broadcast channels.

(i) The protocol

At time T, source s obeys the following algorithm :

```
LOOP
INLOOP
Sense CC until CC becomes idle ;
Wait B * a ;
Sense CC ;
IF CC busy THEN repeat INLOOP
END INLOOP
Send preamble on CC ;
CASE
        Winner :   BEGIN
                   IF DC's busy THEN
                   jam CC until DC's become idle ;
                   Transmit m on DC's ;
                   Exit LOOP
                   END
        Loser :    Nil (repeat LOOP)
END CASE
END LOOP
```

where :

B := 0 for emergency messages and for a message that has been involved in a collision

B := 1 otherwise.

(ii) Comments

Before and while executing the above protocol, promptness control is exercised by source s. Preambles could be static station names. If the degree of unfairness introduced in case of collisions is felt unacceptable, one could "distribute unfairness" by rotating virtual names among stations. This does not raise serious problems and will not be examined further. By restricting the UID mechanism to operate CC and by taking advantage of concurrent transmission of messages on DC's at their nominal speed and of preambles on CC, it is possible to overcome the limitations inherent to UID mechanisms. Indeed, bandwidth achievable on a UID channel is limited to 1/ 2p bits/s which is equivalent to verifying the following inequation :

$$B * L * F < 10^5,$$

where B is in bits/s, L in km and F (F > = 1) is a safety factor.

With this protocol, DC's nominal bandwidth is wasted only when message duration on DC's is smaller than preamble duration on CC. For example, assuming a distance of 1 km, bandwidth on a UID operated channel is limited to 50 Kbits/s approximately if F = 2. For a system supporting at most 256 sources, preamble length is 8 bits and preamble duration on CC is then in the order of 160 us. For message lengths ranging from 160 bits to 16000 bits, bandwidth achievable on DC's ranges from 1 to 100 Mbits/s. Therefore, the proposed solution circumvents the problem of performance limitation with UID mechanisms for a large class of LAN's.

(iii) Impact of faults

Faults that result in the breakdown of one of the DC's are kept transparent. When all DC's are down, it is possible to transmit messages on CC through the UID mechanism where, in the above algorithm, jamming of CC is replaced by the sending of m on CC. When the CC goes down, a new one must be chosen. A simple way of dynamically electing a new CC consists of choosing the up channel that has a specific property, e.g. smallest identity. The UID logic contained in each of the stations will then be used to interface each source with the newly elected control channel.

Of paramount importance are the faults that impact the UID mechanism. Transient faults would not disrupt its functioning but rather affect promptness properties. When occurences of faults get too frequent, a situation similar to a CC breakdown is created. Election of a new CC is then necessary. In a highly noisy environment, it might be interesting to include in message headers the identity of the CC currently in use. This is but one of many ways to take advantage of the existence of naturally redundant message transmission.

4. A DETERMINISTIC PROTOCOL BASED ON MULTIPLE CONTENTION CHANNELS

With this protocol, CC and DC's are operated as conventional CSMA-CD channels. The only variations relate to the strategies used when collisions are detected and to the exploitation of parallel channels.

Every station s is dynamically assigned an index X(s). Indexes take their values on | 1, S | for a system of S stations (all indexes should be different at the same time). When a message is being successfully transmitted on DC's, every station can read, in the message header, the index currently assigned to the transmitting station.

In order to provide emergency messages with very short access delays, it is better to assign stations which are allowed to generate such messages low value indexes. However, this is not mandatory for the protocol to work.

(i) The protocol

At time T, station s obeys the following algorithm :

CASE <emergency message> and <CC busy>:

sense CC and DC's until

either message is read entirely on DC's

or CC becomes idle ;

CASE message read entirely on DC's :

call Compute | Z(s), W(s) | ;

start jamming CC ;

call GetReady | Z(s), W(s) | ;

END CASE

CASE CC becomes idle :

call Message Transmission ;

END CASE

END CASE

CASE not <<emergency message> and <CC busy>> :

call Message Transmission ;

END CASE

Message Transmission :

```
WHEN CC idle DO
    BEGIN
    initiate transmission of m on CC ;
    IF no collision on CC THEN
    WHEN DC's idle DO transmit m on DC's
    ELSE
    BEGIN
    Z(s) := TRUE ;
    W(s) := X(s) ;
    start jamming CC ;
    call GetReady | Z(s), W(s) | ;
    END
    END
```

Compute | Z(s), W(s) | :

```
    BEGIN
    IF X (transmitter of message read) < X(s) THEN
    BEGIN
    Z(s) := TRUE ;
    W(s) := X(s) - | 1 + X (transmitter of message read) |
    END
    ELSE
    BEGIN
    Z(s) := FALSE ;
    W(s) := X(s)
    END
    END
```

```
GetReady | Z(s), W(s) | :

        LOOP BEGIN
        sense DC's for a * W(s) at most ;

        CASE message sensed on DC's :
        read message header ;
        call Compute | Z(s), W(s) | ;
        sense DC's ;
        WHEN DC's idle DO repeat LOOP ;
        END CASE

        CASE no message sensed on DC's after a * W(s) :
        enter  GoAhead  | Z(s), W(s) | ;
        END CASE

        LOOP END


GoAhead | Z(s), W(s) | :

        BEGIN
        initiate transmission of m on DC's ;
        IF no collision on DC's THEN
        BEGIN
        complete transmission on DC's ;
        stop jamming CC (transmission is successful)
        END
        ELSE
        BEGIN (DC's collision resolution)
        IF Z(s) = TRUE THEN defer by a ;
        (corresponds to either a regular message or an emergency message)
        sense DC's ;
        WHEN DC's idle DO
              BEGIN
              transmit m entirely on DC's ;
              (collision resolved)
              stop jamming CC
              END
        END
        END
```

(ii) Comments

Listen Before Talking and Listen While Talking are required on all channels. The occurence of a collision on the Control Channel determines the start of an "epoch". A message that must be transmitted is at most delayed by one epoch. The duration of an epoch depends upon the number of messages involved in a collision (i.e. on the number of messages submitted within a window of size 2p) and upon the number of emergency messages which can enter an epoch. Rank of transmission within an epoch is determined by the index currently assigned to the transmitting station.

An upper bound for access delays exists. Its value can be computed for many different situations, e.g. epochs including 0, 1, 2, ... emergency messages. Remember that occurence of emergency conditions is considered to be "exceptional".

When a station goes down, it is assumed to become silent. In particular, jamming of the Control Channel (if being performed at the time of the crash) ceases. A collision on DC's can occur between one regular message and one emergency message. When such a collision occurs, priority is given to the emergency message. When two emergency messages collide on DC's, priority for transmission is determined so as to minimize deference delays.

(iii) Index assignment

Static indexes (e.g. station names) could be used. If the degree of "unfairness" introduced is felt unacceptable, one can "distribute unfairness" by rotating indexes among stations. One possibility consists in taking advantage of the broadcast nature of the channels. Message headers will then contain the name of the station, randomly chosen on $|1,S|$ by the transmitting station, which will be assigned index 1 for the next epoch. It is straightforward for every station to deduce from this which index it should be using during the next epoch.

The updating of index assignment takes place redundantly on n-1 data channels, thus reducing the probability that stations will not be identically updated.

Another possibility, which is more immune to noise, consists in rotating indexes by one position, in some predetermined fashion, each time the Control Channel is jammed.

(iv) Choice of a Control Channel

Sophisticated election algorithms cannot be used in general by stations for determining which of the n channels is to be used as the Control Channel, for election algorithms require message passing. A simple way of dynamically identifying the Control Channel when a change occurs (e.g. start of the system, crash of the channel previously used as the Control Channel) consists in selecting arbitrarily the up channel that has a specific property (e.g. smallest/highest name, name closest to pi, etc.).

When n-1 channels are simultaneously down, the remaining channel will then be operated under a conventional CSMA-CD protocol. Determinism is lost only in that case.

5. CONCLUSION

Both protocols are deterministic in the sense that all messages which are submitted are eventually transmitted over the channels. It is only because we believe it necessary to exercise promptness control that those messages which have missed their deadlines are not transmitted at all. Even when message "losses" occur, LAN's which use one of these protocols behave deterministically for it is possible to tell, for a given state, which messages are transmitted and which are not.

The behaviour of such LAN's would then be very different from the behaviour of any non real-time LAN that would not transmit messages either because collisions happen too frequently or because the token has not shown up for too long, which might happen with static priority oriented TP protocols. Just because promptness control mechanisms are generally absent in non real-time LAN's, it is not too meaningful to compare message "losses" as they occur in such LAN's and as they occur in the real-time LAN's we consider.

Both protocols have interesting flexibility properties, most of them being inherited from the CSMA philosophy. For example, when a station is disconnected, no deadlock or starvation can occur. This is not the case for some of the proposals which have been published in the past (see | CAP79 | for example).

Similarly, it is sufficient for a station which is being connected to wait until a message is received so as to read the value of current physical time and infer at which time transmission of its first message can be attempted. Provision should be made for avoiding infinite waiting when the network is initialized or reinitialized as well as exceedingly large waiting times when the network is extremely lightly loaded. Solutions are straightforward.

The next major tasks to be conducted in relation with these deterministic multiple access protocols are (i) to prove their correctness and (ii) to evaluate their performances. These tasks are in progress within projet SCORE at INRIA and at the Computer Science Laboratory of University of Michigan, Ann Arbor.

Important design and modeling issues are detailed in a forthcoming joint research report.

Acknowledgments

I want to thank Janine BOUDENANT and Pierre ROLIN for the many fruitful discussions we had on the issue of deterministic multiple access protocols.

REFERENCES

| CAP79 | J.I. Capetanakis, "Generalized TDMA : the multi-acccessing tree protocol", IEEE Trans. on Communications, Com-27, Oct. 1979, 1476-1484.

| CHL80 | I. Chlamtac, W. Franta, "Message-based priority access to local networks", Computer Communications, vol.3, 2, April 1980.

| GOL83 | Y.I. Gold, W.R. Franta, "An efficient collision-free protocol for prioritized access-control of cable or radio-channels", Computer Networks 7, 1983, (North-Holland), 83-98.

| LEL83 | G. Le Lann, "On real-time distributed computing", IFIP Congress 1983, (North-Holland pub.), 741-753.

| MOL81 | M.L. Molle, "Unifications and extensions of the multiple-access communications problem", UCLA Report n° CSD-810730, (Ph. D. Thesis), July 1981, 139 p.

| POW81 | D.R. Powell, "Performance evaluation and comparison of dependable channel access techniques for locally-distributed computing systems", Proc. 2 nd. ICDCS, April 1981, 256-270.

| ROM81 | R. Rom, F.A. Tobagi, "Message-based priority functions in local multiaccess communication systems", Computer Networks, 1981, (North-Holland), 273-286.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique