

A New Real-time Network protocol-Node Order Protocol

Li chanjuan, Nicholas Mc Guire, Zhou Qingguo

Distributed and Embedded System Lab

School of Information Science and Engineering

Lanzhou University

Tianshui South Road 222, Lanzhou, P.R.China

lichanjuan04@gmail.com, mcguire@lzu.edu.cn, zhouqg@lzu.edu.cn

Abstract

It is natural for real-time network protocols to start at the top level with the temporal behavior, so do many of the existing protocols. However, time-triggered protocols not only add extra complexity because related to maintenance of global time but also incur a penalty in network utilisation due to temporal padding. To relieve these issues, we introduce a node ordering concept in which a strict node order is defined and the order of communication events is established by this pre-configured order instead of global time. Thus what is required is maintaining the node order only. Based on such a concept, we propose Node Ordering Protocol (NOP) in which medium access is controlled by strict node ordering. In NOP, there is no concept of global time and communication activities are event-triggered. So NOP eliminates the interval reserved for keeping temporal order and can saturate the network in the general case.

In this paper, we first describe communication semantics of NOP. Then the fault hypothesis is analysed and error detection and handling is introduced in detail. Finally, we supplement consideration in practical implementation of NOP.

1 Introduction

In distributed embedded real-time system, Time Division Multiple Access (TDMA) is a widely used method to access shared medium. In TDMA, the point in time when a node obtains the right to transmit a frame is determined by the progression of real time. This requires that a global time-base is available at all nodes. In a TDMA-based system, the total channel capacity is statically divided into fixed number of slots and unique sending slots is assigned to every node. Thus the protocols based on TDMA have the obvious disadvantage that the slot is not used if a node does not want to transmit a message. This property of TDMA is called temporal padding. Moreover, the intrinsic limitation of global time [1] must be considered, when establishing total order of events in a time-triggered protocol. The determination of the occurrence of the communication event has to be based on sparse time base, instead of dense real-time base. Therefore the network has to be idle

between two active intervals[2].

Finally, the physical clock has negative effect on the precision of the global time due to its physical properties (e.g. sensitiveness to the environment, aging, drift...). Thus communication protocols relying on a well synchronized clock encounter heavy performance penalties and unexpected errors caused by clock synchronization failures[3].

To get rid of these issues, we propose a new protocol, the Node Order Protocol (NOP), which determines the order of accessing nodes to the shared medium by specifying the node order a priori (statically configured order in the simplest case). This results in a communication protocol which is no longer time-triggered, but event triggered. The advantages of NOP are, that the global time is removed from the protocol layer, and thus the temporal padding is eliminated, further failures in time synchronisation do not necessarily lead to a node or system failure. This leads to a tremendous improvement of the network utilisation and also simplifies the protocol de-

mands. However, global time is very important for many applications of distributed systems. Since we eliminated the global time from the underlying communication protocol, we relocate it and put it on top of the communication protocol. This is the contrary approach of most existing protocols as shown in Figure 1. In this paper, we will introduce the concept behind NOP and its communication semantics. After that we analyze the reliability of NOP by stating error detection and containment in detail.

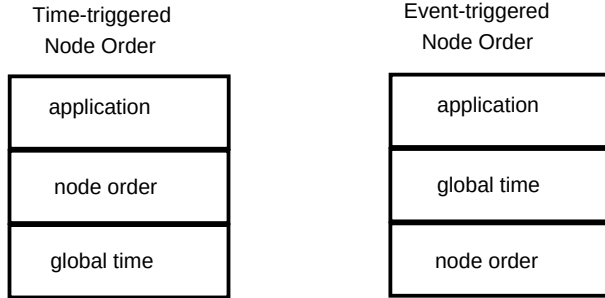


FIGURE 1: *Logical Layer of TTP and NOP*

2 Related work

There are other protocols with the goal to improve the utility of bandwidth in TDMA. One example are TDMA protocols with slot skipping (TDMA/SS)[4], which try to identify unused slots and skip them. TDMA/SS uses an `access_counter` which is set, if after the start of a slot a period of time has passed, and no message has been sent. Now the unused slot can be skipped and the next node is allowed to send his message earlier. TDMA/SS effectively avoids wasted slot in TDMA, but it is still a time-triggered protocol and depends on a global time base. Further this reclamation mechanism has a fairly high latency - which reduces the effectiveness of this measure. POWERLINK [5] proposes another solution to this issue. POWERLINK maintains fixed length cycle. Instead of dividing the cycle into slots, it uses a center node to assign bandwidth in a request/response way. The node can access the medium only, when the center node is polling it. Time slot is based on communication events instead of fixed length. In addition, POWERLINK allows several nodes multiplex a time slot if the period of message is longer than POWERLINK cycle which prevents from wasting network bandwidth. Though POWERLINK integrates event-driven model to improve utility of network, it also relies on well synchronized clocks.

3 Node Ordering Concept

The objective of global time in communication protocols is to maintain the temporal order of occurrences of events and construct a causal order of event chains. While most available communication protocols try to achieve this objective, with a global time base which can be set into relation with real-time, we suggest a solution, which (in contrast to the Lamport algorithm[6]), is based on a node ordering concept, in which all nodes follow a strict order to issue communication events. Based on this ordering, the temporal order of events can be derived and form a totally ordered set. From this totally ordered set, even causal order can be reconstructed. From the perspective of node ordering, network communication protocols in distributed embedded systems can be classified into two categories:

- ordering protocols
- non ordering protocols

TCP/IP, UDP/IP and CAN are obviously non ordering protocols, because the nodes transmit in a random order. Profibus seems to be a node ordering protocol, but the ordering can be reset during run-time. Examples for ordering protocols are TTP and TDM, since TDM provides a round-robin media arbitration, while communication based on a message delivery list (MEDL) is used in TTP [7]. The problem of ordering and non-ordering protocols is in principle not related to the temporal behavior of the connections, though in the two mentioned ordering protocols, the ordering is bounded to global time. We claim that the node order is the main objective, and that temporal order is one way to achieve the goal. In NOP, it is necessary for each node to monitor the correctness of ordering during runtime, in order to detect a potentially faulty node. Hence, multicast or broadcast is required in the LAN.

4 Design of NOP

Though there are many protocols around, there is none designed from the start point of node ordering. The NOP protocol we propose in this paper is designed entirely around the node ordering concept without temporal ordering at all. In the following section, the design details will be introduced and the potential advantage of the protocol will be illustrated.

4.1 Basic Communication Semantics

For NOP to work, the minimum requirement is that all nodes agree on a sending order. In principle, this is a sufficient criteria for NOP. In a practical environment, at least one further requirement has to be added: a timeout for the receipt of packets has to be introduced. Timeouts in NOP need not to be based on a global time, as long as the drift of all clocks is within an upper bound (which can be monitored by exchanging time stamps). The implementation details of timeout will be introduced in section 5.2.

Thus the configuration parameters of NOP are a MEDL, which assigns an order of the nodes and a TIMEOUT value (the maximum waiting time for a packet). Based on these two preassigned parameters, we begin to introduce the basic communication semantics of NOP.

- Message receive verification
 - Keep on listening to the network for at most TIMEOUT interval.
 - If a frame is received before the end of TIMEOUT, the receiver performs a series of fault tests to the received frame. If the frame passes the verification, then mark the state of NOP to NOP_OPERATION, else set it to NOP_SUSPEND.
- Message sending
 - If the NOP state is NOP_OPERATION, a message in message buffer is sent out. If no message to send, KEEPALIVE is copied to message buffer and is transmitted.
 - When the state is NOP_SUSPEND(timeout occurs in this node), a timeout message is sent.

As we mentioned above, there is no global time in NOP, all communication activities are event-driven. The big advantage of NOP is, that it keeps the network saturated all the time, as long as no error occurs. Figure 2 shows a comparison of the medium access assignment in TDMA and NOP. As you can see, the length of each slot is decided by the length of the transmitted frame in NOP, the only limitation is the MTU of underlying network. If the network is in light load, only KEEPALIVE messages are transmitted and the length of cycle is quite shorter compared to TDMA. Although there is no fixed cycle in NOP, the end-to-end delay of a message is bounded by the

number of node and the maximum frame size (MTU - Maximum Transmission Unit).

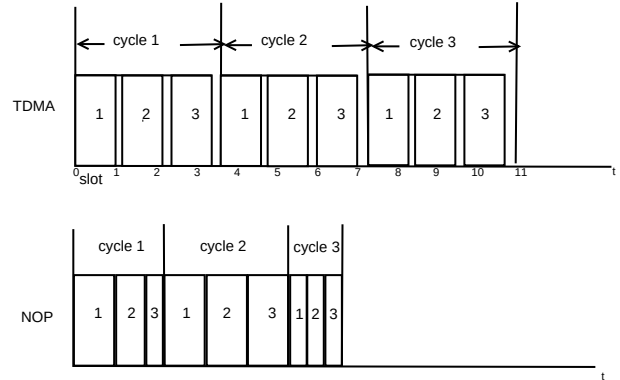


FIGURE 2: *Comparison of TDMA with NOP*

4.2 Failure Mode in NOP

So far, we have only considered the error free case. Error detection and handling have not been included in the communication semantics presented so far. The simple setup presented above, leads to a number of potential faults. In this section we will describe what strategies NOP uses, to make the protocol robust. The fault hypothesis of NOP is defined as follows:

- omission failures
 - at sender side
 - at receiver side
- crash failure
 - sender node crashes
 - receiver node crashes
- clock performance failure
 - timeout too early
 - timeout too late
- receipt of out-of-order frames
- babbling idiot failure

The failures produced by the physical medium are not list above, but they will be considered during the discussion on omission failures.

In NOP, there are basically only two possible error cases:

- a timeout occurs, while waiting for a frames
- an out of order frame is received

these are the erroneous behaviour, which can detect all the failures listed above.

4.3 Error Detection and Containment in NOP

In this section, we will discuss the error detection mechanism in detail. At the same time, a recovery strategy for omission failures is introduced.

Obviously, the node failure and loss of frame at sender side will result in a timeout event, and is detected by other nodes by strict node ordering. In most case, however, clock failure in NOP can be tolerated and has no negative effect on NOP, because communication activities in NOP is event-triggered. Only when the failed clock issues incorrect timeout events, it can be identified. I will take clock failure as an example to illustrate the mechanism of NOP to detect the failure. The process is shown in figure 3. In early timeout case, C timeouts before A's message arrival. Thus C will assume A fails. Until it receives the frame from A, it knows that it timeouts too early and exits quietly because it is not its turn to access medium. So, A and B will timeout when they wait for C's message and identify C failed.

From the figure 3, we can see, that the error detection mechanism in NOP is simple and easy to implement. Each node only has to maintain the state informations (NOP state and expected sender number), which is easy accessible.

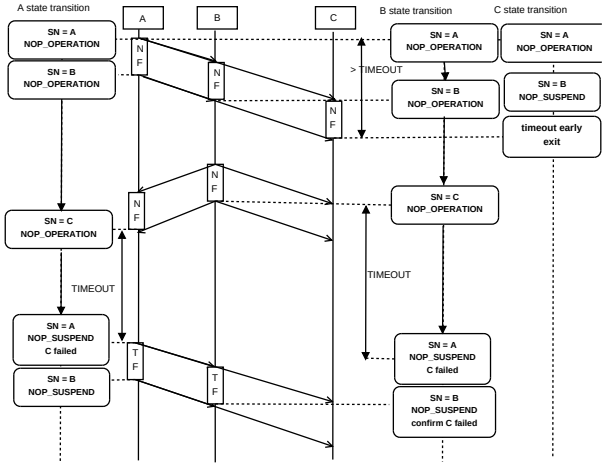


FIGURE 3: *Early Timeout Failure*

It is fairly simple to detect and handle the out-of-order frame by ignoring it and going on waiting for a correct frame. If a node is sending out messages which conflict with the MEDL (or the MEDL at the sending side is corrupted), then this will be detected by TIMEOUT handler (that is the expected message from the node does not arrive).

In NOP, there is no bus guard to prevent an babbling idiot node from monopolizing the network as TTP implements it in hardware. However, babbling

idiot node will issue out-of-order message. Thus this failure can be detected by adding the out-of-order counter on a per node basis at receiver side. If the node clogs the network, the counter will exceed the threshold value and triggers an action to shut down the network. Otherwise, the timeout condition will be hit and the node will be removed from the alive node list.

Errors in a NOP based LAN are basically system level errors. It is not necessary to provide membership protocols at this point. In addition, it would not make much sense to do this at the protocol level as it is very application domain dependent to react adequate to the loss of communication.

Considering that a lost frame can render a network unusable, the availability of the protocol is not very satisfying. So a simple recovery strategy is proposed to tolerate the loss of a frame at receiver side. To recover the lost frame, retransmission is obviously required. The only usable way to do this is that every node retransmits the Nth-1 message along with its current message. This allows recovery without an extra delay of communication traffic in theory. Though this strategy may half the effective bandwidth available in the system, it more or less eliminates the ability of transient errors. However, this strategy is not able to trivially recover from dual omission faults.

5 Implementation Consideration

5.1 Physical Layer

NOP itself does not depend on a specific physical layer. However, for we decided to base NOP on a 100BASE-X Ethernet for our practical implementation. To reduce the transmission jitter and delay, all nodes in the NOP network are connected through hubs instead of switches.

5.2 Timeout Implementation

The TIMEOUT interval is an important parameter in NOP. As mentioned above, timeout management is simply provided if the clock drifts are known. In practice, this can not be assumed. We considered two possible strategies to tackle this problem. One is, to assume a reasonable boundary on clock drift for example 10^{-4} for a quartz crystal. During run time the drift is measured and a violation of the assumption is detected. This strategy is easy to implement

and has no effect on the communication semantics. Obviously, the maximum drift will impact the performance and availability of NOP. Another way is to continuously measure the maximum clock drift of the nodes and adjust the timeout value. To implement this strategy, it is necessary to extend the communication semantics to include a time stamp difference of each node in its message header, thus allowing all nodes to monitor the clock drift and correct the maximum drift in the system. This strategy does not work well at present and needs further investigation.

6 Conclusions

In this paper, we proposed a new event-triggered protocol called NOP, which is based on a node ordering concept. The paper points out the main advantages of such a protocol. To sum it up, these advantages are:

- NOP is based on strict node ordering concept, thus removing dependence on global time at the protocol level. This makes the protocol fairly simple and easy to validate. At the same time, the network utilization is more economic than a TDMA based protocol. Moreover, the TIMEOUT mechanism makes sure that the protocol is robust to the impact of the drift effect of the physical clocks (i.e. no byzantine clock problem).
- Communication activities in NOP are event-triggered and the waste of bandwidth caused by the temporal padding in time-triggered protocols is eliminated while retaining the advantage of a handshake-free protocol like TTP.
- Error detection in NOP is easy to implement and requires very limited state information, which also implies that the implementation in temporal and spatial redundancy is fairly easy in NOP.
- Time of error detection is within a well defined time window bounded by the TIMEOUT value.

However, strict node ordering is not a general requirement of distributed systems. So NOP can solve and relieve the issues of a particular problem domain, most notably those that are now using TDMA or TTP, but it does not provide a general solution for communication. The target domain for NOP currently is tightly coupled systems in the domain of safety related systems (i.e. TMR).

7 Acknowledgment

Special thanks to Nicholas Mc Guire, my supervisor who gives me a such interesting topic and provides plenty of important instructions. Thanks to DSLab for offering the necessary resources and appropriate environment to do my research work.

References

- [1] *Real-Time Systems Design Principles for Distributed Embedded Applications*, Hermann Kopetz, 1997.
- [2] *Sparse Time versus Dense Time in Distributed Real-Time systems*, H.Kopetz, 12th Int. Conf. on Distributed Computing Systems, 1992
- [3] *The Effect of Large Clock Drift on Performance of Event and Time Triggered Network Interfaces*, Dawid Trawczynski, Januaz Sosnowski, Junusz Zalewski, 2007, DepCoS-RELCOMEX'07
- [4] *Analyzing TDMA with Slot Skipping*, B.Andersson, E.Tovar, N.Pereira, 2005, RTSS'05
- [5] *Ethernet POWERLINK Communication Profile Specification version 1.1.0*
- [6] *Time, Clocks and the Ordering of Events in a Distributed System*, Leslie Lamport, 1978, Communication of the ACM.
- [7] *TTP - A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems*, H.Kopetz, G. Grunsteidl, 1993