Pergamon

Control Eng. Practice, Vol. 3, No. 8, pp. 1171-1176, 1995
Copyright © 1995 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0967-0661/95 $9.50 + 0.00

0967-0661(95)00113-1

COMMUNICATION ARCHITECTURES FOR DISTRIBUTED COMPUTER CONTROL SYSTEMS

W. Dieterle*, H.-D. Kochs* and E. Dittmar**

**Department of Computer Science, University of Duisburg, Lotharstr. 1, 47048 Duisburg, Germany*
***ABB Netzleittechnik Gmbh, Network Control and Protection, Wallstadter Str. 53-59, 68259 Ladenburg, Germany*

(Received October 1994; in final form May 1995)

Abstract: The use of distributed computer control systems (DCCS) demands high reliability, adequate real-time behaviour and increasingly economical systems. The last demand requires the use of cheap standard components, whenever possible. The following paper discusses the realization of DCCS with respect to these constraints. Problems due to the conventional use of standardized communication protocols in distributed control systems in general, and highly-reliable systems in particular, are shown. Multicast communication concepts are presented as solutions, using standardized protocols in a problem-specific way. The presented concepts fulfill the necessity of using standard components, as well as the specific demands of DCCS.

Keywords: Control Systems, Local Area Networks, Computer Communication Networks, Distributed Databases, Communication Protocols, Standard Protocols.

1. INTRODUCTION

High demands are placed on distributed computer control systems (DCCS), used in energy distribution, production or process engineering, whereby the costs aspect is more and more dominant. Cost minimization makes the use of cheap, standardized components and the design of simple, modular system concepts mandatory. In the following, specific architectural features for system communication, types of data storage and fault-tolerance in DCCS are derived from the system requirements described. Existing standard communication protocols, e.g. TCP/IP, UDP/IP or ISO/OSI, are not intended to support these features; however, the lack of appropriate standard protocols in the UNIX environment requires the use of the existing ones. Problems with the conventional use of standard communication protocols are shown, and two multicast concepts are presented and evaluated. They are based on standardized communication protocols, but use them in a problem-specific manner. The multicast concepts are very simple (in comparison with existing solutions) and have a very low message overhead, close to the minimal message cost which is determined by simplified border conditions. Experimental results show that the timing characteristics of the first solution (ring multicast) are acceptable for small and medium-sized DCCS. The second solution (datagram multicast) is suitable for large DCCS and systems with specific demands for data transfer time and throughput.

2. BASIC ARCHITECTURE OF DCCS

Modern industrial computer control systems are designed as distributed systems (Fig. 1). The systems considered here consist of approx. 10-15 functional computers, to which the functional modules described below can be randomly associated. Functional computers are connected via Local Area Networks (LAN), typically Ethernet. To meet the high demands for reliability, computers with important functions are redundantly structured. The functional scope of such systems incorporates data acquisition, basic processing of process data (SCADA), and process visualization (MMI), as well as additional functional modules (complex secondary functions) depending on the con-

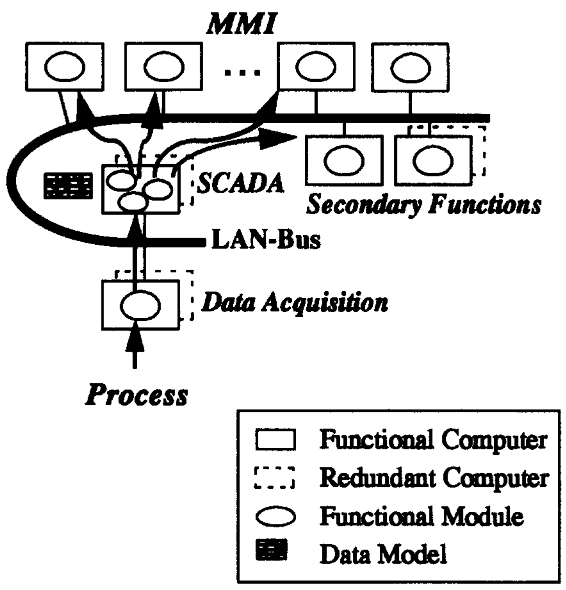

Fig. 1: System Architecture of Distributed Computer Control Systems

crete application purpose. Due to the distribution and redundancy of functions, complex data flows are present in the system. Information flow from the process to the MMI dominates (only this type of data flow is shown in Fig. 1). A technological description of the process and control system is held in static data models. The process state is kept in dynamic data models (up to 200000 process variables). MMI images are held in further data models; they comprise a static component, the image structure, and a dynamic section, the actual process state. The following discussion is only concerned with dynamic data models, which are to be continuously updated.

3. REQUIREMENTS AND EFFECTS ON SYSTEM ARCHITECTURE

3.1 System Requirements

The requirements placed on distributed computer control systems can be divided into *low costs, real-time behaviour* and *reliability/fault tolerance*. More and more, the system costs and follow-up costs are proving to be the most important factors. *Low costs* demand the use of standard components as much as possible, openness of the systems (in the sense of simple expandability and testability), modularity, and simple system concepts, as well as independence from a particular manufacturer. The use of standards concerns hardware (Workstation, PC), operating system (UNIX), and visualization (X-Windows, OSF/MOTIF) as well as the system communication (LAN: Ethernet, protocol: TCP/IP, UDP/IP, ISO/OSI). Components available on the market are integrated to a system and expanded by non-present features at the module interfaces (e.g. fault tolerance). The following paragraphs are mainly concerned with the last-mentioned aspect, the communication system. The required *real-time behaviour* is characterized by short response times, high system throughput, random access to all process data within very short time, permanent actualization of data models and information output at the MMI interface, good system dynamics even under heavy load (e.g., process failure), fast failure recognition and reconfiguration in the case of the failure of redundant components. Due to the effects of centralization in the direction of the higher control levels and the consequences of component failures, high *reliability* by means of structural redundancy and *fault tolerance* is required for industrial computer control systems. For computers with important functions and very high reliability demands, the LAN bus has also to be redundant (Kochs *et al.*, 1993). Redundancy of the computers is realized according to the leader/follower principle, the computers exhibit fail-silent behaviour (Powell, 1991; Kopetz, 1989).

3.2 Architectural Necessities

The requirements to a high degree determine the conceptual features of a system, especially system communication and type of data storage. **Figure 2** shows the requirements and their effects on system architecture. Data acquisition and secondary functions are omitted for the sake of simplicity.

Modern DCCS are based on UNIX workstations; thus it would be desirable to use Client/Server communication, typical of UNIX environments (Fig. 2*a*). Yet "pure" client-server architectures with centralized data storage are not appropriate for DCCS. Continuous actualization of dynamic data models would require cyclical processing of the whole process state. This is not possible - not even with the presently available very powerful computer and communication technology. Event-driven information transfer is necessary: *Producer/Consumer communication*. The data models of MMI images are kept in each MMI computer: *decentralization of dynamic data models*. Furthermore, expanded MMI functionalities (e.g. Zooming, Scrolling) require that the decentralized data models comprise the total process state (Fig. 2*b*).

The most important criterion for distributed systems with decentralized databases is data consistency. In the case of fault-free operation, it is trivial to ensure data consistency. Yet it becomes a problem when failures occur. Process description takes place by means of process alterations (events) on the basis of a consistent original state of each data model. Disruptions lead to faulty and inconsistent data models (a process signal once lost, is lost forever). This demands solutions for the retrieval of information by the transfer of complete data sets, or the avoidance of inconsistencies by means of sophisticated approaches. Time-costly retrieval of information in cases of computer or LAN-bus failure is not practical when using modern MMI images, comprising the total process state. This means *"seamless" reconfiguration* is necessary to maintain data consistency, i.e. immediate reconfiguration without loss, duplication or ordering impairment of information (Fig. 2c).

3.3 Existing Solutions

The problem faced is the consistent update of distributed databases in the presence of failures (interactive consistency), e.g. (Alford *et al.*, 1985). Consistency is expensive in terms of time and messages. Existing commercial solutions are based on centralized structures, and are thus not appropriate in the application area considered. There exist a number of theoretical/experimental solutions for consistency in distributed systems in the presence of failures, which are generally based on so-called "agreement protocols". These concepts can be classified as synchronous and asynchronous. Synchronous solutions (Kopetz, 1989; Christian, 1990) are based on synchronized clocks. They require space redundancy, i.e. message transmission over several channels, which increases the computer load (context switches). Thus, they are not appropriate for the systems considered here. Existing asynchronous solutions (Birman and Joseph, 1987; Özalp, 1990) are targeted at systems

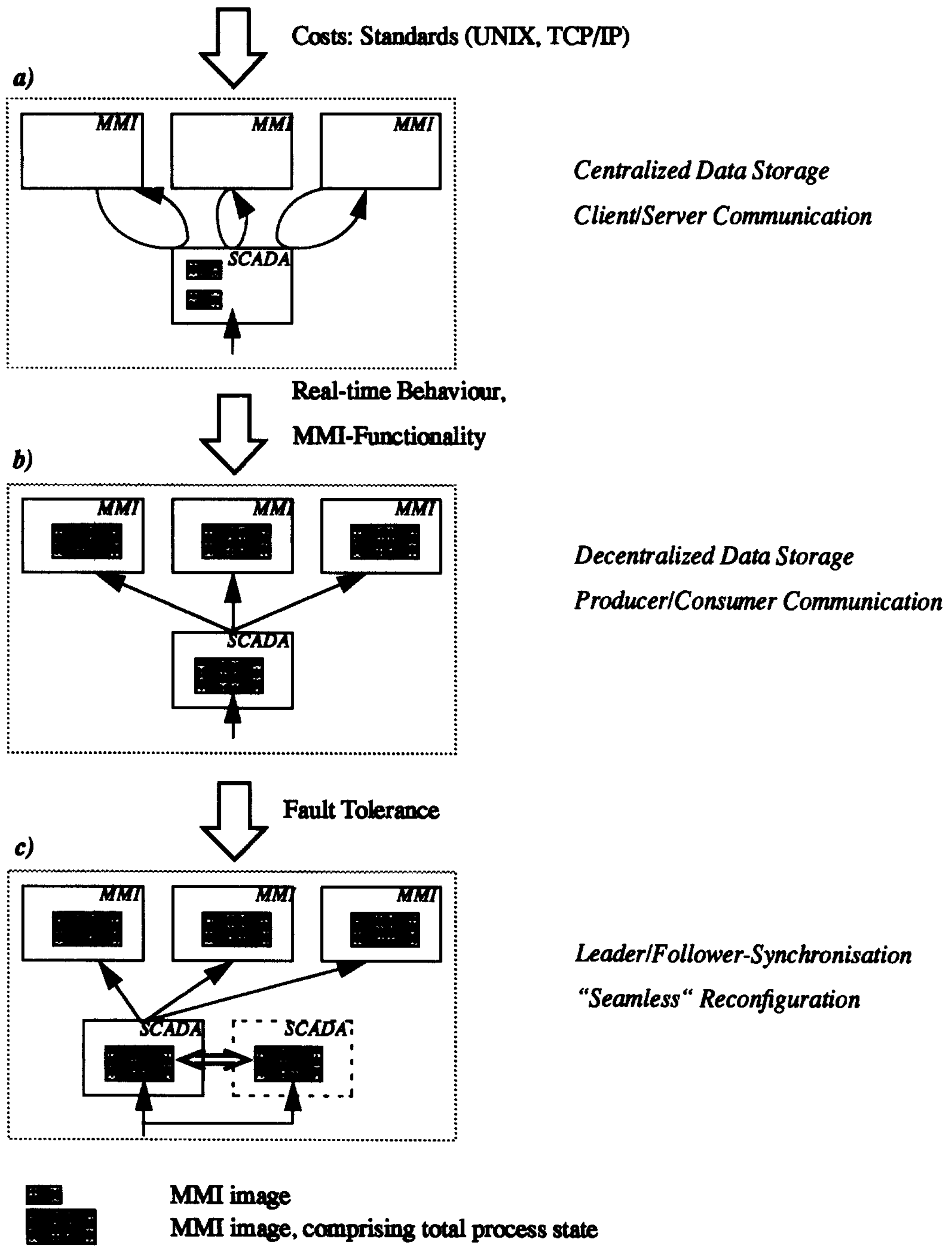
Costs: Standards (UNIX, TCP/IP)
a)
MMI
MMI
MMI
SCADA
Centralized Data Storage
Client/Server Communication
Real-time Behaviour,
MMI-Functionality
b)
MMI
MMI
MMI
SCADA
Decentralized Data Storage
Producer/Consumer Communication
Fault Tolerance
c)
MMI
MMI
MMI
SCADA
SCADA
Leader/Follower-Synchronisation
"Seamless" Reconfiguration
MMI image
MMI image, comprising total process state

Fig. 2: Architectural Features of DCCS

beyond the scope of DCCS, and are thus too costly. Communication solutions for DCCS are shown in (Powell, 1991); however, the concepts are complex and expensive with regard to communication and processing load, and do not meet the targeted system philosophy of expanding existing market components by non-present features. None of the existing concepts fits as an appropriate communication architecture for the DCCS considered. As a consequence, several communication concepts were developed, and will be discussed in the following.

4. STANDARDIZED PROTOCOLS: CONVENTIONAL USE

Solutions have been developed with a particular emphasis on simple (and thus cheap) concepts, modular system architectures with the use of standardized protocols, and comparatively low communication overhead. A specific problem when using UNIX operating systems is that all existing standardized protocols in the UNIX environment (TCP/IP, UDP/IP, ISO/OSI) are dedicated to Client/Server communication with centralized data storage, i.e. an appropriate use of the protocols to implement Producer/Consumer communication is required. A first solution would be the conventional use of standardized communication protocols. This involves the implementation of connections between distributed processes according to an application-specific structure (point-to-point-structure). Conventional use of standardized communication protocols evokes a number of problems to be discussed below. The statements apply to the TCP/IP protocol (Comer and Stevens, 1991), and in similar fashion to ISO/OSI protocols.

Conventional use of standardized communication protocols demands a high level of linking between the computers for data transfer and failure recognition. The latter requires fast and consistent recognition of component failures by all the participants, e.g. (Christian, 1988). Failure recognition takes place by means of connection timeout. Transfer of single process events would be expensive (bus load, context switches); a combined time-driven/amount-driven transfer of process data is required. Connection-oriented protocols support unicast communication only, i.e. messages have to be sent several times, and this is even worse if the sender is redundantly configured: each connection has to be synchronized separately between Leader and Follower. These aspects lead to a high work load for the LAN and computers, in particular for the redundantly configured SCADA computer as the logical centre of the system.

In distributed systems, the problem of causal and total order of transferred and processed data exists, e.g. one has to prevent original data being processed after data derived from the original data. This demands sophisticated measures to ensure causal and/or total order if protocols are conventionally used (Lamport, 1978; Powell, 1991).

The system structure is parametered or even programmed into the communication software (semantics: "send message *to*", "receive message *from*"). Alterations or extensions of the system structure are complex and expensive. Besides (de-)coupling the system via the LAN bus on the hardware side, it is also necessary to detach the computers with their communication protocols on the software side. Failure of components leads to undesired communication feedback due to protocol dependencies. This feedback must be controlled by the sender and the receiver software. Receiver acknowledgement of the TCP/IP protocol cannot be evaluated by sender applications. This means the temporal sequence of data transfer cannot be exactly controlled, and thus leads to possible inconsistencies in cases of failure, which can only be remedied via additional mechanisms. TCP/IP parametering for retransmission and connection timeout due to component failure with the aim of reducing fault latency is limited and not according to the standard (Comer and Lin, 1993).

Further problems concern the necessity of additional buffering of sender data at the application level for the prevention of data loss in cases of connection timeout.

5. MULTICAST CONCEPTS FOR DCCS

Due to the problems with the conventional use of standardized protocols two multicast concepts, based on the UDP/IP protocol, have been developed. The concepts use standardized protocols in a problem-specific manner. UDP/IP constitutes the unconfirmed, non-connected alternative to TCP/IP. The following objectives were aimed at during the development of these concepts:

- Realization of simple concepts.
- Use of standardized communication protocols (thereby preventing the problems mentioned).
- Minimization of dependencies or feedback by the protocols.
- Effective utilization of LAN bus and computers.
- Equal distribution of computer workload due to communication on all components.
- Simple and efficient mechanisms for failure recognition and reconfiguration of computers and bus.
- Simple monitoring and test interfaces.

The procedure described first (ring multicast) is based on ring-configured information transfer within the system, whereby a logical multicast is realized.

The second concept uses the physical multicast mechanism of the datagram-oriented UDP/IP protocol (datagram multicast).

5.1 Ring Multicast Protocol

In the case of the ring multicast concept (Fig. 3), data exchange takes place via a circulating token of variable length. Stations willing to send wait for the token (1) and enter their data upon receipt of the token (2). During the following token circulation (3) the data pass all (potential) receivers. Each station holding the token selects information and adds its own data to the token (4). After a full token cycle data are removed from the token by the sender (5), and new data are entered. Besides the advantages of the concept - discussed later - the protocol could have one possible drawback, when used in large DCCS, comprising a high number of components (more than 10 computers). Due to the circulation of information, unacceptable token rotation times could occur under heavy load. Thus, for large DCCS a second protocol was developed with the aim of a more efficient use of the LAN bus for data transfer and the reduction of token rotation times.

5.2 Datagram Multicast Protocol

In the case of datagram multicast, data exchange takes place by means of physical broad-/multicast using the datagram mechanism of the UDP/IP protocol. Modern operating systems enable multicast transfers, whereby data selection is supported by the receiver hardware. Datagram transfer of the UDP/IP protocol is executed without acknowledgement. For the realization of confirmed transfer as well as for monitoring failures of system components, an acknowledgement ring is installed between the individual communication participants. Datagram transfer in broad-/multicast and token exchange are executed completely asynchronously. On the subsequent receipt of the token, the sender of a datagram enters the datagram sequence number into the token. Use of an additional global sequence number for the datagrams ensures an unambiguous datagram order. If a station does not receive transmitted data (recognizable to the receiver by means of the acknowledgement token) a negative acknowledgement is entered into the token; transfer is repeated by the sender.

5.3 Protocol Properties

The protocol concepts presented above offer the following advantages in comparison to conventionally employed standardized protocols:

- Distribution of events and decentralization of data are implicitly supported by the concept. Consistency-maintaining reconfiguration is easy to realize by means of the multicast concepts.
- Information selection at the receiver, dependencies in cases of changes or failures are minimized (semantics: "send message", "receive message *of type*")

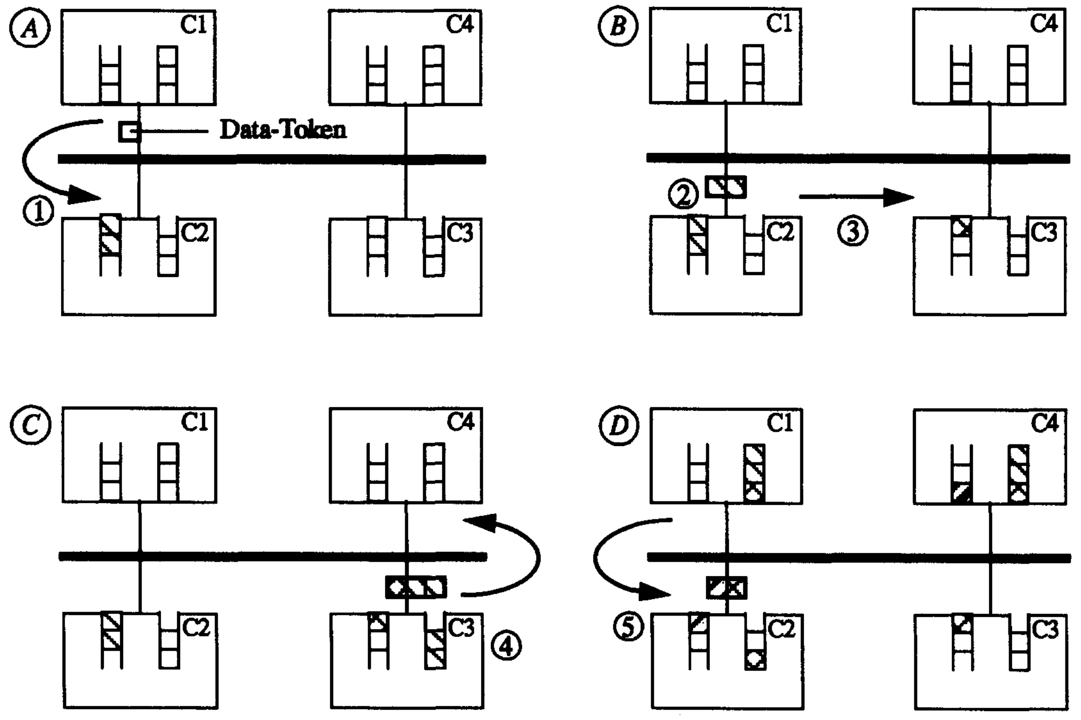

Fig. 3: Ring Multicast Protocol

- The communication workload for bus and computers is decreased (datagram multicast: hardware support for information selection, one data transfer for reliable information transfer to all receivers; ring multicast: collision-free data transfer).
- Causal and total order is ensured by the protocol.
- A simple concept for the consistent monitoring of all computer components (membership-service) without additional workload and information transfer.

5.4 Protocol Evaluation

Some theoretical and experimental results on protocol cost will be presented, assuming that messages are sent to all components in the system (this assumption is justified in modern DCCS, where each component holds a dynamic data model). Due to the implicit acknowledgement scheme of ring concepts, information transfer takes place without explicit acknowledgement. This leads to a very low message cost of the protocols (n messages for one piece of information and n-1 recipients). The use of physical multicasts (datagram multicast) brings further improvements for bus and computer loads. Comparison: data consisteny is comparatively trivial to ensure if the source of information is highly reliable and no communication failures occur (only message loss, caused by transient errors has to be managed by acknowledgements). Even in this simplified case information transfer requires n-1 messages, plus further acknowledgements and additional mechanisms for message ordering and consistent system view.

Experimental evaluation of ring multicast yielded token rotation times of 300-400ms (8 ring participants, token length 30kByte, OS: Solaris 2.2). Datagram multicast yielded token rotation times of about 200ms, due to the reduced token length.

6. CONCLUSION

Modern DCCS are based on decentralized data storage, Producer/Consumer communication and “seamless“ reconfiguration in the presence of failures of important components. Due to cost constraints, the use of standardized communication protocols is mandatory, despite the fact that standardized communication protocols in the UNIX environment are dedicated to Client/Server communication. Problems are shown concerning the realization of DCCS in general and particularly for highly reliable systems under the conventional use of standardized communication protocols. Two multicast concepts are presented as solutions. These concepts are based on standardized protocols: use of the protocols is adapted to the individual problems of distributed computer control systems. The advantages of the presented solutions in comparison to conventional use of standardized communication protocols are demonstrated. The protocols enable standardized protocols to be used, avoiding the above-mentioned problems.

The paper is the result of a R&D project in common with ABB Network Control and Protection.

REFERENCES

Alford M. W., J. P. Ansart, G. Hommel, L. Lamport, B. Liskov, G. P. Mullery and F. B. Schneider (1985). *Distributed Systems, Methods and Tools for Specification.* Springer Verlag, Berlin.

Birman K. P. and T. A. Joseph (1987). Reliable Communication in the Presence of Failures. *ACM Transactions on Computer Systems*, Vol. 5, 1, pp. 47-76.

Christian F. (1988). Agreeing on who is present and who is absent in a synchronous distributed system. *18th International Symposiun on Fault-Tolerant Distributed Computing, IEEE.* Tokyo.

Christian F. (1990). Synchronous Atomic Broadcast for Redundant Broadcast Channels. *Real-Time Systems*, pp. 195-212, Kluwer Academic Publisher.

Comer D. and D. Stevens (1991). *Internetworking with TCP/IP: Principles, Protocols and Architecture* (Volume I). Prentice Hall.

Comer D. E. and J. C. Lin (1993). Probing TCP Implementations. *Purdue Technical Report CSD-TR 93-072*, Purdue University, West Lafayette.

Kochs H.-D., W. Dieterle and E. Dittmar (1993). Reliability of Distributed Computer Control Systems - an Application-Based Analysis. *atp*, 12.

Kopetz H. (1989). Distributed Fault-Tolerant Real-Time Systems: The MARS Approach. *IEEE Micro*, pp. 25-41.

Lamport L. (1978). Time, clocks, and the ordering of events in distributed systems. *Communications of the ACM*, 7, pp. 558-565.

Özalp B. (1990). Fault Tolerant Computing Based on Mach. *ACM Operating Systems Review*, Vol. 24, 1, pp. 27-39.

Powell D. (1991). *Delta-4: A Generic Architecture for Dependable Distributed Computing*. ESPRIT Research Reports. Springer Verlag, Berlin.