IFAC
International Federation of Automatic Control

DISTRIBUTED COMPUTER CONTROL SYSTEMS 1994

*A Postprint volume from the IFAC Workshop
Toledo, Spain, 28-30 September 1994*

Edited by

J. A. DE LA PUENTE and M. G. RODD

PERGAMON

UK Elsevier Science Ltd, The Boulevard, Langford Lane, Kidlington, Oxford, OX5 1GB, UK

USA Elsevier Science Inc., 660 White Plains Road, Tarrytown, New York 10591-5153, USA

JAPAN Elsevier Science Japan, Tsunashima Building Annex, 3-20-12 Yushima, Bunkyo-ku, Tokyo 113, Japan

First edition 1995

Library of Congress Cataloging in Publication Data

A catalogue record for this book is available from the Library of Congress

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 0-08-042237 3

This volume was reproduced by means of the photo-offset process using the manuscripts supplied by the authors of the different papers. The manuscripts have been typed using different typewriters and typefaces. The lay-out, figures and tables of some papers did not agree completely with the standard requirements: consequently the reproduction does not display complete uniformity. To ensure rapid publication this discrepancy could not be changed: nor could the English be checked completely. Therefore, the readers are asked to excuse any deficiencies of this publication which may be due to the above mentioned reasons.

The Editors

Printed in Great Britain

IFAC WORKSHOP ON DISTRIBUTED COMPUTER CONTROL SYSTEMS 1994

Sponsored by
International Federation of Automatic Control (IFAC)
- Technical Committee on Distributed Computer Control Systems

Organized by
Comité Español de Automática, CEA-IFAC

FOREWORD

The 1994 IFAC Workshop on Distributed Computer Control Systems meets for the 12th time in the historic city of Toledo. The IFAC DCCS series has gained wide recognition for its high quality level, which makes it a difficult challenge for the organisers of each new meeting. We expect that both academics and industrial practitioners will find new insight in the field and learn from each others' view.

One of the most important issues in the development of distributed computer control systems is being able to build software and hardware which is both reliable and time deterministic. This is an area where control engineering and computer science naturally meet, and we also expect this workshop to provide a space for cross fertilization between both engineering fields.

I would like to thank the International Programme Committee and its Chairman, Professor Michael Rodd, for their enthusiastic work in setting up an excellent technical programme. The continuous support of the IFAC Technical Committee on Distributed Computer Control Systems, chaired by Professor Ian MacLeod, has been determinant in ensuring the continuity of the DCCS series and making this meeting possible.

Finally, let me thank the support provided by our sponsors, which have provided work and financial aid for the technical and social programme.

Juan A. de la Puente
Universidad Politécnica de Madrid

CONTENTS

REAL-TIME COMMUNICATION ARCHITECTURES

Integration of Temporal Mechanisms in Communication Protocols for Time-Critical Distributed Systems 1
Z. MAMMERI, P. LORENZ

Communication Architectures for Distributed Computer Control Systems 7
W. DIETERLE, H.-D. KOCHS, E. DITTMAR

A Communication Infrastructure for a Fault Tolerant Distributed Real-Time System 13
H. KOPETZ

A Priority-Based Protocol for the 802.3 Network 19
M. LI

TEMPORAL PROPERTIES OF COMMUNICATION SYSTEMS

A High-Precision Time Processor for Distributed Real-Time Systems 23
M. WANNEMACHER, W.A. HALANG

Calculating Controller Area Network (CAN) Message Response Times 29
K. TINDELL, A. BURNS, A. WELLINGS

Distributed Synchronous Processes for Control Systems 35
M.A. PERALDI, C. ANDRE, H. BOUFAIED

Automated Design of Distributed Computer Control Systems with Predictable Timing Behaviour 41
H. THIELEN

ARCHITECTURES FOR DCCS

An Architecture for Real-Time Distributed AI-Based Control Systems 47
J.D. HOLT, M.G. RODD

Distributed Control Architecture for Mobile Robot Operation in Uncertain Environments 53
A. MANDOW, J. GOMEZ-DE-GABRIEL, V.F. MUNOZ, A. OLLERO

Distributed Real-Time Database Approach for Distributed Computer Control Systems 59
D. BUZULOIU, I. DUMITRESCU, D. POPESCU

OPEN AND HETEROGENEOUS DCCS

Open Systems in Distributed Industrial Control - A Critical Review 65
T. RAHKONEN

Distributed Control of Assembly Cells Based on Virtual Manufacturing Device Model 71
G. NAGY, G. HAIDEGGER

An Heterogeneous and Distributed Architecture Based on a Reflective Memory Interconnection Network 77
J.L. LERIDA, D. DEL VAL, J.F. CONDE, A. VINA

Heterogeneous Architectures for Real-Time Control: Design Tools and Scheduling Issues 83
M.J. BAXTER, M.O. TOKHI, P.J. FLEMING

SPECIFICATION AND DESIGN METHODS FOR DCCS

A Target Code Model for Incremental Prototyping 89
N. ZAKHAMA, J.A. DE LA PUENTE

An Object-Oriented Design Method for Distributed Process Control Systems 95
G. HASSAPIS, A. MOSCHOPOULOS, D. THEOS

Graphic Modelisation and Simulation Design Technique Including Temporal Operators 101
G. NOWAK, E. NIEL, A. JUTARD

Comparing RTL and FNLOG- Timing Properties in Real-Time Systems 107
A. SOWMYA

SYSTEM ISSUES

Distributed Control Using a Serial Communication Link 113
E. DUMMERMUTH

Real Time Distributed Control Applied to Process Supervision by SFC Algorithms 119
R. FERREIRO GARCIA

Distributed Control Systems Debugging Based on Global Predicates Detection 125
J. VILA, F. BUENDIA

Replication Issues in the Map/MMS Communication Environment 133
S. MESSINA, P. RAJA

PERFORMANCE ISSUES

System Monitoring for a Real Time Local Area Network 141
J.C. CAMPELO, F. RODRIGUEZ, J.J. SERRANO, M. SANCHEZ, R. ORS

Analysis on the User's Response Time for Mini-Map Systems 147
H.S. PARK, C. LEE, W.H. KWON

Analysis of Temporal Properties of Data Flow Control Systems 153
Gy. CSERTAN, C. BERNADESCHI, A. BONDAVALLI, L. SIMONCINI

A Simulator for Performance Estimation of Open Distributed Computer Control Systems 159
S. HORIIKE, Y. OKAZAKI, H. SOEDA

APPLICATIONS

Structuring in the Design of Real Time DCCS Application to an Industrial Transfer System 165
J.J. SCHWARZ, M. MIQUEL, J. SKUBICH, J.F. PETIT

Implementation of a New Quality Flatness Sensor in the Distributed Computer Control System of Rolling Mills 171
D.F. GARCIA, M. GARCIA, M.A. DEL RIO, J.L. DIAZ, F. SUAREZ

A Case Tool for Modeling and Simulating Distributed Control Systems Based on MMS 177
R. MARIN, J.L. TRILLO, J. GARRIDO

Author Index 183

INTEGRATION OF TEMPORAL MECHANISMS IN COMMUNICATION PROTOCOLS FOR TIME-CRITICAL DISTRIBUTED SYSTEMS

Z. MAMMERI* **and P. LORENZ****

Centre de Recherche en Informatique de Nancy (CNRS URA 262)
* ENSAM, 3 rue de la Rochefoucauld, 51006, Châlons sur marne, France
** ENSEM, 2 Avenue la forêt de haye 54516, Vandoeuvre-les-Nancy, France

Abstract. To deal with faults and dynamic changes in real-time systems, the message and task scheduling is insufficient because there is no scheduling algorithm which can guarantee the respect of all the timing constraints (TCs) under these requirements. So, mechanisms are necessary to tolerate the violation of some TCs. This paper presents an approach to integrate temporal mechanisms in the communication protocols to qualify, with a temporal point of view, data exchanged between distributed processes. These mechanisms enable to know if TCs are met or not. We are especially interested in the producer/consumers communication model.
Key words. Real-time computer systems, Computer communication, Real-time communication, Timing constraints, Temporal mechanisms, Temporal status, Time window, Temporal data validity.

1. INTRODUCTION AND RELATED WORK

Distributed real-time systems (DRTS) are vital for a wide range of applications such as the control and command applications in navigation systems, nuclear power plants, vehicle factories, petroleum plants, ... A real-time system (RTS) is defined as one in which the correctness of its results depends not only on logical computation carried out but also on the time at which the results are delivered (Burns and wellings 1990, Panzieri 1993, Rajkumar 1991). A time-critical system is a real-time system in which the non-respect of timing constraints may lead to production loss, installation deterioration, ...

To meet timing constraints (TCs) in distributed time-critical systems, task scheduling and time-critical management of communication must be combined. Scheduling of time-critical tasks consists of elaborating a processor allocation strategy to guarantee the respect of the TCs. A lot of scheduling algorithms for time-critical systems has been developed (for further details, see Cheng et al 1989, Liu and Layland 1973, Mok 1983, Sprunt et al 1989, Tindel et al 1992, Tripathi and Nirkhe 1991, Xu and Parnas 1991). The proposed algorithms deal with task scheduling with a few consideration for task distribution constraints (i.e., they especially deal with local scheduling). There is no algorithm enabling a global scheduling of tasks with TCs among a distributed system seen as a whole.

An adequate scheduling of tasks is necessary to meet DRTS requirements, but it is not sufficient. In fact, it is necessary to activate the tasks according to their TCs, but it is also necessary to supply the tasks with data at the right moments (i.e., the data used by tasks must be transmitted and received at the adequate times). As, the communications delays are generally non-deterministic, the data arriving at a user entity can become out of use because the temporal data validity is limited by the application nature.

Time-critical (or real-time) communication, defined as communication with explicit timing requirements, is important for networks which interconnect equipment in DRTSs (ISO 1991, Sha 1992). The desirable properties of a network that supports real-time communication include predictable operation and a high degree of schedulability. Timely delivery of messages is essential to the completion of real-time tasks before their deadlines (Zheng and Shin 1992). Like that, task scheduling and communication scheduling are complementary to meet TCs in DRTSs (ISO 1991, Rodd and El-rowairi 1994; Sha et al 1992, Zheng and Shin 1992).

In the beginning of the 80's, the ISO has defined a basic reference model for interconnection of open systems. To reach markets, any network must, as possible, conform to this model. With the emergence of applications requiring not only a reliable delivery of messages but also the respect of TCs, it is necessary to rethink the OSI model. Nowadays, it becomes more and more obvious that the OSI model is a general model that does not integrate time to ensure the temporal validity of the data exchanged between remote tasks. So, it is important to build new mechanisms that take into account the requirements of the communication in the time-critical systems (ISO 1991, Rodd and El-rowairi 1994; Sajkowski 1987).

A time-critical application is composed of several tasks (or application processes) exchanging data with respect to some given TCs. Mechanisms are introduced enabling to know if a variable value is produced, transmitted, received, and consumed according to application timing requirements. Communication between entities may be achieved according to several models: producer/consumers, client/server, client/multiservers, ... models. This paper especially deals with the basic communication model, i.e., the producer/consumers one.

In a time-critical context, once elaborated the TCs must be met at run-time. In practice, if one wants to deal with systems faults and dynamic system changes that are unknown a priori, the messages and tasks scheduling is insufficient because there is no scheduling algorithm which can guarrantee the respect of all the TCs under these requirements. It is why approximate problem solving techniques are used. So, on one hand, one is unable to know that the TCs will be met (because of the variety of the constraints that the task/message schedulers must take into account: TCs, resource constraints, ...) and he wants to tolerate the non-resepct of some TCs. The idea developed in this paper is the definition of some mechanisms to qualify with a temporal view point the data exchanged between tasks. Message scheduling depends mostly on the protocols of the network, and especially on the MAC (medium access control) protocol. The mechanisms proposed in this paper are general and they are not designed for a particular network.

The rest of the paper is structured as follows. In section 2, the concepts of time window and temporal data validity are presented. Section 3 presents the temporal statuses useful to qualify exchanged data. Section 4 presents the rules to respect when elaborating time windows to schedule operations related to time-critical communications. Some conclusions appear in section 5.

2. TIME WINDOWS AND DATA VALIDITY

2.1 *Time window*

In the literature, several models and methods, such as temporal logic, interval logic, and hierarchical multi-state machines, have been proposed for specifying, reasoning about, verifying and validating TCs in real-time systems. As each operation in a communication relationship must occur in a given period of time (but not at a fixed instant) to respect some TCs, the concept of time window is used in this paper to deal with TCs in time-critical communication.

A time window (TW) is defined by its start and finish times. A start (or a finish) instant may be static or dynamic, and the length of a TW may be constant or variable. The association of a TW to an operation means that this operation must be activated after the TW start time and it must be terminated before the TW finish time.

2.2 *Temporal data validity*

In a DRTS, variable values are produced by entities called *producers* and they are used (or consumed) by entities called *consumers*. The producers and consumers are connected by means of communication network(s). A variable value is produced at instant Tp; it is transmitted to the consumer at instant Tt; it is received by the consumer at Tr. At the instant Tc ($Tp < Tt < Tr < Tc$), the consumer wants to use the received variable. An important question arises: the available value is it still valid at instant Tc ?

To understand the notion of data validity, let us consider, as an example, a system controlling the level of a liquid in a cistern. The liquid level is measured every second. The measurements are sent to a control task and to a statistical study task. When the liquid level exceeds a given threshold, the gate must be closed within five seconds. Here, the information "threshold exceeding" is communicated to the task which closes the gate and to the task which stores threshold exceeding instants in a statistical file. The first task must receive and process the information within five seconds after the detection of the threshold exceeding. The second task has only to store the instants of threshold exceeding with no TCs on the storing operation.

The previous example shows the importance of temporal validity, to ensure correctness of actions in DRTSs. Also, the example shows that the life time of a data may be variable according to each consumer. In DRTSs, is it is necessary to clarify the temporal data validity for each variable with regard to the production and consumption time windows to ensure the data consumption coherence. A produced variable value is valid during a particular period of time with regard to each consumer. This period is the *temporal validity* of the variable value. A variable value is valid in a given time window called *Temporal Validity Window*. The consumer must terminate its consumption operation before the end of the temporal validity window. In consequence, emission, receipt and consumption operations must be scheduled by taking into account the end of the temporal validity windows.

3. TEMPORAL QUALIFICATION OF DATA

3.1 *Time windows for time-critical communication*

The communication between a producer and a consumer is achieved according to several steps:
• production of a variable value,
• the variable value passes through the stack of the communication layers, from the application layer up to the physical layer, at the producer station,
• the variable value passes through the stack of the

communication layers, from the physical layer up to the application layer, at the consumer station,
• consumption of the variable value.

In a time-critical context, production and consumption operations but also all the communication layers must respect certain TCs to guarrantee that the end-to-end (i.e., producer/consumer) cooperation will meet the whole application TCs. Each one of the previous steps necessitates a local processing and a message transfer achieved by the lower layer. According to the architecture of the used network, the communication stack may be composed of three layers (1, 2 and 7), ..., or seven layers.

To facilitate the analysis of temporal validity of variable values, and to have a good knowledge about the TCs, we associate a time window with each step; this time window will specify the time interval during which the step must be started and finished. Like that, we have to specify the following TWs:
• a TW for the production step (this TW is called *production TW*),
• a TW for each communication layer (from the application layer up to data link layer (or exactly up to the MAC (i.e., medium access control) sublayer) for the producer station (this TW is called *layer_i_emission TW*, i = 7, ..., 3, MAC)
• a TW for each communication layer (from the MAC sublayer up to the application layer) for the consumer station (this TW is called *layer_i_receipt TW*, i = 7, ..., 3, MAC),
• a TW for the consumption step (this TW is called *consumption TW*).
Notice that no TW is associated with the physical layer because it is very diffuclt and useless to elaborate a temporal status for each transmitted or received bit.

One end-to-end TW, or one end-to-end delay, is often (or even, usually) associated with a communication relationship (i.e., one TW is associated with all the steps of a producer/consumer communication). We propose the use of several TWs for the following reasons:
• to determine a correct and effective end-to-end delay, it is necessary to know the precise delay associated with each step,
• As the proposed mechanisms are usefull to enhance the understanding of the abnormalies and faults causes, the separation of the TWs enables to locate the exact cause of non-respect of TCs.
• To be capable of meeting the CTs, each communication layer must inquire about the CTs it must respect, otherwise how could it know the urgency or the priority of the messagses it processes.
• The present works on time-critical networks, such as FIP (UTE 90) and Profibus (Menden 1992), and on transport layer (Danthine et al 1993) and XTP (eXpress Transfer Protocol) propose the introduction of temporal mechanisms at all the layers of a time-critical communication architecture (ISO 91), in order to control more effecively the respect of TCs.
• Finally, at a given communication step, with an end-to-end delay strategy, a message is sent if its end-to-end delay is not over even though the minimal delay necessary to achieve its final destination is greater than its present time validity. With a TW for each step, a message that will miss its its deadline is known earlier and the message scheduler, at each step, "previliges" messages that have a high probability to reach destination with respect to TCs. Also, this is a means of removing late messages from the network. In consequence, the message scheduling is optimized.

3.2 *Temporal statuses*

A *temporal status* is associate to each step of the communication. This status enables to know if the TCs assigned to the corresponding step (or time window) are satisfied, or not. The *production status* is elaborated by the production station of the variable; it enables to know if the variable value has been produced with respect to TCs. The *consumption status* indicates, to the consumer, the validity of the variable value available to consumption; it allows to know if the variable value available at the consumption station is valid for consumption, or not. It is elaborated by all the consumption stations. The *Layer-<i>-emission status* indicates if the variable value has been processed, at the communication layer i of the producer station, with respect to the fixed TCs constraints (<i> =7, ..., MAC). The *Layer-<i>-receipt status* indicates if the variable value has been processed, at the communication layer i of the consummer station, with respect to the fixed TCs.

The temporal status associated with each communication step is elaborated by an entity controlling the respect of the TCs, this entity is called TCCE (Timing Constraint Control Entity). The sequencing of a time-critical communication is organized as follows:

1) Production of a variable value VV.
2) Once the production operation is over, the production TCCE tests if the production TCs have been respected, and elaborates the production status (PS). The message to send is {VV, PS}.
3) The message {VV,PS} is available at communication entity of the production station. At each communication layer (from 7 to MAC), the TCs associated with the layer are controled and a temporal status is elaborated. The message actually transmitted on the medium is TrM = {VV, PS, L7ES, ... L3ES, LMACES} (where LiES means the temporal status elaborated by the layer i, and E means emission)
4) When the message TrM reaches the destination station, temporal statuses associated with receipt steps are elaborated. So, the message available at the consumer level is RcM = {TrM, L7RS, ... L3RS, LMACRS} (R means receipt).
5) At the instant where the consumer is ready to consume the available message, the consumption TCCE tests if the consumer has respected, or not, the consumption TCs, and elaborates the consumption status (CS).

The consumer does not receive only a variable value but a message containing the produced variable value and several temporal statuses. If all the temporal statuses are set to True, then the variable value is valid for consumption, otherwise it is invalid, and the consumer uses the temporal statuses to know why the value is invalid. When no variable value is received, the consumption TCCE sets VV to a special value to inform the consumer.

At the implementation level, a bit is associated with each temporal status. In consequence, a temporal status control byte (i.e., a byte containing the temporal statuses elaborated by the producer station) is integrated in any variable value message. So it is obvious that the throughput of the network is affected by the presence of the temporal statuses.

4. COMMUNICATION-ORIENTED OPERATIONS SCHEDULING

Variable values may be exchanged periodically (for data sampling, ...) or aperiodically (for file transfer, alarm notification, ...) between producers and consumers. For a periodical communication, the start and the finish times of TWs are mostly determined by using the period of the communication. For an aperiodical communication, the start time is often unknown a priori, but once the start time is determined, it is possible to determine the finish time (this time corresponds to the communication deadline). In some real-time contexts, tasks are executed at known times (for example, "the factory siren must be activated at 12:00 a.m"). So, the start times of communication associated to such tasks are known a priori.
A variable may be consumed by one or several consumers. Once a variable value has been produced, each consumer must consume this value before a given amount of time (i.e., while the variable value is valid for the consumer). The produced value is communicated to each consumer with regard to the temporal validity of the value for this consumer (there is a temporal validity window for each consumer). The communication entity of the production station uses the temporal validity associated to each consumer, for each variable, to schedule the transmission of messages containing variable values. In this section, we analyze how to elaborate the TWs associated with a critical-time communication. Rules to respect when scheduling communication-related operations are introduced. The following notations are used in this section.

Notations:
c : a consumer identifier *v : a variable identifier*
VV : Variable Value *TW : Time Window*
TVW : Temporal Validity Window
i, k : numbers of produced VV
Cn : number of consumers of the variable v
PP(v) : Production Period of the variable v.
PC(v,c) : Period of Consumption of v for the consumer c
$V\Uparrow(v,i,c)$: Start time of TVW for i^{th} value of v for c
$V\Downarrow(v,i,c)$: Finish time of the TVW for i^{th} value of v for c
VL(v,i,c) : Length of the TVW of i^{th} v's value for c
$P\Uparrow(v, i)$: Start time of the production TW of i^{th} v's value
$P\Downarrow(v, i)$: Finish time of the production TW of i^{th} v's value
$X\Uparrow(v, i, c)$: Start time of the TW of the operation <X> of i^{th} v's value for c,
X = E7 (Emission at layer 7), ..., EMAC : (Emission at layer MAC), RMAC : (Receipt at layer MAC), ..., R7 : (Receipt at layer 7), C : (Consumption)
$X\Downarrow(v, i, c)$: Finish time of TW of the operation <X> of i^{th} v's value for c
$P@\Downarrow(v,i)$: Finish time of the production of i^{th} v's value,
$R1@\Uparrow(v,i,c)$: Instant of the arrival of the message containing the i^{th} v's value at the physical layer of c
$R7@\Downarrow(v,i,c)$: Instant of the arrival of the message containing the i^{th} v's value at the application layer of c
$EV@\Uparrow(v,i)$: instant of occurrence of the event leading the production of the i^{th} value of an aperiodic variable v,
Dprd(v) : execution delay to produce a value of v
Demi(v) : maximal delay, for a v's message, to pass through the layers 7 to 1 at the producer station
Drcp(v,c) : maximal delay, for a v's message, to pass through the layers 1 to 7 at the consumer c
Dpr(v,c) : maximal delay of propagation of a message containing a value of the variable v
Dpr(v,c) : processing delay of the received VV by c
t0 : the instant of the application start.

4.1 *Rules for elaboration of time windows*

The lengths of TWs and their sequencing, in time, depend on whether the exchange of the variable is periodic or aperiodic.

Periodic exchange of a variable

A periodic exchange of variable means a situation in which an entity produces periodically variable values that are consumed by other entities. Each consumer may have its own consumption period. First, the production period must be less or equal to the lowest consumption period, otherwise, some consumers are led to consume several times the same variable value. So, the rule R1 must be respected when defining the periods of producers and consumers.

$$R1 : PP(v) \leq \min \{PC(v,c_j), j=1,\ldots,Cn\}$$

Second, a consumer with a period less than the production period is not concerned by all the produced values. In consequence, the producer does not necessarily send the produced values to all the consumers: each consumer is supplied with variable values according to its consumption speed. One may notice that if all the produced variable values are transmitted to all consumers, the network traffic may be increased uselessly (if broadcasting is not available) when some consumption periods are greater than the production one.
There are two basic possibilities to ensure the sequencing of time windows for periodic variables: by using static TWs or by using dynamic TWs.
• *Static TWs*: the start and the finish of a TW are fixed times. Then, a production occurring earlier or later in the production TW has no effect on the start time of the emission TW. In the same way, in the

consumer station, receiving the variable value earlier or later in the receipt TW has no effect on the start time of the consumption of the received value.

• *Dynamic TWs*: two cases must be considered:
- For the producer, the production is the first operation in a producer/consumer relationship; its start and finish times depend only on the production period. In consequence, production TW is always fixed. Once the production is terminated, the variable value is prepared to send. So the emission TW start time is linked to the end of the variable value production.
- For the consumer, the receipt TW begins when the first digit arrives at the consumer station. The finish time of the receipt TW must be static to detect if a value is received or not before a given deadline (i.e., to detect the loss and the late emission of messages). The start time of the consumption TW is not fixed and when the variable value is received it is possible to consume this value. In this case, the consumption TW start is a time immediately after the end of the variable value receipt at application layer.

A dynamic TW may have a fixed or a variable length. With variable lengths, one may use the time saved in a step to extend the duration of the next step; in consequence the TCs of some operations may be relaxed. TCs relaxation is often useful for operation scheduling.

When static TWs are used, the rules R2 must be respected when defining the TWs.

R2.1.1: P↑(v,i+1) = P↑(v,i) + PP(v)
R2.1.2: P↓(v,i+1) = P↓(v,i) + PP(v)
R2.2.1: X↑(v,k+1,c) = X↑(v,k,c) + PC(v,c)
R2.2.2: X↓(v,k+1,c) = X↓(v,k,c) + PC(v,c)

If (n)*PP(v) ≤ PC(v,c) ≤ (n+1)*PP(v), then n variable values are produced in every consumption period, and among these VV only one is sent to the consumer. When dynamic TWs are used, the rules R3 must be respected when defining TWs.

R3.1.1: P↑(v,i+1) = P↑(v,i) + PP(v)
R3.1.2: P↓(v,i+1) = P↓(v,i) + PP(v)
R3.2.1 : E7↑(v,k,c) ≥ P@↓(v,k)
R3.2.2 : EMAC↓(v,k,c) ≤ t0 +(k)*PC(v,c) - D1
D1 = Dprg(v,c) + Drcp(v,c)+ Dprc(v,c))
R3.3.1 : RMAC↑(v,k,c) = R1@↑(v,k,c)
R3.3.2 : R7↓(v,k+1,c) ≤ R7↓(v,k,c) + PC(v,c)
R3.4.1 : C↑(v,k,c) > R1@↑(v,k,c)
R3.4.2 : C↓(v,k,c) ≤ t0 + (k)*PC(v,c)

As previously mentioned, a produced variable value has a specific life time for each consumer. Once a variable value is produced, this variable value must be sent, received and acted upon while it is valid. The following rules (R4) express this constraint. When the temporal validity window length is less than the period of the consumer, the rules R4 are more accurate than rules R3 to delimit the TWs.

R4.1 : EMAC↓(v,i,c) ≤ V↓(v,i,c) - D2
D2 = Dprg(v,c) + Drcp(v,c)+ Dprc(v,c)
R4.2 : R7↓(v,i,c) ≤ V↓(v,i,c) - Dprc(v,c)
R4.3 : C↓(v,i,c) ≤ V↓(v,i,c)

Aperiodic exchange of a variable

An aperiodic communication is issued when a specific event occurs (for example, an alarm notification). Once the event leading to the variable exchange has been detected, a variable value is produced, sent to all interested consumers, and consumed with respect to the temporal validity associated to each consumer. So, the start and the finish times of all TWs are conditioned by the instant of event detection and the variable temporal validity duration associated to each consumer. As the consumer cannot know the exact time of the beginning of the temporal validity of an aperiodic variable, and as it cannot know the exact duration between the value production and the beginning of its receipt, the consumer TCCEs can not adjust the receipt and consumption TWs. This problem may be solved by using the following mechanism: the production station stamps the variable values with its local real-time clock at end of the value production. By using the variable value time-stamp, the consumer computes the time remaining for validity of the variable value, and it adjusts its TWs. Nevertheless, this solution requires that the real-time clocks of the producer and the consumers must be synchronized.

The rules R5 must be respected when sequencing the TWs for an aperiodic communication. The instant of the production end of the variable value, noted P@↓(v, i), is integrated in the message containing the ith VV.

R5.1.1 : P↑(v,i) ≥ EV@↑(v,i)
R5.1.2 : P↓(v,i) < EV@↑(v,i) + A
A = min {(VL(v,i,cj) - Tj), j=1,...,Cn}
Tj = Demi(v,cj)+Dprg(v,cj)+Drcp(v,cj)+Dprc(v,cj)
R5.2.1 : E7↑(v,i,c) ≥ P@↓(v,i)
R5.2.2 : EMAC↓(v,i,c) ≤ EV@↑(v,i)+VL(v,i,c)-D3
D3 = Dprg(v,c) + Drcp(v,c) + Dprc(v,c)
R5.3.1 : RMAC↑(v,i,c) = R1@↑(v,i,c)
R5.3.2 : R7↓(v,i,c) ≤ P@↓(v, i)+VL(v,i,c)-Dprc(v,c)
R5.4.1 : C↑(v,i,c) ≥ R7@↓(v,i,c)
R5.4.2 : C↓(v,i,c) ≤ P@↓(v,i) + VL(v,i,c)

4.2 *Time-critical communication scheduling*

The previous rules for determination of start and finish times of each time window must be used as inputs for the task schedulers in producer and consumer stations. Also, according to each application, the lengths of the temporal validity windows, the periods (for periodic variables) must be fixed and used as inputs of the tasks schedulers. The elaboration of the finish times of TWs is based on the knowledge one has about production, emission, receipt and consumption delays. The maximal values computed for these delays must be estimated in such a way that the probability of respect of TCs is as high as possible. To calculate the different delays, it is necessary to take into account different types of constraints; especially the services of the used networks. In fact, the delays are very different according to the type of the used network Ethernet, MAP or FIP (UTE 90), ...).

In order to define the TWs, the application designer must specify:
• the production and consumption periods for periodic variables,
• the events leading to aperiodic exchanges,
• the temporal validity of the variable values, for each consumer,
• the delays necessary for emission and receipt that are computed according to the characteristics of the network(s) connecting the producers and consumers.

To activate and terminate the operations related to time-critical communications, time-critical-oriented scheduling algorithms must be used. The constraints to taken into account by the schedulers are TCs and precedence constraints (i.e., emission begins after the end of the production, ... and the consumption begins after the end of the receipt). Time-critical scheduling algorithms which may be used are those presented in (Cheng et al 1989, Liu and Layland 1973, Mok 1983, Sprunt et al 1989, Tindel et al 1992, Tripathi 1991, Xu and Parnas 1991).

5. CONCLUSION

When a time-critical application is distributed, the communication system enabling data exchange between tasks must be chosen in such a way that communication delays do not affect temporal validity of exchanged data.
The management of time-critical communications requires time window mechanisms to elaborate temporal statuses, and real-time-oriented algorithms for scheduling operations related to production, emission, receipt and consumption of messages. This paper has presented some mechanisms to elaborate temporal statuses qualifying variable values exchanged between distributed tasks according to the producer/consumers model. The proposed mechanisms represent a beginning for potential extensions of the OSI model to take into account the temporal aspects of communication in time-critical distributed applications. For each OSI layer, we advise the integration of previously presented mechanisms.

It is recognized today that communication systems which are to be used in time-critical distributed systems must be designed with TCs always in mind (Rodd 1994). It is diffuclt, even impossible, to obtain a time-critical scheduling of messages over present networks that are not designed for time-critical systems. New networks must be invented. We beleive that the mechanisms presented in this paper will be integrated within the future real-time communication systems.
Finally, we note that the temporal mechanisms presented in this paper have been validated and implemented in the FIP network (UTE 90). Further work has to be carried on to deal with temporal data validity in other communication models such as client/server and client/multiserver models.

6. REFERENCES

Burns, A., and Wellings, A. (1990). *Real-time systems and their programming languages*. International computer series, Addison Wesley.

Cheng, C.C., Stankovic, J.A. and Ramamrithan, K. (1989). Scheduling algorithms for hard real-time systems, *Real-time systems Newsletter* 3(2):1-24.

Danthine, A. et al. (1993). The OSI 95 Connection-mode transport service - The enhanced Qos. In *High performance networking*. (A. Danthine and A. Spaniol, Ed.), pp. 235-252, Elsevier Sc. Pub. B.V. (North-Holland).

ISO, (1991). Interim report of the TCCA Rapporteurs group of ISO/TC 184/SC 5/WG2 on time-critical Communications Architecture and System. *ISO/TC 184/SC 5/WG 2*, Report N° 254, April 24.

Liu, C., and Layland, J. (1973). Scheduling algorithms for multiprocessing in a hard real-time environment. *Journal of ACM*, 20(1): 46-61.

Menden, R. (1992). Basic information on Profibus. *Eds. Klockner-Moeller* Bonn.

Mok, A. (1983). Fundamental design problems of distributed systems for the hard real-time environment. *PhD Thesis*, MIT, May.

Panzieri. F., and Davoli. R. (1993). Real-time systems : a tutorial. *Performance evaluation of computer and communication systems*. LNCS (729): 435-462.

Rajkumar, R. (1991). Synchronization in real-time systems, a priority inherence approach. *Eds. Kluwer academic publishers*.

Rodd, M.G., (1994). Communications for real-time industrial control: The design issues. In *Real time computing*, (Eds. W. A. Halang and A. D. Stoyenko), Springer Verlag, pp. 111-130

Rodd, M.G., and Al-rowaihi, S.F. (1994). Temporal modelling of real-time communication protocols based on a processor/channel approach. *J. of Real-time systems* (6): 243-262.

Sajkowski, M. (1987). Protocol verification in the presence of time. *Protocol specification, testing and verification, VI*, Elsevier science publishers B.V, pp. 269-280

Sha, L., Sathaye, S.S., and Strosnider, J.K. (1992). Scheduling real-time communication on dual-link networks. *Proceedings IEEE Real-time systems sympo.* Phoenix, Arizona, Dec., pp. 188-197.

Sprunt, B., Sha, L., and Lehoczky, J.-P. (1989). Aperiodic Task Scheduling for Hard Real-Time Systems. *J. of Real-Time Systems* (1), pp. 27-60.

Tindell, K., Burns, A., and Wellings, A. (1992). Allocating Hard Real-Time Tasks: An NP-Hard Problem Made Easy. *J. of Real-Time Systems*. (4): 145-165.

Tripathi, S.K., and Nirkhe, V. (1991). Pre-scheduling for synchronization in hard real-times systems. *Operating systems of the 90s and beyond*. LNCS(653): 102-108.

UTE, (1990). FIP: Application layer and data link layer. *Union Technique de l'Electricité*. Paris.

Xu, J., and Parnas, D.L. (1991). On satisfying timing constraints in hard-real-time systems. *Proc. of the ACM SIGSOFT'91 Conf. on Soft. for Critical Systems*. New Orleans, December, pp. 132-146.

Zheng, Q., and Shin, K.G. (1992). Fault-tolerant real-time communication in distributed computing systems. *22nd Intern. Sympo. on Fault-tolerant Computing*. Boston. October, pp. 86-93.

COMMUNICATION ARCHITECTURES FOR DISTRIBUTED COMPUTER CONTROL SYSTEMS

W. DIETERLE*, H.-D. KOCHS* and E. DITTMAR**

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

Abstract. The use of distributed computer control systems (DCCS) demands high reliability, sufficient real-time behaviour and increasingly economical systems. The last demand requires the use of cheap standard components, whenever possible. The following article discusses realization of DCCS with respect to these constraints. Problems due to 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 towards DCCS.

Keywords. Control Systems; Local Area Networks; Computer Communication; Distributed Databases; Communication Protocol.

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 dominating. Costs minimization makes use of cheap, standardized components and design of simple, modular system concepts mandatory. In the following specific architectural features for system communication, type 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 for support of these features, however, lack of appropriate standard protocols in the UNIX environment requires use of the existing ones. Problems with 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, nearby 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 size 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 considered systems 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. According to high reliability demands computers with important functions are redundantly structured.

The functional scope of such systems incorporates data acquisition, basic processing of process data (SCADA), process visualization (MMI) as well as additional functional modules (complex secondary functions) depending on the concrete application purpose.

Due to 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 process and control

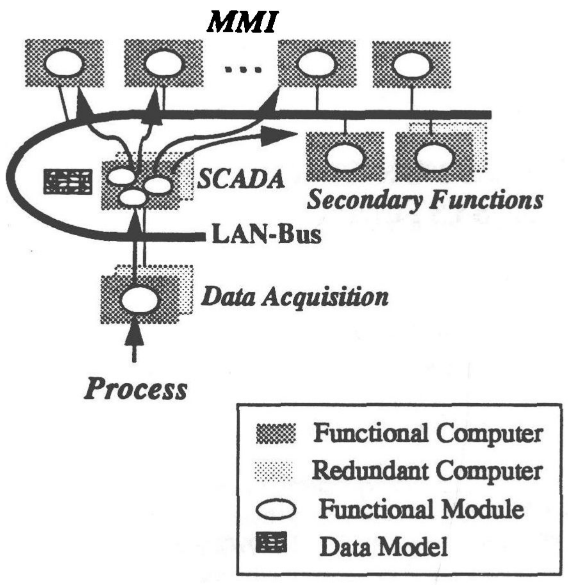

Fig. 1: System Architecture of Distributed Computer Control Systems

system is held in static data models. Process state is kept in dynamic data models (100000-200000 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. In the following we are only concerned with dynamic data models, which are to be continuously actualized.

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, simple system concepts as well as independence from a particular manufacturer. The use of standards concerns hardware (Workstation, PC), operating system (UNIX), 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). In the following we 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 case of failure of redundant components.

Due to centralization effects in the direction of the higher control levels and consequences according to component failures, high *reliability* by means of structural redundancy and *fault tolerance* is required for industrial computer control systems. Computers with important functions and for very high reliability demands also the LAN bus have 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 for 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 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 data bases is data consistency. In case of fault-free operation data consistency is trivial to be ensured. 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 retrieval of information by transfer of complete data sets or to avoid inconsistencies by means of sophisticated approaches.

Time-costly retrieval of information in case 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 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 computer load (context switches). Thus, they are not appropriate for the systems considered. Existing asynchronous solutions (Birman, 1987; Özalp, 1990) are targeted to systems beyond the scope of DCCS and thus are 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.

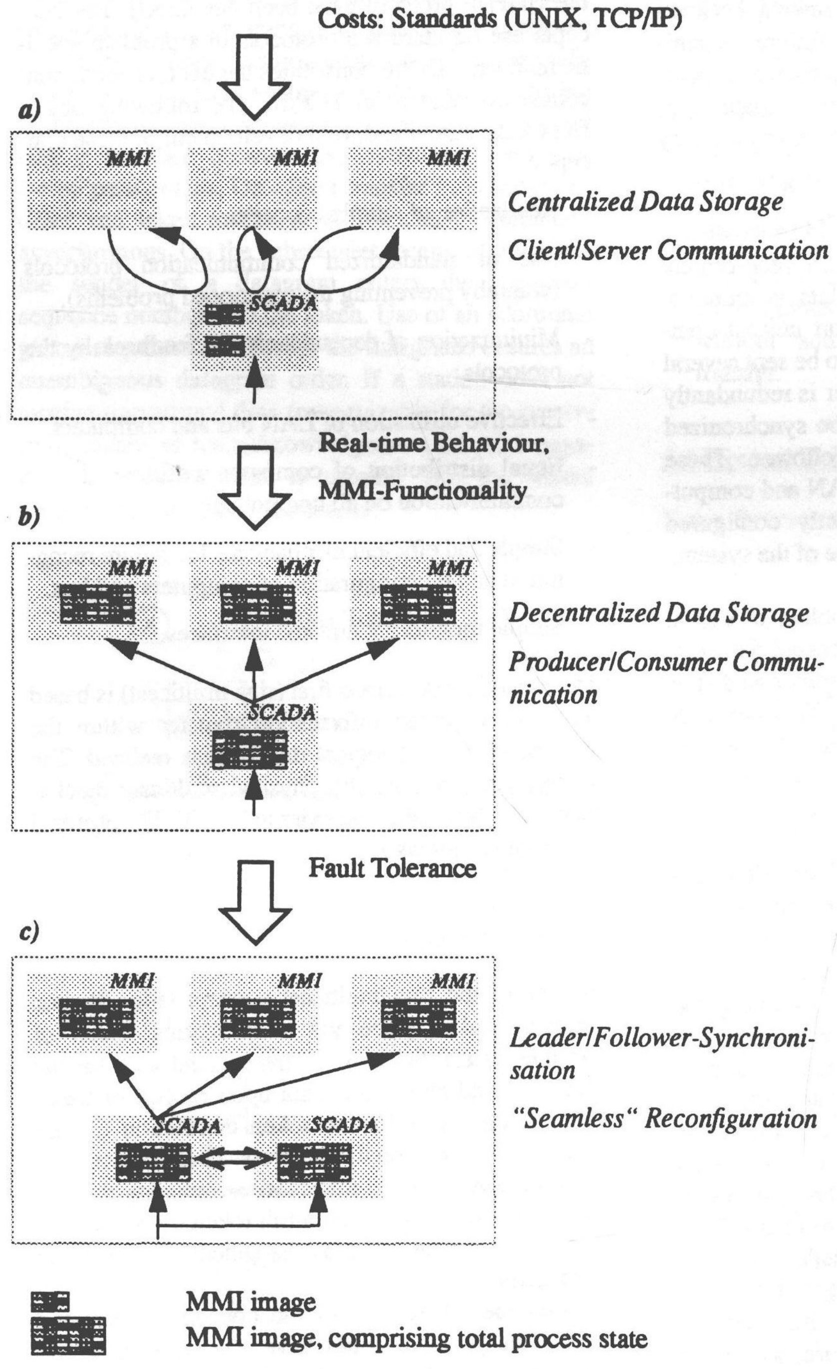
Costs: Standards (UNIX, TCP/IP)
a)
MMI
MMI
MMI
SCADA
Centralized Data Storage
Client/Server Communication
Real-time Behaviour,
MMI-Functionality
b)
MMI
MMI
MMI
SCADA
Decentralized Data Storage
Producer/Consumer Communication
Fault Tolerance
c)
MMI
MMI
MMI
SCADA
SCADA
Leader/Follower-Synchronisation
"Seamless"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

Fig. 2: Architectural Features of DCCS

None of the existing concepts fits as an appropriate communication architecture for the DCCS considered. As a consequence, several communication concepts were developed and shall be discussed in the following.

4. CONVENTIONAL USE OF STANDARDIZED PROTOCOLS

Solutions have been developed with particular emphasis on simple (and thus cheap) concepts, modular system architecture with 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. Conventional use of standardized communication protocols means 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 in the following. The statements apply to the TCP/IP protocol (Comer, 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, 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 high work load for LAN and computers, in particular for the redundantly configured SCADA computer as the logical centre of the system.

In distributed systems exists the problem of causal and total order of transferred and processed data, 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 case 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, 1993).
Further problems concern the necessity of additional buffering of sender data at the application level for the prevention of data loss in case of connection timeout.

5. MULTICAST CONCEPTS FOR DCCS

Due to the problems with 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, nonconnected pendant to TCP/IP. The following objectives were aimed at during development of these concepts:

- Realization of simple concepts.
- Use of standardized communication protocols (whereby preventing the mentioned problems).
- 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

In 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 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), 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).