DISTRIBUTED COMPUTER CONTROL SYSTEMS 1983

Edited by
M. G. RODD

DISTRIBUTED COMPUTER CONTROL SYSTEMS 1983

Proceedings of the Fifth IFAC Workshop
Sabi-Sabi, Transvaal, South Africa, 18-20 May 1983

Edited by

M. G. RODD

University of the Witwatersrand
Johannesburg, South Africa

Published for the

INTERNATIONAL FEDERATION OF AUTOMATIC CONTROL

by

PERGAMON PRESS

OXFORD · NEW YORK · TORONTO · SYDNEY · PARIS · FRANKFURT

U.K.	Pergamon Press Ltd., Headington Hill Hall, Oxford OX3 0BW, England
U.S.A.	Pergamon Press Inc., Maxwell House, Fairview Park, Elmsford, New York 10523, U.S.A.
CANADA	Pergamon Press Canada Ltd., Suite 104, 150 Consumers Road, Willowdale, Ontario M2J 1P9, Canada
AUSTRALIA	Pergamon Press (Aust.)Pty. Ltd., P.O. Box 544, Potts Point, N.S.W. 2011, Australia
FRANCE	Pergamon Press SARL, 24 rue des Ecoles, 75240 Paris, Cedex 05, France
FEDERAL REPUBLIC OF GERMANY	Pergamon Press GmbH, Hammerweg 6, D-6242 Kronberg-Taunus, Federal Republic of Germany

First edition 1984

Library of Congress Cataloging in Publication Data
Main entry under title:
Distributed computer control systems 1983.
Proceedings of the IFAC Workshop on Distributed Computer Control Systems, sponsored by the International Federation of Automatic Control Computer Committee. Includes index.
1. Automatic control—Data processing—Congresses. 2. Electronic data processing—Distributed processing—Congresses.
I. Rodd, M. G. II. International Federation of Automatic Control. III. IFAC Workshop on Distributed Computer Control Systems (5th : 1983 : Sabi-Sabi, South Africa)
IV. International Federation of Automatic Control. Computer Committee.
TJ212.2.D57 1984 629.8'95 83-19494

British Library Cataloguing in Publication Data
IFAC Workshop *(5th : 1983 : Sabi-Sabi)*
Distributed computer control systems 1983 – (IFAC proceedings series)
1. Automatic control – Data processing – Congresses 2. Electronic data processing – Distributed processing – Congresses
I. Title II. Rodd, M. G. III. International Federation of Automatic Control IV. Series
629.8'95 TJ212
ISBN 0-08-030546-6

These proceedings were 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 Editor

Printed in Great Britain by A. Wheaton & Co. Ltd., Exeter

IFAC WORKSHOP ON DISTRIBUTED COMPUTER CONTROL SYSTEMS 1983

Organized by
The South African Council for Automation and Computation (SACAC)
The Council for Mineral Technology (MINTEK)

Sponsored by
The International Federation of Automatic Control Computer Committee (COMPCON)

International Program Committee
R. W. Gellie, Australia (Chairman)
J. Gertler, Hungary
T. J. Harrison, U.S.A.
A. Inamoto, Japan
Lan Jin, China
H. Kopetz, Austria
S. Narita, Japan
M. Sloman, U.K.
G. Suski, U.S.A.
B. Tamm, U.S.S.R.
J. D. N. van Wyk, South Africa
D. Waye, Canada

National Organizing Committee
M. G. Rodd (Chairman)
W. P. Gertenbach
L. F. Haughton
I. M. MacLeod
N. J. Peberdy
G. Sommer
A. B. Stewart
L. van der Westhuizen

PREFACE

One of the greatest values of holding a series of annual Workshops on a given topic is the way in which the growth of an idea can be highlighted at its various developmental stages. This has certainly been true of the IFAC Workshops on Distributed Computer Control Systems, and the Fifth in the series has upheld the tradition. Although attendance at each Workshop has been dominated by participants from the host country, there is a core of authors, of very varied backgrounds, who have presented papers which may appear superficially to be widely divergent. Nevertheless, particularly in the discussion sessions, a common thread of fundamental principles runs through.

The current Workshop was notable for the general agreement which became evident in a number of areas. Probably the most significant was the wide acceptance of the vital role of Real real-time in a distributed system. There are differing views on implementation, but it has become clear that Real real-time must be a fundamental consideration in the planning of all future DCCS architectures.

Communication systems, and the protocols involved, also received much attention. Again, there was, inevitably, some disagreement over issues such as broadcast messages and addressing techniques; also the types of messages required: state, event, immediate, consistent, etc. In spite of confusion over the exact requirements for a generalized approach, many common concepts are being developed. The actual communication media to be used provoked much discussion, and the work of the IEEE-802 Committee is having a serious impact - as is PROWAY.

Another aspect which continues to be of great interest is the language question, as increasing numbers of users are looking beyond FORTRAN. ADA is generally being hailed as a major development, despite some reservations over the interprocess communication primitives available.

Finally, the issue of fault-tolerance recurred, to the extent that many delegates to this event felt that a Distributed Computer Control System should, by definition, be fault-tolerant!

The Workshops have been noted for their high standard of papers, and the 5th Workshop maintained this level. The policy of inviting the bulk of the authors, and of accepting only a limited number of papers has paid dividends. Such presentations, in the Workshop situation, lead to fruitful discussion which is in many ways the highlight of the entire exercise. The Sabi Sabi venue, remote and highly unusual, provided an ideal environment for such debate. It is the Editor's hope that readers of the edited discussions contained in these Proceedings will benefit from that stimulating exchange of ideas.

WELCOMING ADDRESS

G. Brown

President of the South African Council for Automation and Computation.
AECI Limited, PO Box 796, Germiston, South Africa

It is my sincere pleasure, to welcome you all to the 5th IFAC Workshop on Distributed Computer Control Systems. I especially welcome our overseas delegates who have taken the time and trouble to travel great distances to join us here at Sabi-Sabi.

It seems to me that it is most appropriate that this Workshop should be held in South Africa at this time. Whilst South Africa is clearly a developing country and is still small in the industrial sense, we are, I believe, a technologically progressive nation. For example, I think we can lay claim to one of the largest installed proprietary Distributed Control Systems anywhere in the world, referring, of course, to the Sasol II and III oil from coal plants. For those of you with knowledge of these huge plants, it is inconceivable to think that they could be successfully controlled without the facilities provided by Distributed Computer Control Systems. Several other processing industries in this country are also users of Distributed Control Systems on a significant scale.

In parallel with this growing local industrial application of proprietary DCCS, the past few years have seen the vigorous growth of important research work in this field at our Universities and other institutions and notable results have already been achieved by this research. Thus, we now have in South Africa the beginning of an excellent environment for cross fertilisation of ideas between experienced and imaginative industrial users and highly competent research and development teams.

The Sabi-Sabi Workshop thus provides a timely platform for many South African workers in this field to exchange ideas with our esteemed overseas delegates and thus add an international dimension to the beginning of the local environment of cross fertilisation of ideas.

A Workshop of this type does not occur without considerable effort being expended by many people, and I cannot let this opportunity pass without expressing my thanks to all concerned. In particular I must thank the authors for their work in preparing the papers, without which the Workshop could not take place. Our thanks are also due to the International Program Committee, under the Chairmanship of Warren Gellie for selecting a set of papers which are clearly of high standard. Finally, on behalf of SACAC, I would like to thank Mike Rodd and his National Organizing Committee for a tremendous job well done!

CONTENTS

THE DISTRIBUTED DATA FLOW ASPECT OF INDUSTRIAL COMPUTER SYSTEMS

R. Güth and Th. Lalive d'Epinay

Brown Boveri Research Center, CH-5405 Baden, Switzerland

Abstract. Today's computer systems are in general based on the von Neumann type architecture. However, there also exist alternative architectures, for example dataflow systems and (quasi-) continuous systems.

This paper presents a new type of computer architecture which is based on broadcast principle and on source addressing. This system can be viewed as a combination and generalisation of dataflow and continuous systems. It has some inherent advantages over the classical architecture, but there are also some difficulties in efficient implementation. A concept has been established which allows to implement broadcast systems efficiently and hence to utilize their advantages.

There already exist distributed process control systems based on the broadcast and source addressed architecture. Ongoing research work in the field of fundamental concepts and their efficient implementation will broaden the scope of these systems and make them applicable in a wide range of process control and other applications.

Keywords. Dataflow system; control system; continuous system; broadcast system; source addressing; communication ether.

A DATAFLOW VIEW OF CONTROL SYSTEMS

Most of today's computer systems for commercial as well as for process control applications are based on a conventional, von Neumann type architecture. As will be shown in this paper, an alternative architecture has many advantages. We will present a logical model of an architecture that is derived from and will cover two extremes:

1) Dataflow systems: This concept is mainly known and propagated by a relatively small group of computer scientists. Despite its theoretical advantages it has only been practically used for special applications.

2) Continuous systems (analogue computer): This concept is well known by control engineers. The introduction of digital processors has pushed away this concept at least from the realization level of computer control systems.

Dataflow System

A dataflow language has no concepts of variables and explicit control flow: a program is a set of data activated functions connected by unidirectional data paths. In a dataflow program the order of function executions is not defined explicitely (by the control structure of a procedural program), but derived implicitely from the dataflow that is related to the computation (Ackermann, 1982; Agerwala, 1982; Davis, 1982; Dennis, 1979; Treleaven, 1982).

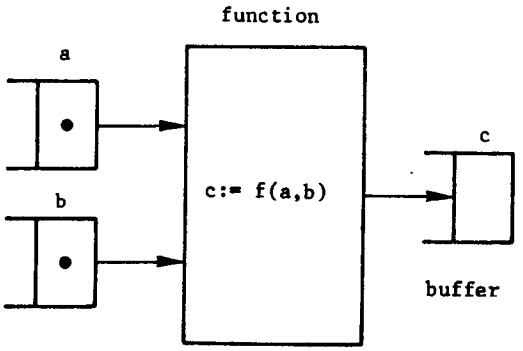

Fig. 1: Element of a dataflow program
The dots (a and b present, c absent) represent the condition which allow firing of the operation f

Fig. 1 shows an element of a dataflow program, where an output c is computed by execution of a function f(a,b). The execution of a function consumes all input values and produces an output value. A function

is executed only if all input values exist (have been produced) and the output is empty (has been consumed). This execution condition is called firing rule. A dataflow program is defined by interconnecting several functions.

Function interconnections can be considered as buffers. By writing a result value a buffer content is produced and by reading a buffer its content is consumed. Function executions are controlled by the associated buffers, i.e. function execution is data driven.

Continuous System

In a continuous system an output funtion is continuously computed from an input function. As long as the boundary frequency of the operator is sufficiently high compared to the relevant frequencies of the signals, a continuous system is an ideal tool for control and simulation purposes. It is a natural extension of a physical process.

A continuous system can be regarded as a dataflow system with continuous firing or in a first approximation, with periodic firing with a sufficiently small period (Fig. 2).

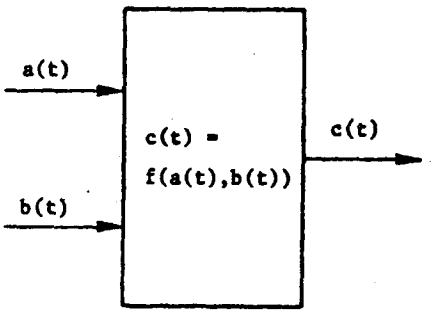

Fig. 2: Continuous System

Quasi-Continuous Systems

Some modern microcomputer-based distributed control systems can be considered as quasi-continuous systems. In such systems continuous control functions are realized approximately by algorithms and executed on microcomputer-based controllers in sufficiently short cycles. There is usually no need for a strict producer/consumer relationship of values, as it is found in dataflow systems. Rather, function executions are normally time cycle driven and function results update previously generated values. That is, functions communicate via (memory) cells: by writing a new value into a cell the cell content is updated, by reading a cell the cell content is not destroyed (Fig. 3).

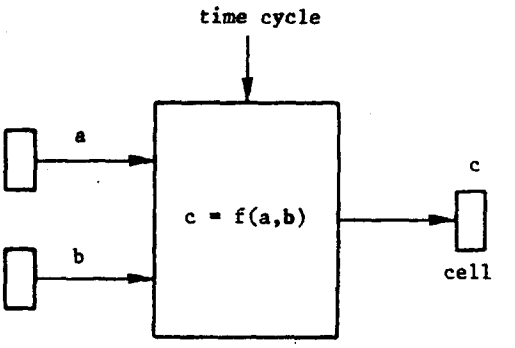

Fig. 3: Quasi-continuous system

Examples

For the sake of illustrating the application of a (pure) dataflow machine and a quasi-continuous system let's have a short look at two typical applications:

1) bank transaction
A bank transaction can be implemented by a dataflow program with the implicit firing rule: the input values must be present, they are consumed and produce an output value. The operation has to be executed exactly once, it may not be repeated nor left out (Fig. 4).

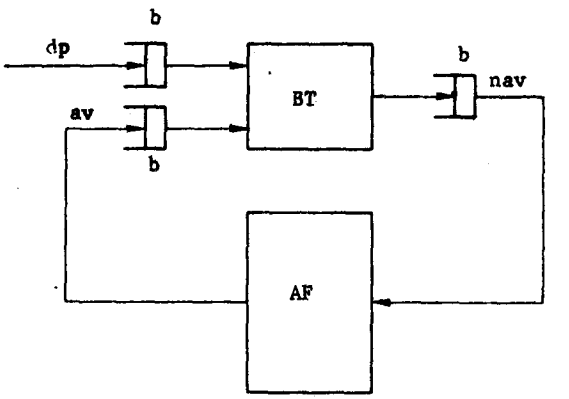

Fig. 4: Symbolic representation of a bank transaction as a dataflow program.

BT = bank transaction
AF = access functions of the account data base
b = buffer
dp = deposit
av = account value
nav = new account value

2) process control
A PID control algorithm is a typical example of an operation, which can be executed quasi-continuously (or as often as possible, using an appropriate integration and differentiation method). (Fig. 5)

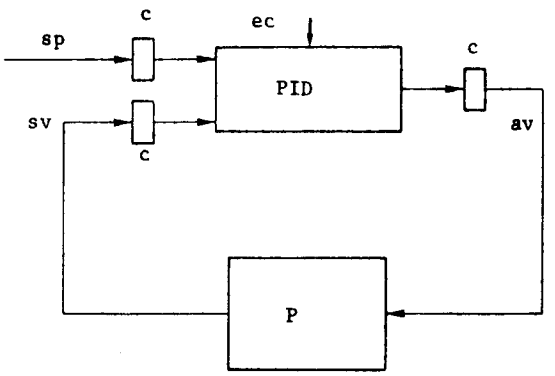

Fig. 5: Representation of a control function in a quasi-continuous system

PID = PID control function
P = physical process
c = cell
sp = set point
ec = execution cycle
sv = sensor value
av = actuators value

BROADCAST SYSTEMS

Motivation

In the following we introduce a type of distributed computer system that has some features in common with dataflow systems. We call these systems Broadcast Systems.

The motivation for broadcast systems is to provide an operational principle that facilitates the construction/configuration of large software systems from predefined building blocks (programming in the large). The building blocks are assumed to be represented by program modules that are internally implemented in procedural high-level languages (programming in the small).

The building blocks from which large programs are constructed, are expected to be maintained in a module catalogue. The user is free to enhance the catalogue by user-defined module types. Modules can represent simple functions, type managers (abstract data types), etc. (see Fig. 6). The catalogue provides module types from which instances are created and added to the already existing part of the software system (see Fig. 7). At program runtime the modules are equipped with work space and can be considered as processes, monitor processes, type managers, etc.

We should recall here the preconditions for an extensive use of predefined modules:

- The modules have to be independent from the environment, i.e. in particular from the hardware and software configuration.

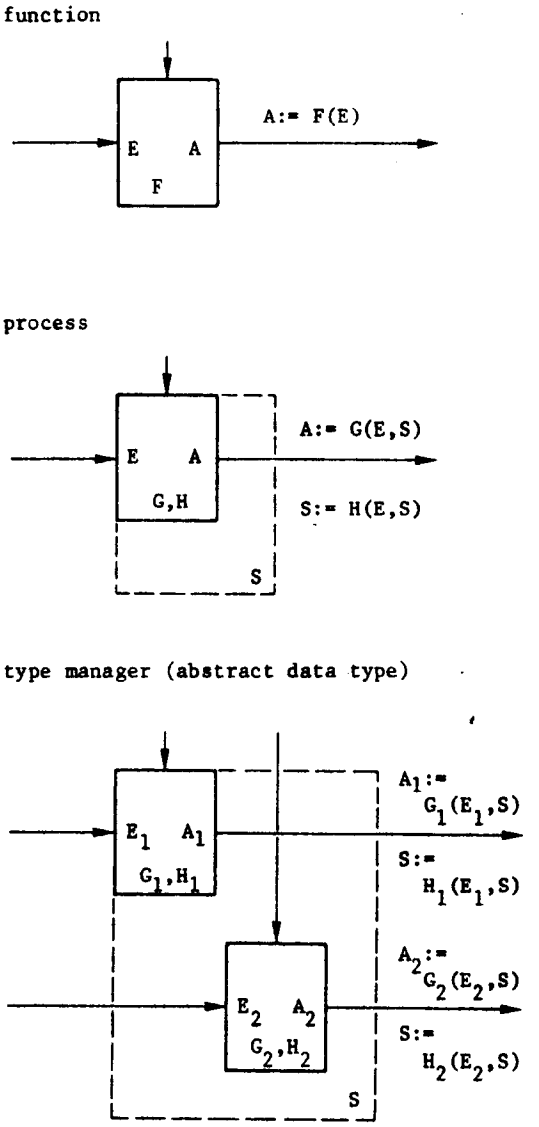

Fig. 6: Examples of program modules

E = input
A = output
S = internal state
G, H = functions

- A uniform framework has to be provided for the interconnection of modules.

The second precondition is a strong demand for appropriate computer architectures.

Basic Ideas

The basic concept of broadcast systems is a communication ether which provides all resources necessary for the transmission of values. All values have a unique source and are available wherever needed.

For the sake of simplicity we will restrict the following discussion to simple functions

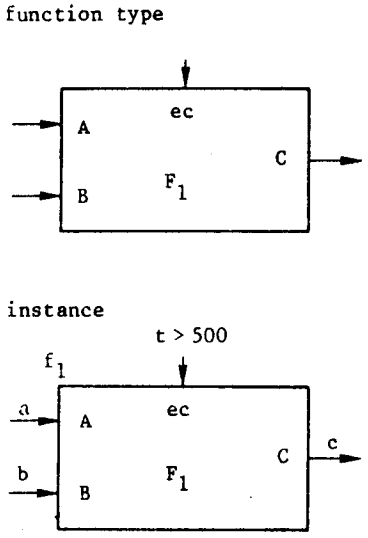

Fig. 7: Creation of an instance from a function type

F_1 = function type
A,B,C = formal parameters
f_1 = function instance
a,b,c = actual parameters
ec = execution condition

instead of general modules. To each function an execution condition is associated that defines the precondition for the execution of the function. The execution condition can be expressed in the form of a boolean expression on all data available in the ether including time. If and only if the execution condition is satisfied the associated function will be executed (Fig. 8).

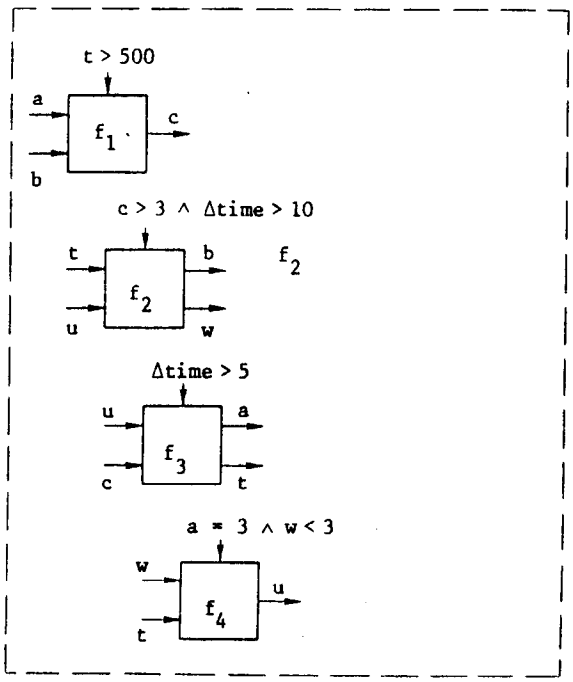

Fig. 8: Communication ether

Data, that are results of function executions, are broadcast in messages with source identification. Thus, data are globally accessible within the ether. All functions listen to transmitted messages and receive only those that are required for their inputs.

Let us summarize the advantages of that simple and robust operational principle for the interconnection and cooperation of functions and program modules:

- there are no side effects of function executions
  - a side effect is the change of a value that is not explicitely associated with an output parameter of a function
- there is no risk in having all values globally known
  - there is no need for scope rules and for passing access rights
- the approach facilitates the
  - incremental construction of software systems,
  - extension of running software systems
- there is a natural support of backward error tracing and the identification of fault sources by
  - source identification of values,
  - definition of execution conditions

Fig. 9 illustrates the extension of an existing program, given in Figure 8, by an additional function. Obviously, the introduction of the new function does not affect the original program.

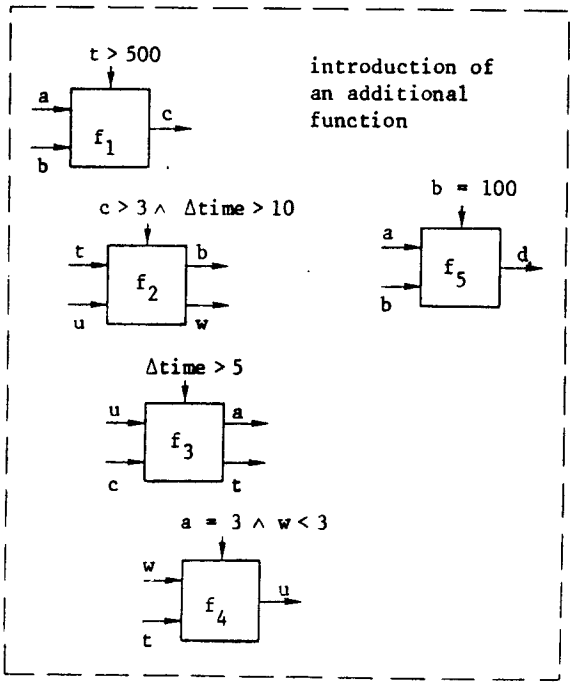

Fig. 9: Extension of the program given in Fig. 8

Let us introduce another model that is partly equivalent to the communication ether: the source addressed memory. The source addressed memory is similar to a conventional memory, with the exception that each cell is associated exclusively to one source, i.e. to one writer. A cell can be read by arbitrary functions of the program, using the appropriate source identification.

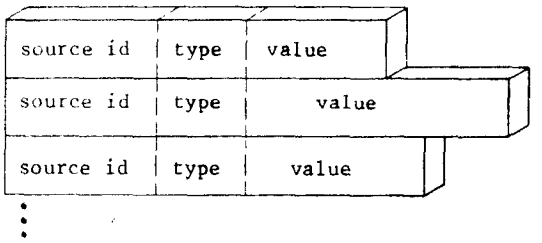

Fig. 10: Model of source-addressed memory

We present the model of source-addressed memory to emphasize the differences between function cooperation in the small and in the large. Conventional memories provide highly flexible and unrestricted means for storing and exchanging information. Since addresses can be created in unrestricted ways, cells of conventional memories can be changed from everywhere in the program system. Thus, side effects cannot be prevented and fault sources are difficult to localize. Features that should be tolerated only within small program units.

We feel that different mechanisms should be provided for programming in the small and software function interconnection in the large.

Realization of Broadcast Systems

A communication ether can be realized straightforwardly in distributed computer systems. Results of function execution are broadcast in messages with source identification. These messages are sent over a communication network and, thereby, are made available to all node computers connected to the network.

Within the node computers dedicated functional units perform particular operations on the communication ether:

- An associative receiver listens to all the messages on the network and copies only those data in its internal memory that are needed by the node computer
- An execution condition evaluator permanently evaluates the execution conditions of the functions installed on the node computer.
- A transmitter broadcasts the results of function executions. The broadcast mechanism supports consistent, simultaneous updates of memory cells at different places. The commitment of result values is atomic, i.e. either all receiving node computers perform an update or no node does.

The interconnected functions are executed on processors that are part of the node computers. Fig. 11 gives an impression of the internal structure and organization of the node computers.

We should point out that message broadcast allows effective use of the communication network because one 1:n transmission replaces n 1:1 (point to point) transmissions. The execution time overhead of associative message receiving and execution condition evaluation is kept small by the dedicated hardware units.

The key issue of the communication system is: how to establish acknowledge of broadcast messages. Since the number of receivers is both variable and unknown, the solution is the logical or of negative acknowledgements. That raises a second question: how to detect a dead receiver that cannot even send a negative acknowledge. That problem is solved by letting each node computer send its negative acknowledge signal regularly. This is then checked by one or more dedicated supervising units.

Programming Environment

As the control-engineer is used to formulate his problem in a way very similar to a data-flow program, he programs the system graphically, using off-site CAD-systems or on-site interactive programming stations, which become more and more efficient and economical.

It is also important to realize that it is possible to convert automatically and unambiguously a machine program into a graphic representation and vice-versa without the need of any additional, auxiliary graphical information.

- The module types are realized internally in procedural high level languages and compiled into machine code over micro code for high-performance functional units. This is the working field of the system specialist. It is "programming in the small", where the overall size of the program is limited, independent of all others and of course also application-independent. This allows to optimize implementations of the operations.
- The application system, which may be distributed over more than 10'000 functional units, is programmed by the application engineer in a non-procedural, possibly graphical language as described

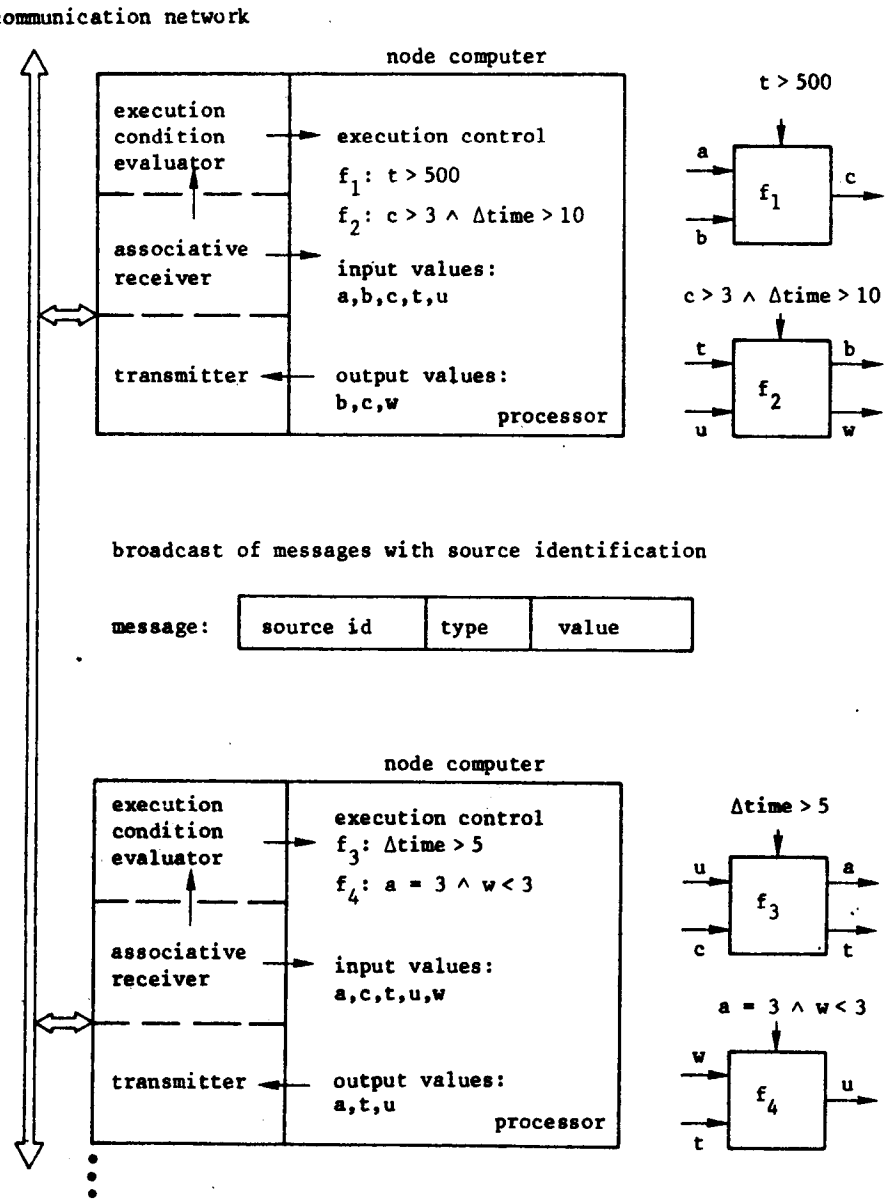

Fig. 11: Realization of broadcast systems

above ("programming in the large"). The principle of source-identification makes it possible that the distribution of the program over the physical modules (functional units) is totally transparent and can change, even while the system is running. This flexibility is very helpful, because it allows to write the application software before the hardware configuration is known. The advantage for reconfiguration and expansion of control system is obvious.

PERSPECTIVES

We have presented the concept of broadcast systems. Even in process control systems the full potential of the concept has not yet been totally used. Some additional and very precise work has to be done to prove the usefulness of the concept for a much wider range of applications. The field of process control, hopefully in a much wider range than up to now, will remain the main application of the concept at least for a certain time.

Corresponding research work is done and the results are expected to provide major impact for future systems.

PRESENT SYSTEMS

A consequent implementation of broadcast aspects has been realized in different products of Brown Boveri & Co., e.g. in the systems PROCONTROL, PARTNERBUS and BBC-Kent P4000. This is not the place to

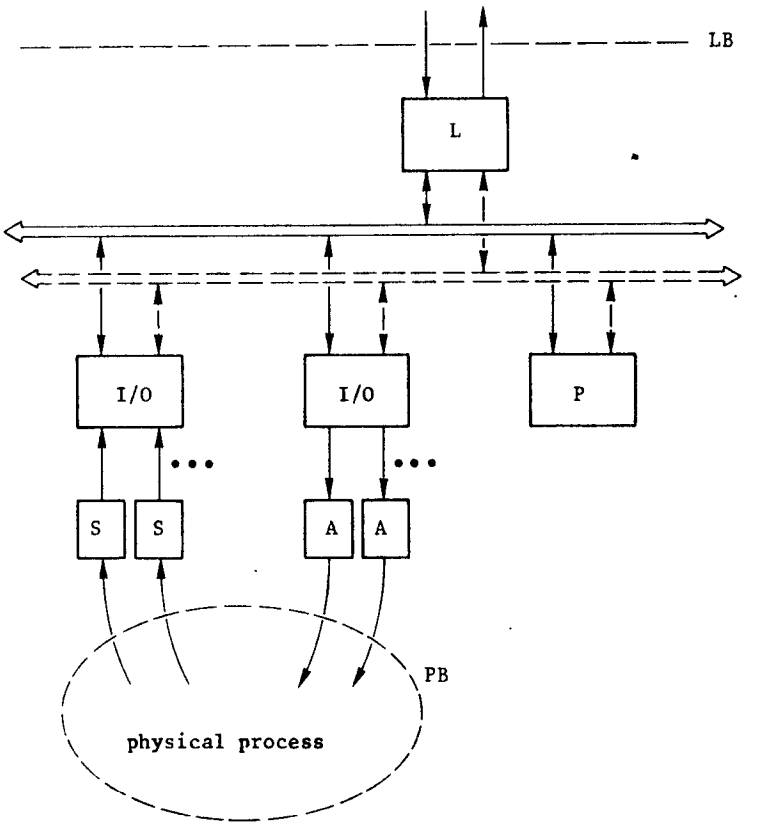

Fig. 12: Simplified view of the BBC-PROCONTROL System

S = Sensor
A = Actuator
I/O = Process Input and/or Output, including control and/or data processing functions
P = Processing Unit (Process Control, DDC, local man-process interface, local man-computer interface)
PB = physical boundary
LB = logical boundary between process and control system

give a detailled product description; there exists an appropriate documentation. As these products are mainly used by the companies' own application departments and because the advantages of the broadcast concepts are not expected to be widely known in the field of process control applications these systems are not explicitely known as broadcast systems. Nevertheless, their inherent advantage over seemingly similar systems is based on this concept.

The systems are specialized according to the hierarchical structure of a process control system.

The system PROCONTROL is optimized to handle process data and to communicate them to and from programmable controllers (representing node computers) and process data input/output equipment.

The communication ether realized by the system PROCONTROL has the effect of moving the logical boundary between physical process and computer control system into the computer control system (Fig. 12). This means that sensor values, independent of the location of their generation, are available wherever needed in the control system and that the actuator values, again independent of the module of their computation, will reach the appropriate actuators. All functions of the system are free of side effects; the system can be programmed without knowing how the functions will be distributed among the physical modules and where these modules will be placed. The system itself is structured to allow efficient solutions for systems ranging from relatively small to very large applications. (The largest system being installed contains more than 5'000 node computers.)

All modules of the system contain associative receivers and transmitters. The communication within the system as well as the execution of the functions is oriented towards efficient quasi-continuous behaviour: The signals are communicated whenever they have changed; the amount of change which triggers a broadcast can be selected and is

combined with minimal and maximal delays, so that a signal on one hand does not saturate the communication network and on the other hand is sent regularly, even if there was no or not sufficient change.

The communication protocol includes a common acknowledge of all connected modules as previously described.

The system includes some interesting features: it can be built up completely redundant without need of any specific programming; if the system is cut the separated parts immediately begin to work independently.

The PARTNERBUS system is oriented towards the communication of minicomputer-like and mainframe-like node computers. It implements the communication functions of the presented broadcast system, i.e. associative receiver and transmitter. The node computers can set up tables which define the signals that have to be received and the location in memory where this information has to be stored. They also define the names and values of the outgoing signals. The broadcast aspect can be restricted to the communication system to allow optimal modularity between nodes that internally are used conventionally. This allows even integration of a Partnerbus system into existing systems without changing their original operating systems (e.g. VAX/VMS*).

The BBC Kent P4000 System makes use of the broadcast concept to achieve a extremely flexible combination of packaged subsystems. This allows to tailor systems to the customer's need with minimal cost.

CONCLUSION

The concept of a broadcast architecture has some potential advantages over conventional systems. The practical exploitation of the concept has so far been believed to be restricted to a few special applications. In the field of process control the concept has been used successfully and corresponding products exist; a more general use is not unlikely in the the future.

ACKNOWLEDGEMENT

We want to express our gratitude to our colleagues in the research center, whose contributions were very helpful and constructive. In particular J. Kriz and S. Züger have directly contributed to this work and are involved in ongoing activities.

*) VAX and VMS are trademarks of DIGITAL EQUIPMENT CORPORATION

REFERENCES

Ackermann, W.B. (1982). Data Flow Languages. Computer, Vol. 15, No. 2, pp. 15-25.

Agerwala, T., Arvind. (1982). Data Flow Systems. Computer, Vol. 15, No. 2, pp. 10-13.

Davis, A.L., Keller, R.M. (1982). Data Flow Program Graphs. Computer, Vol. 15, No. 2, pp. 26-41.

Dennis, J.B. (1979). The Varieties of Data Flow Computers. Proc. 1st Int. Conf. Distributed Computing Systems, New York, IEEE, pp. 430-439.

Treleaven, P.C., Brownbrigde, D.R., Hopkins, R.P. (1982). Data-Driven and Demand-Driven Computer Architecture. ACM Computing Surveys, Vol. 14, No. 1, pp. 93-143.

DISCUSSION

Sloman: I can see a possible problem in using global names. Firstly, you appear to lose some abstraction, and secondly, you have a problem of managing your name-space. If you have to manage global source identifiers in a very large system, such as you refer to in your paper, it appears to be very difficult to choose names which are unique. How can this be overcome?

LaLive d'Epinay: This does not present any problem. The Power Systems people, for example, have a naming concept which is older than computer control systems, in which the name of a variable is uniquely defined by the geographical location in the building as well as the precise location of the place where the signal is created. In such a system there is no problem, since, from the beginning of the design, it is totally clear what the name of any variable is. They have a very large name-space in which they use only a very few possibilities: this is one of the reasons why the systems which we have developed have 24-bits of address space on the communication bus system. There are, however, possibly other areas of application where there is a problem. One of these arises in the case that I did not mention in my presentation, and that is the recognition of duplicate source identifications. This is typically built into the hardware and you can try out the name if you want, and the system will be able to tell you whether the name is already used or not. Of course the whole management of variable names and the names of signals on the cables has to be planned as part of the integral solution.

MacLeod: You have told us that in your approach there is a unique source for each piece of information. Does this not eliminate one of the advantages of distributed control systems, and that is redundancy? In other words, the ability to handle redundant senders of information in the event of a failure?

LaLive d'Epinay: This is also one of the problems which is solved by our solution. Let us say that you have redundant information. In this case there must be some knowledge of this fact. Let us say, for example, that a twin of the information exists. Recognition of this redundancy is built into the hardware - you can even have, for example, the typed-description of one of the products and you can see whether it is the original, whether it is a back-up or whether it is an engineer-supplied auxiliary value. For example, if the measurement of a temperature is not functioning you could take the redundant information as the temperature. You can recognize that in the data field, but, of course, you must know that twins or triplicates exist. Another aspect is that in a truly redundant system, the sensors are duplicated so you effectively have two different values right from the very beginning. They are treated differently. We can also have a redundant communication system which communicates every piece of information in a duplicate form so that we have the same information and the same source identification on both of the communication systems. It is undoubtedly a very complex problem which has to be solved on each level appropriately, but in our solution it can be approached in a very elegant fashion.