International Journal of Computer and Information Sciences, Vol. 14, No. 5, 1985

Towards Formal Specification of a Distributed Computing System

V. K. Agrawal,[1] L. M. Patnaik,[2] and P. S. Goel[1]

Received May 1985; revised September 1985

Onboard spacecraft computing system is a case of a functionally distributed system that requires continuous interaction among the nodes to control the operations at different nodes. A simple and reliable protocol is desired for such an application. This paper discusses a formal approach to specify the computing system with respect to some important issues encountered in the design and development of a protocol for the onboard distributed system. The issues considered in this paper are concurrency, exclusiveness and sequencing relationships among the various processes at different nodes. A 6-tuple model is developed for the precise specification of the system. The model also enables us to check the consistency of specification and deadlock caused due to improper specification. An example is given to illustrate the use of the proposed methodology for a typical spacecraft configuration. Although the theory is motivated by a specific application the same may be applied to other distributed computing system such as those encountered in process control industries, power plant control and other similar environments.

KEY WORDS: Distributed computing systems; protocol design; onboard computers; formal specification.

1. INTRODUCTION

In a distributed computing system, as the complexity of the system increases with the number of nodes and need for continuous interaction among the nodes; there is a necessity for systematic communication procedures which every node of the distributed system must follow to have an error-free and efficient communication among the various nodes. The

[1] Control Systems Division ISRO Satellite Centre, Airport Road, Bangalore-560017, India.
[2] Indian Institute of Science Bangalore-560012, India.

0091-7036/85/0100-0277$04.50/0

distributed computing system considered in this paper is the one encountered in the onboard system of a spacecraft.[1] Such a system requires a simple protocol for a reliable communication among its processing nodes. This application being real-time in nature,[2] its protocol has to satisfy the requirements of promptness, robustness, correctness and flexibility.[3,4] Additionally, spacecraft application calls for high reliability and reduced weight, volume and power. Protocols applicable to a ground-based distributed computing system may not be suitable for onboard applications because of these additional constraints.

Although a spacecraft computing system falls under the category of loosely coupled system, in the sense that each of the nodes has its own processor, memory, peripherals etc., and communicates with other nodes via I/0 channels. The functions performed by each of the processors at a particular node are not independent of those performed at other nodes.[2] Nodes are distinguished by their functions but are not fully autonomous. Most of the time the interaction among these nodes is limited to the exchange of only a few words of information, unlike the other cases where more frequent flow of information among the various nodes is a common feature.

As the sophistication of the onboard system grows, the protocol demands complex interaction among the different computing nodes. Most of the existing protocols are designed for ground-based computer network and they meet some of the earlier requirements of the onboard system. But, it is difficult to provide the required hardware/software interfaces, for the implementation of these protocols, due to the limitations on weight, volume, and power of the onboard system. Therefore, the existing protocols are not considered to be ideally suitable for the spacecraft application and the task of design and development of a simple and easy-to-implement protocol for such an application is a significant one.

The design and development of a protocol may be divided into five phases; (i) the specification of the system for which the protocol is to be designed; (ii) protocol requirements; (iii) protocol design/synthesis; (iv) unambiguous specification, and validation of the designed protocol and finally; (v) hardware implementation and performance evaluation of the protocol. This paper is concerned with the first of the these five phases of the design and development of a protocol.

Most of the research efforts in the area of distributed system pertain to the control mechanism such as task allocation,[5] load balancing,[6] synchronization,[7] concurrency control,[8] etc. Comparatively less efforts have been made in the area of formal specification/description of a distributed computing system. Yau and Yang[9] have described the need for such specification/description methodologies for the software design of dis-

tributed computing systems. Using attributed grammer, Lu and Yau[10] have discussed an approach for this problem. Both these research efforts being meant for software design, do not consider the problem of protocols. However, they deal with the specification of features such as concurrency, sequencing and synchronization. For the protocol design application these features need to be specified in a more detailed manner as brought out in this paper. In view of this, we consider here a methodology to specify various detailed aspects (discussed later) of the important features such as concurrency, exclusiveness, and sequencing. The methodology presented here is simple to use and can be useful in other applications such as process control, etc.

After describing the motivation of the specification methodology in the next section, we present in Section 3 the definitions and properties of various terms needed. In Section 4, we develop a 6-tuple model to specify a distributed computing system and methods are given to extract, from the model, the various properties of Section 3. It is shown that consistency of the specification and deadlock caused due to improper specification can be checked.

2. MOTIVATION

The method of specification of a distributed computing system depends on the extent and type of details to be specified. Lu and Yau[10] have developed a specification methodology for software design of distributed computing systems, using attributed grammer. Though it may be possible to extend this approach to the specification of distributed systems for protocol design, it is difficult to follow such an approach; because of the complexity involved in the approach, when there are large number of processing nodes and processes assigned to these nodes. Even for the software design, the approach does not seem to be simple when there are large number of processes with complex interaction present among them. Also, in this approach, it is possible to check the consistency of design and implementation of the system, but this requires a special purpose programming language that recognizes attributed grammer. However, this work does not contain a method for checking the consistency of the specification. One should consider the following two aspects in much greater detail while using the formal specification method for protocol design.

First is the inter-process relationship so that the protocol can accept or reject the request of process execution. The typical attributes or relationship to be considered are concurrency, exclusiveness and sequencing among processes. Concurrency and exclusiveness are not two entirely independent attributes. Also we do not consider concurrency in the sense

that concurrent processes have to always run in an overlapping manner and for our discussion we consider two classes of concurrency.

Second important aspect is that the attributes described previously are to be stored at all the nodes in some form or other to perform the protocol operations. Our specification methodology ensures easy computation of the different attributes from the information stored at each node.

Like any other case, in the application considered here it is necessary to check the "consistency" of the specification i.e., whether there is any conflicting specification among the various attributes. Also the most common problem associated with a distributed computing system is that of deadlock, especially when sequences of execution of the processes are to be maintained. There should be an easy way of detecting deadlocks arising out of improper sequencing specification.

Considering these aspects of specification requirements, the approach based on formal language and graph theory are not very attractive because of their storage and computational complexities. Instead, we follow a set theoretic/matrix based approach for the specification which is more suitable from these two aspects. Further, it is easy to check consistency and deadlock in our approach.[11]

In the next two sections, we present our specification methodology. First we formally define, using the set theoretic approach, the various terms needed to specify the system with respect to the attributes discussed and then, present a model to specify the system using a matrix-based approach.

3. BACKGROUND FOR THE SPECIFICATION METHODOLOGY

As mentioned earlier, the first phase of the design and development of a protocol is the specification of the system for which the protocol is to be designed. Lesser the ambiguities in the system specification, more is the correctness of the design. It is with this intention that we present a formal approach which specifies the distributed computing system with respect to interprocess relationship such as concurrency, exclusiveness and sequencing attributes discussed earlier. We refer the term "process" as the terminal process which cannot be subdivided further into smaller sub-processes. Any nonterminal process is a collection of such terminal processes. Each of the nodes of the computing system is assigned a set of terminal processes, hereafter referred simply as process. At any given time, a node can run a subset of these assigned processes.

Let SY be a distributed computing system under consideration, and let it be defined by a set of its nodes. It is given by

$$SY = \{N_1, N_2, ..., N_n\}$$

where N_i represents the ith system or node of the computing system and n is the total number of nodes in the system.

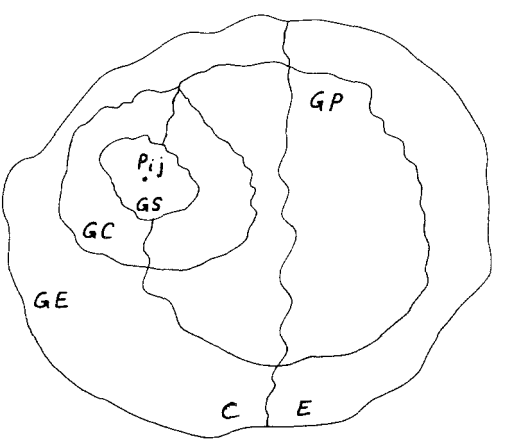
Each of the node N_i is assigned a set of processes and the node is represented by

$$N_i = \{P_{i1}, P_{i2}, \ldots, P_{in_i}\}$$

where P_{ij}'s are the processes and n_i is the total number of such processes assigned to node N_i. Such an assignment is done at a stage before the protocol is designed.

In order to specify the distributed computing system with respect to the various attributes described earlier, we classify the processes into various groups. This classification can be seen from the Venn diagram given in Fig. 1. Formal definitions, their properties and interrelationships of the defined terms are given by the following.

The set of processes in the system is divided into the sets of concurrent (C) and exclusive (E) processes. The set of *concurrent processes* (C) con-



C = SET OF CONCURRENT PROCESSES.

E = SET OF EXCLUSIVE PROCESSES.

GC = SET OF CONCURRENT PROCESSES OF Pij.

GE = SET OF EXCLUSIVE PROCESSES OF Pij.

GS = SET OF STRONG CONCURRENT PROCESSES OF Pij.

GP = SET OF PREORDER PROCESSES OF Pij.

Fig. 1. Classification of processes.

tains all those processes in which any one process can run concurrently with at least another processes in the set. The set of *exclusive processes (E)* contains those processes which cannot run concurrently with any one of the processes in the set. Therefore, we have

$$S = C \cup E$$

where C = set of concurrent processes

$$\triangleq \{C_1, C_2, ..., C_n\}$$

E = set of exclusive processes

$$\triangleq \{E_1, E_2, ..., E_m\}$$

$$\text{and} \quad \left.\begin{matrix} C_i \subseteq N_i \\ E_i \subseteq N_i \end{matrix}\right\} \text{such that } C_i \cup E_i = N_i \quad \text{and} \quad C_i \cap E_i = \varnothing$$

At any instant t, a set of processes being executed in the system is given by

$$R(t) \subseteq S$$

such that

$$(P_{ij} \in R(t)) \wedge (P_{ij} \in E) \rightarrow R(t) = \{P_{ij}\}$$

and

$$(P_{ij} \in C) \rightarrow R(t) \subseteq C$$

Now we define some terms that are used to describe the relationship among the processes in the computing system. Definitions are followed by some simple propositions. Some of these propositions are useful in verifying the consistency of the specifications and for detecting deadlocks arising due to improper sequencing specification.

Definition 1. Global Concurrent (GC) Processes: A process $P_{lm} \in S$ is said to be global concurrent with another process $P_{ij} \in S$, iff both P_{ij} and P_{lm} can run concurrently for a nonzero overlapping period, i.e., $\exists t \mid (P_{ij}, P_{lm} \in R(t))$. The global concurrency between P_{ij} and P_{lm} is represented by

$$P_{ij} \widehat{GC} P_{lm}$$

We give the following propositions based on this definition. The proofs of these propositions are simple and therefore are not given here.

Proposition 1.1. Relation $\widehat{GC}$ is commutative.

Proposition 1.2. $P_{ij}\,\widehat{GC}\,P_{lm} \rightarrow P_{ij}, P_{lm} \in C$

Proposition 1.3. $P_{ij}\,\widehat{GC}\,P_{ij}, \forall P_{ij} \in S.$

Proposition 1.4. $(P_{ij} \in E) \wedge (P_{ij}\,\widehat{GC}\,P_{lm})$

$$\rightarrow ((i = l) \wedge (j = m)).$$

Definition 2. Global Exclusive (GE) Process: A process $P_{lm} \in S$ is said to be global exclusive with another process $P_{ij} \in S$, iff $\sim \exists t \,|\, (P_{ij}, P_{lm} \in R(t))$. It is represented by

$$P_{ij}\,\widehat{\mathrm{GE}}\,P_{lm}.$$

Proposition 2.1. Relation $\widehat{\mathrm{GE}}$ is commutative.

Proposition 2.2. $P_{ij} \in E \rightarrow P_{ij}\,\widehat{\mathrm{GE}}\,P_{lm}$

$$\forall P_{lm} \in S \qquad \text{and} \qquad ((l \neq i) \vee (m \neq j))$$

Proposition 2.3. $P_{ij} \in C \rightarrow P_{ij}\,\widehat{\mathrm{GE}}\,P_{lm}, \forall P_{lm} \in E$

Proposition 2.4. $P_{ij}\,\widehat{\mathrm{GC}}\,P_{lm} \leftrightarrow \sim(P_{ij}\,\widehat{\mathrm{GE}}\,P_{lm})$

Based on these two definitions, the corresponding sets of the processes are defined by the following:

Definition 3a. Global Concurrency (GC) Set.
GC set of a process $P_{ij} \in S$ is given by

$$\mathrm{GC}(P_{ij}) = \{P_{lm} \,|\, (P_{lm} \in S) \wedge (P_{ij}\,\widehat{\mathrm{GC}}\,P_{lm})\}.$$

Definition 3b. Global Exclusive (GE) Set.
GE set of a process $P_{ij} \in S$ given by

$$\mathrm{GE}(P_{ij}) = \{P_{lm} \,|\, (P_{lm} \in S) \wedge (P_{ij}\,\widehat{\mathrm{GE}}\,P_{lm})\}.$$

Proposition 3.1. $P_{ij} \in C \rightarrow |\mathrm{GC}(P_{ij})| > 1$

$$\text{and} \quad P_{ij} \in E \rightarrow |G(P_{ij})| = 1.$$

Proposition 3.2. $\bigcup_{P_{ij} \in S} \mathrm{GC}(P_{ij}) = S$

Proposition 3.3. $\mathrm{GE}(P_{ij}) = \{S - \{P_{ij}\}\}, \forall P_{ij} \in E$

$$\supseteq E, \forall P_{ij} \in C.$$

Proposition 3.4. If $(P_{lm} \in \mathrm{GC}(P_{ij}) \wedge ((l \neq i)\, \forall (m \neq i))$ then

i) $\mathrm{GE}(P_{lm}) \subseteq \{S - \{P_{ij}, P_{lm}\}\}$

ii) $|\mathrm{GC}(P_{lm})| \geqslant 2$

Proposition 3.5. $\mathrm{GC}(P_{ij}) \cup \mathrm{GE}(P_{ij}) = S, \forall P_{ij} \in S.$

The definition of concurrency does not impose the restriction that the concurrent processes have to run concurrently whenever they are invoked but they can run concurrently if the system demands. However, a process may have associated itself another set of processes, in which all the processes run concurrently whenever it is invoked. To characterise this, we define an additional class of concurrency which is a restricted form of the concurrency discussed previously.

Definition 4. Global Strong Concurrency (GS) Set of a process $P_{ij} \in S$ is given by

$$\mathrm{GS}(P_{ij}) = \{P_{lm} \mid (P_{lm} \in S) \wedge (P_{ij} \in R(t) \rightarrow P_{lm} \in R(t)\}.$$

Proposition 4.1. $\mathrm{GS}(P_{ij}) \subseteq \mathrm{GC}(P_{ij})$

Proposition 4.2. $\mathrm{GS}(P_{ij}) = \{P_{ij}\}, \forall P_{ij} \in E$

Proposition 4.3. $|\mathrm{GS}(P_{ij})| \geqslant 1, \forall P_{ij} \in C$

$$= 1, P_{ij} \in E$$

These definitions are used to specify the concurrent and exclusive relationship of the processes. We now define the terms used to specify sequencing attribute of the process. For our discussion, sequencing of processes is with respect to the initiation of the processes and we are not concerned with the termination of the processes. We also discuss the condition for the existence of deadlock arising due to sequencing specifications.

Definition 5. Global Preorder (GP) process: A process $P_{lm} \in S$ is said to be a global preorder process of $P_{ij} \in S$; if the execution of P_{lm} precedes the execution of P_{ij}. It is represented by

$$P_{lm} \widehat{\mathrm{GP}}\, P_{ij}$$

Proposition 5.1. The relation GP is non commutative.

Proposition 5.2.

$$P_{lm} \widehat{\mathrm{GP}} P_{ij} \rightarrow \exists t_1, t_2 (t_1 < t_2) \text{ such that}$$

i) $P_{lm} \in R(t_1)$ and $P_{ij} \in R(t_2)$

ii) $P_{ij} \in R(t)$ for $t_1 \leqslant t < t_2$

iii) $P_{lm} \notin R(t)$ for $t = t_1^-$

where t_1^- represents the time instant just prior to t_1.

Definition 6. Global Preorder (GP) set.
It is given by

$$\mathrm{GP}(P_{ij}) = \{P_{lm} \mid P_{lm} \widehat{\mathrm{GP}} P_{ij}, \forall P_{lm} \in S\}$$

Proposition 6.1.

$$P_{lm} \in \mathrm{GP}(P_{ij}) \rightarrow P_{ij} \notin \mathrm{GS}(P_{lm})$$

Proposition 6.2. $(P_{lm} \in \mathrm{GP}(P_{ij})) \wedge (P_{lm} \in \mathrm{GE}(P_{ij}))$

$\rightarrow P_{lm}$ completes execution before P_{ij} starts execution.

Proposition 6.3. If there exists a sequence of processes

$$P_1, P_2, \ldots, P_n (P_i \in S, 1 \leqslant i \leqslant n)$$

then

$$P_i \widehat{\mathrm{GP}} P_{i+1}, 1 \leqslant i \leqslant n-1$$

Proposition 6.4. $P_{ij} \in \mathrm{GP}(P_{lm})$ and $P_{lm} \in \mathrm{GP}(P_{ij})$

$\rightarrow$ Processes P_{ij} and P_{lm} are in deadlock.

To consider deadlock among n-processes, we define the following additional terms.

Definition 7. Global nth Preorder Set of a process P_{ij} is defined as

$$\mathrm{GP}^n(P_{ij}) = \{P_{lm} \mid P_{lm} \in \mathrm{GP}(P_{pq}) \wedge P_{pq} \in \mathrm{GP}^{n-1}(P_{ij})\}$$

for $n \geqslant 2$.

Where, $\mathrm{GP}^1(P_{ij}) = \mathrm{GP}(P_{ij})$.

Definition 8. Global Cyclic order: Processes P_{ij} and P_{lm} are said to be in cyclic order if $\exists n_1, n_2 \in \mathrm{IN}$ (IN is the set of nonzero positive integers) such that

$$P_{lm} \in \mathrm{GP}^{n1}(P_{ij}) \qquad \text{and} \qquad P_{ij} \in \mathrm{GP}^{n2}(P_{lm})$$

Proposition 8.1. There exists a deadlock between two processes iff they are in a cyclic order.

These definitions are given with respect to the entire computing system and so is the adjective "Global" for each of the terms. The definition can be restricted to span over only a single node. In such a case, the adjective "Global" is changed to "local" and the definitions are modified accordingly. For example, Definition 1 is changed to local concurrent (LC) process and is defined for the ith node as follows.

A process $P_{im} \in N_i$ is said to be local concurrent (LC) to $P_{ij} \in N_i$ iff $\exists t \mid (P_{ij}, P_{im} \in R_i(t))$ and this relationship is represented by

$$P_{ij} \widehat{\mathrm{LC}} P_{im}$$

4. A MODEL FOR A DISTRIBUTED COMPUTING SYSTEM

Enumerating all the sets defined in the previous section is one way of describing a distributed computing system. When the distributed system has sufficiently large number of nodes, such an approach may lead to a conflicting or incorrect specification. In this section, we present a model for specifying a distributed system. The approach is a matrix-based one and various sets presented earlier can be easily derived from this model. Also it is easy to check the consistency of the specification and the presence of deadlocks.

4.1. The System Model

The distributed computing system (DCS) is given by a 6-tuple as follows

$$\mathrm{DC} = \langle N, P, F, G, ST, Pr \rangle$$

where the various terms in the model are defined as follows:

N = set of nodes in the distributed computing system
$= \{N_1, N_2, ..., N_n\}$ where n is the total number of nodes in the system.

P = set of processes in the entire computing system
$= \{P_1, P_2, ..., P_m\}$ where m is the total number of processes in the system.

F is a function which specifies the processes assigned to different nodes of the system and is given by

$F: N \rightarrow \mathrm{IP}(P)$, where $\mathrm{IP}(P)$ denotes the power set of P.

The function F can be represented by an $n \times m$ binary matrix, called F matrix, in which the n rows correspond to the nodes in the system and the m columns correspond to the processes in the system.

The F matrix is given by

$$F = [f_{ij}]_{n \times m} \qquad \text{where} \quad f_{ij} \in \{0, 1\}$$

such that $f_{ij} = 1$ if P_j is assigned to node N_i
$= 0$ otherwise

G is a function that gives the set of processes which can run concurrently with a given process. It is given by

$G: P \rightarrow \mathrm{IP}(P)$ such that

$$P_i \in G(P_j) \rightarrow P_i \widehat{\mathrm{GC}}\, P_j$$

It may be noted that $G(P_i) = \mathrm{GC}(P_i)$. Further, G can be represented by an $m \times m$ binary matrix, called a G matrix, in which the rows and columns correspond to the processes in the system. The G matrix is given by

$$G = [g_{ij}]_{m \times m} \text{ where } g_{ij} \in \{0, 1\}$$

such that $g_{ij} = 1$ if $P_j \in \mathrm{GC}(p_i)$
$= 0$ otherwise

The G matrix is a symmetric matrix with diagonal elements equal to '1' (Proposition 1.3).

ST is a function that specifies the strong concurrent set of a process in the computing system. It is expressed by

$$ST: P \rightarrow \mathrm{IP}(P) \text{ such that } ST(P_i) = \mathrm{GS}(P_i).$$

The function ST can be represented by an ST matrix which is given as follows:

$$ST = [s_{ij}]_{m \times m} \text{ where } s_{ij} \in \{0, 1\}$$

such that

$$s_{ij} = 1 \text{ if } P_j \in \mathrm{GS}(P_i)$$
$$= 0 \text{ otherwise}$$

Pr is a function that specifies the set of preorder processes for each of the processes in the computing system. It is given by the following:

$$Pr: P \rightarrow \mathrm{IP}(P), \text{ such that } Pr(P_i) = \mathrm{GP}(P_i).$$

The corresponding Pr matrix is given by

$$Pr = [p_{ij}]_{m \times m} \text{ where } p_{ij} \in \{0, 1\}$$

such that

$$p_{ij} = 1 \text{ if } P_j \widehat{\mathrm{GP}} P_i$$
$$= 0 \text{ otherwise}$$

4.2. Matrix Operations

We define certain operations on all the binary matrices. These operations are essential for the subsequent discussions.

i) *Multiplication*:
Let A and B are two binary matrices given by

$$A = [a_{ij}]_{m \times n}$$
$$B = [b_{ij}]_{n \times q}$$

where $a_{ij}, b_{ij} \in \{0, 1\}$.

The product of matrices A and B is given by

$$C = A \circledast B \text{ where } C = [c_{ij}]_{m \times q}$$

such that

$$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \cdots + a_{im} \cdot b_{mj}$$

The operations $a \cdot b$ and $a + b$ are the logical AND and logical OR operations respectively on a and b.

ii) *Transpose*: The transpose of a binary matrix is in the same sense used for any general matrix.

iii) *Projection*: Matrix A ($[a_{ij}]_{p \times q}$) can be projected over a matrix B ($[b_{ij}]_{m \times n}$) with the element a_{11} of matrix A coinciding with the element b_{rs} of matrix B if $p \leqslant m$, $q \leqslant n$, $1 \leqslant r \leqslant (m - p + 1)$, and $1 \leqslant s \leqslant (n - q + 1)$. The resulting matrix obtained after the projection of A over B is another matrix C_{rs} ($[c_{ij}]_{m \times n}$) whose elements are defined as follows.

$$C_{rs} = \mathrm{Proj}(A, B) = [c_{ij}]_{m \times n}$$

such that

$$c_{ij} = \begin{cases} b_{ij}, \text{ for } 1 \leqslant i < r \text{ and } 1 \leqslant j < s \\ (a_{(i-r+1),(j-s+1)}) \cdot (b_{ij}), \\ \quad \text{for } r \leqslant i < (r+p) \text{ and } s \leqslant j < (s+q) \\ b_{ij}, \text{ for } (r+p) \leqslant i \leqslant m \\ \quad \text{and } (s+q) \leqslant j \leqslant n \end{cases}$$

For the sake of simplicity, when the matrix A is projected over matrix B with the element a_{11} coinciding with the element b_{11}, the resulting matrix C_{11} is written simply as C.

Considering a special case where

$$A = [a_{ij}]_{p \times q}$$

and

$$B = [b_{ij}]_{p \times q}$$

we have

$$C = \text{Proj}(A, B) = \text{Proj}(B, A)$$
$$= [c_{ij}]_{p \times q}$$

such that

$$c_{ij} = 1 \text{ if } a_{ij} = b_{ij} = 1$$
$$= 0 \text{ otherwise}$$

iv) *Complementation*: The complementation operation of a matrix A flips all the '1' elements of A to '0' and all the '0' elements to '1'. It is a logical inversion of the elements of the matrix A. The complementation operation of a matrix A is represented by $\text{Comp}(A)$.

4.3. Computation of Attributes

In this subsection, we demonstrate that from a given model of a distributed computing system we can generate the various sets characterising the concurrent, exclusive and sequencing relationship of the processes described earlier.

4.3.1. Computation GC and GE Sets

We can compute the GC of a process $P_i \in P$ from the given G matrix, as follows:

$$\mathrm{GC}(P_i) = \{P_j \mid g_{ij} = 1\}$$

Also using Proposition 3.5, the GE set of process P_i is given by

$$\mathrm{GE}(P_i) = \{P - \mathrm{GC}(P_i)\}$$

In the matrix representation, global exclusiveness is represented by a matrix G' given by

$$G' = \mathrm{Comp}(G) \text{ where } G \circledast G' \text{ is a null matrix}$$

The following lemma and theorem present a method to compute the set of nodes in which each node has at least one process that is *exclusive* to a specific process in the system. The knowledge of this set makes the protocol operation faster as in such an operation it is required to check the existence of an *exclusive process* running in the system before starting execution of a new process.

Lemma. An element h_{kl} of a matrix $F \circledast G'$ is '1' iff $\exists i$ such that f_{ki} and g'_{il} are equal to '1', where f_{ki} and g'_{il} are the elements of F and G' matrix respectively.

Proof. Referring to the definition of the multiplication of two matrices, an element h_{kl} of $F \circledast G'$ is given by

$$h_{kl} = \sum_{i=1}^{m} [f_{ki} \cdot g'_{il}]$$

where $\sum[K]$ represents a logical ORing.

Hence, it is clear that $h_{kl} = 1$ iff $\exists i$ such that $f_{ki} = 1$ and $g'_{il} = 1$. ∎

Theorem 1. There exists a process $P_i \in F(N_k)$ which is global exclusive with another process $P_l \in P$, iff the (k, l) element of the matrix $F \circledast G'$ is equal to '1'.

Proof. The proof directly follows from the previous lemma. We have from the definitions of F and G' matrices.

$$f_{ki} \text{ (an element of the } F \text{ Matrix)}$$

$$= 1 \quad \text{iff } P_i \in F(N_k)$$

$$g'_{il} \text{ (an element of the } G' \text{ matrix)}$$

$$= 1 \quad \text{iff } P_i \widehat{\mathrm{GE}} P_l$$

From the lemma,

$$h_{kl} \text{ (an element of } F \circledast G') = 1 \quad \text{iff}$$
$$f_{ki} = 1 \quad \text{and} \quad g'_{il} = 1.$$

Hence the theorem.

Corollary 1.1. For each set of nodes that has at least one process exclusive to a process $P_l \in P$ is equal to

$$\{N_k \mid h_{kl} = 1\} \quad \text{for } 1 \leqslant k \leqslant n$$

Proof. The proof is omitted.

We now present a methodology to compute local concurrency and local exclusive sets of a process. This information is essential to enable the protocol to take decision with respect to the local environment.

4.3.2. *Computation of Local Concurrent (LC) and Local Exclusive (LE) Sets*

The LC set can be obtained in a manner similar to that of the GC set. For this, first we compute from the model an L matrix which is defined by

$$L = [l_{ij}]_{m \times m}$$

where $l_{ij} = 1$ iff

(i) $P_i \widehat{\text{LC}} P_j$

(ii) $P_i, P_j \in P$

(iii) $P_i, P_j \in F(N_k), N_k \in N$

Theorem 2. The L matrix is a matrix obtained by the projection of $(F^T \circledast F)$ over G i.e.,

$$L = \text{Proj}(F^T \circledast F, G)$$

Proof. Without loss of generality, we assume that

$$F(N_i) = \{P_{X(i)+1}, P_{X(i)+2}, \ldots, P_{X(i)+K(i)}\}$$

where

$$X(i) = \sum_{j=1}^{i-1} K(j), \text{ for } 2 \leqslant i \leqslant n$$
$$X(1) = 0$$

and

$$K(j) = |F(N_j)|$$

Then we have the F matrix as follows

$$F = n \begin{bmatrix} \underbrace{K(1)} & \underbrace{K(2)} & & \underbrace{K(n)} \\ J_1 & & & \\ & J_2 & & \\ & & \cdots & J_n \end{bmatrix}$$

where all the elements of J_i's are equal to '1'.

Then $F^T \circledast F$ is given by

$$F^T \circledast F = \begin{matrix} \\ K(1) \\ K(2) \\ \\ K(n) \end{matrix} \begin{bmatrix} \underbrace{K(1)} & \underbrace{K(2)} & & \underbrace{K(n)} \\ J'_1 & & & \\ & J'_2 & & \\ & & \cdots & J'_n \end{bmatrix}$$

The matrix J'_i has all its elements equal to '1'. In general, the elements (h_{ij}'s) of $F^T \circledast F$ are expressed as

$$h_{ij} = 1 \quad \text{if} \quad \left(\sum_{l=1}^{m} K(l)\right) < j \leqslant \left(\sum_{l=1}^{m+1} K(l)\right)$$

where m is such that

$$\sum_{l=1}^{m} K(l) < i \leqslant \sum_{l=1}^{m+1} K(l)$$

$$= 0 \text{ otherwise}$$

In other words,

$$h_{ij} = 1, \text{ if } P_i, P_j \in F(N_k) \text{ where } N_k \in N$$
$$= 0 \text{ otherwise}$$

Now projecting the matrix $F^T \circledast F$ over G, we get a matrix $\text{Proj}(F^T \circledast F, G)$ whose elements are expressed by

$l_{ij} =$ if both h_{ij} (an element of $F^T \circledast F$) and g_{ij} (an element of G) are equal to '1'

$\rightarrow P_i, P_j \in F(N_k)$ and $P_i \widehat{\text{GC}} P_j$

$\rightarrow P_i \widehat{\text{LC}} P_j$, by definition local concurrency set.

Recalling the definition of the L matrix we have

$$L = \text{Proj}(F^T \circledast F, G)$$

Hence the theorem.

Corollary 2.1. The LC set of a process $P_i \in F(N_k)$ is given by

$$\text{LC}(P_i) = \{P_j \mid l_{ij} = 1\}$$

where l_{ij} is an element of the L matrix.

Proof. The proof follows from the precious theorem and definition of local concurrency set.

4.3.3. Local Exclusive (LE) Set LE set of P_i is given by

$$\text{LE}(P_i) = \{F(N_k) - \text{LC}(P_i)\}, P_i \in F(N_k)$$

Also

$$\text{LE}(P_i) = \{P_j \mid l'_{ij} = 1\}$$

where l'_{ij} is an element of the matrix $\text{Proj}(F^T \circledast F, G')$.

4.3.4. Computation of largest GS Set From the given G matrix, it is possible to find the largest possible GS set for each of the processes in the system. Such a set is necessary for checking the consistency of the specification of the concurrency discussed later.

Theorem 3. Let Row (A_i) represents the ith row of any matrix A. A process P_j can be strong concurrent process of P_i iff the jth element of the row Proj(Row (G_i), Row $(G_k)\,\forall k \mid g_{ik}=1$), obtained by projection operation is equal to '1'.

Proof. The element h_{ij} (jth element) of the row given by Proj (Row (G_i), Row (G_k)) is equal to '1' iff $g_{ij}=1$ and $g_{kj}=1$, where g_{ij} and g_{kj} are the elements of the G matrix

$\rightarrow$ (i) $(P_i \widehat{\text{GC}}\ P_j)$
(ii) $(P_j \widehat{\text{GC}}\ P_k)$

Therefore, after projecting all Row (G_k)s $(\forall g_{ik}=1)$ over Row (G_i), the element h_{ij} of the resulting row will be equal to 1 iff

$$(P_i \widehat{\text{GC}}\ P_j) \quad \text{and} \quad (P_j \widehat{\text{GC}}\ P_k), \forall g_{ik}=1$$
$$\rightarrow P_j \in \text{GS}(P_i), \forall h_{ij}=1$$

Hence the theorem.

Corollary 3.1. The matrix GS′ whose rows are given by $\text{ROW}(\text{GS}_i') = \text{Proj}(\text{ROW}(G_i), \text{ROW}(G_k)\,\forall k \mid g_{ik}=1)$ represents the largest strong concurrency set.

Proof. Extending Theorem 3 to all rows of G; where each row corresponds to one of the processes, we obtain the matrix GS′. Since each of the rows represents the largest strong concurrency set of a process, the whole matrix represents the strong concurrent sets of the processes in the system. ∎

4.3.5. Computation of GP Set The GP set is computed from the *Pr* matrix and for a process P_i, it is given by

$$\text{GP}(P_i) = \{P_i \mid p_{ij}=1\}$$

where p_{ij} is an element of the *Pr* matrix.

4.3.6. Consistency of Specification of Concurrency By consistency of specification of concurrency we mean that the specification of strong concurrency and the specification of concurrency expressed by the function G should not be conflicting. The condition for consistency of specification of concurrency is given by the following theorem.

Theorem 4. If Proj(GS', ST) = ST, then the specification of concurrency is consistent.

Proof. Simple and is left to the reader.

4.3.7. Consistency of Specification of Sequencing

We observe from Proposition 6.1 that not all sequences of processes are permissible. The processes that are in strong concurrency with a specific process can not be preorder processes of the particular process. The following theorem presents a general condition for checking consistency of specification of sequencing. By the consistency of specification of sequencing we mean that the specifications of sequencing and strong concurrency should not be conflicting to each other.

Theorem 5. The specifications of strong concurrency and sequencing for a distributed system given by the 6-tuple model are consistent iff Proj(P_γ^T, ST) = 0 where 0 is a null matrix.

Proof. Let h_{ij} be an element of Proj(P_γ^T, ST) then

$$
\begin{aligned}
h_{ij} = 1 \\
&\leftrightarrow p_{ij}(\text{an element of } P_\gamma^T) = 1 \\
&\quad \text{and } s_{ij}\ (\text{an element of } ST) = 1 \\
&\rightarrow \text{(i)} \quad P_i \,\widehat{\text{GP}}\, P_j \\
&\quad \text{(ii)} \quad P_j \in \text{GS}(P_i)
\end{aligned}
$$

The conditions (i) and (ii) cannot be true simultaneously (Proposition 6.1) and hence $h_{ij} \neq 1$. Therefore no element of Proj(P_γ^T, ST) can be equal to 1. Thus we have Proj(P_γ^T, ST) = 0. Hence the theorem. ∎

Deadlock Freeness

According to Proposition 8.1 two processes can be in deadlock, if they are in a cyclic order. To ensure that the sequencing specification is deadlock-free, we can follow the graphical approach. The circles in the graph represent the processes and the directed arc from process P_i to P_j represents that P_i is a preorder of P_j.

In this section, we have presented a model for specifying a distributed computing system and methods are given to compute the various attributes of the system. A summary of these and other additional attributes for local

as well as global environments is given in Appendix A. Algorithms for computation of the various attributes are simple enough and hence are omitted. The next section present an example to illustrate the application of the developed methodology.

5. AN EXAMPLE

Before going through the details of an illustrative example, we first briefly discuss the application environment to which the methodology of specification is applied.

A block schematic of a spacecraft onboard computing system is shown in Fig. 2. It consists of various systems such as Attitude and Orbit Control System (AOCS), Sensor System (SS), Telecommand System (TCS), Telemetry System (TMS), Power System (PS) and Payload System (PLS). Significant functions of each of the these systems are summarized in Table I. A more detailed description of the same can be found in Ref. 1. Each of these systems has its own processor to carry out the functions assigned to it and all the systems are interconnected through a common communication net with a suitable topology.

Now we consider an example of a representative configuration of a spacecraft computing system. The list of processes assigned to the different nodes of the system is given in Table II. The processes are numbered in a

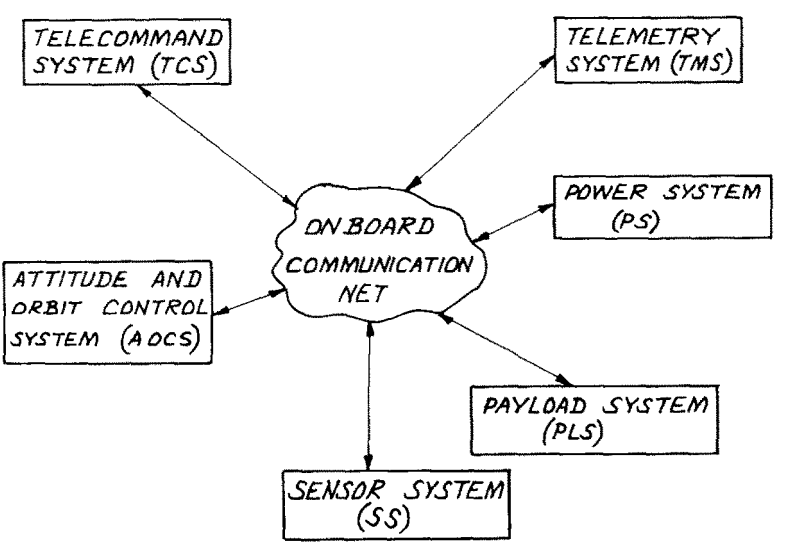


Fig. 2. A typical distributed computing system of a spacecraft.

Table I. Functions of an Onboard Spacecraft Computing System

Systems	Functions
1) Telecommand System (TCS)	
• Information exchange is limited to few words only.	• Receives coded message from ground station via RF/microwave link. • Decodes the message and generates command messages for other systems of the spacecraft. • Performs thermal control operations.
2) Attitude and Orbit Control System (AOCS)	
• Computationally most complex system in the spacecraft.	• Information exchange is limited to few words only. • Executes control algorithms to maintain the attitude and orbit of the spacecraft within the desired limits, in different phases of operation as controlled by the TCS. • Interacts with all other systems.
3) Telemetry System (TMS)	
• Has few words of information exchange with most of the systems except the PLS which sends bulk data to TMS.	• Acquires housekeeping information from various other systems. • Transmits status information via communication link to ground station, after suitable encoding.
4) Sensor System (SS)	
• Communication with other systems is limited to only few words.	• Measures attitude errors. • Supplies processed attitude errors to the AOCS.
5) Power System (PS)	
• Message transfer with all other systems is limited to few words.	• Supplies regulated power to other systems. • Controls the solar panels for optimum power tracking. • Activates emergency mode of operation when the power level falls below a threshold.
6) Payload System (PLS)	
• Large data transfer between PLS and TMS.	• Performs application oriented functions. • Process the relevant data. • Interacts with other systems. • Imposes constraints on the operation of the AOCS and PS.

Table II. A Configuration of a Typical Spacecraft Computing System

Node No.	Process No.	Function
N_1	P_1	3-Axis controller using reaction wheels
	P_2	State estimation process
(AOCS)	P_3	Housekeeping process
	P_4	Command reception and processing
	P_5	Orbit control operation
	P_6	Safe mode operation
N_2	P_7	Ground Command processing
(TCS)	P_8	Status transfer operation
	P_9	Operational mode and parameter commanding
N_3	P_{10}	Status acquisition
(TMS)	P_{11}	Formatting
	P_{12}	Data processing and analysis
N_4	P_{13}	Roll and pitch error computations
(SS)	P_{14}	Star sensor processing
	P_{15}	Yaw computation (using sun sensor)
N_5	P_{16}	Housekeeping operation
(PS)	P_{17}	Optimum power tracking
	P_{18}	Emergency mode operation
N_6	P_{19}	Payload operation
(PLS)	P_{20}	Data processing
	P_{21}	Image information transfer

particular order for the sake of simplicity. However, the relative order is not important in our model. The computing system considered here is given by

$$DC = \langle N, P, F, G, ST, Pr \rangle$$

where

$$N = \{N_1, N_2, N_3, N_4, N_5, N_6\}$$

and

$$P = \{P_1, P_2, ..., P_{21}\}$$

Functions F, G, ST and Pr are given in Figs. 3–6 in matrix form.

From the F and G matrices we can compute the L matrix using Theorem 2. The computed L matrix is given in Fig. 7.

Using this given model and the L matrix, the various sets (defined earlier) for a typical process P_8 can be computed and they are given by

$$GC(P_8) = \{P_1, P_2, P_4, P_5, P_6, P_7, P_8, P_{10}, P_{13}, P_{14}, P_{15}, P_{17}, P_{18}, P_{19}, P_{20}\}$$

$$GE(P_8) = \{P_3, P_9, P_{11}, P_{12}, P_{16}, P_{21}\}$$

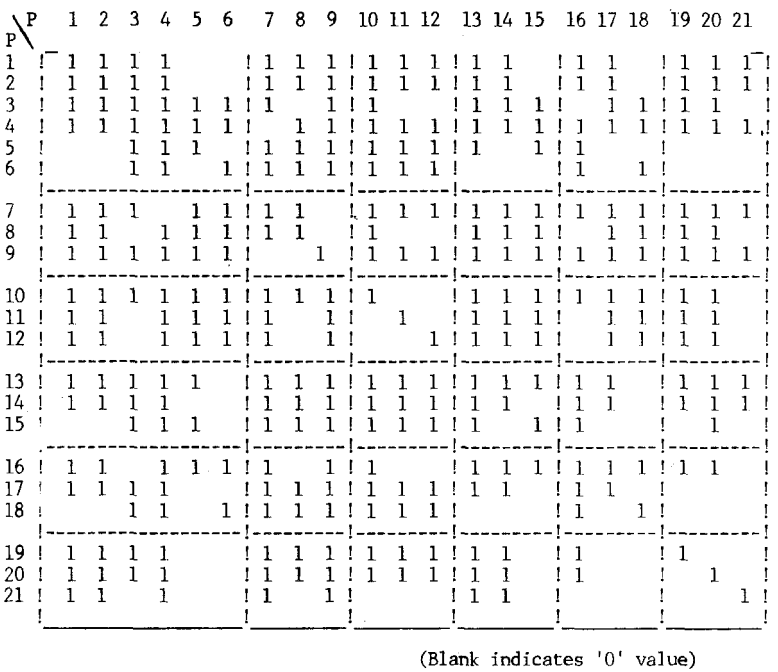
$$LC(P_8) = \{P_7, P_8\}$$

$$LE(P_8) = \{P_9\}$$

N \ P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1	1	1	1	1	1															
2							1	1	1												
3										1	1	1									
4													1	1	1						
5																1	1	1			
6																			1	1	1
	AOCS						TCS			TMS			SS			PS			PLS		

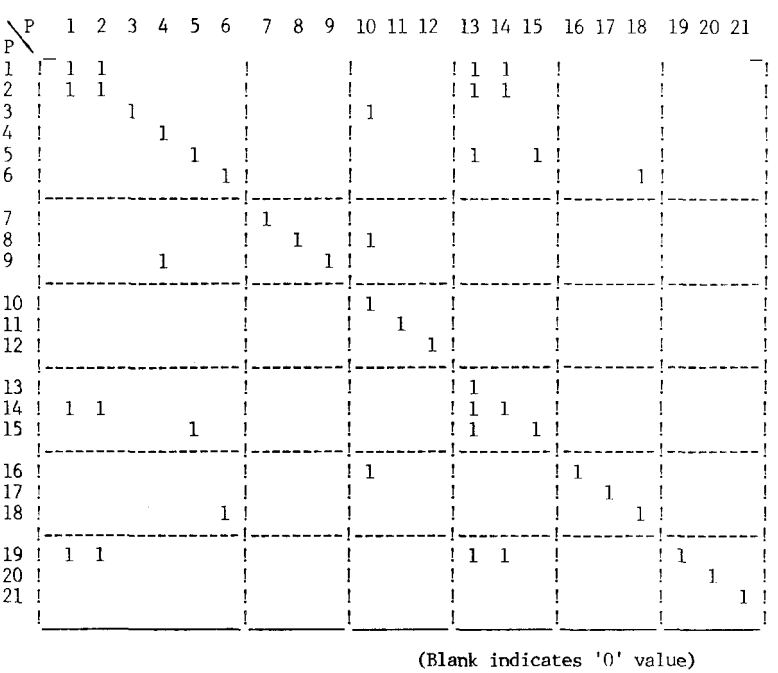
(Blank indicates '0' value)

Fig. 3. F matrix.



(Blank indicates '0' value)

Fig. 4. *G* matrix.



(Blank indicates '0' value)

Fig. 5. *ST* matrix.

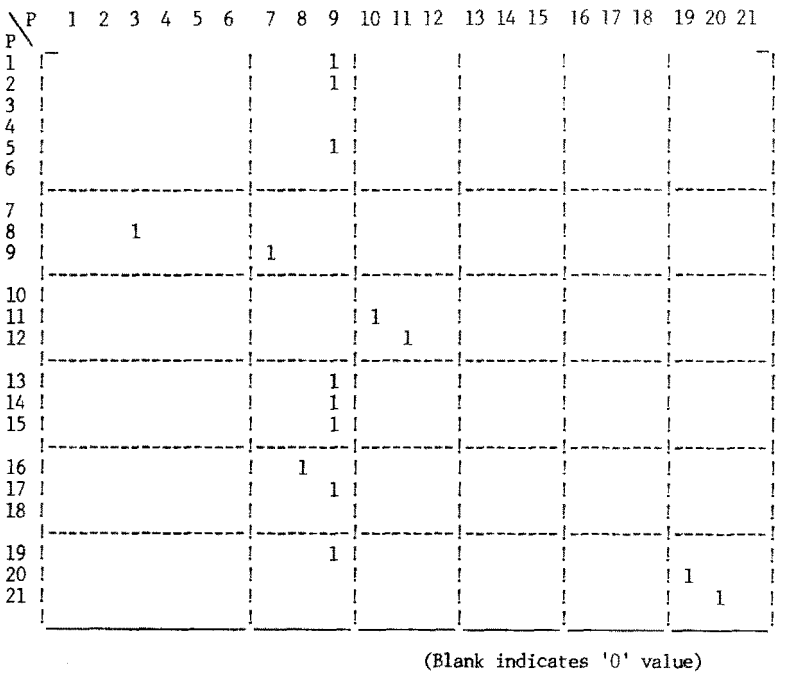


Fig. 6. *Pr* matrix.

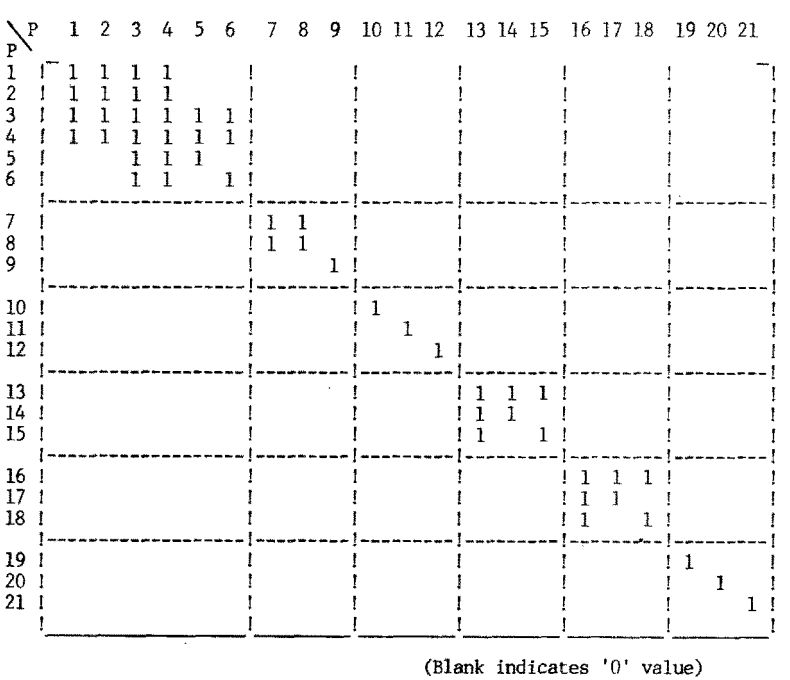


Fig. 7. *L* matrix.

Further, in order to obtain the largest possible strong concurrency sets, we have to compute the GS′ matrix first. Using Theorem 3 and its corollary, we obtain the GS′ matrix and the same is given in Fig. 8.

From the GS′ matrix, we have

$$GS(P_8) \subseteq \{P_8, P_{10}\}$$

Also $LS(P_8) \subseteq \{P_7, P_8\}$ (with respect to node N_2)
The preorder sets of P_8 can be obtained from *Pr* matrix and are given by

$$GP(P_8) = \{P_3\}$$

and $LP(P_8) = \varnothing$ (with respect to node N_2).

5.1. Consistency of Specifications

As discussed in Section 4, there are two consistency conditions for the specification of the distributed system. These conditions are again given

i) Condition for the consistency of specification of concurrency (Theorem 4)

$$\text{Proj}(GS', ST) = ST$$

P \ P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1	1	1								1	1										
2	1	1								1	1										
3			1				1														
4				1																	
5					1					1		1									
6						1									1						
7							1														
8								1		1											
9				1					1												
10										1											
11											1	1									
12												1									
13													1								
14	1	1											1	1							
15					1								1		1						
16										1						1					
17	1	1											1	1			1				
18						1												1			
19	1	1											1	1					1		
20	1	1											1	1						1	
21	1	1											1	1							1

(Blank indicates '0' value)

Fig. 8. *GS′* matrix.

ii) Condition for the consistency of specification of sequencing (Theorem 5)

$$\text{Proj}(ST, P_\gamma^T) = \varnothing$$

It can be easily verified that the given model satisfies both of these conditions and hence the specification of concurrency and sequencing are consistent.

5.2. Deadlock-freeness

Proceeding with the graphical approach as mentioned earlier, we draw a graph representing the preorder relationship obtained from the Pr matrix and the graph is shown in Fig. 9. It can be easily seen that there are no cycles in the graph, hence the specifications are deadlock-free (Proposition 8.1).

Discussion. Let us see how the developed model is useful for the protocol design for the example considered. Suppose TCS (node N_2) receives a request from TMS (Node N_3) for the execution of Status Transfer

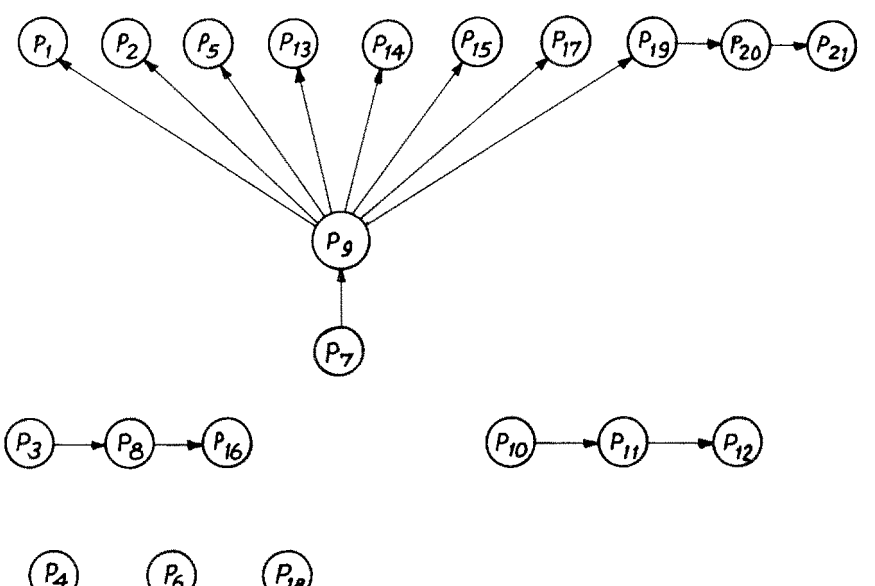


Fig. 9. Graphical representation of the preorder relationship corresponding to the *Pr* matrix.

Operation (P_8). Then in order to find out whether the request is acceptable or not at node N_2, the protocol has to respond to the following queries.

Q − 1. Does any process running in the system conflict with the requested process P_8?

Q − 2. Whether all those processes which should run concurrently with P_8 are already running?

Q − 3. Whether all those processes, need to be initiated prior to the execution of P_8 have been initiated already or not?

It can be easily seen that using the formal language or attributed grammar based approach of specification, such queries are difficult to answer in real-time. On the other hand, the methodology developed in this paper provides an easy way of carrying out the same operation. The corresponding conditions for favorable response of the queries are given by:

i) The set of processes running on the system should be a subset of $GC(P_8)$.

ii) The $ST(P_8)$ should be a subset of the processes running in the system.

iii) $Pr(P_8)$ should be a subset of the set of processes that have been initiated prior to the request of P_8.

Verification of these three conditions yields that for the acceptance of the request for P_8, the processes P_3, P_9, P_{11}, P_{12}, P_{16} and P_{21} should not be running in the system. And P_{10} must be running in the system and P_3 should have completed execution before the arrival of the request for P_8.

We examine another case. Suppose AOCS is required to run the 3-axis control process (P_1) by TCS. Then for the satisfaction of the condition (ii) mentioned previously for process P_1, P_2, P_{13} and P_{14} have to run concurrently with P_1, so the latter three processes should be already running in the system. But P_2 required that P_1 should be running in the system because of their strong concurrency.

Therefore query Q − 2 mentioned earlier will always end in a unfavorable response. There are two possible ways to come up with a favorable response, either P_1, P_2, P_{13}, and P_{14} are initiated at the same time (P_{13} can be initiated earlier too) or processes P_1, P_2, P_{13}, and P_{14} are initiated as soon as all their requests, which come one at a time have arrived. Arriving at such a solution is very difficult in case of formal language or attributed grammar based approach, whereas in our methodology this solution can be obtained by examining the various attributes.

Thus, we observe from this example, that the methodology given in this paper gives more insight of the system and facilitates in carrying out the protocol design.

CONCLUSION

To provide an unambiguous specification of concurrent, exclusive and sequencing behavior of a distributed computing system, a theory is developed in this paper. A model given here precisely describes the distributed computing system from these aspects. From the given model, it is easy to generate the various relationships needed for the protocol operation. Also it is easy to check the consistency of the specifications. Finally, an example of a practical system, a spacecraft computing system, illustrates the applicability of the developed theory. The prime motivation of the development of this theory is towards the design of a protocol for spacecraft computing systems. However, the same concepts can be used for other applications as well. The manner in which inconsistency in the specification of execution sequence is detected by the ordering of the initialization of the processes may give an impression that the technique is applicable only to the detection of deadlock during the initialization of processes. However the way the concurrency (G), strong concurrency (ST) and preorder (Pr) matrices are defined over the entire span of operation it can be seen that the deadlock detection is taken care of throughout the active period of the processes.

ACKNOWLEDGMENT

The authors would like to thank Dr. S. Murugesan of ISRO Satellite Centre, Bangalore, for his helpful discussions. The 'reviewers' comments have improved the clarity of presentation at certain places and the same gratitude is acknowledged.

APPENDIX A

Summary of the relations for computing the various attributes of the processes from the given 6-tuple model, both in global and local environments, is as follows.
(In the following 'Ele' stands for 'Elements of').

1) $C = \{P_i \mid (\exists j \text{ such that } g_{ij} = 1) \wedge (i \neq j)\}$ where $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant m$.
2) $E = \{P_i \mid (\forall j,\ g_{ij} = 0) \wedge (i \neq j)\}$ where $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant m$.
3) $C_i = \{P_j \mid h_{ij} = 1\}$ where $h_{ij} \in \text{Ele}(\text{Proj}(F, F \circledast G)$.

4) $E_i = \{P_j | h_{ij} = 1\}$ where $h_{ij} \in \text{Ele}(\text{Proj}(F, \text{Comp}(F \circledast G))$.
5) $\text{GC}(P_i) = \{P_j | g_{ij} = 1\}$ where $g_{ij} \in \text{Ele}(G)$.
6) $\text{LC}(P_i) = \{P_j | l_{ij} = 1\}$ where $l_{ij} \in \text{Ele}(L)$ and $L = \text{Proj}(F^T \circledast F, G)$.
7) $\text{GE}(P_i) = \{P_j | g'_{ij} = 1\}$ where $g'_{ij} \in \text{Ele}(G')$ and $G' = \text{Comp}(G)$.
8) $\text{LE}(P_i) = \{P_j | l'_{ij} = 1\}$ where $l'_{ij} \in \text{Ele}(L')$ and $L' = \text{Proj}(F^T \circledast F, G')$.
9) $\text{GS}(P_i) \subseteq \{P_j | h_{ij} = 1\}$ where $h_{ij} \in \text{Ele}(\text{GS}')$ and the ith row of GS′ is given by

$$\text{ROW}(GS'_i) = \text{Proj}(\text{ROW}(G_i), \text{ROW}(G_j) \forall j | g_{ij} = 1).$$

10) $\text{LS}(P_i) \subseteq \{P_j | h_{ij} = 1\}$ where $h_{ij} \in \text{Ele}(\text{LS}')$ and the ith row of LS′ is given by

$$\text{ROW}(\text{LS}'_i) = \text{Proj}(\text{ROW}(L_i), \text{ROW}(L_j) \forall j | l_{ij} = 1)$$

where $l_{ij} \in \text{Ele}(L)$.

11) $\text{GP}(P_i) = \{P_j | p_{ij} = 1\}$ where $p_{ij} \in \text{Ele}(Pr)$.
12) $\text{LP}(P_i) = \{P_j | h_{ij} = 1\}$ where $h_{ij} \in \text{Ele}(\text{Proj}(F^T \circledast F, Pr))$.

REFERENCES

1. Report Doc. No. INSAT-II-00-84-04-05–11, ISRO Satellite Centre, Bangalore, India (April 1984).
2. V. K. Agrawal, L. M. Patnaik and P. S. Goel, Specification and Validation of a Protocol for Real-time Distributed Computing System, *Proc. International Conf. on Computers, Systems and Signal Processing*, Bangalore, India, pp. 265–269 (December 1984).
3. G. L. Lann, On Real-time Distributed Computing, *IFIP*, pp. 741–753 (1983).
4. G. L. Lann, Deterministic Multiple Access Protocol for Real-time Local Area Networks, Report No. 246, (October 1983).
5. W. W. Chu, L. J. Holway, M. Lan and K. Efe, Task Allocation in Distributed Data Processing, *IEEE Computer*, **13**(11):57–69 (November 1980).
6. T. C. K. Chou and A. Abraham, Load Balancing in Distributed Systems, *IEEE Trans. Software Eng.*, **SE-8**(4):401–412 (July 1982).
7. W. Kohler, A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems, *ACM Computing Surveys*, **13**(2):149–183 (June 1981).
8. H. T. Kung and J. T. Robinson, On Optimistic Methods for Concurrency Control, *ACM Trans. Database System*, **6**(2):213–226 (June 1981).
9. S. S. Yau and C. C. Yang, An Approach to Distributed Computing System Software Design, *Proc. IEEE Conf. on Distributed Computing Systems*, pp. 31–42 (October 1979).
10. P. M. Lu and S. S. Yau, A Methodology for Representing the Formal Specification of Distributed Computing System Software Design, *Proc. IEEE Conf. on Distributed Computing Systems*, pp. 211–221 (October 1979).
11. Towards Formal Specification of a Distributed Computing System, Technical Report No. CSS-05-85, ISRO Satellite Centre, Bangalore, India, (April 1985).