分散処理システム 24－6
（1984. 11. 16）

協調分散システム―Adaワークステーションを用いた事例研究※

藤田昭平，　大島淳一，　深尾　毅

（東京工業大学　工学部）

1．協調分散システム―その目的

従来のコンピュータ・ネットワーク用のソフトウエアに於いては，人間の介在が前提になっている場合が多く，通信機能にのみ重点が置かれがちであった（[1],[15]）。

これに対して，CIMS(Computer Integrated Manufacturing Systems)におけるロボット・ネットワーク（図1-1）等では，従来のコンピュータ・ネットワークが対象にしていたのとは相異なる新しい問題が生じて来る。例えば，

（i）ロボットが種々のセンサー情報を獲得し，その動作を自律的に修正する機能 [2]，

（ii）複数のロボットが協調作業を行う機能

等の実現に際しては，センサー，ロボットの運動制御アルゴリズムを実行する応用プログラムと通信機能を実現する通信プログラムのインタフェースが一つの重要な問題点となる。

共有メモリにより結合されたプロセッサの集合体をサイトと言う（縮退した場合として，単一プロセッサの場合を含む）。複数のサイトが通信媒体により結合されたシステムを分散システムと云う。ただし、サイト間にはメモリ及びクロックの共有はなく，サイト間にはメッセージの交換のみが許されている。

メッセージとは，

（i）オブジェクトに対する操作の要求，

及び

（ii）オブジェクトに対して履行された操作の結果

を意味する。メッセージの構造は応用アルゴリズムに依存している。メッセージの本質的な特徴は，その内部構造の如何を問わず，次の動作を実行する前に受理しなければならない情報の最小単位である点にある。

協調分散システムとは，各サイトがメッセージの交換を行なうことにより“一つの目的”を達成するシステムである。

協調分散システムを特徴づける大きな要因は

（i）“一つの目的”を達成するようシステム全体の制御が存在する，

（ii）このシステム全体の制御は，集中化された大域情報に基づいてではなく，局所情報にのみ基づいて行われる。

ことである([13])。協調分散システム上で実行され得るプログラムを分散プログラムと云う。特に，局所情報に基づいて，システム全体の制御機能を実現したプログラムの集合を分散オペレーティングシステム（DOS）と云う。

本論文では，並行プログラムと分散プログラムを明確に区別する。並行プログラムの実行に際してはプロセス間における共有メモリの存在が許されるが，分散プログラムを実行するサイト間には共有メモリは存在しない。

ここでは，協調分散システムのソフトウエアに関心があり，通信媒体（のハードウエア）等については論じない。現在最も

※　本研究の一部は，科研費（59460120）による。

欠如しているのは、協調分散システム用のソフトウエアである([注1])。ソフトウエアの質は，そこで用いられている言語に大きく依存する。

2．分散オペレーティングシステム一設計モデル

本節では、高信頼性及び復帰機能を有する分散システムの設計モデルについて述べる。記憶媒体，プロセッサ及び通信媒体等多数の物理素子を用いて、より安定かつ安全なシステムを構成するのが最終目標である。このためには、階層化、モジュール化により安定かつ安全な分散オブジェクトを構築し、これらのオブジェクト間の相互作用はトランザクションに基づいたオブジェクト指向モデルを用いる([9])。

記憶媒体とは、システムの稼働中、種々の原因による情報損失の確率は零ではない物理的な記憶素子を示す。システムの稼働中に、プロセッサがその制御情報を失う確率も零ではない。通信媒体においては、データグラム方式に基づいている等の場合には、メッセージの遅延確率が高いことがある。

以上のような特性を有する多数の物理素子を用いて構築されたシステムを制御し、協調機能を達成させるためのＤＯＳは次のような階層構造を有している。

第一段階としては、安定な記憶機能、安定なプロセッサ機能及び安全な通信機能を実現する仮想素子を構築する。

第二段階では、安定な記憶機能及び安定なプロセッサ機能に基づいて識別子（ＵＩＤ）を生成する。このＵＩＤはシステム全体で一意な名前であり、すべてのオブジェクトにそれぞれ与えられる。すなわち、各オブジェクトに与えられたＵＩＤにより、そのオブジェクトが生成されたサイト名及びそのオブジェクトの型を識別することができる。これらのＵＩＤによりシステムにおける論理的名前空間が定義される。

オブジェクト（の集合）に対する一連の原始的操作（primitive operation）はトランザクションと呼ばれる。このトランザクションにもＵＩＤが割り当てられる。

安定な記憶機能，安定なプロセッサ及びＵＩＤに基づいて安全かつ安定なオブジェクトが構成される。安定なオブジェクトとは，システムの一部に故障等が発生した場合でも，高い確率で生存し続けるオブジェクトである。原始的操作は、これらのオブジェクトに対しては　アトミックである。安全なオブジェクトとは、権限を有しているアクティビティだけがアクセス可能な保護されたオブジェクトである。

第三段階では、以上の機能を用いてさらにＮＴ及びＲＰＣの機能を実現することにより安定かつ安全な分散オブジェクトを構成する（図2-1)。

3．サイトの構造

サイト内には複数の

アクティビティ：　独立な制御の下で実行され得る計算モジュール

が存在している。ここでは，各問題に応じた応用アルゴリズムが実行される。例えば、

（i）　ロボットの運動制御アルゴリズム

（ii）　センサーからデータを獲得するアルゴリズム

（iii）　データベースへのアクセスアルゴリズム

等である。これらのアルゴリズムはタスク ACTIVITY の本体で記述される。

複数のロボットが協調作業を行うためにはロボット間の同期が不可欠である。したがって，各ロボットの運動制御を実行する計算モジュール（アクティビティ）間の同期が必要となる。一方，センサー用計算モジュールは，ロボットの動作が終了するのを待っているような事態は避けるべきである。すなわち，センサーからのデータを処理している計算モジュール（アクティビティ）とロボットの運動制御を行なっている計算モジュール（アクティビティ）は非同期に実行されるべきである。

アクティビティ間のメッセージ交換は，

通信モジュール

を介して行なわれる（間接通信）。リアルタイム・システムでは，事象駆動（event driven）に基づく場合が多く，また事故時の緊急信号等は最優先されねばならないという要求に対処するため，この通信モジュールをタスクとして記述する。通信モジュールの機能は，基本的にはＯＳＩ参照モデル([18])のセッ

ション層に相当している。ただし，協調分散システムの特徴の一つは，汎用部品を用いて各用途に最も適した専用システムを構築することであり，セッション層では，問題指向のプロトコルが記述される。非同期メッセージ交換のためには，一方向系の通信モジュールを用いる（図3-1)。

- エントリ宣言を有し、どのような条件の下でアクティビティから（へ）メッセージを受理（授与）するかを決定する。アクティビティより（へ）受理した（授与する）メッセージを一時的に保管する。（フロー制御）
- このメッセージをどのサイトへ（から）輸出（輸入）するかを決定する。さらに，一対一か一対多の通信かを決定する。（アクセス制御 [注2]）

以上の機能は、タスクcommunication の本体で記述される。

同一サイト内にあるアクティビティ（タスク）間におけるメッセージの交換は，エントリ宣言及びエントリ呼び出しにより，Ａｄａのランデブー機構により達成される。

しかし，Ａｄａのランデブー機構では、各アクティビティ（タスク）はエントリ呼び出しを行なうべき相手の名前を知っている必要がある。各アクティビティ（タスク）が同一のサイト上にある場合には，共有メモリの存在によりこの問題は解決される([注3])。

これに対して、相異なるサイト間には共有メモリは存在しない。したがって，相異なるサイト上にある各アクティビティ（タスク）がメッセージ交換を行なうためには，Ａｄａの言語規定だけでは不充分である。

4．サイト間通信プリミティブ

サイト間の通信プリミティブは，

(i) 上位レベルでは，アクティビティに於いて扱われる応用アルゴリズムに依存し，

(ii) 下位レベルでは，処理系に依存する。

ソフトウエアのポータビリティを高めるためには、通信プリミティブをパッケージとし，応用アルゴリズム及びプロトコルの変更，または処理系の相違に対処できるようにするのが望ましい。サイト間の通信プリミティブを言語に組み込んだものが分散言語である。しかし，種々の応用アルゴリズム，種々のプロトコル，通信媒体の信頼性，さらに種々の処理系に対処できるためには多くの未解決な問題がある（[10],[11],[14]）。

本論文では，Ａｄａのパッケージ機能を用いて，それぞれの用途に応じた通信プリミティブを構築する方法を以下に示す。現時点では，この方法は最も現実的かつ有効なものと思われる。

メッセージとは，抽象的には，オブジェクトに対する操作の要求及びオブジェクトに履行された操作の結果を意味することは既に述べた。具体的な例としては制御信号(control signal),データ等がある。例えば、複数のロボットが協調作業を行うためには同期信号の授受が不可欠である。したがって，メッセージには種々の型，サイズがある。また，センサーとロボット，データベースとロボット間には，データの効率的かつ信頼性のある送受信が必要である。

ここでは，“ポート”の概念に基づいたサイト間通信プリミティブについて述べる([注4])。

(1) メッセージの構造と識別子

制御信号とデータを区別する。

特定の型を有するメッセージを送受信するのに用いられるポートを“TYPED-PORT”と云う。Ａｄａは厳密なデータ型を有する言語であり，TYPED-PORTの採用はＡｄａのコンパイル時における型チェックの思想に適ったものである。メッセージの型は，各アクティビティにおける応用アルゴリズムにより決定される。

ＵＩＤはアクティビティの識別記号であり，サイト番号及びアクティビティ名が記述されている。手続きSEND（Ｍ：MESSAGE;ＵＩＤ)(手続き RECEIVE（Ｍ：out MESSAGE ＵＩＤ))はメッセージを特定サイトの特定のアクティビティへ送信（から受信）するものであり，ロボット間の協調作業を達成するために用いられる。これに対して，手続きSEND（Ｍ：MESSAGE)（手続きRECEIVE(Ｍ：out MESSAGE)）は事故時における緊急制御信号を複数のロボットにブロードキャスト送（受）信するためのものである。

以上の機能は，パッケージTYPED-PORTとして実現されている。

(2) メッセージの一時的な保管

一般に各階層において，メッセージを一時的に保管しておく機能が必要である。これは，ＦＩＦＯのバッファーにより実現される。各ポートにそれぞれ固定または可変サイズのバッファーを配置する。必要な場合には，二重バッファー等の手段を構ずる。非同期の場合には，バッファーの効率的な管理が必要となる。

(3) 通信路の確立

まず，プロセッサが外界（他のサイト）とメッセージの授受を行うためには，I/O ポートへのアクセスが必要である。パッケージPHYSICAL-INTはこの機能を実現したものである。手続きEXPORT(IMPORT)を呼び出すことにより，I/O ポートを経由して一つのASCII 文字が送り出され（受け取られ）る。このパッケージの本体は，LOW-LEVEL-IOパッケージを用いて実現されている。Ａｄａには，LOW-LEVEL-IOと呼ばれるライブラリ・ユニットが用意されている。このパッケージには，既定の手続きSEND-CONTROL及びRECEIVE-CONTROL がある。SEND-CONTROLはデバイス（I/O ポート）に制御情報を送るのに用いられ，RECEIVE-CONTROL はデバイス（I/O ポート）から制御情報を受け取るのに用いられる。これらの手続きは，デバイス（I/O ポート）を指定するDEVICEパラメータと，伝達されるべき制御情報を示すDATAパラメータを有している。ただし，これらのパラメータの型は処理系に依存したものである（[6]）。パッケージPHYSICAL-INTは，プログラムを処理系に依存するモジュールと依存しないモジュールに分離する役割を果たしている。

種々のサイズのメッセージ（を通信網を介して）の授受は，パケットスイッチングにより行う。各パケットに含まれているメッセージ本体は（最大）64のASCII 文字からなるものとする。パケットの授受を実現する機能は，パッケージPACKET-INTで記述されている。このパッケージPACKET-INTはメッセージ長が通信機能に及ぼす影響を検討する際に有用である([注5])。

5．事　例

(5-1) ハードウエア構成

二台のMicro Engine及びSUN-2/120 FSからなる非同質なシステム構成になっている（図5-1)。SUN-2/120 FSは，ネットワーク・シミュレーターとしてここでは用いられている。多重リンク構成になっているのは，リアルタイム機能，復帰機能等のテストを容易に行えるようにするためである。

(5-2) ソフトウエア構成

図 A-1に単一プロセッサ上の並行プログラムを示す。このプログラムは，アルゴリズム上の論理的な正当性をテストするためのものである。ここでは，具体的な分散アルゴリズムとして，非同期並行ニュートンアルゴリズム([4]) が採用されている。分散プログラムを図 A-2に示す。タスクACTIVITYの仕様部，本体ともに全く変化がないことに注意して頂きたい(Transparency)。これにより，単一サイト上で単一プロセッサを用いて行われたアルゴリズムに関する論理的な正当性に関するテストの結果は複数のサイトからなる分散システム上でもやはり有効である。アクティビティ及び通信モジュールはタスクとなっており，各サイト自体がマルチプロセッサ構成になっている場合には，これらのタスクは同時に実行され得る。

通信プリミティブは，階層的モジュール構造を有し，パッケージTYPED-PORTとして実現されている。この通信プリミティブは，通信モジュールタスクの本体に，with-use節によりコンパイル時にリンクされる。この最上位層パッケージTYPED-PORTは，with-use節により下位層からのサービス提供を受ける。

6．あ と が き

ＣＩＭＳにおける分散ソフトウエアのニーズ，特徴，要求される機能を論じ，Ａｄａ言語を用いて分散ソフトウエアを構築する方法及びその具体例を示した。すなわち，

(1) 安定かつ安全なオブジェクト指向分散オペレーティングシステムの設計モデル

(2) 同期，非同期　のメッセージ交換を行うためのサイトの構造

(3) 各用途に最も適したサイト間通信プリミティブを構築する方法

(4) 具体的な分散アルゴリズムを用いたＸＤＯＳの性能

について論じた。

ＣＩＭＳの使命を制するのは分散ソフトウエアであり，例えば信頼性対効率という相反する機能を両立させるために，ハード・ソフト両面に渡る多くの研究課題が解決されねばならない。

従来は，ハードウエアの性能上の制約等により，効率のみを重視した言語，ソフトウエアが主流的地位を占めて来た。しかし，ハード面における進歩が今後も続くものとすれば，（さらに，経済的側面をある程度無視すれば）信頼性の方に重点をおいたＡｄａ等の言語，及びソフトウエアが今後主流になるものと思われる。

［注１］ 従来のオペレーティングシステムは、大域情報が中央（共有メモリ）に貯られておりかつ利用できるという前提の上に成り立っているのがほとんどである。例えば、Star OS[14]：ｉＭＡＸ[5]は集中化した大域情報に基づいている。

［注２］分散システムにおけるアクセス制御問題の詳細については，ここでは言及しない。安全性と効率の兼合いに関しては未解決の問題が多々あり，稿を改めて述べる必要がある。

［注３］例えば，ｉＡＰＸ４３２は共有メモリ型ＭＩＭＤマルチプロセッサである。

［注４］“port”は，“door”を意味するラテン語の“porta”に由来している。ポートは，メッセージがサイトから輸出されたり，またサイト内へ輸入されるための論理的な通信路である。

［注５］ここでは，サイト間はRS-232C I/O ポートにより物理的に結合されているものとした。Cambridge Ring又はEthernet等ではさらに上位レベルの機能まで提供される。

参 考 文 献

[1] Cypser, R.J.: **Communications Architecture for Distributed Systems.** Addison-Wesley Pub., (1978).

[2] Dertouzos, M.L.: "Control Robotics: The Procedural Control of Physical Processes," **Proc. IFIP 74,** pp. 807 - 813, (1974).

[3] Fujita, S.: "On the Observability of Decentralized Dynamic Systems," **Information and Control,** Vol. 26, No. 1, pp. 45 - 60, (1974).

[4] Fujita, S.: "Distributed MIMD Multiprocessor System with MicroAda/SuperMicro(TM) for Asynchronous Concurrent Newton's Algorithms," **Proc. 5th ACM-SIGSMALL Symp.,** pp. 49 - 59, (1982).

[5] Fujita, S.: "Software Components for Real-Time Advanced Robot Control Systems - A Case Study with Ada Workstation," **Proc. Int. Conf. Advanced Robotics,** pp. 409 - 416, (1983).

[6] GenSoft: **STC Ada Reference Manual.** Systems Technology Center, Pittsburgh, (1983).

[7] 藤田昭平: **Ａｄａによるリアルタイム・プログラミング（パケット通信用ソフトウエア部品），** コンピュートロール, 6, (1984).

[8] Kahn, K.C. et al.: "iMAX: A Multiprocessor Operating System for an Object-Based Computer," **Proc. 8th ACM-SIGOPS Symp.,** pp. 127 - 136, (1981).

[9] Lampson, B.W. et al.: **Distributed Systems - Architecture and Implementation: An Advanced Course.** Springer-Verlag, (1981).

[10] Liskov, B.: "Primitives for Distributed Computing," **Proc. 7th ACM-SIGOPS Symp.,** pp. 33 - 42, (1979).

[11] Liskov, B. & M. Herlihy: "Issues in Process and Communication Strucutre for Distributed Programs," **MIT Laboratory for Computer Science,** PMG-MEMO 38, (1983).

[12] Spector, A.Z.: "Multiprocessing Architectures for Local Computer Networks," **Stanford University,** STAN-CS-81-874, (1981).

[13] Stankovic, J.A. & A.V. Dam: "Research Directions in (Cooperative) Distributed Processing," in **Research Directions in Software Technology** (P. Wegner ed.), pp. 611 - 638, (1979).

[14] Strom, R.E. & S. Yemini: "NIL: An Integrated Language and System for Distributed Programming," **IBM T.J. Watson Research Center,** RC9949(#44100), (1983).

[15] Tanenbaum, A.S.: **Computer Networks.** Prentice-Hall, Inc., (1981).

[16] U.S. Department of Defense: **Ada Programming Language,** ANSI/MIL- STD-1815A, (1983).

[17] Vegdahl, S.R. & A.K. Jones: "StarOS Microcode Wizard's Manual," **Carnegie-Mellon University,** Department of Computer Science, 15 April, (1981).

[18] Zimmerman, H.: "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," **IEEE Trans. Communications,** Vol. COM-28, No. 4, pp. 425 - 432, (1980).

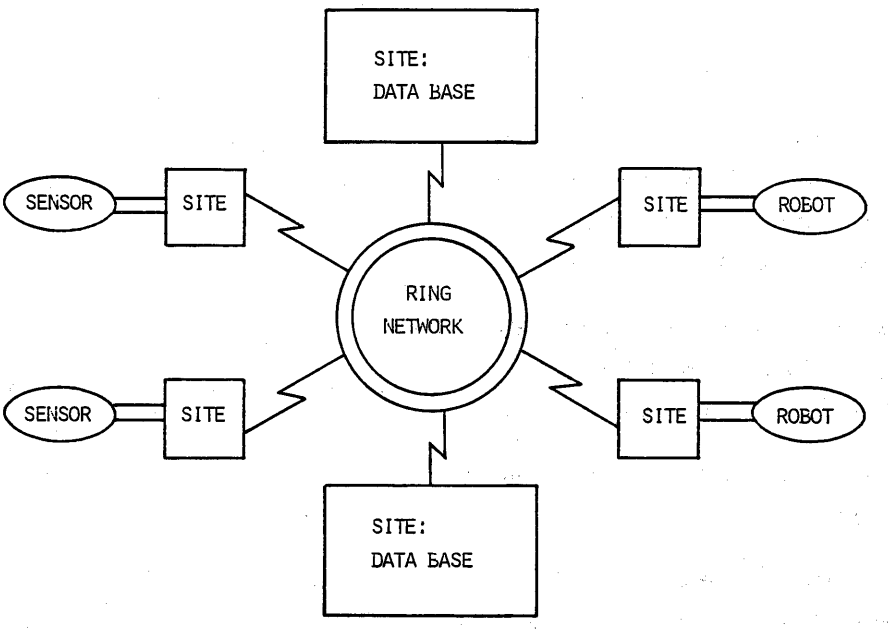

FIG. I.I ROBOT NETWORK

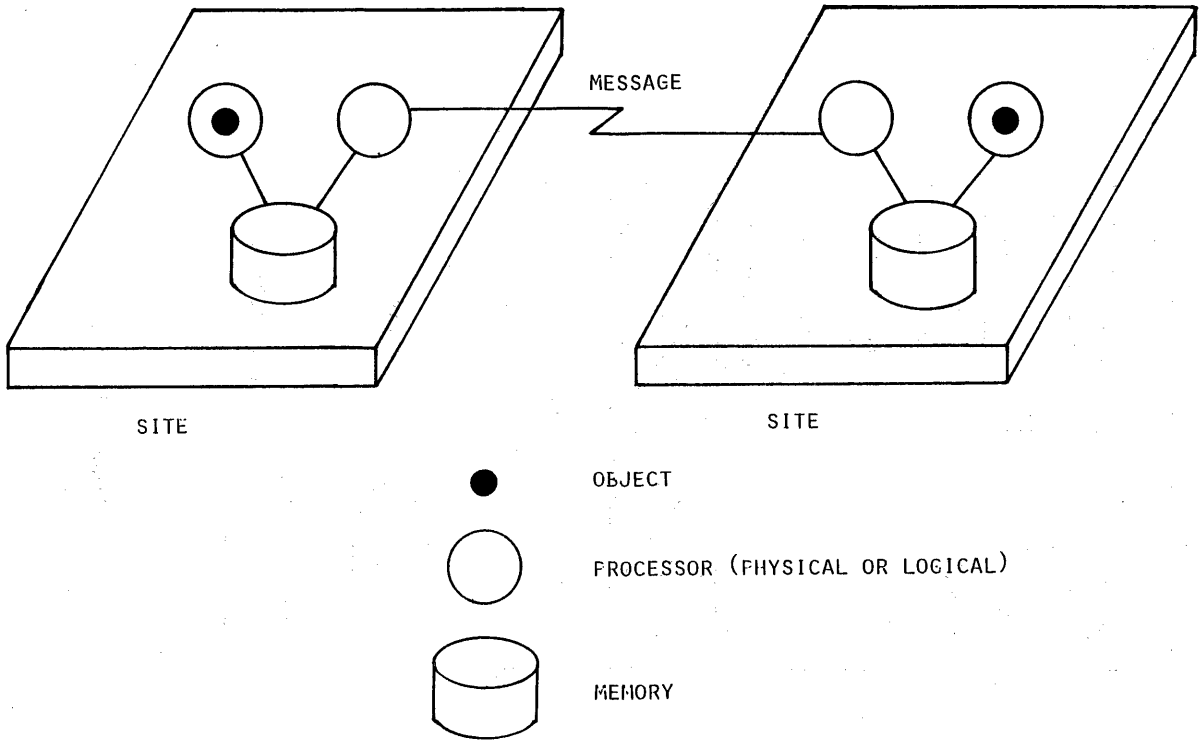

FIG. 2.1 COOPERATIVE DISTRIBUTED COMPUTING

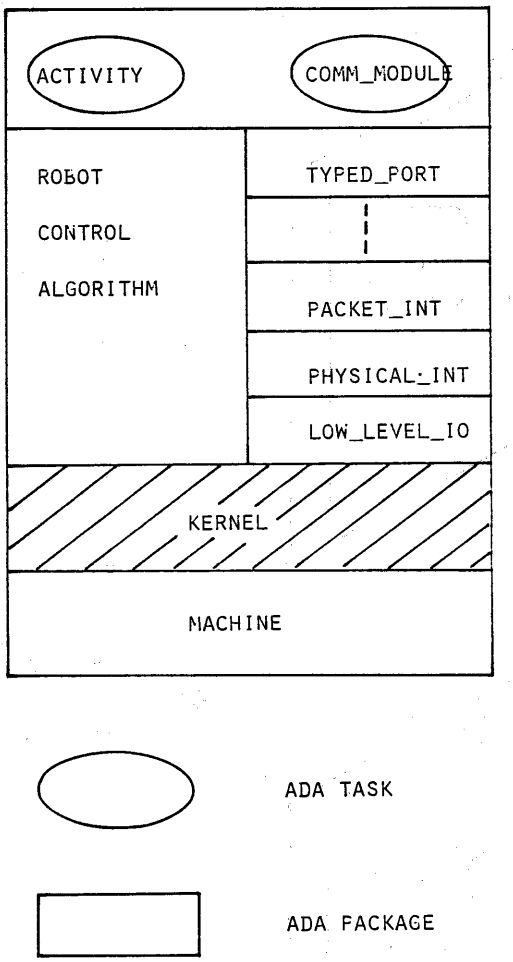

FIG. 3.1 STRUCTURE OF SITE

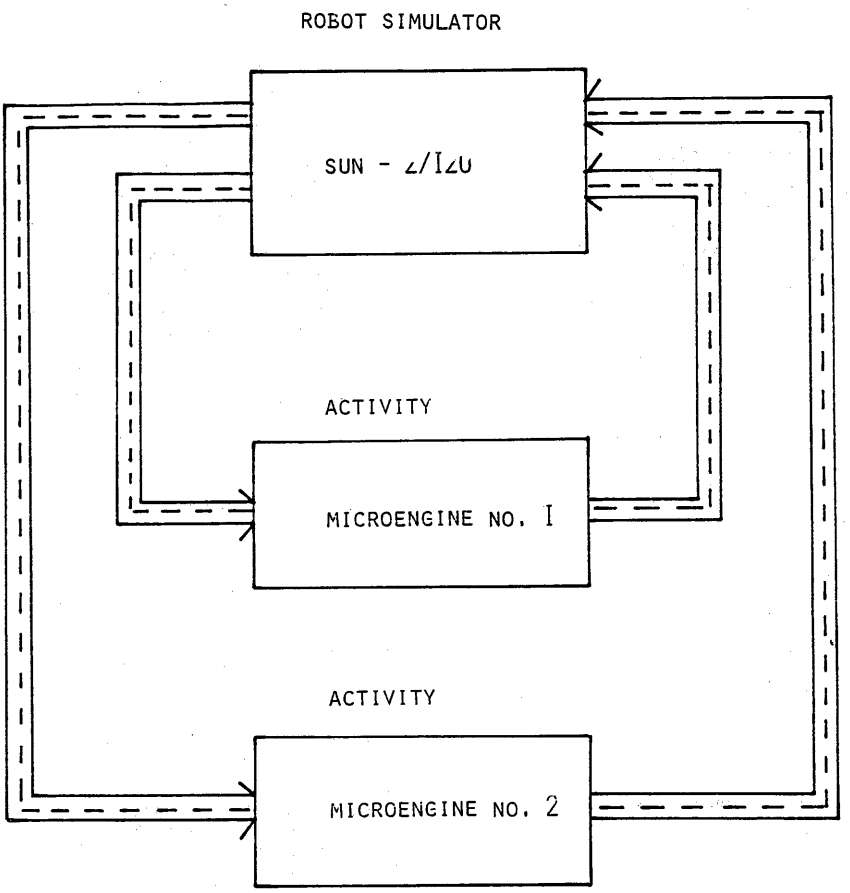

FIG. 5.I EXPERIMENTAL DISTRIBUTED SYSTEM

```
with STC_IO ; use STC_IO ;
with PROBLEM ; use PROBLEM ;

procedure CONCURRENT is

  task ACTIVITY_1;

  task ACTIVITY_2;

  task COMMUNICATION_MODULE_1 is
     entry SEND (J:in RMATRIX);
     entry RECEIVE (J:out RMATRIX);
  end COMMUNICATION_MODULE_1;

  task COMMUNICATION_MODULE_2 is
     entry SEND (X:in RARRAY);
     entry RECEIVE (X:out RARRAY);
  end COMMUNICATION_MODULE_2;


  task body ACTIVITY_1 is separate ;


  task body ACTIVITY_2 is separate ;


  task body COMMUNICATION_MODULE_1 is separate ;


  task body COMMUNICATION_MODULE_2 is separate ;


  begin
     null;
  end CONCURRENT;
```

Fig. A-1 Ada on Single-Processor System

```
with STC_IO ; use STC_IO ;
with PROBLEM; use PROBLEM;

procedure DISTRIBUTED_1 is

   task ACTIVITY_1 is
       pragma STACK_SIZE(1500);
   end ACTIVITY_1;

   task COMMUNICATION_MODULE_1 is
      pragma stack_size(500);
      entry SEND (J:in RMATRIX);
      entry RECEIVE (J:out RMATRIX);
   end COMMUNICATION_MODULE_1;

   task COMMUNICATION_MODULE_2 is
      pragma stack_size(500);
      entry SEND (X:in RARRAY);
      entry RECEIVE (X:out RARRAY);
   end COMMUNICATION_MODULE_2;

   task body ACTIVITY_1 is separate ;

   task body COMMUNICATION_MODULE_1 is separate ;

   task body COMMUNICATION_MODULE_2 is separate ;

   begin
      null;
   end DISTRIBUTED_1;
```

```
with STC_IO ; use STC_IO ;
with PROBLEM; use PROBLEM;

procedure DISTRIBUTED_2 is

   task ACTIVITY_2 is
       pragma STACK_SIZE(1500);
   end ACTIVITY_2;

   task COMMUNICATION_MODULE_1 is
      pragma stack_size(500);
      entry SEND (J:in RMATRIX);
      entry RECEIVE (J:out RMATRIX);
   end COMMUNICATION_MODULE_1;

   task COMMUNICATION_MODULE_2 is
      pragma stack_size(500);
      entry SEND (X:in RARRAY);
      entry RECEIVE (X:out RARRAY);
   end COMMUNICATION_MODULE_2;

   task body ACTIVITY_2 is separate ;

   task body COMMUNICATION_MODULE_1 is separate ;

   task body COMMUNICATION_MODULE_2 is separate ;

   begin
      null;
   end DISTRIBUTED_2;
```

Fig. A-2 Ada on Distributed Multi-Processors System