計算機アーキテクチャ59−16
オペレーティング・システム29−16
(1985・10・25)

分散オペレーティング・システムにおける異常検出

Failure Detection in Distributed Operating System

藤田昭平
Shohei Fujita

東京工業大学 工学部
Tokyo Institute of Technology

1．まえがき

従来のコンピュータ・ネットワーク用のソフトウエアにおいては，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([3] ,[24])。

これに対して，CIM(Computer Integrated Manufacturing)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，

- 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([4])
- 複数のロボットが協調作業を行う機能
- データベース管理機能

等の実現に際しては，センサー，ロボットコントローラへの指令，データベースへのアクセス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り，高信頼性，高稼動率が要求される。

さらに，CIM用のコンピュータ・システムは，物理現象のモニターリングおよび制御に必要な最大許容時間（temporal constraint）— この最大許容時間は物理現象を支配する法則により定められ，既知である場合が多い—に適う性能を有するように設計されねばならない([13] ,[16])。

本研究は，CIM用コンピュータ・システムにおける分散ソフトウエアの設計法及びその実現法を目標とするものである。ロボット，知能センサー，データベース等を分散オブジェクトとして扱い，各オブジェクトの操作はトランザクションにより行う方法を提案する。

2．分散オペレーティング・システム

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云う。（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（例えば，LAN）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可能である。

メッセージとは，

- オブジェクトに対する操作の要求

及び

- オブジェクトに対して履行された操作の結果

を意味する。メッセージの本質的な特徴は，ある動作を実行するのに必要な情報の単位である点にある。各サイトがメッセージの交換を行うことにより“一つの目的”を達成するシステム

を協調分散システムと云う　（図２－１）。

協調分散システムを特徴づける大きな要因は，

- “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
- 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。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・システムと云う。分散オペレーティング・システムの重要な役割は，信頼性のある分散オブジェクトの管理である。このためには，トランザクション機能（アトミック・アクション）および（相異なるサイト上にある）オブジェクト間通信機能の実現が必要となる。

本文では，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。ＣＩＭ用コンピュータ・システムに有用なのは分散プログラムである。

３．アクティビティ

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アクティブなオブジェクト

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

- ロボットコントローラへの指令
- 知能センサーへの指令
- データベースへのアクセスアルゴリズム

等である。アクティビティは一つまたは複数のＡｄａタスクを用いて記述される（図３－１）。ただし，これらのタスクは同一のアドレス空間を共有しているものとする。

同一サイト内にあるアクティビティ間のメッセージ交換はＡｄａのランデブー機構により実現される。Ａｄａのランデブー機構では，各タスクはエントリ呼び出しを行うべき相手の名前を知っている必要がある。各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。

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。相異なるサイト上にある各アクティビティが通信（メッセージ交換）を行なうためには，Ａｄａのタスク規定だけでは不充分である。

相異なるサイト上にあるアクティビティ間の通信（メッセージ交換）を可能にし，相異なるサイト上にあるアクティビティ間の協調動作を与えるのが分散オペレーティングシステムの役割の一つである。

４．通信モジュール

アクティビティ間の通信は“通信モジュール”を介して行なわれる。各通信モジュールはＬＡＮインタフェースにより（論理的）に結合されている（図４－１）。

通信モジュールの機能は，概念的にはＯＳＩ参照モデルのセッション層([30])以上に相当している。ただし，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，通信モジュールは応用指向のプロトコルを実現するための機能を提供しなければならない。

例えば，

- ＬＡＮインタフェースにより提供されるサービスを用いることにより，アクティビティ間の同期通信，非同期通信を可能にする。
- 名前とアドレスの変換を行なう。各アクティビティは固有の名前を有しており，この名前はアクティビティ間の通信の際に用いられる。一方，アドレスとは各アクティビティがどのサイト上にあるかを示すものであり，このアドレスはＬＡＮインタフェースにより用いられる。名前とアドレスの変換機能により，各アクティビティは名前だけを指定することにより通信が行なえ，相手のアクティビティがどのサイト上にあるかを知っている必要がなくなる（ネットワーク・トランスペアレンシイ）。これにより，分散プログラムの作成が容易になる。

4．1　同期通信

通信機構は同期通信と非同期通信に大別される。※　同期通信では，送信（受信）アクティビティは，受信（送信）アクティビティとのセッションが確立し，メッセージが受信アクティビティに受理されるまで一時停止状態となる。この機構のＡｄａによる実現方法は次のようになる（図４－２）。

①　送信（受信）アクティビティはエントリ呼び出し

ＳＹＮ＿ＣＯＭＭ．ＳＥＮＤ（－－－）

（ＳＹＮ＿ＣＯＭＭ．ＲＥＣＥＩＶＥ（－－－））

を行ない，一時停止状態となる。ただし，両方のエントリ呼び出しが行なわれる時刻は相異なっていることに注意。

②　通信モジュール（ＳＹＮ＿ＣＯＭＭ）におけるacceptの実行により，送信アクティビティと受信アクティビティ間のセッションが確立される。

③　サイト間における実際のメッセージ交換は

ＲＥＭＯＴＥ＿ＳＥＮＤ（－－－）

（ＲＥＭＯＴＥ＿ＲＥＣＥＩＶＥ（－－－））

の実行により実現される。これに必要なプロトコルは，ＬＡＮインタフェースが提供する。

④　通信モジュールにおけるｅｎｄの実行により，送信アクティビティと受信アクティビティ間のセッションが解放される。

4．2　同期通信（ＲＰＣ）

第二の同期通信は遠隔手続き呼び出し（ＲＰＣ）と呼ばれるものである。送信アクティビティはメッセージを受信アクティビティへ送り，このメッセージにより受信アクティビティが要求された仕事を実行し，その結果が再び送信アクティビティへ返って来るまで一時停止状態となる（図４－３）。

①送信アクティビティはエントリ呼び出し

ＲＥＰＬＹ＿ＣＯＭＭ．ＳＥＮＤ（－－－）

を行ない，一時停止状態となる。

②送信サイト上の通信モジュール（ＲＥＰＬＹ＿ＣＯＭＭ）は accept を実行することによりセッションを確立する。サイト間における実際のメッセージ転送はＬＡＮインタフェース中の

ＲＥＭＯＴＥ＿ＳＥＮＤ（－－－）

の実行により行なわれる。

③受信サイト上の通信モジュール(ＲＥＰＬＹ－ＣＯＭＭ)は

ＲＥＭＯＴＥ＿ＲＥＣＥＩＶＥ（－－－）

を実行後，エントリ呼び出しを行ない，一時停止状態となる。

④受信アクティビティは accept を実行し，要求された仕事を行なう。

⑤受信サイト上の通信モジュール(ＲＥＰＬＹ＿ＣＯＭＭ)は

ＲＥＭＯＴＥ＿ＳＥＮＤ（－－－）

により，この仕事の結果を送信サイトへ転送する。

⑥送信サイト上の通信モジュール（ＲＥＰＬＹ＿ＣＯＭＭ）におけるｅｎｄの実行によりセッションが解放される。

ＣＩＭ用分散システムにおけるアクティビティ間通信機能としては，同期通信（ＲＰＣ）が重要な手段となる。

4．3　非同期通信

送信アクティビティは，通信モジュールへのメッセージ転送が終われば直ちに実行を再開する。すなわち，送信アクティビティが一時停止状態になるのは，送信サイト上の通信モジュールとのランデブー中だけである　（図４－４）。非同期通信では，メッセージを一時的に貯えるためのバッファ機能が通信モジュールで必要となる。通信モジュールには非決定性の制御構造（select文）が含まれていることに注意（図4-5，図4-6）。

5．ＬＡＮインタフェース

サイト間の通信機能は，

- 上位レベルでは，アクティビティにおいて扱われる応用アルゴリズムに依存し，
- 下位レベルでは，処理系（通信媒体を含む）に依存する。

※　ブロードキャスト通信については省略する。

サイト間の通信機能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，プロトコル，通信媒体の信頼性に対処できるためには多くの未解決な問題がある([14],[15],[23])。

ソフトウエアのポータビリティを高め，応用アルゴリズムおよびプロトコルの変更，または通信媒体における信頼性の相違に対処できるようにするのが望ましい。Ａｄａのパッケージ機構を用いることにより，それぞれの用途に応じた通信機能を実現することができる ([8])。

サイトと通信媒体とのインタフェース（物理層およびデータリンク層の機能）の標準化としては，

ＣＳＭＡ／ＣＤ　　（ＩＥＥＥ　802.3）
ＴＯＫＥＮ　ＢＵＳ　（ＩＥＥＥ　802.4）
ＴＯＫＥＮ　ＲＩＮＧ（ＩＥＥＥ　802.5）

等があり，これらの機能はＶＬＳＩチップとして提供されつつある。各方式には一長一短があるが，ＣＩＭ用としてはＴＯＫＥＮ　ＢＵＳ　方式が有望視されている。

これらの支援に基づいて，ＬＡＮインタフェースはトランスポート層以下の機能を提供する。効率の良い通信機能を実現するためには，さらにトランスポート層までＶＬＳＩチップ化するのが望ましい。ＬＡＮインタフェースはこのようなＶＬＳＩチップとのインタフェースを提供する。

６．異常検出と例外処理

ＣＩＭ用コンピュータ・システムにおいては，高信頼性，高稼動率が重要であることは既に述べた。このためには，通信媒体自身の高信頼化を計ると同時に，異常検出機能および回復機能（例えば，例外処理機能）が要求される。

Ａｄａには例外処理機能が備わっているが，このためには例外の発生を見知することが必要となる。しかし，サイト外で発生した異常事態を検出する方法については言語規定では全く触れられていない。

本節では，他のサイトにおける異常事態を検出する一方法について述べる。簡単のためＡ，Ｂ二つのサイトの場合について考える。

- サイトＢはサイトＡへある周期（ＴＰ）でメッセージを周期的に送る。
- サイトＡはサイトＢからメッセージを受信する毎にタイマーを始動させる。

タイムアウト（ＲＰ）が過ぎてもサイトＢよりメッセージが到着しない場合には，サイトＢは異常状態（ＤＯＷＮ）である（とサイトＡは見なす）。タイムアウト（ＲＰ）以内にメッセージが到着した場合には，サイトＢは正常状態（ＵＰ）である。このことは，サイトＡの状態表に記録される。状態の変化が検出される毎にこの表は更新され，常に最新の状態表が保持される。

ＤＭＡＸ：　サイト間におけるメッセージの平均最大遅延時間

ＤＭＩＮ：　サイト間におけるメッセージの平均最小遅延時間

とすると，

$$RP = TP + DMAX - DMIN$$

である。

通信モジュールとのインタフェース　ＣＯＭＭ＿ＩＮＴＥＲＦＡＣＥの仕様を図６－１に示す。

```
REMOTE_SEND (
    DESTINATION: in  SITE_ID;
    M  :  in MESSAGE;
    S  :  out STATE) ;
```

ＳＩＴＥ＿ＩＤで指定された受信サイトの状態がＤＯＷＮの場合には，例外が発生する。　受信サイトの状態がＵＰの場合には，メッセージを送り出すための動作が開始される。

```
REMOTE_RECEIVE(
    SOURCE : in SITE_ID;
    M  :  out  MESSAGE;
    S  :  out STATE) ;
```

ＳＩＴＥ＿ＩＤで指定された送信サイトの状態がＤＯＷＮの場合には，例外が発生する。送信サイトの状態がＵＰの場合には，メッセージを受け取るための動作が開始される。

例外が発生した場合には，通信モジュールにおけるEXCEPTION HANDLERにより対応策が講じられる。以上の機能を用いることにより，同期通信（ＲＰＣ）における“all-or-nothing”の機能を実現することが可能になる。

7．あとがき

ＣＩＭの基盤技術（“infrastructure”）とも云うべき分散オペレーティング・システムのニーズ，特徴，要求される機能を論じ，Ａｄａによる実現法の概略・問題点を示した。すなわち，

- アクティビティ（アクティブ・オブジェクト）のＡｄａタスクによる表現法
- アクティビティ間の同期通信，非同期通信を行なうための制御構造
- 各用途に適したサイト間通信機能を実現する方法
- トランザクション機能実現に要求される異常検出機能

について論じた。

ＣＩＭ用コンピュータ・システムの死命を制する分散ソフトウエアの実用化のためには，今後さらにハード・ソフトの両面に渡る多くの研究課題が解決されねばならない。

謝　辞

本研究の一部は，科研費（５９４６０１２０）による。本研究グループの各位に深甚の謝意を表する。

参考文献

[1] Bernstein, A. & P. Harter: "Proving Real-Time Properties of Programs with Temporal Logic," **Proc. 8th ACM-SIGOPS Symp.**, pp. 1 - 11, (1981).

[2] Cheriton, D.R. & W. Zwaenpoel: "The Distributed V Kernel and its Performance for Diskless Workstations," **Proc. 9th ACM-SIGOPS Symp.**, pp. 128 - 140, (1983).

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

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

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

[6] 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).

[7] 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).

[8] 藤田昭平：Ａｄａによるリアルタイム・プログラミング，コンピュートロール，No. 6, pp.131-135, (1984).

[9] 藤田昭平：分散ワークステーション上のＡｄａ一事例研究，電子通信学会技術研究報告（電子計算機），Vol. 84, No. 295, pp.13－24, (1985).

[10] 藤田昭平：ＣＩＭにおける分散オペレーティング・システム，コンピュートロール，No. 11, pp.90-95, (1985).

[11] Kahn, K.C. et al.: "iMAX: A Multiprocessor Operating System for an Object-Based Computer," **Proc. 8th**

ACM-SIGOPS Symp., pp. 127 - 136, (1981).

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

[13] Le Lann, G.: "On Real-Time Distributed Computing," **Proc. IFIP 83**, pp. 741 - 753, (1983).

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

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

[16] Mok, A. K_L: "Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment," **MIT/LCS/TR-297**, (1983).

[17] Neumann, P.G. & L. Lamport: "Highly Dependable Distributed Systems," **SRI Int.** Project 4180, (1983).

[18] Popek, G. et al.,: "LOCUS: A Network Transparent, High Reliability Distributed System," **Proc. 8th ACM-SIGOPS Symp.**, pp. 169 - 177, (1981).

[19] Rescher, N. & A. Urquhart: **Temporal Logic.** Springer Verlag, (1971).

[20] Softech. Inc.: **Ada Compiler Validation Summary Report.** (1983).

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

[22] 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).

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

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

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

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

[27] Western Digital Cor.: **WD2511/WD2840 Technical Package.**, (1984).

[28] Wirth, N.: "Toward a Discipline of Real-Time Programming," **Comm. ACM**, Vol. 20, No. 8, pp. 577 - 583, (1977).

[29] Xerox: **The Ethernet: A Local Area Network - Data Link Layer and Physical Layer Specifications.** Ver.1.0, (1980).

[30] 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).

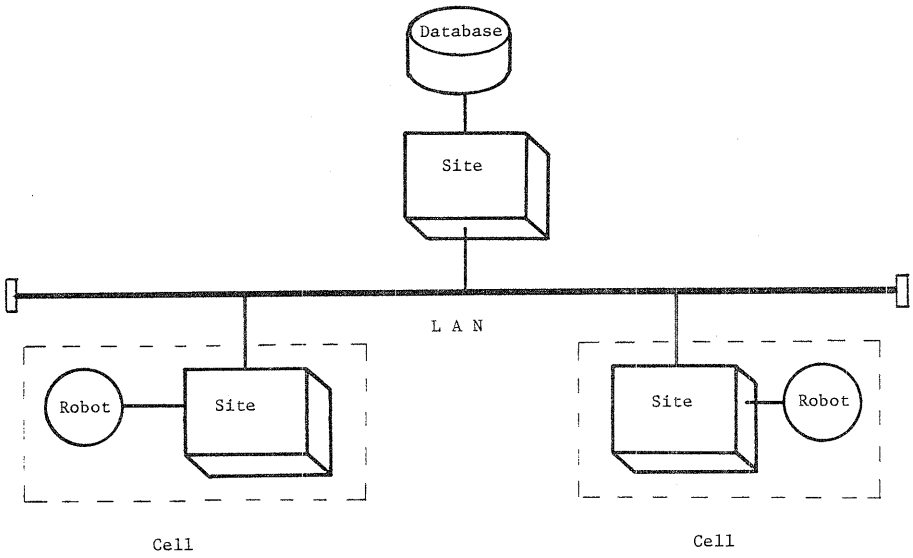

Fig. 1-1 C I M

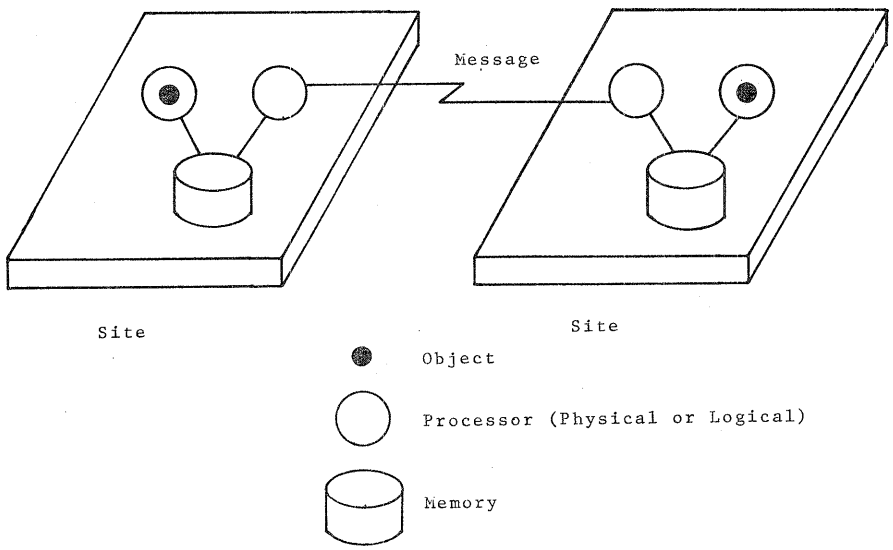

Fig. 2-1 Cooperative Distributed System

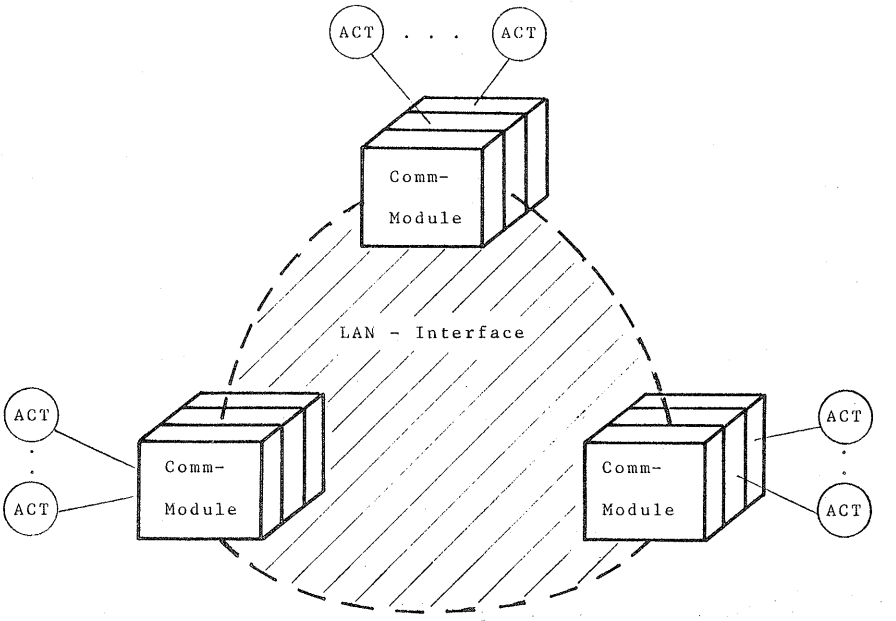

Fig. 4-1 Communication Modules

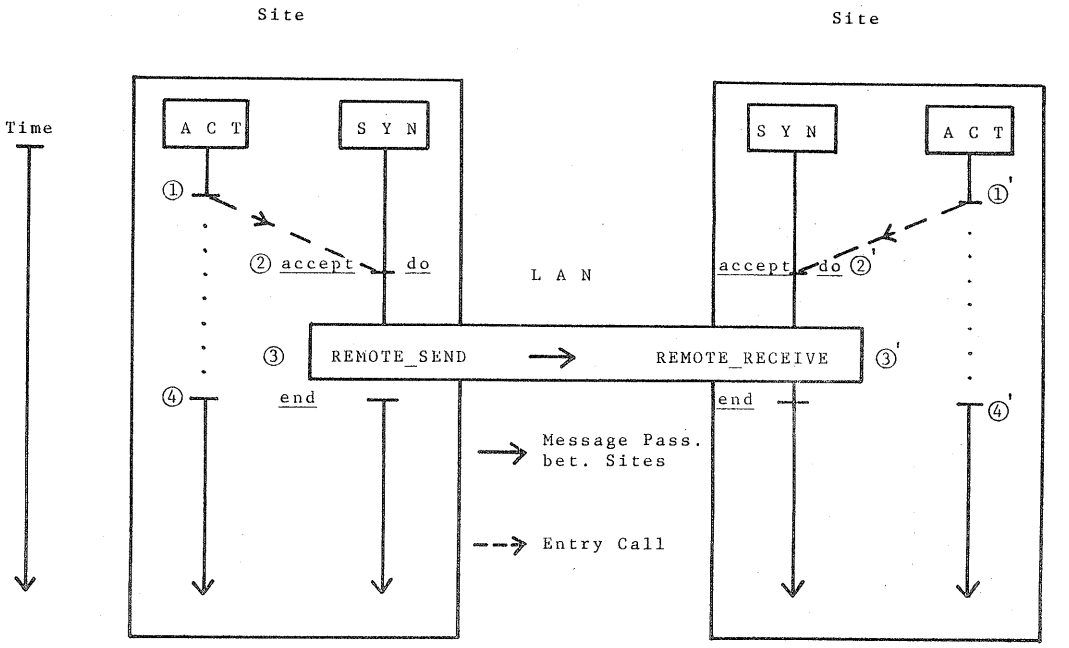

Fig. 4-2 Synchronous Communication

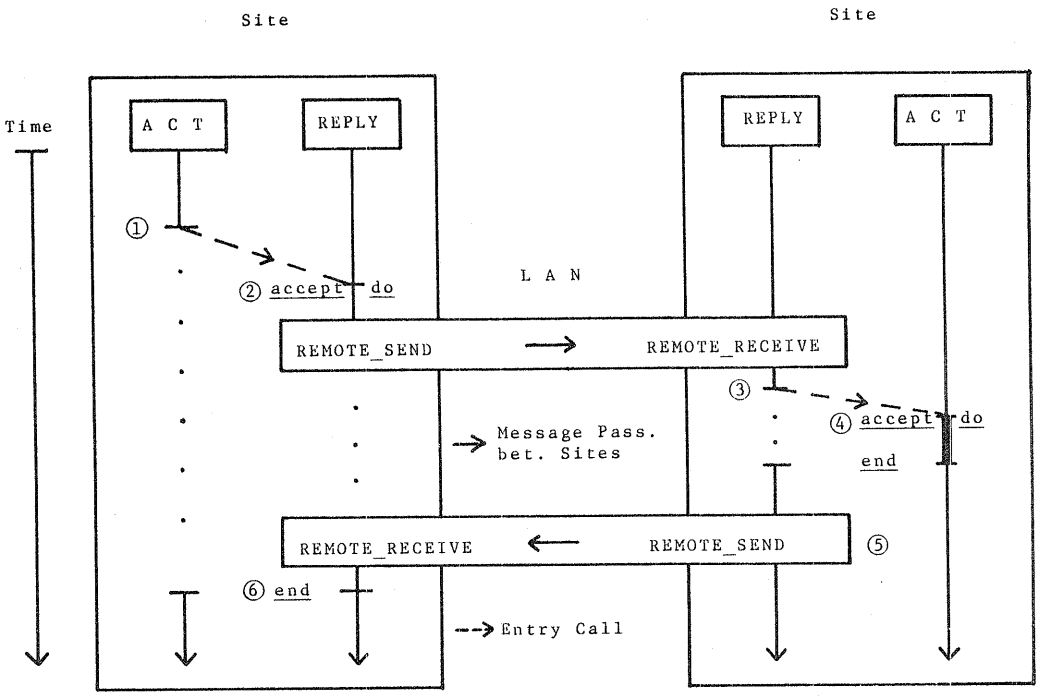

Fig. 4-3 Synchronous Communication (RPC)

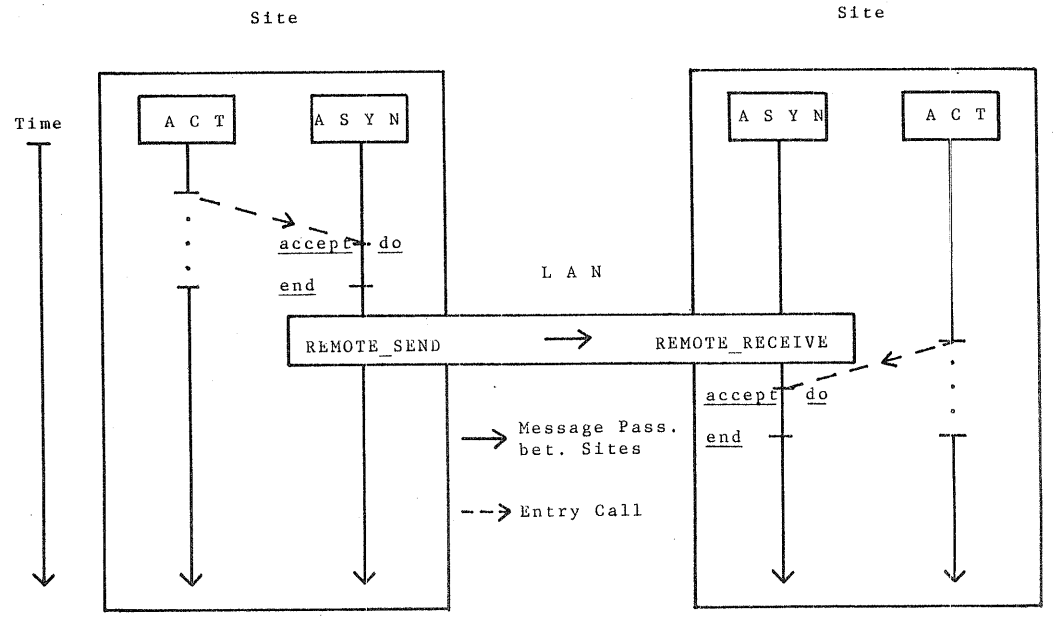

Fig. 4-4 Asynchronous Communication

```
task ASYN_COMM is
   entry SEND (M : in MESSAGE);
end ASYN_COMM;

with BUFFER; use BUFFER;
with COMM_INTERFACE; use COMM_INTERFACE;
task body ASYN_COMM is
          .
          .
begin
   loop
      select
         when not BUFFER_FULL =>
            accept SEND (M : in MESSAGE) do
               BUFF := M ;
            end SEND;
          .
          .
      or
         when not BUFFER_EMPTY =>
            delay T;
            REMOTE_SEND (BUFF);
            --  provided by COMM_INTERFACE
          .
          .
      end select;
   end loop;
end ASYN_COMM;
```

Fig. 4-5 Asynchronous Communication Module

```
task ASYN_COMM is
   entry RECEIVE (M : out MESSAGE);
end ASYN_COMM;

with BUFFER; use BUFFER;
with COMM_INTERFACE; use COMM_INTERFACE;
task body ASYN_COMM is
          .
          .
begin
   loop
      select
         when not BUFFER_FULL =>
            delay T;
            REMOTE_RECEIVE(BUFF);
            --  provided by COMM_INTERFACE
           .
           .
      or
         when not BUFFER_EMPTY =>
            accept RECEIVE(M : out MESSAGE) do
               M := BUFF;
            end RECEIVE;
          .
          .
      end select;
   end loop;
end ASYN_COMM;
```

Fig. 4-6 Asynchronous Communication Module

```
with LAN_INTERFACE; use LAN_INTERFACE
package COMM_INTERFACE is

          .
          .
   type STATE is (UP, DOWN);

   procedure REMOTE_SEND (
      DESTINATION : in SITE_ID;
      M : in MESSAGE;
      S : out STATE);

   procedure REMOTE_RECEIVE (
      SOURCE : in SITE_ID;
      M : out MESSAGE;
      S : out STATE);

end COMM_INTERFACE;
```

Fig. 6-1 Communication Interface