普 及 課

50－S003

# コンピュータ・ネットワークJIPNETの研究開発

昭 和 51 年 3 月

JIPDEC

財団法人 日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和50年度に実施した「コンピュータ・ネットワークシステムの研究開発」の成果をとりまとめたものであります。

# 序

当財団は情報処理技術の研究開発の一環として，昭和48年度より「コンピュータ・ネットワーク・システム」の研究開発に着手いたしました。

複数個のコンピュータを相互に連結し，ハードウェア，ソフトウェアおよびデータ・ベースの共用を目的とする，リソース・シェアリング・コンピュータ・ネットワークの開発は，米国におけるARPANETの成功をきっかけに，現在世界的に急速に進展しつつあります

我が国においても，電電公社をはじめ，官庁，大学，各種企業等の多くの機関でそれぞれの立場に応じた研究開発が実施されつつありますが，実用的なネットワーク・システムを構築するには，まだ相当の技術の蓄積と経験が必要であると考えられます。

当財団は異なったメーカの複数台のコンピュータを有しており，それらの相互間リソースを共有するための技術的問題点を具体的なインプリメントを通して解明することを目的として本研究開発を進めてまいりました。

本年度は，前年度までに開発を進めてきた基本ファシリティとしてのネットワークを更に発展させ，各種プロトコルの改良，ネットワーク・コントロール機能の効率評価等を行うとともに，異機種間のデータや各種リソースの相互利用に際する技術的問題等，コンピュータ・ネットワークにおける今後の新しい技術分野について検討を行い，本書はそれらの成果をまとめたものであります。

本報告書が，この方面に興味ある方々に広く利用され，我が国情報処理技術向上の一助として寄与できることを念願いたす次第であります。

昭和51年3月

財団法人　日本情報処理開発協会

会長　　植村甲午郎

# コンピュータ・ネットワークJIPNETの研究開発

## 目　　次

# ま え が き

1970春，米国の代表的なコンピュータ・カンファレンスであるSJCCに於て，数個の論文からなるARPANETの成果が発表されて大きな反響を呼び，これを契機としてコンピュータ・ネットワークの実用の可能性への第一歩が大きく踏み出された。

ARPANETの創始者であるL．G．Robertsは「コンピュータ・ネットワークとは，お互いにその資源（Resource：ハードウェア，ソフトウェアおよびデータ）を共用（Sharing）し合うことができるように結合され，それぞれ独立した機能を持つコンピュータ・システムの集まり」と定義している。

コンピュータ・ネットワークの基本的に意図するところは，各種のコンピュータのそれぞれ特色あるリソースを相互に共有し合うことによりハードウェア，ソフトウェア，あるいは，データベースにかかわる種々の二重投資を回避するとともに，総合的な強力なリソースの利用を可能とし，且つトータルな信頼性の強化，異機種間の互換性の確立，システム間相互の負荷分担による総合的な処理効率や稼動率の向上，あるいはリソースの分散によるセキュリティ機能の強化等，各種のメリットを産み出そうというものである。

ARPANETの成功をきっかけに，現在世界各国で競ってコンピュータ・ネットワークの計画，開発が行われつゝある。我が国に於ても電々公社をはじめ，大学，官庁，各種企業等，多くの機関でそれぞれの立場に応じた研究開発が実施されつゝあるが，JIPNETは当財団が数個の異った種類のコンピュータ・システム－FACOM, HITAC, NEAC －を持つ立場を利用して，それらの相互間のリソース共用のための技術的問題点を具体的なインプリメントを通して解明する事を目的としている。

本プロジクトは1973年度より着手し，1974末にはデータ交換用のサブネットと，それを通しての2つのHOSTコンピュータ間でのコミュニケーションが可能となり，これを第一バージョンとして，その後各種コントロール機能の効率測定をおこない，その結果によるシステムの評価改良，信頼性の向上，1975年3月にリプレースされた新しいHOSTコンピュータの結合等を含め，1975年度中にバージョン2が完成し，外部への拡張も可能となった。

本報告書は73年度および74年度の報告書の内容を更に発展させ，各種プロトコルの改良，ネットワーク・コントロール機能の効率評価の結果等を報告するとともに異機種間のデータや各種リソースの互換性の確保や相互利用に際しての技術的検討等，コンピュータ・ネットワークにおける今後の新しい技術分野の話題を中心にまとめたものである。

# 1. ＪＩＰＮＥＴの概要

# 1. JIPNET の概要

## 1.1 目　的

JIPNET（JIPDEC Integrated Project Network）は実用と実験の両目的を持つ分散型のリソース・シェアリング・ネットワークである。ネットワークの規模は当初当協会内の国産三機種FACOM，HITAC，NEACを結合したローカルなものではあるが，将来は端末やNAP（Network Access Processor）による外部からのアクセス，また外部システムとの結合等の拡張も考慮している。JIPNETにおける実用上の要求は，地理的に離れている部門からの三機種の総合リソースの利用，三機種間のハード，ソフト，データファイル等の可能な範囲での互換性の確立，2機種以上の協業による特殊ジョブの処理，漢字入出力，マークシートリーダ等のペリフェラルの各CPUからの共用等であり，実験的な目的としては，必要とするユーザレベルのプロトコルの種類，HOST内のネットワーク・コントロール機能の効率やオーバヘッドの測定，コンピュータ・ネットワークのコスト効率やパフォーマンスの評価，異機種間の分散型データ・ベース・マネージメントあるいは仮想マシーンとしてのネットワークの有効性の検討等を具体的なインプリメントを通して行なうものである。

コンピュータ・ネットワークの本質は，夫々特色ある各種リソースを相互に有効に利用することにある。真に効果のあるリソースの共用を実現するには，夫々のHOST側でのリソース提供のサポート機能が，経済性，効率性等の諸点からも充分納得し得る質を保つ事が要求される。しかしながら，従来のOSとの関連もあり，現在のネットワークに於てはこの点に関しては必ずしもまだ充分満足し得る状態にあるとは言い難い。JIPNETに於ける主たる目標は，この事実に焦点を当て，特に国産コンピュータを対象とし，今後我国に於けるコンピュータ・ネットワーク利用の実現可能性と実用性を検証し，その促進のための考察を具体的なレベルで行うことにある。

## 1.2 システム構成

JIPNETは分散型ネットワークのモデルとして現在は図1－1に示すように，3つのHOSTコンピュータとNEAC 3200／50をNODEプロセッサとする3NODEのサブネットおよび各種の最新端末群をNODEに接続した形で形成されている。NODE間は48Kbpsの高速通信回線で接続し，HOST－NODE間は8ビット・パラレル転送のインタフェース・アダプタを製作し，セレクタ・チャネルにより結合している。またNODE 2に結合されているU 200はNAP（Net-

work Access Processor）と呼びARPANETのANTS（ARPA Network Terminal System）に相当する。

システムの構成にあたって，以下のような諸点に留意した。

(1) コンピュータ・ネットワークとしての基本的な機能が全て実験可能な構成であること。

(2) コンピュータ・コンプレックス・システムとして，当センター特有の利用法の実験にも適した構成であること。

(3) 我が国での実用性が比較的高いと思われるARPANETでいうTIP（Terminal IMP）あるいはANTS等を重視した形をとった。

(4) 将来，他のシステムあるいはネットワークとの結合などの拡張が容易であること。

(5) 従来，各HOSTコンピュータで行ってきた作業が，何等変更なしに引き続き稼動可能のこと。

なお現在のソフトウェアの変更なしに16NODEまでは拡張可能で，各NODEには3つまでのHOSTが結合し得る。

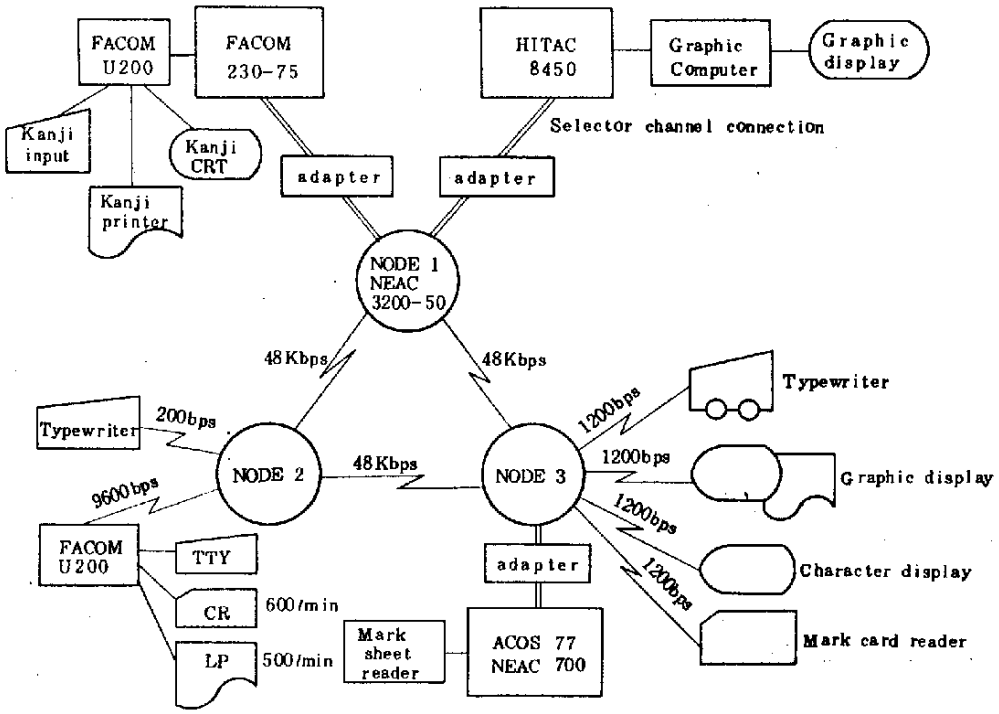



図1－1 JIPNET構成図

## 1.3 JIPNETの機能

JIPNETにおける主な処理パターンの例を図1－2に示す。①～④は2つのHOSTを対象としたもので，例えば

| | HOST A | HOST B | HOST C | HOST D | 端 末 |
|---|---|---|---|---|---|
| ① | S, O | E, P, D | | | |
| ② | S, O, D | E, P | | | |
| ③ | S, O, D, SP | E, C | | | |
| ④ | S, O, C, E | SP, D | | | |
| ⑤ | S, O | E, P | D | | |
| ⑥ | S, O, D | E, C | SP | | |
| ⑦ | S, O | SP, D | E, C | | |
| ⑧ | S, O, SP | D | E, C | | |
| ⑨ | S, D | E, P | O | | |
| ⑩ | T, D | E, P | | | |
| ⑪ | T | E, P, D | | | |
| ⑫ | T | E, P | D | | |
| ⑬ | E, P, D | | | | T |
| ⑭ | E, P | D | | | T |
| ⑮ | E, P | D1 | D2 | | T |
| ⑯ | T | E, P | D1 | D2 | |
| ⑰ | S, O | E1, P1 | E2, P2 | D | |

S：制御情報の入力
O：結果の出力
E（またはE1，E2）：実行
C：コンパイルまたはアセンブル
SP：ソース プログラム ファイル
P（またはP1，P2）：実行形式
プログラム ファイル
D（またはD1，D2）： データ
ファイル
T：端末（SとOを含む）

図１－２　処理パターンの例

①はＨＯＳＴ Ａから制御情報をＨＯＳＴ Ｂに送り（Ｓ），ＨＯＳＴ Ｂにあるプログラム（Ｐ）およびデータファイル（Ｄ）を使用して処理（Ｅ）を行うことを依頼し，結果をＨＯＳＴ Ａに戻し出力（Ｏ）するケースである。また②はＡからＢに同様に処理を依頼するが，その際，データファイルがＡにあり，これを予め，あるいは処理中にＢに送りながら処理を実行する。また③はその際ソースプログラムファイル（ＳＰ）もＡにあり，これをＢに送りコンパイル（Ｃ）も依頼する。

次の⑤から⑨は，Ａ，Ｂ，Ｃの３つのＨＯＳＴにわたって処理を行なう場合の例で，たとえば⑤は，Ｃにあるデータファイルを使用してＢにあるプログラムで処理をすることをＡから依頼する形である。また⑨は，Ｃにある特殊出力装置（たとえば漢字プリンタ）を共用するような場合の例で，ＡにあるデータをＢに送って処理を行ない結果をＣに出力する。

⑩から⑭は，リモートバッチや会話型端末からネットワークにアクセスする形で，端末（Ｔ）はここでは制御情報の入力（Ｓ）と結果の出力（Ｏ）の両機能を含むものとする。

このうち⑩から⑫は，あるＨＯＳＴ（この場合Ａ）につながる端末から他ＨＯＳＴ（この場合Ｂ）の会話型あるいはリモートバッチ処理の使用であり，⑬，⑭は特定のＨＯＳＴを経由せず，ＮＯＤＥに直接結合された端末からのネットワーク使用である。

⑮，⑯，⑰は，ネットワークのやや進んだ使用法であり⑮，⑯は２つのＨＯＳＴにデータファイルが分散しており（Ｄ１とＤ２），⑰は処理自体が２つのＨＯＳＴで分担（Ｅ１とＥ２）されている。これらはそれぞれ，分散型データベースと複数ＨＯＳＴの協業により，１つのジョブを実行する形の例であり，より多くのＨＯＳＴを対象とすることももちろん考えられる。

## 1.4　プロトコル

図１－３にプロトコルの階層を示す。

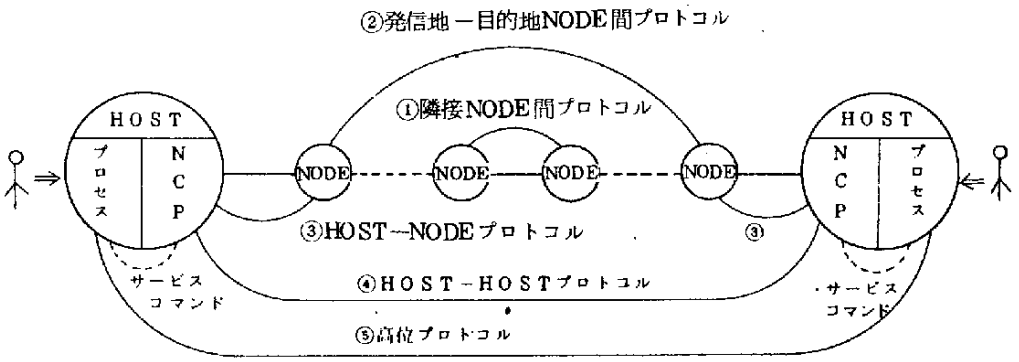



図１－３　ＪＩＰＮＥＴのプロトコル

①　隣接ＮＯＤＥ間プロトコル

隣接したＮＯＤＥ間での伝送データのフォーマット，受信確認方式，伝送エラーの検出と処置，再送の手続，パケットのルート選択および相互の障害と回復の検出と処置等に関する規約がある。

②　発信地ＮＯＤＥ ― 目的地ＮＯＤＥ間プロトコル

発信地と目的地間のＥｎｄ－ｔｏ－Ｅｎｄ処理にともなう規約であり，メッセージのパケット分割やリアセンブル（ｒｅａｓｓｅｍｂｌｅ），メッセージの紛失，重複等に関する伝送エラーの検出と処置，サブネット内のフローコントロール，伝送されたメッセージの順序付けなどに関する規約がある。

③　ＮＯＤＥ－ＨＯＳＴプロトコル

このプロトコルはサブネットとＨＯＳＴの間のデータの授受に関する規約であり，データフォーマット，送受信の手順および確認の方法，相互の障害状態の検出などに関するものがある。

④　ＨＯＳＴ－ＨＯＳＴプロトコル

このプロトコルは，ＨＯＳＴどうしが交信し，相互にコントロール情報およびデータを転送し合い，ネットワークを通して各種の処理を行なうための基本となる規約である。たとえば，ＨＯＳＴ間のデータフォーマット，相互交信のためのＨＯＳＴ間のコネクションの確立，コネクションを通したデータの転送，データ転送に伴うＨＯＳＴ間のフローコントロール，相手側の障害の検出などに関するものである。

⑤高位プロトコル

ネットワークユーザが直接使用するコマンドの形で規定されるハイレベルのプロトコルで，現在DSP(Demand Sevicce Protocol)，RBP(Remote Batch Protocol)およびFTP(File Transfer Protocol)をサポートしている。

これらのうち①～④はシステム・レベルのプロトコルである。これらのプロトコルはNODEプロセッサ内のSCP(Subnet Control Program)と各HOST内のNCP(Network Control Program)により分担して処理されるが，プロトコルの設定におけるHOSTとサブネットの負荷分担は，HOSTの負荷軽減を考慮し，メッセージのパケット化，リアセンブリ，伝送エラーのリカバリ，シーケンジング等，End－to－Endの殆んどをサブネットの分担としている。

## 1.5 サブネット(Subnet)

パケット交換方式を採用しており，ネットワーク全体の信頼性，会話型処理および大量データ転送における効率性，HOSTやNODE数の増加，任意の回線速度等に応じられる拡張性および可変性を重視し設計した。全体的にARPANETの実績をかなり利用したが，各NODEにサブネットのトポロジを意識させている点，長短メッセージのパイプ管理，ルーティングのアルゴリズム，発信地，目的地間のメッセージのACKをHOST間で行なっている点等に関しては，かなり異なっている。Artificial Traffic Generaterを用いて測定した結果，最大スループットは単一パケット・メッセージ(144byte以下)の場合，58Kbit／sec，多重パケット・メッセージ(1152byte以下)の場合，64Kbit／secであった。またround trip timeは100byteメッセージの場合1Hopで27ms，2Hopで52msであり，20Hopでは約0.5秒と推定される。

以下に各機能の概略を述べる。

(1) 蓄積交換機能

サブネットはHOSTから受取ったメッセージをパケット化し，蓄積交換を行って目的地NODEに届ける。目的地NODEではこれらのパケットをリアセンブルし，元のメッセージに再組立し，目的地HOSTに渡す。隣接NODE間の伝送確認は基本的にはパケット単位のACK信号によって行われ，両HOST間のメッセージの伝送確認はRECEIVE信号によって行われる。隣接NODE間は回線を論理的な4チャネルに分割し，回線の利用率を上げている。なおACK，RECEIVEの返送は原則として逆方向のメッセージに便乗させている。

(2) ルーティング

ルーティングの手法には種々のものがあるが，パケットの高速伝送，トラフィック変動に対する適応性，トポロジー変化への追従性，NODEの計算負荷の軽減等を考慮して，「Shortest Q＋Bias」の手法を採用した。この手法は定常状態では外部からは情報をとり入れずに，NO

DE内部における動的な送信待ちキューの長さと，ネットワークの静的構造を示すバイアス値との和を各出力回線ごとに比較し，その最小のものを選びパケットを送り出すというものである。バイアス値は，NODEおよび回線のアップ・ダウンにともなうサブネットの重大な変化の際のみ動的に計算し更新している。

(3) フロー・コントロール

フロー・コントロールはARPANETのパイプの概念を採用し，発信地 — 目的地NODE間にパイプという論理的なパスを設定し，このパイプの容量を制限することによりフローを制御している。JIPNETでは単一パケット・メッセージ（144バイト以下）用のS－Pipe と多重パケット・メッセージ（1152バイト以下）用のL－Pipe との2種類のパイプを設け，夫々別個に管理している。現在S－Pipe には4個までの単一パケット・メッセージを同時に流すことができ，多重パケット・メッセージは目的地においてバッファがアロケートされた事を確認した後L－Pipe を通し，一時に一個だけ流すことができる。これにより，短かいメッセージの応答性が長大メッセージのディレイにより受ける影響を少なくしている。

(4) シーケンシングとエラー検出

サブネットに入る各メッセージには，各パイプごとに0～255までの番号がラウンディング（rounding）につけられて送り出される。この番号により目的地NODEではメッセージのシーケンシング（sequencing）を行っており，発信地 — 目的地NODE間でのメッセージの紛失，重複あるいはRECEIVEの紛失等のエラー検出にもこのシーケンシング機構を利用している。

(5) ロック・アップ

サブネットでは次の4種のロック・アップ（lock up）を想定し対策を考慮した。

- store & forward lock up
- sequencing lock up
- reassemble loek up
- HOST／NODE lock up

これらのlock upに対してはNODE内におけるバッファの管理アルゴリズムを十分に検討して防御手段を講じた。なお，sequencing lock upは一つの目的地にメッセージが集中する場合に発生するものであり，HOST／NODE lock upはHOST－NODE間のアダプタが半二重のために発生するものである。

(6) 障害対策

サブネットの障害対策は次のような点を基本方針としている。

- できる限り早期に発見する。
- 障害の発生およびその回復過程で，サブネット全体の運転は中断させない。

- 一部の構成要素が障害で失われても，それを除いた環境下での運転が可能なこと。
- 障害の回復は自動的に行われる。
- 異なった要素間における障害の切りわけが容易であること。
- ＨＯＳＴ障害の影響をネットワーク全体に及ぼさぬとともにネットワークの障害を各ＨＯＳＴのローカル処理に影響させない。

表１－１に障害の検出と処置の一覧表を示す。

表１－１　障害の検出と処置

| 障害の種類 | 検　出　法 | 処　置 |
|---|---|---|
| 回線障害 | ハードウエア割込みによる | 他ＮＯＤＥに通知，診断パケットの応答が８０回あれば回復とみなす。 |
| 隣接ＮＯＤＥ障害 | 引き続く２０パケットの伝送に対し応答のない場合 | 同上，障害隣接ＮＯＤＥ宛のパケットは棄却し，他のパケットはreroutingを行う。 |
| ディスコネクションＮＯＤＥ | コネクション・マトリックスの演算による | 自ＨＯＳＴに通知し，そのＮＯＤＥ宛のメッセージ送出を一時中止 |
| ＨＯＳＴ障害 | ＮＯＤＥからＨＯＳＴへのＷＲＩＴＥコマンドが10秒以内に終了せぬ時 | そのＨＯＳＴへ送出するメッセージを棄却し，発信地へDOWN RECEIVEを返す。 |
| メッセージの紛失 | 4／16秒以内にＲＥＣＥＩＶＥが返送されぬ時 | 紛失メッセージ番号を付して目的地にInquiry packetを送る。 |
| メッセージの重複 | メッセージ番号（０～255）により目的地で検出 | 先の到着したものを生かし棄却 |
| ＲＥＣＥＩＶＥの紛失 | 発信地からのInquiry packetにより知る | 既に該当メッセージを受取っていればＲＥＣＥＩＶＥを再送し，まだならばメッセージの再送要求を出す。 |
| バッファの充満 | バッファ管理メカニズムによる | ロック・アップ防止のため一定量のバッファを保留しそれ以上のパケットは棄却その旨発信地に通知 |

## 1.6　Ｎ　Ｃ　Ｐ

ＮＣＰ（Network Control Program）は各ＨＯＳＴに存在するネットワーク用のコントロール・プログラムである。

ＪＩＰＮＥＴは汎用のネットワークであり，ＨＯＳＴ間の通信は各処理プロセス間通信の形で実現している。即ち図１－４に示す様に各プロセスは夫々のＨＯＳＴのＮＣＰを通して交信をおこなう。

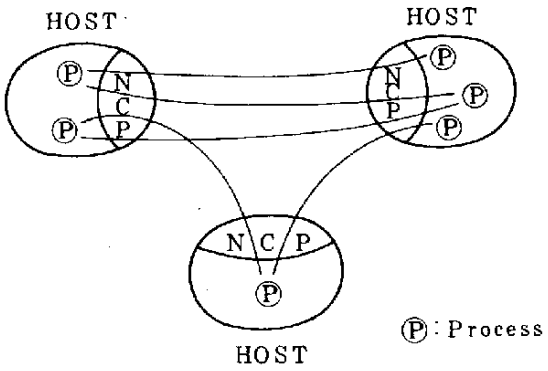



図１－４　プロセス間コミュニケーション

各ＮＣＰは既存ＯＳのもとに付け加えられ，ＦＡＣＯＭではサブモニタとして，他の２機種ではユーザ・プログラム，即ちＨＩＴＡＣではリアルタイム・ジョブとして，ＡＣＯＳでは特権スレーブとして作成している。

ＮＣＰはＨＯＳＴ－ＨＯＳＴおよびＨＯＳＴ－ＮＯＤＥプロトコルの規定にもとづく処理をおこなうものであり，次のような機能を持つ。

(1)　プロセス間のコミュニケーション

バーチャル・コール（Virtual Call）方式にもとづき，プロセス間のコネクションの確立とその閉鎖，コネクションを通してのデータの授受，プロセス間の同期などの処理を行なう。コネクションの確立とは，２つのＨＯＳＴのそれぞれのプロセス間で，データ流のためのロジカルなパスを作ることである。各プロセスにはＨＯＳＴ番号とユーザ番号とからなるネットワークに対してグローバルなidがつけられ，ＪＩＰＮＥＴではこれをport－idと呼ぶ。１つのコネクションは２つのプロセスのそれぞれのpord－idのペアに対して確立される。

また１つのコネクションで両方向のデータ流を許す。

(2)　ＨＯＳＴ間のフロー・コントロール

バッファ確保方式を採用している。ＨＯＳＴ間で１つのコネクションが確立すると，受信側ＨＯＳＴはメッセージを受信するためのバッファを１個以上（各ＨＯＳＴのその時点のバッファ事情により2～10個）確保してその個数を送信側ＨＯＳＴに通知する。送信側は受信側において準備されたバッファ個数分のメッセージ送信を完了すると，新たにバッファ確保の通知がくるまでそのコネクションに対するメッセージ送信を一時中断する。受信側は出来るだけ速かにバッファを確保し，送信側に通知する。

１つのＨＯＳＴで１時に確立し得るコネクション数は30前後であり，ＨＯＳＴあたり100ないし150パケット分のバッファを持つ。

(3) コントロール・コマンドとサービス・コマンド

2つのHOSTのNCP間は，コントロール・コマンド，即ちHOST－HOSTプロトコル・コマンドにより交信される。一方ユーザ・プロセスとNCP間にはサービス・コマンドが設けられており，サービス・コマンドには多重コネクション管理，フロー・コントロールなどのマクロ機能を含んでいる。表1－2に両コマンドの種類を，図1－5に両者の関連の例を示す。(b)はMがユーティリティ・プロセスの場合である。

表1－2 コントロール・コマンドとサービス・コマンド

| コントロール・コマンド | | サービス・コマンド | |
|---|---|---|---|
| RFC | コネクション確立要求 | OPEN | コネクション確立要求 |
| CLS | コネクションの閉鎖 | CLOSE | コネクションの閉鎖 |
| ALLOC | メッセージ・バッファ確保通知 | READ | データの受信 |
| POST | プロセス間の情報の通知 | WRITE | データの送信 |
| SPOST | システム情報の通知 | LISTEN | コネクション確立要求待ち |
| LOADE | プロセスの起動 | WAIT | イベント待ち |
| DELP | プロセスの消滅 | POST | プロセス間の情報の通知 |

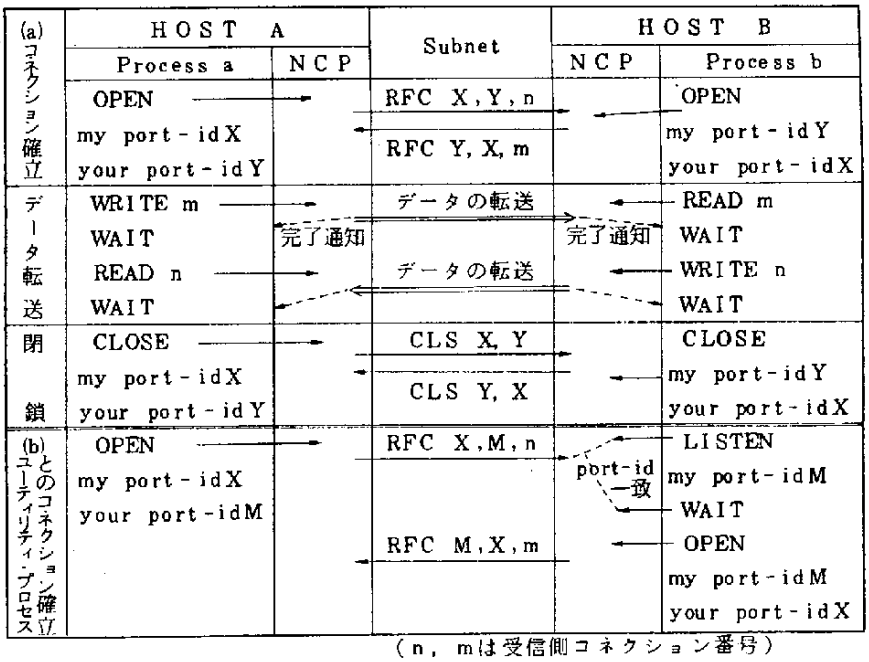



図1－5 両コマンドの関連

## 1.7　高位プロトコルの処理

現在は，DSP（Demand Service Protocol），RBP（Remote Batch Protocol）およびFTP（File Transfer Protocol）がサポートされている。

各HOST内のプロセスにはユーザ・プロセスとシステムレベルのユーティリティ・プロセスとを設け，高位プロトコルの処理は拡張性を重視し，夫々ユーティリティ・プロセスとしてサポートしている。従ってHOSTは夫々のプロトコル毎にUSERプロセスとSERVERプロセスとを持つ。図1－6に各プロトコル処理の概念図を示し，図1－7，1－8はそれに対応するジョブ・デックおよび会話の例である。

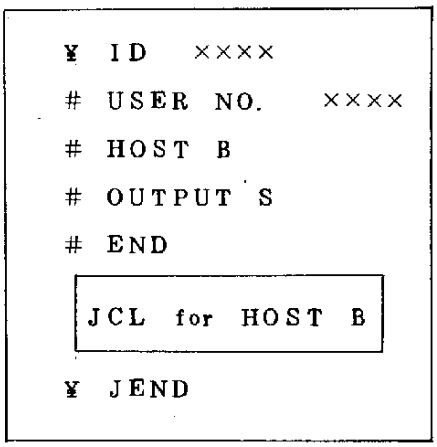



図1－7　RBPのジョブデック

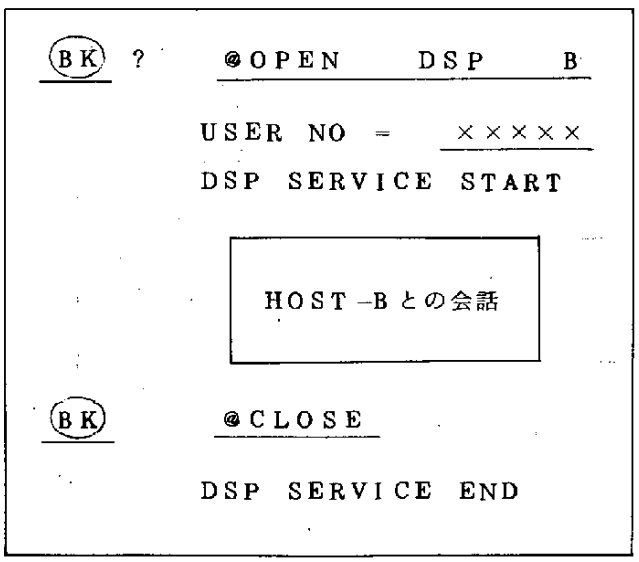
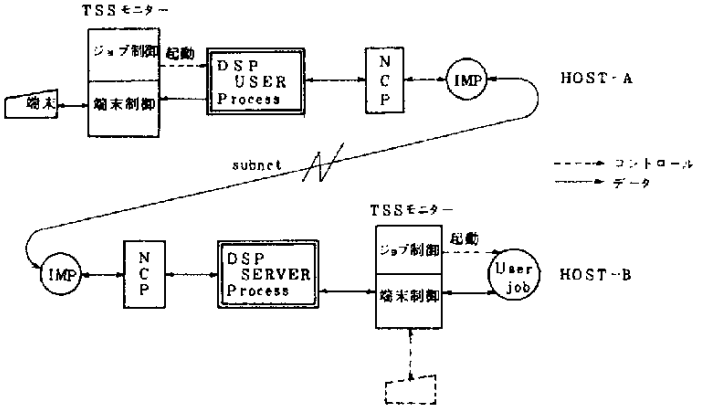
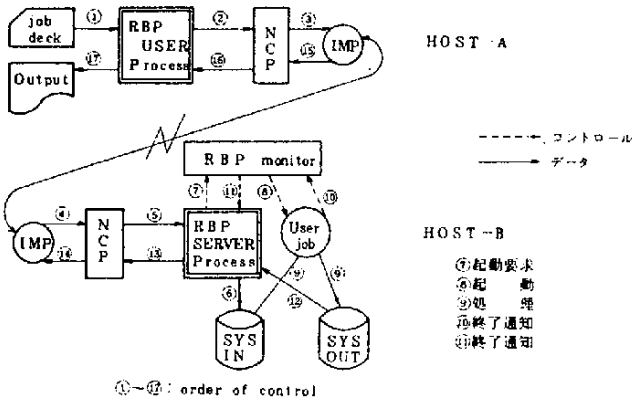



図1－8　DSPの会話の例

ＤＳＰ処理概念図




ＲＢＰ処理概念図




ＦＴＰ処理の概念図

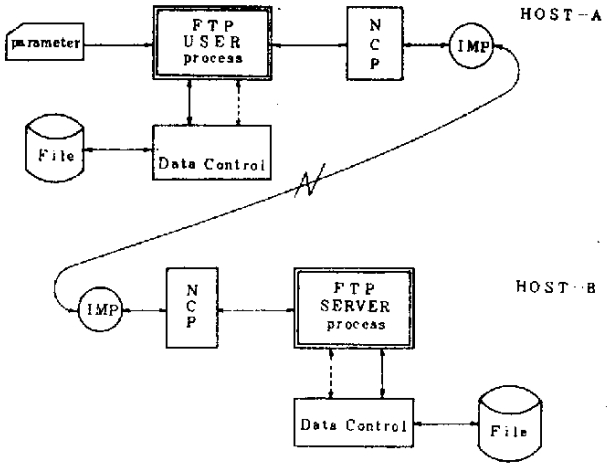



図１－６　高位プロトコル処理の概念図

## 第１章の参考文献

(1) 「コンピュータ・ネットワーク・システムの研究開発」報告書 48－5001 (財) 日本情報処理開発センター

(2) 「コンピュータ・ネットワーク JIPNETの研究開発」報告書 49－5001 (財) 日本情報処理開発センター

# 2. プロトコルの改良

# 2. プロトコルの改良

コンピュータ・ネットワーク JIPNETの開発に着手してからすでに3年が経過した。この間に，サブネット，FACOM230/75-NCP，HITAC8450-NCP，ACOS77 NEAC700-NCPおよびDSP，RBP，FTPの高位プロトコルの第一版が作成された。コンピュータ・ネットワークにおけるプロトコルの重要性は，今さら言うまでもないが，開発を開始してからの2年間の経験を通して最初に作成したHOST-HOSTプロトコルに問題があることが明らかになり，これを現時点において解決しておかなければならないことが判明した。

これをきっかけとして，問題点の解決と同時に，より使い易いHOST-HOSTプロトコルにすべきであるという結論に達し，JIPNETのHOST-HOST プロトコルを後で述べるように改良した。

49年度作成したJIPNETの高位プロトコルは，他系のTSS利用の為のDemand Service Protocol（DSP），他系にリモート・バッチ処理を依頼する為のRemote Batch Protocol（RBP），他系との間でファイルを転送しあうFile Transfer Protocol（FTP）の3種である。これらの高位プロトコルを処理する為のプログラムは昨年度の後半迄に開発され，F/75，H-8450の2台のHOST間でのRBP，FTPおよびTIPからのDSPの使用が可能になり，効率測定等を含め各種の使用経験を積んだ。これらの開発，使用の経験をもとに第1版高位プロトコルを見直した結果，第1版高位プロトコルにはいくつかの問題点，規定のあいまいな点，機能不足の点等があった。新HOST-HOSTプロトコルを採用するに伴い，これらの問題点を解決すべく検討を加え，新規に第2版高位プロトコルを規定することとした。

## 2.1 JIPNETのHOST-HOSTプロトコル

### 2.1.1 JIPNETのHOST-HOSTプロトコルの問題点

JIPNETのHOST-HOST プロトコルは，次にあげる問題点があるのが判明した。

① 基本レベルのプロトコルとしては複雑である。

② フロー制御における問題点

③ コネクションの一方向性

④ バーチャル・コール方式の限界

以下で，それぞれの項目に対する検討をおこなう。

A．プロトコルの複雑さ

ＪＩＰＮＥＴのＨＯＳＴ－ＨＯＳＴプロトコルにおいては，プロセスの発生，コネクションの切り換えの機能を持たせていた。この結果として，ＮＣＰの管理すべき状態が非常に多くなり，コンピュータ・ネットワークに参加するために基本的に必要なソフトウェアの開発に多大な労力が必要となることになった。

ＨＯＳＴをコンピュータ・ネットワークに結合させる労力の削減を考えてみれば，ＨＯＳＴ－ＨＯＳＴプロトコルは，本来できるだけ基本的な機能にすべきである。

この意味において旧プロトコルは，ＨＯＳＴ－ＨＯＳＴプロトコルと高位のプロトコルの間での機能分担に失敗しているということができる。

新プロトコルにおいては，コネクションの切り換えはすべて高位のプロトコルにおいて処理すべき問題であるという観点にたった。この理由は，ＮＣＰがコネクションの切り換えをおこなう場合には，コネクションの両端のプロセスの状態および意図を明確に知り得ないために，すべての状態において切り換え得る処理を用意しておかなければならないが，両端のプロセスのレベルのプロトコルにおいておこなう場合には，前もって両プロセス間において状態の通知が可能であることから，はるかに簡単におこなえることになるからである。

新しいＨＯＳＴ－ＨＯＳＴプロトコルにおいては，このプロトコルも多層の構造を持たせることになった。この理由は，ＨＯＳＴ－ＨＯＳＴプロトコルの機能として，すべてのＨＯＳＴにおいて用意すべき機能，さらに高度な使用形態をめざすサイトにおいてサービスすべき機の２つに分離し，比較的少ない労力でコンピュータ・ネットワークへの参加を可能にすることである。

B．ポートの意味

旧プロトコルにおいては，ポートは入出力のいずれかをおこなう論理的な出入口であるという定義をした。

新プロトコルにおいては，ポートは，コネクションを確立，切断のために利用される入出力兼用の代表名であるという定義にかえた。

このようにすることによって，コネクションは，２つのポート名が一致したものが同一のコネクションとして認識され，２つのうち，少くとも一方が違っていれば他のコネクションを確立できることになる。

C．フロー制御の問題

ＪＩＰＮＥＴのＨＯＳＴ－ＨＯＳＴプロトコルのフロー制御は，ＷＡＢＴ方式をとっていたが，この方式においては各種の問題点があるのが判明した。

基本的問題は，あるＨＯＳＴが他のＨＯＳＴに対してメッセージを送ったときに受信側の状況によってそのメッセージが捨てられることから発生するものである。

例をあげれば，図２－１のように同一のコネクションに対してメッセージを連続的に送った場

合，受信側ＮＣＰに受け取られる順序は，メッセージ１，２，５になり，受信側ＮＣＰはサブネットにおいてシーケンシングされたメッセージに対して再びシーケンシングしなければならなくなる。

ＮＣＰの会話のために利用するコントロール・コマンドもこのアルゴリズムが適用されるために，コントロール・コマンドの送信一時停止，送信再開などのフロー制御に関するコマンドまでも捨てられ再送しなければならなくなる。

以上のような問題点をすべて解決するフロー制御方式はみあたらない。しかしながら，解決しなければならないことには変わりはないので，次のような方式をとることにした。

ユーザ・メッセージのフロー制御方式としては，バッファの確保方式を採用して，メッセージの大量流入およびＷＡＢＴ方式をとった場合に発生するメッセージの再送が起こらないようにする。

コントロール・コマンドに関しては，ＮＣＰのプログラム構造によりメッセージ（コマンド）の再送が起こらないようにする。この結果として，旧プロトコルにおいて，プログラム構造によって再送を防ぐことが不可能なコマンドであるＰＯＳＴコマンドの性格を変えることにした。

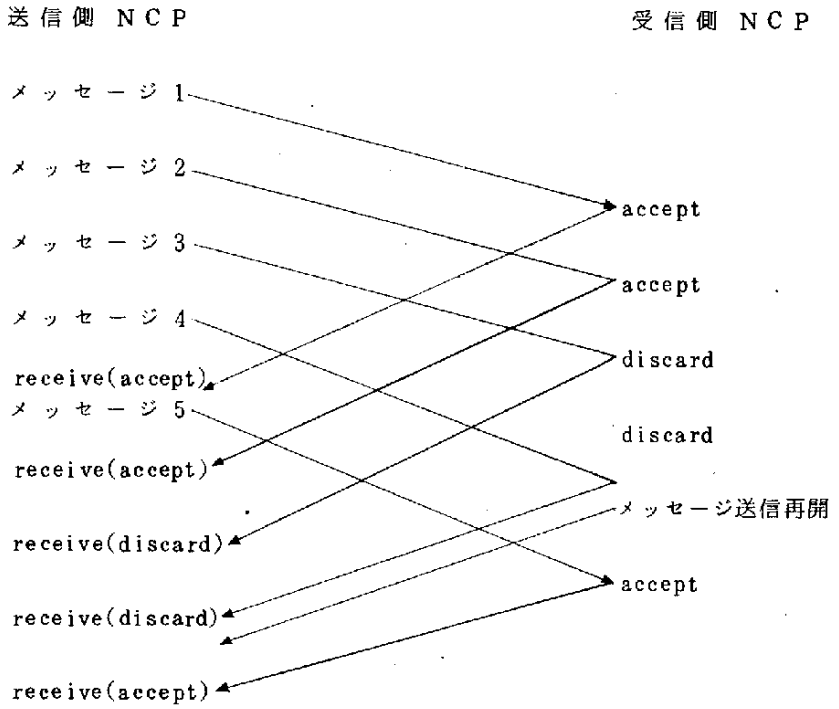



図２－１　ＷＡＢＴ方式の問題点

D．コマンドのすれ違い

コネクションの確立要求が，両プロセスにおいて同時に発生した場合には，両側のＮＣＰからＲＦＣが発信される。

旧プロトコルにおいては，コネクションの確立は，ＲＦＣとＣＯＫの２つのコマンドによっておこなわれたゝめに，図２－２に示すように受信 port－id を持つプロセスが存在するＮＣＰがＣＯＫを発信することによってコネクションの確立を通知した。

ＨＯＳＴ－ＨＯＳＴプロトコルでこのようなタイミングによって処理を変えるのはめんどうなために，新プロトコルにおいては，図２－３に示すように，ＲＦＣ，ＲＦＣの対によってコネクションの確立をおこなう。

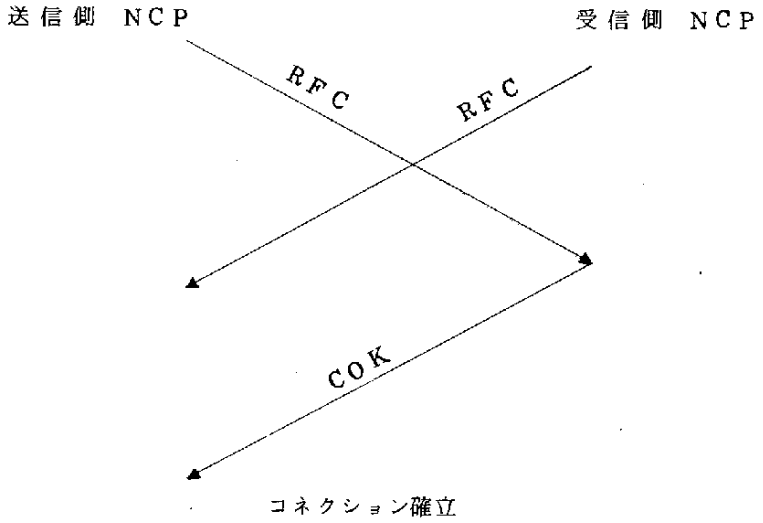



図２－２　旧プロトコルにおけるＲＦＣコマンドのすれ違い

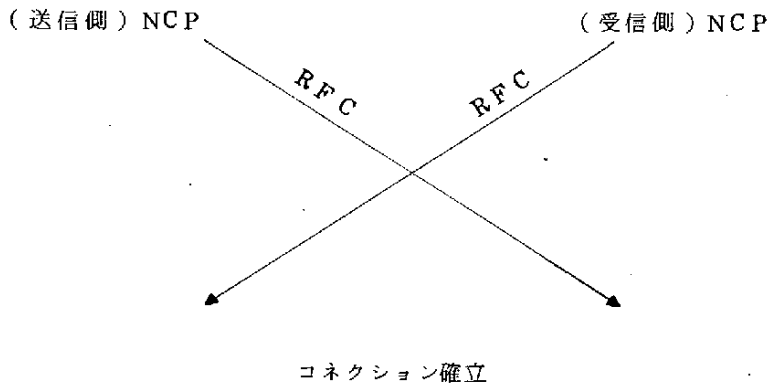



図２－３　新プロトコルにおけるコネクションの確立

E．コネクション単方向性

プロセス間通信をおこない共同して処理を進めていく場合に，単方向のコネクションだけが使用されるのは稀である。

コネクションを確立することは，ＮＣＰにとってもユーザ・プロセスにとってもめんどうなことであるので，できるかぎりこの作業を少なくする方向に持っていくべきである。

このために，新プロトコルにおいては，コネクションは両方のメッセージの流れが可能なようにした。

F．バーチャル・コール方式の限界

ＪＩＰＮＥＴにおいて採用しているプロセス間通信は，コネクションを確立したのちにメッセージを送受信するバーチャル・コール方式である。

この方式では，ｎ個のプロセスが相互に通信をする場合には，ｎ（ｎ－１）本のコネクションを確立する必要がある。このことは，多大なオーバ・ヘッドを要するが，データの収集をおこなうプロセスを想定すれば，使用される可能性は十分にある。

データ・グラム方式をとるとすれば，このような問題は一度に解決するが，ユーザ・メッセージのフロー制御をどうするかという新らたな問題が発生してくる。このような方式に対応できるフロー制御としては，ＷＡＢＴ，Window 方式があるが，いずれにしても手順が複雑になるのは明らかである。

フロー制御を全くとりはずすとすれば，前提として受信プロセスが前もって読み込み命令を発信していればよいので，ＮＣＰがファイルに書き込んでユーザ・プロセスにデータを渡す方式，ユーザ・プロセスが読み込み命令を発信していなければデータを捨てる方式をとることになる。

前者は，ＮＣＰを複雑にし，後者は，ユーザ・プロセスのエラー制御が非常に複雑になり使用にたえない。

このように，現時点においては基本機能としてデータ・グラム方式の採用は効果は十分に認められるが，手順が複雑になることを考えて見合わせることにした。

### 2.1.2　ＪＩＰＮＥＴの新しいＨＯＳＴ—ＨＯＳＴプロトコルの概要

新しいＨＯＳＴ－ＨＯＳＴプロトコルは，すべてのＨＯＳＴにおいてサービスしなければならない基本機能と，より高度な機能をサービスしあうグループ間での付加機能の２つの階層より構成されている。

この本報書では，基本機能についてだけ述べる。付加機能に関するプロトコルは改良することは決定しているが，詳細についての検討が終っていないので，新しい規約ができしだい別の資料で明らかにすることにする。

基本機能は次に述べる４つの項目より構成される。

① コネクションの確立および切断

② ユーザ・コネクションを通るメッセージ・フローの制御

③ 状態情報の通知

④ ＮＣＰ間の状態情報の通知

A．コネクションの確立および切断

コネクションの確立は，ＲＦＣのコマンドによっておこなわれ，ここで確立したコネクションに対してメッセージの送受信を同時におこなうことができる。

コネクションの確立は，同一のポート名を持つＲＦＣの交換によっておこなわれる。ユーザがコネクションを確立する方法は３種類ある。

第一番目は，両プロセスが互いに相手のポート名を知っているときに使用するものであって，ＮＣＰのサービス・コマンドであるOpen を両者が使用することによっておこなう。この例を図２－４に示す。

第二番目は，特定ユーザがユーティリティ・プロセスを作ったときに，これが使用する方式であって，Listen によって自分側のポート名を宣言し，これを相手側ポート名としてＲＦＣがきたとき，そのＲＦＣにはいっているポート名の通知を受けて，Open により確立する方法である。

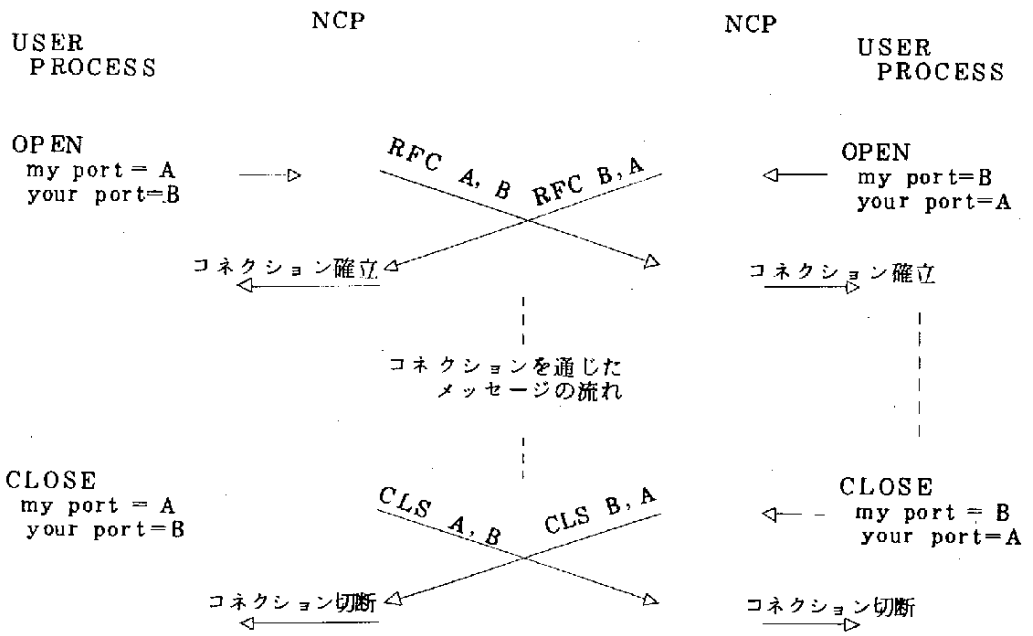



図 2－4　コネクションの確立，切断(1)

第三番目は，高位のプロトコルにおいて利用するもので，第二番目のListenの代りに同一のポート識別番号を持つＲＦＣの通知をするListen anyを利用するものである。

コネクションの切断は，ＣＬＳコマンドの交換によっておこなわれる。

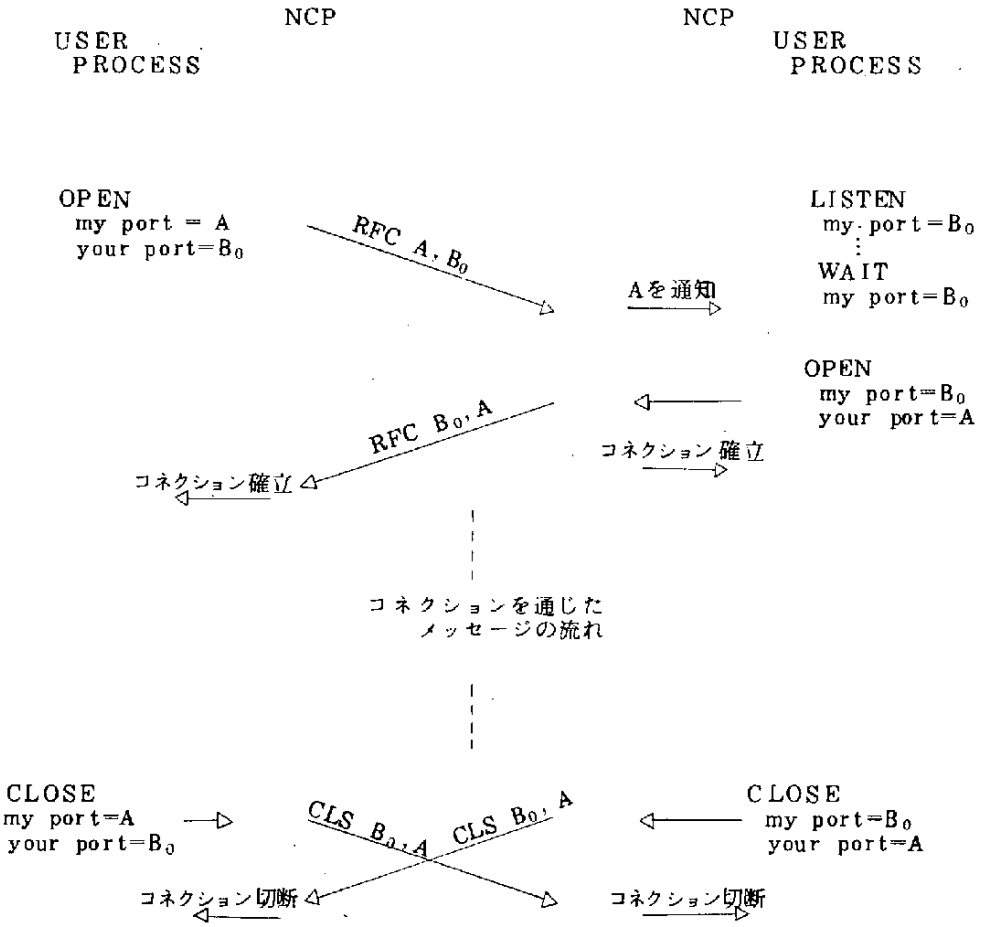



図２－５　コネクションの確立，切断(2)

B．メッセージ・フローの制御

旧ＨＯＳＴ－ＨＯＳＴプロトコルにおいては，ＷＡＢＴ方式をとっていたが種々の問題点があるためにバッファ確保方式を新らたに採用することにした。

コネクションが確立した時点で，受信側ＮＣＰにおいてメッセージを受信するためのバッファを確立して，メッセージ・バッファ個数をＡＬＬＯＣコマンドによって通知する。……メッセージ・バッファ１個は，１個のメッセージを受信するのに十分なバッファである……。

送信側ＮＣＰは，受信側ＮＣＰにおいて準備されているメッセージ・バッファ個数だけのメッセージを送ることができる。当該コネクションに対して準備してあるメッセージ・バッファの個

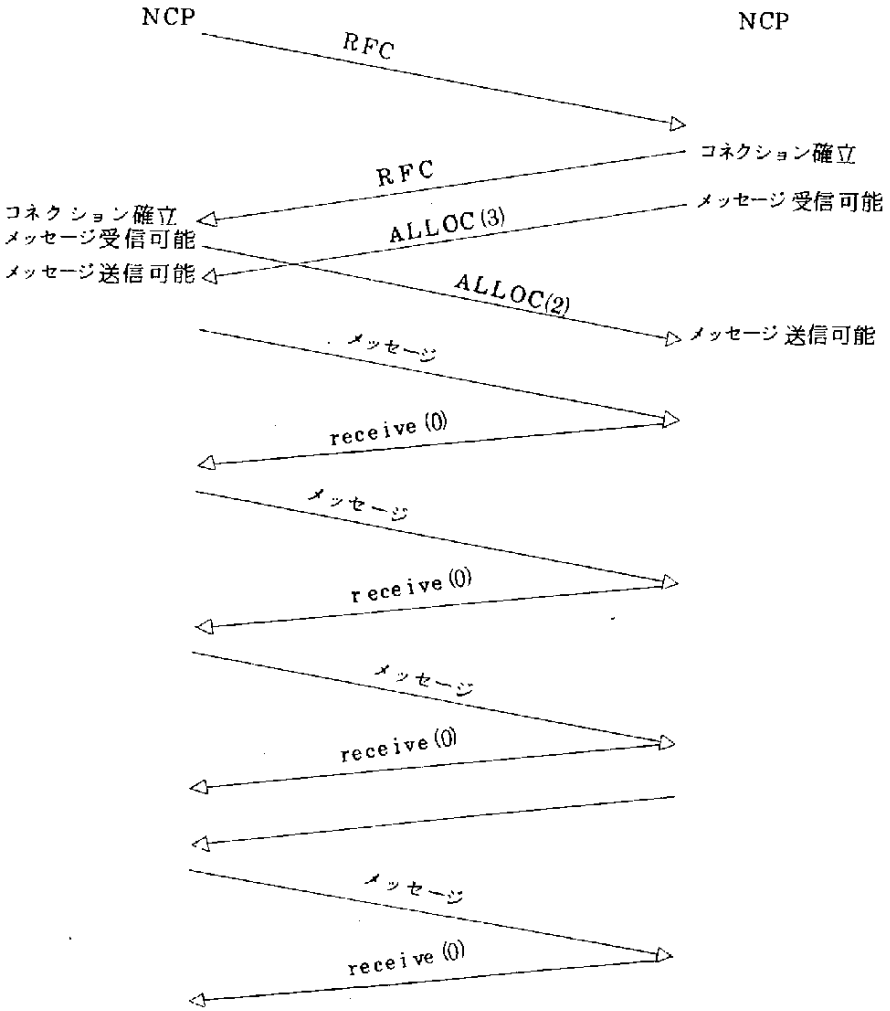
数が0になった場合には，送信側ＮＣＰは新らたに確保されているという通知がくるまでそのコネクションに対してメッセージを送ってはならない。また受信側ＮＣＰはできるだけすみやかにバッファを確保して送信側ＮＣＰに通知しなければならない。

メッセージ・バッファの確保の通知は，コネクション確立時にはＡＬＬＯＣコマンドによっておこなわれるが，こののちは，receive にバッファ確保通知を便乗させることができる。

このために，バッファ確保通知は，ＡＬＬＯＣコマンドによってのみおこなう方法と，ＡＬＬＯＣと receive を混合して使用する方法が許されるが，後者を採用するのが望ましい。

いずれの場合にも，バッファの確保量は，それまでに確保してあったものに新らたに確保通知を受けた量を加算したものが現在のメッセージ・バッファの確保量である。

以上に述べた方式を図で示すと，図２－６，７の如くなる。




ALLOC，RECEIVEのあとのカッコの中は，メッセージ・バッファの確保通知を示す。

図２－６　フロー制御(1)

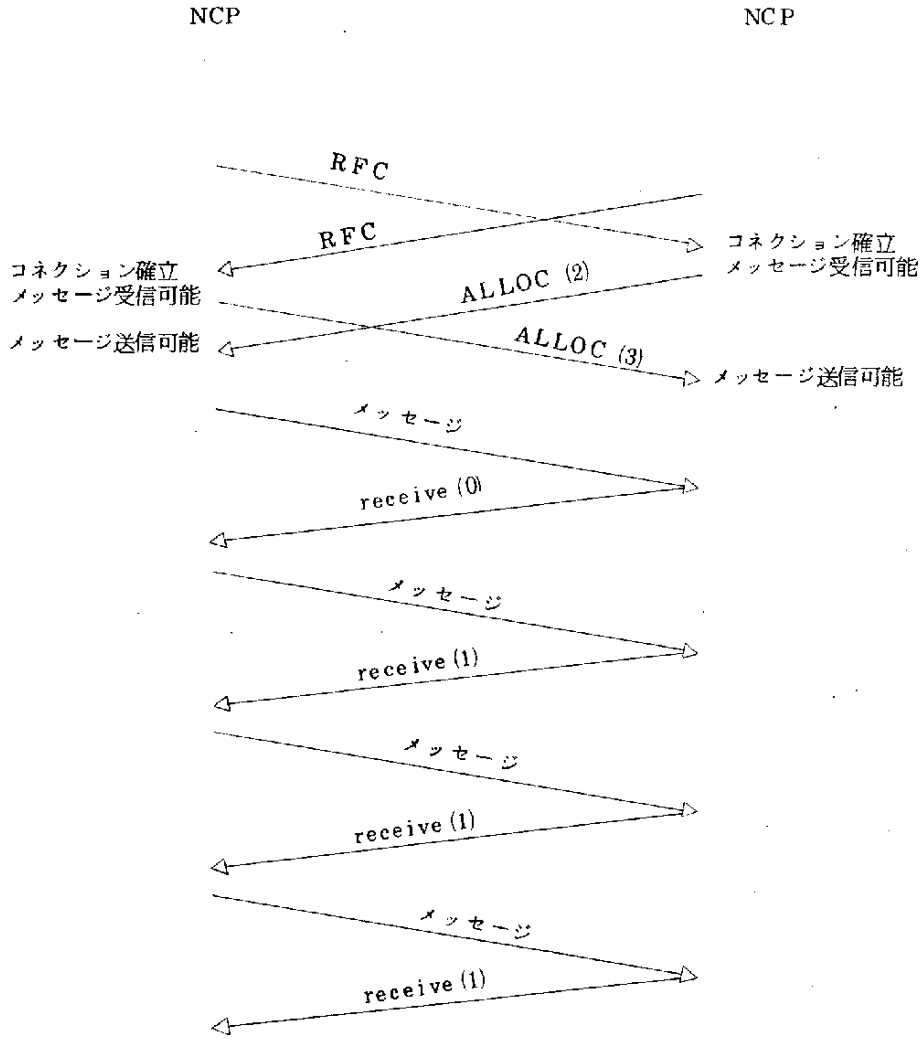



図２－７　フロー制御(2)

C．状態情報の通知

状態情報の通知は，割り込み信号などのようにメッセージとは別の情報を通知のために必要な機能であり，ＰＯＳＴコマンドによって送られる。

ＰＯＳＴコマンドは確立しているコネクションが存在するときにのみ送ることができる。

ＰＯＳＴコマンドがＮＣＰによってどのようにユーザにサービスするかは，プロトコルでは規定しない。

ＰＯＳＴコマンドが連続して２個以上発信された場合には，以前のＰＯＳＴがユーザ・プロセスに受け取られていないと，古いＰＯＳＴが捨てられ，新しいものが有効になる。

D．ＮＣＰ間の状態情報の通知

この機能は，主としてコントロール・コマンドのパラメータ・エラーの通知，サービスしてい

ない付加機能のサービス要求に対する拒否，ＮＣＰを通じてユーザに通知すべき状態情報の通知に利用される。

この機能をはたすのがＳＰＯＳＴである。

### 2.1.3 ＨＯＳＴ—ＨＯＳＴプロトコル

ＨＯＳＴ－ＨＯＳＴプロトコルは，ＮＣＰ間で授受されるメッセージの形式および意味を規定するものである。

A．メッセージの形式

ＨＯＳＴ間で授受されるメッセージの形式はＨＯＳＴ－ＩＭＰプロトコルの規定にしたがっておこなわれる。

ユーザ・メッセージおよびＮＣＰ間における会話のためのコントロール・コマンドの区別は，コネクション番号によっておこなう。すなわち，コネクション番号が０のものがコントロール・コマンドである。

コントロール・コマンドとユーザ・メッセージの処理の区別は，ＨＯＳＴ－ＩＭＰプロトコルにおいてはなく，全く同じ処理をしなければならないが，ＨＯＳＴ－ＨＯＳＴプロトコルのレベルにおいては，コントロール・コマンドにはフロー制御をしない点が異なっている。この意味でコントロール・コマンドがＮＣＰは集中しても，ＮＣＰがロック・アップしないようにインプリメントしなければならない。

B．コントロール・コマンド

コントロール・コマンド（以下コマンドと略称）は144バイト以下のコネクション番号が０のメッセージであり，Ｓ－pipe を通じてＨＯＳＴに送信される。

コマンドの一般形は図２－８のとうりである。

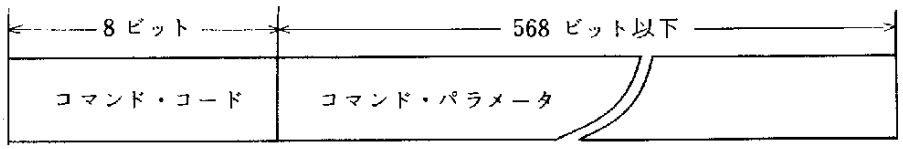



図２－８　コマンドの一般形

コマンド・コードは，コマンド種類を示し，基本機能のコマンドとしては６種類ある。

コマンド・パラメータは各コマンドに付随するパラメータである。

以下に使用する用語の定義，つづいて各コントロール・コマンドの詳細を述べる。

ポート名　　port－id

プロセス間コミュニケーションの出入口となる名前であって，ネットワークにおいてユニー

クに識別できる。

8ビットのＨＯＳＴ番号，24ビットのネットワーク・ユーザ番号，8ビットのポート識別番号より構成される。

ローカル・ポート名　　local port－id

ネットワーク・ユーザ番号，ポート識別番号の32ビットに対する名称であって，コントロール・コマンドのパラメータとして利用される。

自分側ローカル・ポート名　　my local port－id

自分のＨＯＳＴに属するローカル・ポート名をいう。自分のＨＯＳＴ番号を付加することにより，ポート名とすることができる。

相手側ローカル・ポート名　　your local port－id

プロセス間コミュニケーションをする相手のプロセスに属するローカル・ポート名をいう。相手プロセスの存在するＨＯＳＴ番号を付加することにより，ポート名とすることができる。

ポート識別番号

複数のポートを使用するために，ユーザが任意に指定できる番号である。8ビットで構成され，0～255までの値を持つ。0~63までは高位のプロトコルにおいてリザーブされている。

コネクション

2つのプロセス間において，一方のプロセスにおいて出力したメッセージが，他方のプロセスの入力メッセージとなるような関係をいう。

コネクションは，2つのポート名の対によって確立される。2つのポート名が全く同じコネクションを2本確立することはできず，2本以上のコネクションを確立する場合は，少くとも一方のポート名が異なっていなければならない。

コネクションが確立すると，そのコネクションを通じてメッセージの送受信の両方が可能であり，送信および受信は同時にできる。

コネクション番号

コネクション名の一部であって，8ビットより構成される。この番号はユーザが送受信するメッセージのリーダー部に入れられる。通常メッセージを受信したＮＣＰは，この番号により，どのプロセスに渡たすかを知ることができる。

コネクション名

コネクション番号の前に，ＨＯＳＴ番号を追加したものである。一本のコネクションは，メッセージ受信側ＨＯＳＴ番号と，そのＮＣＰがつけたコネクション番号のそれぞれ2つの名前を持っている。

① コネクション確立要求，応答コマンド

・機　　能

2つのプロセス間でプロセス間通信が可能なように，コネクションの確立を要求するコマンドである。

このコマンドは確立応答としても利用される。

コネクションの確立は，同一port－id のペアに対して，ＲＦＣを発信しＲＦＣを受信したときである。

一度確立したコネクションに対して，切断されるまでの間は，ユーザはメッセージを送受信することができる。

確立したコネクションの両端となっている port－id の同じペアに対しては，切断が完了するまでは，別の目的に利用してはならない。

• 一般形

ＲＦＣ　＜my local port－id＞，＜your local port－id＞，＜コネクション番号＞，＜最大メッセージ長＞，＜メッセージ量＞

• コマンド・コード

ＲＦＣのコマンド・コードは，16進で01である。

• パラメータ

＜my local port－id＞　ＲＦＣを発したユーザ・プロセスに存在するport－id である。

＜your local port－id＞　ＲＦＣを受け取るＨＯＳＴに存在するport－id である。

＜コネクション番号＞　コネクションが確立したときに，ユーザ・メッセージの識別をするための番号を示す。この番号は，メッセージの送り側がセグメント・リーダ部に入れなければならない。パラメータ長は８ビットであり，１～240までの値を持つ。

＜最大メッセージ長＞　ＲＦＣの発信側が，このコネクションを通じて送信するメッセージの最大長ℓビットを次の式により計算し，その値を入れる。

$$\left\lceil \frac{\ell + 143}{144} \right\rceil$$

パラメータ長は，８ビットであり０～255までの値を入れる。

＜メッセージ量＞　送信側が，どの程度のスループットを求めているかを示すもので，メッセージの受信は，これを参考にして受信用メッセージ・バッファの確保個数を決めることができる。

パラメータ長は，８ビットであり，０～４までの値を持つ，

この値が大きい程スループットが大きいことを示す。

② コネクション切断コマンド

• 機　能

確立されたコネクションの切断，コネクションの確立要求に対する拒否を通知するコマンドである。

いずれの目的に利用されるにしても，ＣＬＳを発信した場合には，相手側からの対応するＣＬＳを受信することによって手続きが終了する。

コネクションの確立要求に対する拒否は，コネクションテーブルが満杯，対応するコネクション確立要求がユーザ・プロセスから発信されずタイム・アウトになった場合におこなわれる。

• 一般形

ＣＬＳ　＜my local port－id＞，＜your local port－id＞

• コマンド・コード

ＣＬＳのコマンド・コードは16進で02である。

• パラメータ

＜my local port－id＞　切断したいコネクションの自分側のport－idを示す。

＜your local port－id＞　切断したいコネクションの相手側のport－idを示す。

③ メッセージ・バッファ確保通知

• 機　能

メッセージの送信許可を通知する。これはメッセージ・バッファ個数の形で通知され，この個数だけのメッセージを当該コネクションを通じて相手側に送ることができる。

送受信の両側のＮＣＰにおいてコネクションごとにこの個数を記憶しておき，送信側はメッセージ１個送信したとき……再送の場合はのぞく……１減少させ，0になるまで送る。受信側はメッセージを１個受信したときに１減少させ，０になったならば新らたにバッファを確保して送信側に通知しなければならない。

コネクションが確立した時点では，たがいにできるだけはやくバッファを確保して相手側に通知しなければならない。

最大メッセージ長が0の場合にもコネクションが確立した時点でＡＬＬＯＣを発信しなければならない。

• 一般形

ＡＬＬＯＣ　＜コネクション番号＞，＜メッセージ・バッファ個数＞

• コマンド・コード

ＡＬＬＯＣのコマンド・コードは16進で04である。

• パラメータ

＜コネクション番号＞　確保されたバッファが，このコネクションに対して送られてくるメッセージのために使用されることを示す。すなわち，コネクションの確立時に自分が発生したＲＦＣコマンドのパラメータで指定したコネクション番号を入れる。

＜メッセージ・バッファ個数＞　確保したメッセージ・バッファ個数を示し，値は１～255までででなければならない。

バッファ１個は，１メッセージが受信可能な大きさ……したがって最大メッセージ長以上の大きさ……でなければならない。

パラメータ長は，８ビットである。

注意　ＡＬＬＯＣあるいはReceiveによって，256個以上のメッセージ・バッファを保有する状態にしてはならない。

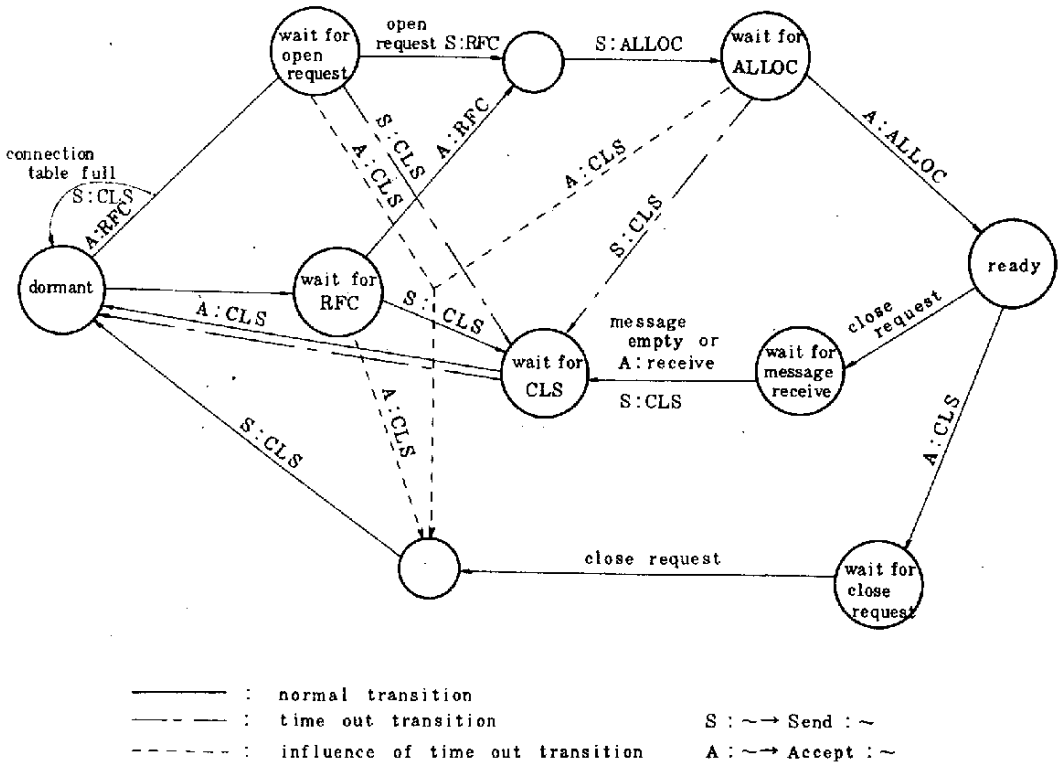



図２－９　コネクションテーブルの状態遷移

④ 状態情報の通知

• 機　　能

コネクションを通る通常メッセージとは別に特別な状態情報を通知するために利用する。相手側の識別のためにコネクション番号をもちいる。

状態情報は16ビットであって，ユーザ・プロセスに渡たらない間に次のものが送られてきた場合には，新しいものが有効になり，古いものは捨てられる。

• 一般形

ＰＯＳＴ　＜コネクション番号＞，＜状態情報＞

• コマンド・コード

ＰＯＳＴのコマンド・コードは16進で10である。

• パラメータ

＜コネクション番号＞　相手側がメッセージを受信するために用意したコネクション番号を示す。この番号により相手側プロセスを認識する。

＜状態情報＞　　相手に通知する16ビットの情報を示す。

⑤ システム情報の通知

• 機　　能

ＮＣＰ間においてコントロール・コマンドのパラメータ・エラーの通知，サービスしていない付加機能のサービス要求に対する拒否，ＮＣＰを通じてユーザに通知すべき状態情報の通知に利用される。

• 一般形

ＳＰＯＳＴ　＜目的＞，＜付加パラメータ＞

• コマンド・コード

ＳＰＯＳＴのコマンド・コードは16進で11である。

• パラメータ

＜目的＞　ＳＰＯＳＴが何を意味しているかを示す。パラメータ長は8ビットであって1～3の値を持つ。

1：コントロール・コマンド・エラー（パラメータ・エラーも含む）

2：付加機能のサービス拒否

3：状態情報の通知

＜付加パラメータ＞　＜目的＞により異なる。＜目的＞の値によって次のような値を持つ。

1：コントロール・コマンド全体，たゞし，568ビットを越えるときは，始めの568ビットのみ。

2：拒否したコントロール・コマンド・コード

3：付加機能が決まり次第追加する。

### 2.1.4 NCPサービス・コマンドに対するコメント

NCPサービス・コマンドは，コンピュータ・ネットワークをプロセスを作ることにより利用しようとするユーザに対して公開されている最も基本的なものである。

本来NCPサービス・コマンドは，各NCPにおいて独自に考えるべきものである。しかしながら，このようにすると，高位のプロトコルの決定，説明に支障をきたすので，例としてとりあげることにした。

この意味において，ここにおける記述および説明は，HOST－HOSTプロトコルと違って何らの拘束力を持たない。

① Listen

ポート名を指定して，そのポート名を対象にしたコネクション確立要求があった場合に，相手側のポート名を通知してもらう。

Listen ＜ポート名＞，＜通知アドレス＞

○ポート名の通知は，Swait で待つ。

○一つのポート名が通知されるとListen の効果は消える。このため再び他の要求の通知を受けるためには再度Listenを発しなければならない。

② Listen any

ポート識別番号を指定して，そのポート識別番号を持つポート名に対してコネクション確立要求があった場合に，相手側のポート名を通知してもらう。

Listen any ＜ポート識別番号＞，＜通知アドレス＞

○ポート名の通知はSwait で待つ。

○一つのポート名が通知されるとListen any の効果は消える。

○Listen によって通知を待っているポート名の一部であるポート識別番号が一致している場合には，Listen のポート名に対する通知が優先する。

○Listen，Listen any のいずれの場合にも，１個のRFCに対してはいずれか１回しか通知されない。

○Listen，Listen any のいずれかが発せられた場合，すでにコネクション確立要求が到着していても，コネクションが確立待ち状態であれば，ポート名の通知の対象となる。

③ Open

自分側および相手側のポート名，メッセージ最大長，要求スループット，完了通知領域などを持つNCCB（Network connection control block）を指定してコネクションの確立を要求する。

Open　＜ＮＣＣＢアドレス＞

○完了の通知は Swait で待つ。

④ Close

ＮＣＣＢを指定して確立済みのコネクションを切断する。

Close　＜ＮＣＣＢアドレス＞

○完了の通知は Swait で待つ。

⑤ Refuse

自分側および相手側のポート名を指定して，Listen あるいは Listen any で通知される原因となったＲＦＣに対するコネクション確立拒否をする。

Refuse　＜my port－id＞，＜your port－id＞

⑥ Read

ＮＣＣＢアドレス，完了通知アドレス，読み込み長セット領域，読み込み先頭アドレスを持つＭＲＣＢ（Message read control block）を指定して，メッセージの入力を要求する。

Read　＜ＭＲＣＢ＞

○完了の通知は Swait で待つ。

⑦ Write

ＮＣＣＢアドレス，完了通知アドレス，書き出し長，書き出しデータ先頭アドレスを持つＭＷＣＢ（Message write control block）を指定して，メッセージの出力を要求する。

Write　＜ＭＷＣＢ＞

○完了の通知は Swait で待つ。

⑧ Swait

通知情報アドレスを１個以上指定して，いずれかの情報が通知されるまで待ち状態にはいる。

Swait　＜通知アドレス＞，＜通知情報アドレス＞，……，＜通知情報アドレス＞

⑨ Post

状態情報の通知をする。

Post　＜ＮＣＣＢアドレス＞，＜状態情報＞

○コネクションは確立済みでなければならない。

○相手側のプロセスに取り込まれる以前に通知すると，古いものは失なわれる。

⑩ Cancel Listen

Listen および Listen any の機能を取り消す。

Cancel Listen　{＜ポート名＞ / ＜ポート識別番号＞}　[，{＜ポート番号＞ / ＜ポート識別番号＞}]

## 2.2 高位プロトコル

昨年度（昭和49年度），ＪＩＰＮＥＴの高位プロトコル第１版として規定したものは，他系のＴＳＳ利用の為のDemand Service Protocol（ＤＳＰ），他系にリモート・バッチ処理を依頼する為のRemote Batch Protocol（ＲＢＰ），他系との間でファイルを転送しあうFile Transfer Protocol（ＦＴＰ）の3種である。これらの高位プロトコルを処理する為のプログラムは，昨年度の後半迄に開発され，F／75，H－8450の２台のＨＯＳＴ間でのＲＢＰ，ＦＴＰおよびＴＩＰからのＤＳＰの使用が可能になり，効率測定等を含め各種の使用経験を積んだ。これらの開発・使用の経験をもとに第１版高位プロトコルを見直した結果，第１版高位プロトコルには，いくつかの問題点，規定のあいまいな点，機能不足の点等があった為，前節で述べた新ＨＯＳＴ－ＨＯＳＴプロトコルを採用するに伴い，これらの問題点を解決すべく検討を加え，新規に，第２版高位プロトコルを規定することとした。

### 2.2.1 ＨＯＳＴ－ＨＯＳＴプロトコル改良に伴う高位プロトコルの変更

前節で述べたように，ＪＩＰＮＥＴのＨＯＳＴ－ＨＯＳＴプロトコルは，コネクションの両方向化，フロー制御方式の改良等により，かなり大巾な変更がおこなわれた。

このＨＯＳＴ－ＨＯＳＴプロトコル改訂に伴い，高位プロトコルにおいても，以下のような関係部分を変更した。

(1) Initial Connection 手順の変更

旧ＨＯＳＴ－ＨＯＳＴプロトコルにおけるコネクションは単一方向であった為，User－Server間に両方向のコネクションを確立するには，UserからServer およびServer からUser への２本のコネクションを張らねばならず，その確立順序も規定しておく必要があった。しかし新ＨＯＳＴ－ＨＯＳＴプロトコルにおいてはport の意味を変更し，コネクションには両方向のメッセージを流す事ができるようになった為，コネクションの確立は１回で済み，各高位プロトコルにおける Initial Connection の手順は，大巾に簡略化されることとなった。

(2) フロー・コントロールの削除

旧ＨＯＳＴ－ＨＯＳＴプロトコルにおけるフロー・コントロール方式では，端末側の処理能力を上まわる速さで，下りメッセージが送られると，ＮＣＰで捨てられ，メッセージの再送が頻発してしまう。これを避ける為に，User ＤＳＰからServer ＤＳＰにバッファ数をＰＯＳＴすることにより，高位プロトコルのレベルでメッセージ・フローをコントロールしていた。しかし，新ＨＯＳＴ－ＨＯＳＴプロトコルでは，バッファ確保方式のフロー・コントロールを採用する事になり，メッセージの再送は起らなくなった為，高位プロトコル・レベルでのフロー・コントロールは不必要となった。従って，ＤＳＰコネクションにおけるフロー・コントロールおよびＦＴ

旧手順における Initial Connection

| HOST 08<br>User DSP | HOST 06<br>Server DSP |
|---|---|
| | LISTEN<br>my port−id = 0000000600 |
| OPEN<br>my port−id = 0600070801<br>your port−id = 0000000600 | WAIT<br>port−id = 0600070801 が<br>POSTされる。 |
| | OPEN<br>my port−id1 = 0600070600<br>my port−id2 = 0000000600<br>your port−id = 0600070801 |
| User DSP→Server DSP の connection確立 | |
| OPEN<br>my port−id = 0600070800<br>your port−id = 0600070601 | OPEN<br>my port−id = 0600070601<br>your port−id = 0600070801 |
| Server DSP→User DSP の connection確立 | |

新手順の Initial Connection

| HOST 08<br>User DSP | HOST 06<br>Server DSP |
|---|---|
| | LISTEN ANY<br>my port−id = 0000000600 |
| OPEN<br>my port−id = 0806000701<br>your port−id = 0606000700 | WAIT<br>port−id = 0806000701 が<br>POSTされる。 |
| | OPEN<br>my port−id = 0606000700<br>your port−id = 0806000701 |
| User DSP↔Server DSP の connection確立 | |

Pのデータ・コネクションにおけるフロー・コントロールは削除することとした。

### 2.2.2 高位プロトコルの機能拡充に伴う変更

高位プロトコルの機能を拡充する為，ＤＳＰにおいてはＮＶＴ（Network Virtual Terminal）の機能の一部変更と，Option 会話の方法を変更し，ＲＢＰは標準コマンドを取り入れ，Option 機能を追加するなどの全面的な改訂を行い，ＦＴＰにおいては，ＦＴＰコマンドを一部追加し，ファイルに各種属性を持たせる事により，ネットワーク内でのソース・プログラム・ライブラリの共用が可能となるような機能追加を行った。

以下に，各高位プロトコルにおける変更点を示し，改訂の狙い等について若干説明する。

(1) ＤＳＰの変更

① ＮＶＴにおける送信権要求

旧ＤＳＰでは，ＮＶＴ側に送信権がない場合，ＮＶＴ側から送信権を要求するには，特殊符号ＲＲＴ（Requcot for Right of Transmission）を通常メッセージとして送信していた。この方法では，Server ＤＳＰ側は，ＮＶＴに対し常に先出し Read を行なわなければならず，Server 側におけるプログラミングの負担を増しており，ＲＲＴとＧＡ（Go Ahead）ＦＳ（Force to send）のすれ違いによる端末操作手順上の問題等を引起していた。

このような問題に対処する為，新ＤＳＰでは，送信権要求用のＲＲＴを削除し，送信権を要求する各機能毎に割込用のＩＰ（Interrupt Process），出力停止要求用のＡＯ（Abort Output），コネクション切断要求用のＤＣＮ（Disconnect）等の信号を，通常メッセージではなく，event送出用のＰＯＳＴ信号として伝える事にした。これにより Server ＤＳＰ側では，先出し Read を出しておく必要はなくなり，ＰＯＳＴ信号とＧＡのすれ違いについてはその処理手順を明示する事によりその悪影響をなくした。

② コネクション切断手順の明示

旧ＤＳＰでは，コネクションの切断手順について明確な規定がなされていなかったので，切断要求用のＤＣＮをＰＯＳＴ信号として規定し，正常時，異常時におけるコネクション切断手順を明示した。

③ Option 会話方法の変更

ＮＶＴにおけるＲＲＴ機能を変更したので，Option 会話要求用のＲＯＰ（Request for Option Negotiation）およびＯＲＦ（Option Refuse）の２つのＰＯＳＴ信号を新設し，Option の会話方法を明示した。

(2) ＲＢＰの変更

① ＲＢＰ標準コマンドの新設

旧ＲＢＰは，ＤＳＰの機能をそのまま利用しており，ＥＢＣＤＩＣデータの扱いやジョブ・

デック送出時の転送効率を上げる為の送信権変更の option 等を追加したものであった。しかし，ＲＢＰの利用形態はＤＳＰとは違い，ジョブ・デックを投入してから，そのジョブが実行され，終了するまでの時間は，かなり長いものになるのが普通である。その間User, Server間での情報交換は何も行わないのにもかかわらずUser RBP，Server RBP間のコネクションを占有し，User ＲＢＰが無為に長時間待っているのはネットワーク全体の利用効率上望ましくない。このような事から新ＲＢＰでは，Remote Batch 特有の動作を数種のＲＢＰ標準オペレーションとして規定し各オペレーションは独自に分割して行えるように変更した。標準オペレーションを行う為のコマンドとしては，入力指令，出力指令，状態問合せ，打切指令，終了指令等があり，終了指令によりコネクションは切断され，User ＲＢＰの動作は終了する形態になっている。したがってＲＢＰのオペレーションはそれ以外の指令と終了指令の組合せでもって指示される事になる。

② Option の追加

旧ＲＢＰは，ＤＳＰコネクションをそのままの形で利用していた為，ジョブ・デックの入力，処理結果の出力メッセージは，ブロッキングせずに送受信していた。新ＲＢＰでは転送効率の向上の為にメッセージのブロッキングを認める option を追加した。

また，出力結果を印字する際の行制御方式についても旧ＲＢＰでは考慮していなかった為，新たにCarriage control 用の option を追加した。

③ エラー時の処置

メッセージの転送中等に入出力エラーが発生した場合の処置や切断手順等が，旧ＲＢＰでは明確でなかった為，新ＲＢＰではこの手順を明示した。

(3) ＦＴＰの変更

① データ・コネクション確立・切断手順の変更

ＨＯＳＴ－ＨＯＳＴプロトコル改訂に伴うPort の意味変更に従い，データ・コネクション確立の手順を変更した。さらに，旧ＦＴＰではコネクション切断の条件および切断手順が明確でなかった為，その条件を明確化し，データ・コネクションおよびコントロールコネクションの切断手順を明示した。

② 文字コード変換のサポート

旧ＦＴＰにおける文字データのコードは，特に明示しなかったが，新ＦＴＰでは，文字コードとしてＥＢＣＤＩＣ（８ビット）およびＪＩＳ（８ビット）コードを標準コードとする事にし，User ＦＴＰからの要求コードがServer ＦＴＰの管理しているコード系と異なる場合は，Server ＦＴＰにおいてコード変換を行うこととした。

③ ファイル属性

旧ＦＴＰにおけるファイルは，属性を持たず，その利用方法は使用者にまかされた形になっ

ていた。しかし，異機種間でソース・プログラム・ファイルを転送し，他機種でそのプログラムを実行しようとすると，機種特有のソース・ライブラリに変換しなければならず，使用者の負担はかなり大きくなってしまう。このような問題の解決の一助として，新ＦＴＰでは，ファイルに対しソース・プログラムとデータという２種の属性を設け，ソース・プログラムについては，Server ＦＴＰにより，そのＨＯＳＴ内の言語プロセッサに入力可能なソース・プログラム・ライブラリ形式に変換して蓄積管理する事とした。これにより，使用者は，ある程度機種にデイペンドしないネットワーク内のソース・プログラム・ライブラリを有する事ができるようになった。

④　プログラムおよびデータの共用

旧ＦＴＰにおいては，ファイルの管理方法が明示されていなかったが，新ＦＴＰでは，個人用ファイルおよび共用ファイルという２つのファイル管理用属性を設け，個人用ライブラリの管理を充実し，またネットワーク利用者間でのプログラムおよびデータの共用が容易に行えるようファイル管理方法を明確化した。

⑤　ＦＴＰコマンドの追加・変更

上記①～④の機能拡張に伴い，ＦＴＰコマンドを追加・変更した。新規のコマンドはＣＨＡＮ（ファイル識別情報の変更），ＤＥＬＥ（ファイルの消去）の２つであり，その他のコマンドのいくつかについてはパラメータ情報等を変更した。

以上，高位プロトコルの改訂に関し，今回の変更点および変更の狙いを記したが，このような高位プロトコルは，ネットワークの利用が進むにつれ，ユーザの要求に応じて次々と追加・拡張されていく性質のものであり，これで完結というものではない。今後も，これらの高位プロトコルの利用経験を通して，より一層の充実を図っていく必要がある。

次項以降に改訂版のＤＳＰ，ＲＢＰ，ＦＴＰを記す。

### 2.2.3　Demand Service Protocol

Demand Service Protocol（ＤＳＰと略称する）は，通常ＴＳＳサービスと称している Demand Service を他のＨＯＳＴから利用するさいにサポートしなければならない高位のプロトコルを規定したものである。

A．概　　要

ＤＳＰは図２－10で示すような構成になっている。

ＤＳＰの実体は実際の端末ユーザが属するＨＯＳＴ内に存在するプロセスUser ＤＳＰと，ユーザジョブが発生するＨＯＳＴに存在するプロセスServer ＤＳＰより構成され，このUser ＤＳＰとServer ＤＳＰとの間でのデータの受け渡し手順，タイミングなどを規定したものである。

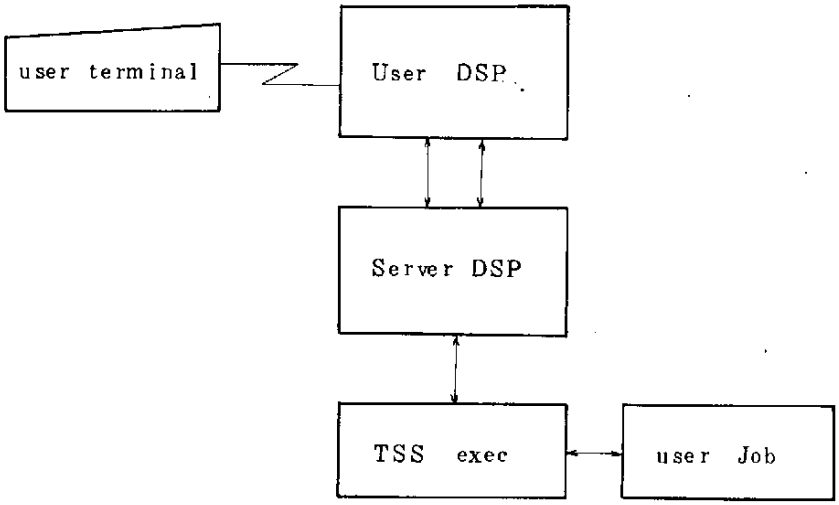



図２－10　ＤＳＰの構成図

ＤＳＰの基本部分は Initial Connection, Network Virtual Terminal （ＮＶＴ）, Option の３つから構成されている。

○ Initial Connection

Uscr DSP と Server DSP間で，コネクションを確立する手順を定めたものである。

○ Network Virtual Terminal

Server DSPは remote terminalに対してメッセージのやりとりをおこなうわけであるが，この remote terminal は各種の端末があり，これのコントロールをすべてServer側でおこなうことは不可能である。このため，Networkに共通な仮想端末を規定して，すべてこれをコントロールする方式をとっている。

○ Option

NVTは，特定の機能の多い端末に対して十分なサポートをおこなうことができないため，User－Server の同意によりNVTの仕様を変更できるようにしてある。これがOption である。

B．Initial Connection

Initial ConnectionはUser DSPと Server DSP との間でコネクションを確立する手順を決めたものである。

あるHOSTのNetwork およびTSSサービスの開始時に，Server DSPは my－port－id＝0をNCPに対し Listen any 状態にしておかなければならない。

User DSPは，Server DSPの識別番号０のPORTに対し，DSP使用者のUser番号，H

OST番号，識別番号よりなる下記のport－idを指定してコネクションの確立要求を出す。

User　　my－port－id ＝＜User HOST番号＞，＜user 番号＞，＜識別番号 n＞

your－port－id ＝＜Server HOST番号＞，＜user 番号＞，＜00＞

この時，識別番号nはUser DSPが任意に決めて良いが，defaultの値としては，Serverがlistenしている識別番号に1を加えたものを使用する。すなわち，ここではn＝1とする。

Server DSPはUser DSPのコネクション確立要求により，port－idがpostされるので，自分側のport－idを下記のように決め，postされたport－idとのコネクションを確立する要求を出す。

Server　my－port－id ＝＜Server HOST番号＞，＜user 番号＞，＜00＞

your－port－id ＝＜User HOST番号＞，＜user 番号＞，＜n＞

この時，user 番号，識別番号nはpostされた値を用いる。

コネクション確立時に指定するメッセージ最大長は144バイト（1152ビット）とする。

コネクションの確立が成功した時点でInitial Connectionの動作は終了する。

User DSPがコネクション確立時に決定するport－idの識別番号nの決定方法は，本来はどのような方法によってもよいはずであるが，各ＤＳＰが勝手な方法で決めると，ＤＳＰにより発生させたユーザ・ジョブがnetwork accessをおこない，他の高位プロトコルを使用する際に識別番号が一致してしまう可能性がある。このようなことを避けるために次の手順によりnを決定することとする。

識別番号00～3$F_{16}$までの64個は高位プロトコルのためにリザーブされており，ユーザが自分の作成するプロセスで何らかの形で高位プロトコルを使用する場合，ユーザ・プロセス間ではそれ以外の識別番号40$_{16}$～FF$_{16}$を利用しなければならない。

DSPではport－idの識別番号として00$_{16}$，01$_{16}$がリザーブされており，defaultの場合はこれを利用してコネクションを確立する。ただし，同一user 番号を持つユーザが同時に2台以上の端末を動かす場合には，同じport－idができることになるので，このような場合User ＤＳＰはユーザの指示により，ユーザの使用できる識別番号40$_{16}$～FF$_{16}$の中から一個をとり出して利用することとする。これにより，同一user 番号を持つユーザは同時に複数個の端末を利用できることになる。次頁の図２－11はInitial Connection確立過程の例である。

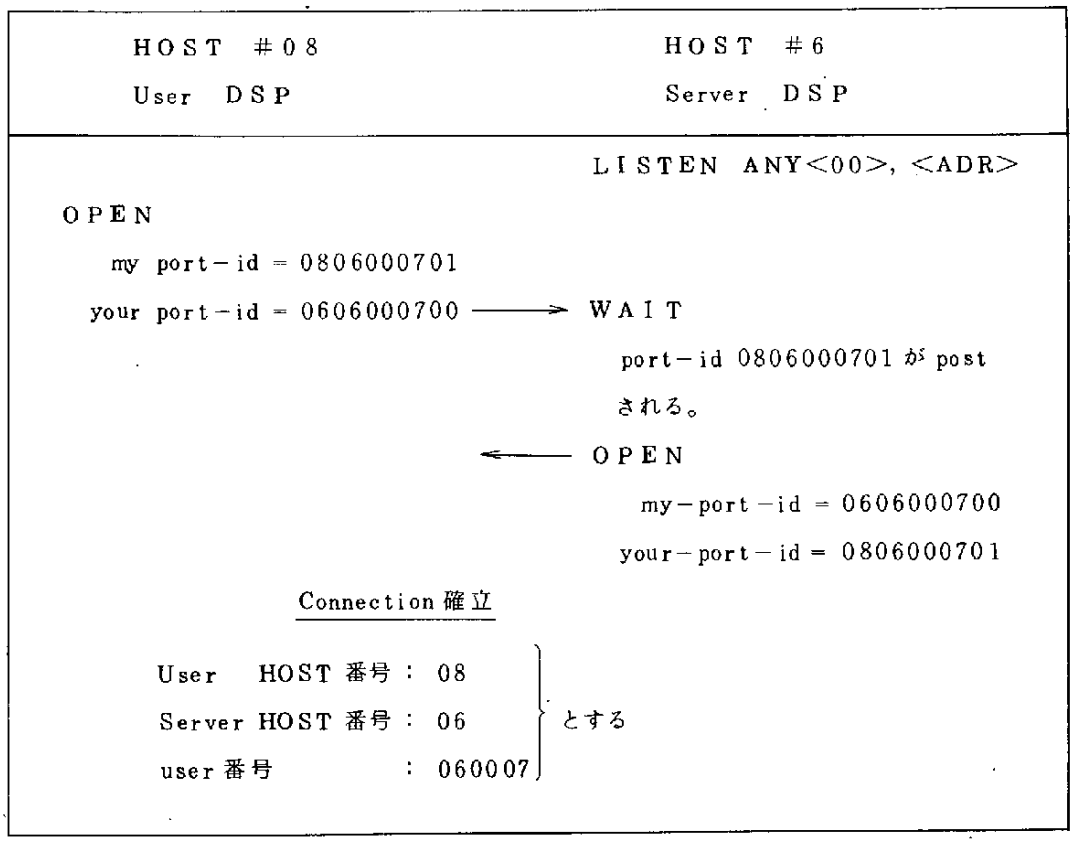
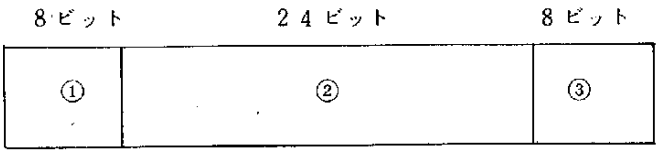



図 2－11　Initial Connection の確立例




① HOST 番号

② user 番号

③ ユーザ指定の識別番号

図 2－12　port-id のフォーマット

C. Network Virtual Terminal

Server DSPはremote terminalに対してメッセージのやりとりをおこなうが，remote terminalは多種多様な機能を持っているために，一元的な制御をおこなうことができない。このような状態では，Server DSPはすべての端末に対する制御方式を用意しなければならなくなり，実際上のインプリメンテーションは非常に困難である。

これをのがれるためには，端末の種類の限定をおこなうかあるいはServer DSPは端末の種類を全く意識しないで標準的なメッセージのやりとりだけをおこない，端末の制御はUser DSP側の責任においておこなうかのいずれかの方法を選択しなければならない。

NVTは，後者の考え方からできたものである。すなわちServer DSPはUser DSPをNVTというnetworkにおいて共通な仮想端末として制御し，User DSPはその制御にしたがって実際の端末を制御する方式である。図2－13はそのモデルである。

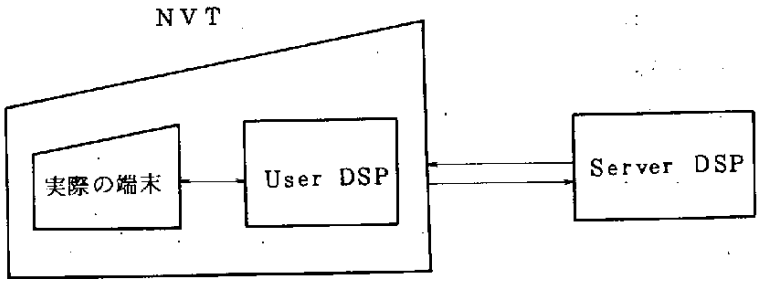



図2－13　NVTの制御モデル

NVTは基本的には全二重回線に接続され半二重動作をおこなう端末タイプライタと考えることができる。しかしながらNVTの機能は固定したものではなく，それを制御するプロセス（たとえばServer DSP）とNVT自身の間で会話することにより両者の合意が得られればその機能を変更することができる。

このため，まずNVTの基本機能およびその制御方法を述べ，つづいて変更可能な機能および変更方法について述べる。

○基本機能

文字単位：8ビット

符号方式：最上位のビットが0のとき，JIS情報交換用符号C6220の7単位符号のローマ文字・カナ文字併用符号とする。

最上位のビットが1のとき，NVTで定めた特殊符号とする。

伝送方式：ブロック伝送（不定長）

動　　作：半二重

○符号方式

付記1に示す如く，codeとしてはＪＩＳ７単位を使用するが，機能キャラクタとしては次のものが意味を持つ。ここで示した以外の機能キャラクタの意味は不定である。

| 機能キャラクタ名 | コード（8進） | 意　味 |
|---|---|---|
| NUL | 000 | 意味を持たない。 |
| BEL | 007 | 警報または合図をする。 |
| BS | 010 | 印字位置を同一行で1文字分後退させる。印字位置が第1文字目の時は意味を持たない。 |
| HT | 011 | 印字行に沿って，あらかじめ定めてある印字位置のすぐつぎの印字位置まで変換させる。 |
| LF | 012 | 次の印字行に移動させる。 |
| VT | 013 | 印字行をあらかじめ定めてある印字行のすぐ次の印字行まで移動させる。 |
| FF | 014 | 印字位置を次のページの定められた最初の印字位置まで移動させる。 |
| CR | 015 | 印字位置を同一行の初めの位置にもどす。 |
| SO | 016 | カナ文字用符号であることを指定する。 |
| SI | 017 | ローマ字用符号であることを指定する。 |

○特殊符号

特殊符号は8ビットの文字コード中の最上位のビットが1のものであってＮＶＴとのやりとりにおいて特殊な働きをなすものである。特殊符号には次のものがある。

| 符号名 | コード（16進） | 意　味 |
|---|---|---|
| SMH | FF | Special mark header<br>以下に記述する特殊符号の前に必ずこの文字をつけなければならない。 |
| GA | FE | Go ahead<br>ＮＶＴを制御するプロセスからＮＶＴに対して送られる符号であって，これを受けたＮＶＴは送信状態となる。 |
| OH | FA | Option header<br>ＮＶＴの機能変更をおこなう会話に使用するヘッダーである。 |
| FS | EF | Force Send<br>強制的に送信権をServer DSP側に持ってくる。User側にあった送信権はなくなり，すでにメッセージが送られている時には捨てられる。 |

○割込信号

割込信号は，User Server 間で特別な情報をやり取りする時に用いられるもので，POSTコマンド（ＮＣＰサービス・コマンド）によってその内容が相手に知らされる。割込信号は次の5種からなっている。

| 名　前 | POST コード | 意　　味 |
|---|---|---|
| I　P | $0_{16}$ | Interrupt Process<br>User DSPからServer DSPに割込をかける。Server DSPはIP受信によりTSSに割込む。 |
| A　O | $1_{16}$ | Abort Output<br>TSSプロセスから転送されるメッセージを捨てる指令。AOを受信したServer DSPは端末に対するRead命令がくるまで，メッセージを捨てなければならない。 |
| DCN | $2_{16}$ | Disconnect<br>コネクション切断の要求を相手側ＤＳＰに知らせる。ＤＣＮを受信したＤＳＰは直ちにcloseを発し，コネクションを切断しなければならない。 |
| ROP | $3_{16}$ | Request for Option negotiation　：　option 会話要求 |
| ORF | $4_{16}$ | Option Refuse　：　option 会話拒否 |

前述の割込信号のとりあつかいについては，項末に参考資料として付記されている。

○ＮＶＴの制御

ＮＶＴとそれを制御するプロセス間における初期設定（すなわちコネクションの確立）はInitial connectionにおいてすでに述べた。

このようにして初期設定が終了した時点でＮＶＴは受信状態，制御プロセスは送信状態になる。

ＮＶＴは通常受信状態になっており，この状態ではプロセスからのメッセージを受信するか，割込用のＰＯＳＴを発信するかのいずれかしかおこなうことができない。ＮＶＴがＰＯＳＴを発信しそれに対するＧＡが返ってきたときにＮＶＴは送信状態になる。

ＮＶＴが受信状態から送信状態に切り換わるのは，上記の場合と，制御プロセスから特殊符号のＧＡを受信した場合だけである。

ＮＶＴが送信状態から受信状態に切り換わるのは，１行分のメッセージを制御プロセスに送信した時である。

制御プロセスは，ＮＶＴに対してメッセージを送信する場合には自分が送信状態になっていることを確認してメッセージを送りだす。もし自分が受信状態ならば，メッセージにＦＳをつけて送り出す。

制御プロセスはＮＶＴに対してメッセージを要求する場合には，ＧＡ単独であるいは送信メッセージの頭にＧＡをつけて送信する。

ＧＡだけの場合には，ＮＶＴは送信状態となり，メッセージの入力が完了し次第送信してくる。ＧＡつきメッセージの場合には，ＮＶＴに対してメッセージを出力したのち，ＧＡだけの場合と同じ処理をする。

○エラーの通知

User Server DSPは端末あるいはＴＳＳとメッセージの送受信を行っている時に回復不能のエラーを検出した場合，ＰＯＳＴでＤＣＮ信号を発してコネクションをclose する。DCNを受けたＤＳＰは直ちにＣＬＯＳＥを発し，コネクションを切断しなければならない。

○Option

ＮＶＴとServer DSPの間において初期設定時には，NVTが基本機能だけで動作することになっているが，両者の了解のもとにその機能を変更することができる。ここでは，変更項目および両者での了解のとり方について述べる。このうち変更項目については現時点でのものであり，将来拡張される可能性はあるが，拡張された部分に対する変更要求がきた場合にはこれを拒否することにより，今までどうりの使用が可能である。

Option 会話用メッセージ・フォーマット

| ＳＭＨ | ＯＨ | オペレーション | 内　　容 |
|---|---|---|---|
| | | | |

オペレーション

$00_{16}$ ： ＷＩＬＬ

$01_{16}$ ： ＷＯＮ'Ｔ

$02_{16}$ ： ＤＯ

$03_{16}$ ： ＤＯＮ'Ｔ

$04_{16}$ ： ＳＢ

$05_{16}$ ： ＯＥＮＤ

内　容

option の種類により異なるが，ＳＢ以外は８ビットでoptionの種類を示す。ＳＢの場合は任意の長さである。

○変更項目

現在なし

○Option 会話の方法

Option の会話はＷＩＬＬ，ＷＯＮ'Ｔ，ＤＯ，ＤＯＮ'Ｔ，ＳＢ，ＯＥＮＤ の６種のオペレーションを使用することによりおこなう。

ＷＩＬＬとはこのコマンドの送信側が送信メッセージを変更したいことを通知する。あるいはＤＯに対する承認を示す。

ＷＯＮ'Ｔとは，送信側になっているコネクションのメッセージを現在の状態からdefaultの状態にもどすことを通知する。あるいはＤＯに対する拒否を示す。

ＤＯとは自分が受信側になっているコネクションに対して受信メッセージを変更したいことを通知する。あるいはＷＩＬＬに対する承認を示す。

ＤＯＮ'Ｔとは，自分が受信側になっているコネクションに対して受信メッセージを default の状態にもどすように通知する。あるいはＷＩＬＬに対する拒否を示す。

ＳＢとはＮＶＴの機能変更の了解がついた時点で，その機能変更の内容を詳細に打合せるために使用する。ＳＢの使用法は，その Option に従属している。

ＯＥＮＤは Option 打合せの会話終了を示すもので，User あるいは Server が自分側からの option 変更要求が無くなった時点で相手側に通知する。option 会話は互いにＯＥＮＤを交換した時点で終了する。

Option 会話の開始の User 側からの要求は前述の割込信号ＲＯＰ（Request for Option negotiation）で Server に知らせる。Server は同じＲＯＰを User に返して会話開始とする。option 会話を拒否する場合はＯＲＦを User に返す。Server から会話要求したい場合は上記の Operation をそのまま送信する。その後上記 operation の交換により，option 会話を行い，ＯＥＮＤにより会話を終了する。

Option 会話の開始要求ＲＯＰや option のやり取りは，送信権ＧＡとは無関係に扱う。従って，option 会話により，会話前の送信権に影響を与えてはならない。

以下の表はＪＩＳ情報交換用符号よりそのまま転載した。

## 付　記　1

日　本　工　業　規　格　　ＪＩＳ

情　報　交　換　用　符　号　　Ｃ6220−1969

Code for Information Interchange

1. 適用範囲　この規格は，情報処理およびデータ伝送を行なうシステムで情報交換に用いる符号について規定する。

2. 用語の意味　この規格で用いるおもな用語の意味は，つぎのとおりとする。

(1) 情報交換　異なるシステム間でも相互に情報が利用できるように，一つのシステムから他のシステムへ情報を伝えること。

(2) キャラクタ　データの構成　制御または表現に用いる要素となるもので，文字，数字，記号および特殊な機能を表現するものからなる。

(3) 機能キャラクタ　特殊な機能を表現するキャラクタで，制御機能の開始，変更，停止などが指定される。

(4) 媒　体　データを物理的状態の変化として表わしうる物質，たとえば紙テープ，カード，磁

気テープなどをいう。

3. 符　号　符号の単位は，7単位および8単位とする。

3.1　7単位符号　7単位符号は，ローマ文字用符号およびカナ文字用符号とし，この両者は単独で使用してもまた併用してもよい。

3.1.1　ローマ文字用符号　ローマ文字用符号は，表1のとおりとする。

3.1.2　カナ文字用符号　カナ文字用符号は，表2のとおりとする。

3.1.3　ローマ文字，カナ文字併用符号　ローマ文字，カナ文字併用符号は，7単位でローマ文字とカナ文字を併用する場合に使うものとし，表1，表2を次の方法で結合する。

表2の機能符号（未定義）部分の0／14および0／15(1)に表1と同義の機能キャラクタSOおよびSIを設け，表1と表2のSOおよびSIの相互利用によって結合する。この場合，SOが先行する符号群は，カナ文字用符号表に示す符号を意味し，SIが先行する符号群は，ローマ字用符号表に示す符号を意味するものとする。

(注) (1)符号表上の位置については6.1参照。

3.2　8単位符号　8単位符号は，表3のとおりである。

表　1　ローマ文字用符号

| ビット番号 $b_7$ | $b_6$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | 列 / 行 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | | | | | | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | | | | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | | | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | | | 0 | 0 | 0 | 0 | 0 | NUL | ($TC_7$)DLE | SP | 0 | @ | P | `(2)(3) | p |
| | | | 0 | 0 | 0 | 1 | 1 | ($TC_1$)SOH | $DC_1$ | ! | 1 | A | Q | a | q |
| | | | 0 | 0 | 1 | 0 | 2 | ($TC_2$)STX | $DC_2$ | "(3) | 2 | B | R | b | r |
| | | | 0 | 0 | 1 | 1 | 3 | ($TC_3$)ETX | $DC_3$ | #(3) | 3 | C | S | c | s |
| | | | 0 | 1 | 0 | 0 | 4 | ($TC_4$)EOT | $DC_4$ | $ | 4 | D | T | d | t |
| | | | 0 | 1 | 0 | 1 | 5 | ($TC_5$)ENQ | ($TC_8$)NAK | % | 5 | E | U | e | u |
| | | | 0 | 1 | 1 | 0 | 6 | ($TC_6$)ACK | ($TC_9$)SYN | & | 6 | F | V | f | v |
| | | | 0 | 1 | 1 | 1 | 7 | BEL | ($TC_{10}$)ETB | '(3) | 7 | G | W | g | w |
| | | | 1 | 0 | 0 | 0 | 8 | $FE_0$(BS) | CAN | ( | 8 | H | X | h | x |
| | | | 1 | 0 | 0 | 1 | 9 | $FE_1$(HT) | EM | ) | 9 | I | Y | i | y |
| | | | 1 | 0 | 1 | 0 | 10 | $FE_2$(LF)(1) | SUB | *(3) | : | J | Z | j | z |
| | | | 1 | 0 | 1 | 1 | 11 | $FE_3$(VT) | ESC | +(3) | ; | K | [ | k | {(3) |
| | | | 1 | 1 | 0 | 0 | 12 | $FE_4$(FF) | $IS_4$(FS) | , | < | L | ¥ | l | \|(3)(4) |
| | | | 1 | 1 | 0 | 1 | 13 | $FE_5$(CR)(1) | $IS_3$(GS) | - | = | M | ] | m | }(3) |
| | | | 1 | 1 | 1 | 0 | 14 | SO | $IS_2$(RS) | . | > | N | ^(3) | n | ¯(3) |
| | | | 1 | 1 | 1 | 1 | 15 | SI | $IS_1$(US) | /(3) | ? | O | _ | o | DEL |

注 (1) 0/10 "LF" および0/13 "CR" の動作を1動作で行なう装置では，0/10を "NL" として使用する。

(2) 2/2 "¨"，2/7 "´"，5/14 "^" および6/0 "`" の記号は，それぞれドイツ語系のウムラウト，フランス語系のアクサンにも使う。その場合には0/8 "BS" を先行させる。

(3) 2/10 "*"，2/13 "="，2/15 "/"，7/12 "|"，7/14 "¯" などは付属書2に示すように多重の意味を有するが，これらは情報送受間の相互了解のもとに記号の意味が混乱しないように固定する。

(4) 一つの符号位置に対して二重にキャラクタを配置した箇所はない。しかし，国際間の情報交換の場合にかぎり，情報送受間の相互了解のもとに2/3 "#" を "£" に置きかえて使用してもよい。

(5) 6/0 "`"，7/11 "{"，7/12 "|" および7/13 "}" の四つの記号は，表1，表2および表3に盛られている以外の記号および文字と置きかえてもよい。

表　2　カナ文字用符号

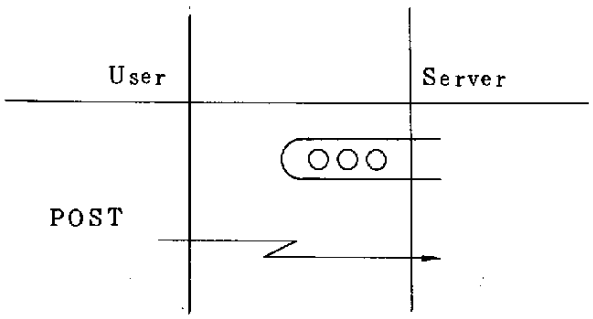
| ビット符号 $b_7$ | $b_6$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | 行＼列 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| | | | | | | | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | | | | | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | | | 0 | 0 | 0 | 0 | 0 | | | SP | — | タ | ミ | | |
| | | | 0 | 0 | 0 | 1 | 1 | | | 。 | ア | チ | ム | | |
| | | | 0 | 0 | 1 | 0 | 2 | | | 「 | イ | ツ | メ | | |
| | | | 0 | 0 | 1 | 1 | 3 | | | 」 | ウ | テ | モ | | |
| | | | 0 | 1 | 0 | 0 | 4 | | | 、 | エ | ト | ヤ | | |
| | | | 0 | 1 | 0 | 1 | 5 | | | ・ | オ | ナ | ユ | | |
| | | | 0 | 1 | 1 | 0 | 6 | | | ヲ | カ | ニ | ヨ | | |
| | | | 0 | 1 | 1 | 1 | 7 | | | ァ | キ | ヌ | ラ | | |
| | | | 1 | 0 | 0 | 0 | 8 | | | ィ | ク | ネ | リ | | |
| | | | 1 | 0 | 0 | 1 | 9 | | | ゥ | ケ | ノ | ル | | |
| | | | 1 | 0 | 1 | 0 | 10 | | | ェ | コ | ハ | レ | | |
| | | | 1 | 0 | 1 | 1 | 11 | | | ォ | サ | ヒ | ロ | | |
| | | | 1 | 1 | 0 | 0 | 12 | | | ャ | シ | フ | ワ | | |
| | | | 1 | 1 | 0 | 1 | 13 | | | ュ | ス | ヘ | ン | | |
| | | | 1 | 1 | 1 | 0 | 14 | | | ョ | セ | ホ | ゛ | | |
| | | | 1 | 1 | 1 | 1 | 15 | | | ッ | ソ | マ | ゜ | | DEL |

機能符号（未定義）　機能符号（未定義）　国字符号部分　国字符号部分

（参考）

割込信号をＰＯＳＴで送受した時のＤＳＰの処理方法

1.　Server が送信権を持っている時




○　ＧＡなし下りメッセージ

●　ＧＡ付　　〃

(1)　ＩＰの時　　User　側：通常の処理と同じ

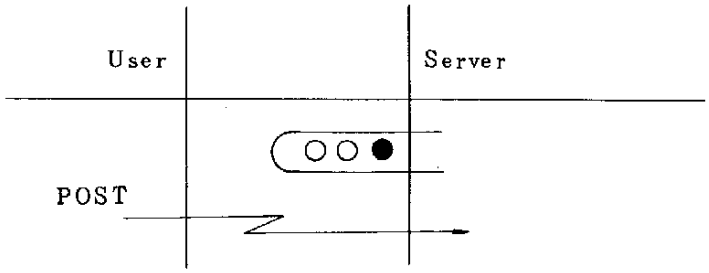
Server 側：ＧＡ付の guidnnce message を送信しＩＰ処理

(2)　ＡＯの時　　User　側：ＧＡが来るまで下りメッセージを捨て，ＧＡが来たら guidunce message を端末に出して端末入力モードにして再開

Server 側：TSS プロセスから read が出されるまで下りメッセージを捨て read が来たらＧＡを送信して再開

(3) DCNの時　User側，Server側とも close する。

2. Server が受信状態の時




(1) ＩＰの時　User　側：通常の処理と同じ

Server側：ＧＡに対する上りメッセージを受けたら，ＧＡ付Guidunce message を送信し，ＩＰ処理

(2) ＡＯの時　User　側：１－(1)と同じ

Server側：そのまま上りメッセージを持つ。

(3) DCNの時　User，Server側とも close する。

### 2.2.4 Remote Batch Protocol

Remote Batch Protocol（ＲＢＰ）は他のHOSTからBatchあるいはRemote Batch機能を利用するためにサービスしなければならない高位プロトコルを規定したものである。

A．概　要

RBPは図２－14で示すような構成になっている。

RBPの実体は，Remote Batchを依頼するユーザが属するHOSTに存在するプロセスUser RBPと，Remote Batch jobが発生するHOSTに存在するServer RBPにより構成される。RBPとはこのUser RBPとServer RBPとの間でのデータの受渡し手順，タイミング等を規定するものである。

RBPの基本部分はInitial connection，DSP connection，Option extention，RBP標準コマンドの４つから構成されている。

○ Initial connection

User RBPとServer RBPの間でコネクションを確立する手順を定めたものである。

○ DSP connection

DSP connectionとは，DSPにおいてUser DSP－Server DSP間に設定されるコネクションのことであり，NVT，割込信号，optionの３つを含む。

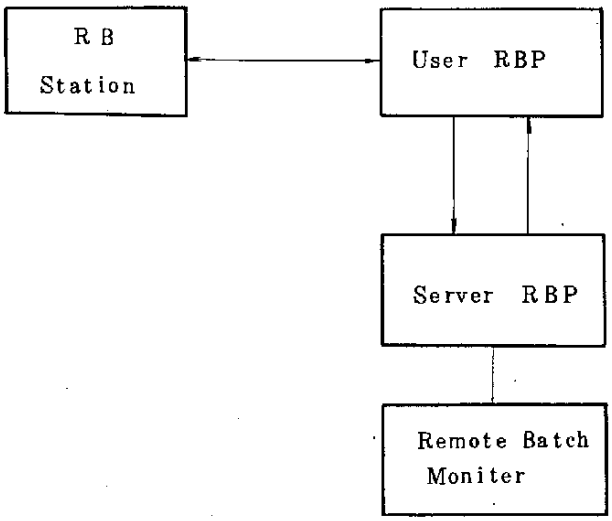



図２－14　RBPの構成図

RBPにおいても基本的にはこのコネクションを利用している。

○Option extention

DSP connectionにおけるoptionは会話型を主体として構成されているが，RBPにおいてはこれを拡張して，より大量のデータを効率良く伝送することを可能にしている。

○RBP標準コマンド

RBP特有の各種動作を規定する標準コマンドである。

B．Initial connection

Initial connectionは，User RBPとServer RBP間でコネクションを確立するための手順を定めたものである。RBPにおけるこの手順は，Server RBPが最初にListen any状態にしておくport idの識別番号が2あるいは8であることを除いて，DSPのInitial connection の手順と全く同じである。

識別番号2と8の選択は下記のようにしておこなう。

User RBP側がTipのようにバッファ容量，プログラム・エリア等の関係でブロック化したメッセージを扱えないHOSTの場合は，識別番号2でInitial connectionを確立する。

User RBP側がブロック化したメッセージを扱う場合は識別番号8でInitial connectionを確立する。

C．DSP connection

DSP connectionは，RBPにおいてコマンド，データ等が転送されるコネクションであり，その仕様はBlocking data optionがない場合は，DSPのそれと全く同じである。

Blocking data option が了解された場合は，特別のメッセージ・フォーマットでデータが送受信され，メッセージ長が1152バイト以内となる。

D. Option extention

option に関する会話は，DSP の option にしたがっておこなわれる。変更項目は次の7種である。

a. EBCDIC data option 〔00〕$_{16}$ (option code)

この option が了解されたとき，一方から他方に送られるデータはEBCDIC code となる。但し，option の変更に関するものはJIS code のままである。

b. Unit address option 〔01〕$_{16}$

この option が了解されたとき，Server から送られるメッセージ（GAだけのものも含む）の頭に必ず unit address byte が含まれている。unit address byte は，1バイトより成り，メッセージ本文（NVTの特殊符号のあとで，通常出力される部分）の最初にこれが入り，以下にデータ・レコードが続く。

このバイトは次のような device を示す。

| device code | |
|---|---|
| $00_{16}$ | 端末タイプライタ |
| $01_{16}$ | ディスプレイ装置 |
| $02_{16}$ | 磁気テープ装置 |
| $03_{16}$ | カード・リーダ |
| $04_{16}$ | 紙テープ・リーダ |
| $05_{16}$ | 未使用 |
| $06_{16}$ | ラインプリンタ |
| $07_{16}$ | カード・パンチ |
| $08_{16}$ | 紙テープ・パンチ |
| $09_{16}$ | XYプロッタ |

この option の了解が得られない場合の Server からのメッセージには unit address が含まれず，データ・レコードのみとなる。

この option が了解されたとき，User 側から Server 側にSBによって前記端末種類が通知される。

このときの format は下記のとおりである。

| SMH | OH | SB | n | device code | ← n → | device code |
|---|---|---|---|---|---|---|

c. Full duplex option

DSP connectionをfull duplexとして利用する。

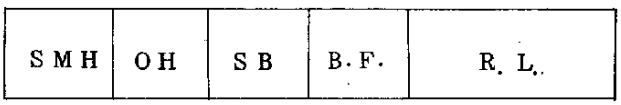
d. exchange right of transmission 〔 03 〕$_{16}$

RBPの送信権を常に端末側にもってくる。これにより，User RBPはGAがなくともServer RBPへ連続してデータを送ることができる。また，この状態でServer RBPからUser RBPにメッセージを送りたい場合はFS付のメッセージとして送信しなければならない。

e. Blocking data option 〔 04 〕$_{16}$

RBPの送受信するデータ（コマンド以外）はブロッキングされることを示す。

このoptionが了解された時，Blocking factorとレコード長が下記のフォーマットでUser RBPからServer RBPに送られる。




B.F.　Blocking Factor　$1 \sim n_{16}$

R.L.　Record length　$nn_{16}$

ブロッキングされたデータを送受信するUser RBP，Server RBPが，実端末あるいはRemote Batch monitorとの入出力を行うさいはBF，RLにしたがいディブロッキングを行なわなければならない。

但し，RBPが送受信するコマンドはブロッキングされず単独で送られる。

Blocking dataのoptionが了解された時のメッセージフォーマットは下記のようになる。

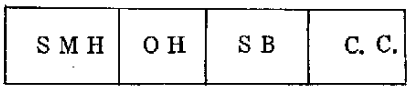
① メッセージ長は1152 Byte以内で，R.L.，B.F.で規定したレコード長（固定），レコード数にてブロック化する。

② トランケートは，転送中，任意のブロックでおこなうことができる。トランケートされたブロックを受取った側では，そのブロックに対し適切な処理を行うこと。

③ RBPコマンドは，ブロッキングせず単独メッセージとして送られる。

f. Carriage control option 〔 05 〕$^{16}$

User RBPが受信するデータ（出力結果）のレコード先頭にcarriage control用制御コードが付加されることを示す。このoptionが了解された時，carriage control方式を示すコードが下記のフォーマットでUser RBPからServer RBPに送られる。




C.C.　carriage control方式を示すコードで下記の値を持つ
$0_{16}$：JIS FORTRAN用のvertical format control方式（当面はこの方式のみ）

この場合，データ・レコードの先頭文字にはvertical format control用の制御文字が付加される。

ブランク：１行送り

0：２行送り

1：頁換え

＋：行換えせず（２重打ち）

carriage control optionの了解がない場合はＮＶＴ機能に準じた方法で，Server RBPが行制御文字を挿入する。この方法として，例えば下記のようなものが考えられる。

（下記はすべてNVTコード）

| | | | | | | |
|---|---|---|---|---|---|---|
| １行送り | ： | (LF) | (CR) | (NUL) | (NUL) …… | 印字本文 …… |
| ２行送り | | (LF) | (LF) | (CR) | (NUL) …… | 〃 …… |
| 頁換え | | (FF) | (CR) | (NUL) | (NUL) …… | 〃 …… |
| ２重打ち | | (CR) | (NUL) | (NUL) | (NUL) …… | 〃 …… |

g．RBP command header 〔06〕$_{16}$

RBPが送受信するコマンドの先頭文字コードを決めるためのoptionであり，User RBPがServer RBPに要求するこのoptionが了解されると，User RBPとServer RBP間で送受するコマンドの先頭文字コードが下記のフォーマットでServer RBPからUser RBPに送られる。

| SMH | O H | S B | C.H. |
|---|---|---|---|

C.H.　command header（NVTコード）

実際に送受されるRBPコマンドは，command headerの２文字とRBPコマンド・コード２文字の計４文字から構成される。

C.H.で知らされた文字が@@だとすると下記のような形式のコマンドが送受される。

@@RI …… Input

@@RO …… Output

@@RS …… Statrcs

@@RA …… Abort

@@RE …… END

@@RB …… Bye

User RBP，Server RBP間で送受するデータのコードがNVTコードの場合は問題ないが，EBCDIC data optionが了解されたものについてはC.H.の文字をEBCDIC文字に

変換して使用しなければならない。

E．RBP標準コマンド

RBPに関する処理を行うために，User RBP，Server RBP 間で下記の6種のコマンドを定める。これらのコマンドはRBP command header option により定められた文字が先頭2文字となり送受される。

a．RBP入力指令

コマンド　RI△　Job－stream－id▲　（▲はコマンドの終りを示しブランクorCRとする。）

User RBPがServer RBPにremote Batch処理を依頼するコマンドであり，User RBPはRI コマンドを送り，続いてジョブ・デックを送出する。ジョブ・デック送信終了は後述のREコマンドでServer RBPに知らされる。

Job－stream－idは，一回のRIコマンドで送出するジョブ・デックに付加するジョブ・ストリーム識別情報であり，10文字以内の英数字で表わす。このJob－stream－idは，その後の状態問合せ，出力依頼等を行う際の識別情報となるので，入力依頼して，出力完了していないジョブのJob－stream－idはユニークに指定すること。

b．RBP出力指令

コマンド　RO△　Job－stream－id〔(Job－id1, Job－id2, …… , Job－id n)〕▲

処理出力結果の返送を依頼するコマンドであり，ROコマンドを受信したServer RBPは，指定されたJob－stream－id のジョブの中で，処理が終了している出力結果をUser RBPに返送する。

出力結果の転送終了はREコマンドによりUser RBPに知らされる。

Job－idが指定されず，Job－stream－id のみが指定された時は，指定されたジョブ・ストリーム内のジョブすべてが出力対象となる。

Job－id が指定された時はジョブ・ストリーム内の指定ジョブのみが出力対象となる。

Job－idは，ジョブ・デック内の remote HOST用JCLに記述したジョブ識別情報である。

c．ジョブ処理状態問合せ指令

コマンド　RS△　Job－stream－id〔(Job－id 1, Job－id 2, ……, Job－id n)〕▲

remote HOST 処理依頼したジョブの処理状態を問合せるコマンドであり，RSコマンドを受信したServer RBPは指定されたジョブ・ストリーム内のジョブの処理状態をUserRBPに返答する。返答する状態情報の種類は，下記の3種とし，処理結果の出力と同様のフォーマットに編集し，User RBPに送出する。転送データの終了はRE コマンドでUser RBP に知らされる。

ジョブ処理状態の種類

① 実行待

② 実行中（経過時間）

③ 出力待

d．処理打切指令

コマンド　RA△ Job－stream－id〔(Job－id 1, Job－id 2, ……, Job－id n)〕▲

remote HOSTに処理依頼したジョブの処理打切を指示するコマンドであり，RAコマンドを受信したServer RBPは，指定されたジョブ・ストリーム内の指定されたジョブの処理依頼を取り止める。実行待のジョブは実行待から除き，実行中のジョブは途中で打切り，出力待のジョブは出力結果を捨てることになる。

RAコマンドの応答は，どのような状態にあるジョブにどのような処置を施こしたかがわかるような情報を，処理結果出力用のデータと同様のフォーマットに編集してUser RBP に返送する。転送終了はREコマンドによってUser RBPに知らされる。

e．転送データ終了指令

コマンド　RE▲

User RBPとServer RBP間で転送されるデータの終了を示すコマンドであり，RIの場合以外はすべてServer RBP側よりUser RBP側に送られる。

f．RBPコマンド終了指令

コマンド　RB▲

一連のコマンドの終了を示すコマンドであり，Server RBPはこのコマンドを受取ると，コネクションを切断する。

g．RBPにおける標準コマンドのやりとり

① Server RBPは，User RBPからの要求によりInitial connection を確立した後，User RBPにGA付のguidunce message を送信し，User RBPからのコマンド入力待状態にしておく。

② RI コマンドの場合。Server RBPは，User RBPから送られてくるREコマンドによりジョブ・デック送出完了を知る。

Server RBPはREコマンドを受信したらGA付のguidunce message（next command 入力要求）を送信し，コマンド入力待の状態にしておく。

③ RO，RS，RAコマンドの場合。Server RBPはこれらのコマンドを受取り，必要な情報をUser RBPに返送した後，REコマンドをUser RBPに送り転送完了を通知する。その後，GA付のguidunce messageをUser RBPに送り，コマンド入力待の状態にしておく。

④ RBコマンドの場合。Server RBPはこのコマンドの受信により，User RBPからのコマンド送出がすべて終了した事を知り，コネクションをcloseする。

User RBPもこのコマンドを送信した後コネクションをcloseする。

⑤ User RBP, Server RBP で回復不能のエラーを検出した場合は相手側にDCNをPOSTしコネクションをcloseする。

⑥ User RBPからServer RBPへInterrupt信号がPOSTされた場合の処理については，各HOSTの機能に依存する為，下記の原則を定めるに留める。

- ○ Server RBPはInterrupt信号を受信することができるようにしておく事。
- ○ Server RBPがInterrupt信号を受信したら，Server側で取れるactionの種類により，適切なguidunce messageをUser側に送信し，User側からの次の指示に備えること。
- ○ Interrupt信号に対するactionはDSPに準ずる。

（参考）RBPにおけるoption会話の方法

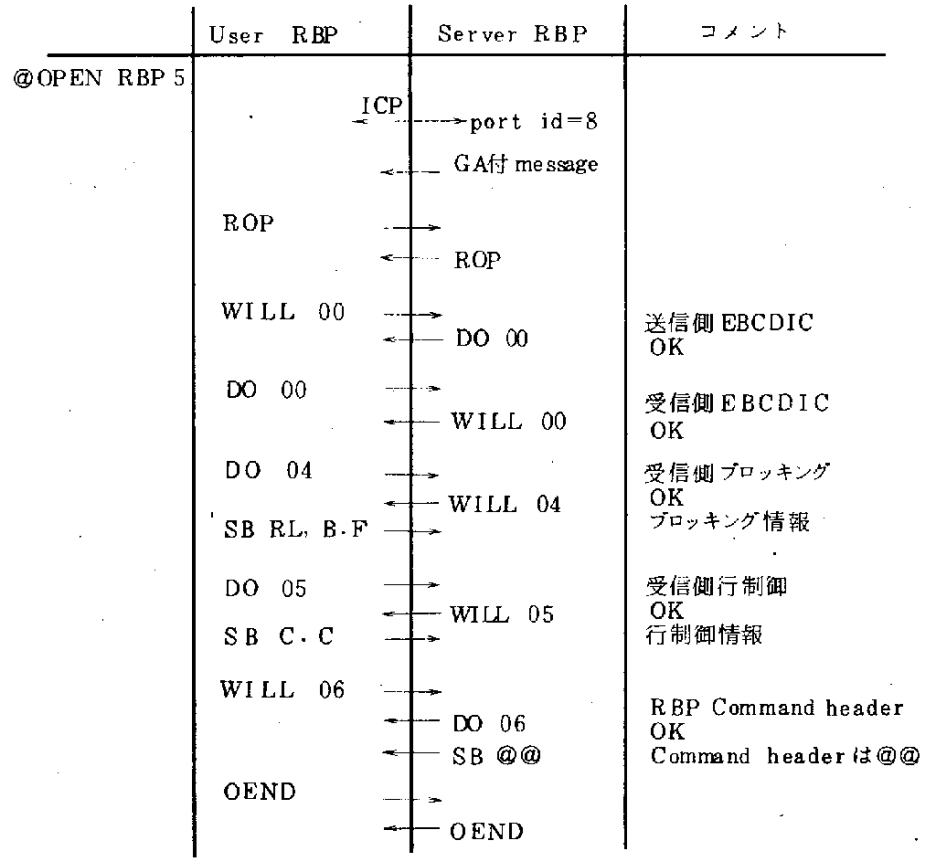
RBPにおけるoptionの会話は，Dで記したoption extentionを，DSPで定めたoption会話方法によっておこなえば良い。

以下に述べるoption会話はTipなどの小型HOSTと大型のHOSTにUser RBPが存在する場合の会話方法の例を示すものである。

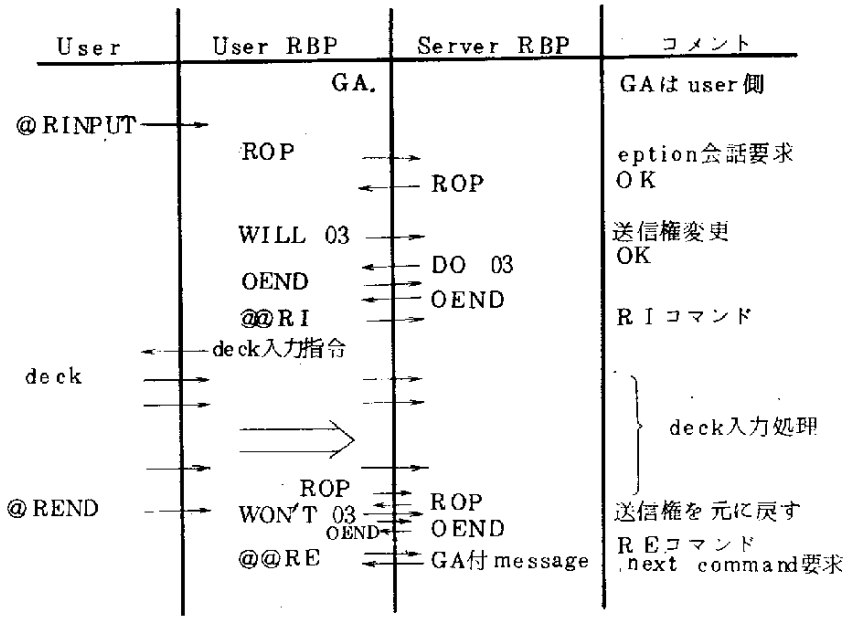
(1) TipにおけるRBP初期会話

| User | User RBP(Tip) | Server RBP | コメント |
|---|---|---|---|
| @OPEN RBP 5 | ICP | Port-id=2 | Initial connection |
| | | GA付message | |
| | ROP | | option会話要求 |
| | | ROP | option会話開始 |
| | WILL 06 | | command header |
| | | DO 06 | OK |
| | | SB @@ | @@をheader |
| | OEND | | option会話終了 |
| | | OEND | 〃 |

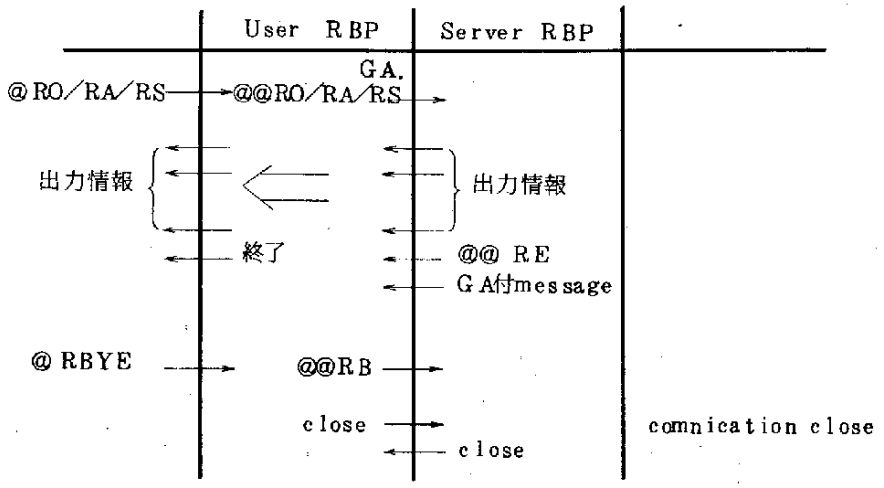
(2) 大型HOST User RBPの場合のRBP初期会話




(3) RIコマンドの為のoption会話



(4) その他のコマンドの動作




### 2.2.5 File Transfer Protocol

File Transfer Protocol(FTP)はネットワーク内のHOST間でFileを転送するための高位プロトコルを規定したものである。

A. 概　要

FTPは図2－15で示すような構成になっている。

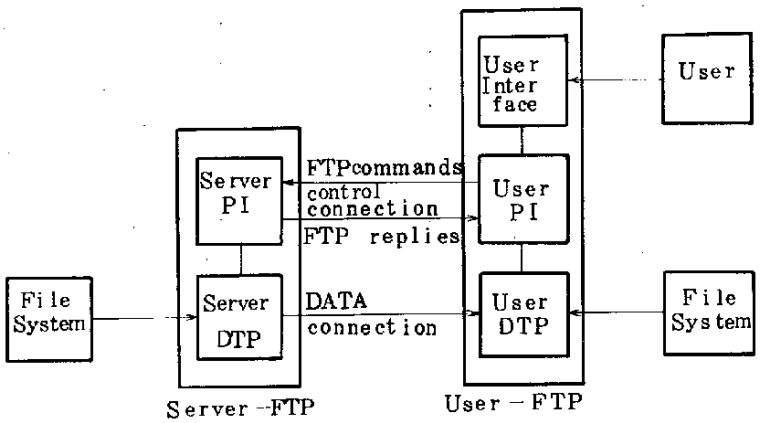



図2－15　FTPのモデル(1)

(1) Server−FTP プロセス

User−FTP あるいは他の Server−FTP と協調して File 転送の機能を果たすプロセスであり，プロトコル・インタプリータ（ＰＩ）とデータ転送プロセス（ＤＴＰ）から構成される。

(2) Server−PI

Server−PI はmy−port id＝4に対して Listen any状態にしておき，User−PI からのコネクション確立要求を持ち，コントロール・コネクションを確立する。Server−PIの役目は User−PI から FTP コマンドを受取り，その応答を返すことと，Server−DTPを管理することである。

(3) Server−DTP

Server−DTP は Server−PI から指示により転送パラメータをセットし他系の User−DTP あるいは Server の Port とのデータ・コネクションを確立し，データ転送をおこなう。Server サイトの file accss も Server−DTP の役割である。

(4) User−FTP プロセス

Server−FTP（１あるいは複数個）と協調して File 転送の機能を果たすプロセスであり，プロトコル・インタプリータ，データ転送プロセス，ユーザ・インタフェースから構成される。

(5) User−PI

User−PI は Server−FTP の port 識別番号＝４に対してイニシャル・コネクション確立の要求をしコントロール・コネクションを確立する。コントロール・コネクションを通してＦＴＰコマンドを送出し，その応答を受取る。

User−FTP が File 転送を行なう場合は，User−DTP を管理する。

(6) User−DTP

User−DTP は，User−PI からの指示により，転送パラメータをセットし Server−DTP とのデータ・コネクションを確立する。Server−FTP とのデータ・コネクションが確立されたら，Server とデータの転送をおこなう。２つの Server 間でデータが転送される時には，User−DTP は働かない。

図2−15に示すFTP の動作を以下に概説する。

User からの指示により，User−PI は Server−FTP と Initial connectionを確立する。確立されたコントロール・コネクションを通して，User−PI は Server−FTP プロセスにＦＴＰコマンドを送る。

Server−PI から User−PI への応答は同様にコントロール・コネクションを通じて返送される。

FTP コマンドは，データ・コネクションおよびFile システムへのアクセスのためのパラメータ群を規定するものであり，Port，データの表現型式，ファイルの Store，Retrieve 等の

ためのコマンドが用意されている。

User-DTP，Server-DTPはFTPコマンドの内容にしたがい転送パラメータを決め，データ・コネクションを確立する。

User-DTP，Server-DTP は指定されたパラメータにしたがって各ファイル・システムへアクセスし，データ・コネクションを通してデータを転送する。

上記のケースとは異なり，Userが存在するHOST とは異なる2つのHOST間でfile転送をおこなう場合は図2-16のようなFTPの構成になる。

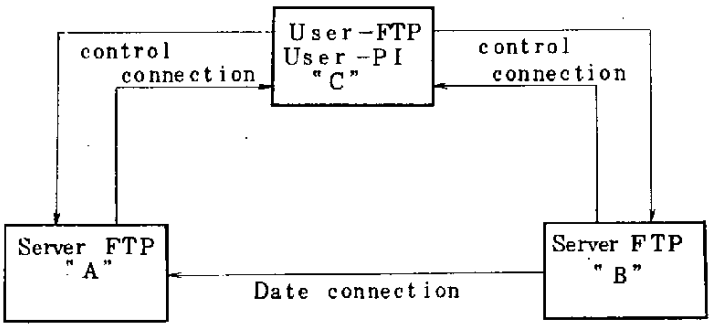



図2-16　FTPのモデル(2)

この場合は，User-FTPは2つのServer-FTP とコントロール・コネクションを確立した後，2つのServer-FTP間にデータ・コネクションを確立し，Server FTP間でデータ転送がおこなわれる。

B．データ表現型式とデータ転送方式

a．データ表現型式

データ転送においてFTPがあつかうデータの表現型式にはBinary型とCharacter型とがある。いずれの場合もDTP間では転送の単位は8ビットを1単位とするが，HOST系のコア上での表現の違いにより上記2つの型を設ける。

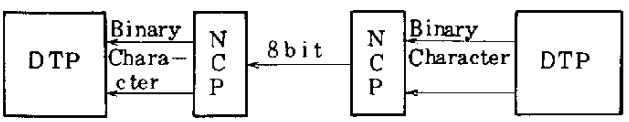



図2-17

(1) Binary 型(B)

コア上連続する8 bit 単位のbit 列を，そのままの image でデータ転送するものである。たとえば1語32 bitのHOSTと1語36 bitのHOST間で4バイト分転送すると図2−18のようになる。

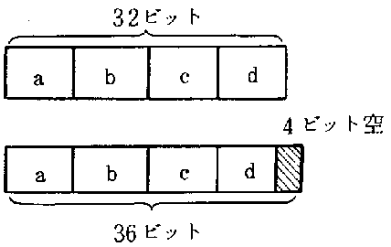



図2−18

(2) Character 型（E，J）

8 bit の転送単位をHOSTの文字表現単位に（から）あわせて変換し転送するものである。例えば1語32 bit 8 bit／ByteのHOST系と1語36ビット9 bit／ByteのHOST系との間でデータ転送すると図2−19のようになる。

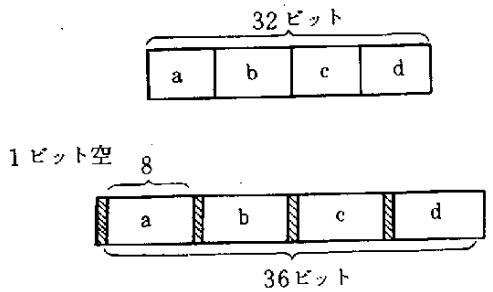



図2−19

CHARACTER型のデータは，Server FTPの存在するHOST系における標準の文字コードE（EBCDIC 8 ビット）あるいはJ（JIS 8単位）で蓄積・管理される。

User FTPの指定する文字コードが，Server FTP側の標準コードと異なる場合は，Server FTPによりコード変換が行われネットワーク・ファイルに蓄積され，あるいはUser FTPに転送される。

b．データ転送方式

ＦＴＰであつかうファイルは，複数のレコードの集りからなるレコード構造のファイルとする。

ＦＴＰはこのレコードを，データ・コネクションを通じて転送するわけであるが，レコード長がHOST－HOSTプロトコルのメッセージ長を超える場合，そのレコードは複数個のメッセージに分割されて転送される。分割した各メッセージには，そのメッセージの種類を示すDescriptorと，メッセージ内のデータ・バイト数を示すデータ・カウント等を含むメッセージ・ヘッダが付加される。

ＦＴＰであつかうレコードの最大長は2048バイトとしメッセージ最大長は1152バイトとする。

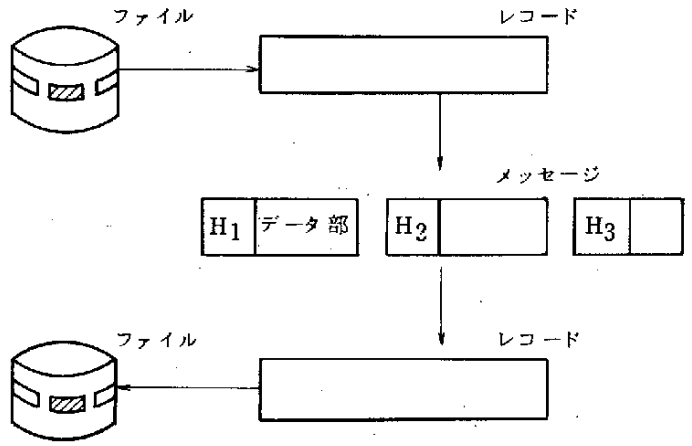



図２－20

ヘッダ情報の詳細は以下の通りである。

header format

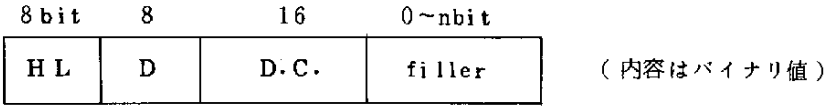



図２－21

(1) ＨＬ：header length（8bit）

ヘッダ部の長さを示し，８ビットを１バイトとするバイト数で表す。

(2) Ｄ　：Descriptor（8bit）

ヘッダ情報の種類を示す（下記以外は０）

EOR：End of Record ：1

EOF：End of file　：2

(3) D.C.：Data Count（16bit）

データ部のデータ・バイト数を表す。

(4) filler：バウンダリ調整のためのエリアで，各HOST特有のバウンダリはこのfillerビットで調整する。

(注) header formatはデータ・コネクション上でのビットパターンを示しており，データ表現型式の違いは考慮していない。したがって特有のバイト表現を有するHOSTのFTPにおいてはこのheaderの組立，解読に十分注意しなければならない。

c．Header lengthの決定法

fillerバイトは，各HOSTによってデータ入出力のためのバウンダリが違うため，そのバウンダリ調整用に設けたものである。header lengthを決めるための手順を以下に示す。

① User−PIは，転送データ表現型式をTYPEコマンドによりServerに指示する。

② Server−PIは，送られたTYPEコマンドの内容と自HOSTのデータ表現方法とから，自HOST内データ入出力バウンダリを判断し，バウンダリ・バイト数をTYPEコマンドの応答として返答する。

③ User−PIはデータ・コネクションの両端DTPのバウンダリ・バイト数の最小公倍数を計算し，その値より大きいか等しい場合はその値をheader lengthとする。

最小公倍数が4より小さい場合は4以上の公倍数をheader lengthとする。

その後User−PIはServerに対しBYTEコマンドによりheader lengthを指示する。

④ 各DTPはheader lengthを加味したメッセージ長によりデータ・コネクションを確立する。

C．コネクション

FTPには，FTPコマンドが転送されるコントロール・コネクションとデータが転送されるデータ・コネクションがある。

a．コントロール・コネクション

(1) コントロール・コネクションの確立

コントロール・コネクションはDSPのInitial connectionとDSP connectionの基本機能部分をそのまま利用することとする。但し，Server FTPが最初にListenanyしているPort−idの識別番号は4である。

FTPサービスの開始時にServer FTPは識別番号4のPort−idをNCPに対してListen any状態にしておかなければならない。

User FTPはServer FTPのPort−id = 4 に対してコネクション確立を要求する。この手順はDSPの Initial Connection と全く同一の手順とする。（図２－22）

User FTP−Server FTP間のFTPコマンドおよびその返答は，すべてコントロール・コネクションを通して転送される。

Serverはファイル転送が終了した後User FTPからの指示（BYE)によりコントロール・コネクションを close する。

User FTPが２つ目のServer FTPとコントロール・コネクションを確立する場合，User FTPは my−port−id の識別番号を 22 としてServer FTPの port−id の識別番号＝４にコネクション確立要求を行う。（図２－23）

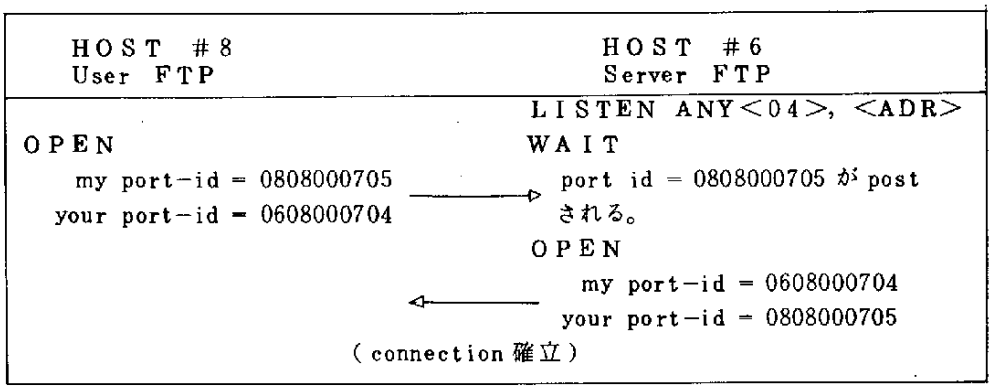



図２－22　FTP Initial Connection の確立 (1)

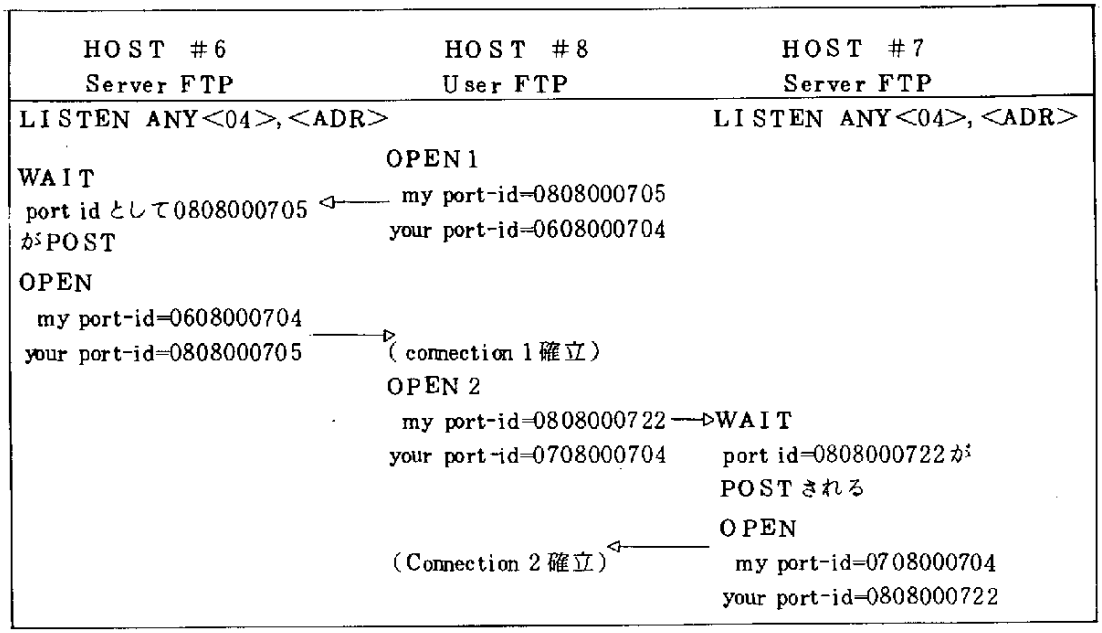



図２－23　FTP Initial Connection の確立 (2)

(2) コントロール・コネクションの切断

コントロール・コネクションは，下記のような条件が発生したら切断される。

① BYEコマンドを送受した場合

BYEコマンドを送受したUser FTP，Server FTPは，直ちにコントロール・コネクションをCLOSEする。

② User FTPあるいはServer FTPで回復不能のエラーを生じ，以後のFTP処理が続行不可能となった場合。（単独のコマンドの実行エラーは，この場合に含めない。）

この場合は，相手側にコントロール・コネクションを通してDCNをPOSTした後，データ・コネクションがOPEN中であればデータ・コネクションをCLOSEし，その後コントロール・コネクションをCLOSEする。データ・コネクションがOPEN中でなければ，DCNをPOSTした後，直ちにコントロール・コネクションをCLOSEする。

DCNを受けたUser FTPあるいはServer FTPは，データ・コネクションがOPEN中であれば，データ・コネクションをCLOSEしてから，OPEN中でなければ直ちに，コントロール・コネクションをCLOSEする。

b. データ・コネクション

データ・コネクションは，User DTPとServer DTP間あるいは2つのServer DTP間に用意されるデータ転送の為のコネクションである。

(1) データ・コネクションの確立

Server FTPは，コントロール・コネクションを通しFTPコマンドをUser FTPから受取り，転送パラメータを決め，データ・コネクションを確立する。その手順は以下に示すとうりである。

Server FTPは先ず，PORTコマンドによりデータ・コネクションの相手先port-idを知る。PORTコマンドにより知らされるport-idの識別番号は，通常$06_{16}$である。次にTYPEコマンドによりデータ表現型式がEBCDIC，JIS，Binary のいずれであるかを知る。データ表現型式と自HOSTのデータ表現方法とにより，自HOSTに最適な入出力バウンダリ・バイト数を決定し，TYPEコマンドの応答としてその値を返答する。TYPEコマンドの応答を返すと，BYTEコマンドにより，データ・コネクションのheader lengthが知らされる。さらにRETRieveあるいはSTOReコマンドにより，ファイル転送要求が送られてくるので，この時点でServerはデータ転送の方向を定め，データ・コネクションを確立する。この場合，Server DTPのmy port-idの識別番号はPORTコマンドで知らされた値が偶数の場合は1を加えたもの，奇数の場合は1を引いたものとする。

$$\text{User}\ \begin{cases} \text{my port-id} = <\text{User HOST 番号}>, <\text{user番号}>, <06_{16}> \\ \text{your port-id} = <\text{Server HOST番号}>, <\text{user番号}>, <07_{16}> \end{cases}$$

Server

my port−id = ＜自HOST番号＞, ＜PORTでのuser番号＞, ＜PORTの識別番号＋1＞＝＜07＞

your port−id = ＜HOST番号＞, ＜PORTでの〃＞, ＜PORTの識別番号＞＝＜06＞

2つのServer間でデータ転送がおこなわれる場合は，User FTPが両Server FTPとコントロール・コネクションを確立した後，両Server FTPにFTPコマンドを送出する。各Server DTPはPORT，STOR or RETRコマンド等によって指示されたHOST のDTPとのコネクションを確立する。この場合のServerのPort−idは下記の通りである。

my port−id = ＜自HOST番号＞, ＜PORTで知らされたUser番号＞, ＜PORTで知らされた識別番号±1＞

your port−id = ＜PORTで知らされたHOST＞, ＜〃＞, ＜PORTで知らされた識別番号＞

(2) データ・コネクションの切断

データ・コネクションは下記のような条件が発生したらCLOSEされる。

① header内のdescriptor = EOFを送受した場合。

EOFを送受したUser FTP，Server FTPは，互いにデータ・コネクションをCLOSEし，次のFTPコマンド処理に移る。

② RETR，STORの実行中にServer FTPがUser FTPにコマンド異常終了の応答を返した場合。

この場合は，User FTP，Server FTP共にデータ・コネクションとCLOSEし，次のFTPコマンド処理に移る。

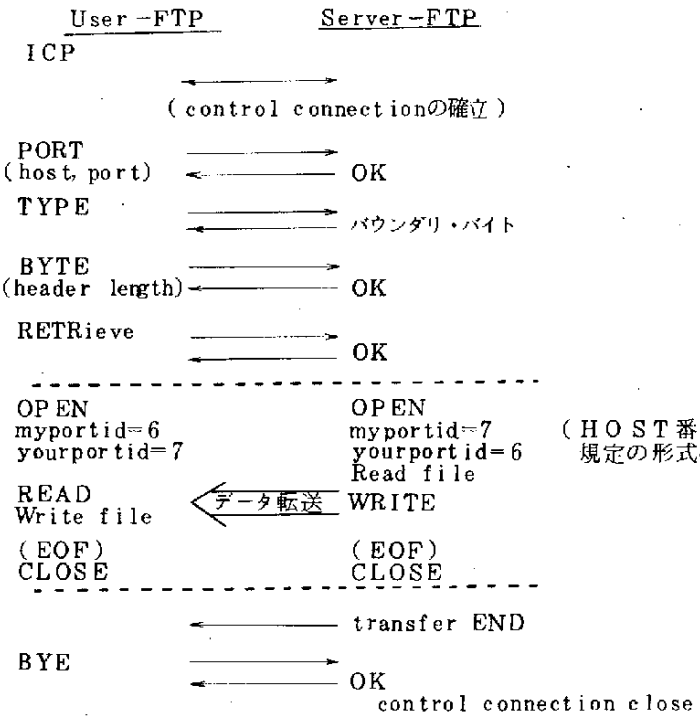
③ RETR，STORの実行中に，User FTPがコマンド実行中止を求めるIP信号（DSPに記されている。）をコントロール・コネクションを通しServer FTPにPOSTした場合。

この場合の処置は②と同様であり，Server FTPはGA付の異常終了2次応答を返した後，データ・コネクションをCLOSEする。

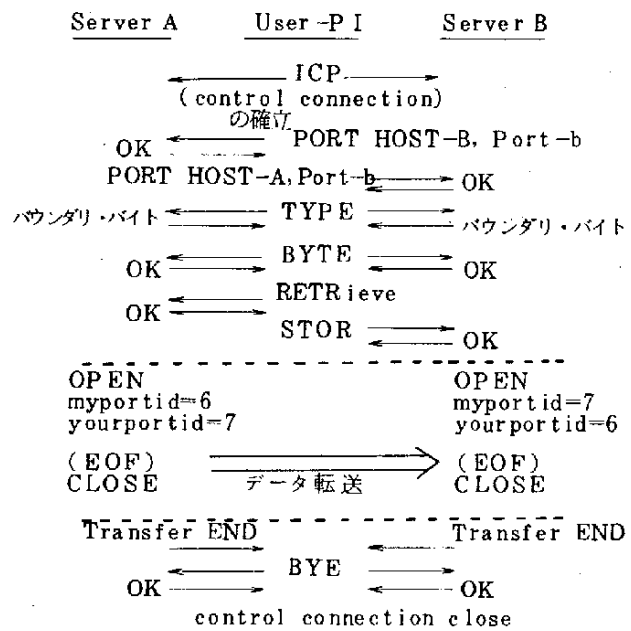
④ RETR，STOR実行中にDCN信号を送受（POST）した場合。

この場合の処置はコントロール・コネクション切断の手順に記す。

データ・コネクションの確立例　1




データ・コネクションの確立例　2



D. FTP コマンド

Server PI はコントロール・コネクションを通して，User-PI から各種のコマンドを受取り，その応答を返送する。

FTP コマンドは NVT の CR を最終文字とする NVT 文字ストリングによって表わされる。

FTP コマンドは FTP アクセス・コントロール用のコマンドと，データ転送のパラメータ設定用のコマンド，FTP サービスを要求するコマンドの 3 グループのコマンド群からなる。

（FTP コマンドの形式は表 2－1 参照）

表 2－1　FTP コマンドの形式

USER SP ＜user name＞CR

BYE CR

PORT SP ＜port－id＞CR

TYPE SP ＜type code＞CR

BYTE SP ＜header length＞CR

RETR SP ＜file identifier＞CR

STOR SP ＜file identifier＞CR

CHAN SP ＜file identifier＞, ＜file identifier＞CR

DELE SP ＜file identifier＞CR

LIST 〔,P〕 CR

＜user name＞ :: =＜string＞

＜string＞ :: =＜char＞|＜char＞＜string＞

＜char＞ :: = CR を除く NVT コード

＜port id＞ :: = JIPNET port－id を示す 16 進 10 桁の NVT ストリング

＜type code＞ :: = B | E | J

＜header length＞ :: = データ・コネクションヘッダ長で 10 進 2 桁の NVT ストリング

＜file identifier＞は次を参照

＜file identifier＞

file identifier の指定方法は下記の通りとする。

（file name 〔, $\{\frac{D}{S}\}$〔,P〕〕）

file name : 8 文字以内の英数字でファイル名称を示す。

D : データ・ファイル（default 値）

S : ソース・プログラム・ファイル

実際は言語の種類により下記の値をとる。

F：FORTRAN
C：COBOL
A：ALGOL
P：PL／I
S：Assembler
O：その他

P：Public（共用）file

Pが指定のファイルは，他のユーザも自由に参照可能であり，Read only file である。

ファイル・ネームの変更，ファイルの削除は，その Public file を作成したユーザのみが行える。

（注１） ソース・プログラム・ファイルについて

ソース・プログラム・ファイルは，データ・ファイルの特殊なものであり，80バイトの整数倍の長さのレコードとして転送する。各ＨＯＳＴでは，ソース・プログラム・ファイルを，そのＨＯＳＴシステム内での言語プロセッサに入力可能なソース・ライブラリとして扱う事ができる形式で蓄積・管理しなければならない。

（注２） ファイルのアクセス管理

ファイルのアクセス管理はユーザ毎に username･file name という形で管理し，共用ファイルは username をＪＩＰＮＥＴとして扱う。

アカウンティングは，username 毎に行う。

a．FTPアクセス・コントロール・コマンド。（　）内は command code

(1) User name（ＵＳＥＲ）

パラメータとしてUser の識別名をNVTストリングで指定する。これは，fileシステムへのアクセス・コントロール用に使用され，コントロール・コネクション確立後最初に転送されるコマンドである。

(2) ＢＹＥ（ＢＹＥ）

パラメータなしのコマンドでコントロール・コネクションを，CLOSEする。

b．転送パラメータ設定用コマンド

(1) port（ＰＯＲＴ）

パラメータとして port id を持つ。

PORTコマンドは Server に相手DTPの port id を知らせるためのコマンドで，16進10桁のNVTストリングで port id を表す。User（使用者）からの特別な指定がない限り，

User−PIが送るport−id識別番号は6である。

(2) Representation type（ＴＹＰＥ）

パラメータとしてデータ表現型式（2.1）を表すNVT文字BあるいはE，Jを指定する。

Server−PIはTYPEコマンドの応答として自HOSTの入出力バウンダリ・バイト数を返答する。

(3) header byte size（ＢＹＴＥ）

パラメータとして10進2桁のNVTストリングでheader lengthを示す。DTPはheader lengthを加味してmessage sizeを決める。

c．ＦＴＰサービス・コマンド

(1) Retrieve（ＲＥＴＲ）

パラメータとしてfile識別情報を持つ。

RETRコマンドを受取ったServer DTPは，指定されたfileのコピーをUser−DTPあるいは他のServerにデータ・コネクションを通して転送する。

転送後，fileの状態，内容を変更してはならない。

(2) Store（ＳＴＯＲ）

パラメータとしてfile識別情報を持つ。

STORコマンドを受取ったServer DTPは，データ・コネクションを通して転送されてくるデータをfileとして記録する。指定されたfileと同名のfileが存在する場合は異常終了する。

同名のfileがない場合は新しくfileを創成する。

(3) Change file identifier（ＣＨＡＮ）

パラメータとしてfile識別情報を2組持つ。

先頭のfile識別情報のファイル名称，属性を後続のfile識別情報のfile名称，属性に変更する。

但し，ソース・プログラム・ファイルとデータ・ファイル間で属性を変更することはできない。

(4) Delete file identifier（ＤＥＬＥ）

パラメータとしてfile識別情報を持つ。

このコマンドを受取ったServer FTPは，指定されたfileを消去する。

(5) List（ＬＩＳＴ）

パラメータなし，あるいはパラメータとしてP（public）を持つ。

このコマンドを受取ったServer FTPは，そのユーザのファイル識別情報一覧をコントロール・コネクションを通して送る。ファイル識別情報は応答コードに続きＳＰで区切り，

5個連続させ，最後にＣＲを置いたものを１セグメントで送出する。最終セグメント以外は１次応答コードを使用し，最終セグメントは２次応答コードを先頭に置く。

Ｐ指定の場合は，ファイル・アクセスのためのユーザ名をＪＩＰＮＥＴとして扱う。

(ｲ) コマンド動作の途中打切

ＲＥＴＲ，ＳＴＯＲ，ＬＩＳＴコマンドに関しては，これらのコマンドの動作途中で,打切りの指示を行うことができる。

打切りはUser FTPがServer FTPにＩＰ（Interrupt Process）信号をコントロール・コネクションを通して発信することによって行われる。ＩＰ信号を受けたServer FTPはコマンド動作を途中で打切り，コマンド異常終了の応答をuser FTPに返す。ＲＥＴＲ，ＳＴＯＲ，ＬＩＳＴコマンド動作時以外にServer FTPがＩＰ信号を受けた場合は，そのＩＰ信号を無視する。

d．ＦＴＰの応答

Server－IPはUser－IPから送られてきたFTPコマンドに対し何らかの応答を返さなくてはならない。

応答は３桁の数字からなる応答コードと必要に応じて説明用の文字ストリングが付加された形式をとり最終文字位置にはＣＲＬＦを置く。（いずれもNVTコード）

- nnn ＣＲＬＦ
- nnn …… ＣＲＬＦ

応答の形式

また，コマンドによっては，コマンドを正しく受取ったという１次応答を返した後，コマンド動作終了後２次応答を返すという２段階の応答形態をとる場合がある。

３桁の応答コードは先頭の１桁で応答の大まかな分類を示し，下２桁でその細分類を示す。（現在Ｘは何でも良い）

０ＸＸ：コマンド動作正常終了

２ＸＸ：コマンドを受取った（２次応答あり）

５ＸＸ：コマンドのシンタックスエラー

６ＸＸ：コマンドの実行は不可能

７ＸＸ：コマンドの動作が異常終了

各コマンドに対する応答を表２－２に示す。

表 2 － 2

| コマンド | 応答コードの1桁目 | | |
|---|---|---|---|
| | 1次 / 2次 | 正常 | 異常 |
| USER | | 0 | 5，6 |
| BYE | | 0 | 5，6 |
| PORT | | 0 | 5，6 |
| TYPE | | 0 | 5，6 |
| BYTE | | 0 | 5，6 |
| RETR | 1 次 | 2 | 5，6 |
| | 2 次 | 0 | 7 |
| STOR | 1 次 | 2 | 5，6 |
| | 2 次 | 0 | 7 |
| LIST | 1 次 | 2 | 5，6，7 |
| | 2 次 | 0 | 7 |
| CHAN | 1 次 | 2 | 5，6 |
| | 2 次 | 0 | 7 |
| DELE | 1 次 | 2 | 5，6 |
| | 2 次 | 0 | 7 |

e．FTPコマンドと応答のやりとり

(1) Server FTPは，User FTPとコントロール・コネクション確立後，GA付のサービス開始メッセージをUser FTPに送出しUser FTPからのFTPコマンド入力待とする。

そのフォーマットは下記の通りである。

SMH　GA　メッセージ（132字以内）　CR

(2) User FTPはServer FTPからGAを受けたらFTPコマンドを送信する。

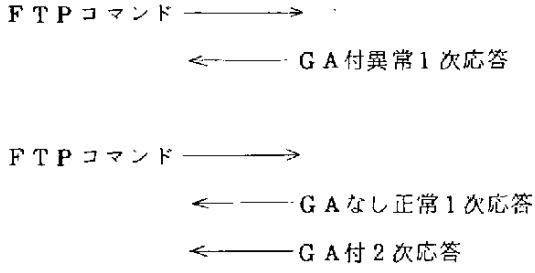
(3) Server FTPがFTPコマンドを受けた後User FTPに返す応答は，1次応答のみのコマンドに対してはGA付の1次応答，2次応答のあるコマンドに対しては正常1次応答はGAなし，異常1次応答はGA付とし，2次応答はGA付とする。

① 1次応答のみのコマンドの場合

FTPコマンド ——→

←—— GA付1次応答

② 2次応答のあるコマンドの場合




(4) コマンド動作の途中打切は前述のＩＰ信号のＰＯＳＴによりおこなう。

# 3. ACOS－NCP

# 3. ACOS－NCP

ACOS－NCPはACOSシステム700のオペレーティングシステムACOS－6の管理下で動作し，OSモジュール的性格のNCPEXECとプロセスにサブルーチンとして結合され，NCPEXECと連絡をとるNCPDISの協同作業でプロセスにプロセス間通信機能を提供する。

ネットワークサービスを受けたいユーザはシーケンシャルファイルアクセスに類似したサービスコマンドをプログラム内で書き，さらにNCPDISをリンケージバインドするだけでよい。

NCPが存在することによるオーバヘッドとして，コアに関して約19k語であるが，そのうちユーザプロセスに関するものはNCPDISの1k語弱である。

またCPUオーバヘッドとしては144バイトメッセージの送受ごとにそれぞれ約23msec，16msec必要である。

なお本NCPの作成に約14人月要した。

## 3.1 ACOS－6

ACOS－6はACOSシステム600／700用のオペレーティングシステムであり

① ローカルバッチ

② リモートバッチ

③ 会話型リモートバッチ

④ タイムシェアリングシステム

⑤ メッセージ交換処理

を多重に行ない，NCPはローカルバッチの1ジョブとして実行される。

ACOS－6の構成は図3－1に示すとおりであり，その中でシステム管理は中核をなし次のような機能を提供する。

(1) ジョブ管理

① ジョブの受け付け，実行および結果の出力。

② システム稼動状況の報告。

③ アカウント情報の出力。

(2) 資源管理

① ジョブの分割単位であるアクティビティに対する資源の割り当てと管理。

② アクティビティのロールイン／ロールアウトとメモリコンパクションの管理。

(3) 実行管理

① 入出力処理やプログラムのロードをはじめとする種々のプログラムサービス。

② ハードウェアシステムの状態把握，エラー処理。

(4) その他

オペレータとの交信，システム状態の表示，エラーロギング，オンラインテスト機能。

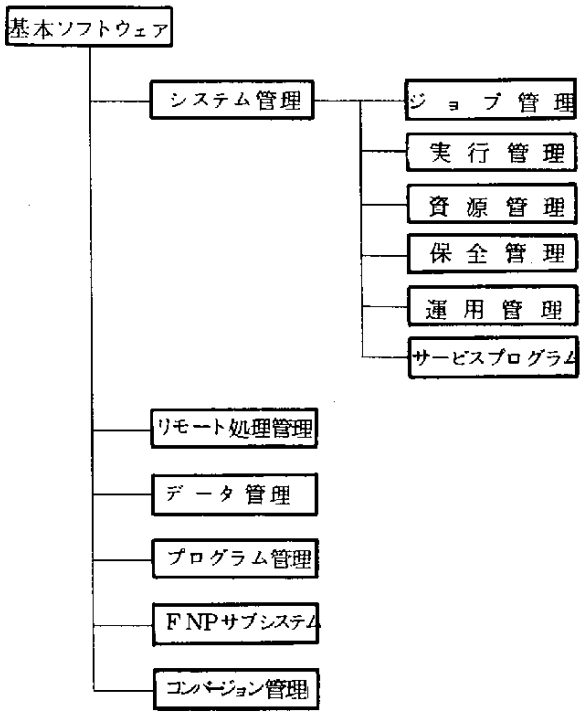



図３－１　基本ソフトウェア体系内でのシステム管理の位置

### 3.1.1 ジョブの実行とアクティビティ

ジョブとはACOS－6が利用者の仕事を処理する単位であり，アクティビティとはこのジョブを構成する単位である。ジョブの実行にあたって，システムはアクティビティをジョブの分割単位かつ資源（コア，ファイル，ＣＰＵ）の割り当て単位として管理している。

図３－２に示すように，１つのジョブを構成する幾つかのアクティビティは，原則として並べられた順に逐次実行される。

なおＮＣＰはアクティビティをプロセスと認識してサービスしており，以後の説明に出てくるサービスアクティビティとはＮＣＰがサービスしているアクティビティのことである。

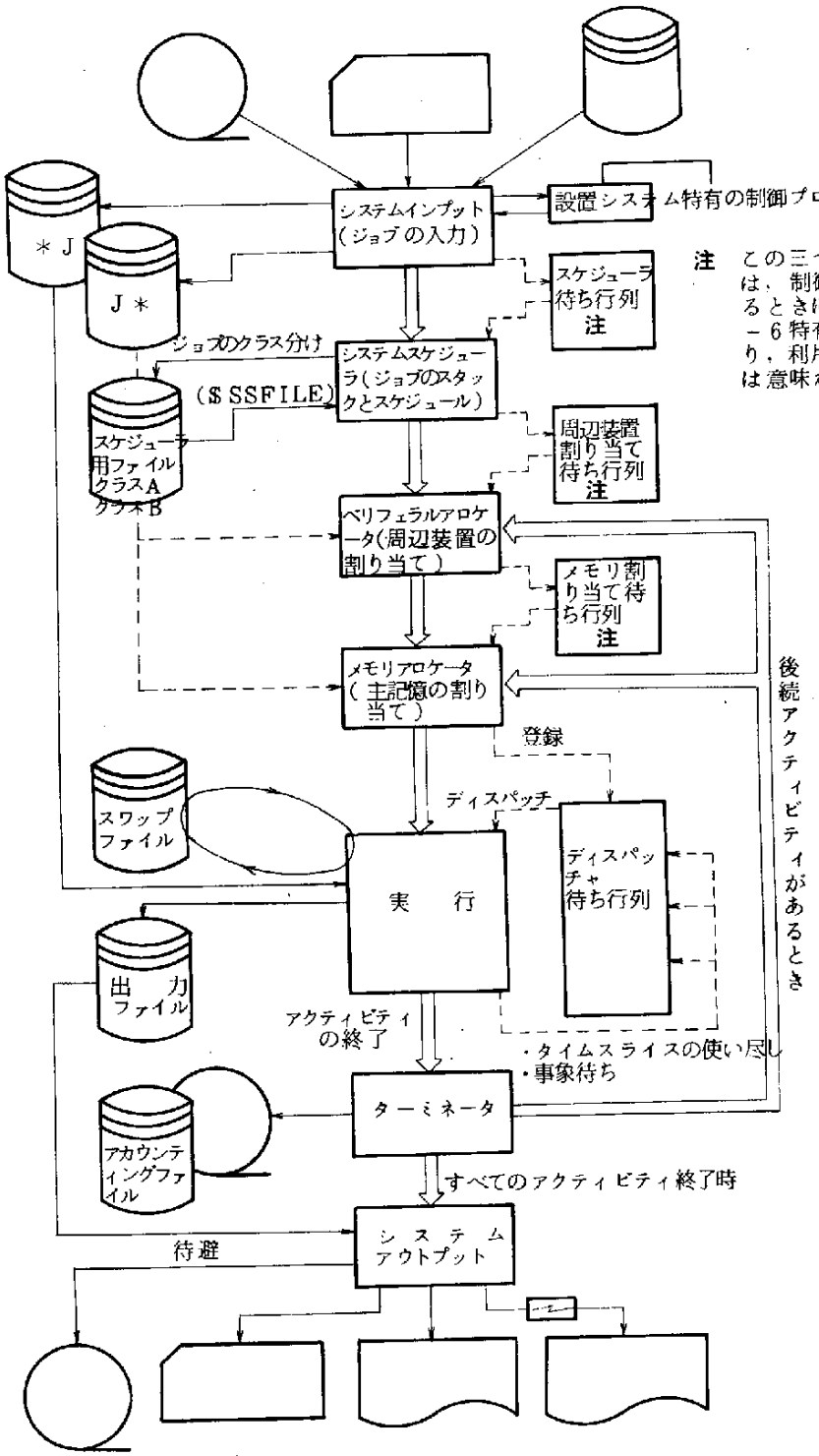



図3－2　ACOS－6ジョブフロー

A．アクティビティの種類

アクティビティにはスレーブアクティビティと特権スレーブアクティビティの２種がある。

(1) スレーブアクティビティ

スレーブモードでのみ動作するアクティビティであり，通常のユーザプログラムはこれにあたる。

(2) 特権スレーブアクティビティ

マスターモードでも動作できるアクティビティであり，TSSエグゼクティブ，NCPEXECはこれにあたる。

スレーブモード，マスターモードはそれぞれハードウェアの動作モードであり，その相異を表３－１に示す。

表３－１　マスターモードとスレーブモードの相異

| 項　目 | マスターモード | スレーブモード |
|---|---|---|
| マシン命令の実行 | 全ハードウェア命令が使える。 | 限られた命令（割込み禁止は含む）しか使えない。 |
| メモリアクセス | メモリプロテクションは働らかないので全メモリにアクセスできる。 | メモリプロテクションが働らき，許されたメモリしかアクセスできない。 |
| ベースレジスタによるリロケーション | オペランドの絶対アドレスを求める際，ベースレジスタによる修飾がされない。 | 修飾がされる。 |

B．アクティビティの実行レベル

アクティビティはメインレベル，コーテシィコール（ＣＣ）レベルの２つのレベルで実行される。

(1) メインレベル

アクティビティが通常実行されるレベルであり，ＣＣレベルの実行により中断され，ＣＣレベルの終了によって再開される。

(2) コーテシィコール（ＣＣ）レベル

ＩＯコマンドとともに特定アドレスを指定しておくと，ＩＯ終了後そのとき実行していたメインレベルの処理を中断し，特定アドレスに制御を移しＣＣレベルで実行される。ＣＣレベルの実行を終了する（MME　GEENDC命令による）と，他にＣＣレベルの実行要求がなければ中断していたメインレベルを実行しはじめる。

ＣＣレベルの実行はＯＳにより優先的にスケージュールされ，同一アクティビティにおけるＣＣレベルの実行は排他制御される。

図３－３にアクティビティの実行レベル遷移を示す。

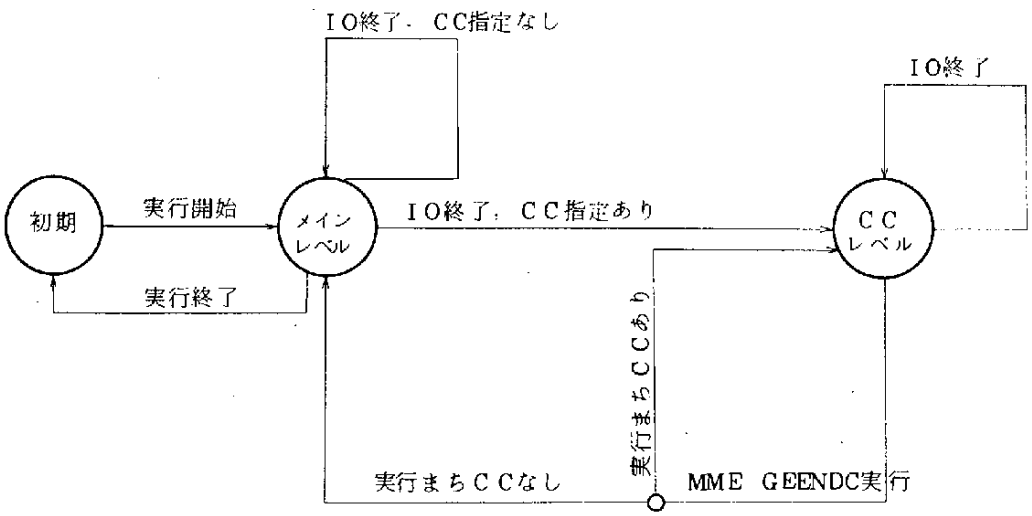



図３－３　実行レベル遷移

### 3.1.2　ＡＣＯＳ－６主記憶レイアウト

図３－４はACOS－6 の主記憶レイアウトを示したものである。

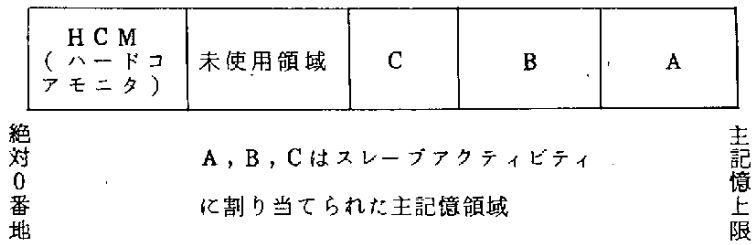



図３－４　ACOS－6主記憶レイアウト

HCM（ハードコアモニタ）とは，ACOS－6 の常駐部分を意味し，その中には次のようなものが含まれている。

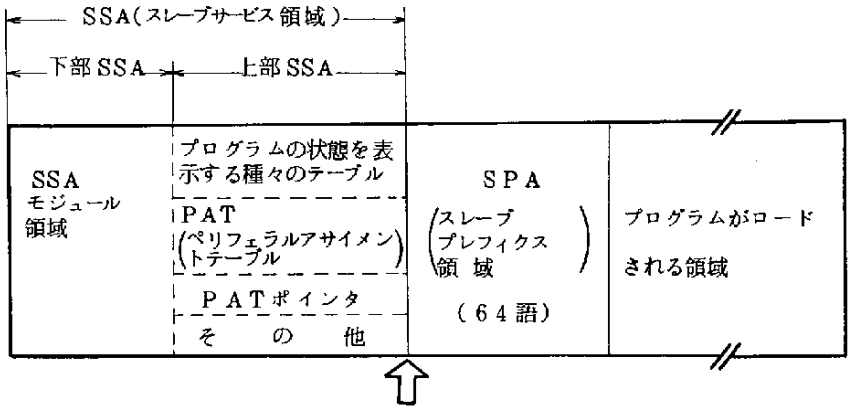
(1)　システムモジュール

ＣＰＵディスパッチャー，割込み処理ルーチン，入出力スーパバイザー，コアアロケータ（厳密な意味ではＨＣＭではないが，コアに常駐しているのでＨＣＭとみなしてもよい）等のＯＳ用常駐モジュール。

(2) ＯＳ用テーブル

システム構成テーブル，コアアロケータ・キュー，ＣＰＵディスパッチ・キュー等のＯＳ用テーブルおよびキュー。

図３－５は１つのアクティビティに対して割り当てられる主記憶領域を拡大したものである。図を見るとわかるように，各アクティビティに対してＳＳＡと呼ばれる領域が割り当てられ，ＳＳＡモジュールがこの中にロードされる。ＳＳＡモジュールとは各アクティビティが固有に要求する種々のサービスを行なうものであり，システム全体からみてあまり利用されないＯＳモジュールはこのＳＳＡモジュールとして実装され，ＨＣＭのサイズをできるだけ小さくすることができる。ＳＳＡの大きさは１ｋ語または２ｋ語である。




相対０番地：ＢＡＲ（ベースアドレスレジスタ）はここを指す。

ＳＳＡ：通常１ｋ語の大きさをもち，アクティビティの固有情報をもつ。

下部ＳＳＡ：512語の大きさをもち，ＳＳＡモジュールがロード実行される。

上部ＳＳＡ：プログラム（アクティビティ）の状態を表示する種々のテーブルと，アクティビティに対してどのような周辺装置やファイルが割り当てられているかを表示するＰＡＴとＰＡＴポインタをもつ。

ＳＰＡ：ACOS－６とプログラムとの通信領域である。

図３－５　アクティビティに対し割り当てられる主記憶領域

### 3.1.3　INTERCOM機能

INTERCOM機能はACOS－６が提供している擬似ＩＯによるアクティビティ間通信機能であ

り，これはプロセス間通信機能に他ならない。これによりあるジョブのアクティビティから別のジョブのアクティビティへデータを移送することができる。その際，両方のアクティビティは同時に主記憶上に存在し，１つのアクティビティはあるインターコムファイルに対しRead コマンドを出し，他方はWriteコマンドを出すという約束が守られなければならない。両コマンドの一致が取れたときコアtoコアでデータが移送される。原則として両コマンドの同期はACOS－6 が行なう。図３－６はINTERCOMによる通信の様子を示す。

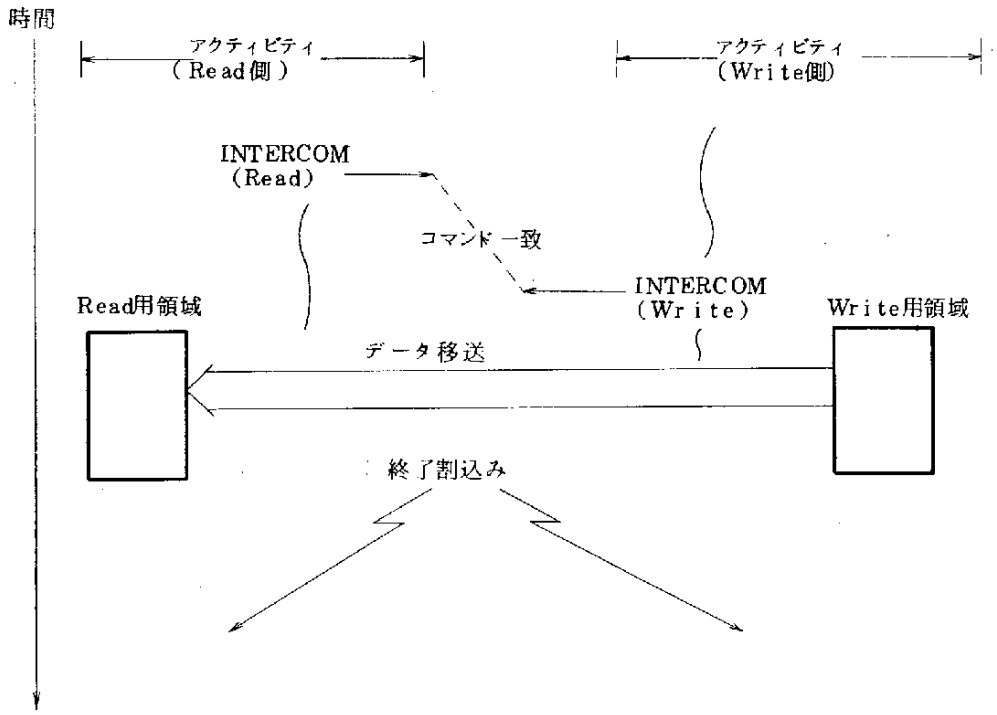



図３－６　INTERCOM通信例

## 3.2　ＮＣＰの概要

ＮＣＰは３つのアクティビティと１つのサブルーチンからなり，サービスアクティビティ（プロセス）にプロセス間通信機能を提供する。以下に各モジュールの説明を行なう。

(1)　ＮＣＰＤＩＳ

(a)　形態：プロセス間コミニケーションを意図する各アクティビティにサブルーチンの形で結合される。

(b)　サイズ：0.8 or 0.6 k語

(c) 機能：アクティビティが出したサービスコマンドから制御情報を作り，インターコムファイルを通してNCPEXECへ通知する。またNWAIT，SWAITサービスコマンドを通して入出力コマンド等の終了を管理する。

(2) NCPEXEC

(a) 形態：システムスタートアップ時にスポーン（Spawn）され，優先度が高くロールアウトされることがまれな特権スレーブアクティビティである。

(b) サイズ：15k語＋1k語（SSA）

(c) 機能：NCPの中枢であり，通知された制御情報を解読してコネクションの開閉，メッセージの転送を管理する。またサービスアクティビティの監視を行ない，必要とあればサービスアクティビティのロールイン，強制終了を行なう。

(3) NCPSUB

(a) 形態：NCPEXECによってスポーン（Spawn）され，優先度が高くロールアウトされることがまれなスレーブアクティビティである。

(b) サイズ：1k語＋1k語（SSA）

(c) 機能：2秒毎にタイマ制御情報をインターコム経由でNCPEXECへ通知し，NCPEXECはそれをトリガとして各種時間監視を行なう。

インターコムファイルを経由してNCPEXECよりアカウンティング情報，コンソール出力情報を受けとり，前者はアカウンティングファイルへ書き込み，後者はコンソールへ出力する。

(4) NCPCN

(a) 形態：コンソールよりスポーン（Spawn）されるスレーブアクティビティ。

(b) サイズ：1k語＋1k語（SSA）

(c) 機能：コンソールよりNCPに対する種々の指示を与えたいときにスポーンされ，タイプインされたコマンドをインターコム経由でNCPEXECへ通知する。

以上述べたNCP構成モジュールとモジュール間データフローを図3－7に示す。

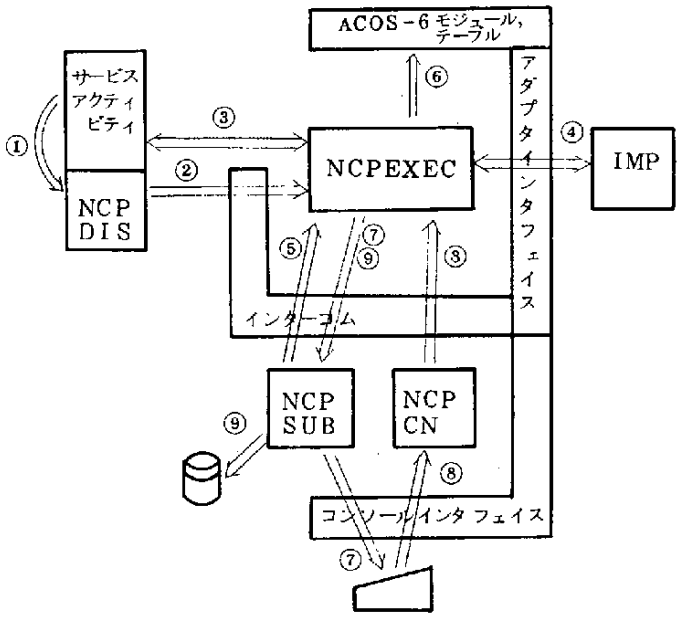



図３－７　ＮＣＰ構成ならびにデータフロー

## 3.3　ＮＣＰの運用

### 3.3.1　ＮＣＰのスタート

ACOS－NCPをスタートする方法を次に示す。

① リクエストキィを押しコンソールより次のコマンドを入力する。

ＳＰＡＷＮ　ＮＣＰ

応答として次のメッセージが出力され，ＮＣＰはスタンドバイ状態となる。

＊Ｓ＃ＮＣＰ　ＡＳＫＳ　ＰＲＩＶＩＴＹ，ＩＤ＝ＡＣＯＳ－ＮＣＰ　．．ＲＵＮ？

② リクエストキィを押しコンソールより次のコマンドを入力することによりＮＣＰはスタンドバイ状態から抜け実行に入る。

ＲＵＮ　ＮＣＰ

実行を開始したＮＣＰは諸々の初期処理が終りレディ状態になると次のメッセージをコンソール上に表示することによりスタートアップが完了したことを示す。

ＮＣＰ　ＲＥＡＤＹ

### 3.3.2 NCPのストップ

ACOS−NCP のサービスを終了さす手順を次に示す。

リクエストキィを押しコンソールより次のコマンドを入力することによりＮＣＰにコンソールより指示を与えるためのプロセスNCPCNを起動する。

ＳＰＡＷＮ　ＮＣＰＣＮ

NCPCNは起動されるとコンソールに対して次のメッセージを出し，NCPに対する指示コマンドのタイプインを要求する。

＊＊ＰＬＥＡＳＥ＊ＮＣＰ＊＊

応答としてＮＣＰにサービス終了を指示する次のコマンドをタイプインする。

ＴＥＲＭ　ＮＣＰ；

応答をタイプインするとＮＣＰＣＮは，再び指示コマンドのタイプインを要求する次のメッセージを出力する。

＊＊ＰＬＥＡＳＥ＊ＮＣＰ＊＊

応答としてコマンドの入力終了を意味する次のコマンドをタイプインする。

ＥＮＤ；

終了要求を受けたＮＣＰは諸々の終了処理を行なった後，コンソールに次のメッセージを出力後ターミネーションする。

ＮＣＰ　ＳＥＲＶＩＣＥ　ＥＮＤ

### 3.3.3 オペレータコマンド

コンソールよりＮＣＰに対して種々の指示を与える為に，オペレータコマンドが用意されており，それはNCPCNをスポーンして行なう。

(1) ＮＣＰＣＮスポーン

コンソールよりリクエストキィを押して次のコマンドをタイプインすることによりスポーンできる。

ＳＰＡＷＮ　ＮＣＰＣＮ

(2) 指示コマンドのタイプイン

起動されるとＮＣＰＣＮは次のメッセージをコンソール上に出力し指示コマンドのタイプインを要求する。

＊＊ＰＬＥＡＳＥ＊ＮＣＰ＊＊

応答として指示コマンドをタイプインする。

指示コマンドをタイプインするとＮＣＰＣＮは再び指示コマンドのタイプインを要求してくる。

これは応答としてＥＮＤコマンドをタイプインするまで続けられる。

(3) 指示コマンド

① ＩＭＰ　ＷＡＫＥ　ＵＰ；

何らかの原因でＩＭＰがdownし，その後ＩＭＰがwake upした場合にＮＣＰに知らせるのに用いる。

② ＦＥＴＣＨ　ＸＸＸＸＸＸ，ＸＸ；

第1パラメータ　第2パラメータ

Ｘ……Ｘ　は8進数

NCPEXECの第1パラメータで示すアドレス以降，第2パラメータで示す長さだけの内容を8進数でコンソール上に出力する。

③ ＰＡＴＣＨ　ＸＸＸＸＸＸ，ＸＸＸＸＸＸＸＸＸＸＸＸ；

NCPEXECの第1パラメータで示すアドレスの内容を，第2パラメータで示す値で置きかえる。

④ ＥＮＤ；

指示コマンドのタイプインを終了することを示す。

⑤ ＤＩＳＰ；

現在ＮＣＰと会話中のアクティビティのSNUMB（ＪＯＢ識別番号）をコンソール上に出力する。

＊＊＊ＣＯＮＮＥＣＴ　ＡＣＴＩＶＩＴＹ＊＊＊

アクティビティのＳＵＭＢ

｝

＊＊＊ＤＩＳＰＬＡＹ　ＥＮＤ＊＊＊

⑥ ＴＥＲＭ　ＮＣＰ；

ＮＣＰサービスをやめることを通知する。このコマンドを入れたときにサービス中のアクティビティがなければＮＣＰは直ちに終了する。

サービス中のアクティビティがあれば，そのアクティビティに対するサービスを終った後に終了する。その間，新たなアクティビティのCNTNCP要求はrejectされる。

⑦ ＳＷＥＥＰ；

コネクト中の全てのアクティビティとの接続を切る。TERM NCP コマンドを入れた後，直ちにＮＣＰを終らせたい場合に主として用いる。

## 3.4　サービスコマンド

### 3.4.1　サービスコマンドの概要

ＮＣＰは各アクティビティにプロセス間コミニケーションを行なわせしめる為に次のコマンドを用意している。

(1)　ＣＮＴＮＣＰ

サービスアクティビティがＮＣＰと交信することを宣言する。

(2)　ＮＯＰＥＮ

コネクションの開設を要求する。

(3)　ＮＲＥＡＤ

コネクションを介して伝送されてきたメッセージを読み込む。

(4)　ＮＷＲＩＴＥ

コネクションを介してメッセージを送信する。

(5)　ＮＣＬＯＳＥ

コネクションの閉さを要求する。

(6)　ＬＩＳＴＥＮ

ポート名を指定して，そのポート名を対象としたコネクション確立要求があった場合に，相手側のポート名を通知してもらう。

(7)　ＬＩＳＡＮＹ

ポート識別番号を指定して，そのポート識別番号を持つポート名に対してコネクション確立要求があった場合に相手側のポート名を通知してもらう。

(8)　ＲＥＦＵＳＥ

ＬＩＳＴＥＮ，ＬＩＳＡＮＹが完了したとき，コネクション確立を拒否したいときに用いる。

(9)　ＳＥＴＥＣＢ

ＣＣＢ名を指定して，他プロセスからのイベント待ちを宣言する。

(10)　ＮＰＯＳＴ

ＣＣＢ名を指定して，他プロセスにイベントを通知する。

(11)　ＮＷＡＩＴ

未終了のコマンドが全て終了するまで，サービスアクティビティの実行権を放棄する。

(12)　ＳＷＡＩＴ

未終了のサービス・コマンドが１つ以上終了するまで，実行権を放棄する。

(13)　ＳＥＴＩＯＳ

ＧＥＩＮＯＳ命令で出したＡＣＯＳ個有の入出力の終了をネットワークサービスコマンドＮＷＡＩＴ，

SWAITで待つことを可能とする。

(14) ECBFRE

指定したecbをecb chainからはずす。

(15) BYENCP

サービスアクティビティがNCPとの交信を終了することを宣言する。そのとき未終了のLISTEN，LISANYはキャンセルされる。

なお，サービスコマンドを使うにあたっては次のことに注意しなければならない。

(1) CNTNCP，BYENCP，SWAIT，NWAIT，ECBFRE以外のコマンドを発すると，NCPDISに情報が取り込まれ，対応したecbがwait control block（NCPDIS内にある）へつながれた後，直ちにコマンドを発した次のアドレスに制御が戻る。これらのコマンドの終了は原則としてNWAIT，SWAITで待たなければならない。コマンドの終了情報はNWAITあるいはSWAITがとけたときにecb領域に入っている。これらのコマンドのecb領域をパラメータとしてECBFREを発した場合は該当コマンドの終了をNWAIT，SWAITで待つ必要はなく，その終了の確認はユーザに任される。

(2) NREAD，NWRITE，SETECBコマンドを発す場合，同一コネクションに関する同一コマンドを前に発していた場合，そのコマンドの終了を待ってから発せねばならない。

(3) サービスコマンドをコーテシィコール中で使用してはいけない。

(4) NCPDISをオーバレイの対象としてはいけない。

(5) ecb領域はコマンドが発せられたときに0 clearされ，コマンドが終了すると各々のコマンドに該当した情報が通知される。ユーザはこの領域の内容を変えてはいけない。

(6) NREAD，NWRITEのパラメータとなる入出力用データ領域は必ずダブルワードバウンダリで始まり，長さは2語の倍数でなければならない（たとえ1バイトのデータのみ送受信するのであっても2語分の領域を用意する）。

(7) いかなるサービスコマンドを用いても，情報をレジスタに入れて戻るものは，そのレジスタを除いて，メインルーチンにおけるレジスタの値は保障される。

### 3.4.2 サービスコマンド解説

サービスコマンドの詳細な説明を次に示す。

終了情報の説明の後に（i，j）で示されているのは，それぞれ

i：DN …… NCPDISがこのビットをセットした

NE …… NCPEXEC 〃

j：W …… 警告エラーである。

F …… 致命的エラーであり，サービスアクティビティはアボートする。

を示し，j＝Fのときにはコンソールに「DN」を表示してアボートする。その場合$75_{(8)}$番地の上位半語に最後にサービスコマンドを呼んだサービスアクティビティのアドレスが入っている。また$76_{(8)}$番地にはコマンドの種類，$77_{(8)}$番地にはecb領域に入っているのと同じ情報が入っている。

A．CNTNCPコマンド

(1) 機　能

サービスアクティビティがNCPと交信することを宣言する。このコマンドは全てのNCPサービスコマンドを出す前に使用しなければならない。

(2) 書き方

CALL　CNTNCP（CNCB番地）

(3) パラメータ

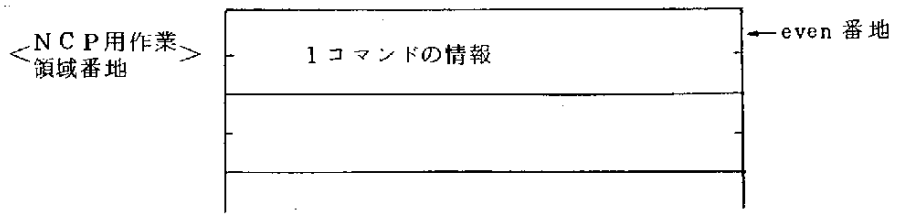
(イ) CNCB番地：NCPに渡すべきパラメータがセットしてあるコネクトネットワークコントロールブロックの先頭番地

| | | |
|---|---|---|
| ＜CNCB番地＞ | ＜ecb番地＞ | MBZ |
| ＋1 | ＜NCP用作業領域番地＞ | ＜作業領域サイズ＞ |

(ロ) ＜ecb番地＞：CNTNCP用ecb領域の先頭番地であり，コマンド終了情報が通知される。

| | |
|---|---|
| ＜CNTNCP用ecb番地＞ | |
| ＋1 | |
| ＋2 | |
| ＋3 | |

(ハ) NCP用作業領域番地：NCPが使用する作業領域の先頭番地で，この作業領域はNCPがアクティビティの出したコマンドを直ちにNCPEXECへ通知できない場合に，そのコマンド情報をキューイングしておく領域であり2語1組である。



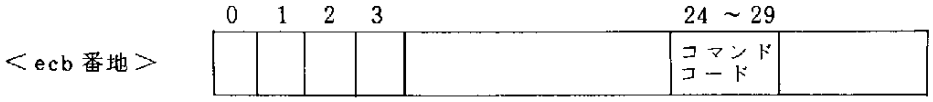

(ニ) 作業領域サイズ：NCP用作業領域の組数を指定する（0～60）。

作業領域サイズ ← waitするまでに出されたNREAD，NWRITE，NCLOSE，SETECB，NPOSTの数 －1

(4) コマンドの終了

このコマンドを発するとコマンドが終了するまでアクティビティに制御は戻らない。

コマンドが終了したときにはecb領域に終了情報が入っている。




コマンドコード：$1_8$

0＝1：終了情報がＰＯＳＴされた

1＝1：異常終了
　＝0：正常終了

2＝1：ＮＣＰとコネクトが取れない　　（ＤＮ，Ｗ）

3＝1：ＮＣＰ　busy　　（ＮＥ，Ｗ）

4＝1：ＮＣＰ使用権なし　　（ＮＥ，Ｗ）

(5) コマンドの使用例

```
CALL  CNTNCP(CNCB)
                ①
CNCB  ZERO  ECB,  0
      ②
      ZERO  WORK,  20
                        ③    ④
ECB   BSS   4
EVEN
WORK  BSS   40
```

① ＣＮＣＢへのポインター

② ecb 領域へのポインター

③ ＮＣＰ作業領域番地へのポインター

④ ＮＣＰ作業領域サイズ

B．NOPENコマンド

(1) 機　　能

コネクションの開設を要求する。

(2) 書き方

CALL　NOPEN（CCB番地）

(3) パラメータ

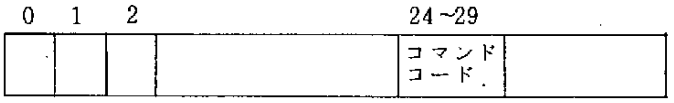
CCB番地：確立すべきコネクションに関する情報が入っているCCB領域（connection control block）の先頭番地。

(4) コマンドの終了

このコマンドの終了はNWAIT，SWAITコマンドで待たなければならない。

コマンドが終了したときにはecb領域1に終了情報が入っている。

＜ecb領域1＞




コマンドコード：$2_8$

0＝1：終了情報がPOSTされた

1＝1：異常終了，
　＝0：正常終了，このとき18ビット目on

2＝1：2重OPEN　　（DN，F）

3＝1：NCP　busy　　（NE，W）

4＝1：相手プロセスよりcloseを受け取った　（NE，W）

5＝1：規定時間内にopenが完了しない　　（NE，W）

6＝1：相手HOST　down　　（NE，W）

7＝1：IMP　down　　（NE，W）

8＝1：送信長さが9の倍数でない，送信モード8のとき　（DN，F）

9＝1：送信長さが長すぎる　（DN，F）

10＝1：受信長さが長すぎる　（DN，F）

11＝1：2重定義　（NE，W）

12＝1：相手プロセスと入力長さの対応がとれない。

13＝1：listen，lisany後のopenが次のいづれかの理由で失敗した（NE，W）

① 他系 close

② タイムアウト

14＝1：オープンキイエラー　（NE，W）

18＝1：OPENが正常終了するとonにされる　（NE）

20＝1：受信転送モードが8，9でない　（DN，F）

21＝1：送信転送モードが8，9でない　（DN，F）

22＝1：スループットが0～4でない　（DN，F）

23＝1：このコネクションは，クローズが終了してない　（DN，F）

35＝1：CNTNCPが出されていない　（DN，F）

C．NREADコマンド

(1) 機　能

コネクションを介して伝送されてきたメッセージを読み込む。このコマンドを出す前にコネクションはopenが完了していなければならないし，前にNREADコマンドを出していればそれは終了していなければならない。

(2) 書き方

CALL　NREAD（MRCB番地）

(3) パラメータ

(イ) MRCB番地：NCPに渡すべきパラメータがセットしてある領域の先頭番地

| | | |
|---|---|---|
| ＜MRCB番地＞ | ＜CCB番地＞ | MBZ |
| ＋1 | ＜領域番地＞ | ＜受信長さ＞ |

(ロ) CCB番地：読み込むべきコネクションのCCBの先頭番地

(ハ) 領域番地：メッセージを読み込むべき領域の先頭番地

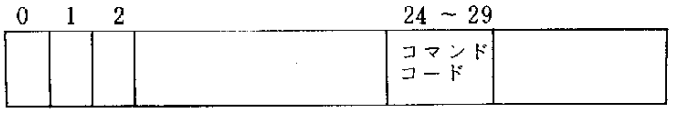
(ニ) 受信長さ：受信長さをバイト数で指定する

(4) コマンドの終了

このコマンドの終了はNWAIT，SWAITで待たなければならない。

終了するとecb領域2に終了情報がpostされる。

＜ecb領域2＞




コマンドコード：$3_8$

0＝1：終了情報がPOSTされた

1＝1：異常終了

＝0：正常終了，＜ecb領域2＞＋1番地の下位半語に実読み込みバイト数が入っている

2 = 1 : コネクションがOPENされてない （DN, F）

3 = 1 : 受信長異常 （DN, F）

4 = 1 : 受信長不整合 （NE, W）

5 = 1 : 相手close要求中 （NE, W）

6 = 1 : 相手HOST down （NE, W）

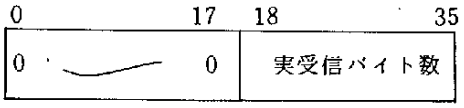
7 = 1 : IMP down （NE, W）

8 = 1 : time out （NE, W）

21 = 1 : 直前のNREADが終了してない （DN, F）

35 = 1 : CNTNCPコマンドが出されてない （DN, F）

＜ecb 領域 2＞+ 1




(5) コマンドの使用例

```
        CALL  NREAD(MRCB)
MRCB    ZERO  CCB
        ZERO  INBUF,1000
INBUF   BSS   250
```

D. NWRITEコマンド

(1) 機　能

コネクションを介してメッセージを伝送する，このコマンドを出す前にコネクションはopenが完了していなければならないし，前にNWRITEコマンドを出していればそれは終了していなければならない。

(2) 書き方

CALL NWRITE（MWCB番地）

(3) パラメータ

(イ) MWCB番地：NCPに渡すべきパラメータがセットしてある領域の先頭番地

| | | |
|---|---|---|
| ＜MWCB番地＞ | ＜CCB番地＞ | MBZ |
| ＋1 | ＜領域番地＞ | ＜送信長さ＞ |

(ロ) CCB番地：書き出すべきコネクションのCCB先頭番地

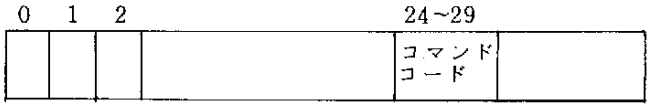
(ハ) 領域番地：メッセージを書き出すべき領域の先頭番地

(ニ) 送信長さ：送信長をバイト数で指定する。但し転送モードが8のときは9の倍数でなければならない。

(4) コマンドの終了

このコマンドの終了はNWAIT，SWAITで待たなければならない。終了するとecb領域3に終了情報がPOSTされる。

＜ecb領域3＞




コマンドコード：$4_8$

0－1：終了情報がPOSTされた

1＝1：異常終了
　＝0：正常終了

2＝1：コネクションがOPENされてない　（DN，F）

3＝1：送信長が長すぎる　（DN，F）

4＝1：送信長が9の倍数でない（送信モードが8のとき）（DN，F）

5＝1：相手側close要求中　（NE，W）

6＝1：相手HOST down　（NE，W）

7＝1：IMP down　（NE，W）

21＝1：直前のNWRITEが終了していない　（DN，F）

35－1：CNTNCPコマンドが出されてない　（DN，F）

E．NCLOSEコマンド

(1) 機　能

コネクションを切断する。このコマンドを出す前に該当コネクションに関するNREAD，NWRITEコマンドは終了していなければならない。

(2) 書き方

CALL NCLOSE（CCB番地）

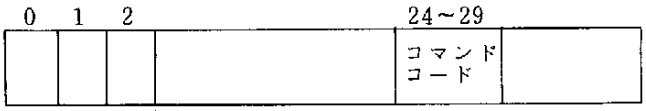
(3) パラメータ

ＣＣＢ番地：切断すべきコネクションを定義したＣＣＢの先頭番地

(4) コマンドの終了

このコマンドはNWAIT，SWAIT で終了を待たねばならない。終了するとecb 領域 1 に終了情報が入っている。

＜ecb 領域 1＞




コマンドコード：$5_8$

0＝1：終了情報が通知された，このとき19ビット目 on

1＝1：異常終了
　＝0：正常終了

2＝1：2重 close　（ＤＮ，Ｆ）

3＝1：NREAD，NWRITEが終了してない　（ＤＮ，Ｆ）

5＝1：タイムアウト，相手プロセスが規定時間内にcloseを出さなかった　（ＮＥ，Ｗ）

8＝1：他系からのmessageがまだあった　（ＮＥ，Ｗ）

19＝1：close 終了　（ＮＥ）

21＝1：open が完了してないコネクションにcloseを出した　（ＤＮ，Ｆ）

35＝1：CNTNCPコマンドが出されてない　（ＤＮ，Ｆ）

上記いずれの場合も，open されているコネクションは close される。

F．ＬＩＳＴＥＮコマンド

(1) 機　能

自分の port 名を宣言し，そのポート名を対象にしたコネクション確立要求があった場合に，相手側のポート名を通知してもらう。

(2) 書き方

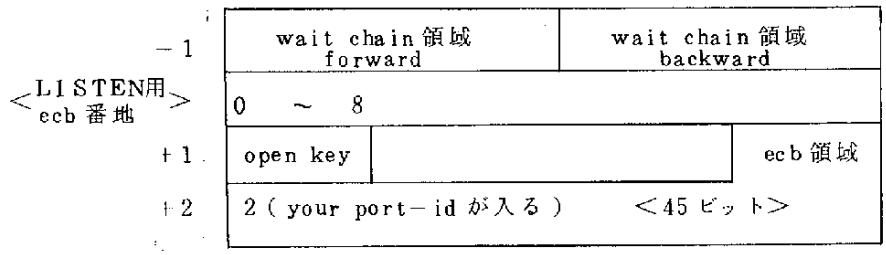
ＣＡＬＬ　ＬＩＳＴＥＮ（ＬＣＢ番地）

(3) パラメータ

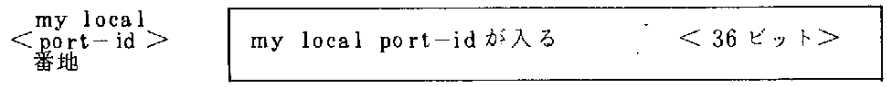
(イ) ＬＣＢ番地：ＮＣＰに渡すべきパラメータがセットしてある領域の先頭番地

| ＜ＬＣＢ番地＞ | ＬＩＳＴＥＮ用ecb番地 | Ｍ　Ｂ　Ｚ |
|---|---|---|
| ＋1 | my local port-id　番地 | Ｍ　Ｂ　Ｚ |

(ロ) ＬＩＳＴＥＮ用ecb番地：listenコマンドの通知情報をセットすべき領域の番地

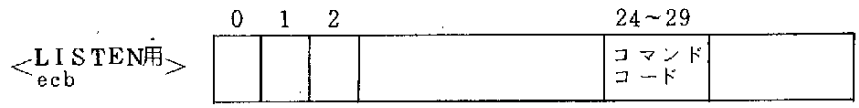



(ｲ) my local port－id 番地：my local port－id が入っている領域の番地




(4) コマンドの終了

このコマンドの終了はNWAIT, SWAITで待たなければならない。終了するとecb 領域に終了情報が postされる。




コマンドコード：$6_8$

0＝1：終了情報が postされた

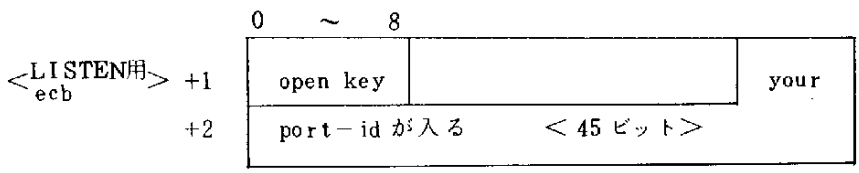
1＝1：異常終了
　＝0：正常終了，ecb領域＋1，＋2に your port－idがセットされている

2＝1：my local port－id異常　　（DN, F）

3＝1：NCP busy　　（NE, W）

20＝1：このecbは使われている　　（DN, F）

35＝1：CNTNCP コマンドが出されてない　　（DN, F）




open key：LISTENが正常にとけたときにセットされ，LISTEN後の open時に，CCB領域に your port－idとともにセットしておかねばならない。

G. LISANY コマンド

(1) 機　能

ポート識別番号を指定してそのポート識別番号を持つポート名に対してコネクション確立要求があった場合に相手側ポート名を通知してもらう。

(2) 書き方

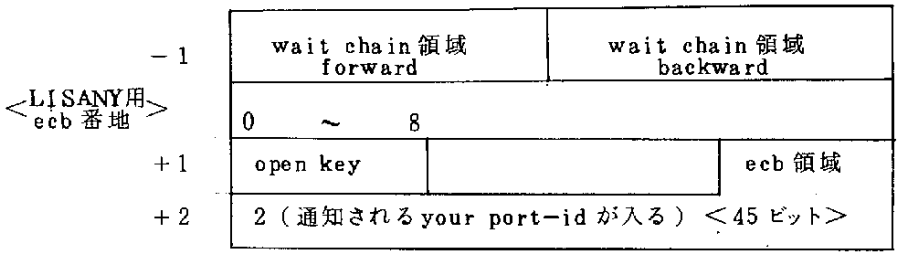
CALL LISANY(LACB番地)

(3) パラメータ

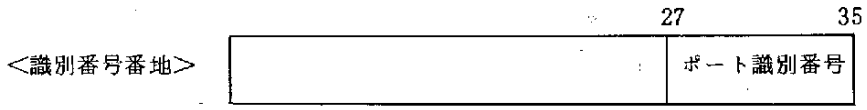
(イ) LACB番地：NCPに渡すべきパラメータがセットしてある領域の先頭番地

| | |
|---|---|
| LISANY用ecb番地 | M B Z |
| 識別番号番地 | M B Z |

(ロ) LISANY用ecb番地：LISANYコマンドの通知情報をセットすべき領域の番地

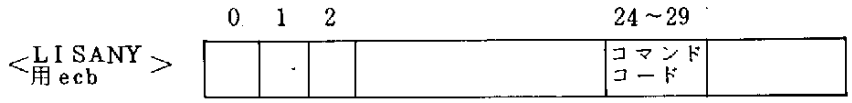



(ハ) 識別番号番地：ポート識別番号が入っている領域の番地




(4) コマンドの終了

このコマンドの終了はNWAIT，SWAITで待たなければならない。終了するとecb領域に終了情報がpostされる。




コマンドコード：$7_8$

0＝1：終了情報がpostされた

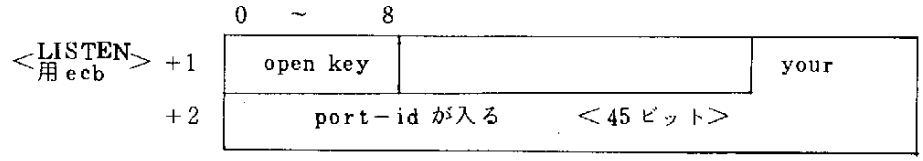
1＝1：異常終了
　＝0：正常終了，ecb領域＋1，＋2にyour port−idがセットされている。

2＝1：ポート識別番号異常　（DN，F）

3＝1：NCP busy　（NE，W）

20＝1：このecbは使われている　（DN，F）

35＝1：CNTNCPコマンドが出されてない　（DN，F）




open key： ＬＩＳＡＮＹが正常にとけたときにセットされ，ＬＩＳＡＮＹ後の open 時に，ＣＢ領域に your port－id とともにセットしておかねばならない。

Ｈ．ＲＥＦＵＳＥコマンド

(1) 機　　能

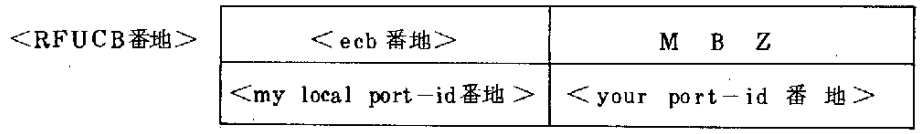
自分側および相手側のポート名を指定してＬＩＳＴＥＮ，ＬＩＳＡＮＹで通知された port－id に対するコネクション確立要求を拒否する。

(2) 書き方

ＣＡＬＬ　ＲＥＦＵＳＥ（ＲＦＵＣＢ番地）

(3) パラメータ

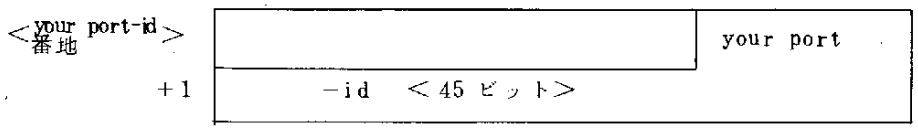
(イ) ＲＦＵＣＢ番地：ＮＣＰに対するパラメータがセットしてある番地




(ロ) ecb 番地：REFUSEコマンドの終了情報が post される番地


(ハ) my local port－id 番地 ：my local port－id が入っている領域の番地


(ニ) your port－id 番地：your port－id が入っている領域の番地




(4) コマンドの終了

このコマンドの終了はＮＷＡＩＴ，ＳＷＡＩＴで待たなければならない。終了すると ecb 領域に終了情報が post される。


コマンドコード：$10_8$

0＝1：終了情報がpostされた

1＝1：異常終了
　＝0：正常終了

20＝1：ecb 2重使用　　（DN，F）

35＝1：CNTNCPコマンドが出されてない　（DN，F）

I．NPOSTコマンド

(1) 機　能

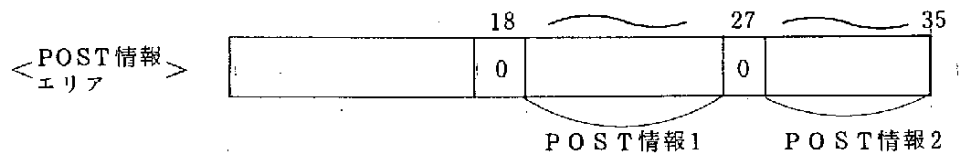
CCB中のPOST情報エリアの内容を相手側に通知する。なお該当CCBで定義されているコネクションはopenされていなければならない。

(2) 書き方

CALL　NPOST（CCB番地）

(3) パラメータ

CCB番地：通知すべき情報エリアがあるCCBの先頭番地であり，該当領域はCCB＋12番地にあり次の形をしている。




(4) コマンドの終了

このコマンドの終了をNWAIT，SWAITで待つ必要はない。ただしNPOSTが異常終了した場合は$77_8$番地に終了情報が入っている。

0，1，20，35ビットON：コネクションがopenされてない　（DN，F）

J．SETECBコマンド

(1) 機　能

CCBの被状態情報通知エリアに他系からのpostコマンドによる通知情報を読み込む。

該当CCBで定義されているコネクションはopenされていなければならない。

(2) 書き方

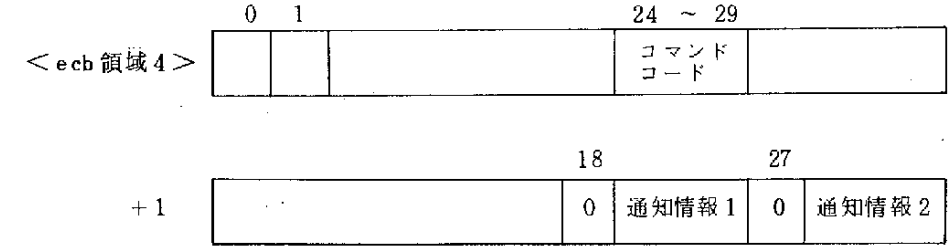
CALL　SETECB（CCB番地）

(3) パラメータ

ＣＣＢ番地：postされるべきコネクションのＣＣＢ先頭番地

(4) コマンドの終了

このコマンドの終了はNWAIT，SWAITで待たなければならない。終了するとＣＣＢのecb領域4に情報がpostされる。




コマンドコード：$11_8$

0＝1：終了情報がpostされた

1＝1：異常終了
　＝0：正常終了

2＝1：自分あるいは相手よりのNCLOSEコマンドが発せられた　（ＮＥ，Ｗ）

20＝1：コネクション not open　　（ＤＮ，Ｆ）

22＝1：直前のSETECBが終了してない　（ＤＮ，Ｆ）

35＝1：CNTNCPコマンドが出されてない　（ＤＮ，Ｆ）

K．NWAITコマンド

(1) 機　能

未終了のサービスコマンドが全て終了するまで，アクティビティの実行権を放棄する。未終了のコマンドがなければ直ちに制御をもどす。

(2) 書き方

ＣＡＬＬ　ＮＷＡＩＴ

(3) パラメータ

ナシ

(4) コマンドの終了

終了待ちになっている全コマンドが終了すると制御をアクティビティへ戻す。

ただしCNTNCPコマンドを出さずにこのコマンドを出すと，$77_8$ 番地の0，1，35ビット目をONにしてアボートする。

L．SWAITコマンド

(1) 機　能

未終了のサービスコマンドが1つでも終了するまで，アクティビティの実行権を放棄する。

終了待ちのサービスコマンドが無い場合は直ちに制御をアクティビティへ戻す。

(2) 書き方

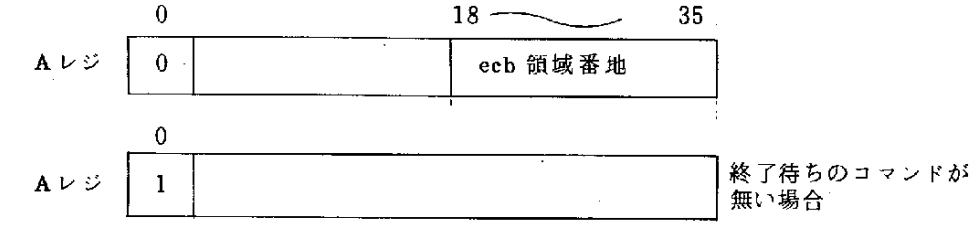
CALL SWAIT

(3) パラメータ

ナシ

(4) コマンドの終了

終了待ちになっているコマンドが終了するとAレジスタに，そのecb領域番地をセットして制御をアクティビティへ戻す。




インディケータレジスタは上記内容をAレジにロードした状態を示す。ただしCNTNCPコマンドを出さずにこのコマンドを出すと，$77_8$番地の0，1，35ビット目をONにしてアボートする。

M. SETIOSコマンド

(1) 機　能

ACOS個有の入出力動作の終了をネットワークサービスコマンドのNWAIT，SWAITで待つことを可能にする。

ACOS個有の入出力(GEINOS)のstatus情報語名をパラメータとしてcallすると，status情報語を含んだecbがNCPDISのwait control blockに登録される。

(2) 書き方

CALL SETIOS( status情報語名 )

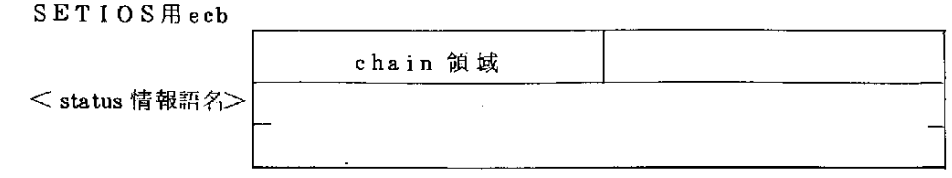
←――― return

(3) パラメータ

status情報語名：終了をSWAIT，NWAITで待ちたい，ACOS個有のI／Oのstatus情報語につけた名前。

ただし，status情報語はSETIOS用ecbに含まれていなければならない。

SETIOS用ecb



status情報語1の24 ～ 29が0ならばSETIOSでセットしたACOSの入出力コマンドである。

(4) コマンドの終了

このコマンドでecbをセットした入出力動作の終了は，NWAIT，SWAIT で待たねばならない。

終了するとstatus情報語名以下2語に入出力動作の終了情報が入っている。終了したときにstatus情報語名領域の24 ～ 29 ビット（ネットワークコマンドのコマンドコードが入るのと同じ場所）が0ならばACOSの入出力コマンドである。

なおこのコマンドを出したときにエラーがあれば，$77_8$ 番地に情報をセットしてアボートする。

0，1，20ビットON：このecbは他で使われている　（DN，F）

0，1，35ビットON：CNTNCPコマンドが出されてない　（DN，F）

(5) コマンドの使用例

```
MME     GEINOS          } ACOS個有の入出力コマンド
  コマンド
ZERO    FCP, DCWP       }
ZERO    STAT            }

CALL    SETIOS(STAT)    } STATをwait control block へ登録

CALL    SWAIT           } 上記入出力コマンドの終了を待つ
```

```
FCP     BCI     1,00000FC
DCWP    IOTD    DATA, 320
DATA    BSS     320

        BSS     1       } SETIOS用ecb
STAT    BSS     2       }
```

N. ＥＣＢＦＲＥコマンド

(1) 機　　能

Aレジで示したecbをwait control blockよりはずす。よって該当ecbに対応したコマンドの終了をNWAIT，SWAITで待つ必要はなく，コマンドの終了管理はユーザに任される。

(2) 書き方

Aレジ ←――― upperにecbアドレス

ＣＡＬＬ　ＥＣＢＦＲＥ

←――― return

(3) パラメータ

な　し

(4) コマンドの終了

このコマンドを発すると指定されたecbをwait control blockからはずし直ちに呼ばれた次のアドレスへリターンする。

なおCNTNCPコマンドを出さずにこのコマンドを出すと$77_8$番地の０，１，３５ビット目をONにしてアボートする。

O. ＢＹＥＮＣＰコマンド

(1) 機　　能

サービスアクティビティがＮＣＰとの交信をやめることを宣言する。このコマンドを出すときは，未終了のコマンドとしてＬＩＳＴＥＮ，ＬＩＳＡＮＹ以外のものがあってはいけない。

このコマンドを出すと未終了のＬＩＳＴＥＮ，ＬＩＳＡＮＹコマンドはrejectされる。

(2) 書き方

ＣＡＬＬ　ＢＹＥＮＣＰ

(3) パラメータ

ナ　シ

(4) コマンドの終了

このコマンドの終了をＮＷＡＩＴ，ＳＷＡＩＴで待つ必要はない。なおCNTNCPコマンドを出す前にこのコマンドを出すと$77_8$番地の０，１，３５ビット目をONにしてアボートする。

### 3.4.3 マクロ

A. ＣＣＢマクロ

コネクション確立に必要な情報を含むコネクションコントロールブロック（ＣＣＢ）を作る。

(1) 書き方

| 1 | 8 | 16 |
|---|---|---|
| | CCB | サブフィールド1，サブフィールド2，……，サブフィールド14 |

(2) サブフィールドの説明

① サブフィールド1

CCB名を指示する。 '$C_1 \sim C_6$'

② サブフィールド2

自分側ローカルポート名を8進12桁で書く

③ サブフィールド3

相手側のHOST識別番号を8進3桁で書く

④ サブフィールド4

相手側ローカルポート名を8進12桁で書く

⑤ サブフィールド5

受信側転送モードを指定する

8 …… 8ビットモード（binary転送）

9 …… 9ビットモード（ASCII転送）

⑥ サブフィールド6

受信最大メッセージ長をバイト数で指定する。

0 ～ 1152

⑦ サブフィールド7

送信側転送モードを指定する。

8 …… 8ビットモード（binary転送）

9 …… 9ビットモード（ASCII転送）

⑧ サブフィールド8

送信最大メッセージ長さをバイト数で指定する。ただし転送モードが8のときは9の倍数でなければならない。

0 ～ 1152

⑨ サブフィールド9

送信メッセージのスループットを指示する，数が大きくなる程スループットは増す。

0，1，2，3，4 のうち1つ

⑩ サブフィールド10

ecb 領域 1 ( NOPEN, NCLOS用 ecb 領域 ) につける名前を指示する。

'$C_1$ ～ $C_6$'

⑪ サブフィールド 11

ecb 領域 2 ( NREAD用 ecb 領域 ) につける名前を指示する。

'$C_1$ ～ $C_6$'

⑫ サブフィールド 12

ecb 領域 3 ( NWRITE用 ecb 領域 ) につける名前を指示する。

'$C_1$ ～ $C_6$'

⑬ サブフィールド 13

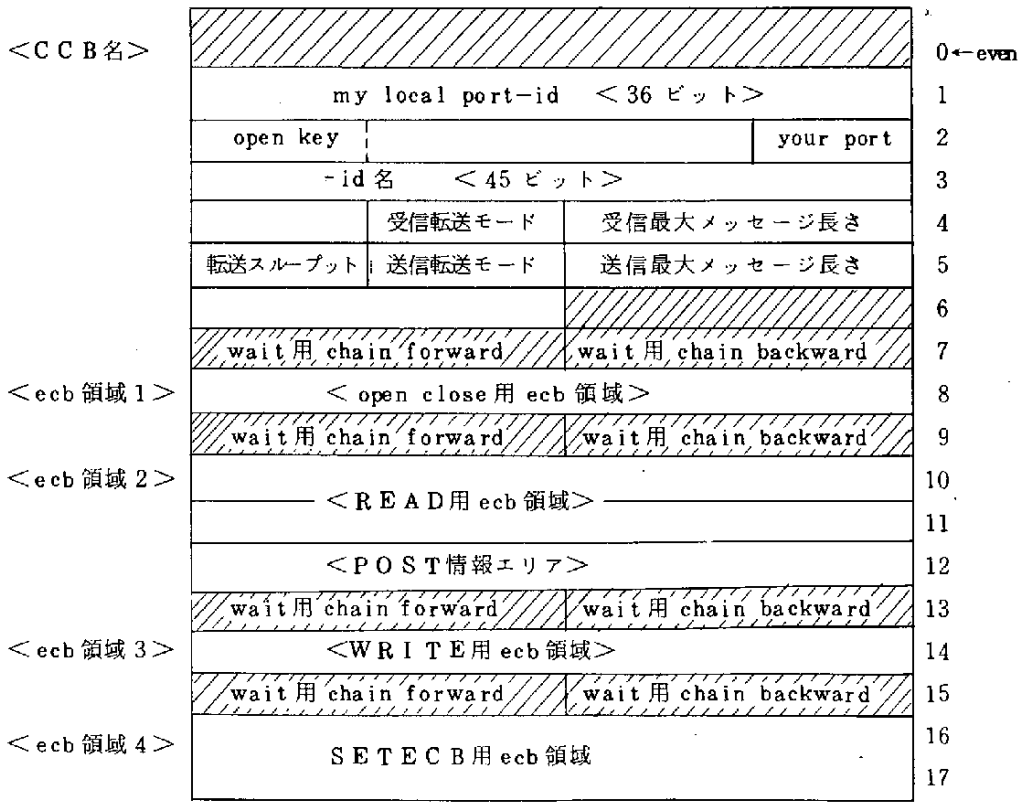
ecb 領域 4 ( SETECB用 ecb 領域 ) につける名前を指示する。

'$C_1$ ～ $C_6$'

⑭ サブフィールド 14

POST情報エリアにつける名前を指示する。

'$C_1$ ～ $C_6$'




注) system 作業領域

open key : LISTEN, LISANY後の open時にNCPより通知された open key を入れる。
この領域が0 ( 初期値も0 ) ならば通常の open とみなされる。

図 3－8 CCBレイアウト

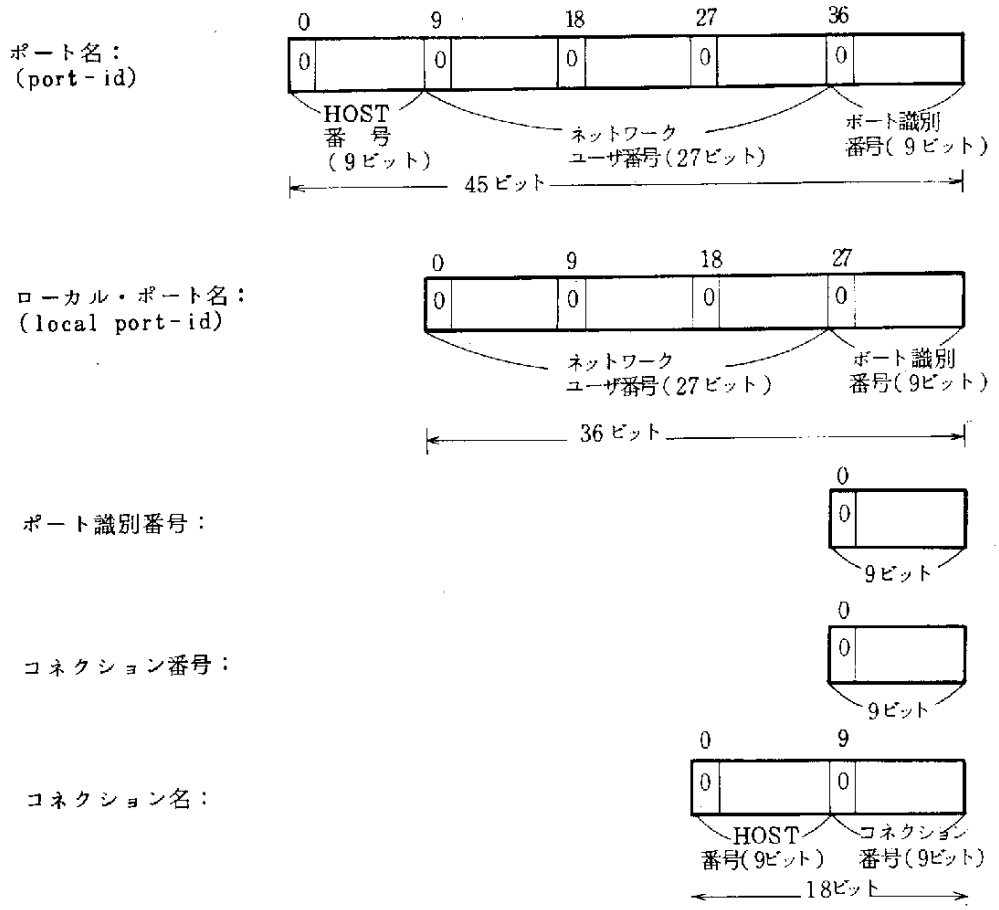



図３－９　ポート名等レイアウト

(3)　マクロ展開型

```
CCB     MACRO
        EVEN
#1      OCT     11
        OCT     #2
        OCT     #3
        OCT     #4
        ZERO    #5,#6
        VFD     9/#9,9/#7,18/#8
        BSS     12.
#10     EQU     #1+8
```

```
#11     EQU     #1+10
#12     EQU     #1+14
#13     EQU     #1+16
#14     EQU     #1+12
        ENDM    CCB
```

B. GCNCBマクロ

CNTNCPコマンドのパラメータであるCNCB，CNTNCP用ecb，NCP用作業領域を作成する。

(1) 書き方

| 1 | 8 | 16 |
|---|---|---|
| | GCNCB | サブフィールド1，……，サブフィールド3 |

(2) サブフィールドの説明

① サブフィールド1

CNCB名を指定する。　　'$C_1$ ～ $C_8$'

② サブフィールド2

CNTNCP用ecb名を指定する。　　'$C_1$ ～ $C_8$'

③ サブフィールド3

NCP作業領域サイズを指定する。　　10進2桁

(3) マクロ展開型

```
GCNCB   MACRO
        EVEN
#1      ZERO    #2          }
        ZERO    *+5,#3      }   CNCB作成
#2      BSS     4           }   ecb作成
        BSS     2*#3        }   作業領域作成
        ENDM    GCNCB
```

(4) 例

```
GCNCB  CCBMEI, ECBMEI, 5
```

上記マクロ指定により，CNCB名がCCBMEIであるCNCB，ecb名がECBMEIであるCNTNCP用ecb及び作業領域サイズが5のNCP用作業領域が確保される。

C. ＭＣＢマクロ

ＮＲＥＡＤコマンドのパラメータであるＭＲＣＢを作成する。
（ＮＷＲＩＴＥ）　　（ＭＷＣＢ）

(1) 書き方

| 1 | 8 | 16 |
|---|---|---|
| | ＭＣＢ | サブフィールド1，……，サブフィールド4 |

(2) サブフィールドの説明

① サブフィールド1

ＭＲＣＢ名を指定する。　　'$C_1$ ～ $C_6$'
（ＭＷＣＢ）

② サブフィールド2

ＣＣＢ名を指定する　　'$C_1$ ～ $C_6$'

③ サブフィールド3

受信用領域名を指定する。　　'$C_1$ ～ $C_6$'
（送信用）

④ サブフィールド4

受信用領域サイズをバイト数で指定　　10進数
（送信サイズ）

(3) マクロ展開型

```
ＭＣＢ      ＭＡＣＲＯ
＃１        ＺＥＲＯ        ＃２
            ＺＥＲＯ        ＃３，＃４
            ＥＮＤＭ        ＭＣＢ
```

(4) 例

```
ＭＣＢ      ＭＲＭＥＩ，ＣＣＢＭＥＩ，ＲＢＵＦ，４０
ＭＣＢ      ＭＷＭＥＩ，ＣＣＢＭＥＩ，ＷＢＵＦ，５０
```

D. ＧＬＣＢマクロ

ＬＩＳＴＥＮコマンドのパラメータであるＬＣＢ，ＬＩＳＴＥＮ用ecb，my port−id領域の
（ＬＩＳＡＮＹ）　　（ＬＡＣＢ）（ＬＩＳＡＮＹ）　　（port識別番号）

作成を行なう。

(1) 書き方

| 1 | 8 | 16 |
|---|---|---|
| | ＧＬＣＢ | サブフィールド1，……，サブフィールド4 |

(2) サブフィールドの説明

① サブフィールド1

ＬＣＢ名を指定する。 'C₁ ～ C₆'
（ＬＡＣＢ）

② サブフィールド2

ＬＩＳＴＥＮ用 ecb 名を指定する。 'C₁ ～ C₆'
（ＬＩＳＡＮＹ）

③ サブフィールド3

my local port-id が入る領域につける名前を指定する。 'C₁ ～ C₆'
（port 識別番号）

④ サブフィールド4

my local port-id を書く 8 進 12 桁
（port 識別番号） （8 進 3 桁）

(3) マクロ展開型

```
GLCB    MACRO
        EVEN
#1      ZERO    #2
        ZERO    #3
#3      OCT     #4
#2      BSS     3
        ENDM    GLCB
```

(4) 例

```
GLCB    LCBMEI, LISECB, MLPID, 021021021021
GLCB    LAMEI, LANECB, PORTNO, 021
```

E. NRESマクロ

NWRITE, NREADコマンドで使う入出力バッファ領域を作成する。

(1) 書き方

| 1 | 8 | 16 |
|---|---|---|
| | NRES | サブフィールド1, ……, サブフィールド3 |

(2) サブフィールドの説明

① サブフィールド1

バッファ領域の名前を指定する。 'C₁ ～ C₆'

② サブフィールド2

転送モードを指定する。 8 or 9

③ サブフィールド3

バッファサイズをバイト数で指定する。　10進数

(3) マクロ展開型

```
NRES    MACRO
        EVEN
#1      BSS     #2*#3/36+1
        EVEN
        ENDM    NRES
```

(4) 例

```
NRES    INBF, 9, 144
```

### 3.4.4 アボート表示

NCPがサービスアクティビティの致命的エラーを見つけたとき，NCPはサービスアクティビティをアボートする。その場合ほとんどのアボート動作はNCPDISによって行なうが，まれにNCPEXECによってアボートされることがある。

A. NCPDISによるアボート

NCPDISでサービスコマンドを処理中にアボートした場合，$75_{(8)}$ 番地の上位半語に最後にサービスコマンドを出した番地が入っている。

(1) コンソールに「DN」が出て，エグゼキューションレポート上に

```
*USERS  DN  MME  GEBORT  ……
```

が表示される場合は$76_{(8)}$番地にその理由が入っている。

18ビットON：NOPENコマンド処理中
19　〃　：NREAD　〃
20　〃　：NWRITE　〃
21　〃　：LISTEN　〃
22　〃　：LISANY　〃
23　〃　：NPOST　〃
24　〃　：SETECB　〃
25　〃　：NWAIT　〃
26　〃　：SWAIT　〃
27　〃　：BYENCP　〃
28　〃　：NCP作業領域オーバフロー
29　〃　：NCLOSEコマンド処理中

30ビットON： ＲＥＦＵＳＥコマンド処理中

31　〃　： ＩＤＥＮＴカードエラー

32　〃　： 作業領域のサイズが０～60でない

33　〃　： ２重ＣＮＴＮＣＰ

34　〃　： ＳＥＴＩＯＳコマンド処理中

35　〃　： ＥＣＢＦＲＥ　〃

なお，$76_{(8)}$ 番地の内容がコマンド処理中のとき，各コマンドに該当したecb領域に詳細な理由が入っている，またecb領域と同じ内容が $77_{(8)}$ 番地にも入っている。

(2) memory fault 表示がされる場合は

① サービスコマンドのパラメータで指示した領域がアクティビティの領域を越えている場合。

② ＮＣＰＤＩＳをこわした場合。

が考えられる。

B. ＮＣＰＥＸＥＣによるアボート

サービスアクティビティの動作が異常だとみなした場合，ＮＣＰＥＸＥＣはコンソールよりのＫＩＬＬコマンドをシュミレートしてサービスアクティビティを殺す。この場合サービスアクティビティのエグゼキューションレポート上にはコンソールよりのＫＩＬＬコマンドによって殺されたと同じメッセージが出力される。かかる状態が生じるのは，ほとんどの場合ＮＣＰＤＩＳが壊されていると考えられる。

### 3.4.5 ＪＯＢデック

ユーザがＮＣＰと交信する為に必要なファイルのアサイン等ＪＯＢデックの構成方を述べる。

A. ＄ＩＤＥＮＴカード

＄ＩＤＥＮＴカードにアカウント番号，識別名を必ず指定する，ただしアカウント番号・識別名はそれぞれ最大６文字である。これらはＮＣＰによるアカウント情報の作成に用いられる。

例

```
1 cal      8 cal      16 cal
↓          ↓          ↓
$          IDENT      0170, AZUMA
```

B. アセンブル時

ネットワークサービスコマンドマクロを使用する場合，＄ＧＭＡＰ文の後にマクロファイルのアサインとアセンブラにマクロをロードすることを指示する為のカードが必要となる。

(1) マクロ・ロード

```
8 cal          16 cal
↓              ↓
LODM           M・DIST
```

(2) マクロ・ファイル・アサイン

```
1 cal   8 cal        16 cal
↓       ↓            ↓
$       USERID       ACOS−NCP$A−NCP
$       PRMFL        **, R, R, ACOS−NCP/DYNAMIC−FILE
```

C. 実行時

実行時（リンク&ゴー）にNCPDISをリンクする為$EXECUTE文の後に次のファイルアサインのカードを必要とする。

```
1 cal   8 cal        16 cal
↓       ↓            ↓
$       USERID       ACOS−NCP$A−NCP
$       PRMFL        *L, R, S, ACOS−NCP/NCP−DIST
                  あるいは
$       PRMFL        *L, R, S, ACOS−NCP/NCPDIS−EXPRS
```

2番目のファイルアサイン文を使うと，実行時にfatalとなるエラーチェックを行なわないNCPDISが結合される。

## 3.5 NCPのインプリメント

本節では，NCPとサービスアクティビティの通信手順およびNCPの中心的役割をはたすNCPEXECとNCPDISに関する詳細な解説を行なう。そして最後にインプリメント上の様々な問題点およびOSに対する要望について述べる。

### 3.5.1 NCPとサービスアクティビティの通信手順

A. 通信手順解説

サービスアクティビティとNCPEXEC間の通信にはインターコムとマスターモードによるダイレクト転送を併用した。インターコムはサービスアクティビティの出したコマンド情報（制御情報）をNCPEXECへ伝える[*−1]のに用い，ダイレクト転送はNREAD，NWRITEコマ

*−1 NCPEXECは制御情報を受け取る為に，あるインターコムファイルに対して常にreadコマンドを出している。

ンドに対応するメッセージの転送および各コマンドの終了情報のecbへの通知に用いられている。

例としてサービスアクティビティがNWRITEコマンドを出した場合の手順を次に示す。

① サービスアクティビティがNWRITEコマンドを出力。

② NCPDISはコマンドパラメータよりCCB（コネクション情報が入っている領域）番地，コマンドタイプ，SNUMB（サービスアクティビティ識別番号），データ領域番地，データ長を含んだ制御情報を作りNCPEXECへインターコム経由で通知。

③ その後，CCBに含まれている送信用ecb（event control block）をwcb（wait control block）へつなぎコマンドが出された次の番地へリターン。

④ NCPEXECは②で通知された制御情報に従いサービスアクティビティ内のデータをマスターモードで自領域のバッファにコピーし，それをセグメント化してIMPへ送り出す。

⑤ サービスアクティビティがSWAITコマンドを出力。

⑥ NCPDISはwcbに終了情報が通知されるまでCPUを放棄する。

⑦ NCPEXECはIMPよりreceiveを受け取ると終了情報をマスターモードでecbへ書き込むとともにウェイト中のサービスアクティビティを起動する。

⑧ NCPDISは終了情報が通知されているecbをwcbからはずし，そのアドレスをパラメータとしてSWAITが出された次の番地へリターン。

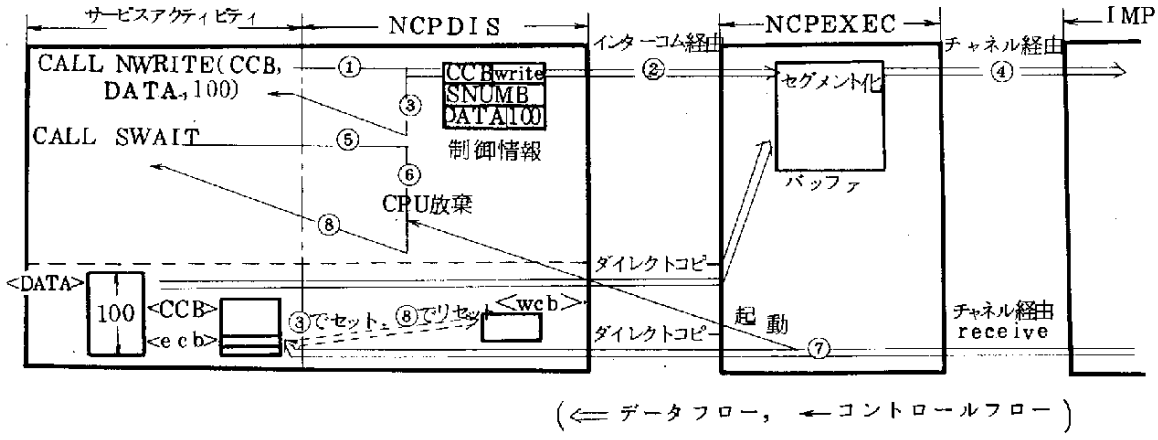



図3－10　通信手順例

B．ダイレクト転送採用理由

NCPEXECとサービスアクティビティとのメッセージ転送にダイレクト転送を行なっている。その為NCPEXECはサービスアクティビティとの同期を自分でとらなければならなくな

っている。かかるデメリットがあるにもかかわらずインターコムを全面的に採用しなかった理由として次のものがあげられる。

(1) インターコムの最大のメリットは通信しているアクティビティの同期をＯＳが管理してくれるということである。しかし１アクティビティが同時に待つことができるＩＯ数がインターコムを使う場合，最大４に制限されているので，１つのアクティビティとＮＣＰＥＸＥＣとの通信の為に１つのインターコムを使うとしても結果として同時に最大３アクティビティとしか交信できない（１つはＩＭＰとの入出力に使われる），それ以上のアクティビティと交信しようとすると同期をＮＣＰＥＸＥＣがとらざるを得ずインターコムのメリットが無い。

(2) インターコムを使うとメッセージの送受信とも最少２回のメッセージコピー（インターコムによるコピーも含めて）が必要となるが，ダイレクトに行なうと１回で済む。

(3) インターコムの終了とネットワークサービスコマンドの終了は異なる。つまりサービスアクティビティがメッセージを送信する場合を考えてみる。メッセージをインターコム経由でwriteし，そのインターコムコマンドが正常に終了しても単にメッセージがＮＣＰＥＸＥＣに渡っただけであり，相手プロセスに渡ったことを意味しない。よって本当の終了を知る為にインターコムread を常に出しておく必要があり手順が複雑になるとともに，数少ないＩＯ待ちエントリを１つ減らすこととなる。

(4) マスターモードになりさえすれば，全メモリにアクセスでき，かつ適当なＯＳモジュールを使用することによりサービスアクティビティとの同期を比較的簡単にとれる（アクティビティのロールイン，起動）。

### 3.5.2 ＮＣＰＤＩＳ解説

プロセス間コミニケーションを意図する各アクティビティにサブルーチンの形で結合され次の機能を遂行する。

① アクティビティが出したサービスコマンドから，コマンド情報を含んだ制御情報（２語～６語）を作り制御情報送信用インターコムを通じてＮＣＰＥＸＥＣへ通知する。

② ＮＷＡＩＴ，ＳＷＡＩＴコマンドを通して，終了待ちのコマンドのうち１つあるいは全てが終了するまで（ecb に終了情報が通知されるまで）ＣＰＵを放棄する。

ＮＣＰＤＩＳはＮＣＰＤＩＳ１とＮＣＰＤＩＳ２の２種類が用意されており，その相違は後者がランタイム時に致命的エラーのチェックを行なわないということであり，コアサイズ，実行時スピードの点で前者より優れているが致命的エラーでアボートした場合に詳細な理由は表示されない。

A．ＮＣＰＤＩＳの特徴

ＮＣＰＤＩＳは次のような特徴を持つ。

(1) 作業用領域，テーブルを自分の領域内にほとんど持たないので，サイズは少さく，またテー

ブルサイズが原因となる制約は全くない。

(2) NWAIT, SWAIT, CNTNCPコマンドを除いて，サービスコマンド処理中にCPUを放棄しない。

B. NCPDISの構造

NCPDISはメインレベルとCCレベルの2レベルで動作し，4つの機能モジュールから構成される。

(1) コマンド処理ルーチンA群

このルーチン群はメインレベルで動作し，CNTNCP, NOPEN, NREAD, NWRITE, NCLOSE, LISTEN, LISANY, REFUSE, NPOST, BYENCPサービスコマンド処理ルーチン群である。

この群の処理手順を示すと図3－12のようになる。ただし，CNTNCPコマンドだけはecb領域に終了情報が通知されるまでCPUを放棄している。

図上，NCP用作業領域とはCNTNCPコマンドで与えられたコマンド情報キューイング用領域であり，wcbは各コマンドに対応したecb領域をチェンニングしておくwait control blockである。

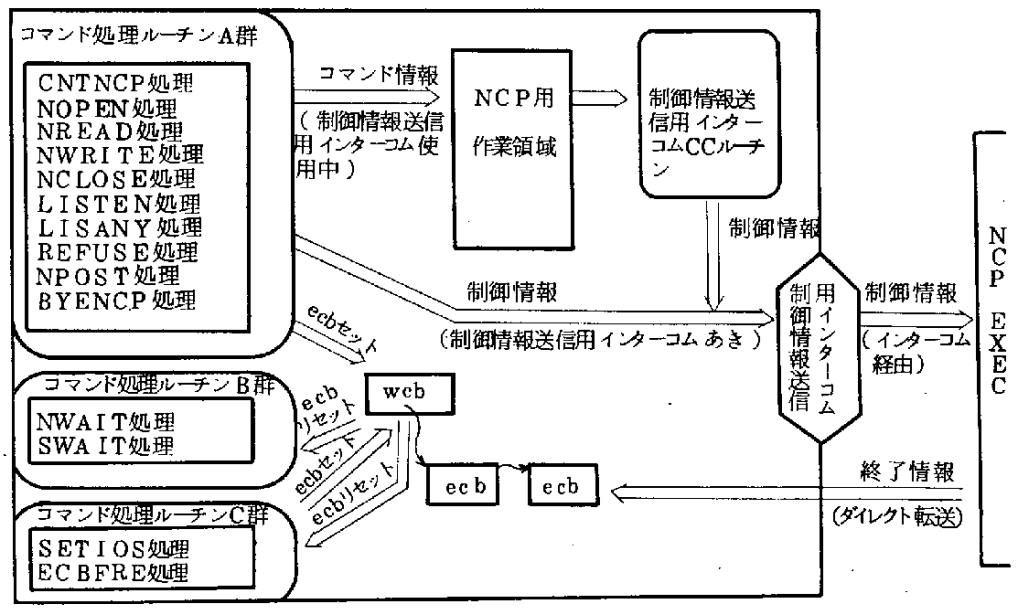



図3－11 NCPDISの構造

（ecb，NCP用作業領域は実際にはNCPDIS内にはない）

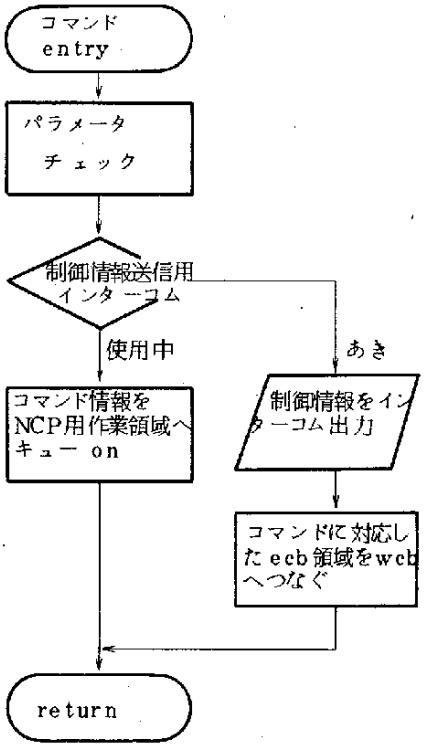



図３－12　コマンド処理ルーチンA群ブロックチャート

(2)　コマンド処理ルーチンB群

このルーチン群はメインレベルで動作し，NWAIT，SWAITサービスコマンド処理ルーチン群である。

この群の処理手順を示すと図3－13のようになる。

図上，CPU放棄は10分を指定したMME　GEWAKE（ACOS－6が提供しているコマンドで，指定された時間がつきるか，未処理のIOが終るか，他のプロセスから起動されるまでCPUを放棄する）により行なっている。

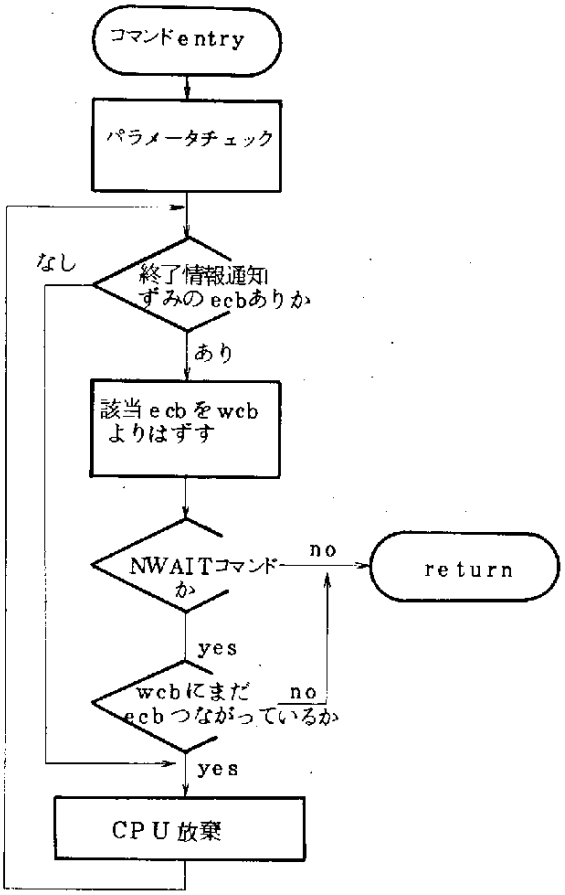



図３－13　コマンド処理ルーチンB群ブロックチャート

(3)　コマンド処理ルーチンC群

このルーチン群はメインレベルで動作し，SETIOS，ECBFRE サービスコマンド処理ルーチン群であり，それぞれ指定されたecbをwcbにセットおよびリセットする。

(4)　制御情報送信用インターコムCCルーチン

このルーチンはコーテシィコールレベルで動作し，制御情報送信用インターコム出力終了とともに制御が渡される。

このルーチンの処理手順を示すと図３－14のようになる。

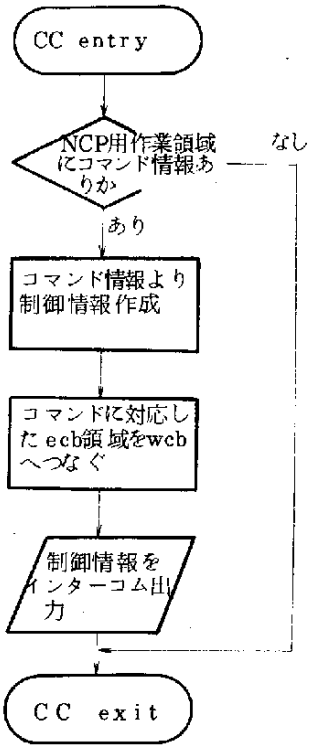



図３－14　制御情報送信用インターコムＣＣルーチンブロックチャート

### 3.5.3　ＮＣＰＥＸＥＣ解説

コア優先度が高くスワップアウトされることがまれな特権スレーブアクティビティでありＮＣＰの中枢機能をはたす。機能としては次のものがあげられる。

①　受け取った制御情報を解読してコネクションの開閉，メッセージの転送を管理する。

②　タイマー制御情報受信をトリガとして，各種時間監視を行なう。

③　サービスアクティビティの監視を行ない，必要とあればサービスアクティビティの起動，ロールイン，強制終了を行なう。

④　ＮＣＰＣＮ経由でコンソールよりタイプインされた指示コマンドを解読し，コネクト中のアクティビティのコンソールへの表示（ＮＣＰＳＵＢ経由で行なう），ＩＭＰとの初期会話，終了会話を行なう。

⑤　ＮＣＰＳＵＢのスポーン，終了を行なう。

Ａ．ＮＣＰＥＸＥＣの特徴

(1)　status 遷移とコーテシイコールの十分な活用

NCPEXECの最大の特徴であり，その動作をstatus遷移でとらえ，遷移はＩＯ（インターコムもＩＯの１つ）の終了割込みをトリガとして全てコーテシイコールレベルで動作するルーチンで行なっている。かかる構造のメリットとして次のものがあげられる。

(a) status 遷移で動作をとらえることによるメリット

① プログラムの作成に先立ち，status遷移図によって動作を十分に吟味することができ，もれが少ない。

② １遷移１モジュールというようにプログラムをモジュール化することが容易であり，以後の修正・追加が楽に行なえる。

③ status 遷移図を用いることにより，プログラムに関するドキュメント能力が格段に向上する。

(b) コーテシイコールによるメリット

① ＣＰＵの優先的使用。

② status 遷移との組み合わせでマルチタスキング的なプログラム構造が作れる。その際コーテシィコールレベルで動作するルーチンはＯＳによって互に排他制御されるので共通テーブル，シリアルリューザブルなルーチンの使用に全く注意をはらわなくてもよい。

(2) 送受メッセージのダイレクト転送

NREAD，NWRITEコマンドによるメッセージ転送の際，メッセージをサービスアクティビティの領域からNCPEXECのバッファへ（ＮＲＥＡＤは逆方向）マスターモードでダイレクトに行なっているので，コピー回数が１回ですんでいる。

(3) バッファのダイナミックアロケーション

NCPEXECではメッセージ送受に用いるバッファにコアバッファを使用しており，その割り当ては２語を単位としてダイナミックに行なっている。これにより最適なサイズのバッファが連続領域として確保できる。

(4) 既存ＯＳへの干渉が少ない。

NCPEXECの作成にあたって，ＡＣＯＳ－６ の改造はインタフェイスアダプタチャネルモジュールの追加を除いて行なわず，ＯＳ用テーブルの参照，ＯＳモジュールの使用のみを行なっている。これによりＯＳのバージョンアップに比較的容易に追従できると考えられる。

B．ＮＣＰＥＸＥＣの構造

NCPEXECの構造およびデータフローを示すと図３－15 のようになる。図上，ＮＣＰＥＸＥＣ用テーブル，ダイナミックバッファプールに対するデータフローは表示されていないが，両方とも全てのＣＣルーチンによってアクセスされる。

以下に主なルーチン，テーブル，キューについて述べる。

(1) 制御情報受信用インターコムＣＣルーチン

このルーチンはコーテシィコールレベルで動作し，制御情報受信用インターコム入力終了とともに制御が渡される。

このルーチンは(a)タイマー監視処理部分，(b)アクティビティリクエスト処理部分からなっており，それぞれ次の機能を果す。

(a)　タイマー監視処理部分

タイマー制御情報を受け取ったときに，各種タイマーを進め，必要とあればタイムアウト処理等を行なう。

ＮＣＰＥＸＥＣが監視するタイマーとして次のものがある。

①　２秒タイマー：２秒ごとにタイムアウトとなり，ロールイン要求を出したアクティビティのロールイン確認時のトリガとして用いる。

②　１分タイマー：１分ごとにタイムアウトとなり，その度にコネクト中のアクティビティの状態を調べる。

③　1時間タイマー：１時間ごとにタイムアウトとなり，コネクションの監視に用いる。

(b)　アクティビティリクエスト処理部分

ＮＣＰＤＩＳから送られてきた制御情報に従って，アクティビティの接続・接断処理，コネクションの確立・接断処理，コネクションを通じたメッセージの送受処理を行なう。

制御情報受信用インターコムＣＣルーチンはその出口で再度，自ルーチンをＣＣアドレスとしたインターコム read コマンドを出しており，完全に自律の１タスクのように動作する。

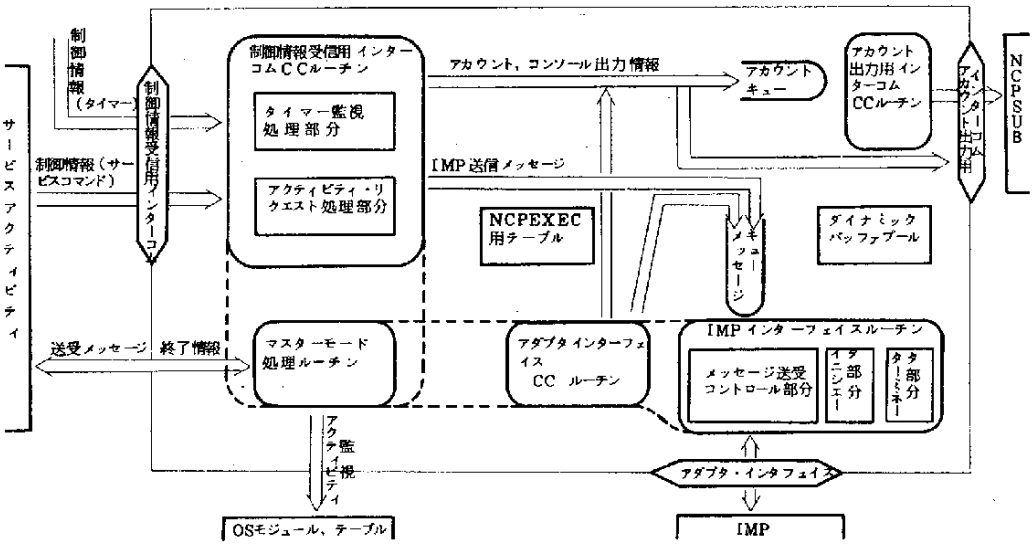



図３－15　ＮＣＰＥＸＥＣの構造

(2) ＩＭＰインターフェイスルーチン

メインレベルで動作する唯一のルーチンであり，(a)イニシエータ部分，(b)メッセージ送受コントロール部分，(c)ターミネータ部分よりなっている。以下にそれぞれの部分について説明する。

(a) イニシエータ部分

ＮＣＰＳＵＢのスポーン，制御情報受信用インターコムの最初の read 命令出力，ＩＭＰとの初期会話を行なった後，メッセージ送受コントロール部分に制御を移す。

(b) メッセージ送受コントロール部分

ＨＯＳＴ－ＩＭＰプロトコルに従ってメッセージキュー上のメッセージのＩＭＰへの送信，ＩＭＰよりのメッセージの受信を行なう。ＩＭＰとの終了会話要求によって制御をターミネータ部分に移す。

(c) ターミネータ部分

ＩＭＰと終了会話を行なった後，ＮＣＰＳＵＢに終了制御情報を送り，ターミネートする（ＮＣＰＥＸＥＣターミネーション）。

以上のべたことをフローチャートで示すと図3－16のようになる。図上，idle はアダプターに対して read 命令を出し（ＩＭＰよりのメッセージ送信要求を知る為），レリンキッシュ命令によりＣＰＵを放棄している。

(3) アダプタインターフェイスＣＣルーチン

コーテシィコールレベルで動作し，ＩＭＰインターフェスルーチンが出したアダプター入出力命令の終了時に制御が渡され次の機能を行なう。

① コントロールコマンド受信処理：ＨＯＳＴ－ＨＯＳＴプロトコルで規定された各種コマンドを受信した場合の処理を行なう。

② メッセージ受信処理：ＩＭＰより受信したメッセージをアクティビティへ転送し，receive を作る。

③ receive 受信処理：ＩＭＰより receive を受信した際，終了情報をアクティビティの ecb 領域に書き込み，アクティビティを起動する。

④ 通信相手ダウン処理：相手ＨＯＳＴダウン，ＩＭＰダウンの場合に，コネクションの接断，アクティビティへの通知を行なう。

(4) マスターモード処理ルーチン

制御情報受信用インターコムＣＣルーチン，アダプタインターフェイスＣＣルーチン等からコールされ，数々のマスターモード処理を行なう。以下にその機能を示す。

① アクティビティ，ＮＣＰＥＸＥＣ間メッセージおよび終了情報のダイレクト転送。

② アクティビティのロールイン要求。

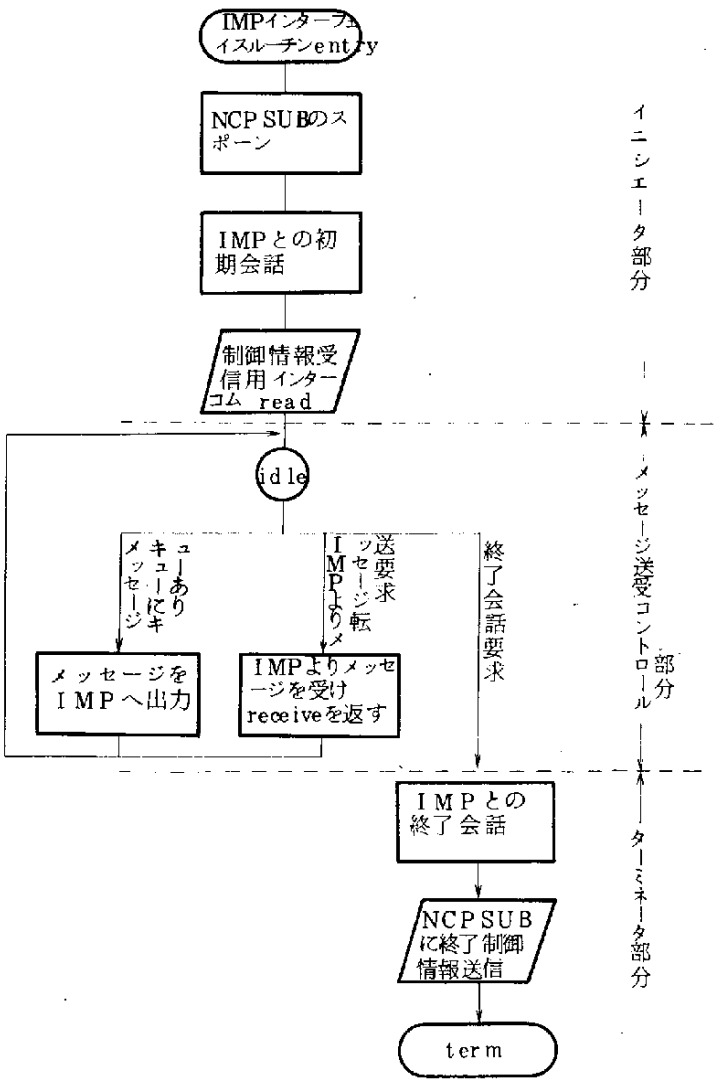



図3－16　IMPインターフェイスルーチンブロックチャート

③　アクティビティの起動。

④　アクティビティの強制終了。

(5)　アカウント出力用インターコムCCルーチン

コーテシィコールレベルで動作し，アカウント出力用インターコムコマンド終了とともに制御が渡され，アカウントキュー上にメッセージがあれば，それをNCPSUBへ送信する。

(6)　NCPEXEC用テーブル

テーブルの種類としては次のものがある。

① アクティビティテーブル：コネクト中のアクティビティ情報をセットする。

② コネクションテーブル：オープン中のコネクション情報をセットする。

③ リスンエニイテーブル：ＬＩＳＡＮＹコマンドを受けたとき，その情報をセットする。

④ リスンテーブル：ＬＩＳＴＥＮコマンドを受けたとき，その情報をセットする。

⑤ タイマーテーブル：タイマー監視に用いる。

上記テーブルのうち①～④は全てsatus情報を持ち，statusテーブルとしての役目も果している。

(7) メッセージキュー

ＩＭＰに送信すべきメッセージ，ＨＯＳＴ－ＨＯＳＴコントロールコマンドがキューイングされる。

(8) アカウントキュー

ＮＣＰＳＵＢに渡すべきアカウント情報，コンソール出力メッセージがキューイングされる。

(9) ダイナミックバッファプール

全体８ｋ語からなる連続領域で，メッセージ，ＨＯＳＴ－ＨＯＳＴコントロールコマンドをセットするバッファとして２語を単位としてダイナミックに確保，解放される。

C．ＳＴＡＴＵＳ管理

ＮＣＰＥＸＥＣがその機能を遂行する為に管理しているstatusとしては次のものがあげられる。

① アクティビティstatus

② ＬＩＳＴＥＮ status

③ ＬＩＳＡＮＹ status

④ コネクション status

⑤ ＮＰＯＳＴ status

⑥ ＳＥＴＥＣＢ status

⑦ ＮＷＲＩＴＥ status

⑧ ＮＲＥＡＤ status

⑨ Ｈ－Ｉ status

上記statusのうち，⑤～⑧は④コネクションstatusのsub statusである。

以下に各statusの説明を行なう。

(1) アクティビティ status

ＣＮＴＮＣＰコマンドでＮＣＰと結合状態にあるサービスアクティビティのstatusであり，図３－17にその遷移図を示す。図上，丸で囲まれた数字はstatusであり，それぞれ次の意味を持つ。

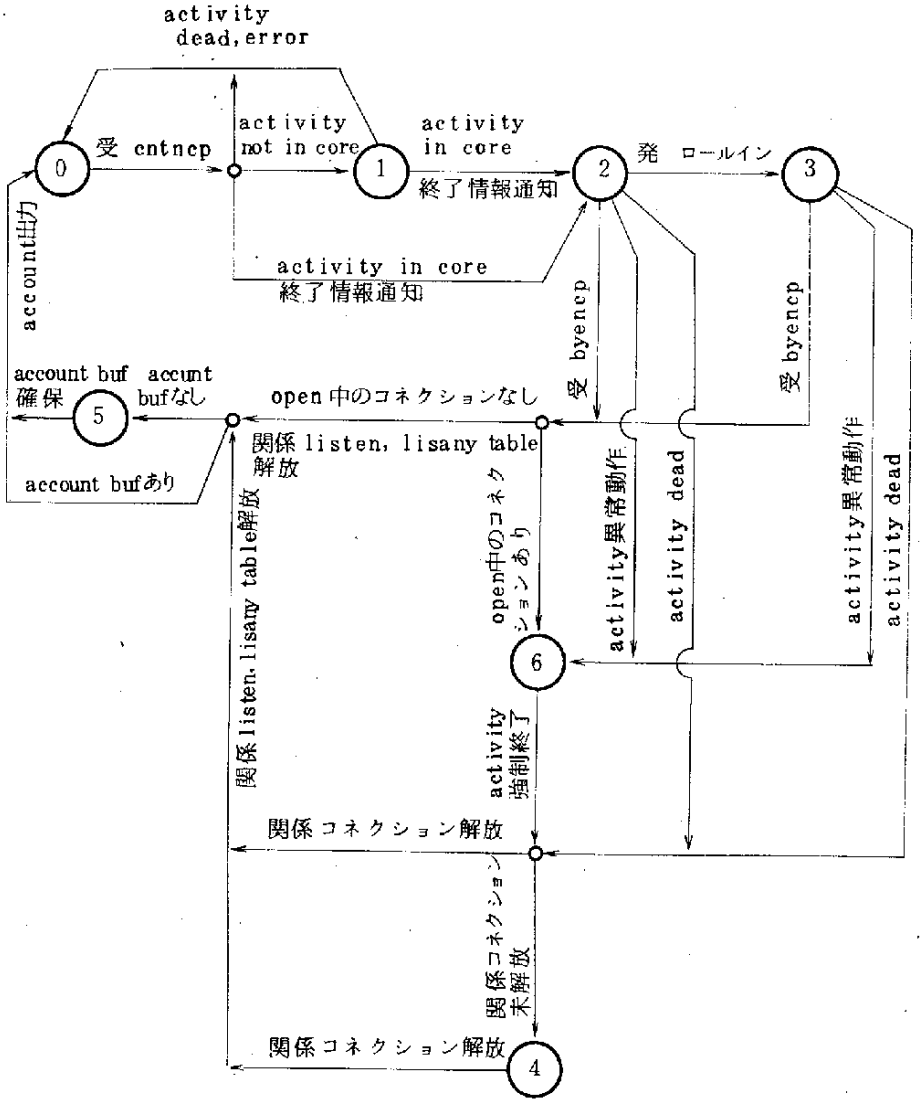



図３－17　サービスアクティビティ管理 status 遷移

0 …… ドルム

1 …… ロールイン待ち。コネクト正常終了を通知する為にアクティビティのロールインを待っている。

2 …… レディ。アクティビティは種々のサービスコマンドを使ってネットワークサービスを受けられる。

3 …… ロールイン待ち。種々のサービスコマンドの終了情報あるいは受信メッセージをアクティビティの対応した領域に書き込むため，そのロールインを待っている。

4 …… コネクション解放待ち。アクティビティとディスコネクト状態（ＢＹＥＮＣＰコ

マンドが出されたか，アクティビティがdead状態）となったときに，関係コネクションの解放が終ってない場合に，その解放を待っている。

5 …… アカウントバッファ確保待ち。アクティビティの終了情報（アカウントの為）をＮＣＰＳＵＢに伝えるのに用いるbuffer の確保待ち。

6 …… 強制終了待ち。アクティビティの動作異常をみつけたので，その強制終了されるのを待っている。

(2) ＬＩＳＴＥＮ status

ＬＩＳＴＥＮコマンドを処理する為のstatus であり，図３－18 にその遷移図を示す。図上，丸で囲まれた数字は status であり，それぞれ次の意味を持つ。

0 …… ドルム

1 …… ＲＦＣ待ち。ＬＩＳＴＥＮコマンドで宣言したmy local port－idに対応するＲＦＣが他系から送られてくるのを待っている。

2 …… ロールイン待ち。ＬＩＳＴＥＮコマンドの終了情報を通知すべきアクティビティのロールインを待っている。このとき アクティビティ status も３である。

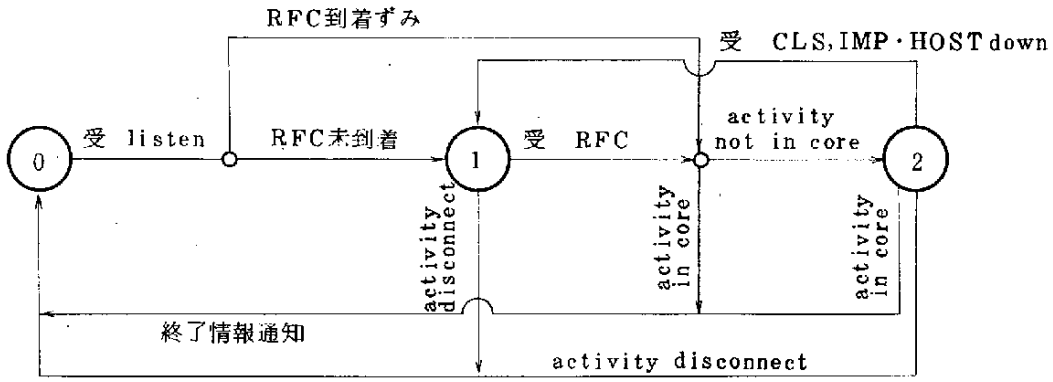



図３－18 ＬＩＳＴＥＮコマンド処理status 遷移

(3) ＬＩＳＡＮＹ status

ＬＩＳＡＮＹコマンドを処理するstatus であり，my local port－idがポート識別番号に変わる以外はＬＩＳＴＥＮと全く同じである。

(4) コネクション status

コネクション状態を管理する為のstatus であり，図３－19 にその遷移図を示す。図 上，丸で囲まれた数字は status であり，それぞれ次の意味を持つ。

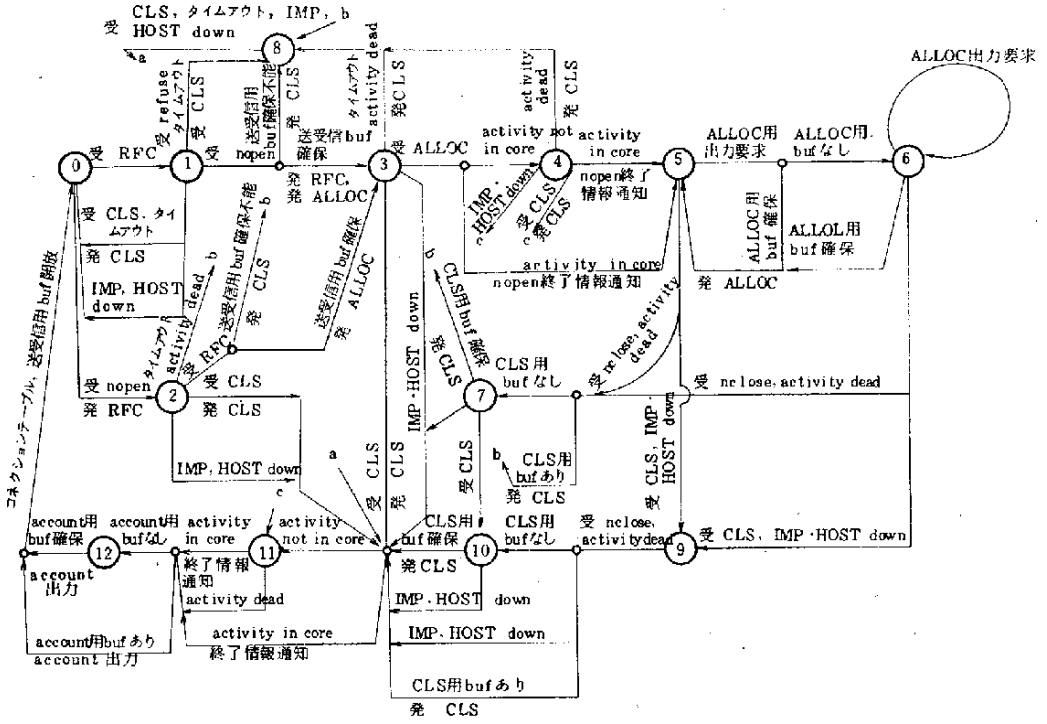



図３－19　コネクション status 遷移

0 …… ドルム

1 …… ＮＯＰＥＮ待ち。アクティビティがＮＯＰＥＮコマンドを出すのを待っている。

2 …… ＲＦＣ待ち。他系よりのＲＦＣを待っている。

3 …… ＡＬＬＯＣ待ち。他系よりのＡＬＬＯＣを待っている。

4 …… ロールイン待ち。ＮＯＰＥＮ終了情報を通知する為にアクティビティのロールインを待っている。このときアクティビティ status は３である。

5 …… レディ。

6 …… ＡＬＬＯＣ buf 確保待ち。他系にＡＬＬＯＣを発信する為の buf 確保を持っている。レディ状態の特殊なものである。

7 …… ＣＬＳ buf 確保待ち。他系にＣＬＳを発信する為の buf 確保を待っている。他系よりのＣＬＳはまだ受け取ってない。

8 …… ＣＬＳ待ち。他系からＣＬＳが送られてくるのを待っている。

9 …… ＮＣＬＯＳＥ待ち。アクティビティがＮＣＬＯＳＥコマンドを出すのを待っている。

10 …… ＣＬＳ buf 確保待ち。status 7 と同じだが他系よりのＣＬＳはすでに受け取

っている。

11 …… ロールイン待ち。コマンド終了情報（通常は対ＮＣＬＯＳＥコマンド）を通知するためアクティビティのロールインを待っている。このときアクティビティstatusは3である。

12 …… アカウントバッファ確保待ち。コネクションアカウント情報をＮＣＰＳＵＢに伝えるのに用いるbufferの確保を待っている。

コネクションテーブルが確保できなかった場合の遷移は遷移図上にあらわれていないが，そのときは他系にＣＬＳを送っているがＣＬＳコマンド作成用のbuf確保の必要が無いよう処理上考慮してある。なおコネクション確立時に必要なＨＯＳＴ－ＨＯＳＴコントロールコマンド用のバッファはコネクションテーブル確保と同時にダイナミックバッファプールより確保し，送信後プールへ返却している。

(5) ＮＰＯＳＴ status

ＮＰＯＳＴコマンドを処理する為のstatusであり，図3－20にその遷移図を示す。図上，丸で囲まれた数はstatusであり，それぞれ次の意味を持つ。

0 …… ドルム

1 …… レディ。コネクション開設中。

2 …… ＰＯＳＴ buf 確保待ち。他系にＰＯＳＴを送信する為のbuf確保を待っている。

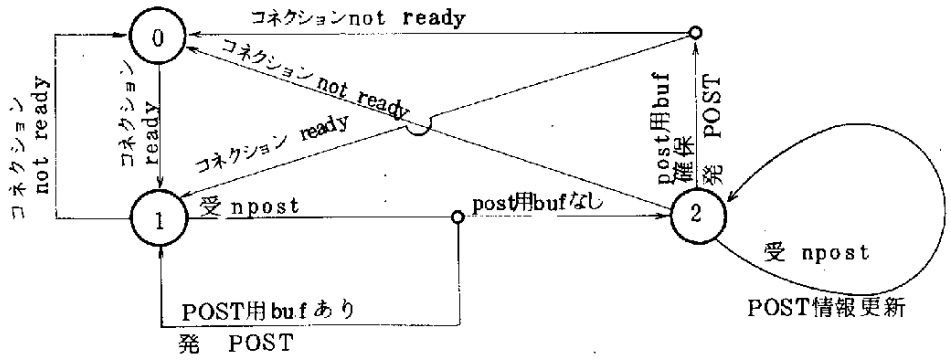



図3－20 ＮＰＯＳＴコマンド処理 status 遷移

(6) ＳＥＴＥＣＢ

ＳＥＴＥＣＢコマンドを処理する為のstatusであり，図3－21にその遷移図を示す。図上，丸で囲まれた数はstatusであり，それぞれ次の意味を持つ。

0 …… ドルム。

1 …… レディ。コネクション開設中。

2 …… ＳＥＴＥＣＢ待ち，アクティビティがＳＥＴＥＣＢコマンドを出すのを待っている。

3 …… ロールイン待ち。他系よりＰＯＳＴで通知された情報をアクティビティに通知する為，そのロールインを待っている。このときアクティビティstatusは3である。

4 …… ＰＯＳＴ待ち。他系よりＰＯＳＴによる情報通知を待っている。

5 …… ロールイン待ち。ＳＥＴＥＣＢコマンドが他系からのＣＬＳによって終了したことを通知する為アクティビティのロールインを待っている。

6 …… ＮＣＬＯＳＥ待ち。アクティビティがＮＣＬＯＳＥコマンドを出すのを待っている。

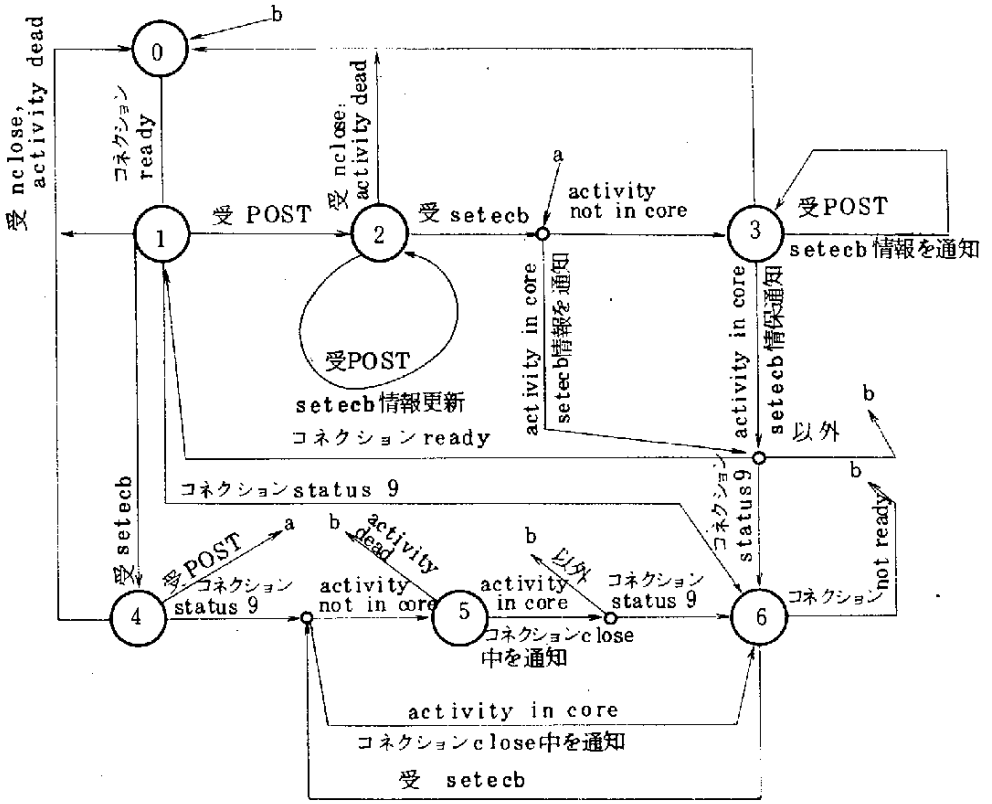



図２－21　ＳＥＴＥＣＢコマンド処理status遷移

(7) NWRITE status

NWRITEコマンドを処理する為のstatusであり，図3−22にその遷移図を示す。 図上，丸で囲まれた数はstatusであり，それぞれ次の意味を持つ。

0 …… ドルム。

1 …… レディ。コネクション開設中。

2 …… ロールイン待ち。送信メッセージをNCPEXEC内のバッファへ転送する為，アクティビティのロールインを待っている。

3 …… ALLOC待ち。メッセージを送信する為他系よりのALLOCを待っている。

4 …… REC待ち。送信メッセージに対応するreceiveが他系より送られてくるのを待っている。

5 …… ロールイン待ち。NWRITEコマンドの終了情報を通知する為，アクティビティのロールインを待っている。

6 …… ロールイン待ち。他系よりのCLSによってNWRITEコマンドが終了したことを通知する為，アクティビティのロールインを待っている。

7 …… NCLOSE待ち。他系よりCLSが到着し，アクティビティがNCLOSEコマンドを出すのを待っている。

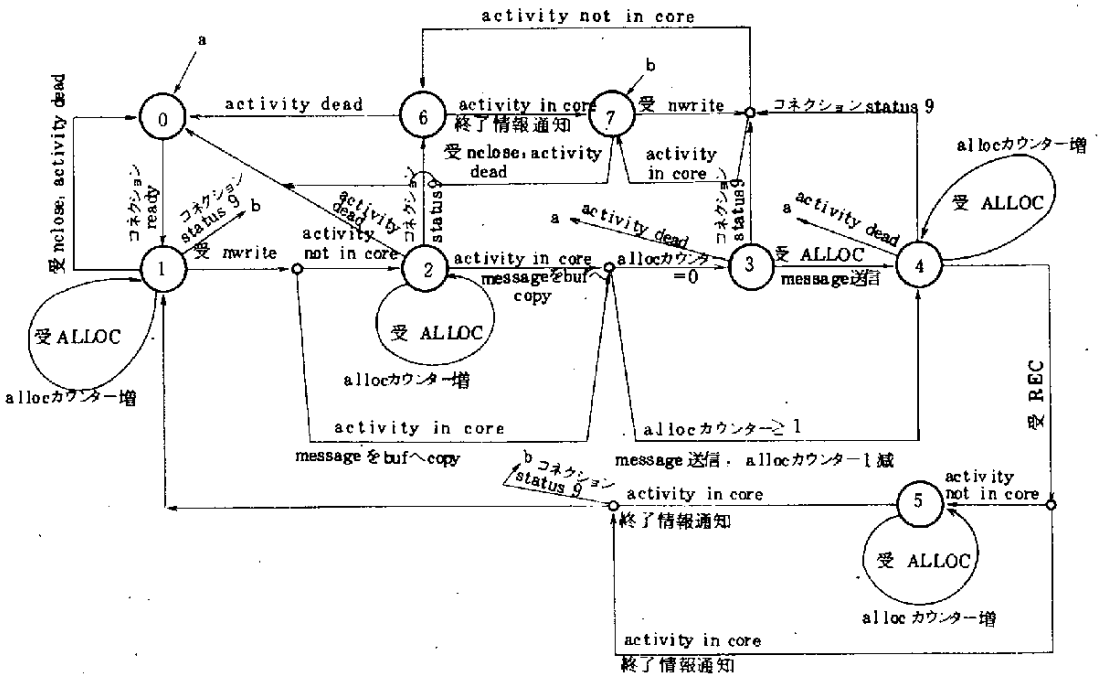



図3－22 NWRITEコマンド処理status遷移

(8) ＮＲＥＡＤ　status

ＮＲＥＡＤコマンドを処理する為のstatus遷移であり，図３－２３にその遷移を示す。　図上，丸で囲まれた数はstatusであり，それぞれ次の意味を持つ。

0 …… ドルム。

1 …… レディ。コネクション開設中。

2 …… メッセージ待ち。他系よりのメッセージ到着を待っている。

3 …… ロールイン待ち。受信メッセージをアクティビティの領域に転送する為，そのロールインを待っている。

4 …… ＮＲＥＡＤ待ち。アクティビティがＮＲＥＡＤコマンドを出すのを待っている。

5 …… ロールイン待ち。他系よりのＣＬＳによってＮＲＥＡＤコマンドが終了したことを通知する為，アクティビティのロールインを待っている。

6 …… ＮＣＬＯＳＥ待ち。他系よりＣＬＳが到着し，アクティビティがＮＣＬＯＳＥコマンドを出すのを待っている。

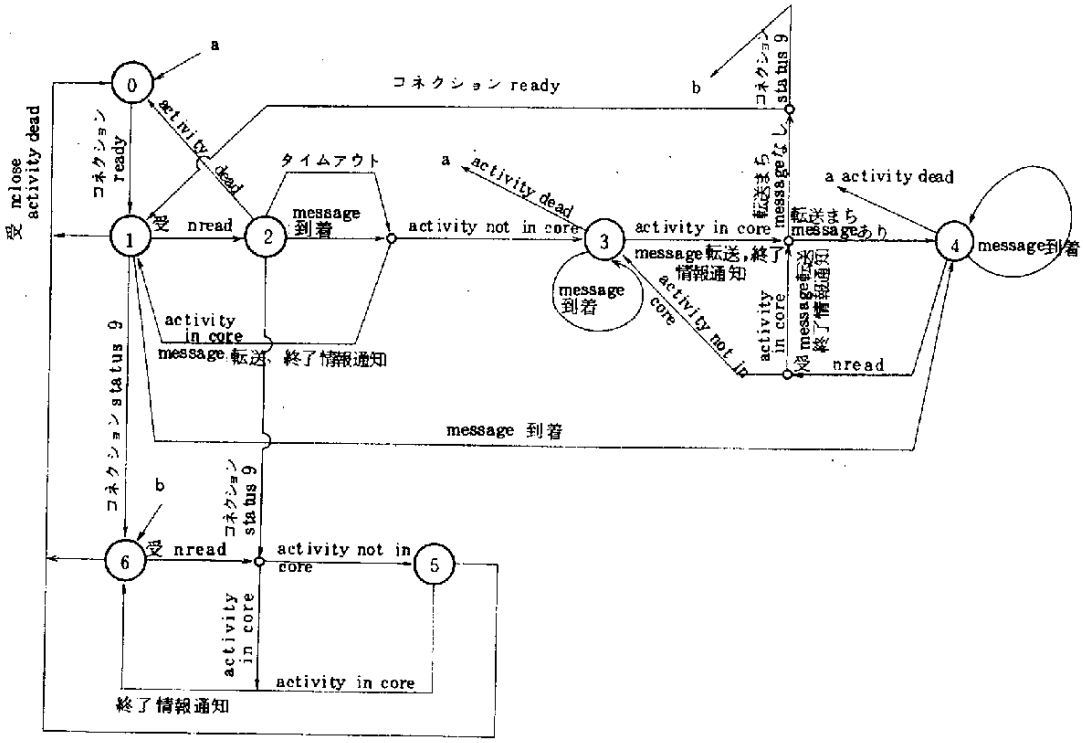



図３－２３　ＮＲＥＡＤコマンド処理status遷移

(9) H－I　status

ＩＭＰインターフェイスルーチンが，ＩＭＰと初期会話，終了会話を行なうときに用いるstatusであり，図3－24にその遷移図を示す。　図上，丸で囲まれた数はstatusであり，それぞれ次の意味を持つ。

0 …… ドルム。

1 …… 初期会話要求中。ＩＭＰインターフェイスルーチンに対して，ＩＭＰと初期会話を行うべき要求がでていることを示す。

2 …… 初期設定中。ＩＭＰよりの初期設定指令（ＨＯＳＴ－ＩＭＰプロトコル規定）を待っている。

3 …… レディ。ＩＭＰと結合状態であることを示す。

4 …… 終了会話要求中。ＩＭＰインターフェイスルーチンに対して，ＩＭＰと終了会話を行うべき要求がでていることを示す。

5 …… 終了会話中1。ＩＭＰよりの接続中止要求（ＨＯＳＴ－ＩＭＰプロトコル規定）を持っている。

6 …… 終了会話中2。ＩＭＰに対して接続中止要求（ＨＯＳＴ－ＩＭＰプロトコル規定）を出すのを待っている。

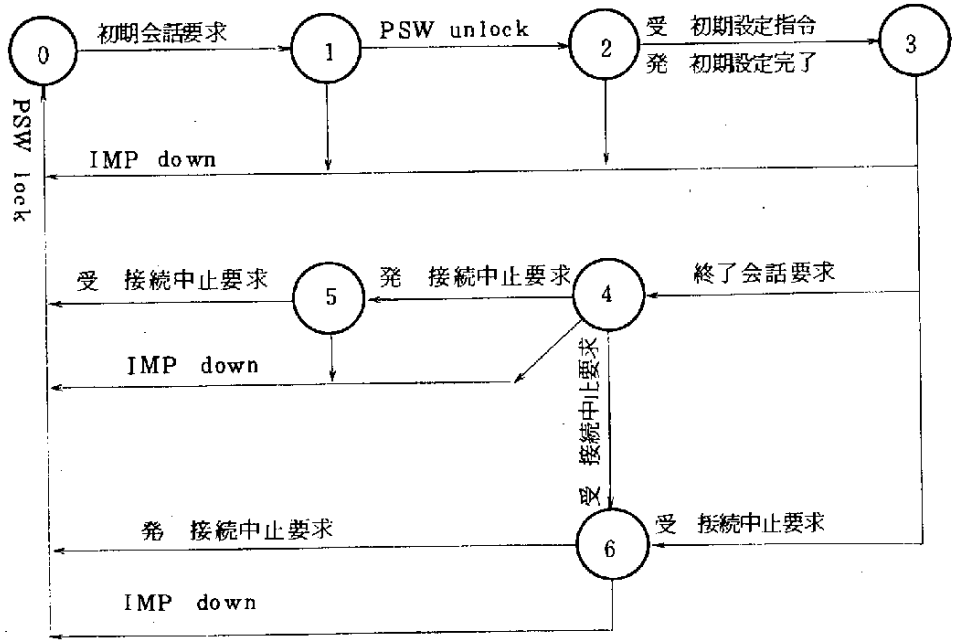



図3－24　H－I　status遷移

status 6が存在する理由は，開設中のコネクションがあった場合，直ちにＩＭＰと終了会話を行なうわけにはいかないからである。

### 3.5.4 ＯＳの検討

作成上の問題点をローカルＯＳによる面，それ以外の理由による面に分けて示し，最後にＮＣＰを作るにあたってのＯＳに対する一般的要望を示す。

Ａ．ローカルＯＳによる問題点

(1) アクティビティに割り込みを通知する機能が無い為ＮＣＰＥＸＥＣはＩＭＰよりのメッセージを受信する為に常に入力コマンドを出しておかなければならず，ＩＭＰへメッセージを送信する際は前記入力コマンドをキャンセルするコマンドを出し，キャンセルが正常に行なわれたことを確認した後に出力コマンドを出すという複雑な手順を行なっている。これによるプログラムの作りにくさ及び実行時ＣＰＵのロスは多大である。

任意時点でコンソールよりＮＣＰＥＸＥＣに対する指示を与えることを可能にする為，同じ理由でＮＣＰＣＮをアクティビティとして作らねばならなかった（コンソールに対して常に入力コマンドを出しておくわけにはいかないから）。

(2) コマンドチェンの機能が無い為，実行効率は落ちていると考えられる。

(3) バイト出力を行なう際，メッセージの最終バイトがワードの右端に入っているようにセットしておかねばならないので，サービスアクティビティのデータ領域からメッセージをＮＣＰＥＸＥＣ内のバッファに転送する手順が複雑となった。

(4) 1アクティビティが同時に待つことができるＩＯ数がインターコムを使う場合，最大4個に制限されているので設計時に苦労をした。

(5) ＡＣＯＳ－6のウェイトコマンドは各ＩＯを別々のイベントとして定義できないので，タイミングによってはウェイトコマンドとＩＯの終了がすれちがい，ウェイトが解けなくなる場合がある（ＮＣＰはこのような状態が起っても，定期的にタイマー情報を送っているのでそれによってこの状態は解ける）。

Ｂ．ローカルＯＳによらない問題点

(1) ＨＯＳＴ－ＩＭＰプロトコルの手順が複雑な為，ＮＣＰＥＸＥＣの該当処理部分が複雑となった。特にコマンド衝突部分の収束に多大の時間を費した。

(2) サービスアクティビティは必らずしも正しく動作するとは限らないという前提があるので，ＮＣＰのアクティビティ監視機能が大きくなった。

(3) ＮＣＰＥＸＥＣとサービスアクティビティ間のデータ転送をＮＣＰＥＸＥＣが自分で行なっているのでアクティビティのロールインコントロール機能をを組み込まねばならなかった。

(4) ＮＣＰＥＸＥＣはＣＣを多用しているが，メインレベルとＣＣのスイッチングにかなりＣＰ

Uを食っていることが判明した。

C．ＯＳに対する要望

ＮＣＰを本ＮＣＰのような形（１プロセスとする）で作る場合，よりユーザＪＯＢに近い形で作る方が望ましい（ＯＳのバージョンアップ等についていく為）。その場合既存ＯＳにあって欲しい機能として次のものがあげられる。

(1) 自系内のプロセ間通信機能

この機能が無いと絶対ユーザＪＯＢとしてＮＣＰを作れない。しかしＡＣＯＳ－６のように事実上同時交信できるプロセの数が著しく制限されるのでは使いがたい。またＪＩＰＮＥＴのようにＮＣＰにアクセスするプロセスの数がダイナミックに変動し，また必ずしも正しく動作するとは限らない場合，ＮＣＰとユーザプロセス間の通信の一致をとるもの（ポート，ファイル名 etc）がユニークにダイナミックに与えられ無関係の他プロセスに乱されないよう維持されなければならない。

(2) プロセスに割り込み的情報を通知する機能

新たなユーザプロセスが参加したことをＮＣＰに伝える，監視タイマーのタイムアウトを伝える，本ＮＣＰの場合はＩＭＰよりのリクエストを受ける等何らかの割り込み的情報を的確に捕える為に必要と考えられる。

(3) マルチタスキング機能

本ＮＣＰを例にとってみれば機能的に①ユーザプロセス応答部分，②ＩＭＰ応答部分，③コネクション管理部分に分かれ，これらはそれぞれ別タスクにでき，そうするとすっきりした形になると考えられる。

(4) 非可分（ indivisible ）なルーチンが作れる事。タスク間での共用テーブルアクセス，シリアルリユーザブルなルーチンを作る為に必要である。１ＣＰＵならば割込み禁止命令が使えればよい（通常ユーザジョブでは使えない）。

(5) プロセスの監視，管理ができるプロセスを作れる事

このようなプロセスは完全なユーザプロセスとは言えないかもわからない。しかし本ＮＣＰのようにユーザプロセスがかならずしも正しく動作するとは限らない場合，最低監視の機能が必要だし，スワッピング機構を持つＯＳだと，ユーザプロセスのスワップイン，スワップアウト禁止がＮＣＰからコントロールできることが望ましい。またＪＩＰＮＥＴのハイレベルプロトコルを維持するプロセスの起動はＮＣＰで行なう方が良いとも考えられる。

# 4. ＴＩＰソフトウェア

# 4. ＴＩＰソフトウェア

## 4.1 基本設計

### 4.1.1 概　要

ＪＩＰＮＥＴのＴＩＰはＡＲＰＡ−ＮＥＴのそれとほぼ同じような機能を果すものである。すなわち，メッセージ交換を行なうＩＭＰ機能に端末装置の制御機能とＨＯＳＴ機能の一部を付加したものである。

ＴＩＰソフトウェアの設計にあたっては，次の点を考慮した。

① 他系ＨＯＳＴのＴＳＳやリモートバッチサービスを単に利用するだけの機能とし，ＴＩＰの中にユーザプロセスを発生させることはせず，実際の端末装置自身をユーザプロセスとして処理する。

② それらサービスをＴＩＰサービスコマンドにより，簡単な手続きで利用できるようにする。

③ ＩＭＰ機能に全く変更を与えないようにする。

④ ＩＭＰとのインターフェイスは新たに作成するmini HOST−IMP interface タスクにおいてすべて吸収する。

⑤ ＩＭＰハードウェアの主記憶装置，32kWのうち16kWをＴＩＰプログラムで使用できる。

⑥ ＴＩＰのサービス開始，終了はconsole type writerより行なう。

さらに，ＴＩＰプログラムはＩＭＰプログラムと同居していること，使用できるプログラムエリアに制限があることなどから，以下のことに特に注意した。

① ＴＩＰプログラムのバグにより，ＩＭＰ機能に影響を与えないようにエラーチェックを厳重にするとともに，プログラムの作り方に細心の注意をはらう。

② プログラムサイズの最小化を考慮しつつプログラムを作成する。

ＴＩＰソフトウェアは実際に他系ＨＯＳＴとの交信を行なうＨＯＳＴ機能部と端末ユーザからのサービスを受けたり，サービス内容に基づき端末装置を制御する端末制御部から構成されている。図４−１にＴＩＰソフトウェアの概念図を示す。

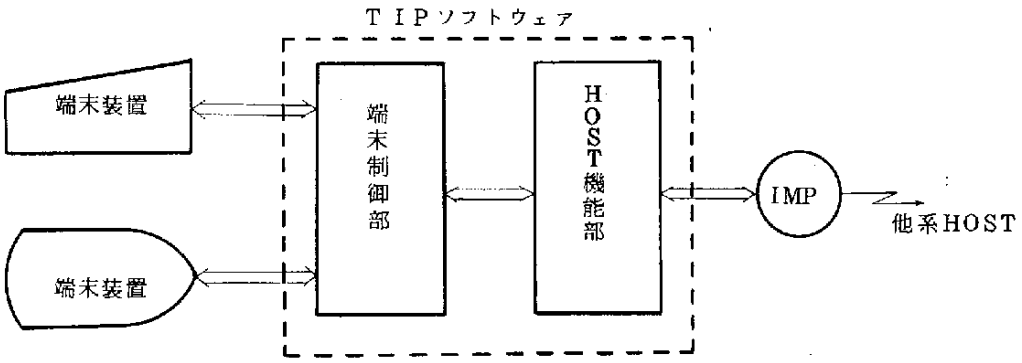



図４－１　ＴＩＰソフトウェアの概念図

### 4.1.2　多重通信制御部の構成と端末装置

次項以降のソフトウェアの説明に入る前に，ここで，ＴＩＰのハードウェア環境を明確にするため，ＴＩＰ構成の中心を占める多重通信制御部と現有の端末装置について簡単に述べる。

(1)　多重通信制御部

ＴＩＰの端末装置を制御する多重通信制御部（ＭＬＣ）の特徴は表４－１に示す通りである。また，ＪＩＰＮＥＴにおける多重通信制御部の構成を図４－２に示す。

表４－１　ＮＥＡＣ３２００／ＭＬＣの特徴

| | ＮＥＡＣ３２００／ＭＬＣ |
|---|---|
| 最大回線数 | 全二重　32回線（オプションとして128回線まで可） |
| 通信速度 | 50，200，1200，2400bps |
| 符号単位 | 5，6，8単位 |
| 同期方式 | 調歩同期式（50，200，1200bps）同期式（2400bps） |
| 接続チャネル | ＤＭＡ |
| 使用チャネル数 | 1 |
| 転送モード | メッセージモード |

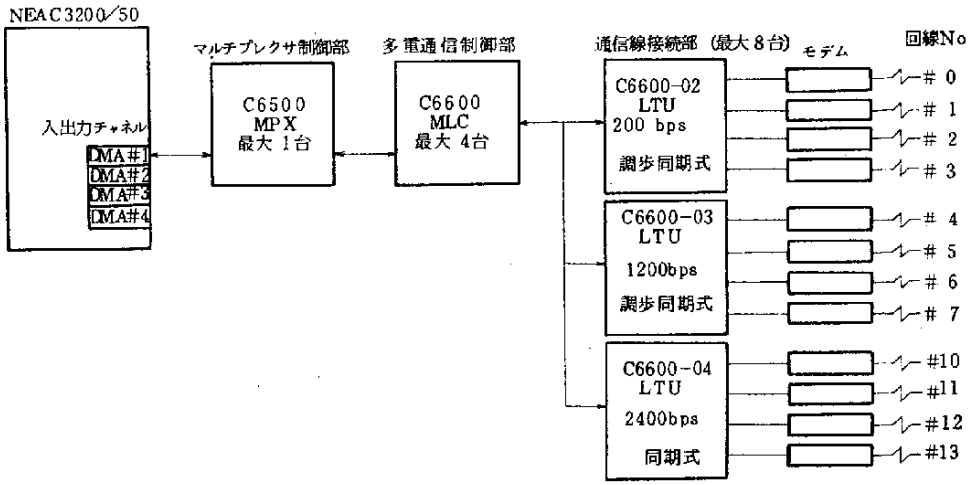



図４－２　ＪＩＰＮＥＴにおけるＭＬＣの構成

図４－２からもわかる通りＪＩＰＮＥＴにおける現在の構成では合計12端末の接続が可能である。

多重通信制御部の動作概要は図４－３に示す通りである。

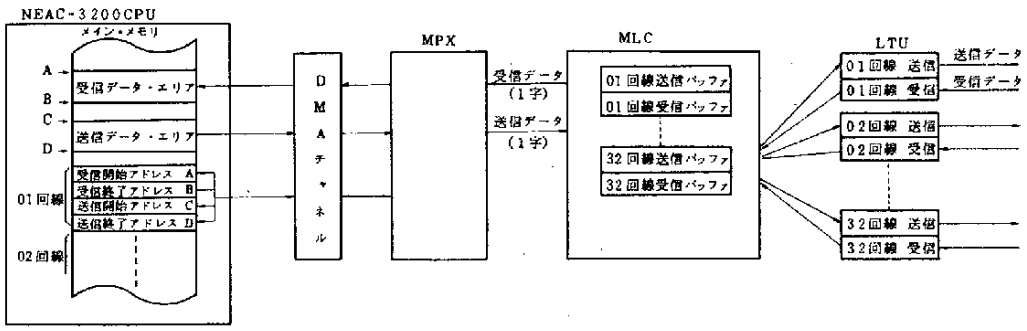



図４－３　ＭＬＣの送受信動作の概略

| | MPX | MLC | LTU |
|---|---|---|---|
| 送信動作 | ① 送信開始アドレスと送信終了アドレスが指定されると送信データエリアから1字ごとにDMAサイクルでデータを引取りMLCへ転送する。 | ② MCLは1200ビット／秒の1ビットタイムの間隔（833μs）で01回線から順に走査し受信データを1ビットごと（2400ビット／秒回線の場合2ビットごと）送信バッファからLTUへ送出する。 | ③ MLCからのビット直列のデータはLTUで送信タイミング（例えば1200ビット／秒は833μs）で回線へ送出される。 |
| 受信動作 | ④ MPXはMLCからの1字のデータをDMAサイクルで受信データエリアへ転送する。 | ② MLCは1200ビット／秒の1ビットタイム間隔（833μs）で01回線から順に走査し受信データを1ビットごと（2400ビット／秒回線の場合2ビットごと）引取りMLC内の各回線ごとの受信バッファで1字に組立てる。<br>③ 1文字に組立てられたデータは制御符号及びパリティ検出が行なわれMPXへ転送される。 | ① 回線から直列に入ってくる受信データはLTUで1ビットごとにロジック情報としてサンプリングされる。 |

(2) 端末装置

JIPNETで現在使用している端末装置の一覧表を表4－2に示す。

表4－2　JIPNETにおける端末装置一覧表

| 項目＼機種 | Termi Net 300 | Silent 700 Model 733 | Tektronix 4010 | Consul 880 | MDR 4000 |
|---|---|---|---|---|---|
| 型式 | TTY | TTY | 簡易グラフィックディスプレイ | 文字ディスプレイ | マーク・ドキュメントリーダー |
| 使用伝送速度（bps） | 200 | 1200 | 1200 | 1200 | 1200 |
| 印字速度（字／秒） | 15 | 30 | 120 | 120 | — |
| 行印字数 | 118 | 80 | 74字×35行 | 80字×24行 | — |
| 文字の種類（ASC11） | 94 | 95 | 63 | 64 | — |
| 読取り速度（枚／分） | — | — | — | — | 100 |
| 使用カードサイズ | — | — | — | — | 8.1 × 28.5 cm |
| オプション | — | カセットMT | ハードコピー | グラフィック | — |
| 使用目的 | 汎用タイプライタ端末として | 汎用タイプライタ端末又はカセットMTによる高速入出力端末として | グラフィック端末または一般端末として | 高速表示一般端末または簡単なグラフィック表示端末として | タイプライタ端末と組合せて特殊入力端末として |
| 備考 | Honeywell Bull製 | Texas Instrument製 | Sony Tektronix製 | ADDS製 | Bell & Howell製 |

### 4.1.3　ソフトウェアの構造

ＴＩＰソフトウェアは，図４－４に示すような構造になっている。

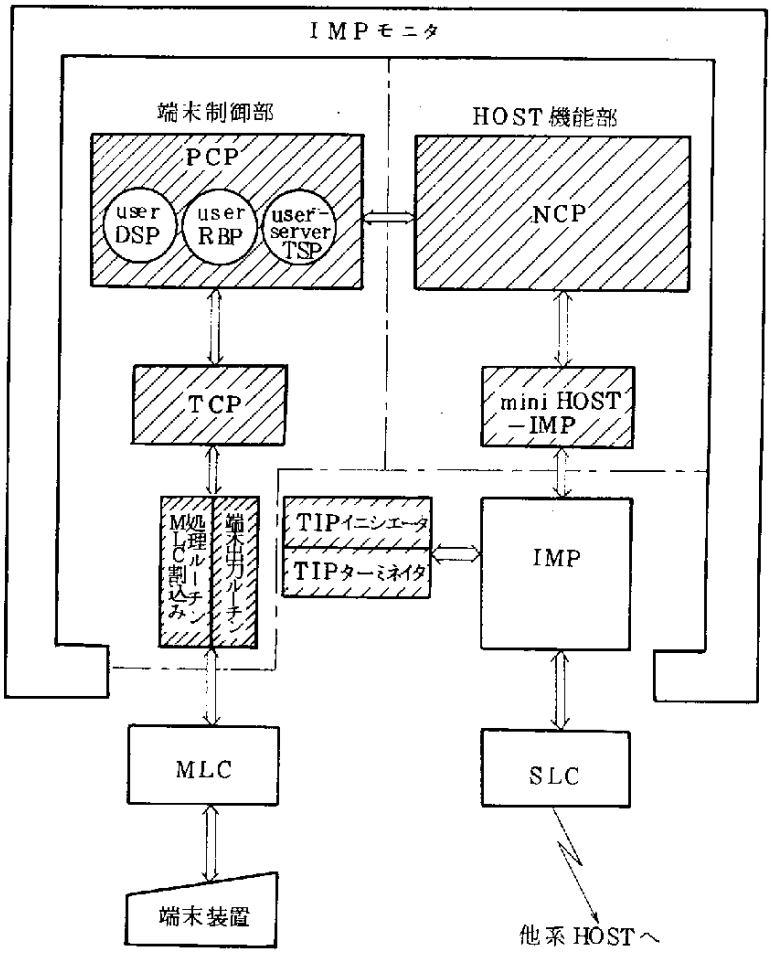



図４－４　ＴＩＰソフトウェアの構造

HOST機能部はNCP（Network Control Program），mini HOST－IMP（mini HOST－IMP interface）タスクよりなっており，HOST－HOSTプロトコル，HOST－IMPインターフェイス，NCPインターフェイスなどをサポートしている。

端末制御部はPCP（Process Control Program），TCP（Terminal Control Program）タスクおよびＭＬＣ割込み処理ルーチン，端末出力ルーチンよりなっており，高位プロトコル，ＮＶＴ（Network Vertual Terminal）ー実端末インターフェイス，ＮＣＰインターフェイスなど

をサポートしている。

ＴＩＰソフトウェアを構成する各タスクはＩＭＰモニタから見た場合，ＩＭＰ機能，ＴＩＰ機能に関係なく同列のタスクとして実行されるがＴＩＰ機能タスクの実行優先権はＩＭＰ機能タスクよりも低く，バックグラウンドタスクよりも高い位置におかれている。また，ＩＭＰの割込み処理ルーチンが，ＭＬＣ（多重通信制御部）の割込みを受け付けたら，ＴＩＰのＭＬＣ割込み処理ルーチンに制御が渡される。

図４－５にＩＭＰモニタとタスクとの関係を示す。

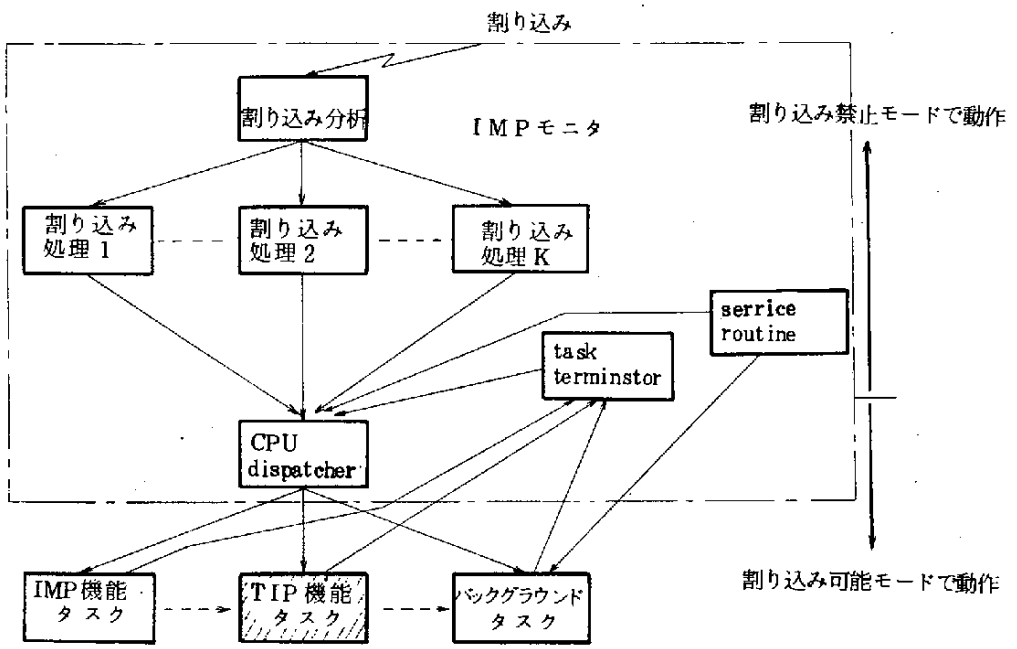



図４－５　ＩＭＰモニタとタスクとの関係

ＩＭＰモニタから見たＴＩＰの各タスクの実行優先順位および各タスクの起動条件となるコミュニケーション起動およびキューの関係，また，各タスクでの制御順序は図４－６に示す通りである。

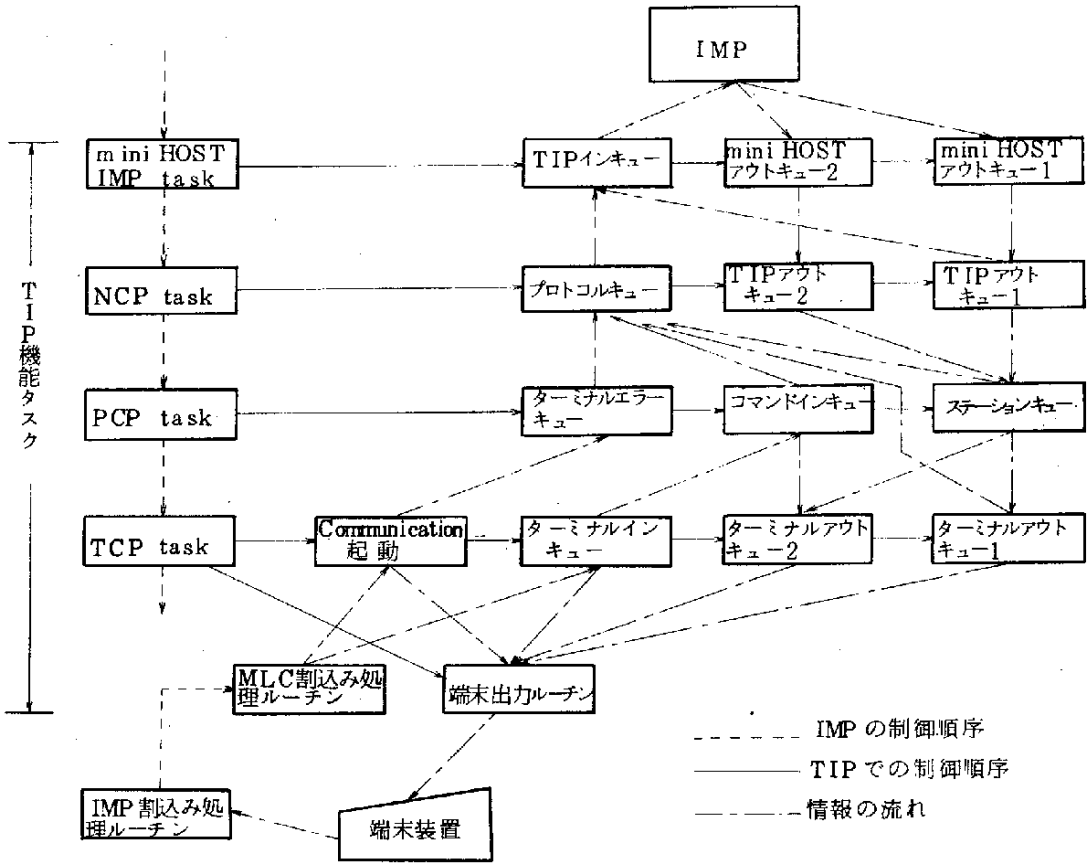



図４－６　制御順序から見たＴＩＰのタスク，キューの関係

### 4.1.4　高位プロトコルの実現方法

ＴＩＰがサービスしている高位プロトコルとしては，DSP(Demand Service Protocol）およびRBP（Remort Batch service Protocol）がある。また，高位プロトコルの範囲には入らないがＴＩＰに接続された端末間での通信，TSP（inter Terminal communication Service Protocol）についてもサービスしている。

ＴＩＰの本質的な機能は，ＴＩＰに接続された１端末から，高位プロトコルをサービスしている複数ＨＯＳＴのうちから任意の１つを選び，ＤＳＰやＲＢＰの利用を可能にすることである。

ＤＳＰおよびＲＢＰのプロトコルについては，2.2.3 Demand Service Protocol，　2.2.4 Remort Batch Protocol に述べてある通りである。ＴＩＰ内のプロセスは，Server側のＨＯＳＴからネットワーク内共通の仮想端末：NVT（Network Vertual Terminal）として制御さ

れる（図４－７）。

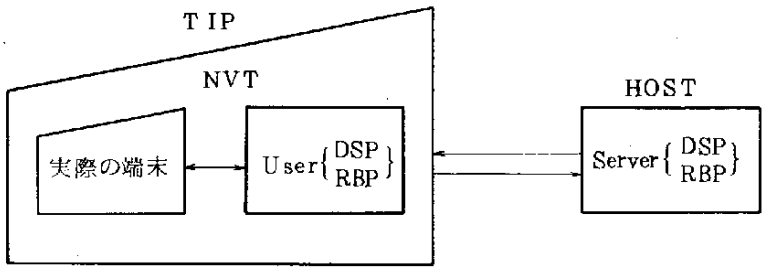



図４－７　ＴＩＰの基本的な制御

ＴＩＰにおけるDSP，RBPおよびTSPの処理はPCP（Process Control Program）によって実現される（図４－８）。

PCPは実端末ユーザからのコネクション確立要求コマンド（OPENコマンド）によって指定されるサービス名により，それぞれの機能をもつプログラムとして実行され，それぞれのプロトコル機能が利用される。

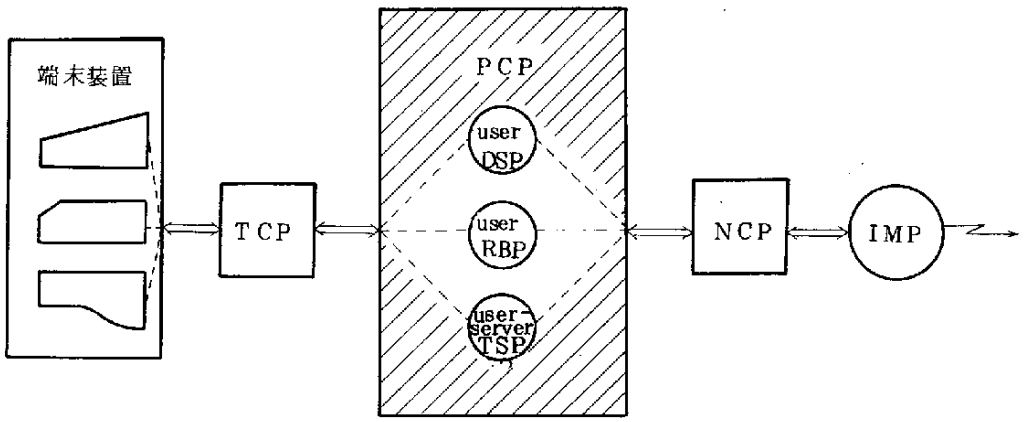



図４－８　高位プロトコルの処理

(1)　ＤＳＰの実現方法

ＤＳＰは，他のＨＯＳＴのＴＳＳをネットワークを通して使用するものである。

図４－９にＤＳＰの処理概念図を示す。ユーザＨＯＳＴ（ＴＩＰ）にはユーザＤＳＰ機能をも

つＰＣＰプロセスがあり，それに対してサーバＨＯＳＴのＮＣＰのもとにはサーバＤＳＰと呼ばれる仮想端末制御プロセスが存在する。サーバＨＯＳＴにおける通常のＴＳＳ使用はサーバＨＯＳＴの実端末（点線）からサーバＨＯＳＴのＴＳＳモニタを経由してユーザジョブと会話を行なうわけである。しかしネットワークを通してユーザＨＯＳＴの端末からサーバＨＯＳＴのユーザジョブと会話する場合は，点線で示した通常の端末へのつながりが，サーバＨＯＳＴのＴＳＳモニタ中でサーバＤＳＰ（仮想端末プロセス）へのつながりへと切り換えられ，両ＨＯＳＴのＮＣＰを通して会話が行なわれることになる。

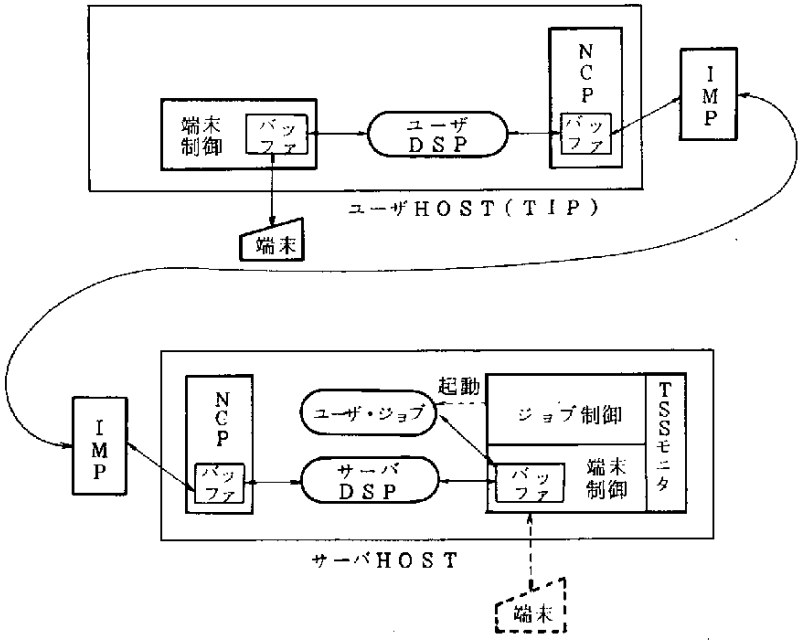



図４－９　ＤＳＰの実現方法

(2)　ＲＢＰの実現方法

ＲＢＰは，端末からソースカードデック（あるいはソースのタイプイン）をリモートＨＯＳＴへ送って処理し，結果を手元に返送させる機能をもったものである。図４－10にＲＢＰの処理の概念図を示す。

ＲＢＰは，ＤＳＰと同様にユーザＨＯＳＴ（ＴＩＰ）にユーザＲＢＰ機能をもつＰＣＰプロセス，サーバＨＯＳＴにサーバＲＢＰプロセスが存在する形で実現される。図中の番号はメッセージおよび制御の順序を示す。サーバＲＢＰは既存のＲＢモニタの人力制御をシミュレートしている。

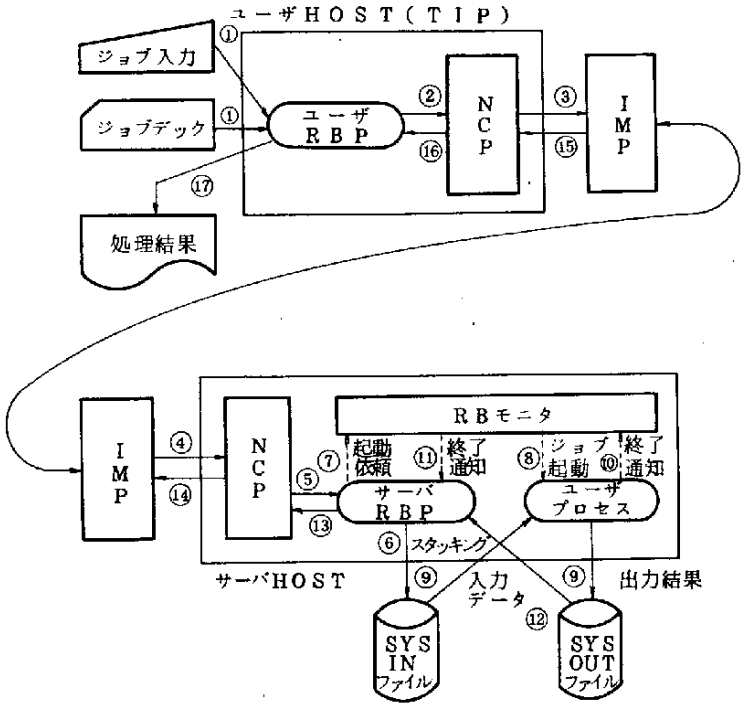



図4－10　RBPの実現方法

### 4.1.5　各タスクの主要機能

(1)　mini HOST−IMP interface タスク

このタスクは図4−11に示すように実際にはIMP機能とTIP機能の中間にあって，TIP

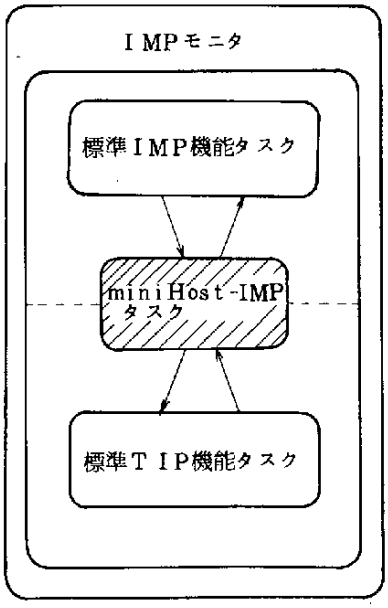



図4－11　mini HOST−IMPの機能

－ＩＭＰ間でのデータの受け渡しおよび目的地ＨＯＳＴの妥当性のチェックなどの機能をもっている。

ＴＩＰ－ＩＭＰ間のデータの受け渡しは，お互いに異なるバッファをもち，そのバッファ間でデータをコピーすることにより行なう。ただし，データの授受にともなう承認としてのＲＥＣＥＩＶＥメッセージに対してはＩＭＰ，ＴＩＰ共通のピースバッファを用いることもある。

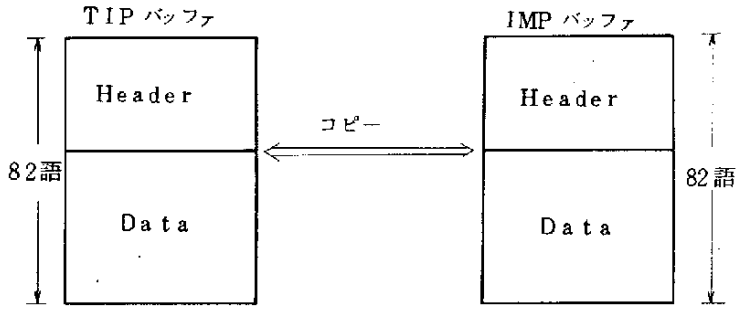



図４－12　ＴＩＰ－ＩＭＰインターフェイス

(2)　ＮＣＰ（Network Control Program）タスク

このタスクは表４－３に示すＮＣＰ制御コマンドの発信と解析およびそれにともなうコネクションの管理，コネクション上を流れるメッセージフローの制御などの機能をもっている。また，一般ＨＯＳＴでのＮＣＰにおいてはＮＣＰサービスコマンドに基づき，ユーザプロセスとのインターフェイスをとっているが，ＴＩＰのＮＣＰにおいてはＮＣＰ制御コマンドのＲＦＣ，ＣＬＳ，ＰＯＳＴをＰＣＰタスク（ユーザプロセス）と直接やりとりすることによりＮＣＰインターフェイスをとっている。

表４－３　ＮＣＰ制御コマンド

| コマンド名 | 機　　能 | パ　ラ　メ　ー　タ |
|---|---|---|
| ＲＦＣ | コネクション確立要求，応答 | my and your local port－id，コネクション番号，最大メッセージ長，メッセージ量 |
| ＣＬＳ | コネクション切断，確立要求拒否 | my and your local port－id |
| ＡＬＬＯＣ | バッファ確保通知 | コネクション番号，バッファ個数 |
| ＰＯＳＴ | 割込み情報等の通知 | コネクション番号，状態情報 |
| ＳＰＯＳＴ | システム情報の通知（エラーなど） | 目的コード（エラー，付加機能のサービス拒否），付加パラメータ |

ＴＩＰにおけるメッセージフロー制御の方法は図４－13に示す通りである。基本的には受信バッファを２個確保し，それに対する２個の受信メッセージを端末に出力し終るのを待って，新たなバッファを確保し相手に通知する方法をとり，ＨＯＳＴ－ＨＯＳＴ間でのメッセージの伝送速度と端末への出力速度の差を調整している。従ってＪＩＰＮＥＴのＨＯＳＴ－ＨＯＳＴプロトコルにおけるフロー制御の特徴であるバッファの個数をＲＥＣＥＩＶＥに便乗させる方法はＴＩＰの場合には採用しておらず受信バッファ個数はすべてＡＬＬＯＣコマンドにより相手に通知している。

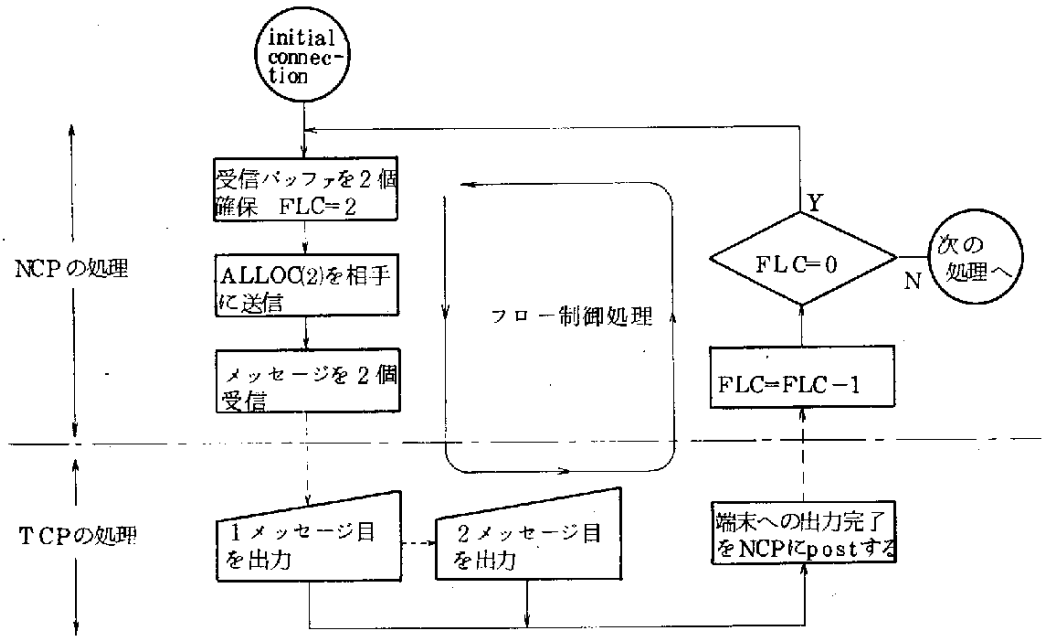



図４－13　メッセージフロー制御の方法

(3)　ＰＣＰ（Process Control Program）タスク

このタスクは表４－４に示すＴＩＰサービスコマンドの解析およびそれらコマンドに基づくステーションの管理，高位プロトコルコマンドによるＮＶＴの制御などの機能をもっている。ＰＣＰとＮＣＰとのインターフェイスはＮＣＰ制御コマンドのＲＦＣ，ＣＬＳ，ＰＯＳＴを直接やりとりすることによりコネクションステータスに関する同期をとっている。

ＴＩＰにおいてはユーザがコントロール端末と入力専用あるいは出力専用端末との組合せを指定することによりステーションとしての機能を利用可能にしている。

ステーションの構成は，入出力可能なコントロール端末のみか，入力専用端末あるいは出力専用端末を必要に応じてＴＩＰ制御コマンドにより付加したものである。ＴＩＰ制御コマンドとステーションを構成する各端末装置との関係を図４－14に示す。

表4 －4　ＴＩＰサービスコマンド

| | |
|---|---|
| @OPEN | DSP，RBPおよび端末間通信サービスを要求する。 |
| @CLOSE | DSP，RBPおよび端末間通信サービスの終了を要求する。 |
| @WAIT | 不特定の端末間通信ユーザに対して自端末を呼び出し待ちの状態にすることを要求する。 |
| @CWAIT | ＷＡＩＴの状態を止めることを要求する。 |
| @SEND | 端末間通信において相手端末にメッセージを送信することを要求する。 |
| @POST | DSPの使用中にリモートホストのサーバーに割込み信号を送ることを要求する。 |
| @ASSIGN | 入力専用端末，出力専用端末をステーションとして定義する。 |
| @FREE | 入力専用端末，出力専用端末をステーションから切り放す。 |
| @TRANSFER | 入力あるいは出力専用端末に対して入出力動作の開始を指定する。 |
| @GIVEBACK | 入力あるいは出力専用端末で行なっていた入出力動作を止める。 |
| @RESTART | 入力専用端末からの入力中に受信エラーを起こしたときその再入力の準備が終ったことを指示する。 |
| @INSERT | newline（ＬＦ）により復帰改行を行なっていたものをＣＲＬＦによる復帰改行に変更することを指示する。省略時は自動的にＣＲＬＦによる復帰改行が行なわれる。 |
| @RESET | ＣＲＬＦにより復帰改行を行なっていたものをnewline（ＬＦ）による復帰改行に変更することを指示する。 |
| @STATUS | ステーションを定義するための入力あるいは出力専用端末の現在の状態の表示を指示する。 |

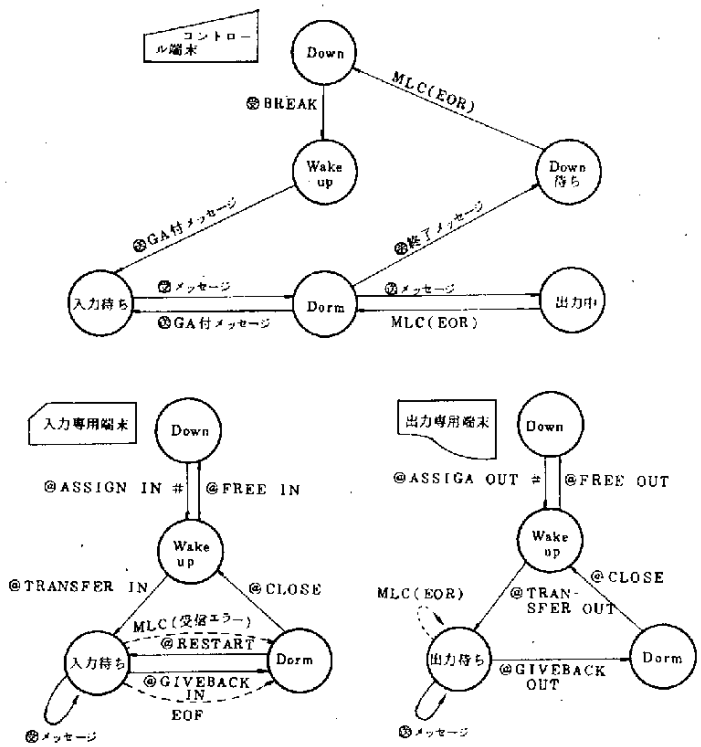



図４－14　ＴＩＰ制御コマンドとステーションの関係

(4) ＴＣＰ（Terminal Control Program）タスク

このタスクは表４－２に示す各実端末装置を制御するものである。端末装置管理テーブルに含まれる端末個有の制御手順およびコード変換情報などに基づき，端末の制御およびコード変換（実端末コード←→ＮＶＴコード）するための機能をもっている。各端末装置に個有な特性の例としては，印字巾の相違，伝送速度と印字速度の相違，復帰改行時間の相違などが挙げられる。

これらの処理はすべて変換ルーチンにそのためのパラメータをもっていくことにより可能にしている。

リモートＨＯＳＴからのメッセージが端末の印字巾を越えている場合の処理として，残りを捨ててしまう方法と次の行に残りを出力する方法があるがＴＩＰでは後者を採用している。そのために図4－15に示すように残り分は別なＴＩＰバッファにコピーをして，それをメッセージ出力待ちキューにキューオンしておき，最初の行の出力完了後，改めて残り分の変換を行ない出力する。

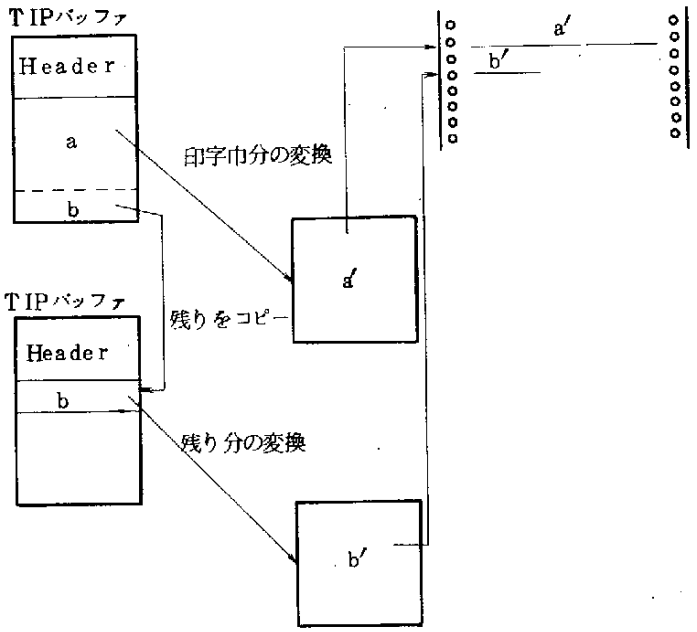



図４－15　メッセージ長と印字巾の相違による処理

伝送速度と印字速度の相違および復帰改行時間の相違などは図4－16に示すように各有効文字の間に非印字文字（ダミー文字）を挿入変換することにより吸収することができる。図４－16はSilent 700 端末の場合の例である。この端末は上記２つの相違に対する処理が必要であり，

m = 3，n = 23 とすることにより正常の印字動作が行なわれる。この値は各端末により異なる。

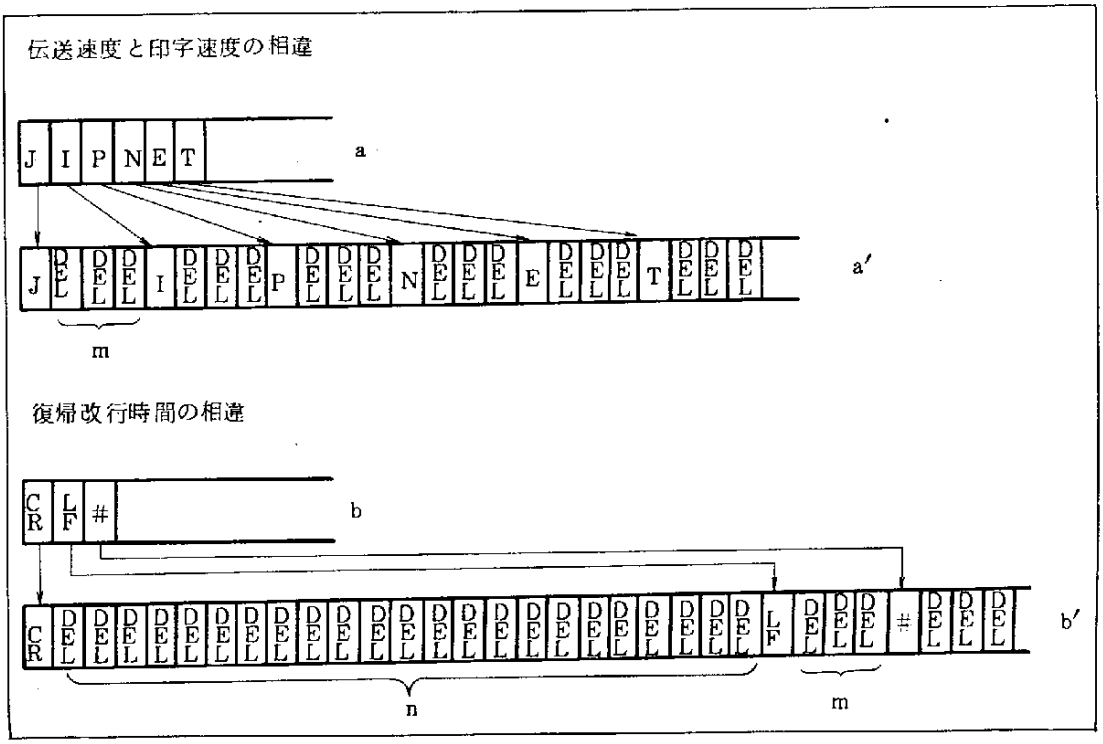



図 4 － 16　伝送速度と印字速度および復帰改行時間の相違による処理（Silent-700 の場合）

(5)　ＩＭＰチェックタスク

このタスクはＴＩＰの機能とは直接には関係ないが，ＴＩＰを１つの独立したＨＯＳＴとしてＩＭＰに接続したり，切断をしたりする場合に使用されるタスクである。コンソールタイプライタからT△START メッセージを入力することにより，ＴＩＰが論理的にＩＭＰに接続されＴＩＰイニシエータに制御が渡される。また，T△ENDメッセージを入力することにより，ＴＩＰターミネイタに制御が渡され，ＴＩＰがＩＭＰから論理的に切断される。これらの関係を図４－17 に示す。

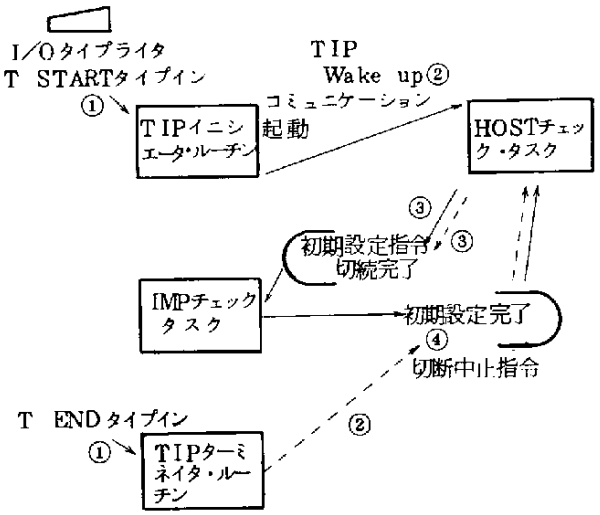



図４－17　ＴＩＰの接続，切断の関係

### 4.1.6　ＴＩＰの制御テーブルおよびバッファの構成

ＴＩＰの制御テーブルは，コネクションの制御を行なうコネクションテーブル，ステーションや端末の制御を行なうステーション管理テーブルおよび端末装置管理テーブルなどから構成されている。ＴＩＰで使用するバッファは，ＴＩＰの各タスク間での情報の授受を行なうＴＩＰバッファおよびＰＩＥＣＥバッファと端末装置に対して入出力を行なうための入力バッファおよび出力バッファより構成されている。

ＴＩＰの制御テーブルおよびバッファの全体的な関係を図４－18に示す。

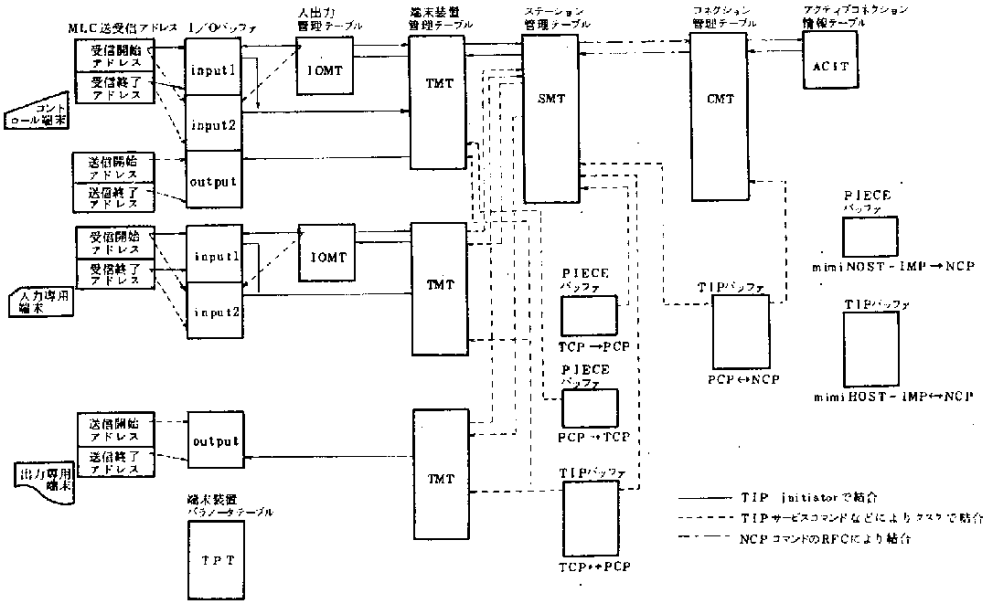



図４－18　ＴＩＰの制御テーブルおよびバッファの関係

(1)　制御テーブルの構成

A．ＣＭＴ

ＣＭＴ（Connection Manegement Table）は，他系ＨＯＳＴとのコネクションを管理するため，ＮＣＰタスクにおいて動的に割り付けられる 26 語の制御情報である。

ＰＣＰタスクからあるいは端末間通信サービスの場合で相手ＴＩＰからＲＦＣ（ＮＣＰコントロールコマンド）が発せられた時，ＣＭＴを割り付ける。同様にＣＬＳ（ＮＣＰコントロールコマンド）が発せられた時，ＣＭＴを返却する。

| | | |
|---|---|---|
| 0 | 次のＣＭＴを指すポインタ | |
| 1 | コネクションの状態 | ＮＣＰの状態 |
| 2 | 目的地ホスト番号 | 受信コネクション番号 ＊ |
| 3 | 目的地ホスト番号 | 送信コネクション番号 |
| 4 | コネクション名 | オプション交渉の状態 |
| 5 | 空 | |
| 6 | my local port－id ＜32ビット＞ | |
| 7 | | |
| 8 | your local port－id ＜32ビット＞ | |
| 9 | | 空 |
| 10 | ＳＭＴを指すポインタ | |
| 11 | 受信バッファキュー （トップ） | |
| 12 | 〃 （テイル） | |
| 13 | 〃 （レングス） | |
| 14 | 送信バッファ個数 | |
| 15 | フロー制御カウンタ | |
| 16 | 送信待ちメッセージキュー （トップ） | |
| 17 | 〃 （テイル） | |
| 18 | 〃 （レングス） | |
| 19 | ＲＢＰコマンドヘッダーコード | |
| 20 | 送信ＮＣＰ制御コマンド数 | |
| 21 | 送信メッセージ数 | |
| 22 | サービス開始時刻 | |
| 23 | 〃 | |
| 24 | 空 | 障害の状態 |
| 25 | ＡＣＩＴを指すポインター ＊ | |

註　＊印の部分はＴＩＰ Initiatorで固定的にセットされる。

図４－19　ＣＭＴのフォーマット

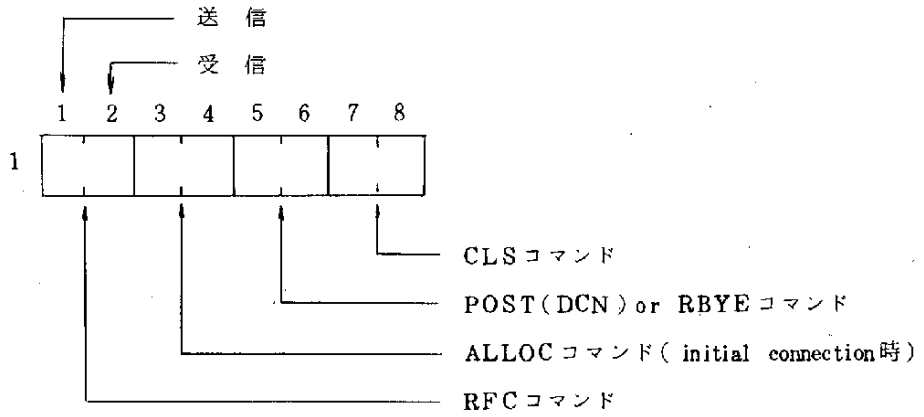
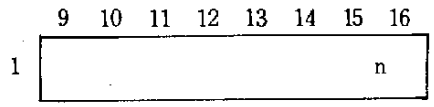
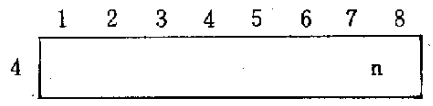



図4－20　コネクションの状態




n＝0　dorm
1　他系ホストからのＲＦＣコマンド待ち
2　他系ＴＩＰへのListen Any
3　自系 user open 待ち
4　他系ホストからのALLOCコマンド待ち
5　Ready
6　他系ホストからのCLSコマンド待ち
7　自系 user close 待ち

図4－21　ＮＣＰの状態




n＝0　TSP（端末間通信）コネクション
1　DSPコネクション
2　RBPコネクション

図4－22　コネクション名

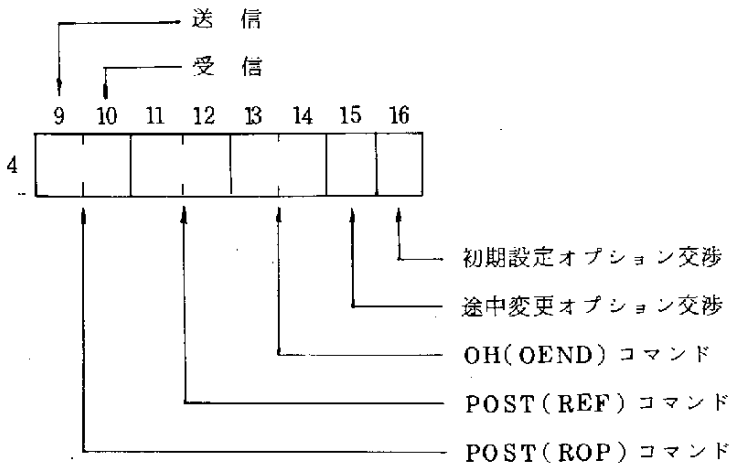



図4－23　オプション交渉の状態

B. ACIT

ACIT(Active Connection Information Table)はCMTに付属したテーブルで3語の制御情報である。

| | |
|---|---|
| 0 | 制御情報 |
| 1 | CMTを指すポインタ　＊ |
| 2 | セットタイマーの値 |

(注)　＊印の部分はTIP initiatorで固定的にセットされる。

図4－24　ACITのフォーマット

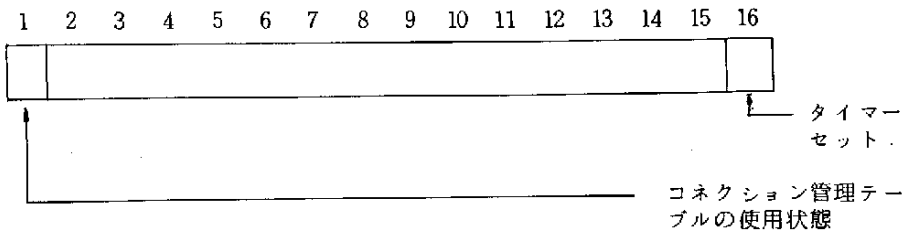



図4－25　制御情報

C. SMT

SMT(Station Manegement Table)は，ステーションを管理するためのテーブルで，TIP initiator routineにより実装されているcontrol terminal(入出力可能な端末)

に対応して8語の制御情報が割り付けられる。

| | | | |
|---|---|---|---|
| 0 | ステーションの状態 | 端末の入力状態 | PCPの状態 |
| 1 | TMTを指すポインタ | （コントロール端末） | |
| 2 | 〃 | （入力専用端末） | |
| 3 | 〃 | （出力専用端末） | |
| 4 | CMTを指すポインタ | | |
| 5 | 送信権待ちメッセージキュー | （トップ） | |
| 6 | 〃 | （テイル） | |
| 7 | 〃 | （レングス） | |

図4－26　SMTのフォーマット

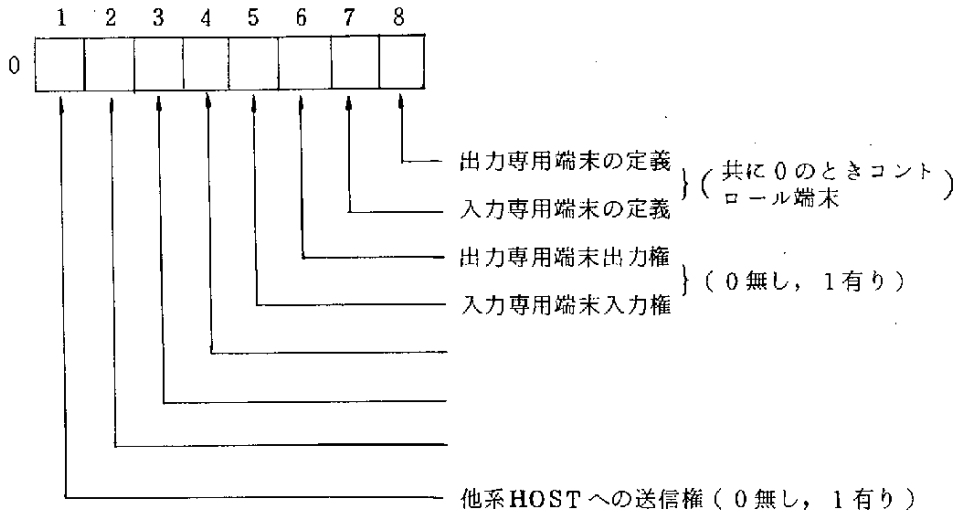



図4－27　ステーションの状態

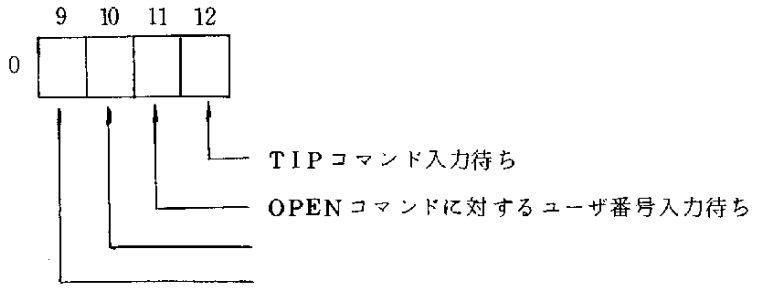
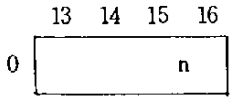



図4－28　端末の入力状態




n = 0　dorm

1　NCP open待ち

2　NCPへのListen Any

3　端末user open待ち

4　端末userからのOPENコマンド待ち

5　Ready

6　NCP close待ち

図4－29　PCPの状態

D. TMT

TMT（Terminal Manegement Table）は，端末を管理するためのテーブルでTIP initiator routineにより実装されている端末装置に対応して16語の制御情報が割り付けられる。

| | | | | |
|---|---|---|---|---|
| 0 | 端末番号 | | 端末の状態 | TCPの状態 | 端末種類 |
| 1 | SMTを指すポインタ | | | |
| 2 | IOMTを指すポインタ | | | |
| 3 | 入力専用端末からの入力レコードカウント値 | | | |
| 4 | 端末出力待ちメッセージキュー1 | | | （トップ） |
| 5 | 〃 | | | （テイル） |
| 6 | 〃 | | | （レングス） |
| 7 | 出力バッファを指すアドレス | | | |
| 8 | 端末出力待ちメッセージキュー2 | | | （トップ） |
| 9 | 〃 | | | （テイル） |
| 10 | 〃 | | | （レングス） |
| 11 | 端末制御の状態 | 空 | TCPの番号 | 改行遅れ挿入ダミー文字数 |
| 12 | HT遅れ挿入ダミー文字数 | | VT遅れ挿入ダミー文字数 | |
| 13 | FF遅れ挿入ダミー文字数 | | 最大ライン巾 | |
| 14 | 空 | | 端末のコード | 印字遅れ挿入ダミー文字数 |
| 15 | 端末装置からの入力エラーカウント値 | | | |

図4－30　TMTのフォーマット

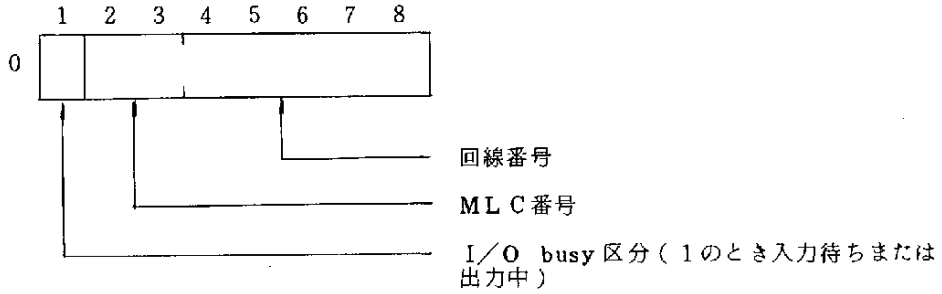
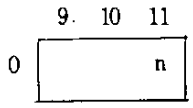



図４－31　端末番号

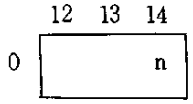



n＝0　Down の状態

1　Ready の状態

2　Run の状態

図４－32　端末状態




n＝0　down

1　wake up

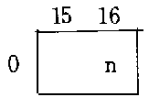
2　入力待ち

3　dorm

4　出力中

5　down待ち

図４－33　ＴＣＰの状態




n＝0　　コントロール端末

1　　入力専用端末

2　　出力専用端末

図４－34　端末種類

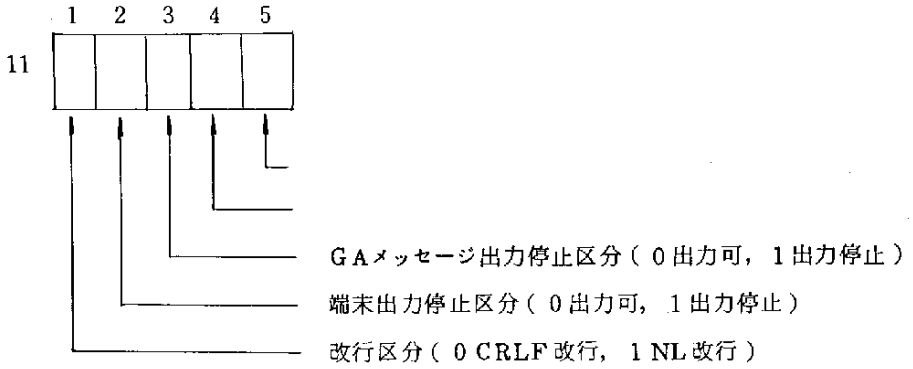
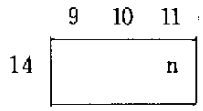



図４－35　端末制御の状態

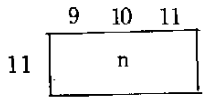



n＝0　　ASCIIコード端末

1　　JISコード端末

2　　EBCDICコード端末

図４－36　端末のコード




n＝1　Terminet 300型制御手順端末

2　Tektronix 4010型制御手順端末

3　MDR 4000型制御手順端末

4　未定

図4－37　TCPの番号

E. IOMT

IOMT（Input／Output Manegement Table）は，入力バッファを管理するためのテーブルでTIP initiatorにより実装されている端末装置に対応して4語の制御情報が割り付けられる。

| | | |
|---|---|---|
| 0 | 端末の状態 | TCPコミュニケーション起動要因 |
| 1 | 現在の入力バッファのアドレス | |
| 2 | 該当TCPに対応したターミナルインキューアドレス | |
| 3 | TMTを指すポインタ | |

図4－38　IOMTのフォーマット

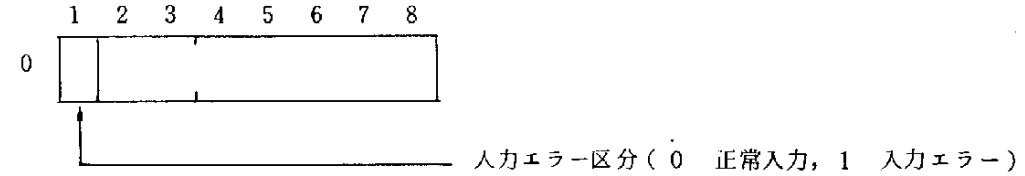
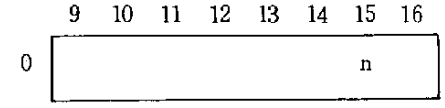



図4－39　端末の状態




n＝1　端末へのデータ送信終了

2　端末からのデータ受信エラー

図4－40　TCPコミュニケーション起動要因

F. TPT

TPT( Terminal Parameter Table )は，端末の個有情報としてＴＩＰ initiatorによる制御情報として用いられるもので５語からなる。

これはＴＩＰの個有情報としてＴＩＰソフトウェアとは別に紙テープの形で作られており，ＴＩＰソフトウェアがローディングされた時，ＴＩＰの固定番地に読み込まれ，TIP initiatorで使用される。

| | | | | |
|---|---|---|---|---|
| 0 | 端末番号 | | TCPの番号 | 改行遅れ挿入ダミー文字数 |
| 1 | HT遅れ挿入ダミー文字数 | | VT遅れ挿入ダミー文字数 | |
| 2 | FF遅れ挿入ダミー文字数 | | 最大ライン巾 | |
| 3 | 空 | 端末種類 | 端末のコード | 印字遅れ挿入ダミー文字数 |
| 4 | 空 | | 出力バッファサイズ | |

図４－41　ＴＰＴのフォーマット

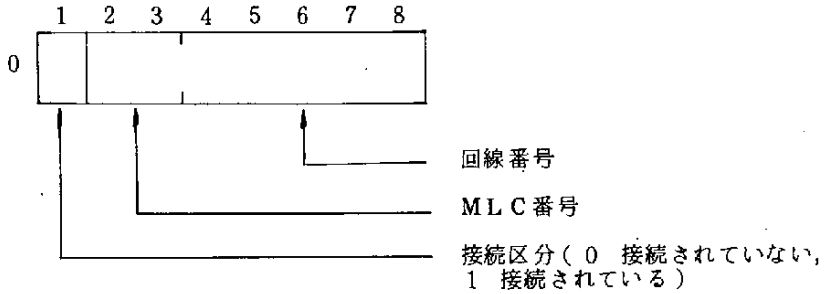


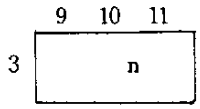

図４－42　端末番号


n＝0　コントロール端末

1　入力専用端末

2　出力専用端末

図４－43　端末種類




n＝0　　ASCII コード端末

1　　JIS コード端末

2　　EBCDIC コード端末

図４－44　端末のコード

(2)　バッファの構成

A．TIPバッファ

TIPバッファは，TIPの各タスク間での情報の授受のために使用される82語からなるバッファであり，30個用意されている。

各タスクで必要に応じてTIP buffer get routine を呼ぶごとに1個のTIPバッファが渡される。またTIP buffer free routine を呼ぶことにより返却される。

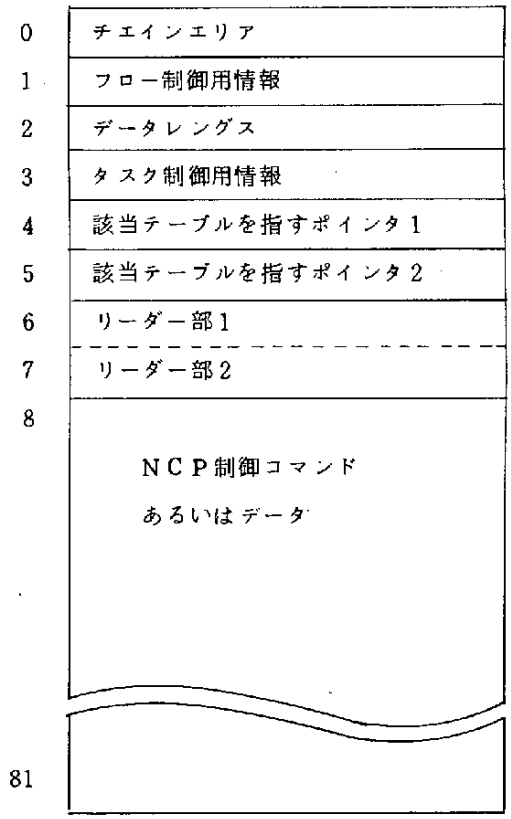



図４－45　TIPバッファのフォーマット

B．ＰＩＥＣＥバッファ

ＰＩＥＣＥバッファは，ＩＭＰおよびＴＩＰの各タスク間での情報の授受のために使用される４語からなる共用バッファであり，180個用意されている。

各タスクで必要に応じてＰＩＥＣＥ get routine を呼ぶごとに１個のＰＩＥＣＥバッファが渡される。また，ＰＩＥＣＥ free routine を呼ぶことにより返却される。

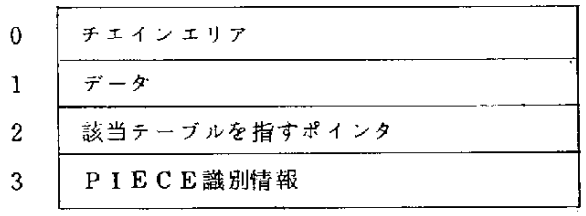



図４－46　ＰＩＥＣＥバッファのフォーマット

C．入力バッファ

入力バッファは，実装されている各端末毎にＴＩＰ initiator で割り付けられる85語（制御情報（５語）＋データエリア（80語））からなるバッファである。入力バッファは各端末毎に２個用意されている。

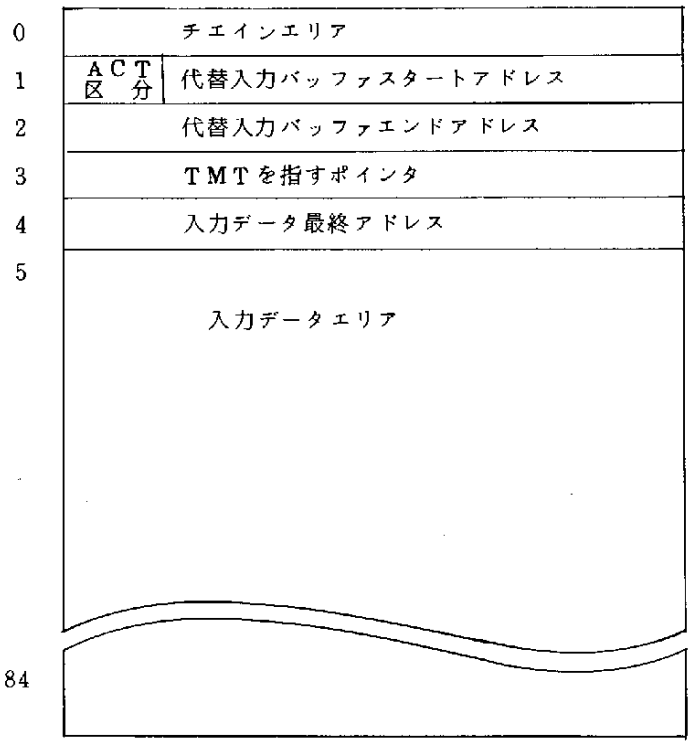



図４－47　入力バッファのフォーマット

D．出力バッファ

出力バッファは，実装されている各端末毎に１個，ＴＩＰ initiator で端末の個有情報に基づいて割り付けられる任意語長からなるバッファである。

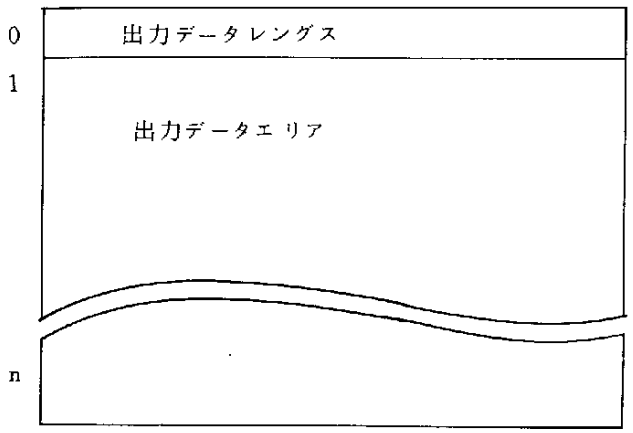



図４－48　出力バッファのフォーマット

## 4.2　論理設計

### 4.2.1　概　　要

4.1節での基本設計をもとに，ＴＩＰソフトウェアの構成要素である各タスク（各タスクに付属するユーティリティルーチンは除く）および重要な処理ルーチンについてその処理概要を述べる。

ＴＩＰの処理機能としては，

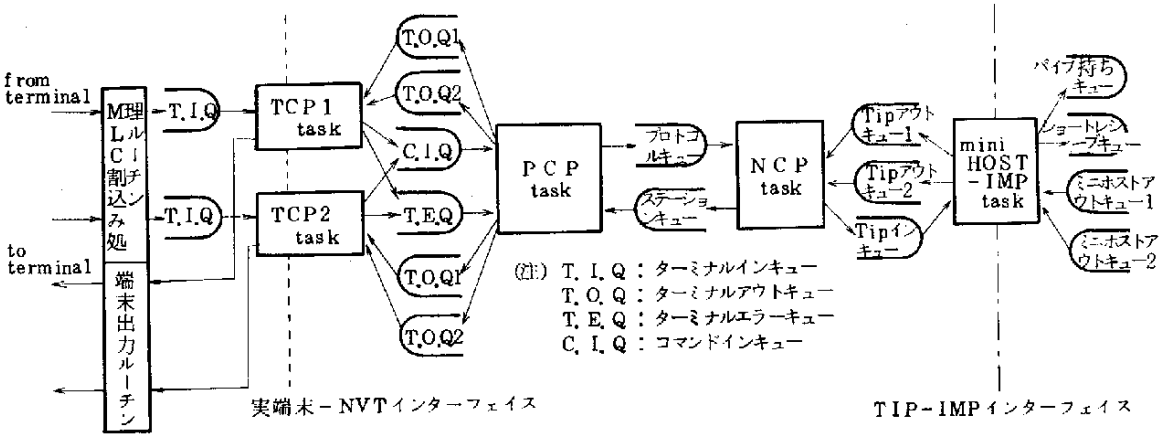



図４－49　ＴＩＰにおけるタスクとキューの関係

(1) ＨＯＳＴ機能部

(2) 端末制御部

がある。また，ＴＩＰソフトウェア自身の起動（初期設定），切断のための機能もある。

これらを処理するために必要となるタスク，キューの関係を図４－49に示す。

### 4.2.2 ＴＩＰイニシエータ・ルーチン

コンソールタイプライタに，"T△START" をタイプインすることにより起動されるルーチンで，以下に述べる初期設定を行なう。

(1) ＴＩＰの諸テーブルエリア，ＭＬＣの割込み情報エリアのクリア

(2) ＴＩＰバッファのフリーチエインの作成

(3) ＴＰＴ（Terminal Parameter Table）の情報にもとずき，図4－18 に示すような各テーブルおよびバッファの結合と情報のセット

(4) ＭＬＣ回線のオープン

### 4.2.3 ＴＩＰターミネイタ・ルーチン

コンソールタイプライタに"T△END"とタイプインすることにより起動されるルーチンで，ＴＩＰの切断を行なう。

### 4.2.4 ＭＬＣ割込み処理ルーチン

ＩＭＰの割込み処理ルーチンより，ＭＬＣの割込みに対して，制御が渡されＭＬＣの割込み分析を行なう。図４－50にＭＬＣの割込み情報を示す

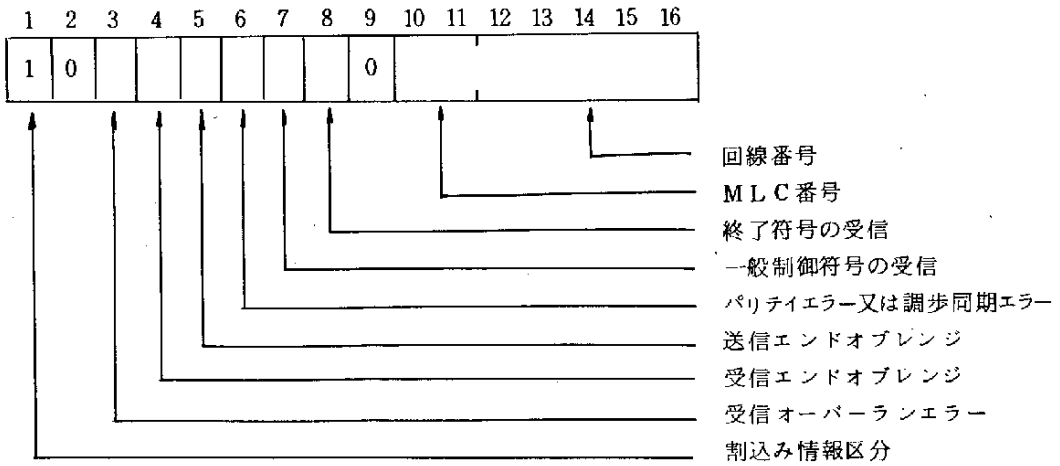



図４－50　ＭＬＣの割込み情報

以下に個々の割込みに対する処理を示す。

(1) 送信エンドオブレンジ割込み

端末へのメッセージの送信終了割込みで，ＴＣＰタスクのキューに出力待ちメッセージがあれば出力させるため，ＴＣＰにコミュニケーション起動をかける。

(2) 一般制御符号割込み

一般制御符号割込みとして現在，ＣＲ，ＥＯＴをサービスしている。ＣＲやＥＯＴは入力データの終了符号として使用し，端末よりの入力データをＴＣＰの入力キューであるターミナルインキューにキューオンする。

(3) 調歩同期エラーあるいは受信オーバーランエラー割込み

調歩同期エラーはハード上のエラーであるが，受信オーバーランエラーは，入力バッファがＭＬＣに対してセットされていないときに端末からデータが入力された場合などに起る。この場合には端末に対して再入力要求を出す必要がある，そのためのメッセージを出力させるため，ＴＣＰにコミュニケーション起動をかける。

(4) パリティエラー割込み

"ＢＲＥＡＫ"符号は入力データが０となるため，パリティエラーとなる。従って，入力データが０でパリティエラーの場合は"ＢＲＥＡＫ"符号として扱い，ＴＣＰの入力キューであるターミナルインキューにキューオンする。それ以外の場合はハード上のパリティエラーとして再入力要求を出す必要があり，ＴＣＰにコミュニケーション起動をかける。

### 4.2.5 端末出力ルーチン

端末への出力メッセージを出力するルーチンであり，ＴＣＰにおいて出力メッセージがＮＶＴコードから実端末コードに変換され，出力の準備が整った時点で呼ばれる。

指定された回線番号の出力指定エリアに出力バッファの先頭アドレスおよび最後のアドレスをセットする処理を行なう。

### 4.2.6 TCP ( Terminal Control Program ) タスク

端末を制御するタスクで，端末制御手順ごとに作成しなければならなく，新しい手順の端末が付けられると作成し追加しなければならないタスクである。

機能として

(1) 端末の制御

(2) コードの変換 ( 実端末コード⟷ＮＶＴコード )

(3) リモートＨＯＳＴからのメッセージを端末に出力し終ったことをＮＣＰに post ( フロー制御)

などがある。

ＴＣＰが起動される条件として，キュー起動とコミュニケーション起動がある。

A．コミュニケーション起動

端末へ出力中に，同一端末あてに，キューオンされたメッセージ（出力待ちメッセージ）は，出力中メッセージの出力終了後，続いて出力しなければならない。出力待ちメッセージのキュー管理用エリアは，端末装置管理テーブルにある。ＭＬＣ割込み処理ルーチンは，端末への送信終了割込みを検知したとき，ＴＣＢ（該当ＴＣＰに対応する）のコミュニケーション起動エリアのセットおよび入出力管理テーブルのコミュニケーション起動要因ステータスをセットする。コミュニケーション起動要因ステータスには，出力待ちメッセージの出力要求と入力データ中のエラーに対する再入力要求メッセージ"ＲＥＴＲＡＮＳ"の出力要求の２種類ある。

ＭＬＣ割込み処理ルーチンにより，コミュニケーション起動されると，ＴＣＰはコミュニケーション起動要因ステータスにより，"ＲＥＴＲＡＮＳ"メッセージを出力するか，出力待ちメッセージキューのメッセージを出力する。なお，入力中のエラーに対してはエラーカウントを＋１して連続して５回以上エラーが起った場合にはピースバッファをgetして，情報をセットしＰＣＰタスクのターミナルエラーキューにキューオンする。もし，"ＲＥＴＲＡＮＳ"メッセージ出力後の再入力により正しく入力されればエラーカウントの値は０にセットされる。

B．キュー起動

(1) ターミナルインキュー

端末ユーザの入力データキューでＭＬＣ割込み処理ルーチンによりキューオンされる。

キューオンされた入力バッファの入力終了アドレスから逆算して入力データ長を求め，それが０の場合には"ＢＲＥＡＫ"符号なので，ピースバッファをgetして，情報をセットしＰＣＰのコマンドインキューにキューオンする。データ長が０でない場合には，ＴＩＰバッファをgetして，コード変換（実端末コード⟷ＮＶＴコード）を行ないＰＣＰのコマンドインキューにキューオンする。

端末がＴＩＰに対して入力権がないときの入力データは"ＢＲＥＡＫ"を除き，棄却され端末に"ＩＬＬＥＧＡＬ　ＩＮＰＵＴ"メッセージが出力される。端末がＴＩＰに対して入力権を得るのは，"ＢＲＥＡＫ"に対する入力促進文字の出力およびリモートＨＯＳＴよりのＧＡ付メッセージを出力したときである。

(2) ターミナルアウトキュー２

端末への出力データのキューである。ＴＩＰサービスコマンドに対する応答メッセージや"ＢＲＥＡＫ"に対する入力促進文字（？）のキューでＰＣＰでＴＩＰバッファやピースバッファとしてキューオンされる。

キューオフされたＴＩＰバッファやピースバッファが指している端末装置がＩ／Ｏ　busyかどうかを端末装置管理テーブルから知り，busy でなければ，コード変換（ＮＶＴコード→

実端末コード）を行ない，端末出力ルーチンに出力を依頼する。また出力の終ったＴＩＰバッファやピースバッファはfree される。もし，該当端末がbusy の場合には端末装置管理テーブルの出力待ちキューにキューオンされる。

(3) ターミナルアウトキュー1

ターミナルアウトキュー2と同様に端末への出力データキューである。このキューにはリモートＨＯＳＴからのメッセージのみがＴＩＰバッファの形式でＰＣＰにおいてキューオンされる。

キューオフされたＴＩＰバッファはターミナルアウトキュー2と同様に処理されるが，出力し終った場合でもfree されず，フロー制御における端末への出力完了として情報を付加し，ＮＣＰタスクのプロトコルキューにキューオンされる。

### 4.2.7 ＰＣＰ（Process Control Program）タスク

ＴＩＰの各サービスを実現するための中心となるタスクで，各サービスのプロトコルの手順に基づいて作成される。ＴＣＰは個々の端末をサービスするが，このタスクはステーションを単位として管理する。

機能としては

(1) ステーションの制御

(2) ＴＩＰサービスコマンドの解析

(3) 高位プロトコルコマンドの解析

などを行なうタスクである。

このタスクで作られたステーションは，ＮＣＰタスクから見た場合，ＮＶＴと仮想して処理される。

このタスクが起動される条件は，キュー起動のみである。このタスクで対象とするキューと処理関係を図4−51に示す。

なお，以下にＤＳＰサービスの場合を想定して処理概要を述べる。

(1) ターミナルエラーキュー

端末からの入力中に回復不可能な障害が起きた場合，ＴＣＰでピースバッファの形式でその情報がキューオンされる。

すでにコネクションが確立されている場合には，リモートＨＯＳＴのサーバＤＳＰに対してＤＣＮ（disconnect）を通知するために，ＰＯＳＴコマンドをプロトコルキューにキューオンし，相手からのＣＬＳコマンド待ちに入る。まだコネクションが確立されていない場合には，このステーションを強制的にclose する。

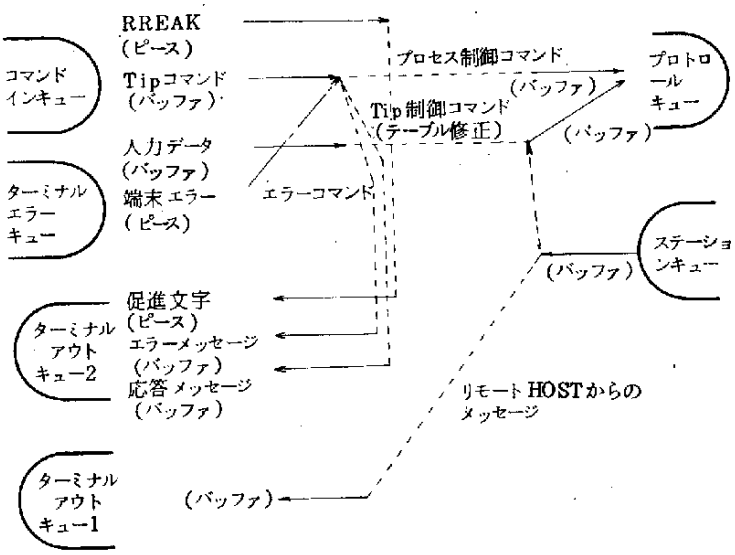



図４－51　対象キューと処理関係

(2)　コマンドインキュー

端末から入力された"ＢＲＥＡＫ"，ＴＩＰサービスコマンドおよびリモートＨＯＳＴへのメッセージに対してＴＣＰでキューオンされる。

キューにはピースバッファおよびＴＩＰバッファが含まれる。従って両者を識別するためピースバッファの第４ワード目は全ビットオンにセットされている。

処理対象が"ＢＲＥＡＫ"情報であれば，それに対する応答として入力促進文字（？）を同じピースバッファにセットして，ＴＣＰのターミナルアウトキュー２にキューオンする。

ＴＩＰサービスコマンドはコマンドの機能により処理後，キューオンされるキューの対象が変わる。ＴＩＰ制御コマンドはステーション管理テーブルあるいは端末装置管理テーブルの内容を修正し，応答メッセージをターミナルアウトキュー２にキューオンする。コネクションの確立，切断要求，割込みおよびＴＳＰ関係コマンドは，対応するＮＣＰコマンドに変換してＮＣＰのプロトコルキューにキューオンする。ＴＩＰサービスコマンドの誤りに対してはエラーメッセージをＴＣＰのターミナルアウトキュー２にキューオンする。

リモートＨＯＳＴへのメッセージの場合には，ステーション管理テーブルから端末側（ user ＤＳＰ）に送信権があるかどうかを知り，送信権があればＮＣＰのプロトコルキューにそのままキューオンする。ＤＳＰにおいては１つメッセージを送信すると送信権がなくなるので送信権ビットをオフにする。端末側に送信権がないときのメッセージに対してはステーション管理テーブルの送信待ちメッセージキューにキューオンする。

(3) ステーションキュー

リモートＨＯＳＴからの端末装置に対する出力メッセージ（高位プロトコルコマンドが含まれる場合もある），高位プロトコルコマンドおよびＮＣＰインターフエイスに対応するＮＣＰコマンドに対してＮＣＰタスクでＴＩＰバッファの形式でキューオンされる。

高位プロトコルコマンドのうち，ＧＡあるいはＦＳのように送信権の授受に関するコマンドはステーション管理テーブルを修正する。またＤＣＮの場合はＣＬＳコマンドに変換し，ＮＣＰのプロトコルキューにキューオンしてステーションを強制的にcloseする。

出力メッセージは，ステーション管理テーブルよりステーションの状態（現在の出力対象がコントロール端末か，出力専用端末かの状態）により対応する端末を決定し，該当するＴＣＰタスクのターミナルアウトキュー1にキューオンする。

ＮＣＰインターフエイスに対応するＮＣＰコマンドについては，コマンドの内容により，ステーション管理テーブルのＰＣＰステータスを修正する。

### 4.2.8 ＮＣＰ（Network Control Program）タスク

このタスクはリモートＨＯＳＴとのメッセージのやりとりを行なうタスクで，実際のサービスの対象となるのは高位プロトコル（端末間通信サービスもこのタスクでは高位プロトコルとして扱っている）のみである。

このタスクはＨＯＳＴ機能部であるが，リモートＨＯＳＴのもつサービスを単に利用するだけであり，相手ＨＯＳＴから利用されることはない。従って，本来のＨＯＳＴ機能の一部であり，ＩＭＰから見た場合，mini HOSTと呼ぶ。

ＰＣＰはステーションを単位としてサービスするが，このタスクではコネクションの管理とともにＰＣＰタスク以下をＮＶＴと仮想して管理する。

機能としては

(1) コネクションの管理

(2) ＮＣＰ制御コマンドの解析

(3) リモートＨＯＳＴとのデータフロー管理

(4) 高位プロトコルに基づくオプション交渉

などを行なうタスクである。

このタスクが起動される条件は，キュー起動のみである。このタスクで対象とするキューと処理関係を図4－52に示す。

(1) プロトコルキュー

このキューにはＮＣＰ制御コマンドやリモートＨＯＳＴへのメッセージがＰＣＰタスクにおいてキューオンされる。他に，フロー制御情報がＴＣＰタスクによりキューオンされる。

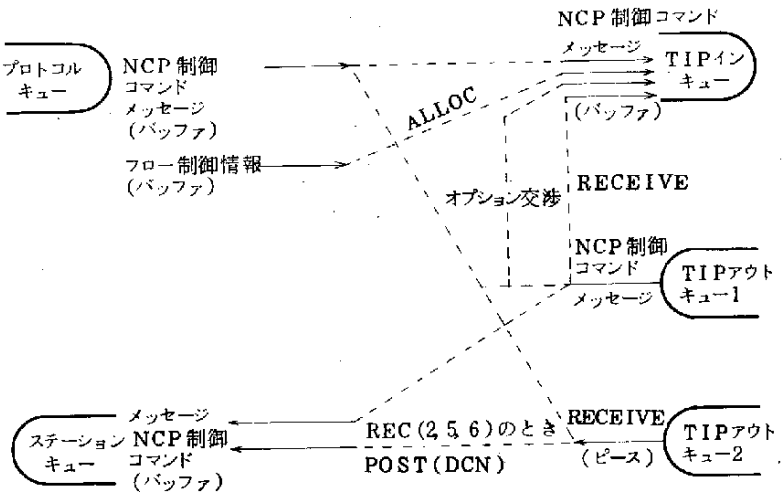



図4－52　対象キューと処理関係

リモートHOSTへのメッセージは，コネクション管理テーブルから送信バッファ個数（リモートHOSTにとって受信バッファ個数）を知り，個数が0でなければそのままTIPインキューにキューオンする。もし，個数が0であれば送信待ちメッセージキュー（コネクション管理テーブル内にある）にキューオンする。

フロー制御情報の場合には，フロー制御カウンターを－1する。フロー制御カウンタが0のときには受信バッファを確保（必ず2個）してALLOCコマンドをリモートHOSTに送る。

NCP制御コマンドは，NCPインターフエイスに基づき解析される。コネクションの確立を要求するRFCコマンドではコネクション管理テーブルをgetし，ステーション管理テーブルとの結合および情報をセットした後，TIPインキューにキューオンされる。その他のコマンドも必要な処理を行なった後，TIPインキューにキューオンされる。

(2)　TIPアウトキュー2

リモートHOSTへのメッセージおよびNCP制御コマンドの送信に対する受取り承認メッセージ（RECEIVE）に対して，mini HOST－IMPタスクにおいてピースバッファ形式でキューオンされる。

RECEIVEの状態には5種類あり，受信拒否，目的地IMP又は目的地HOSTのダウンあるいは目的地IMPなしなどの異常状態に対してはPOST（DCN）コマンドに変換し，PCPタスクのステーションキューにキューオンする。メッセージの送信に対する正常RECEIVEの場合でかつ受信バッファ個数が便乗されている場合には，コネクション管理テーブルに登録し，も

し送信待ちメッセージがあればそれを送信するためＴＩＰインキューにキューオンする。それ以外のＲＥＣＥＩＶＥはすてられる。

(3) ＴＩＰアウトキュー1

このキューには，ＮＣＰ制御コマンド，リモートＨＯＳＴからのメッセージがＴＩＰバッファの形式でmini HOST－IMPタスクでキューオンされる。

リモートＨＯＳＴからのメッセージには，そのまま端末へ出力するメッセージと高位プロトコルのオプション交渉に関するコマンドが含まれている。オプション交渉に関するものについてはこのタスクで解析され，その応答をＴＩＰインキューにキューオンする。それ以外はステーションキューにそのままキューオンされる。

ＮＣＰ制御コマンドはHOST－HOSTプロトコルに基づき解析されコネクション管理テーブルのＮＣＰステータスを修正し，情報をセットする。しかも，ＮＣＰインターフエイスにも関係するコマンドはステーションキューにキューオンされる。

このキューにキューオンされたデータについてはすべてＲＥＣＥＩＶＥを作成し，ＴＩＰインキューにキューオンする。なおメッセージに対するＲＥＣＥＩＶＥの場合には，どのメッセージのＲＥＣＥＩＶＥかを区別するため，このキューに渡されたときのメッセージ番号をセットする。

### 4.2.9 mini HOST—IMP interface タスク

一般ＨＯＳＴのためのＨＯＳＴ－ＩＭＰタスクと同様な機能を有するタスクで，mini HOSTとしてＩＭＰと同一プロセッサ内のＴＩＰを対象に処理するタスクである。このタスクによりＴＩＰは機能的に独立したＨＯＳＴとして扱うことができる。

機能としては

(1) ＴＩＰとＩＭＰ間でのデータの受け渡し

(2) 目的地ＨＯＳＴの妥当性および目的地ＨＯＳＴが同一ＩＭＰ内のとき，ＨＯＳＴのステータス（down or active）により，ダウンのときにはdown RECEIVEをＮＣＰに送る。

(3) ＴＩＰが現在active でないときに，ＴＩＰ向けメッセージを受信した場合にはＴＩＰ down RECEIVEを相手に対して送る。

などを行なうタスクである。

このタスクが起動される条件は，キュー起動のみである。このタスクで対象とするキューと処理関係を図４－53に示す。

(1) ＴＩＰインキュー

ＮＣＰタスクでキューオンされたＮＣＰ制御コマンド，リモートＨＯＳＴへのメッセージおよびＲＥＣＥＩＶＥなどのデータをリーダ部の内容により区別してＩＭＰの各タスクのキューにキューオンする。この場合，ＴＩＰバッファのデータはＩＭＰバッファにコピーしてからキューオン

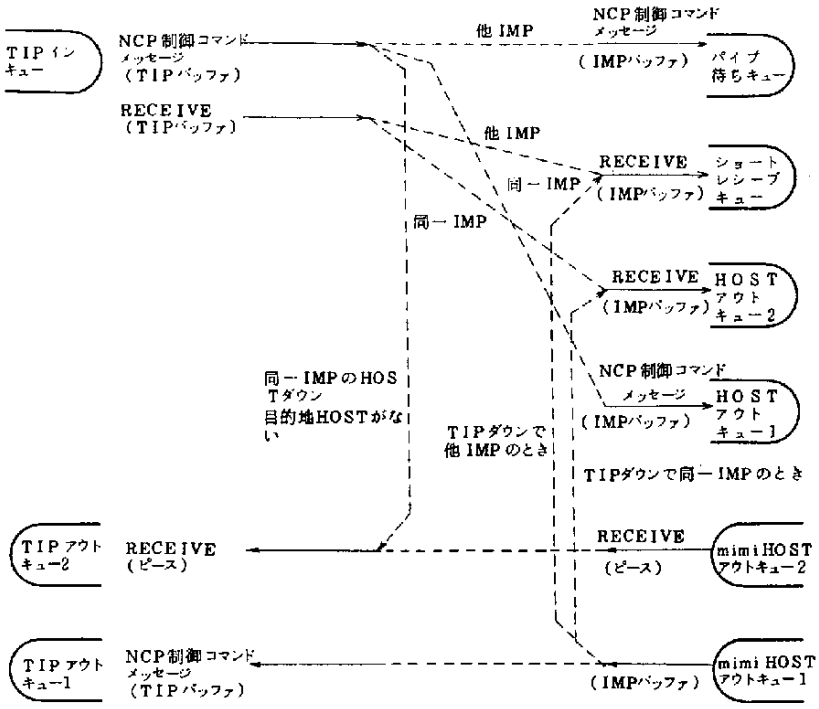



図４－53　対象キューと処理関係

される。同一ＩＭＰ内の目的地ＨＯＳＴがダウンであるとか，目的地ＨＯＳＴがない場合はその旨のＲＥＣＥＩＶＥをＮＣＰのＴＩＰアウトキューにキューオンする。

(2)　mini HOSTアウトキュー２

リモートＨＯＳＴが，ＴＩＰからのＮＣＰ制御コマンドあるいはメッセージを受信したことに対するＲＥＣＥＩＶＥであり，ピースバッファの形式でキューオンされている。このキューの内容はそのままＴＩＰアウトキュー２にキューオンされる。

(3)　mini HOSTアウトキュー１

リモートＨＯＳＴからのＮＣＰ制御コマンドあるいはメッセージがＩＭＰバッファの形式でキューオンされている。従ってＩＭＰバッファからＴＩＰバッファにコピーしてからＴＩＰアウトキュー１にキューオンされる。ＴＩＰがダウンの状態であればdown RECEIVEを作成してＩＭＰのキューにキューオンする。

## 4.3　ＴＩＰの利要

### 4.3.1　概　　要

ＴＩＰがユーザ端末に対してサービスするのは，DSP，RBPおよびTSPである。TIPにおいては station がそれぞれのサービスを受ける単位となる。station とは type writer，display 装置のように会話型で使用できる端末（ control 端末と呼ぶ）が１台と必要に応じて card reader のような入力専用端末が１台，line printer のような出力専用端末が１台より構成される。従って，control 端末のみでも station として扱われる。

ユーザはＴＩＰサービスコマンドにより任意に station を構成し，この station からそれぞれのサービスを受けることができる。

### 4.3.2　システム構成

ＪIPNETの現構成，ＨＯＳＴ番号，端末番号は図４－54に示す通りである。

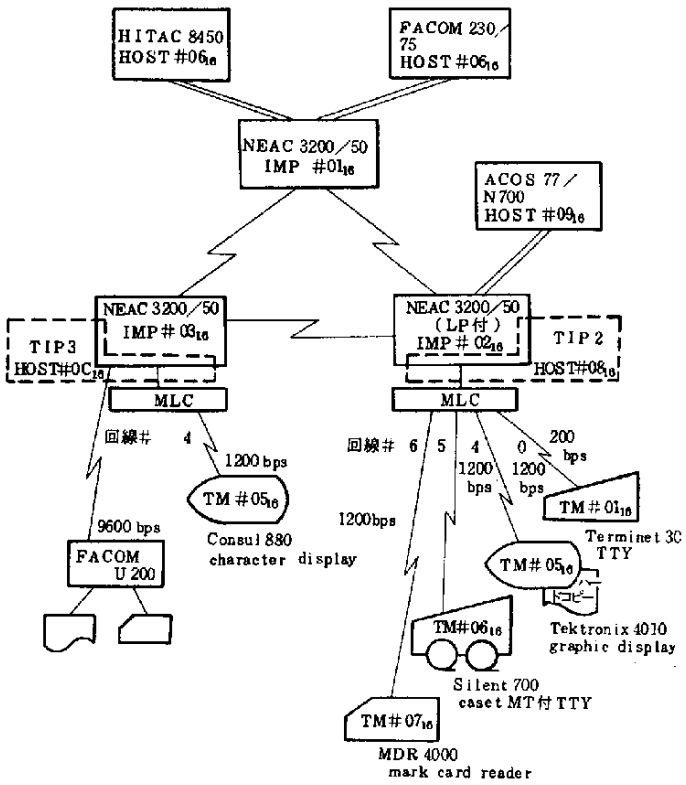



図４－54　ＪIPNETの構成

現在ＴＩＰから使用可能なサービスは，以下の通りである。

ＤＳＰサービス

- ＦＡＣＯＭ　230−75　ＴＳＳ
- ＡＣＯＳ　77／Ｎ700　ＴＳＳ

ＴＳＰサービス

- ＴＩＰ＃２－ＴＩＰ＃３の端末間
- ＴＩＰ＃２あるいはＴＩＰ＃３内での端末間

ＲＢＰサービス

51年度に予定している。

### 4.3.3　ＴＩＰサービスコマンド

ＴＩＰサービスコマンドの機能は次の４つに分類することができる。

- ＴＩＰ制御コマンド

  stationの定義，stationの制御，端末の改行制御および端末状態の問合わせなどを指示するコマンド

- コネクションの確立，切断要求コマンド

  ＴＩＰと他系ＨＯＳＴとのコネクションの確立および切断を指示するコマンド

- 割込みコマンド

  他系ＨＯＳＴに対して割込みを通知するコマンド

- ＴＳＰ関係コマンド

  端末間通信のみで使用するコマンドで，メッセージの送信コマンド

など

以下にＴＩＰサービスコマンドの形式および機能について述べる。

A．ＴＩＰ制御コマンド

ＡＳＳＩＧＮコマンド

一般形

@ＡＳＳＩＧＮ△ { ＩＮ△＃ / ＯＵＴ△＃ }

＃は16進数で端末番号を指定する

機　能

このコマンドによりstationを定義する。このコマンドで指定した入力あるいは出力専用端末はあらかじめreadyの状態（オンライン）になっていなければならない。なおこのコマンドはＯＰＥＮコマンドの前で入力しなければならない。

省略形

@A△{IN△#<br>OUT△#}

FREEコマンド

一般形

@FREE△{IN<br>OUT}

機　能

ASSIGNコマンドにより定義された入力あるいは出力専用端末をstationから切放す。このコマンドを入力しなくても，@CLOSEコマンドにより自動的にstationから切放される。

省略形

@F△{IN<br>OUT}

TRANSFERコマンド

一般形

@TRANSFER△{IN<br>OUT}

機　能

ASSIGNコマンドによりstationとして組み込まれた入力あるいは出力専用端末に対して，入出力動作の開始を指定する。

このコマンドにより入力専用端末に与えた入力権は入力エラーなどを生ずると自動的に放棄され，control端末に返される。

省略形

@T△{IN<br>OUT}

GIVEBACKコマンド

一般形

@GIVEBACK△{IN<br>OUT}

機　能

入力あるいは出力専用端末で行なっていた入出力動作をcontrol端末から行なうように指示する。

省略形

@G△{IN / OUT}

RESTARTコマンド

一般形

@RESTART

機　能

入力専用端末からの入力中，受信エラーを起こした後，restartの準備が終った時点で，入力を再開することを指示する。

省略形

@RES

INSERTコマンド

一般形

@INSERT△CR

機　能

newline（LF）により復帰改行を行なっていたものをCRLFによる復帰改行に変更することを指示する。省略時は自動的にCRLFによる復帰改行が行なわれる。

省略形

@I△CR

RESETコマンド

一般形

@RESET△CR

機　能

CRLFにより復帰改行を行なっていたものをnewline（LF）による復帰改行に変更することを指示する。

省略形

@R△CR

STATUSコマンド

一般形

@STATUS△TM

機　能

stationを定義するための入力あるいは出力専用端末の現在の状態の表示を指示する。

省略形

@S△TM

B．コネクションの確立，切断要求コマンド

OPENコマンド

一般形

@OPEN△ { DSP / RBP } △#$_H$〔，#$_I$〕

#$_H$，#$_T$

機　能

DSP，RBPおよびTSPサービスの開始を要求する。#$_H$は16進数でDSP，RBPおよびTSPを行なう相手のHOST番号である。#$_I$はDSPの利用において同一ユーザ番号をもつユーザが同時に複数端末から使用する場合その識別（同一port idができるのを防ぐため）として，ユーザの使用できる識別番号$40_{16}$～$FF_{16}$の中から1個をとり出して指定する。#$_T$は16進数でTSPを行なう相手の端末番号である。

省略形

@O△ { DSP / RBP } △ #$_H$〔，#$_I$〕

#$_H$，#$_T$

USER－NOの指定

一般形

USER－NO＝#

機　能

OPENコマンドに付属するものでOPENコマンドでDSPおよびRBPサービスの開始を要求した場合，端末にこのメッセージが出力されるので必ず応答しなければならない。#は16進数でDSPおよびRBPを利用するユーザに割り当てられた6桁のユーザ番号である。

CLOSEコマンド

一般形

@CLOSE

機　能

現在行なっているDSP，RBPおよびTSPサービスの終了を要求する。

省略形

@C

C．割込みコマンド

POSTコマンド

一般形

@POST△ { IP / AO }

機　能

DSPに対するコマンドであって，DSP serverに割込み信号（NVTで決められた特殊符号）を送ることを要求する。

IPはInterrupt process，AOはAbort output処理の要求を行なう。

省略形

@P△ { IP / AO }

D. TSP関係コマンド

WAITコマンド

一般形

@WAIT

機　能

不特定のTSPユーザに対して自端末を呼び出し待ちの状態にすることを要求する。

省略形

@W

CWAITコマンド

一般形

@CWAIT

機　能

WAITの状態を止めることを要求する。

省略形

@CW

SENDコマンド

一般形

@SEND△s

機　能

TSPにおいて相手端末にメッセージを送信することを要求する。sは60文字以内の任意の文字列である。

省略形

@SE△s

### 4.3.4 ＴＩＰの使用法

端末よりＴＩＰのサービスを受けようとするユーザは，次の段階の手順をふまなければならない。

① ＴＩＰと端末の接続

② 端末特性の設定およびステーションの定義（任意）

③ サービスの開始要求

④ サービスを受ける

⑤ サービスの終了要求

Ａ．使用手順

(1) ＴＩＰと端末の接続

ユーザはＴＩＰからサービスを受けたい場合には，まず第１にコントロール端末から会話開始要求として"ＢＲＥＡＫ"キーをキーインしなければならない。

この動作により，ＴＩＰは？を端末に対して出力し，ＴＩＰサービスコマンドの入力待ちに入る。

(2) 端末特性の設定およびステーションの定義（任意）

必要があればＴＩＰ制御コマンドにより，入力専用端末や出力専用端末を付加したり，改行の制御などの指定を行なう。

(3) サービスの要求

ＯＰＥＮコマンドにより，ＤＳＰ，ＲＢＰあるいはＴＳＰのサービスを要求する。ＯＰＥＮコマンドによるサービスの要求がＤＳＰあるいはＲＢＰの場合にはユーザに割当てられたユーザ番号を問い合わせてくるので，それに答えなければならない。この応答手順の例は次の通りである。

例：

```
(BK) ?@OPEN  DSP  6
     USER-NO  =  060001
     DSP  SERVICE  START
     以下はリモートHOSTのＴＳＳの手順に従う。
```

(4) サービスを受ける

ＴＩＰより　×××　ＳＥＲＶＩＣＥ　ＳＴＡＲＴと出力した後，この状態に入る。この状態におけるメッセージの入出力はすべてリモートＨＯＳＴのＴＳＳの手順に従う。

この状態のときにＴＩＰサービスコマンドを入力したい場合には，"ＢＲＥＡＫ"キーをキーインし，？が出力されたら続いてコマンドを入力すればよい。

(5) サービスの終了要求

ＴＩＰのサービスを終了させたい場合は，ＣＬＯＳＥコマンドにより行なう。応答形式の例は次の通りである。

例：

(BK)?@ＣＬＯＳＥ

ＤＳＰ　ＳＥＲＶＩＣＥ　ＥＮＤ

B．ＴＩＰのメッセージ

(1) 入力エラーなどに対するメッセージ

a．端末装置からの入力に対するハードエラー

ＲＥＴＲＡＮＳ　　コントロール端末のとき

ＲＥＴＲＡＮＳ　nnn　　入力専用端末のとき

↑

エラーになったカードの位置

b．端末に対して入力を要求していないのに何らかの入力があった場合

ＩＬＬＥＧＡＬ　ＩＮＰＵＴ

c．再入力の要求

?＞

(2) ＴＩＰサービスコマンドなどに対する応答メッセージ

a．ＴＩＰサービスコマンドの入力要求

?

b．ユーザ番号の入力要求

ＵＳＥＲ－ＮＯ＝

c．要求されたサービスの開始

ＤＳＰ　ＳＥＲＶＩＣＥ　ＳＴＡＲＴ

ＲＢＰ　ＳＥＲＶＩＣＥ　ＳＴＡＲＴ

ＴＳＰ　ＳＥＲＶＩＣＥ　ＳＴＡＲＴ

d．要求されたサービスの終了

ＤＳＰ　ＳＥＲＶＩＣＥ　ＥＮＤ

ＲＢＰ　ＳＥＲＶＩＣＥ　ＥＮＤ

ＴＳＰ　ＳＥＲＶＩＣＥ　ＥＮＤ

e．ＴＩＰサービスコマンドに対するエラー

ＥＲＲＯＲ－ＮＯ　nnnnnn

↑

エラーを検出した番地（エラー内容を示す）

f．ＴＩＰサービスの強制的終了

ＳＹＳＴＥＭ　ＣＬＯＳＥ　　ｎｎｎｎｎｎ

↑

終了条件を検出した番地（終了条件を示す）

g．ＴＩＰ制御コマンドの処理完了

ＳＥＴ　ＯＫ

h．ＰＯＳＴ　ＡＯコマンドの処理完了

ＡＢＯＲＴ　ＯＵＴＰＵＴ　ＯＷＡＲＩ

i．ＳＴＡＴＵＳ　ＴＭコマンドの応答

ＴＥＲＭＩＮＡＬ　　ＳＴＡＴＵＳ

＊ＤＯＷＮ　　＃　　＃は端末番号

＊ＲＥＡＤＹ　　＃

＊ＲＵＮ　　＃

j．リモートホストが入力待ちなのにＴＩＰサービスコマンドを入力した場合

＊ＫＥＹ　ＩＮ　ＰＬＥＡＳＥ＊　ＲＥＭＯＲＴ　ＨＯＳＴ　ＩＮＰＵＴ　ＷＡＩＴ

k．端末間通信においてＷＡＩＴコマンド（ listen ）に対して相手端末から呼ばれた場合

ＹＯＵＲ　ＴＥＲＭＩＮＡＬ　ＣＡＬＬ（ＨＯＳＴ－ＮＯ　＃，ＴＥＲＭＩＮＡＬ－ＮＯ　＃）

### 4.3.5　ＴＩＰの使用例

以下にＤＳＰサービスの使用例を示す。

Ａ．ＤＳＰサービスの使用例１（ＦＡＣＯＭ　230−75　ＴＳＳ）

```
?#OPEN DSP 6                                          ＤＳＰサービスの開始要求
USER-NO = 060001
DSP SERVICE START                                     コネクションの確立応答
FACOM 230-75 *DSP* CONNECTION OK *                    FACOM DSP SERVERからの受付け応答
\\CPS                                                 FACOM TSSの呼び出し
. FACOM 230 M-6/7 CPS(V04-L11) KAISI 76.03.05 17:38:12   FACOM TSSとの初期会話
+ YU-ZA MEI ?=NETWORK
+ PASUWA-DO ?=
CPS JOB NO.=#T1.0001
. CPS(V-02/L-00) SIYOKISETUTEI SEIZIYOU SIYUURIYOU
. SIYUTURIYOKU MATI*NASI
#FORTRAN                                              システム名の指定
#OLD STRING                                           プログラムSTRINGの呼び出し
#LIST                                                 リストの指定

 IDENT=000                                            プログラムリストの出力

00010       DIMENSION K1(50),K2(50)
00020       DIMENSION L(80)
00030       WRITE(6,10)
00040 10    FORMAT(1H ,'ENTER NN,SS,STRING=')
00050       READ(5,20,END=90) N,M,(K1(I),I=1,N)
00060 20    FORMAT(2I2,50A1)
00070       M1=M+1
00080       K1(M1)=1H
00090       DO 30 I=1,M
00100 30    K2(I)=I+1
00110       K2(M1)=1
00120       LL=80/M1*M1+1
00130       J=1
00140       DO 60 II=1,N
00150       DO 40 I=1,LL
00160       L(I)=K1(J)
00170       J=K2(J)
00180 40    CONTINUE
00200       WRITE(6,50) (L(I),I=1,LL)
00210 50    FORMAT(1H ,80A1)
00220 60    CONTINUE
00230 90    STOP
00240       END

#RUN                                                  プログラムSTRINGの実行

. PUROGURAMU MEI: FORTRAND, KANRIYOU KO-DO: 000.      コンパイルおよびリンクの終了応答
. PUROGURAMU MEI: LIED    , KANRIYOU KO-DO: 000.

ENTER NN,SS,STRING
>5007NETWORK                                          データ入力要求およびデータの入力
NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK   実行結果
 NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWOR   の出力
K NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWO
RK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETW

?#POST AO                                             出力の停止を要求
ORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NETWORK NET
#ABORT OUTPUT OWARI                                   出力停止要求に対する終了応答
#BYE                                                  ＴＳＳの使用を止める
   ********************************                   ＴＳＳ使用に対するアカウント情報
   *** JOB ACCOUNTING INFORMATION ***
   * TERMINAL          197.000SECS. *
   * CPU TIME            1.000SECS. *
   * MEMORY TIME       176.697SECS. *
   * HCM X TIME       2548871KW.MS *
   * LCM X TIME             0KW.MS *
   * HCM MAX.MEMORY      32832KW    *
   * LCM MAX.MEMORY         0BLOCKS*
   * WORK FILE           2044KB     *
   * GENERAL FILE           0KB     *
   ********************************
. SETUSIYON OWARI. NETWORK      17:41:29              ＴＳＳの終了応答

?#CLOSE                                               ＤＳＰサービスの終了要求
DSP SERVICE END                                       コネクションの終了応答
```

図４−55　FACOM 230−75　ＴＳＳの使用例

図４−55　ＦＡＣＯＭ230−75　ＴＳＳの使用例

## B. ＤＳＰサービスの使用例２（ＡＣＯＳ　77／700　ＴＳＳ）

```
?@OPEN DSP 9                                          ＤＳＰサービスの開始要求
USER-NO = 090001
DSP SERVICE START                                     コネクションの確立応答

ACOS-6 TSS(02.0) ON 03/12/76 AT 15.489  CHANNEL 2300  ＡＣＯＳ　ＴＳＳとの初期会話


user id -NET
password--
[redacted]

SYSTEM ?FORTRAN                                       システム名の要求および入力
OLD OR NEW-OLD                                        プログラムＦ２の呼び出し
OLD FILE?.F2
READY
*LIST                                                 リストの指定

0050      DIMENSION K1(50),K2(50)                     プログラムリストの出力
0060      DIMENSION L(80)
0061      DATA LLL/1H /
0070      WRITE(6,10)
0080 10   FORMAT(1H ,'ENTER NN,SS,STRING ')
0090      READ(5,20)N,M,(K1(I),I=1,M)
0100 20   FORMAT(2I2,50A1)
0110      M1=M+1
0120      K1(M1)=LLL
0130      DO 30 I=1,M
0140 30    K2(I)=I+1
0150      K2(M1)=1
0160      LL=70/M1*M1-1
0170      J=1
0180      DO 60 II=1,N
0190      DO 40 I=1,LL
0200      L(I)=K1(J)
0210      J=K2(J)
0220 40   CONTINUE
0230      WRITE(6,50) (L(I),I=1,LL)
0240 50   FORMAT(1H ,70A1)
0250 60   CONTINUE
0260      STOP
0270      END

ready

*RUN                                                  プログラムＦ２の実行
ENTER NN,SS,STRING
=9906JIPNET                                           データ入力要求およびデータの入力
JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET      実行結果
 JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNE            の出力
T JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPN

?@POST IP                                             ＴＳＳへの割込みを要求
ET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIP
NET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JIPNET JI
ACCEPT IP(BREAK FOR ACOS-TSS)                         割込み要求に対する受付け応答
*BYE                                                  ＴＳＳの使用を止める
**cost:      $51 to date:      $5,099=  0%            ＴＳＳ使用に対するアカウント情報
**on at 15.488 - off at 15.537 on 03/12/76
ACOS-TSS SERVICE END                                  ＴＳＳの終了応答

?@CLOSE                                               ＤＳＰサービスの終了要求
DSP SERVICE END                                       コネクションの終了応答
```

図４－５６　ACOS 77/N700　ＴＳＳの使用例

図４－５６　ACOS 77／N700 TSSの使用例

# 5. ネットワーク・パフォーマンスの測定・評価

# 5. ネットワーク・パフォーマンスの測定・評価

## 5.1 測定・評価の意義

システムの作成は，設計，インプリメンテーション，測定，評価の一連のプロセスを経て完成される。

しかし，システムの規模が大きくなるにつれ，必らずしも一通りの経過で最適なシステムを完成させることはむづかしい。測定の結果，当初目標としたシステム基準に達していない場合，あるいはより改良の予地があると推定される場合，再三再四，設計，インプリメンテーション，測定，評価のプロセスが繰り返されることがあり得る。

コンピュータ・ネットワークは，多少個々の差はあるにしても，現在のシステムの中では少なくとも大規模システムの範ちゅうに含まれるものが多く，特に測定・評価の必要性が高く，且つシステムの改良のためのサイクルが不可欠なものの一つである。

使用者にとっても，設計者にとっても，実際に稼動しているコンピュータ・ネットワークがどの位の性能があるのかは，大きな問題である。ＴＳＳ利用者にとっては，会話的な短いメッセージのラウンド・トリップ遅延（round-trip delay）がどの位なのか，ファイル転送のような大量データ伝送を行う使用者にとっては，スループットがどの位なのかが関心事である。設計者にとっては，これら以外にネットワーク全体でどの位の性能を呈するかが関心事となる。

ここでは，あるＨＯＳＴ内のプロセスがネットワークを通して他のＨＯＳＴ内のプロセスへメッセージを伝送する時のスループット，ラウンド・トリップ・タイムなどの測定結果をとりあげる。

## 5.2 1パケット・メッセージのパケット・オーバヘッド

1パケット・メッセージは，図5－1に示すフォーマットを推移してプロセス間で授受される。プロセスがメッセージの送信要求をＮＣＰに発すると，ＮＣＰは送信メッセージに対して，どのＨＯＳＴの何番目のコネクション上のメッセージかを示すメッセージ・リーダ（32ビット）と呼ばれる情報を前置し，発信地ＩＭＰへメッセージ・テキストと共に送り出す。発信地ＩＭＰはこの情報からパケット（パケット・ヘッダは96ビット）を作成し，隣接ＩＭＰへ伝送する。この時，回線上を通るパケットはハードウェア（伝送制御部）によって開始フラグ（８ビット），ＦＣＳ（Frame Check Sequence）（16ビット）および終了フラグ（８ビット）が付加される。パケットを受け取った目的地ＩＭＰは，パケットを分解し，メッセージ・リーダとセグメントに組立て直し，ＮＣＰに

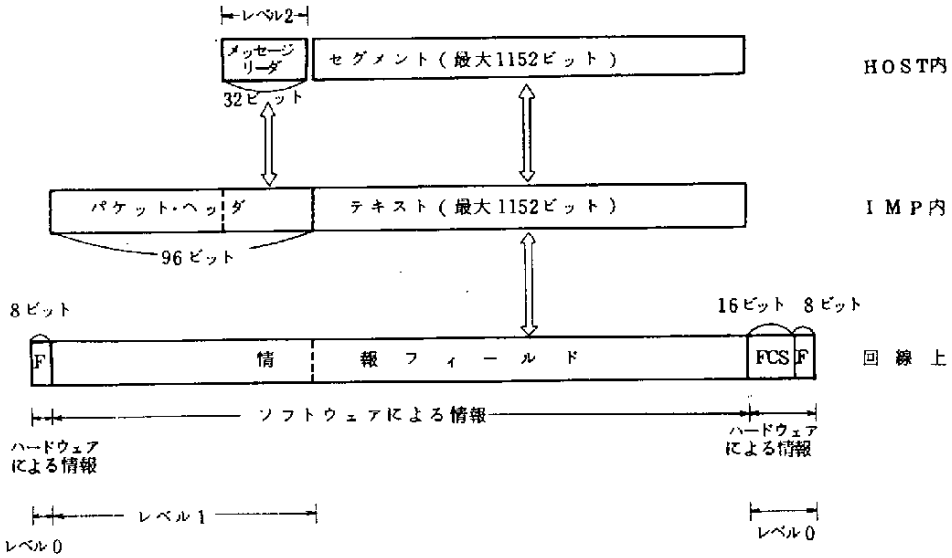



図5－1　1パケット・メッセージ・フォーマットの推移

渡す。これらを受け取ったＮＣＰはメッセージ・リーダからコネクション，即ちプロセスを識別し，プロセスにセグメント（データ）のみを渡す。

回線上を通るメッセージに対するオーバヘッドは，合せて128ビット，すなわち16バイトとなる。ＴＳＳ利用の場合，通常，平均入力データ長は12文字，平均出力データ長は44文字であるので[8]，オーバヘッドは各々57％と27％にもなる。1フル・パケット・メッセージ（144バイト）でも10％に達し，このオーバヘッドは無視できない。

## 5.3　ネットワーク・オーバヘッド

ネットワーク・オーバヘッドとは，ここではプロセス間で授受されるデータ以外のネットワークを通るすべてのビットと定義する。

ネットワーク・オーバヘッドは，次の4つの範ちゅうに論理的に分類することができる。

(1)　レベル－0 オーバヘッド

隣接ＩＭＰ間のパケット伝送のための制御

(2)　レベル－1 オーバヘッド

発信地ＩＭＰと目的地ＩＭＰ間の伝送制御

(3)　レベル－2 オーバヘッド

ＨＯＳＴ間のメッセージ制御

(4)　バックグランド・オーバヘッド

ルーティング情報などのメッセージ

表5－1にネットワーク・オーバヘッドの分類を示す。

表5－1　ネットワーク・オーバヘッドの分類

| 範ちゅう | 名　前 | ビット数 | 説　　明 |
|---|---|---|---|
| レベル－0 | F | 8 | 開始フラグと呼ばれ，ハードウェアが発生するフレームの開始を示すフラグであり，前のフレームの終了フラグで代用することが可能である |
| | FCS | 16 | フレームに対し，ハードウェアが発生するチェックサムである |
| | F | 8 | 終了フラグと呼ばれ，ハードウェアが発生するフレームの終了を示すフラグであり，次のフレームの開始フラグを兼ねさせることができる |
| | ACK | 32 | 隣接IMP間で伝送されたパケットに対するpositive acknowledgement信号である |
| レベル－1 | パケットヘッダ | 96 | データ・パケットの先頭に付加されるパケットの目的地番号などが入っている制御情報である |
| | サブネット制御 | 80 | サブネット内の伝送制御及びフロー制御のための制御パケットである |
| レベル－2 | メッセージリーダ | 32 | HOST間のメッセージの宛名などが入っている制御情報である |
| | Receive・ | 32 | コネクションを通して送信されたメッセージに対するエンド・ツー・エンドの確認信号である |
| | HOST間制御 | 平均：56<br>最大：568 | HOST間のトラフィックを制御するための可変長の制御メッセージである |
| バックグランド | ルーティング | 336 | サブネット内のIMPおよび回線のアップ・ダウンにともなうルーティング情報の制御パケットである |

レベル－0 オーバヘッド: 隣接IMP間のパケット伝送の制御に関するオーバヘッドはハードウェアで発生されるフレーム文字である8ビットのフラグ，チェックサムである16ビットのFCS，そしてIMP間のacknowdgementである32ビットのACKである。ACKの返送は逆方向のパケットがある場合には便乗させている。

レベル－1 オーバヘッド: 発信地IMPと目的地IMP間でのオーバヘッドは各データ・パケット毎の96ビットのパケット・ヘッダと各々80ビットのサブネット制御パケットである。

レベル－2 オーバヘッド：ＨＯＳＴ間のメッセージ制御に関するオーバヘッドは，ＨＯＳＴ－ＨＯＳＴプロトコルに規定された各メッセージの前に付加される32ビットのリーダ，受信側ＨＯＳＴから送信側ＨＯＳＴに返送されるメッセージが受信側ＮＣＰに到着したことを示す32ビットのReceiveメッセージ，そしてAllocateようなＨＯＳＴ－ＨＯＳＴプロトコルの制御メッセージである。

バックグランド・トラフィック：バックグランド・トラフィックはルーティング・パケットと回線ステータスをチェックするためのパケットである。

これらのオーバヘッドが有効データ伝送速度に対しどの位影響を与えるかは，トラフィックの特性に依存し，特に実際に伝送されるメッセージのサイズとＨＯＳＴ－ＨＯＳＴプロトコルの制御メッセージの個数が大きな影響を与えると言われる。

## 5.4　メッセージの伝送制御手順

### 5.4.1　単一パケット・メッセージの伝送制御手順

図５－２に単一パケット・メッセージに対するＨＯＳＴレベル及びサブネット・レベルのメッセージ伝送制御手順を示す。

送信ＨＯＳＴは発信地ＩＭＰへメッセージ・リーダを送り，つづけて第１セグメントを送る。発信地ＩＭＰはまず最初に目的地ＩＭＰへのS-Pipe(４本)が１個以上空いているか調べる。もし，空いていれば直ちにそのメッセージ(パケット)が発送されるが，全部使用中の場合はメッセージの発送を保留する。その目的地からの既に送出した単一パケット・メッセージに対するReceiveが返送されてきた時，Ｓ－Pipeが１個空き状態となり，このメッセージの発送が可能となる。もし目的地ＩＭＰがメッセージを受けとるバッファに余裕がない場合，到着したそのメッセージは捨てられる。その際目的地ＩＭＰではこのメッセージをバッファ予約要求とみなし，その要求をキューイングしておき，バッファが空くとRetransmitパケットを発信地ＩＭＰへ送り，再送を要求する。これを受け取った発信地ＩＭＰはメッセージを再送する。このため，発信地ＩＭＰには単一パケット・メッセージのコピーがあり，Receiveを受け取った時にそのコピーを消去する。目的地ＩＭＰはパケットからメッセージを再組立てし，受信ＨＯＳＴへ送る。メッセージを受け取った受信ＨＯＳＴは直ちにReceiveを目的地ＩＭＰへ渡す。Receiveは目的地ＩＭＰから発信地ＩＭＰを経由して送信ＨＯＳＴへ運ばれる。

一方，ＨＯＳＴレベルでもコネクションに対し伝送制御が行なわれており，送信ＨＯＳＴはそのコネクションに割り当てられた受信側のメッセージ・バッファの個数が０の場合は，メッセージの送信を一時中断する。受信ＨＯＳＴはそのコネクションに対し新たにメッセージ・バッファが割り当てられた時，ＡＬＬＯＣコントロール・コマンドを送信ＨＯＳＴへ送る。それを受け取った送信

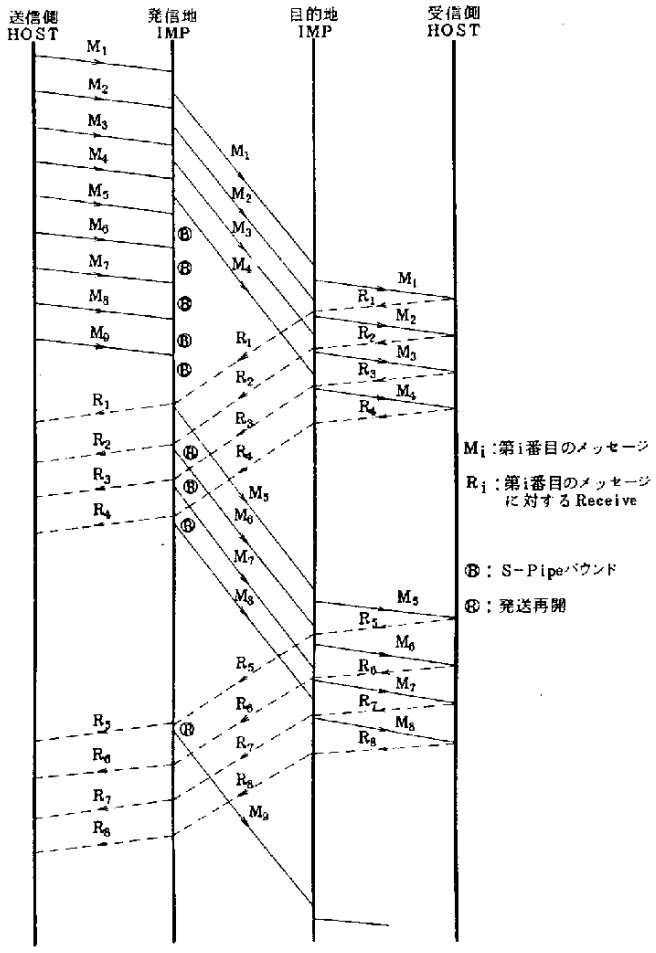



図５－２　ネットワーク・レベルの単一パケット・メッセージ伝送制御手順

ＨＯＳＴはメッセージの送信を再開する。また送信したメッセージに対しReceiveが返送されるまでは，送信ＨＯＳＴはプロセスからのそのコネクションに対するメッセージの送信要求をブロックする。

### 5.4.2　多重パケット・メッセージの伝送制御手順

図５－３に多重パケット・メッセージに対するＨＯＳＴレベル及びサブネット・レベルのメッセージ伝送制御手順を示す。

送信ＨＯＳＴはまずメッセージ・リーダのみを発信地ＩＭＰへ送る。発信地ＩＭＰはまず最初に目的地ＩＭＰへのＬ－Pipe（１本）が空いているかを調べる。もし，空いていれば，次に目的地ＩＭＰに既に８パケット分のリアセンブリ・バッファが確保されているかどうかを調べる。もし確保されていなければ，その８パケット分のバッファを確保するため，ＲＦＡ（Request For

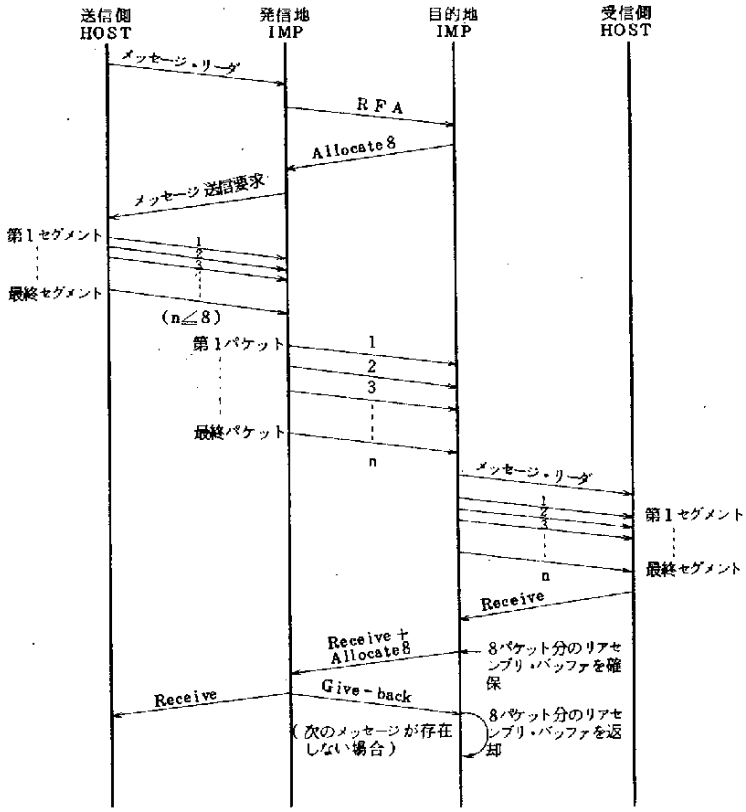



図５－３　ネットワーク・レベルの多重パケット・メッセージ伝送制御手順

Allocation）パケットを発信地ＩＭＰから目的地ＩＭＰへ送る。これを受けた目的地ＩＭＰはできるだけ速やかに８パケット分のリアセンブリ・バッファを確保し，Allocate-8 パケットを発信地ＩＭＰへ送る。以上でＬ－Pipe と目的地ＩＭＰに８パケット分のリアセンブリ・バッファが確保されたので，発信地ＩＭＰは送信ＨＯＳＴへメッセージの送信要求信号を送る。これを受けた送信ＨＯＳＴは直ちにセグメント単位でメッセージを送り出す。発信地ＩＭＰはメッセージを複数個のパケットに分解し，目的地ＩＭＰへ向けて送り出す。

一方，目的地ＩＭＰではそのメッセージを構成するすべてのパケットを受け取ると，メッセージをリアセンブリして，セグメント単位で受信ＨＯＳＴへ送る。メッセージを受け取った受信ＨＯＳＴは直ちにReceive を目的地ＩＭＰへ送る。Receive を受け取った目的地ＩＭＰはそのパイプを通る次のメッセージのために８パケット分のリアセンブリ・バッファを確保してから，Allocate-８をReceive に便乗させ，Receive パケットを発信地ＩＭＰへ返送する。これを受け取った発信地ＩＭＰはそのReceive を送信ＨＯＳＴへ送る。また同一目的地行きの多重パケット・メッセージが待っていない場合は，直ちにそのバッファをキャンセルするため，Give－back パケットを目的地ＩＭＰへ送る。Give－backパケットを受け取った目的地ＩＭＰは既に確保した８パケット分

のリアセンブリ・バッファを返却する。

## 5.5 ネットワーク・パフォーマンスの制約要因

ＨＯＳＴ間で授受されるデータに対するパフォーマンスの制約要因について検討する。

### 5.5.1 ハードウェアの制約要因

回線容量の制約はネットワーク・パフォーマンスに直接影響する。ＨＯＳＴは１メガ・ビット／秒の半二重アダプタでＩＭＰに接続されており，スループットはこの１メガ・ビット／秒を越えることができない。一方，ＩＭＰ間は48キロ・ビット／秒の回線で各々接続されているので，Adaptiveルーティングによるトラフィックの分流がなければ，スループットはこの48キロ・ビット／秒に抑えられてしまう。

通常のトラフィック量ではＩＭＰの処理能力（bandwidth）は制約要因とはならないが，ＨＯＳＴの処理能力は数百キロ・ビット／秒であり，パフォーマンス低下の主要な要因となる。

ＨＯＳＴとＩＭＰ間は両コンピュータのデータ・チャネルを半二重のインタフェス・アダプタにより接続されている。アダプタは半二重であるが，両系に対し特に制御局／従属局という関係を設けず，コンテンション方式を採用している。コンテンション方式であるので，双方が同時にＷＤＩ（Write Data with Interrupt）コマンドを発信した場合，コマンドの衝突が起る。この場合，ＪＩＰＮＥＴのＨＯＳＴ－ＩＭＰプロトコルでは，ＩＭＰ側が送信権を持ち，再度ＷＤＩコマンドを発し，一方ＨＯＳＴ側は受信状態に入り，ＩＭＰ側よりのAttentionを受け付けてからＲＤ（Read）コマンドを発信し　実際のデータ転送が行なわれる。両方向同時通信ではもちろんのこと単方向通信でもコネクションが２本以上になると，アダプタ上でメッセージ・リーダとReceiveが衝突する可能性があり，トラフィックの増加と共に衝突頻度が極度に増加する。例えば３本のコネクションでは３メッセージに１回の割合で衝突が起る。コマンドの衝突に対する処置に費やすＨＯＳＴ側の処理時間は数ミリ秒にもなり，パフォーマンスの著しい低下の主要な要因となる。ＪＩＰＮＥＴのような非同期なデータ転送を行う同格のＣＰＵ間を接続するには，半二重ではプロトコルが非常に複雑となり，両系のオーバヘッドが増大するので，ハードウェア上は全二重にすべきである。

### 5.5.2 フロー制御の制約要因

A．コネクション

送信ＨＯＳＴのＮＣＰはコネクションを通して送信したメッセージに対し受信ＨＯＳＴからそのメッセージに対するReceiveが返ってくるまでは，そのコネクションを通る次のメッセージの送信を見合せている。この方式はストップ・アンド・ウェイト（stop and wait）方式と呼

ばれる。ストップ・アンド・ウェイト方式の実効スループット率は送信ＨＯＳＴと受信ＨＯＳＴ間に存在する中継ＩＭＰの台数に依存する。従って，実効スループット率$\mu$は，

$$\mu = \frac{Tm(1-Pe)}{Tm + Ta}$$

となる。ここで，Tmはメッセージを送信プロセスから受信ＮＣＰへ運ぶに必要な伝送遅延時間，Taはそのメッセージに対してのReceive（End-to-End Acknowledgmentに相当）を受信ＮＣＰから送信プロセスへ運ぶに必要な伝送遅延時間，そしてPeはメッセージが再送要求される割合である。明らかにTmはTaに比して大きくとるべきである。しかし，Tmが大きいことは中継ＩＭＰの段数を一定とすれば，即ちメッセージ長（パケット長）を大きくすることであり，メッセージ長を大きくしすぎると，メッセージ誤り率Peが大きくなり，再送要求の頻度が高くなり，実効スループット率が減少する。

また，平均ラウンド・トリップ遅延（mean round-trip delay）Trは，

$$Tr = (Tm + Ta)(1 + Pe)$$

となる。

一方，コネクションを通してのメッセージ伝送では，受信側ＨＯＳＴにバッファが用意されていない場合は，送信を一時中断する。ＴＩＰなどのバッファ容量の少ないＨＯＳＴでは１メッセージ毎に，ＨＯＳＴ－ＨＯＳＴプロトコルのＡＬＬＯＣコントロール・コマンドを発信している。受信ＨＯＳＴからのＡＬＬＯＣが到着するまで，送信ＨＯＳＴがユーザ・メッセージの送出を待つ場合，実効スループットは著しく減少する。バッファ量の少ないＨＯＳＴと交信しているＨＯＳＴはしばしばＡＬＬＯＣを受信しなければならないので，ＮＣＰのオーバヘッドが増大する。

B．パイプ

前述したように，サブネット内では発信地ＩＭＰと目的地ＩＭＰ間で論理的なパスであるパイプを設定し，このパイプの容量を制限してフローを抑制している。

メッセージを送信するためには，発信地ＩＭＰは空きパイプ１本と目的地ＩＭＰにバッファを確保しなければならない。これらのリソースは有限であり，他のＨＯＳＴと共用しなければならない。これらのリソースの欠如は単一ＨＯＳＴのスループットの低下をもたらす。

ＩＭＰには72個分のパケット・バッファしかなく，回線当り１個，蓄積転送（store-and-forward）用に１個，リアセンブリ用に８個が固定的に割り当てられ，のこりのパケット・バッファはそれぞれの目的に共用されている。

パイプとしては，すべての発信地ＩＭＰと目的地ＩＭＰ間の対に対しＳ－Pipe用に４本とＬ－Pipe用に１本しか設けていない。あるＩＭＰに接続されているすべての送信ＨＯＳＴは，任意のあるＩＭＰに接続されているすべての受信ＨＯＳＴとの間でこれらのパイプを共用しなければならない。送信ＨＯＳＴと受信ＨＯＳＴが数hop以上離れている場合，パイプは数百ミリ秒以

上空き状態にならないので，ＨＯＳＴ間の相互干渉が問題となり，スループット低下の著しい要因となる。他のＨＯＳＴの干渉がなくとも，ＨＯＳＴは同じ目的地ＩＭＰへ単一パケット・メッセージに対しては４個，多重パケット・メッセージに対しては１個しか同時に発送することはできない。この制約はラウンド・トリップ遅延が大きい場合に問題となる。理想的には，このパイプの大きさは，発信地ＩＭＰと目的地ＩＭＰ間のhop数と，ネットワーク全体のトラフィック量に従って動的に変更すべきである。パイプの容量は回線伝送速度とラウンド・トリップ時間の積がメッセージ長とパイプ容量の積よりも大きい場合にはスループット低下の第一要因となる。

ラウンド・トリップ遅延はAdaptive ルーティングによって減少し，同時にスループットは増大する。数hop離れたＨＯＳＴ間でのスループットはAdaptiveルーティングにより迂回ルートがある場合30％位増加する[10]。

### 5.5.3　ルーティングの制約要因

ルーティングの手法には種々のものが，ＪＩＰＮＥＴではパケットの高速伝送，トラフィック変動に対する適応性，トポロジ変化への追従性，ＩＭＰの計算負荷の軽減などを考慮して，「Shortest Queue ＋ Bias」の手法を採用している。この手法は定常状態では外部からは情報をとり入れずに，ＩＭＰ内部における動的な送信待ちキューの長さと，ネットワークの静的な構造を示すバイアス値との和を各出力回線ごとに比較し，その最小のものを選びパケットを送り出す方法である。バイアス値は，ＩＭＰおよび回線のアップ・ダウンにともなうサブネットの重大な変化の際のみ動的に計算し，更新している。

通常バイアス値は，目的地までの中継ＩＭＰが無負荷状態であることを前提として決定されるが，ＪＩＰＮＥＴでは各中継ＩＭＰに各出力回線毎に出力キューに１個のパケットが並んでいると仮定して，バイアス値を決めており，バイアス項に若干多くの荷重を与えている。即ちバイアス値の荷重の方が出力キューの長さの荷重よりも大きいので，固定ルーティングに近くなっている。表５－２に１hopに対する単位のバイアス値を示す。

表５－２　単位バイアス値と回線速度の関係

| 回線速度（キロ・ビット／秒） | 単位バイアス値 |
|---|---|
| 48 | 1 |
| 9.6 | 5 |
| 4.8 | 10 |
| 2.4 | 20 |

発信地から目的地までの最短ルートのhop数が迂回ルートのそれよりも数hop以上小さければ，トラフィックの大部分は最短ルートを通ることになり，迂回ルートはめったに使用されない。迂回ルートの存在はネットワーク・パフォーマンスの向上よりもむしろネットワークの信頼性の面で重要である。

## 5.6 ネットワーク・パフォーマンスの測定・評価

### 5.6.1 ファイル転送の測定・評価

1ブロックが576バイトのファイルを1hop離れた2つのHOST間で転送した場合の測定によると，JIPNETを使って無負荷状態でFACOM 230-75からHITAC 8450へのスループットは25.7キロ・ビット／秒，ラウンド・トリップ・タイム（RTT，round-trip time）は179ミリ秒である。F-H間では別に2hopの迂回ルートもあるので，それも併用するとスループットは30.1キロ・ビット／秒に増加し，RTTは153ミリ秒に減少し，サブネットのadaptiveルーティングの効果が現われている。

同様な実験[1]が1970年ごろのARPANET Version-1を使ってSRIのXDS 940からUtah大学のPDP-10で行なわれており，スループットは26.3キロ・ビット／秒，RTTは175ミリ秒が測定されている。その後IMPが改善されており，現在測定すればスループットは30.3キロ・ビット／秒，RTTは152ミリ秒になることが推定される[2]。

2つのネットワークの測定結果を比較するとほぼ等しくなっている。これは両ネットワークとも同じ規模の回線，IMPコンピュータそしてHOSTコンピュータを使っているためで，プロトルの差異は影響していないことがわかる。

### 5.6.2 最大スループットの測定・評価

図5－4のような迂回ルートのあるトポロジで2つのHOST間で2つの異なったメッセージ長に対してコネクションの本数を変化させて最大パフォーマンスを測定した（図5－5参照）。

第1のケースは，パケット・オーバヘッドが最小になるようにメッセージ長を最大パケット長（フル・パケット）の1152ビットにした。そのようなメッセージに対する最大スループットはルート$R_1$のみを使用した場合は約43キロ・ビット／秒であり，activeコネクションが3で飽和値に達している。

これは，サブネットのIMPが48キロ・ビット／秒の回線で結合されており，パケット・オーバヘッド（128ビット）を除いた実効伝送帯域が約43キロ・ビット／秒であるためで，回線バウンドが原因となっており，この値以上のスループットは実現されない。

ルート$R_1$と迂回ルート$R_2$を併用した場合の最大スループットは約50キロ・ビット／秒に上が

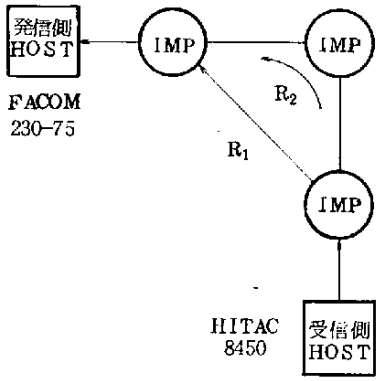
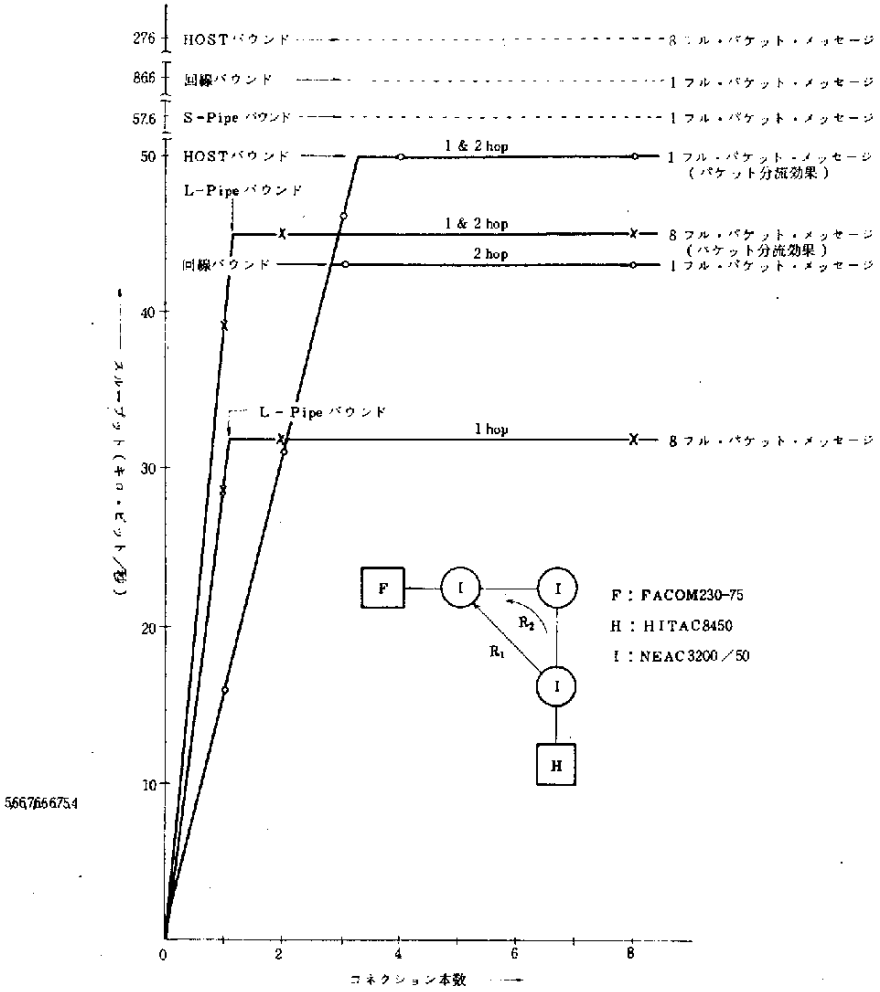



図５－４　ネットワーク・トポロジ




5667666754

図５－５　最大スループットの測定

る。

これはサブネットのadaptiveルーティングの効果である。Adaptiveルーティングとしては Shortest Queue＋Biasを採用しており，最短ルートの出力回線の送信待ちキューが長くなると迂回ルートの出力回線が選択される。Adaptiveルーティングによってサブネット内だけの最大スループットは57キロ・ビット／秒であり，ＨＯＳＴ間でもＲＴＴが減少するが，ＨＯＳＴ側のバウンドのためにＨＯＳＴを含めた最大スループットは50キロ・ビット／秒にしかならない。

サブネットのフロー・コントロールは，発信地ＩＭＰと目的地ＩＭＰ間にパイプ（ pipe ）という論理的なパスを張り，そのパイプの容量を制限して輻輳を防止している。サブネット内ではメッセージを単一パケット・メッセージ（ 1152 ビット以下）と多重パケット・メッセージ（ 9216 ビット以下）とに分け，単一パケット用のパイプ（ Ｓ－Pipe ）を４本と多重パケット用のパイプ（ Ｌ－Pipe ）を１本設けている。各パイプはコンカレントにメッセージを流すことができ，多重パケット・メッセージは目的地ＩＭＰにバッファを確保した後，Ｌ－Pipe を通し，一時に１個だけ流すことができる。

Ｓ－Pipe は４本であるので，５個目の１パケット・メッセージは出力回線が空いていても，Ｓ－Pipe が空くまで送信が見合わせられる。このような現象はＳ－Pipe バウンドと呼ばれる。例えば$R_1$と$R_2$の２本の回線を併用した場合の最大スループットは86キロ・ビット／秒になるはずであるが，Ｓ－Pipe バウンドにより57キロ・ビット／秒しか出せない。

第２のケースは，メッセージを最大メッセージ長（８フル・パケット）の9216ビットにした。そのようなメッセージの最大スループットはルート$R_1$のみを使用した場合は約32キロ・ビット／秒であり，非常に効率が悪い。これはパイプ・バウンドであり，Ｌ－Pipe が１本しかないため，別のコネクションのメッセージはＬ－Pipe が空くまで送信が待たされるためである。ルート$R_1$と迂回ルート$R_2$を併用した場合の最大スループットは約45キロ・ビット／秒に増加する。これはメッセージの一部（パケット）が迂回ルートを通り，全体としてＲＴＴを減少させており，L－Pipeが早く空き状態になるためである。

ロング・メッセージの最大スループットは図５－６に示す1970年ごろのARPANET Version－1の約70キロ・ビット／秒に比べると非常に悪くなっている。ARPANET Version－1ではロング・メッセージを特別扱いせず，リンク毎にフロー・コントロールを行っていた。しかし特定のＨＯＳＴへのactiveリンクが５本を越えるとreassemble lockupが起りうることが判明して，その後フロー・コントロールが強化されたため[2]，Version－2（1974年ごろ）では約38キロ・ビット／秒，現在のVersion－3（1975年１月から）では約37キロ・ビット／秒である[10]。

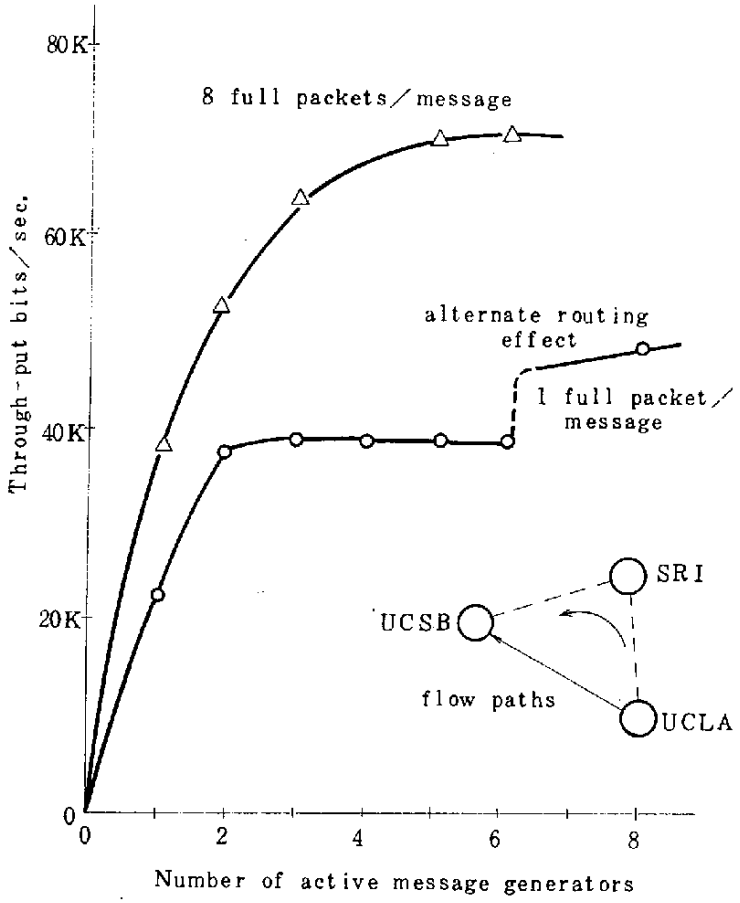



図5－6　Through－put Measurements Between UCLA and the Neighboring UCSB IMP

### 5.6.3　ラウンド・トリップ・タイムの測定・評価

A．単一パケット・メッセージ

単一パケット・メッセージのメッセージ長を変化させてＲＴＴを測定した結果を図５－７に示す。

図５－７からＲＴＴの実験式Tr（h，b）を求めると

$$Tr(1, b) = 42 + 0.18b \quad (1)$$

$$Tr(2, b) = 50 + 0.34b \quad (2)$$

である。ここでhはhop数，bはメッセージに含まれるバイト数である。

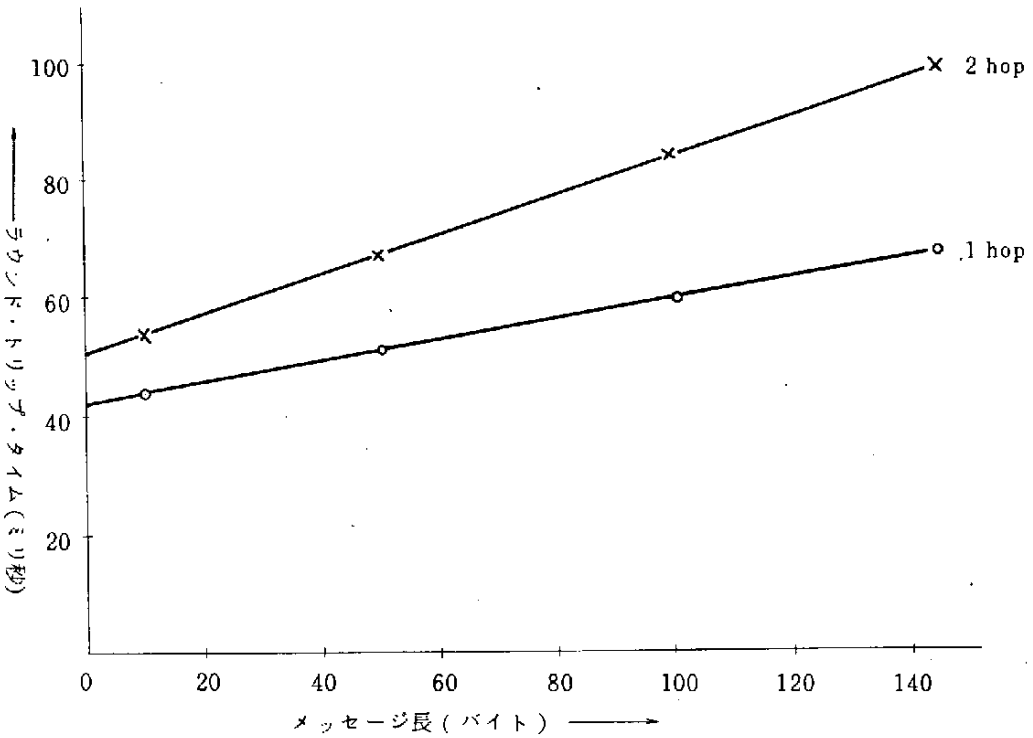



図５－７　最小ラウンド・トリップ・タイムの測定
（１パケット・メッセージの場合）

単一パケット・メッセージのＲＴＴの理論式は以下のような要素からなる。

| | | |
|---|---|---|
| 両ＨＯＳＴのオーバヘッド | 32 | ミリ秒 |
| ＩＭＰ間のパケット伝送遅延時間 | 0.18×(b + 16) | ミリ秒 |
| ＩＭＰ間のＡＣＫパケット処理・伝送遅延時間 | 0.18×12 | ミリ秒 |
| ＩＭＰ間の Receive パケット伝送遅延時間 | 0.18×14 | ミリ秒 |
| ＩＭＰ間のＡＣＫパケット処理・伝送遅延時間 | 0.18×12 | ミリ秒 |

これらにはキューイングによる待ち時間は含まれていない。以上から理論式 Tr(h, b) は,

$$Tr(h,b) = 32 + 0.18\,(b + 16 + 12 + 14 + 12)\,h$$
$$= 32 + 0.18\,(b + 54)\,h \qquad (3)$$

となる。

h = 1の場合，理論式 Tr(1, b) は,

$$Tr(1,b) = 42 + 0.18\,b \qquad (4)$$

となり，h = 2の場合，理論式 Tr(2, b) は,

$$Tr(2,b) = 51 + 0.36\,b \qquad (5)$$

となり，実験で求めた最小ＲＴＴの式(1),(2)と非常によく一致しており，理論式で実際のＲＴＴを推定することが可能である。

ARPANETの1 hopの理論式 Tr(1, b)は,

$$Tr(1, b) = 20.0 + 0.24b \quad (6)$$

である[1]。

式(4)と式(6)の差異は，JIPNETではHOST－IMP間の伝送速度は高速（1メガ・ビット／秒）であり，HOST－IMP間のメッセージの伝送遅延時間は無視できる位小さい値であるので無視しているが，一方ARPANETではこれを無視することができず(100キロ・ビット／秒)，この値も考慮しているが，IMP間でのパケットの伝送遅延時間をメッセージ長に依存しない一定の値(7.5ミリ秒)を使用していることによる。

B．多重パケット・メッセージ

メッセージの長さを最大パケット長（1152ビット）の倍長で変化させてRTTを測定した結果を図5－8に示す。

4パケット・メーセージの場合RTTは1 hopでは179ミリ秒，2 hopでは219ミリ秒である。1 hopの場合，2 hopの迂回ルートを併用すると153ミリ秒に減少する。この153ミリ秒は1 hopルートへ3パケットを流した場合（150ミリ秒）にほぼ一致しており，このことから1 hopルートへ3パケット，2 hopルートへ1パケットを振り分けたことになる。以下同様に推定すると表5－3と図5－9に示すようにパケットが分流されたことになる。

図5－8からRTTの実験式 Tr(h, p)を求めると,

＜1 hopの場合＞　$Tr(1, p) = 60 + 30(p - 1) + 30$ (7)

＜2 hopの場合＞　$Tr(2, p) = 100 + 30(p - 1) + 30$ (8)

となる。ここでhはhop数，pはメッセージに含まれるパケット数である。各々の式の最初の定数項は先頭パケットとRFAなどのサブネット内のフロー伝送制御パケットが目的地IMPへ到着するのに必要な時間の和であり，hop数の関数である。p－1の係数30は1フル・パケットが1 hopだけ前進するのに必要な時間であり，30(p－1)は最終パケットが先頭パケットよりもどの位遅れて目的地IMPへ到着するかを表わしている。最後の定数項30は両HOSTの送受信オーバヘッドであり，HOSTの処理能力に依存する。

Hop毎に一度にメッセージ全体を伝送すると伝送遅延時間はp×hに比例することとなるが，メッセージを小さな伝送単位のパケットに分解することによってh+(p－1)に比例することとなり，伝送遅延が減少する。この効果はパイプライン(pipeline)と呼ばれる。

以上から次のRTTの近似式 Tr(h, p)が導き出される。

$$Tr(h, p) \fallingdotseq 60h + 30(p-1) + 30 \quad (9)$$

p＝7（8064ビット）の場合は,

$$Tr(h, 7) = 60h + 210$$

であり，ARPANETのフル・メッセージ（8063ビット）の場合の式[8]

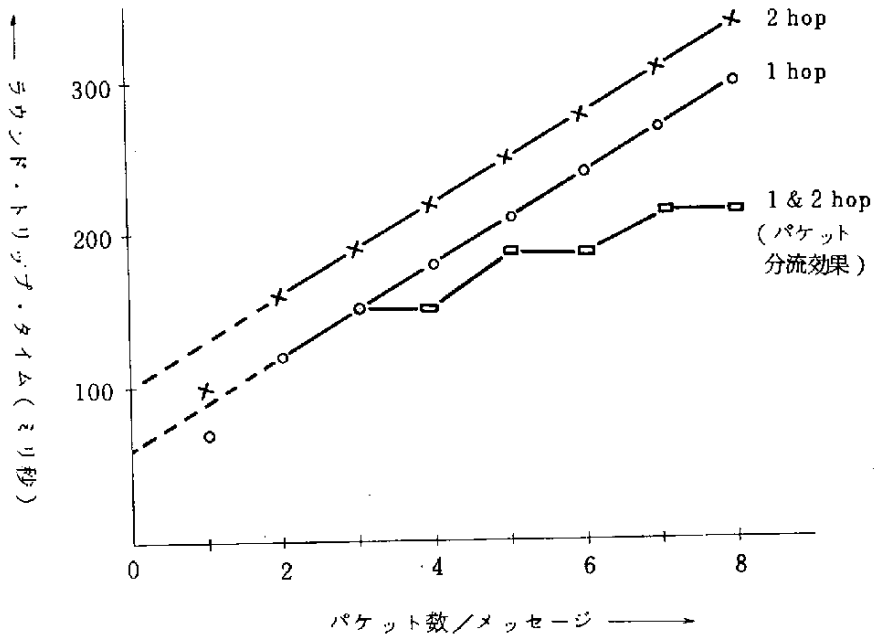



図５－８　最小ラウンド・トリップ・タイムの測定

表５－３　パケット分流

| 選択ルート / パケット数 | 最短ルート（1－hop） | 迂回ルート（2－hop） |
|---|---|---|
| 2 | 2 ($P_1$, $P_2$) | 0 |
| 3 | 3 ($P_1$, $P_2$, $P_3$) | 0 |
| 4 | 3 ($P_1$, $P_2$, $P_3$) | 1 ($P_4$) |
| 5 | 4 ($P_1$, $P_2$, $P_3$, $P_5$) | 1 ($P_4$) |
| 6 | 4 ($P_1$, $P_2$, $P_3$, $P_5$) | 2 ($P_4$, $P_6$) |
| 7 | 5 ($P_1$, $P_2$, $P_3$, $P_5$, $P_7$) | 2 ($P_4$, $P_6$) |
| 8 | 5 ($P_1$, $P_2$, $P_3$, $P_5$, $P_7$) | 3 ($P_4$, $P_6$, $P_8$) |

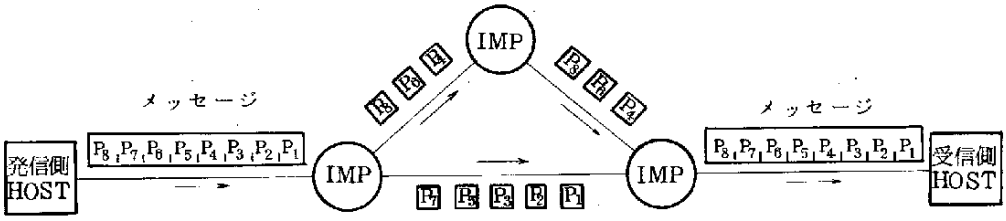



図５－９　パケットの分流（８パケット・メッセージの場合）

$$Tr(h,8) = 60h + 200 \qquad (10)$$

とほぼ一致する。

## 第５章の参考文献

(1) Cole, G. D.
Performance measurements on the ARPA computer network.
Proceedings of the Second ACM IEEE Symposium on Problems in the Optimization of Data Communications Systems, Oct. 1971, pp. 39-45

(2) McQuillan, J. M., Crowther, W. R., Cosell, B. P., Walden, D. C., and Heart, F. E.
Improvements in the design and performance of the ARPA network.
AFIPS Conference Proceedings, Vol. 41, Dec. 1972, pp. 741-754

(3) McQuillan, J. M.
Throughput in the ARPA network — Analysis and measurement.
BBN Report No. 2491, Jan. 1973, 34p.

(4) Cerf, V.
An assessment of ARPANET protocols. RFC 635, NIC 30489, April 1974, 21p.

(5) Kleinrock, L. and Naylor, W.
On measured behavior of the ARPA network.
AFIPS Conference Proceedings, Vol. 43, May 1974, pp. 767-780

(6) Opderbeck, H. and Kleinrock, L.
The influence of control procedures on the performance of packet-switched networks.
Proc. National Telecommunications Conference, Dec. 1974, 8p.

(7) Crowther, W. R., Heart, F. E., McKengie, A. A., McQuillan, J. M., and Walden, D. C.

Issues in packet switching network design.

AFIPS Conference Proceedings, Vol. 44, May 1975, pp. 161-175

(8) Wood, D. C. and Trehan, R. K.

An assessment of the performance of packet switching networks.

Proc. European Computing Conference on Communications Networks, Sept. 1975, pp. 1-11

(9) Irland, M.

Simulation of CIGALE 1974.

Proc. 4th Data Communications Symposium, Oct. 1975, pp. 5-13-19

(10) Kleinrock, L.

Throughput in the ARPANET — Protocols and measurement.

Proc. 4th Data Communications Symposium, Oct. 1975, pp. 6-1-11

(11) Wood, D. C.

Measurement of user traffic characteristics on ARPANET.

Proc. 4th Data Communications Symposium, Oct. 1975, pp. 9-2-2

(12) Kleinrock, L., Naylor, W. E., and Opderbeck, H.

A study of line overhead in the ARPANET.

Communications of the ACM, Vol. 19, No. 1, Jan. 1976, pp. 3-13

# 6. 仮想ネットワーク

# 6. 仮想ネットワーク

## 6.1 仮想ネットワークの概念

コンピュータ・ネットワークの究極の目標は，ユーザがネットワークの全リソースを，一つのコンピューティング・システムとしてみなすことができるようにすることだと言われている。この場合，ユーザはＨＯＳＴ特有のオペレーティング・システムや，ネットワークの多様なリソースにアクセスする為の各種プロトコルなどに習熟する必要はなく，要求したリソースの存在場所すら知る必要がない。

すなわち，ユーザにとってＨＯＳＴマシンは完全にトランスペアレントであり，ユーザはHOST特有のジョブ制御文やリソースへのアクセス法を知らなくとも，また特定の処理ＨＯＳＴを指定しなくとも，要求する処理を標準的な指示によって入力すると，最適のＨＯＳＴまたはＨＯＳＴ群が割当てられ，処理が実行され，結果が戻ってくるという様な高度なユーティリティが真のコンピュータ・ネットワークの姿だと言われている。このような概念を，仮想ネットワークと呼んでいる。

仮想ネットワークの概念の中には，次のような要素が含まれる。

① 各ＨＯＳＴのハード・ソフトの各種リソースの種類・能力等のスタティックな情報を一元的に収集・管理する機能。

② 各ＨＯＳＴの現在の負荷状態やリソースの使用状況等のダイナミックな情報を管理する機能。

③ 要求された処理内容を解釈し，どのようなリソースが必要であるかを判断する機能。

④ ①，②，③の情報を総合し，その時点で最適のＨＯＳＴまたはリソースを割当てる機能。

⑤ ユーザからネットワークへの標準的なネットワーク・アクセス言語またはハイレベルなネットワーク・アクセス・プロトコルの設定。

上記①～④で述べたような機能は，ＨＯＳＴ相互間の負荷の分担や，障害時のバック・アップを目的とするものであり，一般にロード・レベリングあるいはロード・シェアリング等と呼ばれている。同一のＨＯＳＴマシンからなる，いわゆる均質のネットワークにおいては，このロード・レベリングが目的の一つとなり，ある程度，実用化されている例もある。しかし異機種ネットワークにおいては，ロード・レベリングはもちろんのこと，その為に必要となる異機種間のプログラムおよび，データのシェアリングといった基本的な問題すら解決し切ってはおらず（文献１）さし当っての目標は，ユーザの操作性やネットワーク利用の容易性の向上といった面に向いている。

このような仮想ネットワークを実現する為の方法としては，現在までに様々のアプローチがとられている。

まず第1は，ＢＢＮによって開発されたARPANETにおけるRSEXECのように（文献２，４，５），既存のコンピュータ・ネットワーク・ファシリティの上に，同一機種（ＰＤＰ10），同一オペレーティング・システム（TENEX）からなるＨＯＳＴグループを構成し，それらの各HOSTにユーザ・レベルのネットワーク・ジョブ管理プロセス（RSEXEC）を置き，ユーザからみたＴＥＮＥＸ ＨＯＳＴグループのＨＯＳＴバウンダリを取り除こうとする試みである。RSEXECでは，ユーザ・ジョブはどのＨＯＳＴでも実行でき（自動選択ではないが），ネットワーク・ワイドなファイル管理システムを利用することが可能である。現在ＴＩＰからみたRSEXECは，ある程度仮想ネットワークとしての機能を果しており，今後は，ＨＯＳＴ群がますます密接に結合され，多重ＨＯＳＴとしての性格を強めて行くだろうと言われている。

第２のアプローチは，カリフォルニア大学・アービンのＤＣＳ，メリーランド大学のＤＣＮのように，設計の当初から仮想ネットワークを目指したネットワーク・オペレーティング・システムを開発しようとしているものである。（文献２）ＤＣＮでは，ネットワーク・ワイドなヴァーチャル・メモリを実現し，リソースの割当，ファイル・ＣＰＵの場所などは，ユーザには見えず，最適なリソースをシステムが決定するようになっている。これらのシステムは，実験的なものではあるが，ロード・レベリングまで目指しているという事で，注目されている。

第３のアプローチは，ＡＲＰＡにおけるＮＢＳのＮＡＭ（文献６），MITRE社のＲＥＸ（文献２，３）にみられるような，異機種コンピュータ・ネットワークにおけるユーザの操作性やネットワーク使用の容易性を高めようとする試みである。ＮＡＭは，ユーザとネットワークの間にミニコンによるネットワークアクセス手続標準化機構を置き，各種のシステム呼出しの違いによるわずらわしさからユーザを解放しようとするものである。また，ＲＥＸは，各ＨＯＳＴの各種リソースに関する情報（概要，手続，コスト等）を一元管理し，ユーザに情報サービスを行うとともに，各ＨＯＳＴにおける各リソースのガイダンス機能に対しては，自動的にそのリソースを獲得し，ユーザをそのリソースのＨＥＬＰ機能に接続してくれるものである。このような，既存ネットワークにおけるユーザへのサービス性を向上するには，ＮＡＭやＲＥＸのようなアプローチが現実的であり，また，最も望まれるものであろう。

第４のアプローチとしては，ネットワーク・コマンド言語を規定し，既存のオペレーティング・システムのジョブ制御言語の特殊性をユーザに意識させず，ネットワークのどの地点からでも同一の方法でアクセスできるようにする試みがある。このアプローチとしては，ＩＢＭ／440ネットワーク（文献２）やフランスのＳＯＣネットワーク（文献９）などのように均質ネットワークにおいては，ある程度実験的運用がなされているが，異機種ネットワークも包含するような完全な意味での汎用ネットワーク・コマンド言語（文献10）については，まだ，研究段階である。

第５のアプローチとして，Logical Network Machineの考え方（文献15）がある。これは，CYCLADES の一環としてグルノーブル大学で行われている研究であり，ネットワーク・レベル

で遂行される機能に対する各種コマンドを各機種共通な中間言語によって設定し，その解釈と制御を各サイトのサブシステムが協調して行うような論理的なネットワーク・マシンを実現しようとするものである。このＬＮＭは，ユーザから見た場合，異機種のコンピュータ・ネットワークが，単一のコンピューティング・システムとして見えるような，仮想ネットワーク機能を実現しようとしている。

第６のアプローチとしては，今まで述べたようなユーザからみたＨＯＳＴバウンダリがとり除かれたと仮定して，（均質ネットワークのもとで，あるいは異機種ネットワークにおけるＬＮＭのような機能が実現したとして）ロード・レベリングを行う為に，ネットワーク・ジョブの処理ＨＯＳＴの自動選択およびスケジューリングを行おうとする考えがある（文献17, 18, 19 ）。このようなアプローチは，現状では，まだ研究段階であり，解析モデルを用いた数学的アプローチが２，３試みられているに過ぎない。

以上に述べたような仮想ネットワークに対する各種のアプローチは，各所において色々のレベルで試みられているが，これらのアプローチは，その一つが実現すればそれで完結という性質のものではなく，このようなアプローチを通して，仮想ネットワーク実現に至る問題点を明確にし，解決の糸口を見出し，それらの集大成により真のコンピュータ・ネットワークとしての仮想ネットワークが実現されていくというものであろう。

## 6.2　仮想ネットワークの技術動向

本節では，仮想ネットワークの技術動向を，いくつかに分類し，各分野における代表的な論文を引用しながらその動向を把握したい。

まず，6.2.1においてコンピュータ・ネットワークにおけるリソース・シェアリング（リソースの共同利用）という観点からユーザに対するネットワーク機能のサービスとして，ロード・レベリング，プログラム・シェアリング，データ・シェアリング等についての問題点をM. Schneiderの論文を引用し列挙してみる。

次に6.2.2において，仮想ネットワークの概念を非常にうまく表現しているH. S. Elovitz の "What is a Computer Network "という論文の抜すいによりその概念を紹介する。彼は，"コンピュータ・コミュニケーション・ネットワーク"と"コンピュータ・ネットワーク"を明確に区別し，その違いを，ユーザからみたネットワークの透明度のレベルによって分類し，それぞれの例を示している。その分類にしたがうと，彼の言う"コンピュータ・ネットワーク"すなわち仮想ネットワークには，現状では，ＲSEXEC，ＤＣＮ，ＤＣＳぐらいしか値しないと述べている。

必に6.2.3では，異機種ネットワークにおける，ユーザの操作性の向上，利用の容易性の向上という観点から，ARPANETにおけるＮＢＳのＮＡＭ，ＭＩＴＲＥ社のＲＥＸについて，その概要を紹

介する。

次に6.2.4では，ネットワーク・コマンド言語の状況について，まず，ネットワーク・コマンド言語というものの考え方を示し，次に，ＳＯＣネットワークにおけるある程度均質なネットワークにおけるネットワーク・コマンド言語の実例を紹介し，次に，Chupinの論文により，異機種ネットワークにおけるネットワーク・コマンド言語実現にあたって考慮しなければならないポイントを整理した。

次に6.2.5では，論理的ネットワーク・マシンの概念を同じくChupinの論文により紹介した。異機種コンピュータのネットワークにおいて，仮想ネットワークを実現しようとする場合，この考え方は，非常に重要であり，この考え方の実験システムとしての分散型データ・ベース管理システムであるCYCLADESネットワークのSOCRATEシステムの実験成果と今後の動向が注目される。

最後に，6.2.6では，仮想ネットワーク実現の為の最後の難関になると思われる，ネットワーク・ジョブの自動スケジューリングについてW. D. Roomeの論文により，その考え方を示す。この論文で示されているのは，ネットワーク・ジョブのポータビリティが解決されたものとして，ロード・レベリングの効果を解析モデルによって示したものである。この分野における他の論文も，まだ，これと似たような研究段階であり，仮想ネットワークにおける重要な実験目標とは成り得ていない。この問題をとりまく各種の問題解決を図りながら今後とも研究・実験が必要な分野であろう。

### 6.2.1 リソース・シェアリングにおける問題（文献1より抜すい）

(1) ロード・レベリング

ロード・レベリング（Load Levelling）とは，プログラムおよびそれに関連するデータを，ネットワーク内の他のノードに転送し，ＨＯＳＴマシンの計算負荷を平滑化しようとするものである。地理的に散在する多数のノードからなるネットワークの場合，各ノードの負荷は一様ではなく，例えば，ARPANETにおけるＩＬＬＩＡＣ－Ⅳのように，非常に魅力的なリソースを有しているノードには，大量のトラフィックが発生するであろう。また，タイム・ゾーンの違いにより，あるタイム・ゾーンのＨＯＳＴはフル稼動しているが，夜間のタイム・ゾーンのＨＯＳＴは非常に空いているという状態も考えられる。

このようなノード負荷の不平衡に対し，ロード・レベリングの効果は明らかに認められるであろうが，この問題はネットワークにおいて，その設計の主要な目標とはなり得ていない。

現状では，異機種間のロード・レベリングを行うためには大きな問題があり，労多くして益少なしと言われており，計算負荷の共用という場合は，暗に同機種のネットワークの事を指している例が多い。従って，互換性を有さないような異機種のネットワークでは，大低，この問題を考慮対象から省いている。しかし，同機種のネットワークであるとしても，ロード・レベリングを達成するには次のような大きな問題が残っている。

① あるＨＯＳＴコンピュータ，あるいは特定のリソースが使用可能になったというダイナミッ

クな情報をどのようにして得るのか？

② 使用可能になった場合，遠隔地のノードのロードが自ＨＯＳＴのロードより軽いという事をシステムはどのようにして決定するのか？

③ 軽負荷のノードにジョブを転送し，実行後送り返すのに要するコストは，自ＨＯＳＴで待機して，空いたら処理するというコストより大きくならないのかどうか？

④ 異ったシステムにおけるコマンドやオプションの違いについては，どう対処すればいいか？このためには，これらの違いを適当に変換する機構や，ネットワーク内でのある種の統一コマンドが必要になろう。

(2) プログラム・シェアリング

プログラムの共用とは，データ・ファイルをユーザのコントロールにより遠隔地のノードに転送し，そのノードに存在するプログラムによって処理し，処理結果を依頼元に返送するような使用法のことを言う。これにより既存のソフトウェアを利用できるようになるので，ユーザにとっては非常に望ましいサービスとなる。プログラムの共用により，プログラミングの労力は大きく減らす事ができ，重複開発もさけられるようになる。このような機能は，多くのネットワークでサービスされているが，異機種ネットワークにおいてプログラムの共用を図るには，記憶モード，コード，ファイルの取りあつかいなどの機種による違いを変換するという難しい問題の解決が必要である。

これらの変換は，複雑さの程度により，送信ノード，コミュニケーション網でのインタフェース，あるいは受信ノード等の各所で行われる。より一般的な解決法としては，ARPANET のＲＡＮＤ Corp で研究されているように情報の変換操作を完全に分離し，集中化されたデータ再編成サービス（Data Reconfiguration Service ）で行い，データの変換はこのＤＲＳを経由するような考えがある（図６－１）

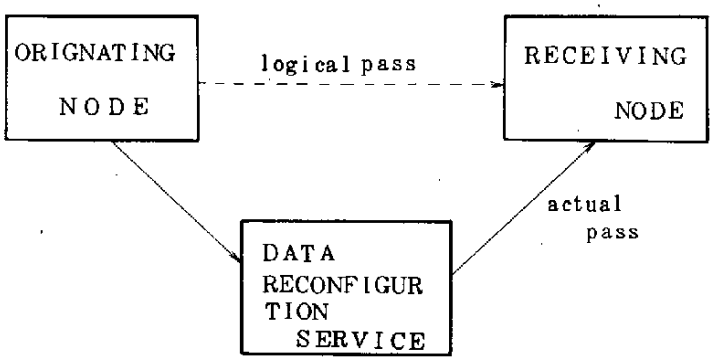



図６－１　Data Reconfiguration Service

この場合，DRSは情報を受信側ノードのプログラムで使用可能なフォーマットに変換する責任を果たすことになる。

このような，ネットワークにおけるデータの共用および変換技術については，次章において詳しく述べられている。

(3) データ・シェアリング

データの共用とは，特別なデータベースが存在するノードにプログラムをユーザのコントロールにより転送しそのデータを利用し，プログラムを実行させるという形態を言う。これは，プログラムの共用とともにユーザにとって非常に有用なサービスである。

あるノードが非常に大容量のデータベースを有している場合は，データの存在場所にプログラムを転送した方がその逆の場合と比べるとより経済的である。

CANUNET の設計段階では，ネットワークを利用するデータ処理の全国的な組織に対するサービスとして数十の大容量データベースが，重要な対象となるであろうことが指摘されている。このようなサービスは各ネットワークでとりあげられているが，この種のサービスを行うための設計時の問題を明らかにしておく必要がある。

オブジェクト・コードレベルでの共用は，同一のマシン以外では，非現実的であろう。したがって，この場合は，ソース言語レベルあるいは中間言語でのプログラムの共用というのが一般的であり，受信したHOSTにおいて自HOSTの機械語にコンパイルされた後，実行されるということになる。FORTRANとかCOBOL といった標準的な言語であっても，メーカの違いによる文法上の違いがかなりあり，このようなソース・プログラムでもネットワーク内で自由に利用しあえるという状況にはなっていないという事を認識する必要があろう。このような非互換性の主要なものは以下の通りである。

① 用語範囲の違い：あるコンパイラにより認識されるソース・ステートメントの集りが，インプリメント毎に異なること。これは，標準化が欠けていることや，特定のメーカにおける機能の違いによりおこる（例Burroughs ALGOL ）

② 言語の意味というものが正式に定義されていないため，同じステートメントの解釈がインプリメント毎に異なること。

③ コンパイラの機能のうち多次元配列の記憶方法のようにマシン・ディペンドなものの扱い。

④ 文字の扱いや，内部記憶における扱い（パック・アンパック等）標準化されていないもの。

⑤ ジョブ制御用ステートメント等，機種，OSにより異なるもの。

このような違いを解釈するには，人手による再プログラミングや，これらの違いを認識し修正するようなある種のネットワーク・フィルターを作ることが必要となる。言語等の標準化はこの種の問題を軽減するのに役立つので，今後とも積極的にすすめていく必要があろう。

### 6.2.2 仮想ネットワークの考え方

本項では，仮想ネットワークの考え方として，非常に明確な分類を試みているHoney S. Elovitzの論文"What is Computer Network"（文献２）を中心に引用しながらJohn W. Benoit等の論文（文献３）によりＢＥＸを，Robert H. Thomas の論文（文献４）ＢＢＮ report の№3012（文献５）等によりＲＳＥＸＥＣの概要を紹介する。

(1) 文献２"What is a Computer Network"の概要

コンピュータ・システムにおける最近の傾向は，データ伝送とパケット・スイッチング技術を用いて，一般に"コンピュータ・ネットワーク"と呼ばれるシステムを構築するようになってきている。しかし，このようなネットワークにおいて，"コンピュータ・コミュニケーション・ネットワーク"と"コンピュータ・ネットワーク"という名前を区別して用いることが重要であるが，この両者はしばしば混同されて呼ばれている。"コンピュータ・コミュニケーション・ネットワーク"においては，ユーザ自身がコンピュータ・リソースを明示して指定し管理しなければならないが，一方"コンピュータ・ネットワーク"においては，これらのリソースはネットワーク・オペレーティング・システムによって，自動的に管理される。TYMNETやARPANETのような既存の"コンピュータ・ネットワーク"と呼ばれるものの殆んどは，正確には"コンピュータ・コミュニケーション・ネットワーク"と称すべきである。

(2) コンピュータ・ネットワークの分類

コンピュータ・ネットワークという言葉は，現在，広域のデータ処理設備を称するのに用いられている。最も一般的な意味では，「コンピュータ・ネットワークはコミュニケーション・システムによって互いに結合されたコンピュータ群とターミナル群の集合」ととらえられている。しかしながら，このような多様な処理設備を一つの言葉で表わすことは，現存のネットワーク類の重要な区別を不明瞭にしている。この区別は，ユーザのネットワークに対する観点により種々の方法があろうが，この論文では，ユーザに対するネットワークの透明度（Transparency）でもってコンピュータ・ネットワークを分類している。この透明度という基準によれば，ネットワークを２つのクラスに分類して考えることができる。

２つのクラスの各ネットワークに共通な点は，コミュニケーション・システムを管理していることであり，異なる点は，コンピュータ・リソースの管理という点である。

第１のクラスのネットワークでは，リソース管理はユーザの責任であり，第２のクラスでは，ユーザが必要とするリソースの獲得と操作をネットワーク・オペレーティング・システムの助けを借りて行うことができる。

この論文でとりあげるコンピュータ・ネットワークとしては，複数のＨＯＳＴコンピュータが通信回線を介して情報を交換する（イン・ハウスでも可）分散型のコンピュータ・ネットワーク

に限定し，集中型のネットワークおよび，マルチ・プロセッサ・システム等は対象外として論を進めることとする。

以下に2クラスのコンピュータ・ネットワークについてそのとらえ方を説明する。

(3) コンピュータ・コミュニケーション・ネットワーク

"コンピュータ・コミュニケーション・ネットワーク"の基本的な特徴は，ユーザにとってそのネットワークが多様なサービスと能力を有する複数のコンピューティング・システムの集りとして見えるということである。この場合，ユーザはネットワークで使用可能な多数のコンピュータ・システムの中から，自分のジョブを実行させるシステムを明示して選択しなければならない。さらに，選択したシステムとネットワークを介してコネクションを確立しなければならない。そしてコネクションが確立すれば，ユーザは，そのシステムとローカル・ユーザの如く通信できるようになる。

この種のネットワークでは，ユーザは，ソフトウェア，サブルーチン，パッケージ，特殊ハードウェア，データ・ベース等の多種多様のリソースが使用可能となる。しかし，"コンピュータ・コミュニケーション・ネットワーク"のユーザが，これらのリソースにアクセスする為には，先ず，そのリソースがどのシステムに存在するのかを知らねばならず，また，特定のシステムのリソースを呼出すのに必要なＨＯＳＴ特有のコマンドに習熟しなければならない。したがって，複数の異なったコンピュータ・システムのリソースを使用しようとするユーザは，各システムについて精通していなければならない。このような異なるシステムへの共通のユーザ・インタフェースは，用意されていない場合が多い。下図６－２はユーザから見た，この種のネットワークの

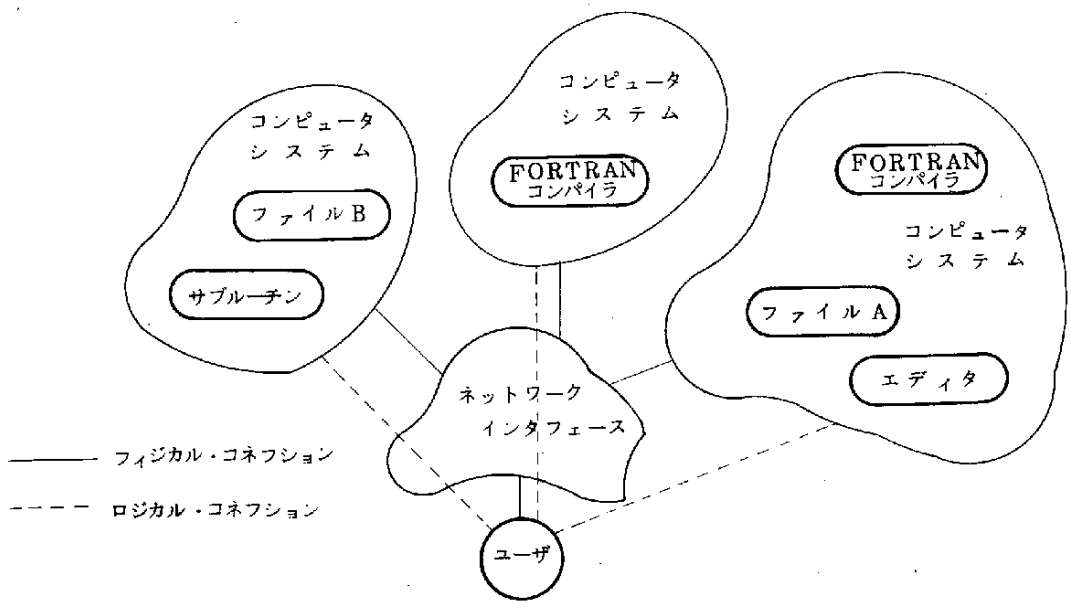



図６－２　ユーザからみた"コンピュータ・コミュニケーション・ネットワーク"

概念図である。

(4) コンピュータ・ネットワーク

"コンピュータ・ネットワーク"においても，ユーザは多様なコンピュータ・リソースを使用できるが，前記の"コンピュータ・コミュニケーション・ネットワーク"の場合に比べて，"コンピュータ・ネットワーク"のユーザは，ネットワーク全体を1つの大きなコンピューティング・システムとしてみなす事ができる。したがって，この場合，種々のオペレーティング・システムとか，ネットワークの多様なリソースにアクセスする為の各種プロトコルなどをマスターする必要はない。ユーザ・ジョブを実行するプロセッサはユーザにとって透明になっており，要求した各リソースの存在場所などを知る必要はない。

このように"コンピュータ・ネットワーク"の本質的な特徴は，ユーザにかわって決定を下すネットワーク・オペレーティング・システムが存在することがある。たとえば，もしネットワークで使用可能な指定のリソースが複数ある場合は，指定されたジョブを遂行するのにどのリソースを使用すれば最適であるかをシステムが決定するということである。さらに，システムは，ハードウェアあるいはソフトウェアの事故に対するバック・アップも行うことになる。指定されたリソースが，もし事故にあったなら，そのジョブを遂行するのに必要なバック・アップ・リソースを用意するのはネットワーク・オペレーティング・システムであり，ユーザは事故が起った事すら知る必要はない。図6－3は，ユーザから見た"コンピュータ・ネットワークの概念図である。

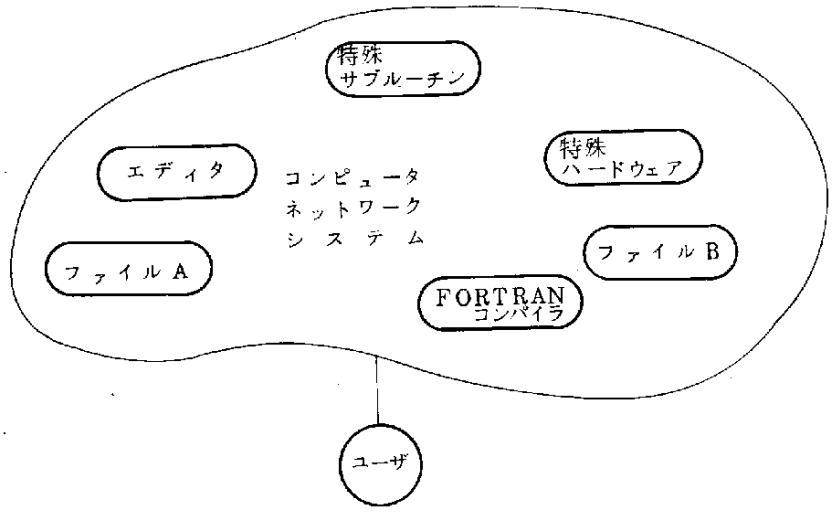



図6－3　ユーザからみた"コンピュータ・ネットワーク"

以下に，2つのクラスのネットワークの例のいくつかを示す。

ARPANETは"コンピュータ・コミュニケーション・ネットワーク"の代表例であり，一方RSEXEC（Rsource Sharing Ex-ective），DCN（Distributed Computers Network），は"コンピュータ・ネットワーク"と呼ぶことができる。

(5) ARPANETとREX

"コンピュータ・コミュニケーション・ネットワーク"ではあるが，ARPANETには，ILLIAC－Ⅳを始めとする非常に魅力的なハードウェア・リソース，およびコンパイラ，エディタ，データ・マネジメント・システム等の各種ソフトウェア・リソース等が多数存在している。特に，ネットワーク設計の基本概念としてプロセス間通信をサポートしている為，ファイル転送などの機能がユーザに提供されている。したがって，ユーザが適当なコマンドを発すれば，ファイルのバック・アップ用コピーなども自動的に創成できるようになっている。

しかしながら，ARPANETにおける各種リソースを利用する為の手続はかなり複雑であり，ユーザがある特定のリソースを利用しようとすると，次のような2つの問題に直面する。まず第1は，ユーザは特定のリソースに対してその存在場所，そのコスト，有用性等に関する情報を収集しなければならず，これは現状ではかなり難しい仕事である。第2に，ユーザが，ある特定のリソースを使用する事を決定したとしても，そのリソースを呼出す為に必要なHOST特有のコマンドを覚えなければならない。このように，ユーザは時には，そのシステムの運用者の助けも得ず，明確なドキュメントも無しに自分自身が各種の異なったシステムに習熟しなければならないことになる。

ごく最近まで，このような問題を解決する為の助けとなるようなネットワーク・ワイドな案内機能というものは，ARPANETには存在していなかったが，最近になってようやく，ごく限定されたものではあるが，MITRE Corp.のグループによる，この分野における解決への努力がなされた。これはREXと呼ばれるネットワーク・リソース・マネージャであり，ユーザに対してARPANETにおける多様なリソースに関する情報サービスを提供しようとするものである。MITRE Corp.のグループはユーザが選んだリソースに自動的に接続する為に必要なプロトコルを開発しようとしており，これによれば，指定したリソースにアクセスしようと欲するユーザは，もはやHOST特有のコマンド言語等を覚える必要はなくなる。このREXについては6.2.3のネットワーク操作性の向上のB."REXによるユーザ・サービスの向上"において詳しく紹介している。

(6) RSEXEC

前述のように，ARPANETは，"コンピュータ・コミュニケーション・ネットワーク"としてクラス付けしたが，ARPANETには，"コンピュータ・ネットワーク"としての基本的な特徴を有する論理的なサブネットが存在している。

それは，ＰＤＰ－10のTENEX オペレーティング・システムのもとに，複数個のユーザ・プロセスとしてインプリメントされたRSEXEC（Resource Sharing Exective）である。RSEXECは，現状ではそのデシジョン・メーキングの機能が限定されたものではあるが，"コンピュータ・ネットワーク"の範ちゅうに入れることができる。

"コンピュータ・ネットワーク"における特性の一つは，ユーザが，自分のプログラムやデータの存在場所について関知しなくても良いということである。RSEXECは，ユーザに自分のプログラムがRSEXEC をサポートしているいくつかのTENEXのいずれであっても実行可能であるようなトランスペアレンシーを提供している。ユーザが，いくつかのシステムのうちのいずれでも自分のジョブを実行できるということは，結果的にネットワークのアクセシビリティと信頼性を向上することにつながる。

ＴＩＰを通してARPANET を利用しようとするユーザは，自分がどのサイトのTENEXで実行するかを指定しなくとも，自動的にアサインされるようになっている。この場合，ユーザがコミュニケートしようとするサイトのトランスペアレンシーは，RSEXECによって保証されている訳である。

特定のプロセッサに依存しないで，あるタスクを実行するには，RSEXECは，ユーザ・ジョブを実行しているサイトから，リモート・サイトのファイルへアクセスする為の機構を，備える必要がある。RSEXECは，現在はコマンド言語レベルに限定されているが，このようなリモート・ファイル・アクセス機構を有している。ユーザはこのコマンド言語を用いて，自分のＨＯＳＴから，リモートＨＯＳＴのファイルの創成，消去，追加，ファイル名変更等を行うことができるようになっている。RSEXECは，すべてのRSEXEC ＨＯＳＴに渡るファイルのネーミングの変換をサポートしており，これによりRSEXECのＨＯＳＴ のバウンダリを超えて，名前によりファイルが参照できる。各RSEXEC のファイル名のディレクトリは，そのファイルが存在するサイト名をユーザ・ファイルのパスネームに含ませている。ユーザは，システムでユニークに識別するのに必要なパス・ネームのある一部だけを指定することも可能でありその場合のパス・ネームの複合化は，システムによって行われる。このようにしてRSEXECは，ファイルの扱いをユーザにトランスペアレントにしている。これらのファイルの管理方法・アクセス法については，次章に詳説されている。

RSEXEC はユーザ・ジョブを実行するのに最適なプロセッサを決定することまでは行っていないが，ＴＩＰからのアクセスに対しては，次のような機能をサポートしている。

ＴＩＰには，その能力的限界から，ユーザにとって便利なコマンド言語解釈ルーチンがない為，ネットワークの状況とか，各種ＨＯＳＴの状態，リソースの状況等を知ろうとすると，あるＨＯＳＴにロギングして聞いてみるより方法がない。このような要求に応える為，RSEXEC ではＴＩＰユーザが必要としているコマンド言語解釈ルーチンを提供しており，ある特定のＨＯＳＴへ

のアクセスに先立ってRSEXECをネットワーク・コマンド言語解釈ルーチンとして利用できるようになっている。この場合のＴＩＰからのRSEXECへの接続は，ブロードキャスティング・イニシャル・コネクション・メカニズムをとっており，ＴＩＰからのサービス要求に対して，最初に応答してきたTENEXに接続される事になっている。

RSEXECでの経験は，ARPANETのTENEX・HOST間でのリソース・シェアリングを援助できることを示している。これは，それ以前のARPANETにはなかった事であり，ユーザはローカル・システムの境界を超えてリソースにアクセスできるようになった。RSEXECが発展するにつれ，TENEX HOSTはさらに密着に結合されていくだろうし，多重ＨＯＳＴという目標に近ずいていくことであろう。

(7) ＤＣＮ

メリーランド大学で開発中のＤＣＮ（Distributted Computer Network）は，ネットワークの複数のコンピューティング・センタで独立に実行されている協同オペレーティング・システムの集りとしてみえるような，ネットワーク・オペレーティング・システムをユーザに提供している。

ＤＣＮは，数台のＰＤＰ11／45Ｓを使用してインプリメントされており，ネットワーク・オペレーティング・システムは各ＰＤＰ－11 に備えられているメモリ管理ハードウェアを利用して，ネットワーク・ワイドな仮想記憶装置を実現している。ＤＣＮの重点目標は，ユーザに一定した形式のプロセス間通信を提供すること，プロセスの存在場所からの独立性を与えること，ネットワーク上では同一のプロセス構造を実現することである。この設計思想により，ＤＣＮはユーザに対して①指定されたプロセスに対してどのリソースを割当るべきか，②ファイルはどこに存在するか，あるいは存在すべきか，③そのプロセスをどのプロセッサで実行させるか，の３つの決定を自動的に行う。ＤＣＮは，ネットワーク全体をプロセス・オリエンテッドな構造として扱っており，システムあるいはユーザのいずれかによって要求された各アクティビティを遂行するプロセスが発生され，ネットワーク内の入出力装置や各リソースは夫々に対応するプロセスとして扱われている。

ネットワーク・ワイドな通信は，プロセス対プロセスのレベルで行われ，この場合は，システム内のプロセスの存在場所は知る必要はなく，必要なリソースあるいはサービスに対する要求は，単に，適当なプロセスに対しメッセージを送るだけで良い。このように，ＤＣＮでは，ユーザの要求に対してどのリソースが最適であるかという決定をシステムが行っている。この決定は，各ＨＯＳＴからの入札制のような形でネットワーク・オペレーティング・システムによって次のようになされる。リソースを要求する新しいプロセスが生成されると，すべてのＨＯＳＴのプロセス管理部が協議して新しいプロセスの要求を満足できるのは，どのネットワーク・リソースであるかということを決定する。そのリソースがもしプロセッサであるとすると，そのプロセスを実

行する為のプロセッサがネットワーク・オペレーティング・システムによって選ばれることになる。したがって，リソースの割当は，システムによって自動的に行われ，ユーザは特殊なファシリティの存在場所などは，何ら知る必要がない。

ＤＣＮでは，ネットワーク・ワイドなファイル・システムをインプリメントする為にもプロセス間通信とプロセス・オリエンテッドな構造を用いており，ファイル・アクセスが要求された時にユーザと通信するファイル管理プロセスを各ＤＣＮのＨＯＳＴが保持している。

必要なファイルがユーザのローカル・ＨＯＳＴに存在する場合は，そのＨＯＳＴのファイル管理部が要求されたファイルにアクセスする為の特別なプロセスを発生させ，ローカルなファイルでない場合は，要求を各ＨＯＳＴのファイル管理部に通知（broadcast）し，そのファイルの存在場所を決定する。

そのファイルが自分の管理しているものであれば，そのリモートＨＯＳＴのファイル管理部が，そのファイルにアクセスする為のプロセスを発生させる。ファイルがローカル，あるいはリモートＨＯＳＴのもののいずれであっても，ユーザプロセスは，ファイル・アクセス用のプロセスと通信するようになっている。

現状のＤＣＮは，プロセス間通信を通して２台のＰＤＰ１１／４５Ｓでもって運用可能であり，ソフトウェアとしては，ＰＤＰ－１１アセンブラ，ＢＡＳＩＣ，ストラクチャード・プログラミング用のＳＩＭＰＬ－Ｘが使用可能であり，ユーザに仮想記憶と，自動的に管理される物理的メモリを提供している。

将来計画としては，あと数台のＰＤＰ－１１／４５Ｓ ＨＯＳＴの追加を考えており，リンクの事故等に対するトポロジーの変更や，ダイナミックなエラー回復機能の研究環境を強化するつもりであり，さらには，ロード・レベリングを行うようなプロセスおよびリソースの管理の研究を今後共進めていく予定になっている。

(8) ま と め

この論文で"コンピュータ・ネットワーク"として分類されたシステムは，いずれも実験段階のものであり，異機種ネットワークにおけるロード・レベリングやプロセッサに依存しないでタスクを実行するなどの問題については今後より一層の研究が必要である。

一方，"コンピュータ・コミュニケーション・ネットワーク"においては，必要とするリソースにアクセスする為の，ＨＯＳＴ毎に異なる複雑なコマンド言語を排除しなければならないという必要性が広く認識されてきしいる。現在ユーザ・インタフェースを単純化および標準化しようとしてＲＥＸシステムのようなハイ・レベル・プロトコルの開発が進められているが，このようなシステムを今日の"コンピュータ・コミュニケーション・システム"に早急に取り入れることを，この論文では強く推奨している。

### 6.2.3 ネットワークの操作性の向上

#### A. Network Access Machine によるアプローチ（参考文献 6, 7, 8 ）

(1) はじめに

異機種コンピュータ・ネットワークのリソースを使用するためには，ユーザは多くの異なったリソース・アクセスの手続を覚えなければならない。各ＨＯＳＴシステムに対してイニシャル・コネクションの確立，ログオン，サブシステムへのアクセス，ログオフ，ＨＯＳＴとの切断等を行おうとする場合ＨＯＳＴ毎に異なった手続をとらなければならない。特定のＨＯＳＴに依らないで仕事をやろうとするようなユーザが異なったＨＯＳＴへの各手続をすべて知らなければならないということは非常に不便である。このようなユーザ・オリエンテッドな問題を解決する手段としては，すべてのＨＯＳＴにおける手続を標準化するとか，より上位のレベルに標準手続を設け各ＨＯＳＴで変換を行うとか，ユーザ側に標準手続を設け各ＨＯＳＴ向に変換する等各種の方法が考えられる。

ＮＢＳでは，このような問題に対処する為にNetwork Access Machine （ＮＡＭ）と呼ばれるものを開発した。ＮＡＭは，ユーザが作成したマクロを，任意の異機種ネットワークやリモート・システムへのアクセスに必要なインタラクティブな会話手続を創成するように展開するものである。リモート・システムからの応答は，ＮＡＭによってリアル・タイムに解析され，個々のコンピュータ・システムの特定のリソースに対するアクセスは，ユーザがあらかじめ登録した手続を用いたり，実行時にダイナミックにその手続を創成したりすることができるようになっている。

(2) ＮＡＭの概要

ＮＡＭはミニコンピュータ・ベースで作成されており，端末ユーザとコンピュータ・ネットワークのリソースとの間のアクセス・パスを確立する機能を有している。複数のネットワーク・システムやネットワーク内の異機種システムを使用する場合，異なったマシンの似たようなリソースあるいはマシンの型式は同一であっても設置場所が違うマシンのリソースへのアクセスには夫々異なった手続を必要とする。（文献８）図６－４，図６－５はMEDLINE システムおよびＡＲＰＡのＮＩＣに対するアクセス手続の例である。

これらを一元的に扱うには，アクセス手続の標準化が必要であるが，現状ではほとんど標準化されていない。現状では，個々のシステムではもちろんのこと，同一システムであってもバージョンが異なる場合は，新しい異なったアクセス・コマンドを使用しなければならない。この違いは，図６－５のＴＩＰからＮＩＣを使用する場合のように，異なったシステムを階層的に使用して，目的のリソースへのアクセス・パスを設定しようとすると著しく表われる。

コンピュータ・ネットワークへのアクセスを単純化するには，前述のように種々のアプローチ

```
PLEASE TYPE THE LETTER D

PLEASE LOG IN:  NLM

PASSWORD:  ;MEDNBS0L

PLEASE LOG IN:  NLM

PASSWORD:  ;MEDNBS01

THIS TERMINAL IS CONNECTED TO THE MEDLINE RETRIEVAL FILE SET

HELLO FROM NLM/MEDLINE. THE MEDLINE AND SDILINE DATA BASES NOW
CONTAIN JUNE 1973 DATA.  DO YOU WISH THE NEW-USER OR EXPERIENCED-USER
FORMAT?  TYPE N OR E AND STRIKE THE CARRIAGE RETURN KEY.

USER:
```

図6－4　MEDLINEへのアクセス手続

```
1       SM  "HELLO 306"
2       UC  "@L 2"        Logon command, Host address no. 2
3, 4    SM  "LOGGER R OPEN T OPEN"
5       SM  SYSTEM ID
6       SM  FACILITY ID
7       SM  OPERATING SYSTEM ID
8       SS  "@"
9, 10   UC  USER NO/ACCOUNT NO
11      SR  "IDENT = "            Identification request
12      UC  IDENTIFICATION CODE
13-16   SM  JOB NO/ TT NO/ DATE/ TIME
17      SM  STATUS MESSAGE
18      SS  "@"
19      UC  ENTER
20      SS  "*"
-------------------------------
21      UC  "E L"             Execute Logout
22-27   SM  JOB NO/USER NO/ACCT NO/TT NO/DATE/TIME
28, 29  SM  COMPUTE TIME / CONNECT TIME
30      SS  "@"
31      UC  "C"
32, 33  SM  "T CLOSED R CLOSED"
```

図6－5　ＴＩＰ端末からのＮＩＣ へのアクセス

があろうが，現存のネットワークはアクセス手順が複雑であり，このアクセスをより便利にする為には，ネットワークの再設計が必要となろう。将来のネットワークは標準的なアクセス手続に適合できるようになろうが，現存の異機種ネットワークにおけるこの問題への対処法の一つとしてＨＯＳＴマシン間の通信プロトコルの機能を拡張することが考えられる。すなわち，ネットワーク内のすべてのＨＯＳＴマシンが，共通のアクセス言語として解釈できるような高位のプロトコルをサポートする方法である。

ＮＡＭの開発は上記とは別のアプローチであり，ユーザとネットワークの間にミニコンピュー

タを置き，それによって任意のリソースに対する適切なアクセス手順をジュネレートする方法である。図6 －6にその概念図を示す。

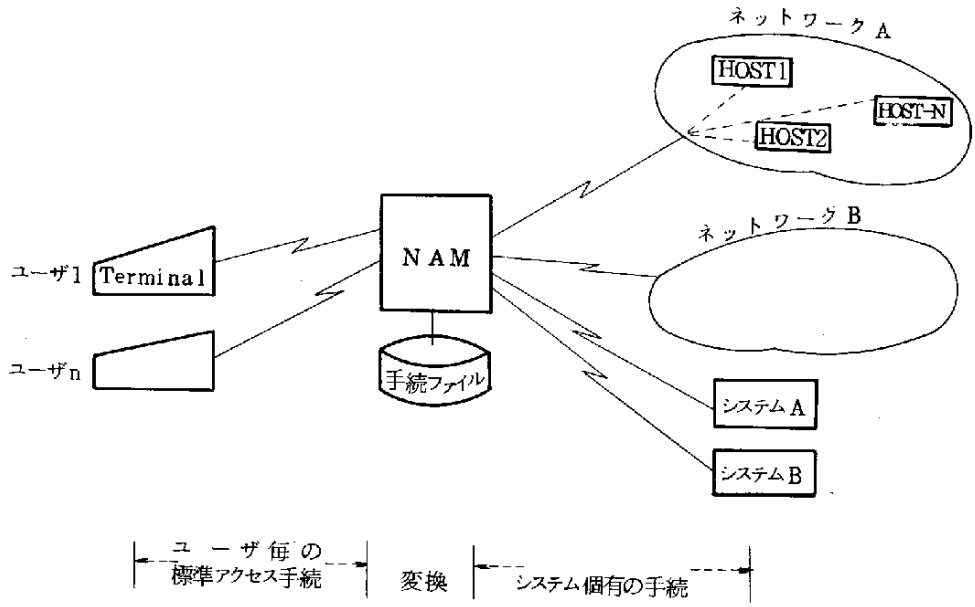



図6 －6　NAMの概念図

NAMは各種のリソースへのアクセス手順を簡単に参照するためのファイル・ディレクトリを有しており，これらの参照は個々のユーザが自分専用に用意した標準アクセス手続ごとにおこなわれている。これらのアクセス手続はNAMによってマシン毎に異なった会話手続に展開される共通のマクロ命令を用いて，個々のユーザ毎に定義登録される。この会話手続は，リモート・システムに送り出されるメッセージと，そのシステムからの期待応答とから成っており，リモート・システムからの実際の応答は，NAMによって期待応答と比較され，リモート・リソースへのアクセスが正常に行われるよう処理される。

以下にNAMによる使用例を示す。

あるコンピュータ・システムへログオンする為には，パラメータとしてユーザ名，パスワードそしてアカウント番号を指定する次のようなマクロLOGが定義される。

```
○MACRO LOG USER PASSWORD ACCOUNT
○TERM '〔CRLF〕@'
○MATCH '@'
○OUTPUT 'LOGIN USER PASSWORD ACCOUNT'
○ENDMACRO LOG
```

NAMユーザは次のようにタイプインしてこのマクロを呼び出す。

LOG USER1 PASS1 ACC1

このメッセージは，次のような形に展開されたメッセージとなってリモート・システムに送出するメッセージとして用意される。

LOGIN USER1 PASS1 ACC1

ＮＡＭは，リモート・システムからのメッセージを監視して，最初のＣＲ，ＬＦ続いて@が受信されるとシステム応答が完結したものとみなされる。次に応答@と，マッチ・ストリングとして定義された@が比較され，一致したら，先程のLOGINメッセージがリモート・システムに送出されることとなる。

このような機能を有するＮＡＭによるアプローチをとったＮＢＳでは下記のような見解を示している。ユーザからの各種の異機種ネットワークへのアクセス手続にかかる労力を軽減するには，最初に述べたように既存ネットワークの標準化，ネットワーク・プロトコルの追加，新しいネットワークには標準アクセス手続を採用するなど各種の方法が考えられる。その為には，ユーザからのアクセス手続のみにとどまることなく，新しいアクセス・コマンドやネットワーク制御言語を設計し，作成し，テストを行うための試験的環境として利用し得るような汎用のネットワーク・アクセス・マシンをつくりあげることが，問題を解決する為に最も近道であり生産的なアプローチであると確信している。

**Ｂ．ＲＥＸによるユーザ・サービスの向上**（文献３より引用）

(1) はじめに

本論文は，コンピュータ・ネットワークにおけるオンラインのユーザ援助機能の必要性と，提供されるべきサービスのタイプを明確にしたものである。

ここでは，ＡＲＰＡネットワークにおいて開発されたＲＥＸシステムと呼ばれるプロトタイプのリソース・マネージャについて説明している。端末使用者は，簡単な会話型問合せ言語によりＲＥＸを使用し，ネットワークのリソース情報を得ることができる。さらに，ＲＥＸユーザは，リソース獲得の要求を出すだけで，適当なリソースに自動的に接続する事が可能である。

(2) 概　要

ＡＲＰＡネットワークの多くのユーザが疑問に思うことは下記のようなことであった。

① ネットワークを利用するにはどうすれば良いか？

② ネットワークのリソースやサービスにはどんなものがあるか？

③ どうしたらそのようなリソースやサービスを利用できるのか？

このような事を知る為の，実用上唯一の方法は，各ＨＯＳＴ・サイトの専門家に直接問合せる事である。

しかしこのような方法では適当な専門家を見付けるのには時間がかかり，度重なる連絡，確認にはアプリケーションのユーザは耐えられなくなり，ついにはあきらめて，その仕事をネットワ

ークを利用しない形に変更する事になってしまう。ARPANET においてはこの種の問題が増えてきている事は，文献2,6,7 にも述べられている。

最近まで，ARPANET においては，新しいユーザ・サポート機能の提供や，既存機能の拡張，さらに個々の機能の統合化といった面にかなりの力が注がれていた。

個々の機能の統合化の問題に関しては，リソース管理のプロジェクトがいくつかあり，ソフトウェアや特殊なハードウェア等のネットワーク・リソースをネットワーク全体に渡る標準的な方法でユーザに提供したり，リソースの要求に関するＨＯＳＴ特有のコマンドを自動化するような試みがなされている。ユーザ援助の機能もリソースの一つと見なす事ができ，リソース管理エグゼクティブはユーザとやり取りしながら，どのような援助機能がそのユーザに必要なのかを決め，そのような援助機能の自動的な獲得も可能にしている。

本論文では，ARPANET ユーザが利用可能なオンラインの援助機能について調査し，そのうちの一つである著者が開発したＲＥＸシステムについて詳説している。

(3) ＲＥＸによるユーザ・サービス

コンピュータ・ネットワークにおいて現存するユーザへのオンライン情報サービスは，機能的に３つのグループに分ける事ができる。これらは，①インタラクティブ・システムに組込まれた機能，②ユーザ・サービスのディレクトリ ③リソース管理サービスである。①は＂help＂機能と呼ばれているもので，ＴＳＳなどにおいて遠隔地の単一システムのユーザの要望に答えるべく開発させたものであり，コマンド促進や用語診断，オンラインのレポート作成機能や使用法指導の機能等が含まれる。このような機能は，すべて，当初はネットワーク化されていない単一のコンピュータ・システム用に備えられたものであり，一般には，ユーザがネットワークへの最初のアクセス情報（例えばローカル・コンピュータ・センタの利用方法）を得る為に使用するものである。

②，③のユーザ・サービスのディレクトリ，およびリソース管理サービスの２機能は，コンピュータ・ネットワークにおいて発生する種々の新しい問題を解決する為に生まれたものである。これらの問題は，コンピュータ・ネットワークにおけるリソースの多種性多様性から派出するものであり，さらにユーザとそのリソース機能が地理的に離れている事から出てくるものである。このような状況下においてユーザは，地理的に散在する多種多様な能力とサービスを有するコンピュータ・システムの集りの中から，望むリソースを見つけ出さなければならないという問題に直面し，コストやコンパティビリティ，アベイラビリティ等の競合要因を比較した上で最適のリソースを選択する事が必要となろう。

ユーザ・サービス・ディレクトリというのは，単に，ネットワーク内の各コンピュータ・システムで使用できるリソース情報の集中的管理機能を提供するものである。通常，このディレクトリには，ローカルな情報サービスのアブストラクト類が納められており，ユーザはこのアブスト

ラクト類から，ネットワーク内のコンピュータに関する情報を得ることができるようになっている。ARPANET におけるこのたぐいのサービスとしてはＮＩＣ（Network Information Center ）のリソース・ノートブックがあり，オンライン，オフライン印刷のいずれの型式でも利用できる。

ここで得られる情報はネットワーク内の個々のコンピュータ・システムに関するものであり，ネットワーク・ユーザが使用できるリソースのネットワーク内における相対的な情報は得る事ができない。

これに比べ，リソース管理サービスは，技術的により高級である，リソースに関する比較情報を得る事ができ，各リソースの使用方法指導およびリソースの選択，獲得の為のガイダンス機能なども有している。ARPANET におけるこのようなユーザ・サービスの最初のものはＢＢＮによって開発されたRSEXEC（Resource Sharing Exective ）である。

これは，複数のコンピュータに渡る管理プログラムであり，地理的に散在するネットワークのコンピュータ群を一つのタイムシェアリング・システムとして利用できるような機能を提供するものである。（これはネットワーク内の論理的サブネット logical subnet ）と呼ばれる。RSEXECはTENEXオペレーティング・システムを有する複数のＰＤＰ－10で実行され，協同作業を行う各サイトのTENEXとの通信を行うことができるよう設計されている。TENEXサイトがアクティブである時はRSEXEC ユーザはいつでも下記のような機能を利用できる。

① TENEXシステムに，現在ログイン中の特定のユーザの場所を見つける事

② 特定のTENEXサイトにおけるアクティブなユーザの識別情報を決定する事

③ 協同作業を行うサイト群に渡るコンポジット・ファイル・ディレクトリを創成し，これらのサイト上でファイルを操作すること。

④ 妥当なアカウントで，協同作業を行うサイトにジョブを発生させる事

これらに加えて，RSENEXでは，各TENEXサイトの現在の状態（稼動中 or ダウン）や，ネットワーク内の各協業ＨＯＳＴの負荷状態等を知る事ができ，ネットワーク・ニュース・ファイルもアクセス可能である。

このような試みの目的は，コンポジット・ファイル・ディレクトリの扱いとファイルの交換，およびTENEXシステム間での状態情報のやり取り等の実験である。

この試みは，現在，TENEXシステム間に限られており，異機種ＨＯＳＴ間には適用されていない。

RSEXEC では，特定のリソース（特定のオンライン・ユーザ名を除く）を見つけるような事はできず，ユーザがそれらのリソースを比較するような事もできない。

次節で述べるＲＥＸシステムは，MITRE社によって開発されたものであり，ネットワーク内の特定のコンピュータ・システムの特定のリソースに関する情報を提供するものであり，その目

的はネットワーク内リソースに関する所在情報や，リソースの獲得方法等の情報を提供するものである。これらの最終的な目標は，ユーザが選択するリソースとユーザを結合する多種多様なプロトコルをＲＥＸが実行できるようにすることである。

要約すると，コンピュータ・ネットワークの現状の技術では，このようなユーザ・サービスの道具としては，実験的試み，あるいは試作品がARPANET で現在利用可能になっているだけである。これらの目標は，ユーザが利用しているローカルなリソースを上まわる機能を有するネットワーク・リソースをユーザに見えるようにする事であると言える。

(4) ＲＥＸの基本機能

本節では，ＲＥＸシステムの使用法について，ネットワークのリモート・ＨＯＳＴのリソースの，所在情報を識り，比較し，選択し，使用するといったサービス機能の詳細について述べる。ＲＥＸシステムはネットワーク内の各種ＨＯＳＴに存在するリソースに関する情報を提供するものであり，最終的にはユーザが要求するリソースとユーザを結合するものである。したがってこのようなユーザ・サービス機能も一種のリソースであり，ＲＥＸは他のネットワークに渡るあるいはＨＯＳＴコンピュータにおけるhelp機能を統合したものとして動作する。現ヴァージョンでは情報検索に関してはフル機能を有しているが，実際のリソース獲得については数種のネットワーク・サービスの指導サービス機能に対する自動接続に限られている。

各リソースに関して得られる情報の種類としては，リソース獲得の為の手続，コスト，help機能等がある。

これらの各リソースに対する情報あるいはhelp情報は，ＲＥＸシステムが存在しているＨＯＳＴにファイル化して保管されている。ターミナル・ユーザがリソースの存在場所を見出し，それに関する説明を得るために，専用の簡単な問合せ言語が用意されている。ＲＥＸシステムは，フル機能を有するリソース管理システムに拡張することが可能であり，マシンとマシンのインタフェースは単なる要求と応答の交換として設計されている。

下記にＲＥＸの情報検索機能とリソース獲得の機能について説明する。

(a) Interactive Information Language

ＲＥＸのユーザ言語はリソースの所在情報を検索し，リソースの集り，あるいはカテゴリーに関する情報を提供するようなコマンドとして用意されており，各コマンドの使用方法を説明するオンラインのhelp機能も提供されている。

情報検索用のコマンドとしては，ＦＩＮＤおよびDESCRIBEの２つがあり，両コマンドともキーワードの組合せを伴って指定される。キーワードは下記の４種のグループに分けることができる。

① Resource Name（リソース名）

これは，特定のリソースの実際の名称を参照するものであり，現在のデータ・ベースには，

下記のようなリソース名が入っている。

SNOBOL － a compiler
DATACOMPUTER － a hardware／software data management system
PDP－10 － a type of CPU
OS－MVT － an operating system
QED － an editor program
ILLIAC － a host name ( host systems are also considered to be resources )
DELPHI － a help facility describing the Case-Western TENEX system.

② Attributes of Resources (リソースの属性)

現在使用されているのは下記の3種のみである。

INVOCATION － how to get access to the resource
HELP － how to get access to help facilities for the resource
COST － cost of use of resource ( if determined )

③ Categories of Resources (リソースのカテゴリー)

リソースはいくつかのカテゴリーに分類される。カテゴリーは，より一般的なカテゴリーの下に包含される場合もあり，例えばFORTRANというリソースはLANGUAGESというカテゴリー内に含まれるし，MACSYMAというリソースはENGINEERINGのカテゴリー中にありさらにそれはAPPLICATION－SYSTEMというカテゴリー中に含まれるようなことになる。

④ HOST Names (HOST名)

HOST名はREXコマンド中でリソースの存在場所を示すなどの特殊な文脈として用いられ，リソース名が指定されていてもHOST名が指定される場合もある。HOST名としては下記のようなものがある。

BBN－TENEX
CCA
ILLIAC
MIT－DMCG
MULTICS

リソースの完全な階層構造は例えば下記のようになる。

```
DATA－MANAGEMENT
        IMS1
        DMS－1100
              INVOCATION : RUN DMS
              COST : 5 CENTS／RETRIEVAL
              HELP : HELP－1100
```

⑤ The FIND Command

ＦＩＮＤコマンドはリソースあるいはリソースの組合せの存在場所を見つけのるに用いられる。リソースの組合せは集合演算子ＡＮＤ，ＯＲ，ＮＯＴを用いて表現され，ＦＯＲ結合子はある特別なリソースの属性を識別する際に使用される。下記はＦＩＮＤコマンドの使用例である。

○FIND FORTRAN

ＦＯＲＴＲＡＮ コンパイラを有するＨＯＳＴ一覧をつくる。

○FIND FORTRAN NOT PDP10

ＦＯＲＴＲＡＮ コンパイラを有し，ＰＤＰ10 ではないＨＯＳＴ一覧をつくる。

○FIND (FORTRAN OR WATFOR) AND MACSYMA AND (MIT-DMCG OR USC-ISI OR SRI )

ＭＩＴ－ＤＭＣＧ，ＵＳＣ－ＩＳＩ．ＳＲＩのＨＯＳＴ群の中でＭＡＣＳＹＭＡ と（ＦＯＲＴＲＡＮあるいはＷＡＴＦＯＲ ）の両方を有するものを選ぶ

○FIND HELP FOR FORTRAN AT USC－ISI

ＵＳＣ －ＩＳＩにおけるＦＯＲＴＲＡＮに対する援助（ help ）機能の名前を探す。

ＦＩＮＤコマンドのフォーマットは下記の通りである。

ＦＩＮＤ＜site－expression＞

ここでsite－expression はリソース名とＨＯＳＴ名のBoolean結合である。

⑥ The DESCRIBE Command

DESCRIBEコマンドは，リソースのカテゴリーや集合に関する情報を得るのに使用される。

"ＡＴ"はsite－expressionを示すのに用いられ，ＦＯＲは要求するリソースの属性を示すのに用いられる。

下記はDESCRIBEコマンドの使用例である。

○DESCRIBE FORTRAN AT BBN

これはＨＯＳＴ ＢＢＮにおけるＦＯＲＴＲＡＮ の属性 （ｉ･ｅ 使用手続，help 機能の使用方法，実行のコスト等）を説明する。

○DESCRIBE HELP FOR FORTRAN AT BBN OR MULTICS

ＢＢＮとＭＵＬＴＩＣＳにおけるＦＯＲＴＲＡＮに対するhelp機能を説明する。

○DESCRIBE LANGUAGES AT NOT TENEX

これは，ＴＥＮＥＸオペレーティング・システム以外のＯＳを有する各ＨＯＳＴでのすべての言語とその属性を説明する。

```
AT SITE MIT-DMCG,
LANGUAGES:
ASSEMBLERS:
        MACRO-10:
            INVOCATION: MACRO
            HELP: ?
            COST: $6/CPU-MIN
        FAIL:
            INVOCATION: FAIL
            HELP: HELP
            COST: $6/CPU-MIN+25% SURCHARGE
COMPILERS:
        FORTRAN:
                    .
                    .
        LISP:
                  .
                  .
```

○DESCRIBE CASE-10

これはCASEにおけるHOST属性のすべてを説明する。DESCRIBEコマンドのフォーマットは下記の通りである。

$$\text{DESCRIBE} \left\{ \begin{array}{l} <\text{attribute}>\text{FOR}<\text{resource}> \\ <\text{resource}-\text{expression}> \end{array} \right\} (\text{AT}<\text{site-expression}>)$$

DESCRIBE コマンドを使用する場合は{ }内の2つのいずれかを選択する必要がある。

<resource-expression>はリソース,カテゴリー名, HOST名が, (コンマ)ブランクあるいはANDで区切られたリストである。例えば

DESCRIBE FORTRAN, COBOL AND SNOBOL

では, FORTRAN, COBOL, SNOBOLに関する情報が説明される。

<attribute>FOR<resource>という表現は, 特定の名前を持つリソースの標準的あるいは既定義の属性を示すのに用いられ, 例えば

DESCRIBE COST FOR SOS

では, SOSエディターのコスト属性として記録されている値が知らされる。

<site expression>はここではオプションであり, FINDコマンドと同じシンタックスである。もしこれが指定されない場合は, HOSTに依存しない一般的なリソースに関するデータ・ファイルから検索されリソース情報として使用される。このファイルに対するアクセスでは要求されたリソースに関する簡単な説明のみが得られるようになっている。例えば

```
DESCRIBE SNOBOL
```

では，SNOBOL :a string-proccessing compiler language のような説明が得られる。

FINDとDESCRIBEコマンドを一緒に使用することにより，ネットワーク・リソースに関するより多くの情報を引出すことができるようになる。例えば，下記の例は特別のアプリケーションに対するデータ・マネージメント・システムを見つけるような使用法を示したものであり，下線部分はユーザが入力したものである。

ユーザがUCSBのIMS1が最適なデータ・マネージメント・システムであると決定したとすれば，それを使用する当の手続を始めることになる。

```
DESCRIBE DATA-MANAGEMENT
DATA-MANAGEMENT:
  IMS1: DMS GENERALLY AVAILABLE ON IBM 360 AND 370
        SYSTEMS
  LISTAR: A DMS . . .
  MIDAS: . . .
    .
    .
    .
  DATA-COMPUTER: A HARDWARE/SOFTWARE DMS CAPABLE
                 OF HANDLING VERY LARGE DATA FILES
    .
    .
  FIND IMSI
  THE FOLLOWING HOST HAVE THE REQUESTED RESOURSE:
  UCSB-MOD75 IS HOST NO. 3
  DESCRIBE COST FOR IMSI AT USCB
  AT SITE UCSB,
    COST: CONNECT-TIME: $5/HR, CPU-TIME: $8/MIN,
          FILE STORAGE: $5/VOLUME/MON
```

⑦ Help Facilities（援助機能）

REXシステムでは自分自身に対するhelp機能も有しており，もし"？"が入力されたとすると，利用可能なコマンド一覧と各コマンドの使用方法に関する簡単な説明が出力される。この説明により，ユーザは使用可能なコマンドを思い出すことができ，新しいユーザはHELPコマンドの存在を知る。HELPコマンドは，より詳しい情報を得る為に使用され，このコマンドによりREXシステムと全コマンドに関する2頁分の説明書が得られる。この説明は，REXの新しいユーザにとって十分なものである。

FIND，DESCRIBEコマンドはアーギュメントとしてキーワード表現式が用いられる。

KEYWORDSコマンドは，ＦＩＮＤ，DESCRIBE で許されるキーワードの説明を得るためのものである。

KEYWORDS コマンドのシンタックスは下記の通りである。

KEYWORDS { RESOURCE / ATTRIBUTE / CATEGORY / HOST / ALL }

ここで，RESOURCE，ATTRIBUTE，CATEGORY，HOST が指定されると，そのグループ中で許されるKeyword の一覧表が出力され，ＡＬＬあるいは指定なしの場合はすべてのグループ内のKeyword 一覧が出力される。この指定はブランクで打切るか，ＡＮＤで結ぶなどして複数グループに対しても行うことができる。

⑧ The QUIT Command

ＲＥＸシステムから抜け出る場合にはＱＵＩＴコマンドが使用される。

⑨ Commands and Keywords Abbreviation

コマンド，キーワードは先頭の一文字あるいは数文字で省略形の記述が許される。

F SNO AND COB

は，下記の表現と同じである。

FIND SNOBOL AND COBOL

(b) The Initial REX ACQUIRE Capability

ACQUIRE コマンドは，リソースをトランスペアレントな方法（他ＨＯＳＴへのコネクト手続などをユーザにはやらせない）で獲得する為のものであり，ユーザがリソースを指定するだけで，ＲＥＸシステムが自動的にそのリソースに接続してくれ，ユーザはそれ以上の指定，動作を行う必要がない。

現在の初期ヴァージョンでは，ACQUIRE コマンドは指導用のhelp 機能サービスに対してのみと制限されているが，

ACQUIRE TENEX-INTRO

と指定すれば，TENEX ＯＳの紹介指導サービスに自動的に接続される。ここで指定される指導機能はＲＥＸシステム内のものではなく，各ＨＯＳＴシステムによって提供されているサービスである。ＲＥＸシステムは要求された指導機能にARPANET を介して接続し，ユーザとその指導機能間にトランスペアレントなインタフェースを提供するものである。現在，ACQUIREコマンドのシンタックスは下記のようになっている。

ACQUIRE ＜tutorial－name＞〔AT＜site－name＞〕

要求された指導機能サービスが複数ＨＯＳＴに存在する場合は，＜site－name＞ 句を指定しなければならない。ＲＥＸシステムは指導機能との間にコネクションを張り，ユーザと指導機能が存在するＨＯＳＴシステム間のインタフェースとして動作するわけである。という事は，ＲＥＸシステム自身が指定された指導機能のユーザとなって動作する訳でこのインタフェースは，ＨＯＳＴシステムの特殊性から真のユーザをシールドする機能となる。

指導機能サービスから出されたのではないメッセージ，たとえば，管理プログラムからのエラー・メッセージなどは，ＲＥＸできえぎられ，実行可能なように処置される。ユーザが指導機能から抜け出る指定をすると，ＲＥＸは自動的にコネクションを切断し，ＲＥＸコマンドのレベルまで戻るようになっている。

現在，考えられているＲＥＸの拡張は，すべてのリソースを獲得できるようにすること，特にネットワーク内のユーザ・サービス機能はすべて獲得できるようにすることである。

(5) ま と め

ARPANET のような汎用の異機種コンピュータ・ネットワークにおいては，そこで提供されるサービスが数多く，多種多様であることと，そのリソースとユーザが遠く離れているということから，ユーザ・インタフェースに関して特別な問題が生じる。このような問題はコンピュータ・ネットワークの進展に大きな影響を与えるので，ネットワーク上の技術で現在検討すべきことは，このようなユーザの援助機能の必要性であると強張されている。ネットワークへのアクセス法，アカウンティング・リソースの使用手続などのレベルのユーザ・インタフェースの標準化が試みられてはいるが，このような自律性を重じる，同盟関係のうすいネットワークにおいては，その達成は困難である。ＲＥＸのようなシステムがネットワークをある統一的なものとして扱い，個々のＨＯＳＴの独特の特徴部分をユーザにかかわりなくしていくにはこれからもかなりの期間を要するであろう。現状では，このような試みの完全な実現はネットワークのハードウェア・アーキテクチャおよびＯＳの違いによる難しさの為まだ達成されていない。現時点では，ARPANET におけるＨＯＳＴのようなシステムの異質性をユーザから完全にみえなくする事が可能かどうかは明らかではない。このようなトランスペアレンシーは異質のＨＯＳＴシステムの強みを不明瞭にする為，ある場合は望ましくないかもしれない。しかしながら，コンピュータ・ネットワークの現時点の状況では，ユーザ・サービスの要望を十分満足させ，特殊なＨＯＳＴのサービス内容を理解させるような道具を提供する為に力を注ぐ事は必要である。CODASYL のComm - and Language Standardization Groupe により進められているＨＯＳＴ間のコマンド言語の標準化は，集中形のリソース管理機構を非常に単純化するものではあるが，異機種システムを同一にふるまうように扱うＲＥＸのようなリソース・マネージャとはおもむきを異にすると思われる。

### 6.2.4 ネットワーク・コマンド言語のアプローチ（参考文献　文献1，文献9，文献10）

#### A. Network Command Languageの考え方

Network Command Language（ＮＣＬ）の設計およびインプリメンテーションの考え方には2つの異なったアプローチがある。

① ユーザが，自分の使用するコンピュータのコマンド言語の使用法を習得しておき，それと同時に，そのシステムの呼出しを行うための共通の手続に相当するＮＣＬを使用するというアプローチ。

この方法は，既存のコマンド言語の外側に小量のコマンド言語を解釈する機能を追加するだけなので，ＮＣＬのインプリメント上は最も容易である。

しかし，ユーザ側の観点からいうと，遠隔地のＨＯＳＴで提供されるサービスがごく限られている場合や，受信側のＨＯＳＴのリソースが極端に少ない場合以外は，このようなＮＣＬ機能のみでは満足できないであろう。このようなアプローチをとっている設計者達はこれらの制限を十分認識しており，ＨＥＬＰ等のTutorialな機能を用意するなどして，ローカルなコマンド言語をユーザに使いやすくするよう配慮している。

この種の言語設計のアプローチの例としては，ARPANETのTELNET 言語があげられる。図6－7に一例を示す。これは，ARPANETのUtah大学のＰＤＰ－10とＳＲＩのＸＤＳ－940間でTELNETを用いて会話を行っているものである。

```
•LOGIN                          }
•R TELNET                       } PDP-10 Command Language
ESCAPE CHARACTER IS             }
CONNECT TO SRI                  } TELNET
@ENTER CARR.                    }
@CAL.                           }
CAL AT YOUR SERVICE             } XDS-940 Command Language
@READ FILM FROM NETWORK.        }
 NETWORK:DSK:MYFILE.CAL   TELNET
      .                         }
      .                         } Additional XDS-940 Commands
      .                         }
```

図6－7　TELNETの会話例

② ネットワーク内のすべてのコンピュータにおけるすべてのサービスが利用できるような統一されたコマンド言語としてのＮＣＬを用意する。

完全な意味での汎用ＮＣＬ（すべてのコンピュータのコマンド言語の単なる寄せ集めとして

ではなく）の開発はおそらく不可能であろう。というのは，各コンピュータの機能範囲や操作法は多種多様であり，それらに対する要求やサービスというものは，単一のＮＣＬではカバーしきれないと考えられるからである。

このようなアプローチの例としてはＩＢＭ NETWORK／440におけるＡＣＬ制御言語，ＳＯＣネットワークのＮＣＬ等があげられる。ＡＣＬのコマンド例は図６－８に示してあるが，ここで，ネットワークに対するコマンドはＡＣＬコマンド言語のステートメントの集りとして与えられている。ネットワークのコントローラが中央システムに存在し，これらのプログラム・ステートメントを解釈し，ネットワークを通してのジョブやデータの転送を管理する適当なコマンドを発信するようになっている。

```
label:  ROUTE  n  FROM  unn1  TO  unn2
        OUTPUT n  FROM  unn1  AT  unn2
        EXECUTE  n  FROM  unn1  AT  unn2
        READ  dsn, var
        WRITE  dsn, var
        IFXX  var2, label        (XX=EQ, NE, LT, GT, GE, LE)
        GOTO  label
        ASSIGN  var1, var2       where unn=users network node
        START                            n=name of process or data file
        END                            dsn=destination
                                       var=variable
```

図６－８　ACL Syntax

アプローチの方法がどうであれ，ＮＣＬとして持たなければならない特性は以下のようなものであろう。

① プログラムおよびデータファイルの創成および定義ができること。

② 使用可能なネットワーク・リソースのすべてにアクセス可能で，現在未使用のリソースが獲得可能であること。

③ ユーザにとってトランスペアレントなネットワーク・プロトコルを用意し，ネットワークのリソースをローカルなリソースを使用するのと同程度の容易さでアクセスできるようにすること。

④ プロセスが生きている間は，いつでも呼出し可能であること。これは，プロセスの実行前，実行中，実行後のいつであってもネットワーク・リソースを働かせる事ができるようにすることを意味する。

⑤ プログラマブルであること。これは，実行時にその制御の流れを実行時変数，ラベル，条件

・無条件ジャンプ等を用いる事により，動的に変更できるようになっていなければならないということである。

ＮＣＬというものは，ネットワークのアーキテクチャや考え方の中にあらかじめ含んで考えないことには，実現し得ないであろうことは明らかである。

いずれにしても，理解し易く，容易に使えるということが，ネットワーク全体の利用度に他の何よりも大きく影響するであろう。

## Ｂ．ＳＯＣネットワークにおけるＮＣＬ

(1) はじめに

ＳＯＣとは＂Systeme d'Ordinateurs Connecte's＂（Connected Computer Systems）を意味しており，ＩＢＭ France と大学および政府研究所とで共同開発されているネットワーク・プロジェクトのことである。

ＳＯＣネットワークはリソース・シェアリングを目的とする実験的なコンピュータ・ネットワークであり，その研究の主力は，ネットワークを通して実際の仕事をする際の各種のファシリティを提供する為のソフトウェア技術を見出すということにある。通信技術に関する研究は，既にARPANET 等で十分行われているので，ＳＯＣの研究対象からは外されている。従って開発対象としているソフトウェアは，ローカルなＯＳとネットワークのインタフェースおよび，独立した自律的なＯＳ間でのロジカルな通信などを行うものである。ユーザからの要望は，バッチあるいはインタラクティブ両モードを許すようなネットワークコマンド言語（ＮＣＬ）として対処している。

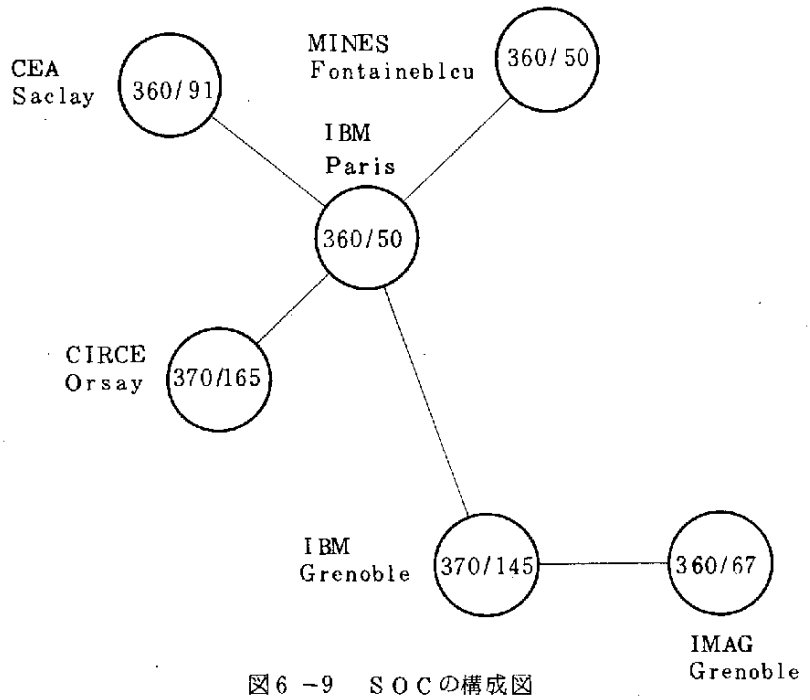



図6 −9　ＳＯＣの構成図

ＳＯＣネットワークの構成を図6 －9に示す。この構成は，一見スター型のネットワークに見えるが実際には分散型である。ＩＢＭはParisとGrenobleとに2つのＨＯＳＴを有し，IMAG，MINESは大学，ＣＥＡ（Atomic Energy Commission），CIRCE（National Research Center）は政府の研究機関である。

機種はすべてＩＢＭマシンなので，ハードウェア・レベルでは同機種であるが，ソフトウェアはＯＳのレベルが異なるので異種と見なして良い。特に，ＩＭＡＧのＯＳはＣＰ／ＣＭＳという特殊なＴＳＳモニターを使用している。

(2) システムの特徴

ネットワークに対する仕事の依頼方法としては2種類ある。その一つはバッチ・モードであり図6－10の左側に示されており，もう一つはインタラクティブ・モードであり図6－10の右側に示されている。

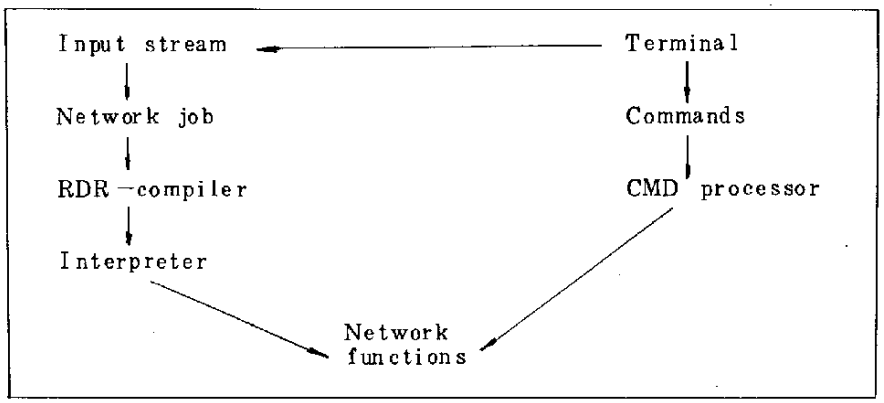
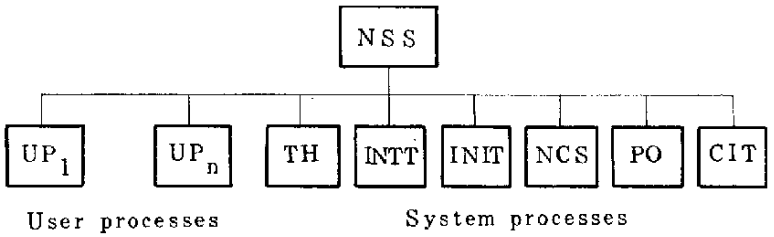



図6－10　Network operations

バッチ・モードのユーザはまずネットワーク・コマンド言語を用いて，ネットワーク・ジョブを組立てる。このネットワーク・ジョブはネットワークに対する処理依頼に関するコマンド・ステートメントの集りとして構成されている。次に，このネットワーク・ジョブはreader Compilerにより，ある種の内部言語の型式（場合によってはネットワーク・マシン型式に）に翻訳される。ネットワーク内の各サイトには，インタプリタが在り，この内部言語を理解し，ネットワーク・ファンクションを呼出すことにより内部コマンドを実行する。

一方，インタラクティブ・モードでは，ターミナルからの同様の機能があり，ターミナル・ユーザがネットワーク・ジョブをバッチ・モードで転送することもできるし，リアル・タイムに動作するような，コマンドを直接送りバッチのインタプリタと同様のコマンド・プロセッサによりネットワーク・ファンクションを呼出すことができるようにもなっている。

図6－11は各サイトにインプリメントされているネットワーク・システムの概要を示すものである。




Reader--compiler

Interpreter

Internal reader

Writer

Utilities

図6 −11　Network system

ＩＢＭ　ＯＳが使われているサイトでは，ネットワーク・システムは１つのジョブとして実行され，360／67ではＣＰの下での１つの " virtual machine " として実行されている。ネットワーク・システムは，プロセスの実行時間や記憶領域を管理したりプロセスの創成を行う為のプロセス・ディスパッチャとしてのネットワーク・システム・スーパバイザにより構成されている。プロセスの種類としては，システムとユーザ・プロセスの２つのタイプが決められている。システム・プロセスはレジデントであり，そのジョブがイニシエートされた時に創成される。これには，まずコミュニケーション・インタフェース（ＣＩＴ），ＡＲＰＡにおける logger に相当するプロセス（ＰＯ）があり，ネットワーク・システムが起動されると各サイトのＰＯ間に論理的リンクが確立され，このリンクは制御用のリンクとしてあつかわれる。次にオペレータとネットワークのインタフェースとしてのネットワーク・コンソール・プロセス（ＮＣＳ），ユーザのネットワーク・ジョブをスケジューリングするイニシエータ・ターミネータ・プロセス（ＩＮＩＴ，ＩＮＴＴ）がある。ＩＮＩＴ，ＩＮＴＴは，ネットワークとローカルなＯＳとのインタフェースとなり，ジョブの依頼や処理結果の引取り等を行う。ＴＨは，ターミナルがないシステムに対するターミナル・ハンドラであり，ユーザ・プロセスの一部として動作するようになっている。

ユーザ・プロセスとしては，まず reader　compiler，interpreter，internal reader がある。internal reader は他のサイトからの内部言語コマンドを受取る為に使用される。さらにネットワーク・ジョブの出力を行うための writer や，特殊目的用の機能に応じた機能を有する多様なプロセスがある。

図６－11はこれらの関係を示している。

(3) ネットワーク・コマンド言語

ネットワーク・コマンド言語は，既存ＯＳのジョブ制御言語の特殊性を，ユーザに意識させずネットワークのどの地点からでも同一の方法でアクセスできるようにするものである。図6－12にその例を示す。

```
Network job definition;
   NETIN name,account;
   ...
   NETOUT;
Declarations;
   KEYWORD ID := DESIGNATION
   CPU       A := SAC91,   B := IMAG;
   UNIT      C := A/2314,  D := IMAG/TAP9;
   VOL       E := C/MVT111;
   DSN       F := E/NETFILE;
   MBR       G := (X)/F ;
   INP    JOB1 := '//.' ;
                 ...
                 ...
                 '//.' ;
   SPACE    S1 := (TRK,160,100,20);
   BLKSIZE
   LRECL
   DSORG
   REC      R1 := (FB,800,80);
```

図6－12　ネットワーク・コマンド言語の例

コマンドと変数はＯＳのジョブ制御言語から派生しているが，言語としては制御言御としての機能を十分満足するような体系になっている。

このコマンド言語には，宣言文とコマンドの２つのタイプのステートメントがある。宣言文はコンピュータやファイル，ボリュームといったネットワークのエンティティをエンティティ変数により扱うことができるようになっている。

ネットワーク・ジョブは，NETINコマンドで始まり，NETOUTコマンドで終了する。宣言文の一般型は，キーワードとそれに続く識別情報から構成されており，識別情報には修飾の指示ができる。例えば，図6－12の例のＣＰＵはコンピュータを定義するキーワードであり，Ａとい

う名前でSaclayにある360／91を表現している。またUNIT CはAすなわちSaclayの2314示している。

また，ユーザが自分で用意したジョブ・デックを入力するためには，キーワードINPとジョブ名（JOB1），それにデックの区切りを示す記号列を宣言し，その後にジョブ・デックを用意すれば良い。

その他，ファイルに関するエリアの容量，ブロック長，レコード長，ファイル編成，レコード様式等を示すパラメータを指定し，ファイル情報を宣言するものもある。

図6－13はコマンド機能の例である。

まず，ファイルに関するステートメントは，例で明らかなように，ファイルの創成，消去，内容消去，ヴォリュームのマウント，ディスマウント，指定された場所へのファイルのカタログ，カタログ消去，ファイル内容の印刷，ファイルの他の場所へのコピー等の機能に対するものが用

```
File statements:
        CREATE    DS1 (attributes);
        DELETE    DS1 ;
        EMPTY     DS1 ;
        MOUNT     MVT111;
        DISMOUNT  TAP9;
        CATALG    DS1 ON CPU1;
        UNCATALG  DS1 ON CPU1;
        LIST      DS1 AT IMAG;
        COPY      DS1 TO DS2 ;
        ADD       DS1 TO DS2 ;

Remote execution:
        RUN JOB1 AT IMAG LISTON SAC91;
        RUN JOB1 ;
        RUN JOB1 LISTON IMAG;

Arithmetic expressions:
        RECL := 2*RECL1+RECL2-16;
```

図6－13　言語機能

意されている。ここで指定されている変数は，前述の宣言文により事前に定義されている。

次に遠隔地での実行を指示するものとしてRUNコマンドがある。図6－13の例における

RUN JOB1 AT IMAG LISTON SAC91 ； というRUNコマンドは，前述の図6－12で示したJOB1というジョブをIMAGで実行しSAC91に出力を印刷する指定である。

実行および出力の場所の指定がない場合はローカルのサイトで処理される。

その他，算術式の表現も可能であり，これは例に示すように主にレコード長などを決定する際に用いられる。

これらはネットワーク・コマンド言語としてはまだ完全とは言えず，スタートについたという状態であるが，ファイル転送やリモート・ジョブ・エントリーには十分満足できる。

(4) ネットワーク内部言語

ネットワークに投入されたネットワーク・ジョブは，ローカル・サイトの reader-compiler により翻訳され各サイトのインタプリンタで解読可能な内部言語（ＮＩＬ：Network Internal Language ）で表現された内部言語プログラムに変換される。図６－14は内部言語プログラムの構造を示す。

NIL PROGRAM STRUCTURE:

| Information block |
|---|
| Data block |
| Instruction block |

INFORMATION BLOCK:

Program length
Instruction counter
Abend instruction pointer
Abend error level

DATA BLOCK:

| Type | Length |
|---|---|
| Identifier | |
| Pointer | |

MBR
DSN
VOL
UNIT
CPU

INSTRUCTION BLOCK:

| Action code | IL | Operands |
|---|---|---|

CREATE DS1 (*attributes*)

| MOUNT | 4 | VOL |
|---|---|---|
| CR | 4 | DS1 |

図６－１４　ネットワーク内部言語

ＮＩＬプログラムは図６－14の例のように情報ブロック，データ・ブロック，命令ブロックの３種により構成される。情報ブロックでは，プログラム・サイズ，命令カウンタ，異常終了時の命令カウンタおよびその時のエラー処置のレベルなどが記述されている。次にデータ・ブロックは，宣言文で定義された各変数のタイプに関する内部表現の情報を保持している。ここには，変数が修飾関係にある時，下位の変数へのポインタも持たれる。さらに，命令ブロックは，命令の種類を示すコード，この命令の長さ，および数種のオペランドからなる。図６－15は，内部言語命令を示すものである。

```
File manipulation
Initial connection
   RFP          : Request for Process
   RFC          : Request for Connection
Synchronization
   SCP          : Suspend Correspondent Process
   HDJ          : Hold Job
   RLJ          : Release Job
   WUP          : Wake up Process
   RFR          : Reguest for Reconnection
Transmission
   SINFO        : Send Information
   RINFO        : Receive Information
   TRMS         : Transmit Send
   TRMR         : Transmit Receive
```

図６－15　内部言語命令

例えばイニシャル・コネクション用に，プロセス創成要求のＲＦＰ，プロセス間のコネクションを確立する為のＲＦＣがある。また複数サイトでの協同ジョブとして処理するのに必要な，ジョブ実行の同期をとるための命令，制御用情報を転送する為の命令，ファイル転送用の命令等がある。

図６－16にネットワーク・ジョブの１例を示す。これはNETINで始まり，まずＢ，Ｃ２つのＣＰＵを定数しファイルＤＳ１，ＤＳ２の存在場所を示し，CPU ＢのＤＳ１をCPU ＣのＤＳ２に転送している。

次に，この新しいＤＳ２を参照するＰＬ／Ｉジョブを定義し，このジョブをCPU Ｃで実行し，最後に更新されたＤＳ２を再度CPU Ｂ側に返送するようなネットワーク・ジョブである。

このようなネットワーク・ジョブがどのような内部形式に変換されるかを示したものが図６－17である。

ネットワーク・ジョブがサイトＡで投入されたとするとreader－compilerは自分自身も含め

て各サイトに転送する内部コードを生成する。

```
NETIN SMITH,A360;
CPU   B := IMAG, C := CIRCE;
DSN   DS1 := SOC.NETLIB/MVT111/2314/B,
      DS2 := SOC.NETLIB/MVT192/2314/C;
COPY  DS1 TO DS2;
INP   JOB1 := '//,/'
//PL1 JOB...
//EXEC PL1FCLG
//PL1.SYSIN DD*
...
...
/*
//GO.DSLIB DD DSN = SOC.NETLIB,UNIT=2314,
//            VOL = SER=MVT192,DISP=OLD
/*
'//,/' ;
RUN JOB1 AT C;
COPY DS2 TO DS1;
NETOUT;
```

図6 −16　ネットワーク・ジョブの例

| A | B | C |
|---|---|---|
| RFP B | | |
| RFC B | | |
| TRMS IL(B) | | |
| SCP | | |
| RFP C | | |
| RFC C | | |
| TRMS IL(C) | | |
| SCP | | |
| WUP B | RSV DS1 | RSV DS2 |
| | RFC C | RFC B |
| | TRMS DS1 | TRMR DS1 |
| | SCP | SCP |
| RFR C | | RFR A |
| TRMS JOB1 | | TRMR JOB1 |
| | | INSERT JOB1 |
| SCP | | SCP |
| | RSV DS1 | RSV DS2 |
| | RFR C | RFR B |
| | TRMR DS1 | TRMS DS2 |
| | SCP | SCP |
| JEND | JEND | JEND |

図6 −17　ネットワーク内部言語のプログラム

この内部コードの例を以下に簡単に説明する。

最初のRFP Bはサイト B に対し内部コードを読むプロセスの起動要求である。これにより A, B に対応するプロセスができる。次に，この2つのプロセス，すなわち，B の内部コードを読む

プロセスとAのインタプリタ・プロセス間でＲＦＣ命令によりコネクションを確立する。これまでのやりとりは前述したＰＯに割当てられたコントロール・リンクを通して行われ，２つのプロセスが結合された時点で制御情報が転送できるようになる。ここでＡはＢに内部言語を転送し，転送が終了すると，Ｂのプロセスは一時サスペンドされる（ＳＣＰ）。同じようなやりとりがＡとＣ間で行われる。Ｂでの内部言語の解釈はＡからＢにプロセスの再起動要求（ＷＵＰ）が送られることによって開始され，それ以降各サイトで同期をとりあいながらネットワーク・ジョブが実行されていく。

(5) インタラクティブ・モードの扱い

ＳＯＣネットワークにおけるインタラクティブ・モードには，次のような機能がある。

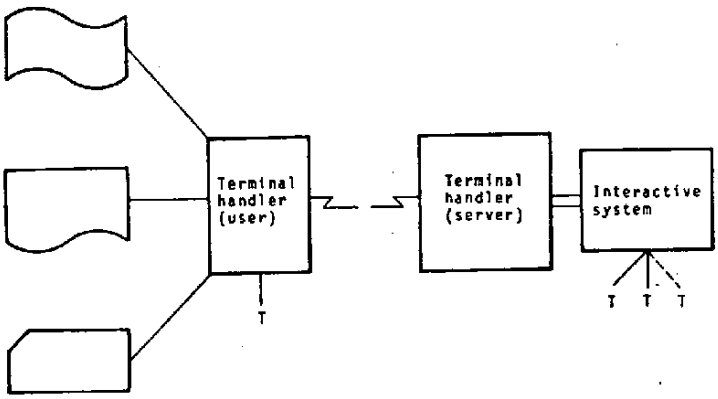
① ネットワーク内のどのインタラクティブ・システムでもアクセスできる。

② 同時に複数のインタラクティブ・システムと接続できる。

③ 出力印刷やカード・デックのようなユニット・レコードのファイルを，リアルタイムで転送できる。

④ ターミナルからネットワーク・ジョブを投入するバッチ機能を有する。

⑤ ある特定のインタラクティブ・システムで使用を許していない端末も，端末側のシステムでシミュレートして機能変換を行うので，サポート可能になる。




```
CONNECT IMAG/IMAG Pnbr
DISCON      /DISCON OFF
ESC         /ESC Char
DEL         /DEL Lchar Cchar
PURGE Device
STACK
REQUEST
STATUS Device
PRINTER SINK = Ctrno SOURCE = Ctrno
PUNCH              "
READER             "
XFER Filename
```

図６－18　インタラクティブ・システム

図６－18に示す例は，ユーザが存在するuser サイトとインタラクティブなシステムでサービスを行うserver サイトの関係を示すものである。

インタラクティブ・ユーザは，カード・リーダ，プリンタ，テープなどのリソースがuser 側ターミナル・ハンドラによりターミナルとして接続されているものとしてみなすことができるようになっており，これらの入出力はスプール機能により行われる。

Server 側では，インタラクティブ・システムからの入出力要求とリモート・サイトのuser 側ターミナルとをある種の変換を施こして結合する。

図6－18の下半分には，ユーザによる指定方法の例が示されており，コネクションの確立，切断，エスケープ文字の定義等の種類がある。

(6) ま と め

ＳＯＣネットワークの開発を通して著者は以下のような拡張計画および今後の見通しを述べている。

まず，ユーザがネットワークをローカルなＯＳの延長とみなすことができるようインタラクティブな機能の拡張が必要である。これまで述べたような方法により，ネットワーク・ジョブはリモートのリソースにアクセスできるが，これに実行中のプログラム間の通信機能を持たせることが考えられる。

次に，ネットワーク自動化の考え方がある。これは，ユーザが，ネットワークの実行場所を明示して指定しなければならない現在の機能を，システムで自動的に指定できるようにすることである。しかし，この自動化の実現には，まだ非常に長い道程が待っているだろうと述べている。

最後に異機種ネットワークにおけるＮＣＬの実現については，データ表現等に関するハードウェア上の違い，ファイル編成等に関するソフトウェア上の違いがあり，多くの問題をかかえている。したがって，異機種ネットワークを考える以前に，ある種の標準化が成されなければならないかを力説している。

**C．異機種ネットワークにおけるＮＣＬの考え方**

Chupinは『Command Languages and Heterogenuous Networks』と題する論文の中で，異機種の分散型ネットワークにおいて，ユーザが分散型アプリケーションを遂行しようとした場合，データの参照と複数サイトのタスク間の同期において，データの存在場所の指定，検索，変換等の問題，あるいは異なるサイトにおけるタスク間同期の要求とその解釈に対するローカルなＯＳとのインタラクション等の困難な問題が待ち受けており，これらを解決するには，ネットワーク・タスクの実行を制御する為に特別に設計されたネットワーク・コマンド言語の実現が必要であると述べている。

このようなネットワーク・タスクの実行を可能にする為には，既存のオペレーティング・システムおよびコマンド言語に下記のような機能追加・修正が必要になってくる。

① 遠隔地のエンティティ（ファイルやリソース）の記述が可能であること。

② コマンド言語はリモート・サイトからのリソース割当の要求に適応しうること。ここで，ロ

ーカルなコマンド言語はリソースの予約と割当の各々に対して，夫々分離されたコマンドを用意する必要があろう。

③ これらのコマンドは，ネットワーク・レベルでのインタラクティブなリソース予約を保障する為，ブロック（予約完了まで制御が戻らない形）されていない形で実行しうること。これは，ローカルな要求とリモートからの要求の両者を受け入れるようなサイトにおけるネットワーク・リソース・アロケーション方法を開発する際には，本質的なことである。

④ ローカルなコマンド言語の修正により，オペレーティング・システム内の機能に対するネットワークからのアクセスを可能とすること。特に，サイト間の同期をとるための状態情報などに対するネットワークからのアクセスが可能でなければならない。

下図6－19はネットワーク・タスクを実現する為に必要となる機能（□内），それに関わるローカルなコマンド言語における問題点（○印の記述）を示すものである。

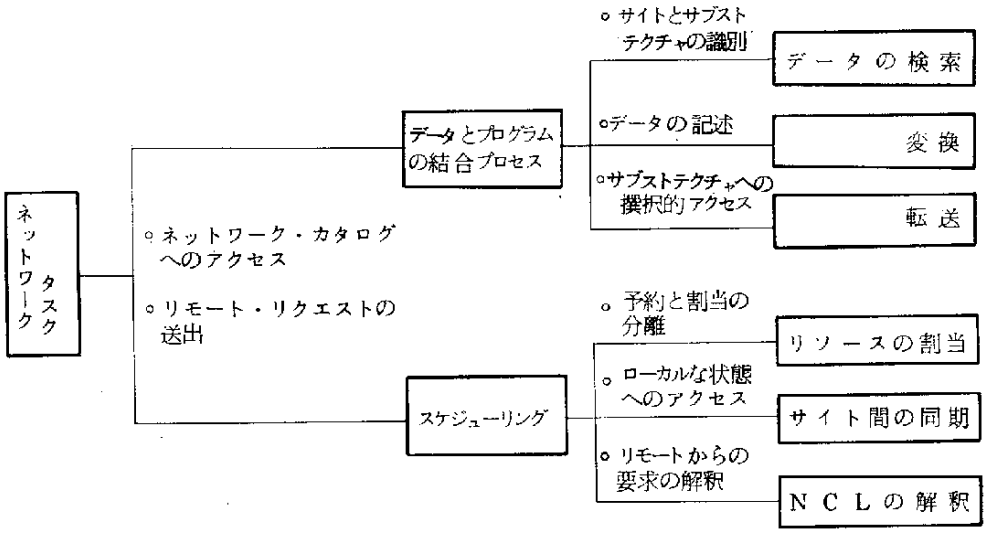



図6－19　ネットワーク・タスクにおける問題

このような異機種間のネットワーク・タスクの実行をユーザにとって実現しやすくする為にChupinはリソースのアロケーションとサイト間同期をとる為のＮＣＬ：Network Command Languageを提唱している。

このようなＮＣＬの実現方法は，図6－20に示すようにトランスペアレントの程度の違いにより異なってくる。

そして，ＮＣＬに含まれなければならない機能として，以下の5点をあげている。

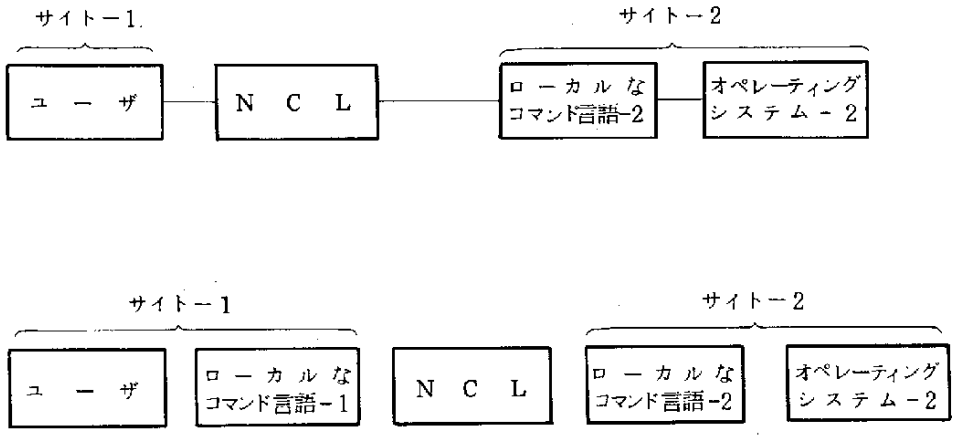



図6－20　ＮＣＬの実現方法

① データ・ファイルの存在場所とロジカルな性質を決定する為のデータの宣言と記述方法がなければならない。データ・ファイルのロジカルな性質の記述に対しては，その為のデータ記述言語がＮＣＬの一部として必要である。

② 実行するプログラムの記述ができなければならない。

これはローカルなオペレーティング・システムとその下でのプログラミング言語の両者に依存している。

③ ユーザは，異なったサイトで同時に実行される複数のジョブのコーディネイトができなければならない。この事は非常に重要であり，もしこれができないと，ネットワークは単なるリモート・ジョブ・エントリ・システムに機能低下してしまう。

④ ＮＣＬは，ファイルやデータ・ベース等に関する高レベルのプロトコルをユーザが容易にインプリメントできるような方法を提供しなければならない。

⑤ ＮＣＬは，ネットワークを単一のコンピューティング設備としてみなせるような機能に対するネットワーク・ジョブ制御言語（Network Job Control Language）をつくりあげる為にも使用できなければならない。しかし，これに必要な制御機能を得る為には，ネットワークの運用に関する定常的でかつ正しい情報を収集し維持しなければならない。この為にはローカルなコマンド言語の拡張を通してアクセスできる汎用のカタログ機能が必要となろう。

### 6.2.5 論理的ネットワーク・マシンによるアプローチ（参考文献　文献15, 16 ）

(1) はじめに

この論文はデータ・バンク・アプリケーション（SOCRATE）の為に使われる論理的ネットワーク・マシン（LNM :Logical Network Machine ）にとって必要な制御のメカニズムについて述べているものである。

コンピュータ・ネットワークにおける今までの研究は，資源共有というよりも通信ネットワークの面が強張されてきており，今後は，ネットワーク・アプリケーションをより発展させる為に高度な制御メカニズムが必要となるであろう。

ネットワーク・サービスとは，ネットワークに対するトランスペアレントなインタフェースをユーザに提供することであると定義できるが，その究極の目標は，コンピューティング設備の集合を，あたかも単一のネットワーク設備のように見えるようにすることである。さらに特定のアプリケーションごとに特別なサービスと制御が必要となるかも知れない。

したがって，ここでは，まず，論理的ネットワーク・マシンの一般的な制御の概念（アプリケーションの種類によらない汎用の概念）について述べ，続いてデータ・バンクアプリケーションに特有の制御の問題について述べる。（注：データ・バンク特有の問題については次章に記す）

ネットワーク・オペレーティング・システムを提示するのがこの論文の目的ではなく，むしろ，各サイトに存在するオペレーティング・システムを適合させ，充実させることが目的である。しかし，ここで提示される概念のいくつかは，将来の設計における指針となりうるであろう。

(2) LNMとは

まず，最初にコンピュータ・ネットワークを，自律的な，多分類似していないコンピューティング・センタ（サイト）が，ある通信設備を介して互いに接続されたものと定義する。

ここでは，ネットワーク・レベルで遂行される機能（function ）とローカルなOSの監視の下に実行されるプログラムに対応しているタスクとは区別して考える。各サイトは，そのサイトに置かれるコマンドと制御機能を取扱う論理的なサブ・システムを含んでいる。通信設備でもって結合されたこれらのサブ・システムの集りが論理的ネットワーク・マシン（LNM ）であると定義される。このLNMがユーザから見た場合，単一のコンピューティング設備であるように見える事がこの論文の関心事である。

(3) LNMの機能

論理的ネットワーク・マシンの機能は，ユーザが規定した機能およびタスクの整合性をとることである。これには2つの問題がある。すなわち

○先ず第1に，各サイトにおいてタスクが使用する情報は，そのタスクの実行に先立ってそのタスクと結合されなくてはならない。この結合のプロセスは，ネットワークにおいては汎用

のマシンにおけるよりも困難であり，データの変換や検索，転送の機構が必要となる。

○第2に，ＬＮＭは１つ以上のサイトで実行されるのであろう複数のプログラムからなるユーザが規定したタスク群に関してその制御と管理の責任を負わなければならない。

このことは，ローカルなＯＳと以下のようなインタラクションがあることを意味している。

① リモート・サイトからの要求を受付け，解釈すること（ネットワーク・コマンド言語）

② タスク群の状態へのアクセスと報告

③ タスク群の実行とタスク間同期の制御

④ リソース割当へのアクセス

⑤ バッチ・モニタの使用等

これらの機能を概略的に示したのが図６－２１である。

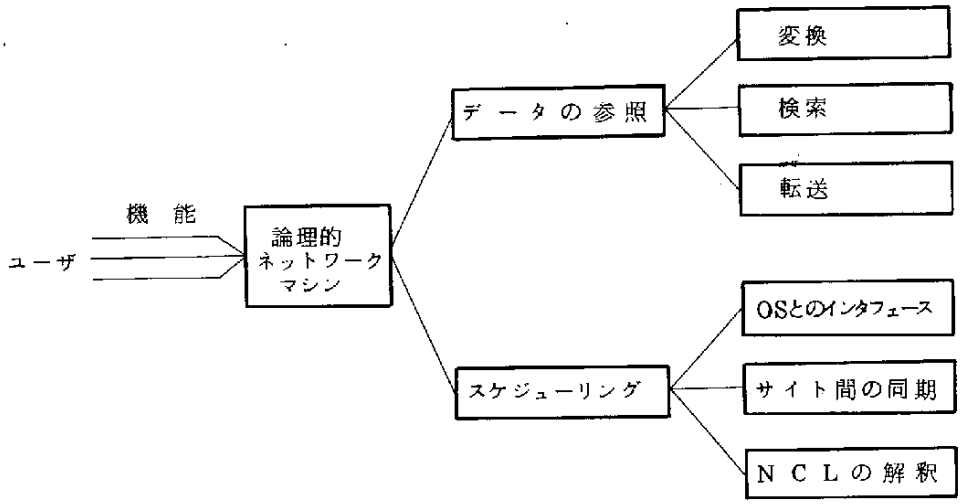



図６－２１　ＬＮＭの主な機能

(4)　ＬＮＭの制御

非集中型（分散型）のネットワークを扱おうとすると，その制御は必然的に広範囲に分散されることとなる。したがって，制御に参加している各ノードにはローカルな制御サブ・システム（汎用のものであれ，あるアプリケーション専用のものであれ）が，対応して存在することになり，全体的なＬＮＭの制御を行うためには，これらのサブシステムの調整が必要となる。この点についてはローカルなＯＳとのインタラクションが深く関係してくる。関連するサイトの識別を含むようないくつかのステップの後，実質的な制御のアクションに移る。この制御アクションは遂行され，その後報告される形をとる。一連の制御アクションが関連しあう時は，遠隔地でのWaitのメカニズム（同期）が必要となる。

ネットワークの環境下では，すべて要求および応答が，サイト間で交換される一定の論理的メッセージの型式をとる。これらのメッセージは，特別なコンポーネントとして扱うこともできるが，メッセージを取扱う機能をつかさどるシステム交換アーキテクチャを設計する方がより望ましい。

このようなシステムの目標を以下に記す。

① 1組のプログラム間でのグローバルなパラメータ（優先権，モードなど）に従いセッションを定義すること。

② 共通のリンクを通してデータと応答の交換が可能なこと。

③ 接続されているリンクの型はトランスペアレントであること。

④ セッション内でのレコードのマルチプレシングが可能であること。

⑤ 半二重，全二重に対して柔軟なアクノレッジ機構を持つこと。

⑥ 要求された動作の完了を知らせるようなPost 機構をつくりあげること。

⑦ セッション内の論理的メッセージ群に相互関係を持たせること。

⑧ 特定の論理的メッセージに対するパラメータを変更できること。

⑨ より高レベルの交換プロトコルが受け入れられるような拡張機能をすること。

この交換アーキテクチャは，プログラムに，そのプログラムが機能として解決しなければならない論理的情報の交換手段を与える。標準的で使用頻度の多い機能については，この考え方は，ネットワーク・コントロール言語へと拡張することができる。この言語は，汎用マシンにおけるジョブ制御言語におけるサービスと同様のサービスを与えるであろう。

図6-22に現存のオペレーティング・システムを使用する論理的ネットワーク・マシンの典型的な構成例を示す。

ここで注意しなければならないのは，機能サブシステムが存在する所には，必ず制御サブシステムが存在しなければならないことであり（逆は必ずしも必要ではない），また，必要な制御機能を用意する為には，ネットワークとその操作に関する情報を定常的にかつ一貫して維持していかなければならないということである。

このようなＬＮＭの概念のより具体的な実例については，次章のSOCRATE の説明において詳しく述べられている。

### 6.2.6 実行ＨＯＳＴの自動選択（引用文献　文献17 ）

(1) はじめに

これまでのコンピュータ・ネットワークに関する研究は，主にコミュニケーション関係に集中しており，最近になってようやく分散型のコンピュテーションの設計に関するものがとりあげられてきた状態である。分散型のネットワークにおいては，あるプロセッサから他のプロセッサに

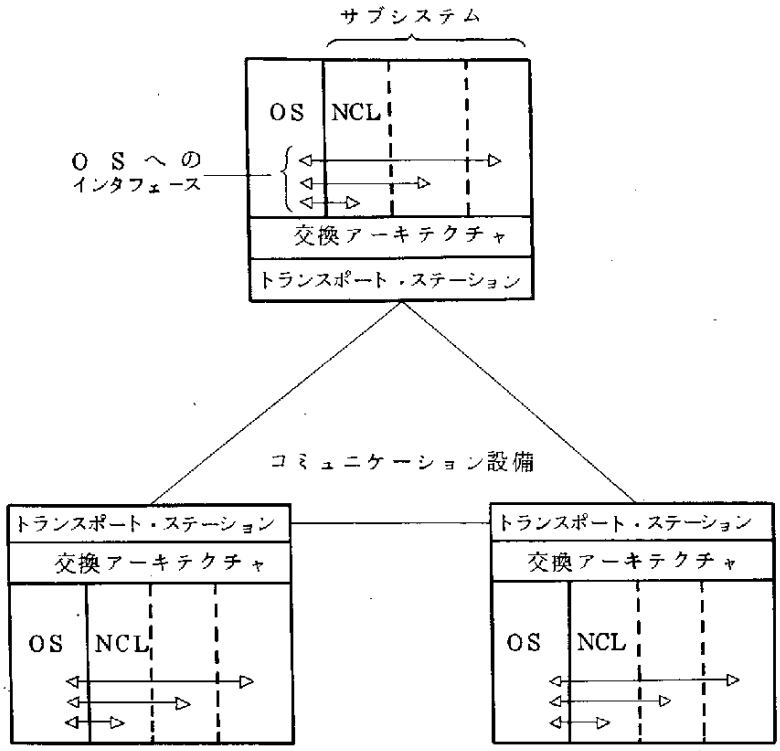



図6 −22　LNMの構成

ジョブを転送するのに，かなり大きなオーバ・ヘッドを考慮しなければならず，今迄のマルチプロセッサにおける研究とは，考え方が異なってくる。本論文では下記のようなことについて検討している。

① コンピュータ・ネットワークにおけるプロセッサ能力と通信システムのスピードを選択し，そのパフォーマンス解析を行うようなネットワーク・モデルの提示。

② ネットワーク内の各リソースで受取られるワークロードを自動的かつダイナミックに分配するようなネットワーク・スケジューリング方法の開発

③ このダイナミックなスケジューリング方法は，ジョブの実行時間を事前に知ることができなくとも，統計的な見積りができさえすれば，非常に効果的であることを示すこと。

④ 分散型あるいは集中型のネットワークにおけるメリットを評価すること。

ここでは，ネットワークを地理的に分散したN個のサイトが，メッセージ交換用通信サブシステム（ＣＳ）により互いに結合されたものと考える。各サイトは処理装置，ユーザの共同体のいずれか，あるいは両方を有するものとする。ここでは，バッチ処理のみを考えるので，ユーザの共同体は，各サイトにおけるジョブ送出のストリームと考えることができる。したがって，ジョブの実行に必要な入力情報は，まず処理装置のサイトに転送され，ジョブ実行による出力情報は，

実行後，一括して返送される。

さらに，ここでは，ネットワーク内のすべての処理装置が互換性を有するような均質ネットワークに限定しているので，どのジョブも，いずれの処理装置でも実行できるが，その処理スピードは処理装置によって異なる。ここで考えるモデルはより大きな均質ネットワーク，あるいは，より汎用的なネットワークにも適用可能であり，例えば，ARPANET における各サイトからジョブが転送されるARPANET ロード・シェアリング・サブネット（互換性のある数ＨＯＳＴから成る）の設計および制御にも，この論文で述べる方法は利用できよう。

ネットワークのスケジューリングは，次のように行われる。ジョブはユーザによって発信地サイトから送出され，実行サイトの処理装置で処理される。実行サイトは送出時にユーザによって選択されるか（ユーザ選択）送出待ジョブ・キューの長さとジョブ・サイズをもとにネットワークにより選択される（自動選択）

この論文の方法では，実行サイトの選択は，送出後は変更しないものと仮定する。Bowdon や Howell 等はジョキーイング（jockeying）やロード・レベリングの方法を提案しているが，残念ながら，そのような方法には，かなり重要な欠点が見うけられる。問題点の一つは，それらの方法には，調整しなければならない２つのパラメータ，すなわち，スーパバイザリ時間とロードの不平衡を決めるためのスレッショールドが存在することである。効果的なバランシングを行うためには，スーパバイザリ時間は少なくなければならず，スレショールドはせまくしなければならない。しかし，これらの値が大きすぎると，ネットワークは不安定になり，不平衡状態を正す為に永久にジョブがやり取りされるといった無益な現象が生ずる。これは，ページング・システムにおけるスラッシング（thrashing）と良く似た現象である。

他の問題は，多数のジョブが一度以上転送されることになるので，通信システムに，かなり高い負荷がかかることである。

各種のネットワークにおいて，送出時に実行サイトをインテリジェントに選択を行うことは，上記のようなロード・レベリングによる方法よりすぐれているという事が本論文の論点である。ロード・レベリングは，対象ジョブの処理に長時間要し，ジョブ転送の頻度があまり高くないようなワーク・ロードの場合は効果的であろうと思われる。

実行サイトの選択という問題は，蓄積交換ネットワークにおけるルーティングと似ており，興味深いことである。ARPANETにおける，各通過ノードで新しいルートを選ぶという方法は，ロード・レベリングの考えと同じである。最近になって，メッセージがネットワークに投入される時点で固定ルートを選ぶという方法の方が良いということがGerla 等により提唱されており，この方法は，この論文の non-jockeying法と同じである。

(2) ネットワーク・モデル概要

この論文で述べられている，実行サイトの選択方法はユーザ・オリエンテッドなものであり，

ネットワークが，各プロセッサに対し，あるプロセッサが選択された場合のターン，アラウンド時間を見積っておき，その中で最小時間のものを選び出すという方法である。

ネットワークの各サイトは図6－23に示すように4要素により構成されるものとする。

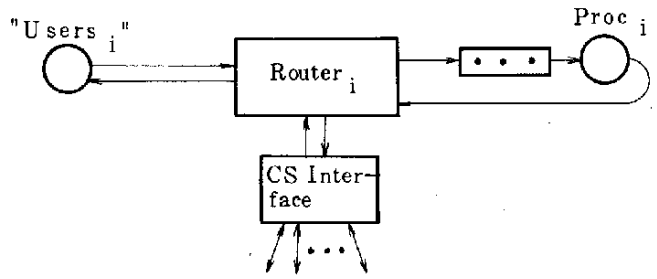



図6－23　サイトiの論理的な構造

ここで"users"はジョブ送出ストリーム（入力）とジョブ返却ストリーム（出力）とで表わされ，プロセッサ（Proc i）は"router"の指示によりジョブを実行するもので，キューはプロセッサの一部として考えられている。"Routeri"はサイトiにおけるネットワークに対する代表者としてみなされるようなロジカルなエンティティであり，他のプロセッサに対するユーザからのアクセスとネットワークの他サイトからの自プロセッサiに対するアクセスをコントロールしている。ユーザが投入したジョブは，"router"により実行サイトが自動的に選択される。Proc iはそのすべてのワークロードを"Router i"から受取るようになっており，そのジョブがどこから送出されたのかなどということは知る必要がない。

ＣＳインタフェースは"Router i"を他のすべての"router"に結びつけている。

実際のインプリメント上は，"Router i"と"Proc i"は図に示されているように独立にはならない。

ローカル・プロセッサのＯＳによって読込まれたジョブ・カードは，ＯＳのジョブ・キューにスタックされる。"router"は，各ジョブの読込が完了した時点で，そのジョブの実行サイトを選択するようなサブタスクになると思われ，リモート・サイトが選ばれた場合は，"router"がそのサイトへのジョブ転送を行う。

"router"は，他サイトからのジョブをProci のジョブ・キューに投入，ジョブ出力の転送・実際の出力印刷キューへのスタックについても責任を持つ。このように，この論文のネットワークモデルは，ローカルＯＳによるカードのフィジカルな読取りと，フィジカルなプリントは除いて考えている。ジョブの送出は発信地サイトでローカル・ジョブ・キューに最後のカードがスタックされた時点で行われ，ジョブの返却は出力の最終行が受取られた時点で完了する。

(3) シミュレーションの結果

以下に，例題によって，分散型の自動選択ネットワークの有利性を示す。

まず，3つのサイトからなるネットワークを考え，夫々のサイトのジョブ送出ストリームは以下のようなパラメータを持つものとする。

| | |
|---|---|
| 入力メッセージ数 | $1/\mu Ii = 100$ message／job |
| 平均実行ステップ | $1/\mu Pi = 32$ Mega Instruction／job |
| 実行ステップの標準偏差 | $\sigma GPi = .30$ |
| 平均出力メッセージ数 | $1/\mu Oi = 150$ message／job |
| 出力メッセージ数の標準偏差 | $\sigma GOi = .30$ |

平均の実行時間は，ＩＢＭ360／65 の場合で約1分位となる。次に，全ジョブ送出率を $\lambda = 360$ jobs／時間と仮定すると，各サイトによって，分布は違うが，通常は $\lambda i = 120$ jobs／時間／サイトとなる。しかし，1つのサイトの送出ロードが高い時は，他サイトの送出ロードは軽いことになる。たとえば，サイト2においてピークロードの50％を送出するということは $\lambda_2 = .50\lambda = 180$ jobs／時間であり，残ったロードはサイト1，サイト3に90 jobs／時間づつ振り分けられることになる。

ネットワークは，送出ロードが一様であるようなケースで，良いパフォーマンスが得られなければならないが，どのサイトでも送出ロードの80％位までは極端なパフォーマンス低下があってはならない。この実験では，すべてのケースで，トータルな送出ロードは360 jobs／時間と固定しており，3つの送出ストリームに対する各ジョブのサービス時間統計は変らないものとする。

以下に，図6－24のように定めた4種類のネットワークのパフォーマンスを，ピーク・ロードを変化させてみて比較する。図6－24の○はサイト，riはプロセッサ・スピードを表し，実線は通信パスを表わし，パス上の数字は1秒当りの転送メッセージ数を示しており，サイト間の伝送遅延時間は0と仮定する。4種のネットワークのトータル・コスト（通信コスト＋プロセッサ・コスト）はすべて同一としている。

NET Aは自動選択の分散型ネットワークである。プロセッサのスピードは適当に1：2：4の比率で分けており，通信路のスピードは大体は一様であり少し早いパスで，遅いプロセッサと早いプロセッサを結合している。NET BはNET Cはサイト3に集中化されたネットワークである。NET Aのサイト3が早いプロセッサなので比較する時に適当であろうと思われる。B，Cは通信路とプロセッサ間のトレード・オフのレベルが異なっている。すなわち，同一のネットワーク・コストであるがNET Bでは早いプロセッサとやや遅い通信路，NET Cはやや遅いプロセッサと早い通信路になっている。

次にNET Dは各サイトが完全に分離しているネットワークであり，比較の為に示してある。

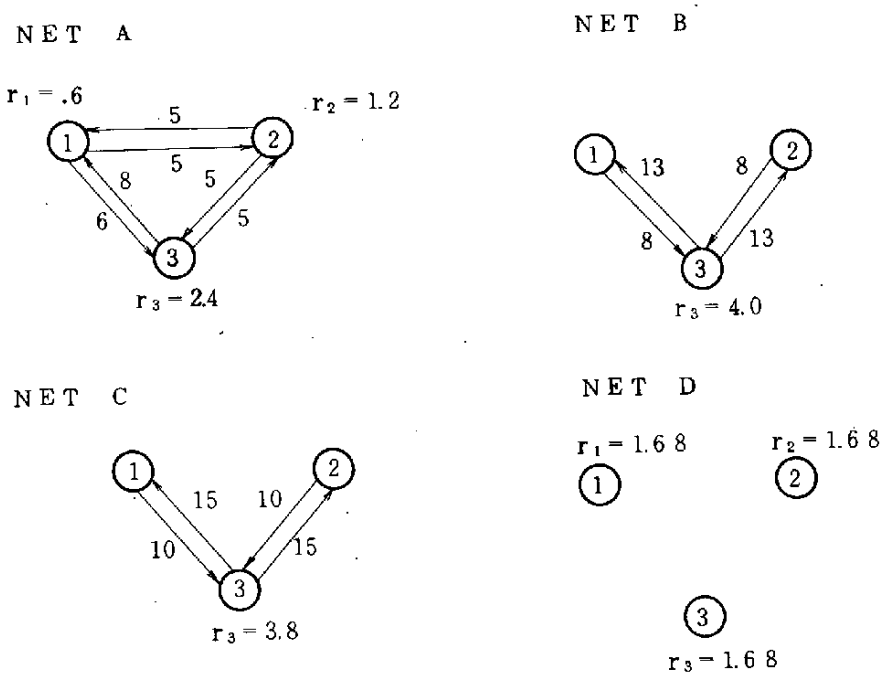



図6－24　4種のネットワークの通信パスとプロセッサ・スピード

通信路の分のコストは3つの結合されていないプロセッサに一様に分配されている。

表6－1はこれらの4種のネットワークのピーク・ロードを変えた場合のターン・アラウンド時間としてのパフォーマンスを示している。

NET Aに対する結果はシミュレーションによって得られており，他のものは，シミュレートされた各サイトのジョブ・ストリームのサンプル平均と標準偏差を用いて求めている。シミュレーションにおいて，3つのジョブ・ストリームのサンプル平均サービス時間は，正確には同一でなく，したがって2つの集中型ネットのパフォーマンスはサイト1とサイト2のピークに対しては異なった値となっている。

表をみてわかるように，送出ロードが一様に分布している場合は（Even），NET Dが最良のパフォーマンスを示しているが，ロードのピークの変化に対しては非常にセンシティブであり，急激にパフォーマンスが低下してしまう。同じく一様なロードに対してNET AはNET Dに最も近く，どのサイトにおいても過度の送出ロードに堪えることができる。集中型のネットワークは，どちらも極端なパフォーマンス低下を起さずに，リモートのピークを処理することはできないがサイト3については期待通り良好な結果を示している。

NET Aで最悪のケースのロードはサイト2のピーク時である。このことは，サイト1が最も

| Peak Load | Net A | Net B | Net C | Net D |
|---|---|---|---|---|
| Even | 65 | 78 | 91 | 59 |
| $Site_1$: 40% | 66 | 81 | 93 | 67 |
| 50% | 68 | 90 | 98 | 1873 |
| 60% | 71 | 111 | 107 | ∞ |
| 70% | 75 | 174 | 122 | |
| 80% | 84 | ∞ | 166 | |
| 90% | 93 | | 263 | |
| 100% | 102 | | ∞ | |
| $Site_2$: 40% | 66 | 81 | 93 | 68 |
| 50% | 67 | 90 | 98 | 204 |
| 60% | 72 | 110 | 106 | ∞ |
| 70% | 78 | 188 | 125 | |
| 80% | 89 | ∞ | 165 | |
| 90% | 99 | | 282 | |
| 100% | 177 | | ∞ | |
| $Site_3$: 40% | 65 | 78 | 88 | 67 |
| 50% | 64 | 68 | 84 | 273 |
| 60% | 65 | 63 | 81 | ∞ |
| 70% | 68 | 59 | 77 | |
| 80% | 71 | 56 | 75 | |
| 90% | 77 | 53 | 73 | |
| 100% | 82 | 50 | 70 | |

表6－1　ピーク・ロードを変えた場合の平均ターン・アラウンド時間（秒）

遅いプロセッサであることを考えると，意外であるようにみえるが，サイト1はサイト3に対して早い通信路を有しているが，サイト2にはそれがないことによると思われる。

サイト2のピーク・パフォーマンスはサイト2→サイト3の通信路を4，サイト3→サイト2を6とすることによって多少改善され，また，プロセッサ2のスピードを少し下げて，その分のコストを通信路の速度向上（2，3メッセージ／秒）にあてれば，おそらくより向上することであろう。

(4) ま と め

自動選択によって制御されたコンピュータ・ネットワークは，パフォーマンスの面と同様に信頼性の面からも集中あるいは完全分離のシステムに比べ優れているのは言うまでもないことである。この実験を通して，次の2点が明らかにされた。

① 自動選択は，分散型ネットワークのパフォーマンスを顕著に向上させている。

② 自動選択は，ジョブサイズに関し統計的な評価ができれば，既知でなくとも有効性を示し得る。

## 第6章の参考文献


(1) Michael Schneider (Wisconsin Univ.)

"A Survey report on Computer Networks"

NTIS PB-231876

(2) Honey S. Elovitz 他(NRL)

"What is a Computer Networks"

NTC 74 PP 1007-1014

(3) John W. Benoit (MITRE)

"Evolution of Network User Services — The Network Resource Manager"

'74 Computer Networks : Trendes and Applications

PP 21-24

(4) Robert H. Thomas (BBN)

"A Resource Sharing Exectives for the ARPANET"

NCC 73 PP 155-163

(5) BBN-NO. 3012 '75. 2

"Distributed Computation and TENEX Related Activities"

NTIS AD/A-006735

(6) R. Rosenthal 他(NBS)

"Automated Access to Network Resource— A Network Access Machine"

'74 Computer Networks : Trends and Applications

COMPCON '73 PP 71-74

(7) Jonn R. Pickens (UCSB)

"Computer Networks from the User's point of View"

PP 71-74

(8) A. J. Neumann (NBS)

"User Procedures Standardization for Network Access"

NTIS-COM-74-50135

(9) S. Girardi (IBM UK)

"The SOC Network : Design and Inplementation"

Network Systems and Software PP-337-357

Infotech State of Art Report 24 '75

(10) Jean Claude Chupin (CII Grenoble)
"Command Languages and Heterogeneous Networks"
Command Langnages, C.Unger Editor.
North-Holland Publishing Company ('75)
PP-311-318

(11) J. du Masle (IMAG)
"An Evaluation of the LE/1 Network Command Language Designed for the SOC Network"
Command Language, C. Unger, Editor.
North-Holland Publisbing Company ('75)
PP 319-326

(12) I.A. Newman (UK)
"Command language design for Networks of Processers"
Eurocomp 75 PP-519-535

(13) P. Schicker (COST 11)
"Job Control in a Heterogeneous Computer Network"
Eurocomp 75 PP537-547

(14) I.A. Newman (UK)
"Towards an effective facilities network using the unique machine independent command language"
Eurocomp 75 PP647-657.

(15) Jean Claude Chupin
"Control Concepts of Logical Network Machine for Data Bank"
Information Processing 74
North-Holland Publisbing Company ('74)
PP 291-295

(16) Jean Clande Chupin
"Distributed Applications on Heterogeneous Networks"
Third Annual Symposium on Computer Arehitecture
Jan' 76 Florida USA

(17) W.D. Roome (Cornell univ)
"Modeling and Design of Computer Networks with Distributed Computation Facilities"

'74 Computer Networks : Trends and Applications
PP30－38

⑱ P.V.McGreger 他
"Optimal Load Sharing in a Computer Networks"
ICC75 PP41－14～41－19

⑲ J.T.Fitzgerald ( Uniu. of Illinois )
"Load Regulation and Dispathing In a Network of Computers"
NTIS PB211 783

# 7. 分散型データ・マネージメント

# 7. 分散形データ・マネージメント

## 7.1 データの共有に対するニーズ

データの共有に対するニーズは，種々の方法によって分類することができる。分類方法の一例をあげれば次のとおりであろう。

(1) 重要性：データを共有するためにどれだけの費用が投入できるかを決める要因となる。

(2) 応答性：オンラインで即時応答か，メッセージ交換か，あるいはオフラインで磁気テープを交換するのかを決める要因となる。

(3) 範　囲：同一企業内，系列企業間，同種企業間，サービス会社と利用者，官庁と同種企業，官庁間あるいは国際的なレベルであるか。運営方法を決める上での重要な要因となる。

(4) その他：実現性など。

ここではまず(3)の分類にしたがい，それぞれの範疇でどのような例あるいは計画，構想があるかをみてみよう。

A. 同一企業内

単に同一企業内といってもその規模はさまざまであり，比較的小規模のものとしては，研究室あるいはグループ内で同一のファイルを使用するものから複数の研究室あるいは，企業で言えば部門間にまたがった程度のものがあろう。

このレベルの例としては，電子技術総合研究所でおこなっているパターン情報処理システムEPICS（Electro technical Laboratory's Pattern Information Computing System）におけるオンライン・パーマネント・ファイルをあげることができる。このシステムの基本思想は，中央に主計算機（Tosbac 5600／170）をおき，チャネル経由で副計算機（FACOM 230／35，HITAC 8350，Tosbac 3400／41，PDP-11，NEAC 3200／50）と結合し，主計算機はコンピューティング能力および大容量ファイル，副計算機では種々の特殊装置および局所処理を担当させるものである。

企業全体にわたる業務において，データを共有をおこなっている例としては，住友銀行におけるオンライン・バンキング・システムなどがある。

全国規模にわたる同一企業内のネットワークとしては，旭化成のＡＣＴをあげることができる。

ＡＣＴの背景と目的は次のようなものである。

旭化成は，札幌から宮崎にいたるまで２本社，14事務所，７研究所，31工場を持ち，全事業部が全国にちらばっており，これらの事業所が多種多様な商品をとりあつかっている。これらの

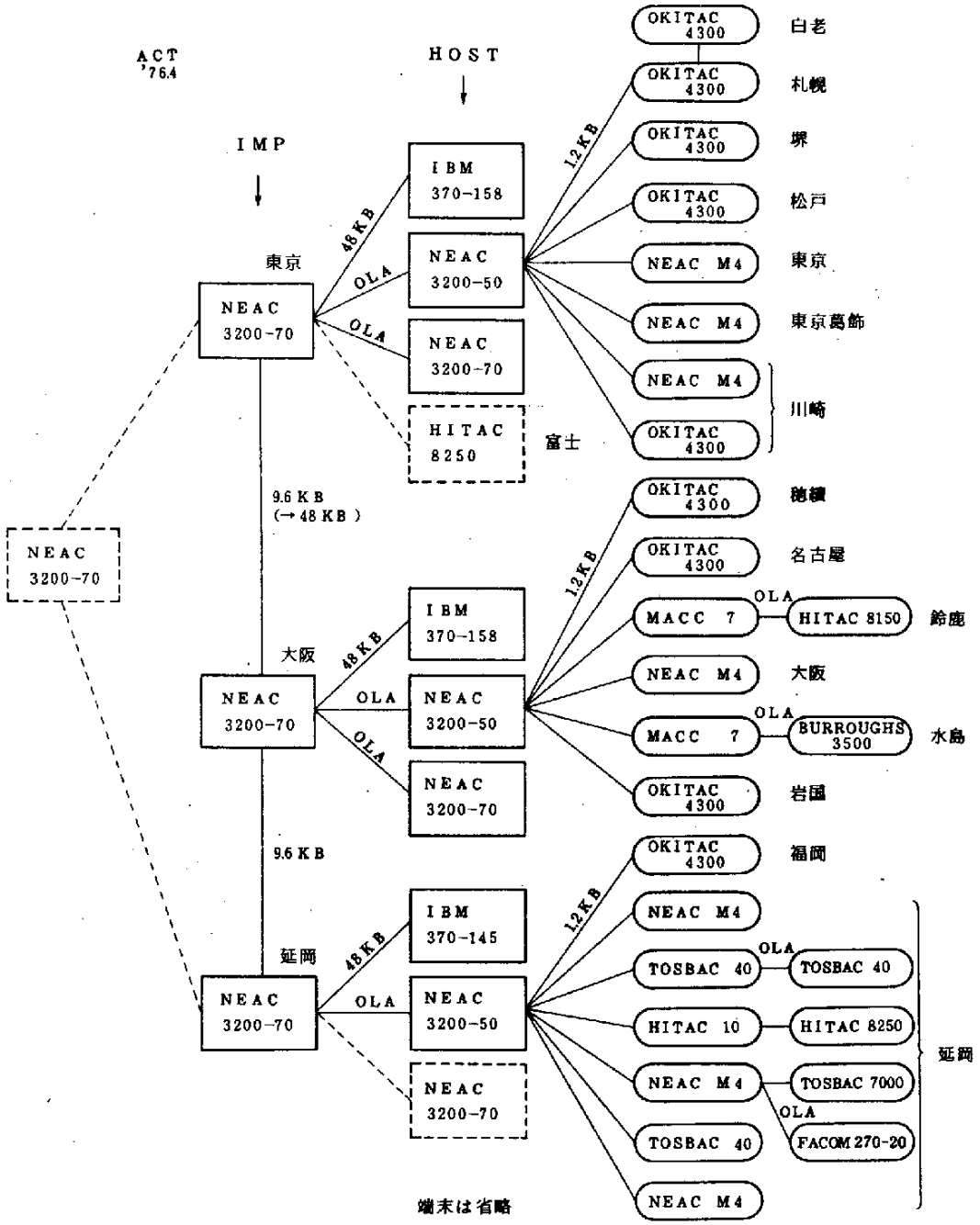



図7－1　ACTのシステム構成図

事業所における情報処理は自然発生的に生成してきた。その結果として，多種多様の汎用および制御コンピュータが各地に導入されてきた。

また社内情報の伝送のために多数のテレタイプを使用していたが，中継など途中で人が介在するために能率が良くなかった。

このような状況の下で，ＡＣＴは２つの目的を持って開発されることになった。

第一番目は，全国規模の業務を東京，大阪の本社コンピュータに集中し，生産関係の業務は工場が属する地区の汎用コンピュータに集中することである。すなわち，ＡＣＴは，全国に散在する汎用コンピュータ相互間の通信路および，同一地区内の汎用コンピュータと制御用コンピュータまたはミニ・コンピュータ相互間の通信路となってファイル転送，データのコード変換をする役割をはたすことである。

第二番目は，テレタイプ網の中継の自動化をはかり，省力化および業務の質の向上をめざすものである。

図７－１にＡＣＴシステムの構成を示す。

B．同種企業間

同一企業内での情報処理のコンピュータ化がうまくいったときに，より高度なサービスあるいは迅速，的確な情報交換のためにとり上げられたのが同種企業間のデータ共有である。比較的早い時期に作られたものとしては，地銀協の為替交換システムがある。また近畿日本ツーリスト・オンライン・ネットワークおよび日本キャッシュサービスのシステムもこれに該当するものである。

前者のシステムは，日本航空（ＪＡＬ）のＩＢＭ 360／65，全日本空輸（ＡＮＡ）のHITAC8500，東亜国内航空（ＴＤＡ）のNEAC 2200－500，近畿日本鉄道のUNIVAC 418 Ⅱの４台のコンピュータがそれぞれおこなっている座席予約サービスを近畿日本ツーリストの端末から利用しようとするものである。

後者のシステムは，キャッシュ・ディスペンサーの共通利用を目的とするものである。すなわち都市銀行13行，地方銀行11行，相互銀行12行の各コンピュータ・センターと端末とをオンラインで結合し，参加銀行のＣＤカードのいずれであっても，その銀行のコンピュータ・センターの預金ファイルをチェックすることによって，現金の支払いをするものである。

同種企業間でのデータの共有は，サービス・レベルでは非常に進んでいるが，技術レベルでは端末装置レベルの共有をおこなっているだけである。技術レベルとしては低いところにおさえてある理由としては，現状のサービス・システムを変更しないで最も簡単におこなう方法としては端末装置によるデータの共有が最適であることによる。

C．サービス会社と利用者

種々の利用者がデータの共有をおこなう場合に，直接利用者間でおこなうこともあるが，特定

のサービス会社を経由してたがいにデータを共有する方法もある。

同種企業間でのデータの共有の例としてとりあげた，地銀協システムなどはこの例である。これ以外にも，ＴＳＳベンダーを利用してのデータの共有が考えられる。

ここでは米国のＴＳＳサービス会社であるTYMNETにおける利用例をみてみよう。この例の中で最も興味深いのは，医療データ・バンクをＴＳＳによって利用できることである。

TYMNETは，TYMSHAREという名のタイムシェアリング会社が運用サービスをしている分散形コンピュータ・ネットワークである。TYMNETは２つの目的を持っている。

(1) TYMSHAREの商用タイムシェアリング・サービスに遠隔地からのアクセスをサポートする。

(2) ユーザが自分自身のコンピュータに端末からアクセスできるようにする。

このネットワークは，70都市に90台のノードを持っており，その一部はパリ，ブリュッセル，ロンドンまでサービスされている。このネットワークの特徴は，表７－１に示すように種々のコンピュータ・メーカのコンピュータを設置しており，利用者はこのうちのどれでも使用することができる。

使用者からみたときにTYMNETは，端末から利用できるタイムシェアリング・システムであるが次のようなアプリケーションを持っている。

表７－１　TYMNETの処理コンピュータ

| |
|---|
| ＩＢＭ　model 360／40，50，65，67 |
| ＩＢＭ　model 370／155，158 |
| Burroughs　B 6700 |
| Control Data　6400 |
| ＤＥＣ　ＰＤＰ10，9，15 |
| ＸＤＳ　940 |
| Honeywell　G 275 |
| ＵＮＩＶＡＣ　1108 |
| ＸＤＳ　Sigma 7 |

(1) 科学技術計算

(2) オーダ・エントリ

TYMNETを使用することによって，セールスマンが客先で，在庫量などのチェックができる。

(3) 情報検索

いくつかの主な情報検索システムがTYMNETにリンクされ，大量データ・ベースに対する

遠隔地からのアクセスができる。たとえば，Battelle Memorial InstituteはBASISという大量情報検索システムを持っており，これは$10^9$文字以上のデータ・ベースであるが，TYMNETからアクセスできる。また，政府が援助している医療データ・バンクであるMEDLINEもTYMNETからアクセスできる。

D. 官庁間

行政官庁では，それぞれ個別に情報処理機能を持っており，主として個別業務あるいは統計集

表7－2　官公庁でのデータの交換例

| データ使用官庁 | 業務名 | データ名 | データの入手先 | オンライン方式による | オフライン方式による | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | 磁気テープ | 紙テープ | カード | その他 |
| 警察庁 | 自動車登録番号照会 | 自動車登録ファイル | 運輸省自動車局 | ○（予定） | | | | |
| 経済企画庁 | 経済分析業務 | 交通モデル | 運輸省 | | | | | ○ |
| 〃 | 〃 | 産業連関データ | 通商産業省 | | | | | ○ |
| 北海道開発庁 | | 道路関係データ | 建設省，北海道庁 | | ○ | | | |
| 科学技術防災センター | 地震解析業務 | 強震計データ | 強震計設置場所 | | | | | ○ |
| 〃 航空宇宙研究所 | 機体実験業務 | 機体関係データ | 東京大学金属材料技術研究所 | | | | | |
| 〃放射医学総合研究所 | がん治療業務 | 病歴ファイル | 国立ガンセンター | | | | ○ | |
| 法務省 | 日本人出帰国管理業務 | 旅券発給データ | 外務省 | | ○ | | | |
| 外務省 | 貿易統計検索 | 大蔵省通関統計 | 大蔵省関税局 | | ○ | | | |
| 大蔵省 | 貿易統計業務 | 通産省貿易統計 | 通商産業省 | | ○ | | | |
| 社会保険庁 | 社会保険料徴収業務 | 被保険者資格記録データ | 東京都各社会保険事務所 | | | ○ | | |
| 通商産業省 | 通商白書作成 | 通関統計 | 大蔵省 | | ○ | | | |
| 〃 | 情報蓄積加工サービス | 卸売物価指数 | 日本銀行 | | ○ | | | |
| 〃 | 〃 | 米国輸入，海外市場 | ＪＥＴＲＯ | | ○ | | | |
| 〃 | 〃 | 国連統計，ＯＥＣＤ統計 | アジア経済研究所 | | ○ | | | |
| 特許庁 | 特許情報検索 | 検索用蓄積データ | 外国特許局 | | ○ | | ○ | |
| 文部省緯度観測所 | 星庫の編集 | 星の位置 | 東大東京天文台 海上保安庁 | | ○ | | | |
| 運輸省情報管理部 | 内航海運輸送統計 | 内航海運輸送データ | 総理府統計局 | | ○ | | | |
| 〃 〃 | 観光出入国統計 | 観光出入国データ | 〃 | | ○ | | | |
| 〃 東京航空管制部 | 航空管制業務 | 気象観測データ | 気象庁 | | | | | ○ |
| 気象庁 | 気象予報業務 | 気象データ | 外国気象機関 | ○ | | | | |
| 海上保安庁 | 天体位置計算 | 星のカタログ | 米国天文台 | | ○ | | | |
| 〃 | 水路編纂 | 海洋観測データ | 気象庁，水産庁 | | | | | ○ |
| 〃 | 天体観測 | 天体観測データ | 天文台，国土地理院 | | | | | ○ |
| 郵政省 | 電波監理業務 | 国際周波数登録データ | 国連国際電気通信連合 | | ○ | | | |
| 建設省 | 工業用水解析 | 工業統計 | 通商産業省 | | ○ | | | |
| 建設省国土地理院 | 天文測量計算 | 星表 | アメリカスミソニアン天文台 | | ○ | | | |
| 〃 〃 | 地形補正 | 地形データ | 東大地震研究所 | | | | ○ | |
| 〃 建築研究所 | 構造物の地震応答 | 各地強震記録 | 防災センター | | | | ○ | |
| 自治省 | 交付税計算 | 国調データ | 総理府統計局 | | | | | ○ |
| 〃 | 〃 | 道路河川台帳 | 建設省（予定） | | | | | ○ |

計といった作業を行なっていたが，官庁相互のタテ割り意識が，お互の情報流通を妨げてきたため，政策立案に必要な情報を迅速的確に他機関から入手することは困難であった。このため，同じような調査・集計を複数の省庁で行なうなどの無駄も多かった。

このような背景はあるにしろ，省庁間におけるデータの共有（データの交換）は，表７－２に示すような状況で存在している。

官庁間の個別の情報処理機能は，行政サービスの簡素化と質の向上という面，政策情報の必要性という２つの面から，複合化という構想が生じてきた。

これらの構想は，各省庁において異なったものになっているが，大別すると４つの形になると考えられる。

第一番目は，行政管理庁の提案している行政情報通信ネットワーク（Administrative Information Communication Network：AICON）に代表されるものである。これは，データ交換網の共有化を最初におこなっていくものである。データ交換網の共有化は，通信費用の軽減をはかるばかりでなく，同一の網に各省庁のオンライン網を結合することになり，コンピュータの共同利用，データの共有などを推進する大きな力となるであろう。

第二番目は，総理府統計局などが提案しているデータ・バンク構想である。行政官庁では，各種統計データを持っており，これらを各官庁間で総合的に利用しようとする手段をあたえることである。この構想は，各省庁などがそれぞれ保有するデータを個別のデータ・バンクという形でとらえ，これをネットワーク化することによって，総合的なデータ・バンク化を目ざすものである。このために，個別情報の総合化の手順として分散形ネットワーク，ネットワークにおけるデータ・バンク情報センター（クリアリング・センター）という方向を持つものである。

第三番目は，既にデータ交換をおこなっている省庁間で，ネットワーク化をはかってゆくものである。これらの動きとしては，通産省と経済企画庁の間などにみられる。

第四番目として，工業技術院内にある「電子計算機利用に関する技術研究会」がおこなっているものであり，50年度より，リソースシェアリング研究班を発足させ，省庁間におけるコンピュータ・ネットワークの形成に意欲的に取り組んでいる。

E．国際的なレベル

国際的なレベルでのデータの共有としては，気象観測データの交換などがある。ここで紹介するのは，ヨーロッパにおいて，科学技術に関する情報ネットワークを作ろうとしているEURO－NETである。

1971年に，ヨーロッパの閣僚会議において，科学技術情報やドキュメンテーションに関する協力についての合意がみられた。これにもとづいた検討の結果75～77年の間に，次の３つの分野の活動をすることが企画された。

(1) 種々の分野（農学，物理学など）におけるシステムの開発作成

(2) 情報処理のための物理的なネットワークの作成

(3) 情報工学における技術と道具の開発

EURONETは，長期間の計画であり，利用者数は1976年に6万，80年に96万，85年に235万人に増加することを前提として作られている大がかりなものである。

## 7.2 データの共有に関係する事項

現在各方面に於てデータの共有に対する強いニーズがあり，これはコンピュータ・ネットワークにとっての最も重要な目標の1つであることはすでに述べたとおりである。この目標を達成するために，これに関する種々の研究，実験，実用化が現在世界的に多数おこなわれている。この項では，分散されたデータをあつかう場合の諸問題に関しての主要な研究成果について考察してみる。

データの共有にあたって問題となるのは，データの最適配置と，データの共有手段である。したがってこゝではこの2つに関して述べてみることにする。

### 7.2.1 データの最適配置

企業あるいは政府省庁などで共有したいデータがある場合に，そのデータを何処に配置したら最も効率がよいかという問題が発生してくるであろう。この問題は，データの配置場所の選択が許され，全体的なコストとサービス性だけを上げればよいような場合に考慮しなければならないものである。

最適配置問題は，データ蓄積コスト，データの伝送コストなどにかかる全体の費用を，コンピュータのファイル容量，応答時間を一定の制限の下で最少にするものである。これは，非線形問題となり，線形化あるいはヒューリスティックなアプローチをおこなうなどの手法により解を得るというものである。

Chuの研究

ファイルの最適配置における最初の研究成果は，Chuによっておこなわれたものである。

Chuがおこなったのは，複数のコンピュータにより構成されたシステムにおいて，ファイルをどのように分散させるかという研究である。彼は，ファイルの蓄積コスト，データの伝送コスト，ファイルの容量，データの使用率が既知のときに，それぞれのコンピュータのファイル容量，応答時間を一定の制限の下に，全体の費用を最小とするようなファイルの割り付け方式の問題の解法に関する研究成果を発表している。

彼の研究成果の概略は次のとおりである。

$X_{ij}$ ： i番目のコンピュータに，j番目のファイルが存在することを示す変数

$L_j$ ： j番目のファイルの容量。

$b_i$ ： i番目のコンピュータのファイルを記憶できる容量。

$1/\mu_j$：それぞれのトランズアクションに対して，j番目のファイルを伝送するのにかかる平均時間。

$C_{ij}$ ： 一定時間のあいだ，i番目のコンピュータにおいて，j番目のファイルを保存した場合にかかる単位容量当りのコスト。

$C'_{ik}$ ： k番目のコンピュータから，i番目のコンピュータに単位の長さのメッセージを送るのにかかる伝送コスト。

$u_{ij}$ ： 一定時間のあいだの，i番目のコンピュータからj番目のファイルの使用率。

$\ell_j$ ： トランズアクション当りの，j番目のファイルの長さ。

としたときに，一定時間のあいだ動作するとかかるコスト（これをCとする）は次のようになる。

$$C = \sum_{i,j} C_{ij} L_j X_{ij} + \sum_{i,j,k} C'_{ik} \ell_j u_{ij} X_{kj} (1 - X_{ij})$$

記憶コスト　　伝送コスト

$$= \sum_{i,j} C_{ij} L_j X_{ij} + \sum_{i,j,k} C'_{ik} \ell_j u_{ij} X_{kj} - \sum_{i,j,k} C'_{ik} \ell_j u_{ij} X_{kj} X_{ij}$$

$k \neq i$ に対して，$X_{kj} X_{ij} = 0$，$C'_{ii} = 0$ であるので

$$C = \sum_{ij} D_{ij} X_{ij}$$

ここで

$$D_{ij} = C_{ij} L_j + \sum_k C'_{ki} \ell_j u_{kj}$$

動作コストであるCを次の4つの制限の下で最小にする。

$$X_{ij} = \begin{cases} 1 & : \text{j番目のファイルがi番目のコンピュータにあるとき。} \\ 0 & : \text{上記以外。} \end{cases}$$

$$\sum_i X_{ij} = 1 \quad : \text{すべてのjに対して}$$

$$\sum_j X_{ij} L_j \leq b_i \quad : \text{すべてのiに対して}$$

$$\sum_j u_{ij} \frac{1}{\mu_j} X_{kj} < 1 \quad : \quad k \neq i \text{に対して}$$

これは，非線形の0－1整数プログラミングの問題となる。

Chuは，これを線形の0－1整数プログラミングに変形することによって解いている。

Caseyの研究

Chuの研究においては，データの冗長性ということが考慮されていなかった。Caseyは，情報ネットワークにおいて，同一ファイルのコピーをどのように配分すれば運転コストが最も少なくなるかという問題に対する研究をおこなった。

彼は，ファイルにアクセスするトランズアクションは，問合せと更新の2種類があり，ファイル管理にとって全く性格が異なることを示した。問合せに対しては，同一のファイルを複数個持つことによって蓄積コストは大きくなるが，他のコンピュータへメッセージを送るための伝送コストが少なくなる。他方，更新が起った場合には，ファイルのコピーを持っているとすれば，同一ファイルのすべてのコピーの内容を一致させるために余分なトラフィックが発生する。

以上のことを総合的に判断して，ファイルのコピーをどこに配置するかという問題の解法についての成果を発表している。

彼の研究の概略は次のとおりである。

$\sigma_k$ ： k番目のコンピュータにファイルを持ったときに必要なコスト。

$\lambda_j$ ： j番目のコンピュータから発せられる問合せのトラフィック。

$\phi_j$ ： j番目のコンピュータから発せられる更新のトラフィック。

$d_{jk}$ ： j番目からk番目のコンピュータに問合せのトランズ・アクションがいくときに，それに要する単位量当りの伝送コスト。

$d'_{jk}$ ： j番目からk番目のコンピュータに更新のトランズ・アクションがいくときに，それに要する単位量当りの伝送コスト。

I ： コンピュータのセット。

としたときに，ファイルのコピーをどこにおくかというのは，

$$C(I) = \sum_{j=1}^{n} \left( \sum_{k \in I} \phi_j d'_{jk} + \lambda_j \min d_{jk} \right) + \sum_{k \in I} \sigma_k$$

で示されるC(I)を最小にする問題となる。

ここで，このコスト関数は，次のように書きなおすことができる。

$$C(I) = \sum_{k \in I} U_k + \sum_{j=1}^{n} G_j(I)$$

$$U_k = \sigma_k + \sum_{j=1}^{n} \phi_j d'_{jk}$$

$$G_j(I) = \lambda_j \min d_{jk}$$

この形は，すでに他で工場配置として，混合線形整数プログラミングで解かれていた。

この手法は非常にコストがかかるので，ヒューリスティックなアプローチもとられてきた。

彼は，コスト関数の特性を調べることによって，一般的な解決に対してヒューリスティックな修正をおこなった。

Morgan と Levinの研究

Chu，Caseyの研究においては，データが中心であって，これらを蓄積したファイルをどのよう

に配置するかというものであった。

MorganとLevinは，データを共有するときには必ずデータを処理するプログラムが存在することに目をつけ，プログラムの蓄積場所，実行する場所をもパラメータとして，ファイルの最適配置問題の研究をおこなった。この考え方は，コンピュータ・ネットワークにおいて，データをコピーしてきて実行するのか，プログラムをデータの存在するところに持っていくかという問題をも含んでおり，非常に重要な成果であるといってよい。

彼らの研究は，ファイルおよびプログラムのコピーの蓄積コスト，トランズアクションなどの伝送コストを総合的に考えたときの最適問題であり，概略は次のとおりである。

$\lambda_{ipf}$ ： iのコンピュータからpのプログラム経由でfのファイルへいく問合せのトラフィック。

$\lambda'_{ipf}$ ： iのコンピュータからpのプログラム経由でfのファイルへいく更新のトラフィック。

$C_{ij}$ ： iからjのコンピュータに向う問合せトランズ・アクションの単位当りの伝送コスト。

$C'_{ij}$ ： iからjのコンピュータに向う更新トランズ・アクションの単位当りの伝送コスト。

$\sigma_{jf}$ ： jのコンピュータにおけるfのファイルの蓄積コスト。

$\sigma_{jp}$ ： 〃 pのプログラムの蓄積コスト。

$\alpha$ ： 問合せトランズ・アクションの拡大係数。

$\beta$ ： 更新 〃

問合せ／更新トランズ・アクション（Q）は，プログラムの処理によって，ファイルに対するアクセスのトラフィック（G）を発生させる。

G／Qを拡大係数と呼ぶ。

$Y_{kf}$ ： ファイルfのコピーがkのコンピュータに存在するとき1，他は0。

$Y_{jp}$ ： プログラムpのコピーがjのコンピュータに存在するとき1，他は0。

$X_{jkf}$ ： jのコンピュータからfのファイルへのトランズ・アクションがkのコンピュータに送られるとき1，他は0。

$X^{f}_{ijp}$ ： iのコンピュータからfのファイルへpのプログラム経由のトランズ・アクションがjに送られるとき1，他は0。

としたとき，次の関数を最小とする問題となる。

$$C = \sum_{f,j,i,p} \lambda_{ipf} C_{ij} X^{f}_{ijp} + \sum_{f,j,i,p} \lambda' C'_{ij} X^{f}_{ijp} + \sum_{f,j,k} \rho_{jf} \alpha C_{jk} X_{jkf}$$

プログラムに達するまでの問合せトラフィックのコスト

プログラムに達するまでの更新のトラフィックのコスト

プログラムからファイルに達するまでの問合せのトラフィックのコスト

$$+ \sum_{f,j,k} T_{jf} \beta C'_{jk} Y_{kf} + \sum_{f,k} \sigma_{fk} Y_{kf} + \sum_{j,p} \sigma_{jp} Y'_{jp}$$

プログラムからファイルに達するまでの更新のトラフィックのコスト　ファイルの蓄積コスト　プログラムの蓄積コスト

ここで

$$\rho_{jf} = \sum_{i,p} \lambda_{ipf} X^{f}_{ijp} \quad : \quad j\text{で処理されるファイル}f\text{の問合せのトラフィック}$$

$$T_{jf} = \sum_{i,p} \lambda'_{ipf} X^{f}_{ijp} \quad : \quad j\text{で処理されるファイル}f\text{の更新のトラフィック}$$

### 7.2.2 データの共有手段

コンピュータ・ネットワークは，データの共有のために種々のレベルのサービスを利用者にしている。これらのサービスには次のようなものがある。

(1) 端末装置の共有

(2) ＴＳＳ／リモート・バッチ共有

(3) ファイル転送

(4) プロセス間通信

これらの，それぞれのサービスがどのように利用できるか，どのように発展させ，より高度な利用形態とすることができるかを考察してみることにする。

端末装置の共有

端末装置の共有というのは，特定のコンピュータのオンライン・システムと利用者の端末装置とを何らかの方法によって結合することにより，オンライン・システムのデータを共有しようとするものである。

この共有によってできる仕事は，結合できるオンライン・システムの種類と数によって決まると言える。逆の言い方をすれば，目的とする業務に必要なシステムとだけ結合すれば十分であるということもできる。

前項で紹介した日本におけるオンラインでのデータの共有の例である共通のキャッシュ・ディスペンサーなどはまさにこれにあたる。この場合には，１つの仕事あたりで参照しなければならないデータは，当該クレジット・カードに関するもの１個だけである。このために，サービスできる範囲に対する自由度は，端末装置のオペレータには全くない。

同様なシステムである近畿日本ツーリストの例では，端末装置側での自由度はある程度でてくる。すなわち，東京 ── 札幌の航空券を買うときに，ＪＡＬ，ＡＮＡ，ＴＤＡのいずれにアクセスす

るかは端末側の問題であり，考え方によってはかなり高度なサービスが可能である。たとえば，顧客が日時だけを指定して航空券を申し込んだ場合などは，ある手順にしたがって航空3社のコンピュータへのアクセスをおこなうことになる。この手順の自動化は不可能なことではないであろう。また，航空機の乗り継ぎなどの場合も，端末オペレータの1回の操作で予約できるようにする可能性があるといえる。

ファイルの転送

データを共有する場合におこなわれている最も基本的なものである。ファイルによるデータの共有は，コンピュータ・ネットワークが出現する以前にも，磁気テープの交換という形でおこなわれていた。

コンピュータ・ネットワークによって，ファイルの転送がオンラインで容易におこなえるようになったことから，今まで処理の対象とならなかった業務が，新らたにその対象となってきたと考えるべきであろう。

ファイルの転送を利用することによって，必要なデータをすべて処理コンピュータに集め，それから処理プログラムによって業務をおこなう手順がとられるであろう。ここで注意をしておかなければならないのは，ファイルの転送によって授受できるデータには制限があることである。

まず第一に，転送できるファイルの形式が制限されており，一定の規格に合っているものだけが送受の対象となることである。

第二番目は，ファイルの更新がファイルの転送時間および，処理コンピュータで処理している間に起こらないことである。

第三番目は，利用するデータが，ファイル全体の量に対して比較的大きいことである。

以上のような条件を満たさない場合には，ファイルの転送というのは，データの共有手段としては不適当である。しかしながら，バッチ処理であつかうデータは通常の場合には，上記の3つの条件を満たすことは十分に可能である。このために，バッチ処理レベルでのデータ共有手段として，種々の応用ができるサービスと考えられる。

TSS／リモート・バッチ共有

この形態によるデータの共有というのは，データと共有というよりはむしろプログラムの共有の副産物としてでてきたものである。

これにおいては，データがあるところにプログラムを持って行く，あるいはプログラムを作成することによってデータの共有をしようとするものである。この形態ではファイルの転送よりもはるかに強力なデータの共有が可能である。この理由は，プログラムをデータの存在するコンピュータにおいて実行する，すなわち全く通常のコンピュータの処理形態と同じ形で処理されるために，フ

ファイルの転送のときに起った制限条件がなくなるからである。

この形態において，A，B，Cの3台の処理コンピュータにあるデータ参照することによって仕事が達成できるとすれば，最も単純におこなうには，ユーザが順番にそれぞれのコンピュータにジョブを送ればよい。この方式をより高度な形でサービスしたとすれば，端末の代りにミニコンなどを使用したインテリジェント・ターミナルとして，これらのジョブの分割を自動化することが考えられる。

ＴＳＳ／リモート・バッチおよびファイルの転送

ＴＳＳ／リモート・バッチとファイルの転送を組み合わせることによってデータの共有を達成するとすれば，非常に範囲の広い強力なサービスが可能となる。

図７－２はリモート・バッチとファイル転送を組み合わせることによってコンピュータA，B，Cに存在するデータA′，B′，C′を使用して，最終結果を利用者に送りとどける仕事を図示したものである。このように3台のコンピュータで実行される3個のジョブと2個のファイル転送より構成されるネットワーク・ジョブを新らしく考えることができる。この例で注意しなければならないのは，このネットワーク・ジョブを誰れが作りだすかということである。

第一番目に考えられるのは利用者が，各コンピュータのリモート・バッチおよびファイル転送の方法を注意深く考えておこなうものである。この方法では，最初コンピュータAに対してリモート・バッチによってジョブを作り出し，処理結果をコンピュータAに作り上げる。ジョブが正常に完了したのを確かめてから，処理結果をコンピュータAからBにファイル転送によって送り込む。ファイル転送の正常完了を確認して，Bで処理すべきジョブをコンピュータBに送り込む。Bでの処理が正常に完了したのを確認して，Bでの処理結果をファイル転送によってコンピュータCに送る。これが正常終了したら，Cでの処理をコンピュータCに送り込み，最終処理結果を利用者端末に送りとどける。

以上の手順を，正確にそれぞれのコンピュータのジョブ制御言語によって記述し，オペレーションをすることになる。これは言うは易いが実際には非常にむつかしいことである。

第二番目は，インテリジェント・ターミナルにおいてジョブの進行状態の監視をおこない，オペレーションの自動化だけをおこなう方法である。このオペレーションの自動化をおこなうだけで，図７－２のようなジョブは格段に利用しやすくなるであろう。

第三番目は，ネットワーク共通言語によってジョブの分割および管理などを自動化することであろう。このような利用はネットワーク・ユーティリティという意味でコンピュータ・ネットワークにおける究極的な目標であろうが，この章の範囲ではないのでこれ以上述べない。

前例においては，リモート・バッチによって説明したが，同様なことはＴＳＳにおいても可能である。ＴＳＳの場合には，応答の即答性があるので，端末ユーザの判断あるいは端末プロセッサの

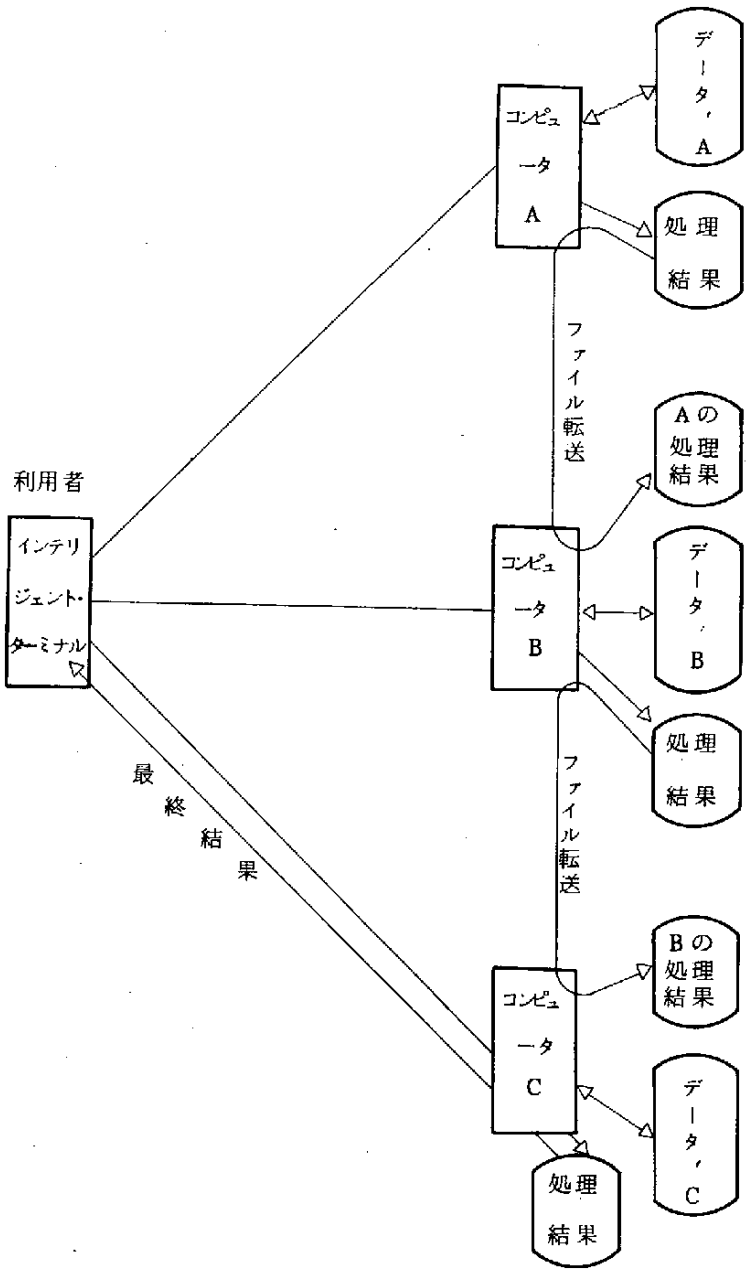



図7－2　リモート・バッチとファイルの転送によるデータの共有

処理により，リアルタイムの仕事までも含めて処理することができる。

プロセス間通信

プロセス間通信によるデータの共有は，端末装置の共有，ＴＳＳ／リモート・バッチの共有，ファイル転送などによるデータの共有と異なって，非常にきめの細かいサービスを目的とするもので

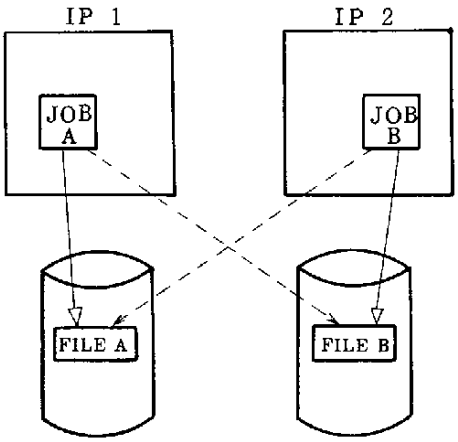



図7－3　デッド・ロック

ある。これらの中には，たとえば，データの項目ごとのアクセス，データ表現の違いをシステム側で変換することによりサービス性をあげる，データの存在場所などの案内あるいは管理を自動化させるなどがあげられるであろう。

プロセス間通信によるデータ共有の最大の利点は，同時に複数個のデータがアクセスできること，アクセスしたデータを読むと直ちに処理できることであろう。これらの利点は，同時に危険をも併なったものである。たとえば，ファイルの排他的な利用においては，図7－3に示すようなデッド・ロックを生む結果ともなる。

現在のところ，より高度なデータの共有に対するアプローチとしては，すべてプロセス間通信を主体として考える方向になっている。この理由は2つあろう。第一番目は，汎用コンピュータ・ネットワークにおいてはプロセス間通信を基本サービスとしてサポートしており容易に利用できることである。第二番目は，すべてのＨＯＳＴにあるファイルを，他のＨＯＳＴからあたかも同じＨＯＳＴにあるファイルと同様なアクセスを許すようなサービスをすること。また，さらに高度な分散形のデータ・ベースをも実現しようとする方向に向っている。これらの高度な利用を達成するには，柔軟性のあるプロセス間通信によって，各ＨＯＳＴでの処理および管理が必須となるからである。

## 7.3　プロセス間通信によるアプローチ

プロセス間通信は，汎用コンピュータ・ネットワークにおける最も基本的な機能であり，同時に非常に拡張性のある機能である。このために，コンピュータ・ネットワークにおいて種々の機能を実現する手段として最も一般的に使用されている。この項では，プロセス間通信を手段として，デ

ータの共有がどのように達成できるかを検討してみることにする。

### 7.3.1 データのアクセス方式

それぞれの処理コンピュータにおいて，データに対するアクセス方式は，種々の階層をなしている。これを大きく分けると4つの層になる。

最も低い第一層は，物理的な入出力装置に対するアクセス方式である。このレベルは，磁気テープあるいはカード・リーダなどを一台の入出力装置とみなして，入出力チャネルなどにコマンドを発するチャネル・アクセス・プログラムによって処理される。

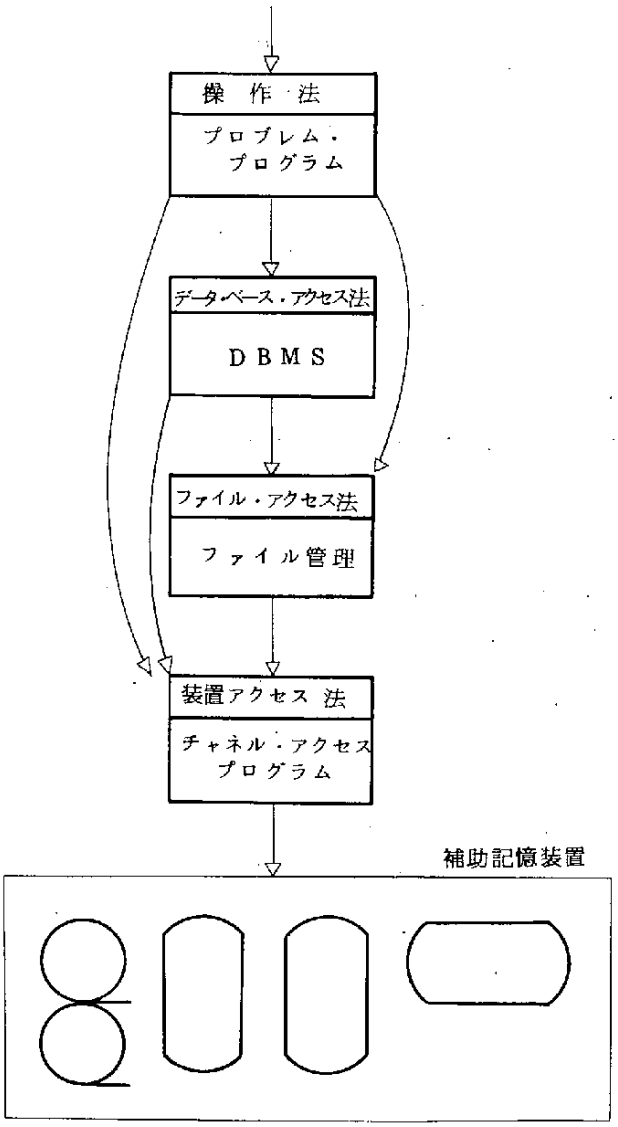



図7－4　データのアクセス

第二層は，ファイルに対するアクセス方式である。これのレベルでは，磁気テープのように1個のプログラムで占有される装置も，ディスク・パックのように複数のプログラムで分割して使用される装置も，ファイルという概念で統一し，プログラムの装置からの独立を達成するものである。このレベルのプログラムをファイル管理と呼ぶことにする。

第三層は，データ・ベースに対するアクセス方式である。このレベルは，複数個のアプリケーションにおいて最適な形で使用できるように，可能なかぎり冗長度を少なくして蓄積された，相互関係を持つデータの集合であるデータ・ベースに対するものである。これは第二層の装置からの独立に対して，データの独立という形を実現するものであり，データ・ベース・マネージメント・システム（ＤＢＭＳ）によって処理される。

第四層は，他層のデータ・アクセス法を利用することによって作られたデータ処理プログラムであって，利用者は自分の目的としたデータの検索，加工などを行なって結果をとりだす。この処理プログラムをプロブレム・プログラム，アクセス法を操作法と呼ぶことにする。

これらの四つの層を図示したのが，図７－４である。より高位の層は，任意の下位の層を使用することによって目的を達している。

プロセス間通信によって，分散処理を行い，データの共有をはかる際に，分散対象となるアクセス法は，第二層のファイル・アクセス法，第三層のデータ・ベース・アクセス法，第四層の操作法である。

第一層の装置アクセス法は，各ＨＯＳＴコンピュータのローカルなハードウェアに密着しているために分割するのが非常に困難であり，同機種，同一装置などという制約条件がつかないかぎり，分散処理は不可能であろう。

### 7.3.2　ファイル管理の分割

ファイル管理は，利用者ファイルの管理をおこなうものであるが，利用者側からファイルへのアクセスをみたときには，ファイルのオープン，データ・アクセス，ファイルのクローズという手順を実行するものである。

これらの機能を考えるときに，ファイルのオープン／クローズ，データ・アクセスの２つをどのようにおこなうかが問題になる。

すなわち，ファイルのオープン／クローズでは，ファイルの存在場所をどのようにして識別するかということ，データ・アクセスとしては，どのようなアクセス法を許し，どのようにおこなうかが問題となる。これらのそれぞれについて以下に詳しく述べてみることにする。

A．ファイルのオープン／クローズ

ファイルのオープン／クローズにおいて，最も問題になるのが，ファイルの存在場所の検出および，ファイルのクローズ後の処置すなわち，消去するのか保存するのかということである。

通常ＨＯＳＴのファイル管理は，使用者が指定した制御カードによっておこなうが，異機種分散形のコンピュータ・ネットワークにおいては，このファイルの指定方法をどのようにしたらよいかが第一番目の問題である。これに対するアプローチとしては，３つの方法が提案されている。ネットワーク・ジョブコントロール言語のようなネットワークに共通して使用できるジョブ制御言語によって指定する方法と，ネットワークのファイル・カタログを持つもの，ファイル名の中にＨＯＳＴ名を持たせるもの等である。

Peeblesの提案

彼の提案では，ファイルのアクセス方法，ファイルに対する制御カードの指定事項は，それぞれのＯＳによって違いが見られるが，目的とすることは余り変わらないので統一的なネットワーク・コントロール言語（ＮＣＬ）を作ることが可能であり，これを設定しようというものである。彼は，ＩＢＭ 360のＯＳ／360，ＵＮＩＶＡＣ－1108のＥＸＥＣ－８とを比較して，彼の設定したＮＣＬとこれらのＯＳのジョブ制御言語の類似性を述べている。

表７－３に，ファイルを記述する制御言語の例を示すが，これらの類似性をふまえて，ＮＣＬをネットワーク・データ管理で解読し，それぞれ固有のジョブ制御言語に変換してアクセスできるようにしようとするものである。

表７－３　Peeblesの提案するＮＣＬ

| NCL version | OS／360 | EXEC－8 |
|---|---|---|
| FILE-REQ | DD（statement） | @ ASG |
| filename | DSNAME | NAME |
| generation | （part of DSNAME） | （part of NAME） |
| node－name | — | — |
| buffer address | — | — |
| ＜s＞ | — | KEY 1／KEY 2 |
| NEW | DISP＝NEW | （implied） |
| device | UNIT＝（type） | （implied） |
| space | space＝ | RESERVE |
| CATLG | CATLG | C－catalogue |
| DECATLG | UNCATLG | U－uncatalogue |
| DESTROY | DELETE | D－delete |
| TEMP | DELETE | K－delete |

### du Masle の提案

彼の意見もやはり，ネットワーク・ジョブ制御言語（ＮＪＣＬ）の提案である。ＮＪＣＬを作り上げるという意味では，Peeblesのそれと全く同じであるが，アプローチが全く異なっている。すなわち，du Masleは幾種類かのＪＣＬを調査し，ネットワーク利用者の将来の必要性を評価したのち，新しい言語を作成するという方法を否定している。

彼の意見では，将来の利用者はＪＣＬに予期されていなかったサービスを期待することは確かであり，伝統的なＪＣＬは，機能を仕様決定時に固定されてしまい，利用者を満足させることはできない。このために，ＮＪＣＬのあるべき姿としては，利用者に高級プログラム言語の伝統的機能を提供できるような言語であると同時に，期待しうるサービスを具体的に定義できなければならない。

このような機能を持たせるためには，従来のパラメトリックなＪＣＬではなく，高級言語としてのNJCLにしなければならないことを強調している。これらを満足させる高級言語としてPASCALを利用することを提案している。その要因としては次のものがあげられている。

(1) 単純でかつ満足できる能力を有する

(2) データタイプの定義機能が備わっている。

(3) 効率が良い

(4) ポータビリティがある

PASCALを使用することに，利用者はジョブ制御言語ばかりではなく，手続きも同時に記述できるために一つの言語ですべての用が足せることを利点として述べている。

### ネットワーク・カタログ

この方式は，それぞれのＨＯＳＴで持っているようなファイル・カタログを，ネットワーク全体にまで広げようとするものである。

図７－５は，FarberがＤＣＳにおいて利用しているネットワーク・カタログの例である。この方法では，ネットワーク全体の詳細なファイルのカタログを持つのではなく，階層ごとに詳しさが異なっているのが普通である。すなわち，この例のように利用者ごとのカタログがあり，次にファイル名ごとのカタログになり，実際のファイルに到達できるというぐあいである。

### ＨＯＳＴ名の指定

ファイル名の前に，ＨＯＳＴ名などを追加することによって存在場所を明示する方法である。この方法は，ファイルごとに必ず利用者が意識して指定するものと，これらの管理がシステムにまかされるものとがある。後者の方がすぐれているのは明らかであるが，このようにするため

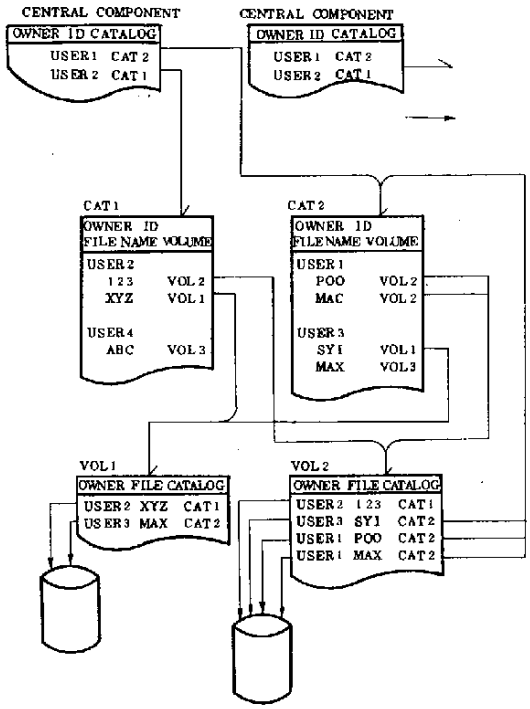



図7－5　ネットワーク・カタログ

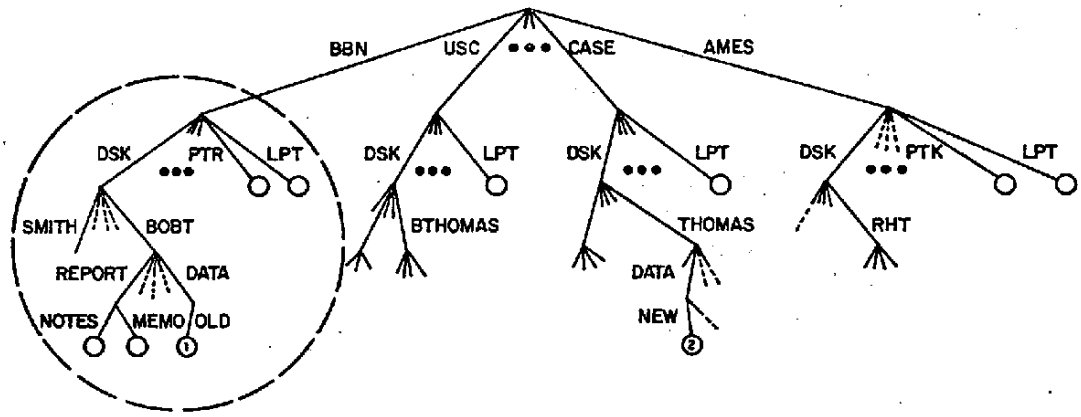



図7－6　RSEXECのファイル・ディレクトリ

には，ファイル・カタログのようなものを持たせる必要があろう。

HOST名の指定の例としては，ARPAネットワークにおいて稼動しているRSEXECのファイル管理をあげることができる。これは図7－6のような形で管理されている。すなわち，HOST名 ── 装置名 ── 利用者名 ── 個別ファイル名などによって一意的になる。

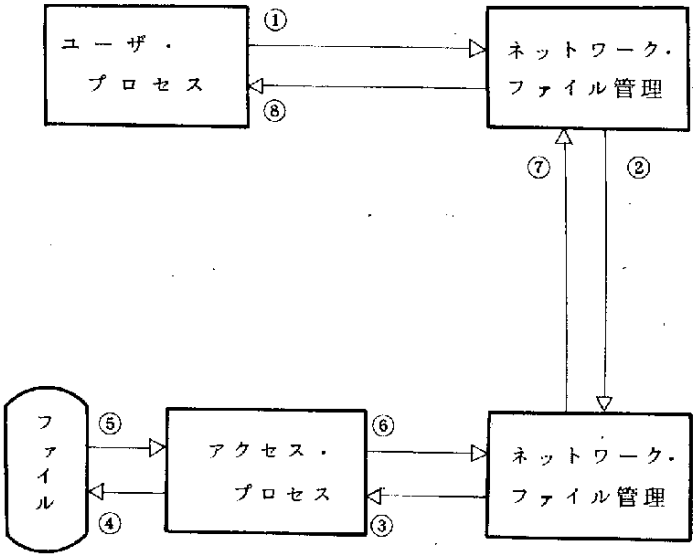
B. データ・アクセス

ファイルがオープンされてから，実際のデータ・アクセスがおこなわれるわけであるが，どのような形でおこなってよいかが問題となるであろう。

この問題点のキー・ポイントになるのは，ネットワークにおいてアクセスできるファイル形式はどのようなものがあるかということである。ＨＯＳＴコンピュータで持っているファイル形式としては，順編成ファイル，分割形順編成ファイル，直接編成ファイル，順インデックス・ファイルが存在する。すべてのＨＯＳＴコンピュータにおいて，これらの４つのファイル形式に対するアクセスがサービスされているわけではない。たとえば，ＪＩＰＮＥＴのＨＯＳＴの中で，ACOS 700とHITAC 8450においては分割形順編成ファイルに対するアクセスはサービスされていない。

このように，どのファイル形式をアクセスできるようにするかという問題は，最終的にはすべてをアクセスできる前提とするか，最小限の機能をアクセスできればよいかということになるが，実際にはすべてのアクセス方式に共通する内部プロトコルを作り上げて，ファイルの形式は各ＨＯＳＴの実状にあわせてサポートする方式がとられるであろう。

CYCLADESにおいては，直接アクセス法をシミュレートできるような形で，このレベルのプロトコルが設定されている。




① データ要求／データ送信要求
②，③ プロトコルに従ったデータ要求／データ書き込み
④ データ読み出し／データ書き込み
⑥，⑦ プロトコルに従ったデータ転送／完了情報
⑧ データ転送／完了情報

図７－７ プロセス間通信によるデータ・アクセス

プロセス間通信を使用してデータ・アクセスをおこなう場合には，図７－７に示すような形で実現される。このような方法で実現したときネットワーク・ファイル管理の間でどのようなプロトコルが必要かを，Peeblesの提案でみてみよう。

彼の提案では，18種類のPeeblesコマンドによりファイルのアクセスが可能であることを示している。

Peeblesの提案したコマンド

(1) Read－n

このコマンドは，１個以上のレコードをリモートＨＯＳＴのファイルから読み出してくる。このコマンドの一般形は次のとおりである。

Read－n　ファイル名〔，キー・リスト〕〔，バッファ・リスト〕〔，X〕〔，＜S＞〕

ここでXは排他制御の指定などに利用し，＜S＞はパスワードを指定する。

(2) Write－n

このコマンドは，１個以上のレコードをリモートＨＯＳＴのファイルに書き出すためのものである。このコマンドの一般形は次のとおりであり，パラメータはRead－n のそれとほゞ同じである。

Write－n　ファイル名〔，キー・リスト〕〔，＜S＞〕〔，バッファ・リスト〕

(3) Release－n

このコマンドは，排他的制御のレコードをリリーズする役割をする。もちろん同じレコードに対する書き込みによっても同様の機能をはたすことができる。このコマンドの一般形は次のとおりである。

Release－n　ファイル名〔，キー・リスト〕

(4) Open－n

このコマンドは，利用者に対してファイルをアクティブにするために使用される。このコマンドによってリモートＨＯＳＴにおいてアクセス・プロセスがファイルのオープンコマンドを発信する。一般形は次のとおりである。

Open－n　ファイル名 $\left\{\begin{array}{c}\mathrm{I}\\ ,\mathrm{O}\\ \mathrm{U}\end{array}\right\}$〔，ＬＥＡＶＥ〕〔，X〕〔，＜S＞〕

ここで，Ｉ，Ｏ，Ｕはそれぞれファイルが入力，出力，更新ファイルであることを示す。

ＬＥＡＶＥは，ファイルが処理される前にリワインドされることを示す。

X，＜S＞はRead－nのそれと同じである。

(5) Close－n

このコマンドは，ファイル処理の終了を示すものであり，ファイルをオープンするOpen－nの逆の役割りをする。一般形は次のとおりである。

Close－n　ファイル名〔，LEAVE〕

(6) Find－n

このコマンドは，アクセス・プロセスにおいて特定のレコード位置に位置付けするものである。これは，順インデックス・ファイルにおいて順編成アクセスの起点を指定するために利用される。一般形は次のとおりである。

Find－n　ファイル名，キー〔，＜S＞〕

(7) Cntrl－n

このコマンドは，入出力装置の位置制御をおこなうためのものであり，リワインド，バック・スペースなどに利用される。一般形は次のとおりである。

Cntrl－n　ファイル名，コントロール・コード〔，＜S＞〕

ここでコントロール・コードは次のような意味を持っている。

S（n）：nレコードをスキップする。

B（n）：nレコードをバック・スペースする。

RESET：リワインドをおこなう。

(8) Copy－n

このコマンドは，同一HOSTにある2つのファイルの複数レコードを一方から他方にコピーさせるように指令するものである。一般形は次のとおりである。

Copy－n　ファイル名1，ファイル名2，$\left\{\begin{array}{l}\text{，キー・リスト}\\ \text{，ALL}\end{array}\right\}$〔，W〕〔，＜$S_1$＞〕

〔，＜$S_2$＞〕

ここで，ファイル名1，ファイル名2はそれぞれオリジナルとコピーされたファイルを示す。キー・リストはコピーされるレコードを示すが，ALLのときには全ファイルをコピーする。Wの指定は，コピーの完了を待つかどうかを指定する。

(9) File－req

このコマンドは，リモート・ファイルのアクセス要求をするために利用される。これによってアクセス・プロセスの作成とユーザ・プロセスとのコネクションの確立がおこなわれる。一般形は次のとおりである。

File－req　ファイル名〔，世代〕〔，存在HOST〕〔，バッファ・アドレス〕

〔，＜S＞〕〔，NEW〕〔，装置名〕〔，スペース〕

$\left\{\begin{array}{l}\text{DECATLG}\\ \text{，CATLG}\\ \text{DESTROY}\end{array}\right\}$〔，TEMP〕

ここで，NEWは新しいファイルを作成することを意味する。

装置名は，DISK，TAPE，PRINTERなどを指定する。

スペースは，新しいファイルのときに確保するファイル容量を示す。

DECATLG，CATLG，DESTROYは，それぞれファイル・カタログからはずす，ファイル・カタログに登録する，ファイルを消去することを示す。

TEMPはテンポラリ・ファイルであることを示す。

(10) End－req

このコマンドは，ファイル・アクセス・プロセスに対してFile－req コマンドがもう終ったということを通知する。一般形は次のとおりである。

End－req

(11) Comac

このコマンドは，ＤＤＡＳの内部においてコマンドの実行がうまくいったむねの通知をするのに使用される。一般形は次のとおりである。

Comac　ファイル名〔，ＮＦ，キー・リスト〕

ここで，NFは，アクセス・プロセスがRead－nコマンドを実行したとき，１個以上のレコードが見つからなかったことを示す。

キー・リストは，見つからなかったレコードのキーを示す。

(12) Syser

このコマンドは，ＮＣＬコマンドの実行に失敗したときなどのエラー状態を通知するのに使用される内部的なものである。一般形は次のとおりである。

Syser　エラー・コード〔，ファイル名〕

(13) Terminate

このコマンドは，アクセス・プロセスに対して終了することを通知する。すべてのファイルの処理が終了したことの通知もできるし，一部の終了の通知も可能である，一般形は次のとおりである。

Terminate　〔ファイル名リスト〕

(14) Activate

このコマンドは，将来の拡張のために作られているもので，アクセス・プロセスをユーザが作ってこれを呼び出せるようにしたものである。一般形は次のとおりである。

Activate　アクセス・プロセス名

(15) Save

このコマンドは，アクセス・プロセスを作成したのち，保存しておくためのものである。一般形は次のとおりである。

Save　アクセス・プロセス名〔，＜Ｓ＞〕

(16) Delete

このコマンドは，アクセス・プロセスをプログラム・ファイルから消去するものである。一般形は次のとおりである。

Delete　アクセス・プロセス名〔，<S>〕

(17) Describe，Get－description

これらのコマンドは，ネットワーク・ライブラリに登録されたファイルのレコード構造も保存しなければならない。このレコード構造を保存するためのものがDescribeコマンドであり，読み出しをするのがGet－descriptionコマンドである。

### 7.3.3 ＤＢＭＳの分割

分散形データ・ベースの必要性については，コンピュータ・ネットワークの関係者のみならず，種々の分野の人々によって論じられている。しかしながら，これらの議論は，机上のものが多く，実用面を考慮しているものはきわめて少ない。

フランスのCYCLADESネットワークにおける，分散形データ・ベースの議論は，実用化までを考慮したものの数少ない例であるので，ここではＣＹＣＬＡＤＥＳにおける分散形データ・ベースの考え方を紹介する。

論理ネットワーク・マシン

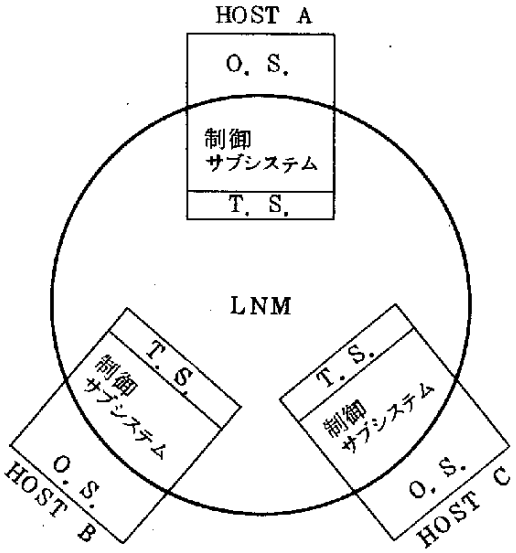
論理ネットワーク・マシン（ＬＮＭ）は，ネットワーク・コントロール・プログラムとアプリケーション・プログラムの間にあって，データの参照とスケジューリングをおこなうものである。

データの参照は，指定，検索，変換，転送を含んで，アプリケーションに独立な目的を達成するものである。このリファレンス機能の遂行のために，異なったＨＯＳＴに存在するものの間での高いレベルでのアドレス付けを可能にする。

スケジューリングは，異なった場所において局所的なジョブやプロセスの並行的な実行を生み出すネットワーク・ジョブやネットワーク・プロセスの実行をまかされるものである。その為に，局所的なオペレーティング・システム，ネットワーク言語のインタープリート，ＨＯＳＴ間機構とのインターフェース機能を含んでいる。

現在のオペレーティング・システムにおいては，ネットワーク・ジョブを管理するような機能がないので，ＨＯＳＴに新しいソフトウエアを追加しなければならない。これらのモジュールは制御サブ・システムと呼ばれ，これの集合体がネットワークの管理機能を形成する。

ＬＮＭは，図７－８のように全体的な制御を達成するために，異なったＨＯＳＴ間で制御が分散されている。異なった制御サブシステムは，トランスポート・ステーションを使うことによって情報を交換しなければならない。




T. S.:トランスポート・ステーション
O. S.:オペレーティング・システム

図7－8　LNMの概念図

ネットワークの異機種性は，コマンドとプログラミングの両方の言語に対して，統一的な言語に賛成する強い議論がある。

もしそうでなければ，ネットワーク・ジョブは異なったシステムの制御下のいくつかのジョブを生むので，すべてのＨＯＳＴは，ネットワーク内のすべてのコンピュータのコマンド言語を知らなければならない。それ故すべての参加ＨＯＳＴに理解できる統一的な言語を定義するのが望まれる。

この言語は，受け取ったすべてのＨＯＳＴにおいてインタープリートされる仮のコードでなければならない。

このレベルにおいては，コマンドとプログラミング言語が互いに異なっているべきであるという理由はない。

CYCLADESではインタープリートするアプローチがとられた，この方法は，デバッグ段階において高度の柔軟性をもつ。

あとで詳しく述べるように，それぞれのホストは仮のコードで書かれたプログラムを実行する役割をはたすIGORと呼ばれる汎用インタープリターを持っている。

仮のコードは，システム・コールばかりではなく，算術，論理，分枝，条件命令を含む，そしてそれらは，アルゴリズムの定義を許す。さらに，仮のコードのプログラムのインタープリート中に，局所的なコードで書かれたプログラムの呼び出しを可能にする。

CYCLADESでは，インタープリートする方法で，いかなるＨＯＳＴにおいても実行し得るコマンドとプログラミング言語の混合したアルゴリズムを定義する注目すべき機能を得た。

これは，システム・コールを含んでいたとしても，モジュールの全体的なトランスポータビリテ

ィをあたえる。

制御サブシステムは，それが存在するオペレーティング・システムのタスクの一つである。

その全体的な構造は，二つのタイプのプロセス ── ユーザとシステム・プロセス ── を制御するプロセス管理核（PMK）に存在する。

プロセス管理核（SYNCOPと呼ばれる）の設計は，それがネットワークの異機種性を考慮しなければならないので非常に重要である。

それは，局所的なオペレーティング・システムにかかわりなく，標準的な，プリミティブの創成，消去，同期を受けつける。

それは，設計者に，トランスポータブルな制御モジュールの手段をあたえる主要な要素である。

SYNCOPプロセス管理核は，標準的な同期構造をあたえて，別々に異なったコンピュータでインプリメントされた。

インタープリタは，FANNYと呼ばれるPL／360のような言語で書かれた，FANNYはSTAGEIIマクロ・ジェネレータを使うことによって異なったコンピュータに移すことができる。

分散形データ・ベース

論理ネットワーク・マシンの概念は，CYCLADESネットワークにおける，分散形のデータを管理するシステムであるSOCRATESに応用された。SOCRATES自身は，決してネットワーク専用に考えられたデータ・ベース言語ではなく，汎用コンピュータに対するものであった。この言語の概要に関しては後述することにする。

分散形データ・ベース管理システムは，次の3つのセットから構成されている。

(1) 利用者（端末からの利用者であってもよいし，バッチのプロセスであってもよい）

(2) データ・ベース管理システムの内部機能

(3) 物理的なデータ・ベース（各HOSTのオペレーティングシステムにとって標準であるファイルの集合）

これらのセットは，別の角度からみれば，端末アクセス法，論理的なアクセス法，物理的なアクセス法である。

分散形データ・ベース管理システムは，一つあるいはそれ以上のアクセス法が分散されたデータ・ベースである。

とられたアプローチは，アクセス法の概念を保持し，異機種，分散形ネットワークのアプリケーションとして，それぞれのレベルを設計することによって分散を確実にするものである。

結局，どのレベルが分散されるかということに対して，単純なユーザの分散から分散された論理的な"backend"までの範囲のいくつかのタイプについて調べることができる。

(1) ネットワーク直接アクセス法

はじめに説明したように，このアクセス法は物理的なデータ・ベースをあつかう（たとえば，通常の考え方によるファイルのセット）。ファイル移送に対比して，アクセスの分散による方式は，一致性の問題を避けている。

利用者（システムあるいはアプリケーション・プログラム）は，ＩＧＯＲにおいて使っている用語のプロセスである。

関連するアルゴリズムは，局所的コードとＩＧＯＲコードが混在するプログラムで構成される。

機能レベルのサブシステムは，異なったオペレーティング・システムによって標準としてあたえられている直接アクセス法によって記述されている。（ＣＩＩ VDAM, IBM BDAM, など）

オペレーティング・プロトコルは移送され，次の点に留意しながらＩＧＯＲによりインタープリトされる。

- 遠隔プロセスの創成／消滅
- 同期
- 異なったオペレーティング・システムとファイル管理システムとのインターフェース。

図７－９は，そのようなアーキテクチャを説明している。

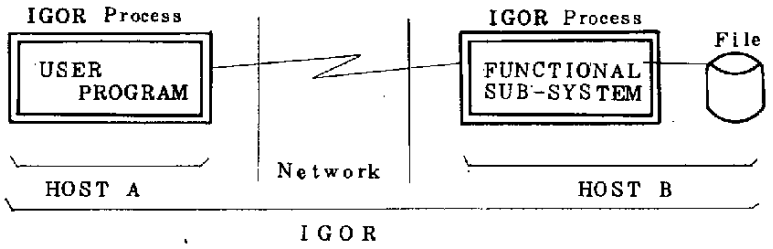



図７－９　遠隔ファイルのアクセス

SOCRATEの物理的なアクセス法を，ネットワーク直接アクセス法でおきかえたものは，異機種ホストにデータを記憶する可能性ばかりでなく，物理的なデータ・ベースの地理的な分散を自動的にあたえ，図７－10のような構成にすることができる。

(2) 論理的アクセス法

このレベルは，たぶん相互関係を持つ構造化されたデータの操作に関係を持つ。

この方法は，場合によっておこる更新の操作中に論理的な一致が保持されることが許される最小のセマンティックス・レベルで操作することを意味する。アクセス法がいくつかのデータ・ベース管理システムの間で共有されるように，データ記述言語とは独立であることが望まれる。

最小セマンティック・レベルは，次のようでなければならない。

- 利用者の使用言語と独立である。

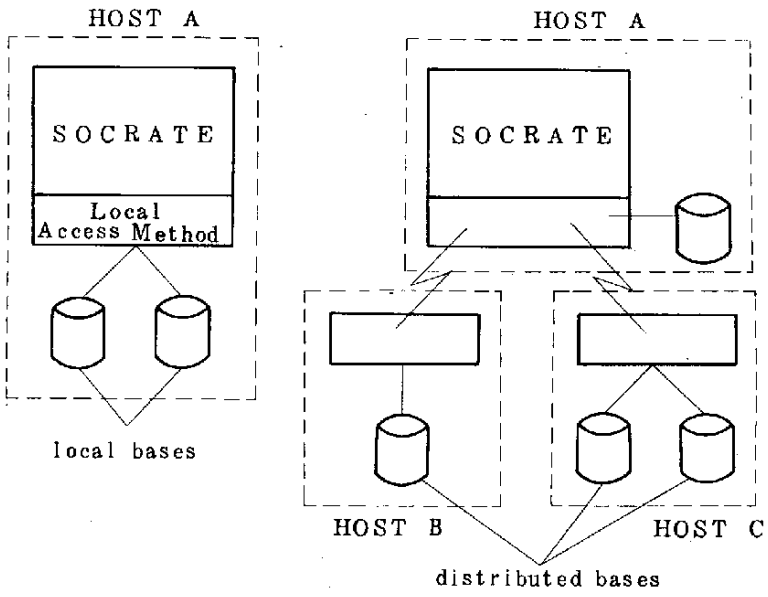



図7－10　ファイルの分散

- 物理的なデータ構造と独立である。
- アクセス・パスに独立であることが望まれる。

異なったホストにある異なった機能レベルを許すために，レベル間の特別なインターフェースを定義する必要がある。

インターフェースには，局所的なアドレス指定を使うことができない，というのは，そのような概念は，一つのコンピュータから別のものへのトランスポータビリティを失なうからである。

さらに，機能レベルの階層は，ネットワークのトラフィックと応答時間を最少にするように非常に気をつけて決めなければならない。

特に，大きなテーブルの更新は，もし２つが異なったＨＯＳＴに属しているとすれば，２つの異ったレベルによって行われるべきである。独立というよりは，異なったアプリケーション間であるアクセス法を共有する手段をあたえることは特に興味があると思われる。このことは，アクセス手法レベルにある種の＂loginメカニズム＂がなければならない，そして，高位のレベルは結合／コネクション・プロトコルを使うことによってアクセス法を導入することを意味する。

SOCRATESにおいては，利用者の使用言語は三つのフェーズによってコンパイルされる：編集，仮のコードの発生，最後のコード発生である。

少しの修正によって，SOCRATEの仮のコードは全くトランスポータブルになる。それは，分枝命令によって分けられた論理アクセス法のプリミティブのセットを含んでいる。

幸いにも，このレベルは，最小のセマンティックス・レベルを示す要求と一致する。それ故，分散がおこなわれるのはこのレベルである。

オペレーティング・プロトコルに対する表現形式として，ＩＧＯＲの仮のコードが採用することが決定された。

これは，SOCRATEの仮のコードがIGORのそれに変換されるか，あるいは，SOCRATEコンパイラがIGORの仮のコードを直接発生するように修正することを意味する。

唯一の残った問題は，サイトの選択である。

まず第一に，ネットワークに適応したデザインのデータ・ベースを採用しなければならない。

ネットワーク・データ・ベース名は，SOCRATEコンパイラにわかるところのサイト／ＨＯＳＴの識別名を一つあるいは別の方法で含んでいなければならない。

第二番目に，発生された仮のコードをブロックに構成しなければならない，それらのそれぞれはＨＯＳＴと同意である。

この構造化は，異なった方式によってなされ得る。：それぞれの利用者が直接実行できる小さな仮のコードのブロックを要求するのと，インタープリトする前に，同じホストに該当するブロックをつなげるものである。

あとの場合には，IGORは，同期のために二つの新しいオペレーション：仮の待ち（PSDWAIT）と仮のポスト（PSDPOST）：を用意しなければならない。

このアプローチは，適当な同期とネットワーク・トラフィックの最小化をする。

例として，要求言語で書かれた，次のようなユーザ・プログラムがあるとしよう。

```
form :  begin
        names of all persons having a red car ?
        types of all cars ?
        end
```

ここで，PersonsはサイトＡにあり，CarsはサイトＢにあるとする。

次のような二つのIGORの仮のコードのブロックが発生させられる。

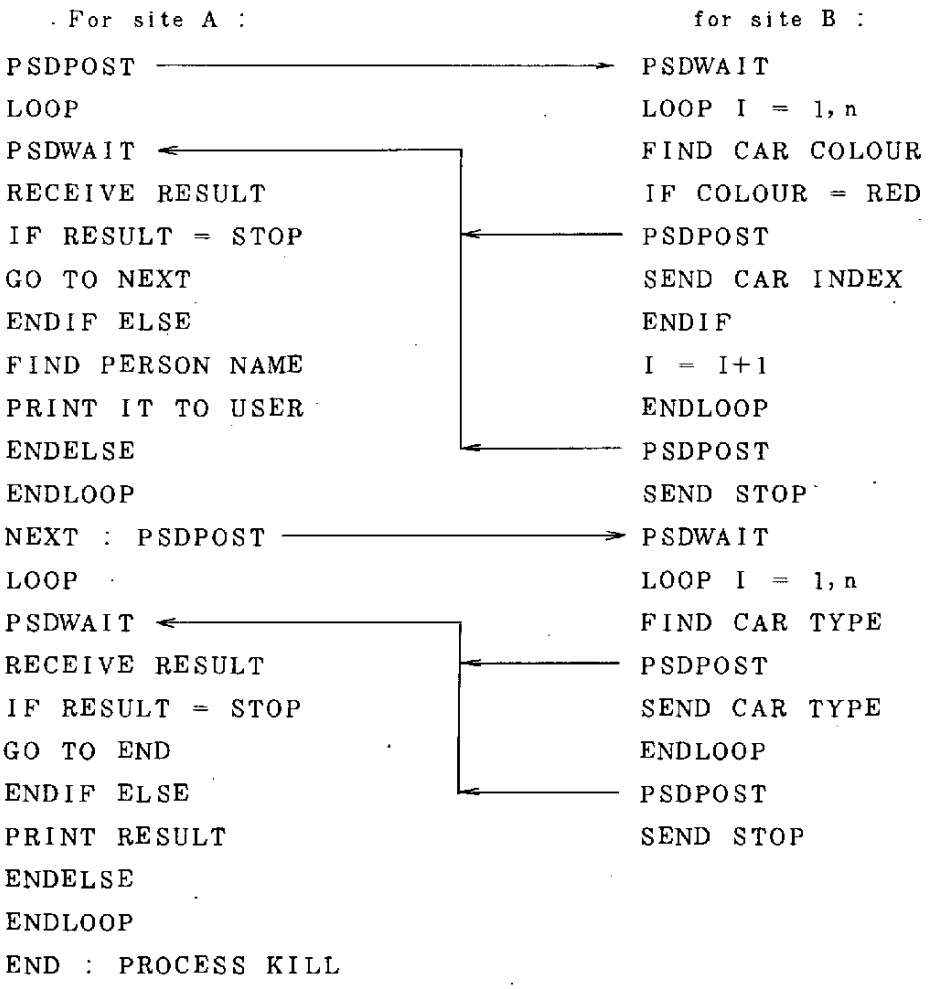



図7－11　分散データ・ベースのアクセス・プログラム

## SOCRATEの概説

データ・ベースとは，ある事実を記述したファイルの集りである。一般に，これらのファイル集合は，その構造，アクセス法，セキュリティ（安全性）に関して，一定の条件を満たしていなければならない。

SOCRATEは，これらの面で次のような特徴を有する。

- ユーザは，その情報が他のユーザにどのように使われるかということを全く意識せずに，自分に合った形でデータを記述することができる。
- 各ユーザには，特定のアクセス権が与えられる。これにより，機密保護が保証される。また，予期しないようなデータの修正をまぬがれることができる。
- ユーザは，データの関係を定義し，自分の要求に合ったデータ構造を構築することができる。

- シーケンシャル構造およびハイアラキー構造の両方がサポートされている。
- 会話型アクセスが可能であり，オンライン・データ・ベースを編成することができる。
- データ・ベースに蓄えられた情報に関してセキュリティを保証する。

(1) SOCRATEで扱われるデータ構造

SOCRATEで扱う基本的なデータには，エレメントとエンティティがある。

エレメントは，具体的な実体を表わし，値（value）によって表現される。エンティティは，それらの関連ずけられたものとして表現される。エンティティによって実現される構造そのものをデータ・ベースと考えることができる。従って，データ・ベースの中には，基本的なエレメント情報とエンティティが存在することになる。例えば，顧客を表現するデータであれば，これは"client"というエンティティから構成され，一方，エンティティ"client"はclientという識別子（identifier）と"order"というエンティティから構成することができる。また，この場合，エンティティ"command"は，そのエンティティに属する情報として顧客の購入した製品等の情報を含むことができる。

また，SOCRATEのエレメントの中では，リファレンスと呼ばれるデータ・タイプも扱われる。リファレンスは，エンティティの関係を表現するポインタ情報であり，これにより複雑なデータ構造の表現が可能になる。

例えば，エンティティ"vendor"を定義する場合に，前の例で定義したエンティティ"command"と関係づけるためには，"command"の定義の中に"vendor"に対するリファレンスを含めればよい。このようにして定義された関係は，"command"から"vendor"を参照することも，また"vendor"から"command"を参照することも可能にする。

(2) 会話型システム

以上述べたような，SOCRATEのデータ構造を扱うために，特別な言語が用意されている。ユーザは，この言語を使ってデータ・ベースを作成したり，また，それに対して問い合わせをしたりということが容易にできるようになっている。また，このシステムは，リアル・タイムで動作することができるから，データの時々刻々の変化を適格にとらえることができる。

(3) マクロ・ジェネレータ

非常に頻繁に表われるような問い合わせの形式等に対しては，マクロ・ジェネレータを使って，手順を簡略化し，ユーザの独自の要求にあったマクロ言語を作成することができる。

(4) セキュリティ

SOCRATEのデータ・ベースは，ユーザの要求に合った，無駄のないデータ構造を実現する。従って，当然，情報は内部に於て，冗長性のない形で表現されている。このような環境に於ては，情報の紛失は，致命的な問題を引きおこす。（同じ情報が重複して記録されていないから復元不可能という意味）このため，ハードウェア障害，またはソフトウェア障害に対してシステムを保

護することが非常に重要な問題となる。

SOCRATEに於ては，障害があった場合に，その最も近い過去の状態にデータ・ベースを復元するセキュリティ・システムが準備されている。

(5) システム構成

システムは，直接アクセス装置（ディスク）をベースとしている。勿論，複数ボリュームにわたってデータ・ベースを編成することも可能である。

### 7.3.4 プロブレム・プログラムの分割

コンピュータ・ネットワークにおいて，複数のＨＯＳＴに分散されたジョブが，たがいに局所的なデータやＣＰＵなどのリソースを使用しながら全体としての役目をはたしていくようなものを考えることができる。このような役割をはたすための基本機能はプロセス間通信であることは言うまでもない。コンピュータ・ネットワークにおいてのジョブの遂行形態はいずれもこのような方法によっておこなわれている。たとえばUser ＤＳＰとServer ＤＳＰは協調してＴＳＳの共有の役割りをはたしている。

ファイル管理にしろ，ＤＢＭＳにしろ利用者からみればオペレーティング・システムの一部と考えられているものは，通常コンピュータ・メーカから提供されているものである。このようなプログラムを，コンピュータ・ネットワークのインプリメンタが修正するのは困難である。

しかしながら，ファイルの共有，データ・ベースの共有をおこなうためには，何らかの形で既制のファイル管理，ＤＢＭＳなどに機能を追加しなければならない。このような機能追加の方法として，一般的なユーザ・プログラムの形態を利用するのが最も容易であると考えられる。

この項で，プロブレム・プログラムのインターフェースを新らたに，データのアクセス法として持ちだしたのは，まさにこの理由によるものである。この意味で，プロブレム・プログラムによる方法は，たとえば図７－７におけるネットワーク・ファイル管理やアクセス・プロセスをユーザ・プログラムとしてインプリメントすることを示すものである。

## 7.4 データ共有に対するアプローチ

今までみてきたように，分散されたデータの共有の手法は各種のものが有る。これらの手法の中には，ＪＩＰＮＥＴにおいてすでに実現されているものもある。しかしながら，これはＴＳＳやリモート・バッチの共有という形をとうしておこなうことができるものであって，そのニーズが顕在化しつゝあるネットワーク全体にまたがって存在するようなデータ・ファイルあるいはデータ・ベースの処理は今後の問題である。

ＪＩＰＮＥＴには実用と実験という２つの目的が課せられており，異機種間のデータ共有の手段お

よび具体的な技術の検討は既設のネットワーク・ファシリティの上でインプリメントし，実験を行うことを目標としている。以下にその基本的な概念を述べる。

### 7.4.1 データの共有に対する問題点

コンピュータ・ネットワークのHOST間に分散して蓄積されているデータを論理的に同一のコンピュータに蓄積されているのと同様なとりあつかいができるようにするのが，データの共有に対する要求であるということができる。現在，日本においておこなわれているデータ処理のほとんどはファイルを中心としたものである。一方，データとプログラムの独立性を達成する目的のデータ・ベースが大きな波として押し寄せていることも事実である。

JIPNETにおいては，ファイルの共有もデータ・ベースの共有もいずれも，プロブレム・プログラムの位置づけによっておこなう。JIPNETのベーシック・レベル・プロトコルを処理するプログラムのインプリメントの経験によれば，処理プロセスのオーバヘッドは，それがモニタ・プロセスであるかユーザ・プロセスであるかは余り関係がなく，良いプロトコルであるかどうか，適当な方法でコンピュータとインターフェースをとったかどうかによって決まることが明らかになっている。また，コンピュータ・メーカのサービスしているシステムを修正することは，その困難さの度合にくらべて実利益が少ないことも明らかになった。

この意味で，データの共有を達成するための処理プロセスはできる限りユーザ・プログラムで作り上げるべきであると考えている。

コンピュータ・ネットワークにおいてデータを共有するときに発生する問題点には次のようなものがある。

(1) データの存在場所

(2) データの形式

(3) 排他制御に起因するデッド・ロック

(4) 多重コピーを持ったファイルに対する更新から発生するファイルの不一致

(5) データの最適配置

A. データの存在場所

データの存在場所とは，ファイルあるいはデータ・ベースがどこに存在するかを知ることである。このための手段としては，ネットワーク・データ・カタログによるものが最良である。この理由は，2つあり，

第一番目は，利用者がファイルの所在地に対して何らの考慮もはらう必要がないこと。

第二番目は，利用者がファイルの所在地を知らないので，ファイルを消去するなど安全性が問題になることが少ないことである。

ネットワークがデータ・カタログを持つ場合に問題となるのは，どの程度のくわしさまで持つ

必要があるか，あるいは，すべてのHOSTで保持するのか，一部のHOSTで保持しておけばよいかということである。

JIPNETにおける基本的な考え方としては，ネットワーク・データ・カタログを持つHOSTを複数個設置するのがよいということである。これをネットワーク情報センターとして，ファイルの所在地などをプログラムから問合せるばかりでなく，端末ユーザから会話モードでの問合せに応じられるようにすべきであると考えている。このような重要な役割りをはたすネットワーク情報センターは，常時稼動していなければならない。このためには複数個のサイトをネットワーク情報センターとして，ここで保有する情報はすべて一致させておかなければならない。この問題は，すでにARPAネットワークにおいて解決されているので，それを採用すればよいであろう。

B．データの形式

データの形式に対する論点は2つあると考えられる。ファイル形式に関するものと，データのコードに関するものである。前者については，基本的なファイル形式だけをアクセスできるようにして，各HOSTでサービスできるものについてインプリメントする。後者に関しては，コード変換をどこでするのかという非常にむつかしい問題を含んでいる。

これに関しての考え方は2つある。すなわち，ビット・レベルのトランスペアレンシーを保障するという名目で何もしないもの，もう一つは，データ変換プロトコルをインプリメントしてサービスするものである。後者については，RANDで実験をおこなったDRS（Data Reconfiguration Service）があるが，実用化するにはほど遠いものである。また，データの変換といってもたぶんEBCDICとJISコードなどの比較的単純なものがほとんどであることが予想されるので，このようなものであれば，利用者プロセスにまかせて自由におこなわせるのが適当であると考えている。

C．排他制御に起因するデッド・ロック

デッド・ロックに関する研究は，Havenderをはじめ種々の研究があるが，コンピュータ・ネットワークのデータの排他的利用に起因して発生するデッド・ロックを前もって防止する手段として利用するには不適当であると考えられる。この理由としては，排他的利用をする場合のデータに関する情報のすべてを，少くとも一ケ所において収集してチェックしなければならない。このようにすると，ネットワーク間でこれのために往来するメッセージが多くなりすぎるであろう。他の方法，たとえばデータの参照順序を一定にするなどの方法によってデッド・ロックを防止する手段もあるが，データの参照順序を決め，これを守らせるのは不可能である。

このようなことから，デッド・ロックに関しては防止するのではなく，発生する可能性がある使用方法を禁止するのが最も単純でかつ不便さが少ないと考えられる。

すなわち，複数個のファイルあるいはデータを排他的に使用することによってデッド・ロック

が発生するのであるから，同時に複数個の排他使用を禁止することにすればよい。複数個のファイルあるいはデータを排他的に使用したい利用者は，あるファイルあるいはデータを排他的に使用するときには，他のものの排他使用を解除しなければならない。

利用者が複数個のファイルあるいはデータを同時に排他的に使用しないようにするチェックはネットワーク・ファイルなどの管理プロセスでおこなわれる。

D. 多重コピーの問題

多重コピーを持つファイルなどにおいて，更新があったときに，コピー間で不一致が発生し，これを一致させるメカニズムが必要になってくる。ここでは，ARPAネットワークにおいてBBNが開発した方法で，データ・ベースに対する2つのアプリケーションとインプリメンテーションに使った技術について列記する。

(1) TIPSER－RSEXECは，それぞれのTIPSER－RSEXECサイトにおいて，TIPニュース・ファイルのコピーを保持する。ニュース・ファイルの更新は，ニュース・アイテムの追加だけである。

システムは，TIPSER－RSEXECサイトにおいて初期化されるようにデータ・ベースに追加することを許し，すべてのそのような更新が伝えられデータ・ベースのすべてのコピーに編入されることを保証する。

(2) TIPログイン・システムは，ネットワーク・ユーザ・IDのデータ・ベースがすべてのTIPSER－RSEXECサイトにおいて一致して保持されていることを要求する。

このデータ・ベースのそれぞれのコピーは，互いに独立のユーザ・エントリの集合である。このデータ・ベースに対する許される更新は，個々のユーザ・エントリの追加，修正，および消去である。

いかなるサイトにおいて初期化されようとも，更新を許し，すべてのデータ・ベースのコピーを一致するように編入することを保証するデータ・ベース・マネージメント技術が開発された。

"一致するように編入される"ことは，もしすべての更新アクティビティが中止されれば，すべてのコピーは最後には一意的なものとなることを意味する。

NETNEWSとユーザ・IDのデータ・ベースを保持するために使われた技術は，それぞれ独立な次の2つの部分より構成される。

(1) すべての更新がすべてのデータ・ベースのサイトに渡たる（一度そしてたゞ一度だけ）ことを保証するために，エントリを更新するサイトにおいて使われる信頼性のある，データに独立な更新の通知と分散メカニズム。

(2) 更新コマンドが到着したときは，データ・ベース・サイトにおいてアクティブされるデータに従属した更新処理手順。

NETNEWSに対しては，更新手順はNETNEWSが到着したときにデータ・ベースに付加するという比較的単純なものである。ユーザ・ＩＤのデータ・ベースに対しては，もっと複雑な更新手続きが要求される。

データ・ベースのエントリに対する最新の更新は古いものにかぶせるという方法である。たとえば，ユーザのパスワードが変えられたならば古いパスワードは，新しいものと単純に入れ変えられる。

更新手続きは，異なったデータ・ベースのサイトのそれぞれにおいて，更新の事象の順に再構成し，動作するために，タイム・スタンプのメカニズムを利用するのが基本になっている。

さらに，データ・ベースのそれぞれのエントリ（と，修正可能なサブフィールド）は，現在の値にしたところの更新のタイム・スタンプを保持している。

ほとんどの更新コマンドがデータ・ベースのサイトに到着したとき，それが参照しているデータ・ベース・エントリのタイム・スタンプと単純に比較することにより編入あるいはリジェクトのいずれかを行うことができる。

エントリの消去と創成は少し違ったあつかいを必要とする。たとえば，もし，一つのエントリに対して創成と消去の両方のコマンドが異なったサイトで発生したならば，ネットワークあるいはシステムの機能が不完全であると三番目のサイトには，消去コマンドのあとに，創成コマンドがくるような事が起こる。そのような場合をうまく処理するために，データ・ベースの更新手続は，消去コマンドが到着する以前に初期化されたエントリに対するすべての更新コマンドが確定するまで，消去コマンドに対する最終アクションが延期される。そして安全にデータ・ベースからエントリを消去できるのはその時点だけである。

ＴＩＰアカウント・システムの動作は，セグメント化されたデータ・ベースの創成と操作を生みだす。

アカウンティング・アプリケーションにおいて第一に関係するのは，データ・ベースの構成と便利なデータ・ベースのアクセスである。そのデータ・ベースには，次のようなものが要求される。

(1) カタログ

データ・セグメント（インクリメンタル・アカウンティング・ファイル）をアクセスするために，それらの所在を知る必要がある。

カタログ機能は，RSEXEC分散形ファイルシステムによってあたえられる。

(2) エントリの重複がないことを保証する。

エントリは，アカウンティング情報を含むので，重複があってはならない。データの重複配置は行われているが，データ収集プロトコルに於て，データ・エントリ自体の重複は存在しないように注意深く設計された。

(3) アカウンティング合計が作られるとき，それぞれのデータ・ベース・エントリはたゞ一度だけ処理されることを保証する。タイム・スタンプが，"たゞ一度だけ"の処理を保証する機能を持つということは興味あることである。

E. データの最適配置

データの最適配置とは，ファイルあるいはデータ・ベースのオリジナルと必要に応じてそのコピーを，最も効率良く分散配置する方法を論ずるものである。最適配置問題は，経済性およびサービス性と密接な関係を持っているが，これ自体は独立した問題と考えることができる。すなわち，パケット交換網の設計におけるトポロジーの問題と同じようなものである。

比較的広域のネットワークに於て，データの配置を任意に選択しうる場合，伝送コスト，データ・アクセスや更新の頻度や状況，地理的条件等の諸要素の兼ね合いにより，データの最適配置あるいは最適配分を決定する事は極めて重要である。しかしながら，公共的なネットワークあるいはセキュリティの観点などから既設のデータベースの所在を変更できぬ場合等では，最適配置を実施する余地がなく，むしろ現在の配置を前提として，あるいは一応既存の配置を最適とみなして，その条件の元で最適なデータの共用手段を考えねばならぬ場合もあろう。

ＪＩＰＮＥＴに於てはむしろ後者の立場，即ち既存の配置にもとづくデータの共用手段に重点をおき，最適配置問題は別途，独立した問題として検討することとする。

### 7.4.2 検討すべき方向

ＪＩＰＮＥＴにおけるデータの共有手段として今後検討を進めて行くべきものは，次の５つのパターンである。これらのうちで，(1)はファイルの共有という形であり，他の４つはデータ・ベースの共有という形である。

(1) ネットワーク・ファイル管理の実現

(2) DBMSのネットワーク・インターフェース方式

(3) ネットワーク・データ・ベース・アクセス方式

(4) 制御の集中方式

(5) 分散DBMS方式

A. ネットワーク・ファイル管理

ファイル利用を一つのコンピュータから，ネットワークに結合されたＨＯＳＴすべてにまで範囲を広げようとするものである。これは図7－12のようにして，ネットワーク・ファイル管理プロセスにアクセスすることによって，局所的なファイルも遠隔ファイルもアクセスできるようにするものである。ファイルの存在場所などに関してはネットワーク情報センターの助けにより捜しだすのを前提とする。

この方式では，完全な汎用性を求めないかぎり，技術的にはかなりの部分が解決しており，実

用レベルのものを作り上げることは可能であろう。

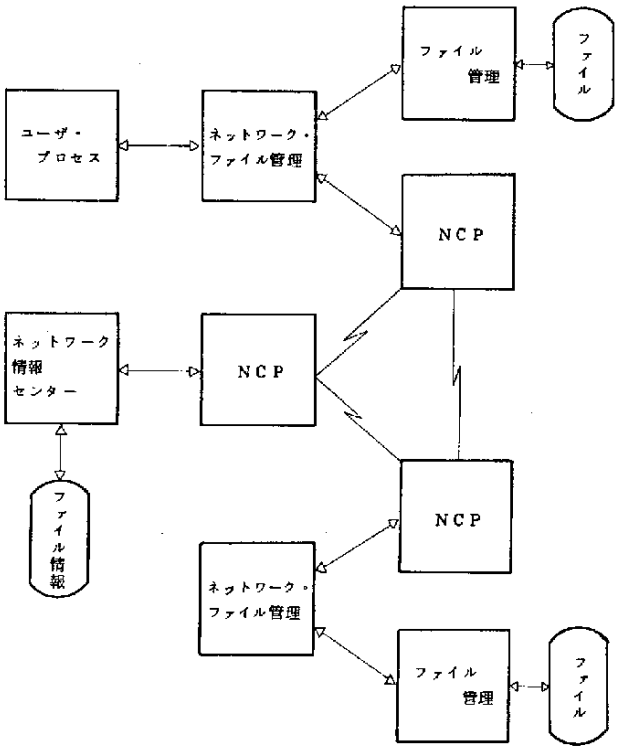



図7－12　ネットワーク・ファイル管理

B. ネットワーク・インターフェース

図7－13のごとく各HOSTが持っているデータ・ベースとDBMSを遠隔地から利用するインターフェースとしてサービスする方式をネットワーク・インターフェース方式と呼ぶことにする。

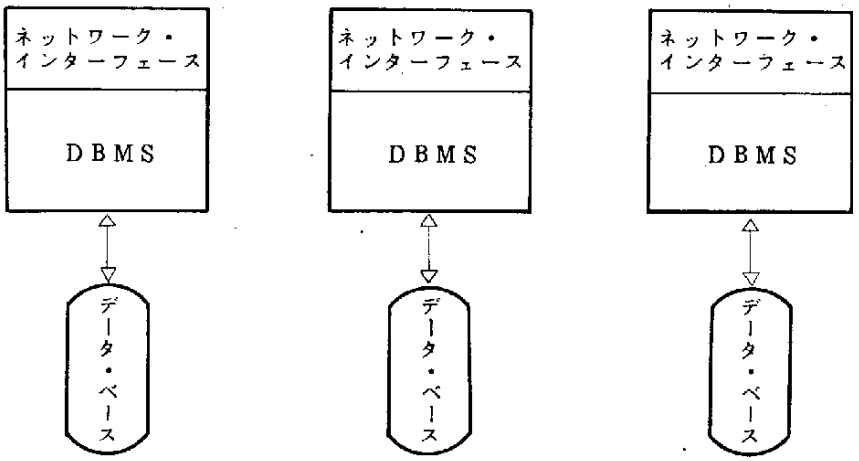



図7－13　ネットワーク・インターフェース方式

この方式は，プロセス間通信によって他のHOSTから当該HOSTのデータ・ベースを利用できるようにするものである。DBMSが，パラメータにより利用者から使用される方式においては，ネットワーク・インターフェースは容易にインプリメントすることができる。すなわち，カード読み取り装置あるいはローカル端末から入力していたパラメータを，プロセス間通信により遠隔地端末から入力することにすればよい。同じような手法は，ＴＳＳ，リモート・バッチの共有などに利用されているので問題はない。

DBMSがCODASYLのDBTG提案のように，プログラムから利用できるようにした場合には，データの構造などの記述をどのようにして，利用者プログラムとDBMSとの間で授受するかなどの問題が多い。

C．ネットワーク・データ・ベース・アクセス方式

図7－14のように，それぞれのDBMSに対して共通のアクセス法をサポートするシステムを作成するものである。それぞれのデータ・ベースは独立に存在するが，共通の方法によってデータ・アクセスできることから，HOSTの独立性を重んずるコンピュータ・ネットワークにおいては，最適であると考えられる。しかしながら，共通のアクセス法が作成できるかどうかという点に大きな疑問がある。

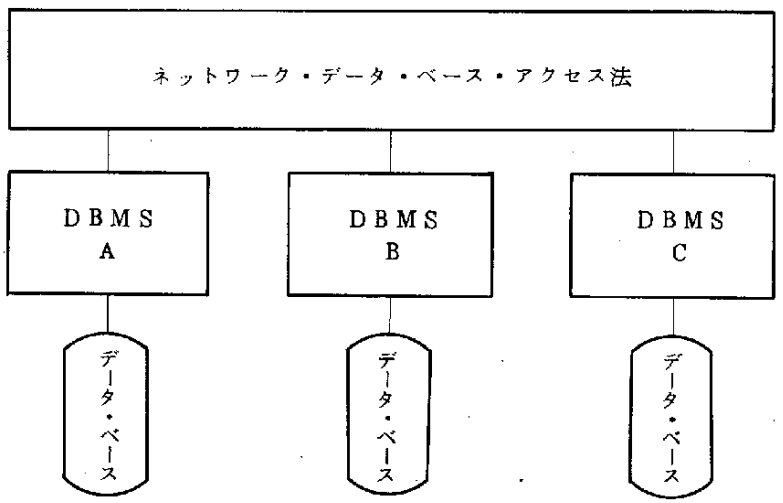



図7－14　ネットワーク・データ・ベース・アクセス方式

D．制御の集中方式

図7－15のごとく，特定のHOSTにDBMSが１つだけ存在し，このDBMSのみから，プロセス間通信によって各HOSTのファイル管理とのデータの交換が可能となる。各HOSTのデータは，ファイル管理によってファイルとして管理されているが，すべてのHOSTの関係するデータ・ファイルによって１つのデータ・ベースが構成される。

この方式では，データ・ベースへのアクセスはすべてDBMSを通じておこなわれるので，たとえデータが自分のHOSTに存在したとしても，直接参照はできない。

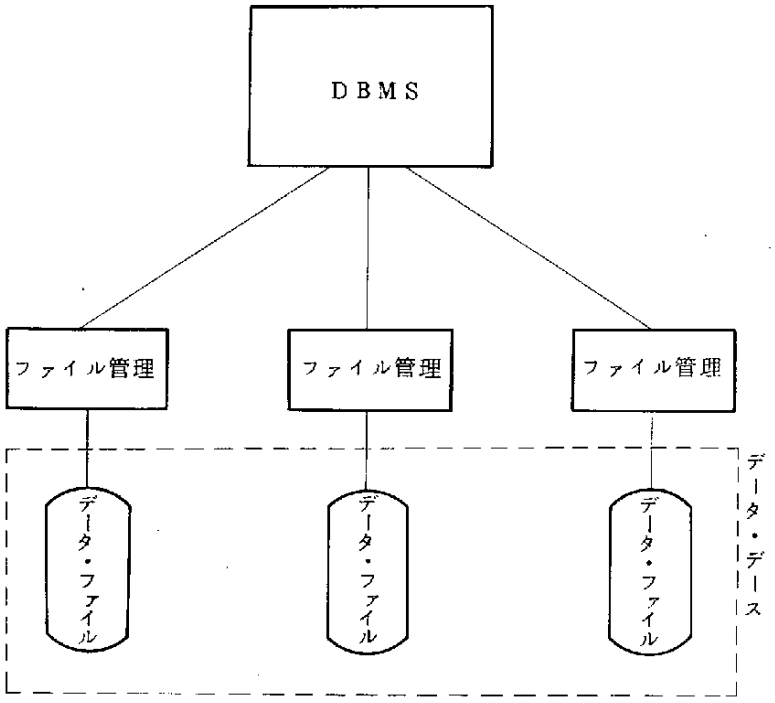



図7－15　制御の集中方式

E．分散DBMS方式

図7－16のごとく，各HOSTに分散してDBMSを置くものである。これは制御の集中方式における唯一のDBMSを，各HOSTに分散させたものであると考えられる。各HOSTにある分散

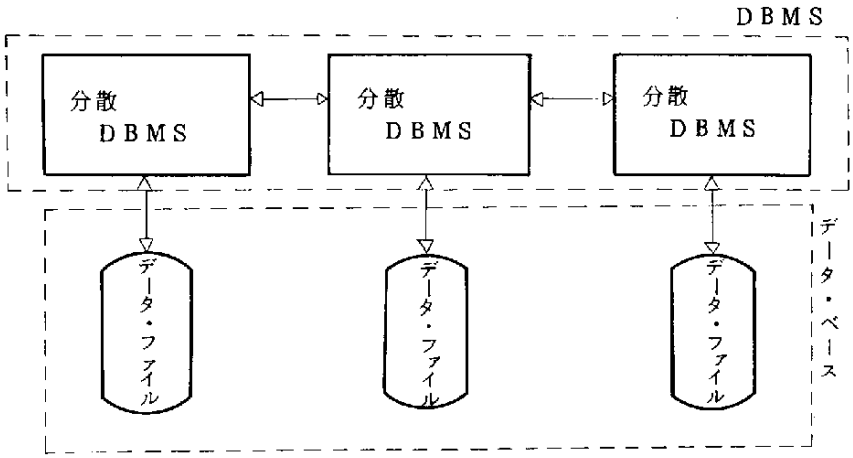



図7－16　分散DBMS方式

DBMSは，そこに存在するデータ・ファイルを，他の分散DBMSと協力してデータ・ベースとして管理する。

この方式においては，自分のHOSTにあるデータは原則として自分のところにある分散DBMSによって読み出すことができる。

**第7章の参考文献**


(1) Chupin J. C.
Control concepts of a logical network machine for data Banks. IFIPS 74 Stockholm. p. p. 291～295

(2) Chu W. W.
Optimal file allocation in a multicomputer information system. IFIPS 68 Edinburgh. pp. F 80～F 85.

(3) Casey R. G.
Allocation of copies of a file in an information network. SJCC 72. vol. 40. pp. 617～625.

(4) Casey R. G.
Design of tree networks for distributed data. NCC 73. vol. 42. pp. 251～257.

(5) Booth G. M.
The use of distributed data bases in information networks. ICCC'72. pp. 371～376.

(6) Morgan H. L, Levin K. D.
Optimal program and data locations in computer networks. AD／A－001008, 1974. p. 22.

(7) Bihan J. L.
A survey of problems in distributed data bases. Reseau Cyclades DAT 510, July 1975, p. 6.

(8) BBN
Interface Message Processor for the ARPA computer network. Quarterly technical report №2. 1 April to 30 June. BBN report №3106. July 1975. p. 78.

(9) Farber D. J. Heinrich F. R.
The structure of a distributed computer system — The distributed file system, ICCC '72, pp. 364～370.

(10) Peebles R. W.
Design Considerations for a distributed data access system. AD 775569. May 1973. p. 401.

(11) CII
SOCRATE, manual d'utilisation, 1973, p. 124.

(12) Chupin J. C, Seguin J., Segeant G.
Distributed Application on Heterogeneous Networks. 3 rd Annual Symposium on Computer Architecture Florida, 1976. 1.

(13) 名和小太郎
分散型ネットワークシステム ── 旭化成のＡＣＴについて ── 関東ＮＥＡＣユーザー会第４回例会資料, S. 49, p. 27.

(14) 名和小太郎
企業内ネットワークについて, 情報処理に関するシンポジウム予稿, 日本情報処理開発センター, S. 51. pp. 58～64.

(15) 上野　滋
官庁間コンピュータ・ネットワークのニーズと現在開発中, 計画中のネットワーク・システムの概要について, 情報処理に関するシンポジウム予稿, 日本情報処理開発センター, S. 51, pp. 51～57.

(16) 日本情報処理開発センター
複合ＮＩＳ形成に関する調査研究報告書, 47－R 006, S. 48, p. 255.

(17) 電子技術総合研究所
パターン情報処理研究システム ── 計算機複合体EPICS ──, S. 49; p. 239.

(18) 野垣内　章
旅行業者と運輸・航空機関とのコンピュータ・リンク, ビジネス・コミュニケーション '75, vol. 12, №2, pp. 48～55.

(19) Combs B.
TYMNET : A distributed Network, DATAMATION '73, 7. pp. 40～43.

(20) Tymes L. R.
TYMNET－A terminal oriented communication network. SJCC '71. pp. 211～216.

(21) 日本情報処理開発センター
米国におけるコンピュータ・ネットワークの技術動向, S. 49, p. 184.

(22) CODASYL
DATA BASE TASK GROUP REPORT ON THE CODASYL PROGRAMMING LANGUAGE COMMITTEE APRIL 1971, '71, p. 273.

(23) Thomas R. H.
A resource sharing executive for the ARPANET NCC'73, pp. 155～163.

# 8．海外の技術動向

# 8. 海外の技術動向

## 8.1 はじめに

ARPAコンピュータ・ネットワークが大々的に世の中に発表されたのは，1970年のSJCCであった。

このネットワークは，1969年末に米国の西海岸にある，SRI（Stanford Research Institute），UCSB（University of California Santa Barbara），UCLA（University of California Los Angels），UTAH（University of Utah）の4ケ所のノードおよびホストの結合からはじまった。

ARPAネットワークは，こののち順調に拡大し，1975年5月現在，56ノード（うち25台がTIP），ホストは104台（TIPも含む）となった。

ARPAネットワークの成功を契機として，世界各国においてコンピュータ・ネットワークの研究開発が競って開始され，タイム・シェアリング・システムの次の世代の代表的なコンピュータ利用の形態としてリソース共用の限界を飛躍的に拡大させる新世面を開いた。

現在，各所で開発されたコンピュータ・ネットワークは恐らく20を越え，いくつかのシステムは実用レベルで稼動しつつある。

現時点でコンピュータ・ネットワークの研究活動に与えられたテーマの一つは，真に有効なリソースの共用技術の開発であり，欧米の幾つかの大学あるいは研究機関に於て，それぞれユニークな研究活動が行われている。

JIPNETに於ても現在基本的ファシリティとしてのシステムのインプリメントが一段落し，今後は更に一歩を進めたネットワークの新しい高度なリソース共用技術の開発に入るべき時点に至っている。この機に応じて先進的な欧米の研究動向を調査し，今後のネットワーク研究の指針を定める参考に資することを目的とし，代表的な8機関を選択し調査を行った。

本章は，先づ訪問した8機関の研究内容と，ヨーロッパの代表的コンファレンスであるEUROCOMPに於けるコンピュータ・ネットワークのセクションの概要を紹介し，そのあとARPAネットワークを中心としたパケット交換ネットワークの設計上の問題点と，CYCLADESを中心としたコンピュータ・ネットワークの開発方法につきまとめたものである。

日　程　表

| 月　日 | 曜　日 | 訪　問　先 | 所　在　地 |
|---|---|---|---|
| 9・5 | 金 | University of Hawaii | Honolulu |
| 8 | 月 | Rand Corporation | Los Angeles |
| 10 | 水 | NBS | Washington D.C |
| 12 | 金 | BBN | Boston |
| 15 | 月 | IBM-France | Paris |
| 16 | 火 | IRIA | Paris |
| 18 | 木 | IMAG | Grenoble |
| 22 | 月 | CTNE | Madrid |
| 23 | 火 | EUROCOMP '75 | London |
| 25 | 木 | | |

出張者　伊藤　哲史　(財)日本情報処理開発センター開発部開発第一課課長代理
鍛冶　勝三　(財)日本情報処理開発センター開発部開発第一課課長代理
後藤　龍男　日本電気㈱情報処理官庁システム営業本部　システム部主任
下田　正文　日立ソフトウェアエンジニアリング㈱　第一設計部技師

## 8.2　各　論

### 8.2.1　Hawaii大学

訪問先　University of Hawaii
所在地　2540 Dole Street
Honolulu, Hawaii 96822, U. S. A.
訪問日　9月5日(金)
面接者　Prof. Norman Abramson
Mr. Moris Wilson (大学院生)
Mr. Chris Harrison (大学院生)

A. 調査の目的

今年(1975年)5月に米国アナハイム市で開かれたNCC(National Computer Conference)では，コンピュータ・ネットワークのセクションに丸1日が割り当てられ，その内の半日は新しいパケット無線ネットワーク(Packet Radio Network)の発表で占められた。

ネットワークとネットワークの結合，衛星回線の利用などと同様に，無線ネットワークは今後のコンピュータ・ネットワークの焦点になると考えられる。

ハワイ大学では，最初にこのパケット無線ネットワークを提案し，実用化した。又，最近衛星回線を介してALOHANETとARPANETが結合された。

訪問の目的は，パケット無線ネットワークと衛星通信の実用化の動向を聞くことである。

B．調査内容

ALOHA SYSTEMの概要

ハワイ大学では過去5年間にわたって実験用UHF無線コンピュータ・コミュニケーション・ネットワークTHE ALOHA* SYSTEMを開発している。現在，実験的に稼動中であり，upper UHF帯域中の407.350MHzと413.475MHzの両チャネルとも24,000ボーで使用している。システムはランダム・アクセス無線チャネル多重化の新しい方式を用い，ARPANETで採用されたパケット交換方式を使用している。最近，THE ALOHA SYSTEMはARPANETの最初のサテライト・ノードになった。ハワイ大学のキャンパスにあるTIP（Terminal Interface Processor）はINTELSAT Ⅳを使用してカリフォルニア州にあるNASA／AMESのTIP間で50キロ・ビット／秒のデータ・チャネルで通信している。このことはTHE ALOHA SYSTEMとARPANETの結合を意味する。50キロ・ビット／秒の衛星チャネルはINTELSAT Ⅳ上のPCM音声チャネルの1個を占有する。

現在の主要な研究は，無線通信と衛星通信の2つである。無線通信に関しては，プログラマブル端末や中継器（repeater）として使用されるミニコンピュータを使用してPhase Ⅱ ALOHA SYSTEMを現在開発中である。衛星通信に関しては，経済性の良い小型地上局と共にNASAの通信衛星ATS−1を含むプロジェクトが進行中である。この研究は，別のALOHA SYSTEMプロジェクト，すなわち，太平洋沿岸の先進国や開発途上国にあるコンピュータをリンクするために研究中のPacific Educational Computer Network（PACNET）のプロジェクトとも結びつく。

THE ALOHA SYSTEM

THE ALOHA SYSTEMは，Advanced Research Projects Agency（ARPA），National Science Foundation（NSF）などの援助により2つの主要なテーマ：コンピュータ・コミュニケーションとコンピュータ・ストラクチャから成っている。

コンピュータ・コミュニケーション関係の研究開発は，(a)無線と衛星通信を利用したコンピュータ・コミュニケーションの研究，(b)無線式タイム・シェアリング・ネットワークのプロト・タイプの開発，(c)米国，日本，オーストラリアそして太平洋地域諸国の主要大学をリンクするPacific Area Computer Communication Networkに関するシステムの研究と計画から成っている。

---

* Additive Links On-line Hawaii Areaの略称。

コンピュータ・ストラクチャ関係の研究開発は，マルチプロセッサ・コンピューティング・ストラクチャ，コンピュータ・ネットワーク，そして広域コンピュータ・システムから成っている。これはさらに2つのフェーズ，1) 研究用実験装置の組立て，そして2) 研究そのものに分けられる。この実験装置として，現在開発中のBCC500コンピューティング・システムを使用する予定である。このシステムは数年以内に完成予定であり，多数の大学ユーザ端末を処理する能力をもち，大学の重要なリソースとなるのであろう。

パケット無線通信ネットワーク

1960年にコンピュータと遠隔地にある端末装置を結ぶコミュニケーション・システムすなわちオンライン・テレプロセシング・システムの概念が現われて以来，数多くのタイム・シェアリング，リモート・ジョブ・エントリ・システムやオンライン・システムが開発されてきた。現世代のコンピュータ・コミュニケーション・システムは専用回線あるいは電話交換回線など主として有線接続の使用を基本としている。多くの場合，これらの通信設備を利用して要求に応じた大規模なコンピュータ・コミュニケーション施設を設計することはできる。しかし，特殊な条件のもとでは，コモン・キャリアが提供するデータ通信システムでは大規模情報処理システムの可能性が制約される。

1968年以来，ハワイ大学のTHE ALOHA SYSTEMプロジェクトは，広域に分散されたコ

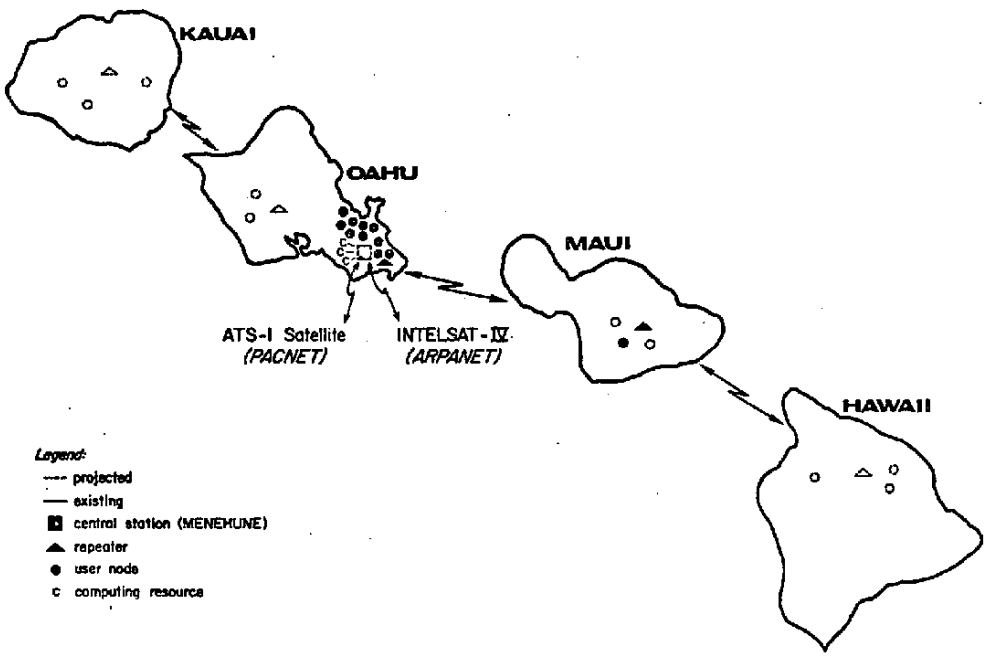



図8－1　THE ALOHA SYSTEM, Geographic Map

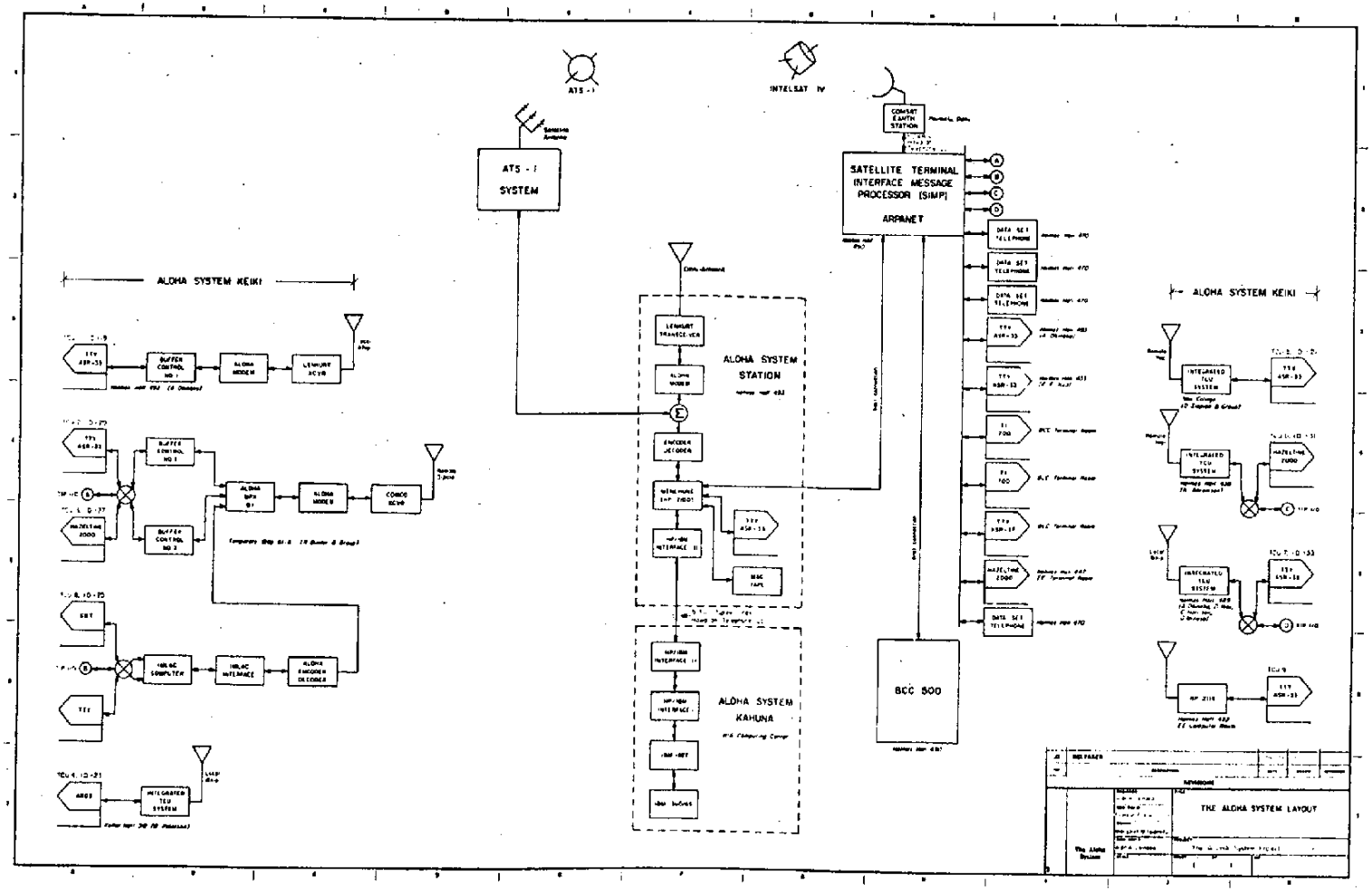



図 8 – 2 THE ALOHA SYSTEM, Layout

ンピュータ・システム間のリンクとして通常の有線によるデータ通信の制約から解放された場合，システム設計者はコンピュータ・コミュニケーション・ネット内でデータを伝送するため各種の方式を使用することができるようになる。THE ALOHA SYSTEMプロジェクトは州規模の大学コンピューティング・システムに対しランダム・アクセス・コミュニケーションの新しい簡単な方式の使用法を研究し，ＵＨＦの無線でリンクされたコンピュータ・システムが1971年の中ごろに作成された。

THE ALOHA SYSTEMの現在のgeographic mapを図8－1に，layoutを図8－2に示す。

大学の中央コンピュータは主記憶容量2.5MBのIBM 360／65で，OS MVTの下で稼動しており，バックグランド・バッチに加えて2つのタイム・シェアリング・システムAPLとTSOが動いている。THE ALOHA SYSTEMはTSOユーザの1つであり，データ・コンセントレータとマルチプレクサのサービスを行うHP 2100コンピュータを介して360コンピュータに接続されている。2100マシンの機能は，ARPANETで使用されているInterface Message Processor（IMP）の機能とほぼ同様であるので，2100をMENEHUNE（メネフネ，ハワイ語でハワイの伝説的妖精（imp））と呼んでいる。モデムは24,000ボーで動作し，差動位相変調（differential phase shift keyed modulation）を使用している。復調器（demodulator）はビット・タイミングをリカバーするためのphase－locked－loopと，雑音がある場合の信号をリカバーするための調整フィルタ（matched filler）を使用して，特にコヒーレント（coherent，継続的な）信号に対して最適に動作するように設計されている。送受信機（transmiter－receiver）は，システム仕様を満足させるため標準的な商用ＦＭトランシーバを改造したものである。端末（KEIKI，ハワイ語で子供）ステーションのフロント・エンドにはALOHA SYSTEMで開発したTerminal Control Unit（ＴＣＵ）と呼ばれるcommunication moduleが付いている。TCUはUHFアンテナ，トランシーバ，モデムとバッファ制御装置からなり，これらはコンパクトなシャシーの中にまとめられている。この様にＴＣＵはすべての端末ステーションとHost側にも付いてる（図8－3参照）。TCUは24,000ボーまでの速度で動作する様々な端末にインタフェースすることができる。現在，テレタイプ型端末（ＴＴＹ），CRTディスプレ（Hageltine 2000），グラフィック・プロセッサ（ARDSとImlac）とミニコンピュータ（HP 2114とImlac）をTCUに接続させている。ミニコンピュータに接続する場合は，ＴＣＵのバッファ制御機能がミニコンピュータ中のソフトウェア・パッケージで操作されている。ALOHAバースト・ランダム・アクセス通信の方式によってチャネル容量を効果的に使用しているため，現在，チャネルの負荷は少々である。アクティブな500台以上の英数字用端末を現在のTHE ALOHA SYSTEMコミュニケーション・チャネルに収容できることが計算されている。

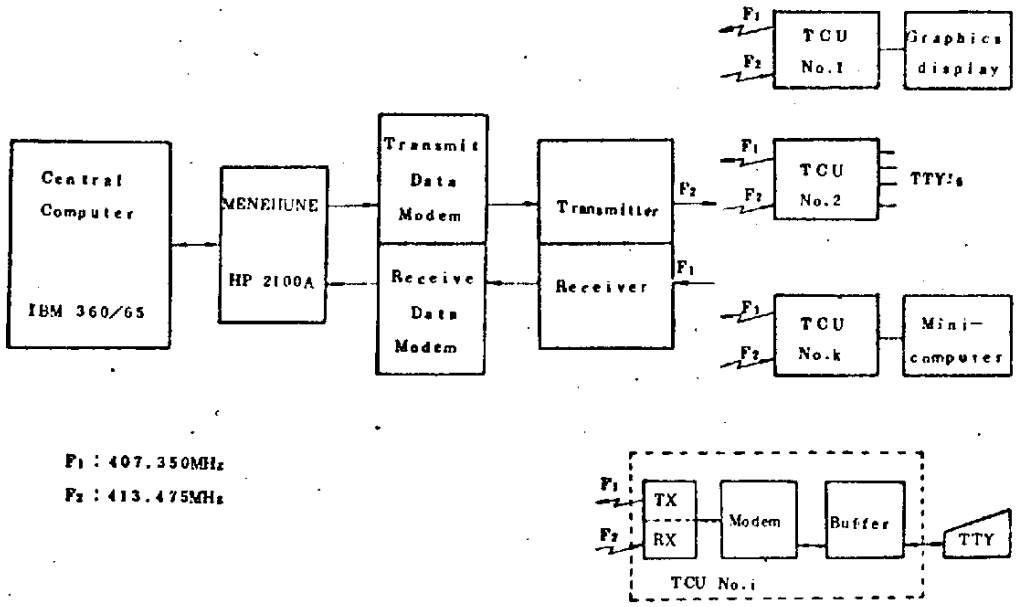



図8−3　THE ALOHA SYSTEM basic configuration

現在，異なったモード（非同期，同期，コンテンション，ファイル・スケジューリング，複数個の周波数など）で使用するランダム・アクセス・チャネルの特性に関して，解析法やコンピュータ・シミュレーションの両方を使って研究が行なわれている。チャネル・プロトコルを変更することによってシステム・パフォーマンスにどんな影響を与えるかもまた研究している。ハードワイヤなバージョンよりももっとフレキシブルなＴＣＵを開発するため，１つの集積回路チップでできているＴＣＵは，システムが可変長パケットや character−by−character 伝送などの様々な異なった伝送プロトコルに対処することを可能にした。各種のエラー制御手順をこの新しいＴＣＵにインプリメントすることによって研究することができる。

## 8.2.2　RAND

訪問先　RAND Corporation

所在地　1700 Main Street, Santa Monica, Los Angeles California 90406, U. S. A.

訪問日　9月8日（月）

面接者　Mr. Eric F. Harslem
（Manager, Systems Development）

A. 訪問の目的

RANDにおいては，IBM 360／65をHOSTとしてもち，ARPAネットワークに参加してい

る。ここでは異機種コンピュータ・ネットワークにおいてデータ・コード違いの問題を解決するためにＤＲＳ（Data Reconfiguration Service）の研究をおこなっている。これの詳細と，目標などを明確につかむのが調査の目的であった。

B. 調査内容

ＤＲＳ

ＤＲＳは，異機種コンピュータ・ネットワークにおいてデータ・コードの標準化が非常にむつかしく，これを解決する方法として作ったものである。最初の発想は，TELNETプロトコルを作成にあたって，それぞれのタイムシェアリング・システムにおいてコードが違っており，これを標準的に処理する方法を開発しようとするものであった。

DRSは，システムの開発をおこない，unofficialプロトコルになっているが，officialプロトコルにすることは考えていない。この理由は，officialプロトコルにすると，メンテナンスなどを十分におこなわなければならなくなるが，予算がないからである。このシステムは処理もおそく，実験レベルで終了してしまったと考えてよい。この意味においてＤＲＳに対する期待は裏切られたが，期待していなかったＮＣＰの作り方などについての成果があった。

ＮＣＰの構造

RANDにおいて最初にARPAネットワークに参加したときには，IBM 360／65がHOSTコ

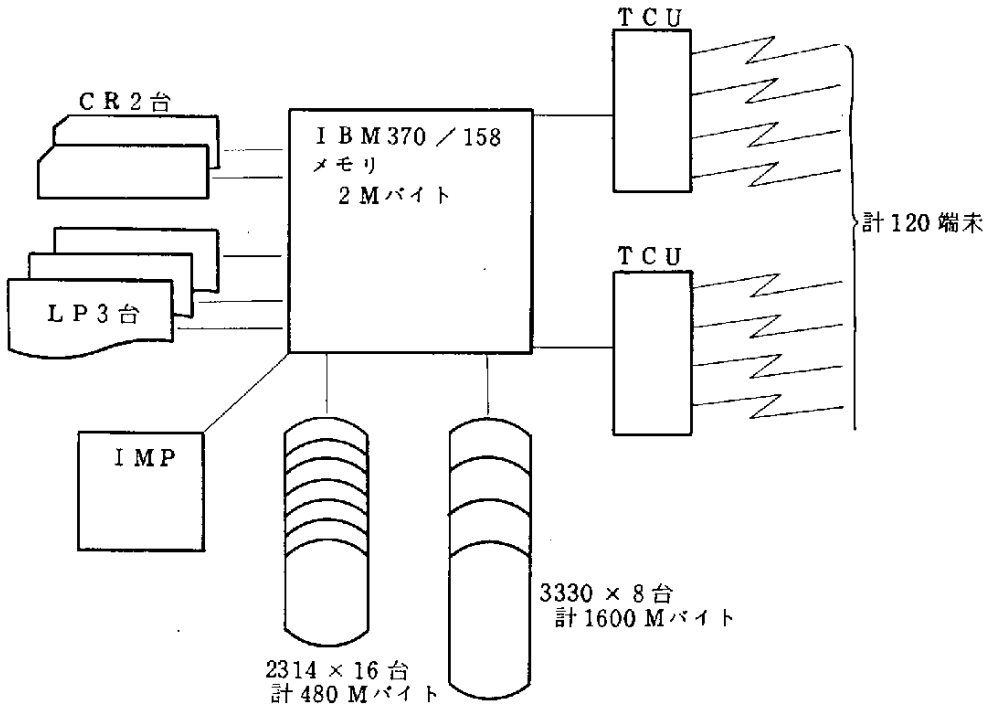



図８－４　ＲＡＮＤのシステム構成

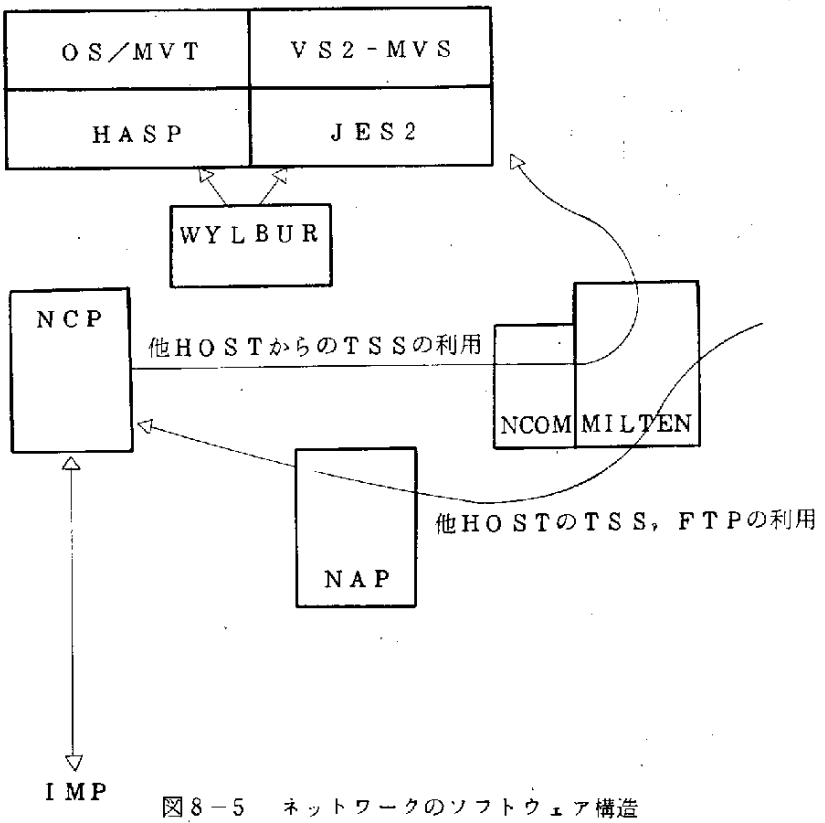



図8－5　ネットワークのソフトウェア構造

ンピュータであったが，我々が訪問したときには，IBM 370／158に置き換っていた。このシステム構成は図8－4に示すとうりであり，RANDでは120の端末からタイムシェアリングで利用している。

オペレーティング・システムとしては，OS／MVTあるいは，VS2－MVSのいずれか一方を使用している。ネットワークの参加にあたっては，ファイル容量があまりなかったので，ファイル転送はユーザ側を作ることにし，Telnetはユーザ／サーバの両側を作ることになった。

これらのプロトコルの実現は，図8－5のような構造になっている。この中で，NAPはユーザFTP，ユーザTelnetの機能をはたすプログラムであり，MILTEN，NCOMは，サーバTelnetの機能をはたすものである。

NCP，NAPなどはユーザ・プログラムである。このような構造のシステムは，オペレーティング・システムのPLPA（Program Linkage Pack Area）とCSA（Common Service Area）およびOS本体の中にウィンドを持たせ，プロセス間通信機能を実現することによって可能になった。

これをインプリメントするのに要した工数は，表8－1のように合計18～19人月，コア占有量は（95k＋α）バイトである。

表8－1　ネットワーク・ソフトウェアの開発コスト

| プログラム | プログラム・サイズ | 工　数 | 備　考 |
|---|---|---|---|
| N C P | 50kバイト＋データ | 9人・月 | 容　易 |
| N A P | 25～30kバイト | 3～4人月 | 容　易 |
| MILTEN NCOM | 約15kバイト | 6ケ月 | 非常に困難 |

NCPの作り方としては，HOST内に実現する方法と，フロント・エンド・プロセッサにおいて80％を作り，残りの20％くらいをHOST内に持たせる方法などがある。特に後者は，フロント・エンド・プロセッサとHOSTの間に標準化したプロトコルを設定しようとする動きがあることを聞いた。

### 8.2.3　NBS

訪問先　National Bureau of Standards

所在地　Washington D. C. 20234, U. S. A.

訪問日　9月10日（水）

面接者　Mr. Robert Rosenthal（Dr.）

Mr. Albrecht J. Neumann（Dr.）

A. 訪問の目的

米国商務省に属する標準局では，標準化ということを目的として，システムの測定，評価をおこなっている。

また，タイムシェアリング・システムへのアクセス方法の標準化と，それとは全く異なるアプローチとして，Network Access Machine（NAM）という利用者が自分の決めた標準言語によってすべてのネットワークにアクセスする手段をあたえるシステムを開発している。これらのことを調査するのが目的である。

B. 調査内容

Network Access Machine

ARPAネットワークの成功は，利用者に広範囲のリソースを提供することを完成させたということができる。しかしながら，これらのリソースを利用しようとする利用者に対して，今までに数倍するマニュアルを読むことを強制することにもなった。すなわち，種々のHOSTにおいてサービスしているタイムシェアリング・システムを使用するときに，ログ・オン，コマンド，ログ・オフなどすべてが異なっていることから，ARPAネットワークの方式では，利用するすべてのHOSTのタイムシェアリング・システムのコマンド体系を知っていなければならない。

このようなことは，利用者が他のＨＯＳＴのタイムシェアリング・システムを利用するのに大きな障害になることは明らかである。この障害をどのようにとりのぞくかというアプローチは種々あるが，その一つがＮＡＭである。

ＮＡＭの基本思想は，利用者（利用者端末）とＨＯＳＴコンピュータの間に通訳の役割りをするコンピュータ（ＮＡＭ）をおき，ＨＯＳＴ特有のコマンド体系をＮＡＭによって標準コマンドあるいは利用者の定めたコマンドに変換しようとするものである。

すなわち図８－６に示されるように，どのタイムシェアリング・システムでも，同じコマンドを使用することによって利用しようとするものである。

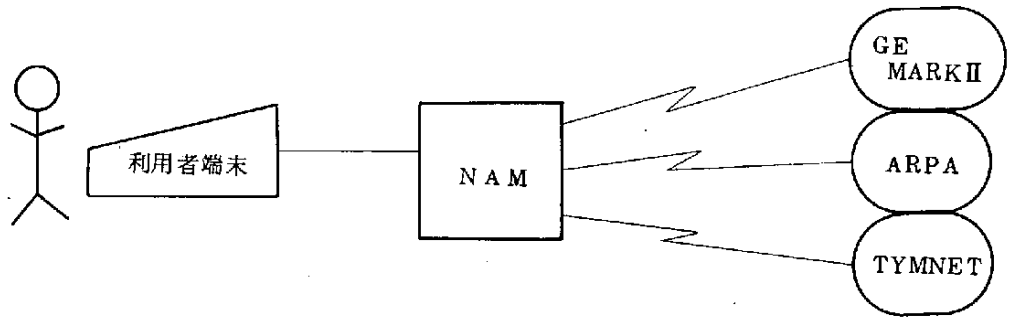



図８－６　ＮＡＭの概念

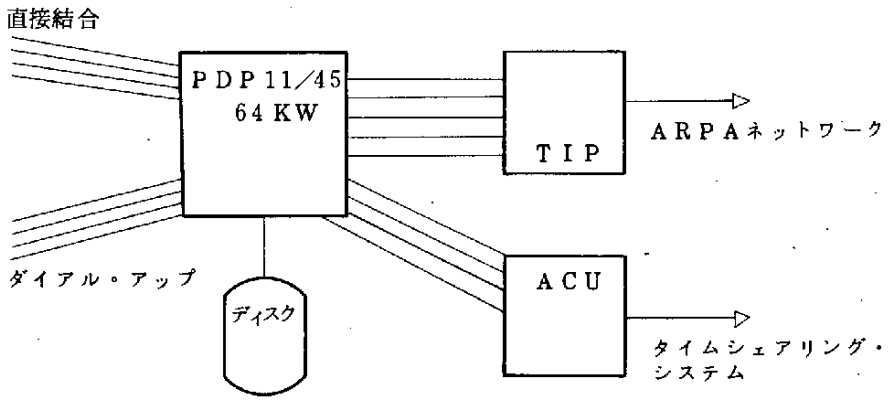



図８－７　ＮＡＭの構成概念

ＮＡＭのハードウェア

ＮＡＭはＰＤＰ１１／４５で６４ｋ語の記憶装置とディスクパックを持っている。ＮＡＭと利用者の結合方式は，回線を通じて直接つながっているものと，ダイアル交換装置を通じてつながっているものの２種類がある。ＮＡＭとタイムシェアリング・システムの結合方法は，ＮＢＳに設置されているＡＲＰＡネットワークのＴＩＰを経由するものと，ＡＣＵ（自動ダイアル装置）によって特定のタイムシェアリング・システムを電話で呼び出すものとの２種類がある。これを示した

のが図８－７である。

NAMのソフトウェア

NAMのソフトウェアは，マクロ・エクスパンダとレスポンス・アナライザおよびこの２つが利用するデータ・ベースより構成される。

NAMの利用者は，各人が自分専用のネットワーク・アクセス手順を決め，マクロの形でシステムに登録する方法をとっている。マクロ・エキスパンダは，利用者の入力したメッセージをコマンドとして受け取り，これに対応するタイムシェアリング・システムに対するコマンドを，データ・ベースからさがしだして発生させ，同時にどのような応答がシステムから返されるかという期待応答を作り上げる。レスポンス・アナライザは，期待応答にしたがって，手順がうまく進行しているかどうかのチェックをおこなう。

ここで一番問題になるのは，各人の専用ネットワークアクセス手順を，容易にマクロの形で表現できるかどうかということであるが，これはそんなに簡単ではないと考えられる。しかしながら，あるグループの誰かが決めれば，全員がそれを使用できることにすることによって，問題点を解決している。

もう一つの問題点は，すべてのシステムに対する応答手順がNAMで処理できるかどうかということであるが，我々のみたところでは，完全に処理するのは困難であると思われた。

NAMの今後の計画

ハード的には，利用者をチェックするためにサイン（署名）のチェック装置，音声の入出力装置を追加していこうという計画である。

ソフト的には利用者に対して情報をあたえたり，間違いなどを訂正する tutorial assistance をおこなうこと，期待応答時間とロードの平均化によってコンピュータを選択する機構を追加する計画である。

Network Measurement System

コンピュータ・ネットワーク・システムとそのサービス性のパフォーマンスを測定し，解析し，評価するのが，このNetwork Measurement System（NMS）の目的である。ネットワーク・オペレーションの内部メカニズムよりも端末ユーザに提供されるサービス性に焦点を合せて測定が行われている。この測定は，測定中のネットワーク・システムを干渉することなしに行うことができる。

このNMSは，Dialogue Monitor，Terminal Environment Simulator，Network Measurement Machine などの一連の測定や評価道具をまとめたものである。

### 8.2.4 BBN

訪問先　Bolt Beranek and Newman Inc.

所在地　50 Moulton Street, Cambridge, Massachusetts 02138, U. S. A.

訪問日　9月12日(金)

面接者　Mr. Frank E. Heart (Vice President)

Mr. David C. Walden

Mr. William R. Crowther

Mr. Alex McKenzie

A. 訪問の目的

BBNを訪問した目的は，今さら説明を要しないであろう。すなわちARPAネットワークの設計，インプリメンテーション，オペレーションをおこなっている会社であり，ここを訪問して，ARPAネットワークの現状およびこれからの動向について調査したかったからである。

B. 調査内容

プラリバスIMP

この新しいノードは，High Speed Modular IMPとも呼ばれている。これの基本的な考え方は，図8－8のように14台のロッキードSUEプロセッサとメモリおよびI／Oを結合したシステムである。

このシステムの開発の背景には，2つの要素があると考えられる。1つは，セントラル・オフィス，もう1つは，サテライトIMPの開発である。

セントラル・オフィス

コンピュータ・ネットワークは，最初の予想よりもはるかに大きくなることが判明した。ARPAネットワークにおいても，最初は20ノードくらいのものを想定していたが，現実には60ノードにならんとしている。このようなネットワークの巨大化は，特殊なネットワークの集合化をしてさらに大きくなる傾向にある。

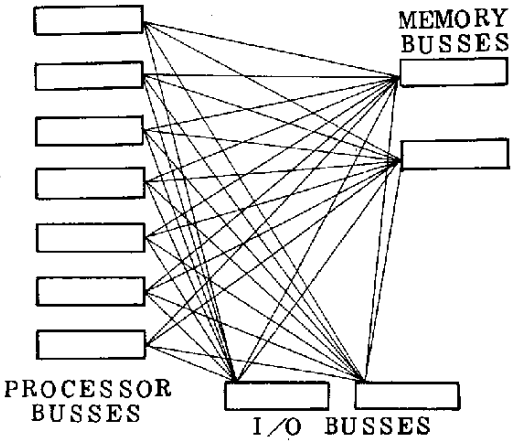



図8－8　プラリバスIMPの概念

初期のARPAネットワークでは，IMPはHOSTのあるところに設置されていた。HOSTがふえるにつれて，同一の場所に設置するのが困難になり，回線経由のHOSTとIMPの結合方式も開発された。IMPを各サイトに設置すると，IMPの数が増大するばかりではなく，IMPが採用しているアルゴリズムにも問題点がでてくる。

このようなものを解決するためには，各サイトに1個ずつ設置してあったＩＭＰを何台か集めて，1台のＩＭＰとし1ケ所でサービスする必要がある。現在のIMPは，DDP 516で750 k bit／sec，DDP 316で500 k bit／secのトラフィックが処理できるだけである。セントラル・オフィスで使用するためには，これだけのスループットでは不十分である。また安定性も格段に向上させる必要があることから，これらのものをすべて満足するＩＭＰを開発する必要がでてきた。

プラリバスＩＭＰは，14台のCPUに対して，それぞれ異なった電源装置などをも持っており，どのＣＰＵが故障しても代りのものが仕事を遂行できるようにして，安定性を向上させている。14台のCPUによって処理できるスループットは7.5 Mega bit／secとなり，DDP 516 ＩＭＰの10倍の性能を持っている。

サテライトＩＭＰ

ARPAネットワークにおいてノード間（IMP間）を結ぶリンクは，50 k bps の地上通信回線である── もちろん9.6 k bpsから230.4 k bpsのものも利用できる。これ以外に，カリフォルニアにあるNASA－AMESとハワイ大学，ワシントンD. C.にあるSDACとノールウェーのNORSARを結ぶのに，それぞれ50 k bps，9.6 k bpsのサテライト・リンクを使用している。

ARPAネットワークでは，海外への拡張と，国内衛生通信を利用するために，サテライトＩＭＰという新しいIMPが開発されている。

このサテライトIMPが利用する技術の背景は，ALOHAシステムが開発したチャネルを共用するものを発展させた手法である。このサテライトIMPは，1.5 M bpsの広帯域を使用するため，プラリバスIMPを使用している。

HOSTのソフトウェアの開発

ＮＣＰなどの開発は，１人年５万ドルの人がやったとして，現在のところ6～24人月の工数が必要である。これだけの工数で開発できるのは，NCP，ICP，User／Server Telnet の3つであり，これだけあれば最低限の機能を持ってネットワークに参加することができる。開発期間にばらつきがでるのは，オペレーティング・システムにプロセス間コミュニケーションのサポートがあるかどうかというのが非常に大きな要因になっている。これだけの金をかけるのは安くはないけれども知識を得るための費用と思えばよいであろう。また，同じコンピュータが沢山使用されれば，ＮＣＰは最初のところでだけ作ればよいのだから，ネットワークが拡張されてくれば安くなるであろう。

### 8.2.5 IBM—France

訪問先　IBM－France

所在地　B. P. 149

92102 Boulogne Billancourt, France

訪問日　9月15日(月)

面接者　Mrs. Monique Somia (Dr.)

A. 訪問の目的

ここを訪問先として選んだのは，図8－9で示す実験ネットワークであるSOC(Systéms d'Ordinateurs Connectés)がどのような成果を残して終了したかを調査するためである。

B. 調査内容

紹　介

SOCプロジェクトは，CEA，IMAG，CIRCE，MINESの4つの大学とIBMの2つのコンピュータ・センターを4.8 k bit／secの回線で結合し，バッチ，会話形の両方の処理を相互から使用できるようにした実験プロジェクトで，1970年から1974年にかけておこなわれた。

このプロジェクトの目的は次の4つである。

① コンピュータ・ネットワーク利用の機能の実現。

② ネットワークにおけるソフトウェアの問題の解決をする試みをおこなう。

③ 実験ネットワークの実現。。

④ ユーザの動向を知ること。

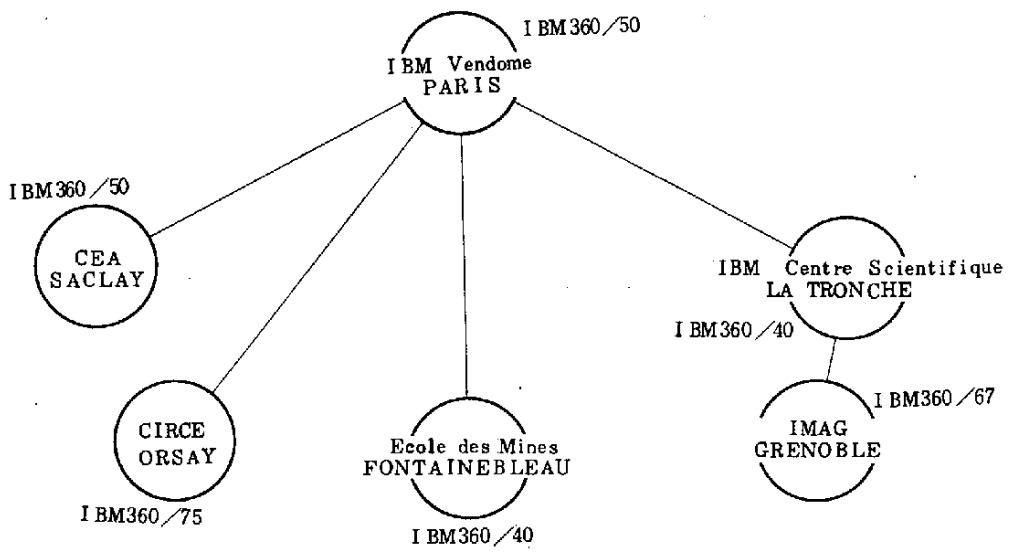



図8－9　SOCの構成

NCL

SOCのアプローチで最も注目しなければならないのは，ネットワークに対する利用者側のアプローチが第一に考えられたことである。これはARPAネットワークのプロトコルを重要視した——したがって，インプリメント側が主体となった考え方となる——アプローチと対称的な位置にあるものである。

これは，実際には，ネットワークに共通して利用できるNCL（Network Command Language）をユーザに対して公開する方向がとられた。

このNCLの設計方針は次のとうりである。

① 使い易さ。

② ローカル・オペレーティング・システムからの独立。

③ 特別な機能の利用

④ 次の3つのものと交信ができること。

——ローカル・ユーザ

——他のコンピュータ

——ローカル・オペレーティング・システム

このような方針で，NCLが作られたが，コンピュータ・ネットワーク機能の利用を指向した高級言語であり，機能としては，データ・セットの転送，任意のリモート・コンピュータにおいてジョブが実行できるということを実現したものである。

NCLは，宣言ステートメントと，実行ステートメントを持っており，前者は実行コンピュータ，ファイルの所在などを示すものであり，後者は前者で定義したコンピュータへファイルなどをどのような手順で転送して，どのような手順で実行し，何を結果としてどこへ返すかを指定するものである。

NCLの宣言ステートメントは，CPU－UNIT－VOLUME－DATA SET－MEMBER（分割形順編成ファイルのみ）－INPUT STREAMという形で表現される。この例を図8－10に示す。

```
CPU ,A : = IMAG67 , B : = CIRCE75;
UNIT  U1 : = 2314/A , U2 : = 2314/CIRCE75;
VOLUME V1 : = MFT111/U1,
       V2 : = MFT222/2314/CIRCE75;
DSN  D1 : = SOC.OBLIB/V1,
     D2 : = SOC.OBLIB/MFT222/U2;
MEMBER  M1 : = MAIN/D1;

INP      ALPHA : = '///*'
         Data cards
///*
```

図8－10　NCL宣言ステートメント

図8－10の例をみれば明らかなように，ＩＢＭのジョブ制御言語を拡張する形で実現されており，ＩＢＭのコンピュータを日常的に利用しているユーザにとっては使い易いであろう。しかしながら，ローカル・オペレーティング・システムからの独立という意図はＩＢＭオリエンティッドであり実現されていないと考えるべきである。このあたりがＩＢＭという一つの企業がおこなっているプロジェクトにおいて，ローカル・システムからの独立の限界を示すものであろう。

ＮＣＬの実行ステートメントは，データ・セットの創成／消去／内容の入れかえ／カタログ化／カタログの消去，ボリュームのマウント／ディスマウント，データ・セットのコピー／追加およびジョブの実行を指令するものであり，図8－11のような表現方法をとる。

```
CREATE    DS1;
DELETE    DS1;
EMPTY     DS1;

CATALG    DS1 ON CPU1;

UNCATALG  DS1 ON CPU1;

MOUNT     VOL1;
DISMOUNT  VOL1;
COPY      DS1 TO DS2;
ADD       DS1 TO DS2;
RUN       DS AT A;
```

図8－11　ＮＣＬ実行ステートメント

```
NETIN SMITH;
CPU B:=IMAG67, C:=CIRCE75;
DSN DS1:=SOC.OBLIB/111111/2314/B,
    DS2:=SOC.OBLIB/MVT192/2314/C;
COPY DS1 to DS2;
INP PL1JOB:='. ./'
//PL1 JOB ......
// EXEC PL1FCLG
//PL1.SYSIN DD *
     MAIN: PROC OPTIONS(MAIN);
          :
     END MAIN;
/*
//GO.DS DD DSN=SOC.OBLIB, UNIT=2314,
           VOL=SER=MVT192, DISP=OLD
//GO.SYSIN DD *
           :
          Data
           :
/
. ./
RUN PL1JOB AT C;
COPY DS2 TO DS1;
NETOUT;
```

図8－12　ネットワーク・ジョブ

このＮＣＬを使用してジョブを実行する例を図8－12に示す。

SOCにおいては，プロトコルとしてNIL（Network Internal Language）があり，これをHOST間で交換して実際の処理をおこなうが，あまり重要でないのでここでは省略する。

### 8.2.6 ＩＲＩＡ

訪問先　IRIA : Institut de Recherche d'Informatique et d'Automatique

所在地　78150－Rocquencourt, France

訪問日　9月16日（火）

面接者　Mr. Hubert Zimmermann（Dr.）

A. 訪問の目的

フランスの国家プロジェクトであるCYCLADESの現状，プロジェクトのマネージメント方法，新しいアプリケーションである分散形データ・ベース，ネットワークに分散したジョブなどの研究成果について調査することであった。このうち，ネットワークに分散したジョブ，分散データ・ベースの研究に関する報告は6章，7章でおこなったので，ここでは省略する。

B. 調査内容

CYCLADEの開発方法

CYCLADESの目的は，次の4つである。

① 分散形データ・ベースの実現

② プロトタイプ・ネットワークの実験

③ 産業界のサポート

④ 標準化（ネットワークの相互結合など）

このプロジェクトにおけるIRIAの役割りは，調整およびマネージメントである。最初は4人でスタートし，現在は5人が常駐し，この他に15～25人のメーカなどからの人間がいるが，入れ替わりがはげしい。実際にIRIAがおこなっている仕事は次のとうりである。

① CIGALEパケット交換網の開発

② CIGALEの運用

③ プロトコルの設定

④ 基本となるものの開発

⑤ 参加センター間の調整

これらの仕事は，IRIA自身でおこなうものと，産業界と協同しておこなうものとがある。この産業界というのは，CIIのようなコンピュータ・メーカ，SESA，CAP－SOGETI，TELESYSTEMES，CERC，CISIのようなソフトウェア・ハウス，TITNのようにハードとソフトの両方をするもの，ＴＶＴのように端末を作るところをいう。

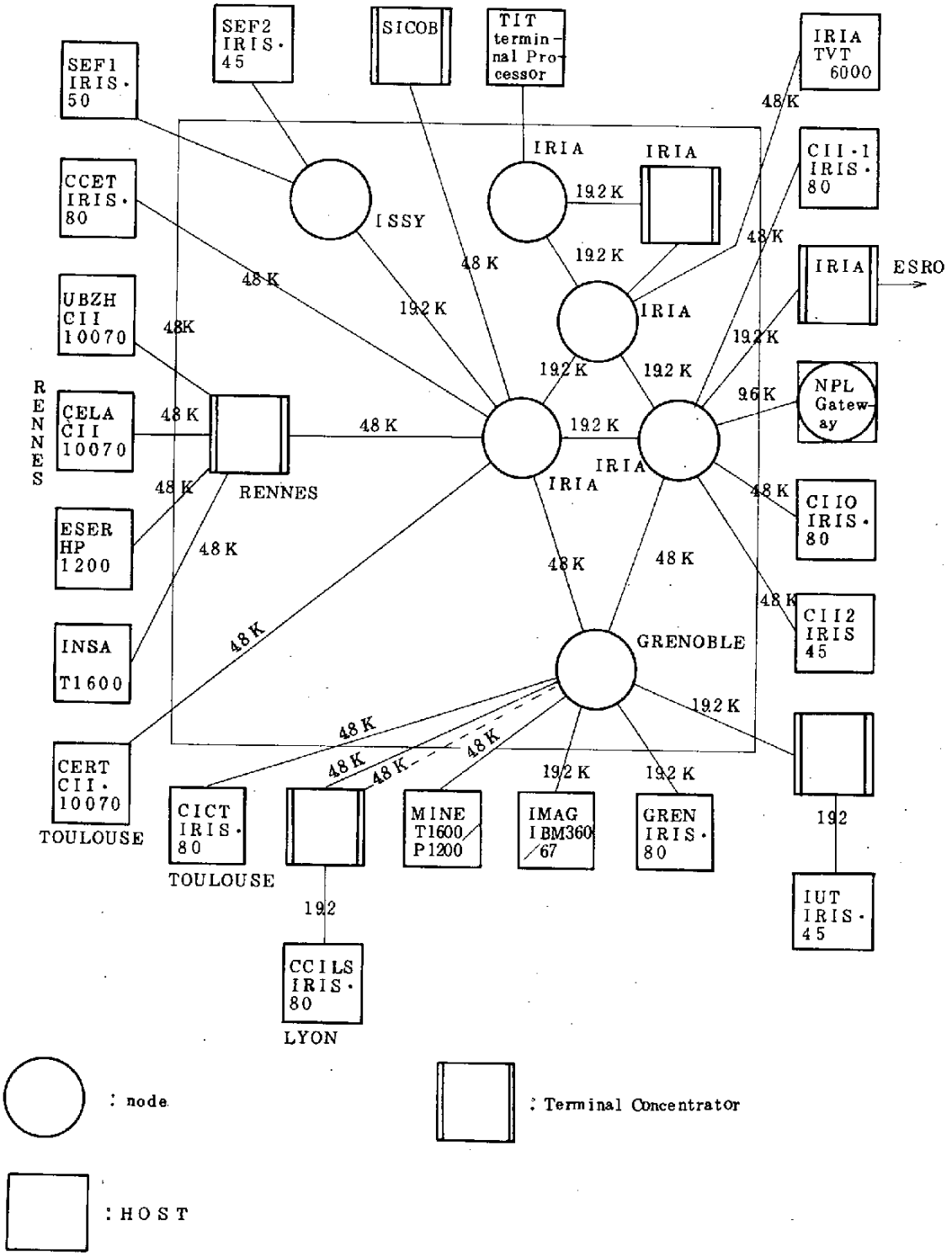



図8－13　CYCLADESの論理マップ

IRIAと産業界の関係は，コマーシャル・ベースで契約しておこなうもの，ネットワークに関する情報などを得るために無料で参加しているものなどがある。これらの人間に，参加センターの者を合わせると現在約70人がCYCLADESに参加している。

CYCLADESの現状と今後

CYCLADESの9月16日現在の論理マップは図8－13のとうりである。IRIAにネットワークのコントロール・センターが設置され，回線およびノードの障害監視をおこなっている。

他のネットワークとの結合に関していえば，NPLとはすでに結合されているが，EINとはできるだけ早い時期に結合する予定である。これとは別にEurop－Spaceのネットワークと結合したので，これのもつドキュメンテーションにアクセスすることができる。

ネットワークの拡大としては，マルセーユなどの南仏および北仏にのばす計画がある。また現在の参加センターは研究機関ばかりであるが，大蔵省，国防省，産業省などと実際につなぐということに関してディスカッションしているところである。

パケット交換網は現在10 a. m. から6 p. m. までの稼動であるが，75年末までに12時間稼動，さらに76年には24時間稼動になる予定である。

CYCLADESの成果としては，パケット形式の標準化案，End－to－Endプロトコル（HOST－HOSTプロトコルと同じ）案などの国際標準化に関するものと，商用化されたシステムの実現というものがある。

商用化されたものには，プロトタイプのCIGALE aに対して，CIIによって作られた商用版のCIGALE b，CYCLADESの商用版であるCYCLADES bisなどがある。

### 8.2.7 Grenoble大学

訪問先　École nationale supérieure d'informatique et de mathématiques appliquées.

所在地　B. P. 53
38041 Grenoble－cedex, France

訪問日　9月18日（木）

面接者　Professor J. du Masle

A. 訪問の目的

ここを訪問先として選んだのは，SOCとCYCLADESの両プロジェクトに参加しており，特にCYCLADESにおいて最も精力的に参加し研究開発を進めている。このためここへ行けばネットワークに参加するユーザの意見を聞くことができると考えたからである。

B. 調査内容

グルノーブル大学は，IBM 360／67とCII IRIS 80の2台のHOSTを保有してCYCL

ADESに参加している。ここでは，高度なアプリケーションについても説明を受けたが，その内容は，第6，7章においてすでに述べたので，ここでは省略する。

ホストとノードの結合

IBM 360／67は，2703と2701という装置を持っている。2703は4.8kbit／secまでしかサポートしておらず，より高速なものが必要であった。2701はPDP 8をつけるためのポートと，1つのBSC手順を収容するポートを持っている。IBMは9.2kbit／secまでしかメンテナンスしないが，高速にするために19.2kbit／sec の回線を使いこれを半二重手順で使用することにした。このために，他が使用しない手順をCYCLADESに持ち込み，悪い結果をもたらした。

他に種々の結合方法がある。それらの次のようなものである。

① ネイティブ・モードでプログラムされたIBM 3704／5

② CTCリンクでフロント・エンド・プロセッサとS／360と結合する。

③ 直接にS／360のチャネルにマイクロ・プログラムを持った装置をつける。

④ ターミナル・コンセントレータと結びついたノードとして働くフロント・エンド・プロセッサ（この方法は，IRIS 45で使用され，将来我々のところにあるIRIS 80にも使われる）。

この方法はIBM 3705で容易にインプリメントすることができる。

Grenobleのノード

Grenobleのノードは，図8－14で示すように6台のHOSTと他のノードへの2回線を持っている。

現在のところ，ノードは8本のポートしか持っていないので，これ以上のHOSTを結合できない。このために，ノードに16本のポートが持てるようにする予定である。

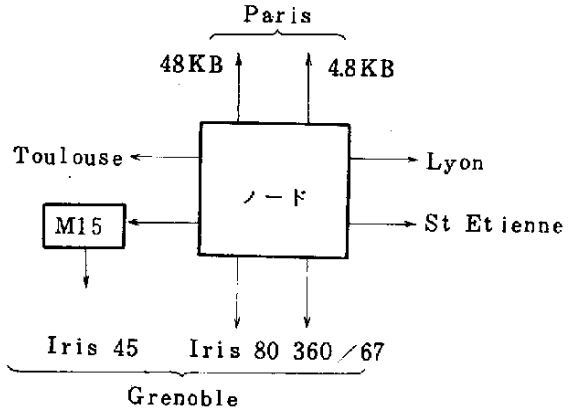



図8－14　Grenobleノード

HOSTとノードの結合の信頼性を上げるためにはHOST・ノード間を二重に結合する方法がよい。さらに信頼性を上げるには，二重に結合したノードを変える方法がよい。グルノーブルのノードはパリにあるコントロール・センターによって監視されている。故障したときには，S／360のコンソールにメッセージが示されるので，専任のオペレータは必要ない。このノードは非常に安定しており，我々が訪問する直前にGrenoble一帯の停電によってダウンするまで4ケ月間全くトラブルがなかった。

ノードの設置場所としては，センター内部よりも，ＰＴＴの局に設置した方がよい。これは，図8－15でもわかるように，多くの回線が局を経由することによって安定性が下がるからである。

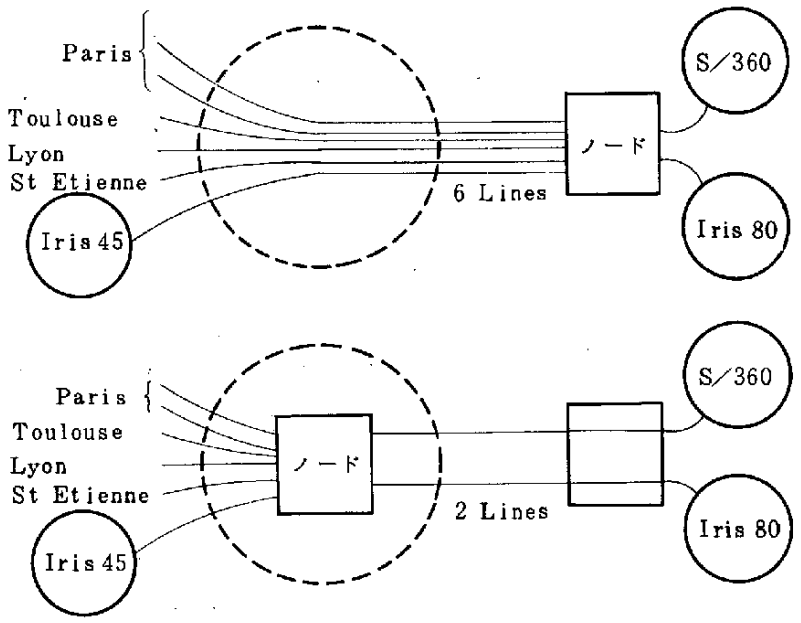



図8－15　ノードの設置場所

ソフトウェア

S／360のオペレーティング・システム（OS／MVTとCP／CMS）はネットワークを前提として作成されていないので，ハイレベルのアプリケーションを考えるときに非常に制約となる。

CP 67ではノードとHOSTの結合は，バーチャル・マシンにまかされた実デバイスとしておこなわれている。このマシンはリアル・タイム・アプリケーションを実行しなければならないので，ネットワークのようなアプリケーションには不向きである。

一つの代替の方法は，CYCLADESのソフトウェアをCP／67の核として組み込むことである。これはCP／67の日常業務に非常な影響をあたえる。将来は，フロント・エンド・ミニコンピュータあるいはマイクロプログラム化されたデバイスにこの種のソフトウェアを持っていきたいと考えている。

CYCLADESのソフトウェアサポートをいくつかのバーチャル・マシンに分散することも考えられた。これには何らかの形でバーチャル・マシン間のコミュニケーションを必要とする。VM／370ではダミーCTCを使用することによって解決されている。S／360に結合された特別なCTCを設計した，そして近い将来それをテストしようと考えている。

MVTでは，1メガバイトの記憶装置のうちわけは図8－16のとうりである。これをみれば明らかなようにシステムが，記憶装置の約50％をしめている。この主な原因はASPである。このために，CYCLADESソフトウェアをASPの中にインプリメントしなかった。そして将来ASPは使用しないようにしようと考えている。

| | | |
|---|---|---|
| MVT | 128 K | 174 K |
| Link Pack Area | 46 K | |
| ASP | 200 K | |
| RDR | 52 K | 266 K |
| WTR | 14 K | |
| CYCLADES | 110 K | 120 K |
| ASPTASK | 10 K | |

図8－16　MVTでの記憶装置占有量

### 8.2.8　C. T. N. E.

訪問先　Compania Telefonica Nacional de Espana

所在地　Avda del Brasil 17. Madrid 20. Spain

訪問日　9月22日(月)

面接者　Jesus Manjarres Hurtado

他4名

A. 訪問の目的

C. T. N. E.は1971年からパケット交換サービスをおこなっており，訪問した時点においては唯一の商用パケット交換網であった。

ここを訪問したのは，パケット交換網の運用および料金体系などについて調査したかったからである。

B. 調査内容

概　要

C. T. N. E.は，電話およびデータ伝送をサービスする半官半民の会社である。株式の大半は国に保有されているが，残りは大手の銀行が保有している。1947年まではI. T. T.が多くの株式を持っていた。この関係からか，C. T. N. E.とI. T. T.との関係は密接である。

1968年以前は，ネットワークは，航空会社などの数社のものがC．T．N．E．の回線を借りて稼動していた。1968年に大手の銀行が1000台以上の端末からなるネットワークを計画したが，回線費用が高くつきすぎるという問題があった。これがきっかけとなって，C．T．N．E．は，メッセージ交換を安くサービスするためにこの分野で公衆サービスをおこなうことになった。

このパケット交換サービスは，1971年から開始され，1974年から黒字になった。この網の利用者は，900端末，処理コンピュータは8台であり，4つが銀行，1つが運輸会社，3つが工業製造会社である。

C．T．N．E．のパケット交換網は，マドリッドとバルセロナにノードがあり，UNIVAC 418である。これらは安全性を高めるためにデュアル・システムになっている。メッセージ・スイッチングのためにIBM 3968，ターミナル・コンセントレータとしてHoneywell 716を使用している。このようにノードを大形機とした理由は，6年前ではミニコンピュータがあまり良くなかったからであり，現在では中形ないし小形を使用することも考えている。

このサービスをおこなうのにかかっている人間は次のとうりである。

| | |
|---|---|
| アナリスト | 10 人 |
| スペックを決める人 | 25 人 |
| プログラマ | 35 人 |
| オペレータ | 100 人 |

提供されているサービス

現時点で提供されているサービスは以下の5つである。

① リアル・タイム・サービス

ユーザ端末と処理コンピュータ間での交信をサービスするもので，即時応答を要するものに利用される。

このサービスにおいては，端末からのメッセージを簡単にするために，端末ごとにどこのコンピュータに宛てるものなのかを前もって決めておくことができる。

このサービスの応答時間の80％が2秒以下である。

② 特殊大量データ伝送サービス

このサービスは，ユーザ端末間あるいはユーザ端末と処理コンピュータ間での大量のデータを低いエラー・レートで送るものである。

③ メッセージ交換サービス

このサービスは，ユーザ端末間あるいは処理コンピュータ間でのメッセージの交換をおこなうものである。

これの特徴は，どこの端末あるいはコンピュータにでも送ることができること，メッセージは検索できること，コードやスピードが変わっていてもよいことである。

④ 内部ネットワークサービス

このサービスは，ネットワーク装置間での内部機能に関する情報を送るのに利用される。

⑤ 加入者センター間の相互結合サービス

このサービスは，異なった加入者の処理センタが構成しているネットワーク間でのメッセージの交換をするものである。このサービスの応答時間の優先度はリアル・タイム・サービスと同じレベルでおこなわれる。

料金体系

このネットワークでの定額制と従量制を合わせたものであり次のようになっている。

① 定額制

端末とターミナル・コンセントレータ，あるいは，ノードの処理コンピュータ間は基本的には距離に比例した固定費を支払う。

表8－2　コンピュータの固定料金

| 伝送速度（kbps） | 固定料金（月間） |
|---|---|
| 4.8 | 4,000～30,000ペセタ（約 20,000～150,000円） |
| 9.6 | 8,000～60,000ペセタ（約 40,000～300,000円） |
| 48. | 25,000～250,000ペセタ（約125,000～1,250,000円） |

表8－3　端末の固定料金

| 伝送速度（bps） | 固定料金（月間） |
|---|---|
| 133<br>200 | 2,500～10,000ペセタ（約12,500～ 50,000円） |
| 1,200 | 2,500～15,000ペセタ（約12,500～ 75,000円） |
| 2,400 | 2,500～20,000ペセタ（約12,500～100,000円） |

② 従量制

月間固定料金に加えて，次の3つの要因によって決められたメッセージ量に比例した課金がなされる。

── 発信地と着信地の位置

── 時間帯

0時～8時，8時～14時，14時～24時の時間帯および休日かどうかにより異なる。

── サービスの形

これらの料金はだいたい1000文字当り3ペセタ（約15円）である。

③ その他

加入者がオンラインになっていた時間，モデムのレンタル料も支払わなければならない。

## 8.3 EUROCOMP

通信ネットワークに関するヨーロッパ・コンピュータ会議（The European Computing Conference on Communications Networks）は，1975年9月23日から3日間にわたって，ロンドンのヒースロー空港に近隣したヒースロー・ホテルで開催された。

参加人員は約400名であり，10セッションにわかれ，38件の論文が発表された。以下に各論文の概要を述べる。

### Contemporary Systems

| | |
|---|---|
| The UPARS Network: an extensive network servicing several diverse applications | *J C Goodlett* |
| Concepts of System Network Architecture | *J H McFadyen*<br>*E M Thomas* |
| DCTS: a data communication terminal system for the data exchange between air traffic control centres | *M Baum* |
| The state of the art in the evolution of private data communication networks | *D W Mann* |
| Digital network architecture | *N A Teichholtz* |

### Future Facilities

| | |
|---|---|
| Packet switching in a public data transmission service: the TRANSPAC network | *R Despres, A Danet*<br>*B Jamet, G Pichon,*<br>*P Y Schwarz* |
| Telenet: principles and practice | *L G.Roberts* |
| UK Post Office DDS Proposals: user and DP industry implications | *D L Hebditch*<br>*R D Bright* |

### Interconnection of Networks

| | |
|---|---|
| Interconnection of packet switching networks: theory and practice | *M Gien, J Laws*<br>*R Scantlebury* |
| Virtual call issues in network architectures | *L Pouzin* |
| Alternative approaches to the interconnection of computer networks | *P T Kirstein*<br>*D Lloyd* |
| Gateway design for computer network interconnection | *D C Walden, R D Rettberg* |

## Reliability and Security



## Communications and People



## Designs in Detail



## Modelling and Assessment



## Operational Aspects

## Services and Terminals

| | |
|---|---|
| The design of host and terminal interface systems for data communication networks | *J A Bowie*<br>*B M Wood* |
| The impact of prospective public data network facilities on terminal design | *F A K Frampton* |
| Practical experience with the telephone used as a data terminal | *G Arndt, K Stemberger* |

## Protocols and Languages

| | |
|---|---|
| Command language design for networks of processors | *I A Newman*<br>*N S Fitzhugh* |
| Network independent high level protocols | *A S Chandler* |
| Job control in a heterogenous computer network | *P Schicker, W Baechi*<br>*A Duenki* |
| Inter-mainframe communication via spooling RJE machines | *S Cardy, P N Jackson* |

Contemporary Systems

(1) The UPARS Network : an extensive network servicing several diverse applications

James C. Goodlett

United Airlines, USA

United Airlines UPARSネットワークは，広範囲のデータ・コミュニケーション・ネットワークである。これは，航空会社の座席予約，荷物手続，食事サービス管理情報システム，Western International Hotelsの予約システムとして使用される。現在このネットワークは，113都市，15の連合予約オフィス，25のホテル，21の食事調理所に対してサービスしている。これは米国のほとんどと，カナダ，メキシコシティまで57,000マイルに広がっている。

この論文は，UPARSネットワークの概要と特徴のいくつかを述べる。

(2) Concept of System Network Architecture

J. H. McFadyen, E. M. Thomas

IBM System Development Division Kingstone Laboratory, USA

データ処理システムは，数個の周辺装置を持った中央処理装置から，情報処理ネットワークを作ることまで急速に発展した。ネットワークの構成とアプリケーション・プログラムが非常に複雑になったので，適応性の不足はシステムのより良い発展を妨げた。しかしながら，より重要なことは，より融通のきくネットワーク設計が考えられる統一的なネットワーク構造がなかったことである。

この論文は，ネットワークの発展の概要，利用者とデータ処理システムの関係の変化の概要についてを最初に述べる。それから，プログラマブルなプロダクトとしてのネットワークは，利用者の要求に対する応答としてのサービスをするシステムであると見られるという概念を導入する。

System Network Architectureは，サービスに対する利用者の要求を満たすために開発された統一的なネットワーク構造である。種々のタイプのサービスに対する議論のあと，ＩＢＭが最近発表したこれらの概念に基づいたものを例として述べる。この論文は，System Network Architectureの基本概念の概要を述べて終っている。

(3) DCTS : a data communication terminal system for the data exchange between air trafic control centres

M. Baum

Eurocontrol Maastricht U. AC., Netherlands

Data Communication Terminal System（DCTS）は，MADAP と外部センター間の航空トラフィック・サービス（ATS）や気象に関するメッセージの伝送をおこなうための独立的なフロント・エンド・コンピュータ・コンプレックスである。

航空トラフィック制御センター間のデータ交換は，制御されているフライトに関して誤りのない最新の情報をあたえる重要な役割りを演ずる。ヨーロッパの隣接した制御センター間の調整は，ハードウェア，ソフトウェア，オペレーションの違っているシステムを適応させる。国内における交換網で使用されている伝送手順の違いは，受け入れられなければならない。そしてこれはＤＣＴＳに対して異なった伝送手順を同時に処理すること，ある手順から他のものへの変換を強制する。コンパティビリティの要求と，データの安全性が応答性よりも優先された。DCTSは，１日24時間，週７日間連続でデータのロスなしに稼動するよう設計された。このためチャネルあるいはシステムそれ自身が悪くなったとき，機能を切り離すことに関するいくつかの段階があたえられるよう特別な配慮がなされている。

(4) The state of the art in the evolution of private data communication networks

David W. Mann

Logica Limited, UK

この論文は，プライベートなコンピュータがベースとなったデータ交換網に対する現時点の要望について述べている。これは，現在のところ満足させるのが最もむつかしい利用者の要求のクラスを示し，それらに最もよいシステムの構造を示す。

このわく内で，特別な領域の問題を解決するために開発された技術の状態について考えてみる。

(5) Digital network architecture

Nathan A. Teichholtz

Digital Equipment Corp. USA

コンピュータ・ネットワークの主な利点は，オペレーティング・システムの伝統的な領域（タイム・シェアリング，リアルタイムやバッチ）にまたがった問題に対する完全な解決策をあたえるた

めに，異なったコンピュータがそれぞれ，異なったセットのアプリケーションを用意し，それらを相互に結合することを可能にすることである。

この論文は，DECにおいて広い範囲のコンピュータ（PDP－8，11，15，DEC－10）とオペレーティング・システムにネットワーク機能をあたえるために利用されているアーキテクチャについて論じている。このアーキテクチャは，一般的であってすべてのDECシステムに適応させることができ，多くのDEC以外のものにも，同様に適用できる。

Future Facilities

(6) Packet switching in a public data transmission service : the DATAPAC network

A. Danet, R. Despres, B. Jamet, G. Pichon, P. Y. Schwartz

Centre Commun d'Etudes de Télévision et Télécommunications-France

公衆パケット交換サービスは，いくつかの必要条件を持っている。

— 利用者のプロトコルに対する簡素性と安定性

— 間違ったあるいは悪意のある使用法のときにでも安全性が確保され，種々のサービスがされること。

— 制御なしに他のネットワークと相互結合できる。

— 種々のトラフィック・パターンに対してふさわしい料金であること。

この論文は，CCITTのパケット・モード・オペレーションに関する現在の成果，サービスに対する経済性，上記のものを考慮したTRANSPACにおいてなされた主なる選択について記述している。

利用者が使える基本的なサービスは，パケット交換に対してのバーチャル回線（交換あるいは永久的な）のあつかいである。多重アクセス・ターミナルの場合には，ネットワークはセレクティブなフロー制御をおこなう。このようにして，データはネットワークの中で破壊されないし，バッファの要因でバーチャル・コールがブロックされることはない。TRANSPACのアーキテクチャは，ローカル，リモートのマルチプレクサー，コンセントレータを持った6～20台のスイッチング・ノードによって構成される。データ・グラム・サービスの追加の可能性に関しては，コンジェスション制御，サービスの質，加金体系に対しての研究を進めているところである。

(7) Telenet : principles and practice

Lawrence G. Roberts

Telenet Communications Corporation, USA

この10年間のエレクトロニクスとコンピュータ・テクノロジーの発展は，データ交換の市場の要求と交換システムの開発に対するテクノロジーの両者に実質的な変化をきたした。計算コストの急

速な低下は，非常に多くのものが以前には人間によってやられていたが，これを自動化することによって経済的になるため，会話形あるいはトランズ・アクション・オリエンティッドなデータ交換の需要が今までになかったような率で増大している。計算コストにおける同様な傾向は，トランズ・アクション・オリエンティッドなデータ・トラフィックを非常に効率良くあつかうパケット交換原理を使用して全自動の交換システムを急速に開発することを可能にした。パケット交換は，プライベート・ネットワークの開発段階から出現し，公衆の通信サービスとして多くの国でとり入れられている。米国におけるTelenetのパケット交換サービスの導入に対する市場の反応は，利用者の全く共通の要求としてサービスされるデータ通信を初めてみつけだしたという形になっている。

(8) UK Post Office DDS Proposals : user and DP industry implications

R. D. Bright, Post Office, U. K.
D. L. Hebditch, Pliener Associates Ltd., U. K.

この論文は，英国の郵政省によっておこなわれた仕事の目的となっているものの背景，方法論と予備の結果を記述し，提案されている範囲のデジタル・データ・サービスに対する同様な要求についても述べている。

## Interconnection of Networks

(9) Interconnection of packet switching networks : theory and practice

M. Gien, IRIA, France
J. Laws, R. Scantlebury, NPL, U. K.

現在のところかなりの数のパケット交換技術を利用したコンピュータ・ネットワークがある，そしてある時点で必ず，利用者が他のシステムのリソースを得ることができるようにこれらのいくつかを共に結合するという問題が持ち上がる。結局これは，公衆通信機能を提供することによってなされなければならない，また現存するネットワークの通信システムから考え合せなければならない，しかし最終的にはうまく設計された通信ネットワークであたえられるだろう。現在おこなわれている投資を犠牲にしないで現状から望ましい形にまでどのように推移するかという永遠の問題がある。通信ネットワークのレベルばかりではなく，ネットワークに結合されたコンピュータ間の標準プロシージャのレベルまでのCYCLADESとNPLのネットワークの相互結合の実験が，IRIAとNPLによっておこなわれた。

(10) Virtual call issues in network architectures

Louis Pouzin
IRIA, France

仮想回線の概念は，パケット交換網における end-to-end の制御メカニズムのセットに対して使用される。結合（Liaison）と呼ばれる同様のメカニズムは，コンピュータ・ネットワークにお

けるより高位のレベルで見られる。これらの特徴が，ポートのアクセス，エラーとフロー制御を主体として評価されている。

仮想回線の種々の形態は，現存するあるいは計画中のパケット網に含まれている。しかし，ある種のものは全く持っていない。end-to-end 制御メカニズムはより高位において常に存在するので，パケット網において仮想回線がコスト的に価値があるか明らかではない。

仮想回線を持ったコンピュータ・システムのインターフェースは，特に高位において結合と組み合わさったときにいくつかの問題点が持ち上っている。最後に，実体を持つ回線の代りとしての仮想回線を考えることが影響の最も少ないアプローチであることを述べている。

(11) Alternative approaches to the interconnection of computer networks

David Lloyd, Peter T. Kirstein

University College London, U. K.

利用者が相互にリソースを共有したり交信したりできるようにコンピュータ・ネットワークの相互結合をすることによる利益が増大しつつある。これを達成するための代替の手段は，政策的，技術的，経済的要素の背景を評価しなければならない。この論文は，相互結合に対する主たる代替のアプローチ，実際上の相対的な重要性を決める要因について概説する。

(12) Gateway design for computer network interconnection

David C. Walden, Randall D. Retterg

BBN, USA

Gatewayと呼ばれる実体を通してのパケット交換網の相互結合に関する意見が述べられている。Gateway virtual networkが提案され，プロトタイプ的なインプリメンテーションが述べられている。

## Reliability and Security

(13) Security and design considerations in a distributed criminal justice computer network : the State of Missouri's approach

Sarwar A. Kashmeri

Regional Justice Information System, USA

この論文においては，最初にミズリー州における犯罪裁判情報システムについて述べ，次にハードウェア，結合，ファイルに関して州のネットワークについて述べる。それから，ソフトウェアとサイトの観点からネットワークに組み込まれた安全性の特徴について述べる。このネットワークを制御する政策的な考え方と，この3つのコンピュータに分散されたシステムとして達成された効果について記述しようと考えている。ミズリー州は，両側の2つのメトロポリタン・エリア（Kansas CityとSt. Louis）とその間の田園地域で特徴づけられる。考え出されたネットワークは

これをはんえいし，大都市のニーズに対してサービスすると同時に州全体の先進的な例となっている。

(14) A minicomputer network to enhence computer security

James A. Painter

Defense Communications Agency, Department of Defense, USA

この論文は，受け入れられるような程度のコンピュータの安全性でインプリメントされた仮想的なコンピュータ・ネットワーク・システムにおいて，ハードウェアとソフトウェアの正当でない変更の問題について述べている。コンピュータの安全性は，制御を共有することと，リソース共有形の多数ユーザをおのおの独立させるようなハードウェアとソフトウェアより成っている。"minimonitors"と呼ばれるミニコンピュータ群よりなるシステムは，故意あるいは故意でない修正を発見する機能があたえられている。minimonitorsは安全監視機能もあたえることができる。"analizer"と呼ばれる付加ユニットは，minimonitor ネットワークのためのネットワーク制御と分析をなす。minimonitor と analizer のそのようなネットを導びきだす開発されたプログラムのことが述べられている。

(15) Approaches to controlling personal access to computer terminals

Ira W. Cotton, Paul Meissner

National Bureau of Standards, USA

タイムシェアリング・システムとコンピュータのネットワーク化の出現は，コンピュータ・ユーザの増大をひきおこした，ユーザの多くはサービスするコンピュータから離れたところにいる。これは，正当でないユーザがコンピュータにアクセスする機会を増し，正当なユーザを識別し，認証する問題に注意が向けられることになった。米国においては，最近法律化された1974年のプライバシー法の必要条件としては，連邦政府による個人情報のとりあつかいにおいていくつかの防備を要求している，そして個人の識別は，この法律の実行の重要な側面である。

この論文は，個人の識別と認証の種々のアプローチを考えている，すなわち個人が知っていることにもとづくもの，個人の持ち物および顔だち，手書き文字，声，指紋，手形などの外観的なものである。評価基準のセットが種々のアプリケーションにおける個人の識別システムの選択のガイドとして述べられている。現在利用されているシステムは，多くの場合間違った認識をする。そしてそれ故に，誰によって何の為にアクセスされているかという履歴をとるための検査の機能をも含んだ種々の防備を使用するハイアラキーを持った安全性システムが合同して使用されるべきであるということが指摘されている。

Communications and People

(16) The effect of a data communications network on a large laboratory site

K. A. Bartlett ; P. T. Wilkinson
National Physical Laboratory, UK

ＮＰＬデータ通信ネットワークは，２年以上稼動してきた，その間にこれは基本的なサイト・サービスになった。それは共用網として働き，非常に一般的な通信機能の範囲を提供するので，alphanumeric data baseや集中ファイル蓄積を含むいくつかの互いに独立なリモート・アクセス・サービスの開発が可能になった。ネットワークの影響は，小形コンピュータの利用の増加やタイプ・サービスの要求の減少というように広い範囲に現われてきた。

ネットワークの利用において得られた経験は，分散形のコンピューティングの利用とコンピュータの遠隔地からの利用におけるより進んだ開発の方法を示している。

(17) Viewdata : an interactive information service for the general public

S. Fedida
Post office Research Centre, UK

コンピュータでない利用者に使われるコンピュータに基づいた情報媒体とサービスであるＶＩＥＷＤＡＴＡの概念は，コンピュータに基づいた情報システムの"全体としてのコスト── 効果"の引用によって説得される。そのようなシステムの作成手段は，低コストで満足のゆく形の設計，それに関係するキー・パッドを持った電話システムに結合された丈夫な信頼性の高い端末，内部あるいは外部のデータ・ベースにアクセスできるように分散され結合されたコンピュータ・システム，効果的な利用のために何の訓練も受けることを必要としない情報検索のための"自然言語"のプロトコル，性能を最大にし，それ故コストを最少にする設計されたソフトウェア構造を含む。

(18) The impact of integrated message processing facilities on administrative procedures and inter-personal interactions

Peter T. Kirstein, Stephen R. Wilbur
University College London, UK

この論文は，コンピュータ・ネットワークの利用がもたらし得る個人メッセージ・サービスの改良に関するものである。メッセージの組立て，分散，評価，ファイリングの総合化はＰＴＴによっておこなわれている現存のサービスでは容易ではない，あるネットワーク（ARPANET）であたえられる機能が述べられている。商用メッセージ・システムにおいてあたえ得るサービス，ある現存するシステムの経済性およびＰＴＴの独占になっているこれらの活動に対するコメントが述べられている。

(19) Euronet project

G. W. P. Davies
Commission of the European Communities, Luxembourg

EURONETという科学技術情報のためのネットワークが，ヨーロッパ社会のために開発されつ

つある。EURONET構築のための3ケ年計画の活動がいま行われている，それの目標は，特別な分野におけるデータ・ベース・システムの開発，情報をとりあつかうための共有ネットワークの構築と情報検索技術の振興を含んでいる。

この論文は，EURONETのプログラムで行われている主な活動を述べ，特に潜在的な利用者の広い範囲の要求によって影響をうける設計の方法について強調している。

Designs in Detail

⑳ The design of the packet switched network for the EIN Project

F. Poncet　　J. B. Tucker

SESA, France　　Logica Ltd, UK

EINプロジェクトの研究をサポートするパケット交換網の役割りが調査された。ネットワークに対する要求が述べられ，いかに問題に対処したかが説明されている。通信の標準の問題について特に言及されている。

㉑ RPCNET features and components

P. Franchi, G. Sommi

IBM Scientific Center, Pisa, Italy

この論文においては，REEL Project Computer NETwork（RPCNET）の一般概念が示されている。第一の機能的な違いは3つの層により区別される：通信機能，インターフェース機能，アプリケーション。

次にそれぞれの層は異なった機能のボックスに分けられ，そのボックスの概略が述べられる。

最後に，計算機，ハードウェアの特徴，インプリメントが計画されているオペレーティング・システムが述べてある。

㉒ The problems of linking several networks with a gateway computer

Peter L. Higginson, Andrew J. Hinchley

University College London, UK

この論文は，University College Londonにある実際のネットワーク・ゲートウェイを概説する。ゲートウェイに対するオペレーティング・システムの機能とともに内部的なコネクション・プロトコルの利用について議論する。それからゲートウェイ機能の利用と関連する問題が評価されている。

㉓ Network node with integrated circuit／packet switching capability

Christian J. Jenny, Karl Kümmerle, Helmut Bürge

IBM Research Laboratory Data Communication Center, Switzerland

現在および将来の公衆データ網は2つの交換法 ── 回線交換（C／S）あるいはパケット交換

（P／S）の1つを使う。将来の要求に対する予想の不確定さは，2つの方法のうちのいずれが長い期間にわたってよく合うかという論争点をあいまいにする。結果として柔軟性が近い将来の網のデザインとしての第一の目標となる。

この目標を満足させるように提案されたアプローチは，1つの公衆網の中で回線交換とパケット交換を統合することである。P／SはC／Sネットワークを基礎として作ることができる。C／S機能と共有しないパケット交換の部分は最少 $\Delta$ でなければならない：

$$P/S = C/S + \Delta$$

この論文は，そのような公衆網の交換ノードについて記述している。統合されたネットワーク概念と同じように，回線交換ノード構造でパケット交換を課するような新しいノードの概念が述べられている。

記憶モジュールに組み込まれた非常に小さい単純なプロセッサがパケットをあつかう。考えられる環境の下では，プロセッサの数は40個を越えるかもしれない。

マルチ・プロセッサ・システムのよく知られた飽和効果は ── すなわちプロセッサ・モジュールの増加に対して追加したプロセッサ・モジュールあたりのスループットの増加の減少 ── 特別なデータ通信装置を使用することによる新しいモジュール間通信の概念で克服された。システムのスループットは解析され，モジュールの数とほゞ線形であることが示されている。

## Modelling and Assessment

(24) An assessment of the performance of packet switching networks

David C. Wood, Ranvir K. Trehan

The MITRE Corporation, USA

この論文は，ARPAネットワークのユーザが得られるパフォーマンスの評価をおこなっている。そのようなネットワークにおいて間接的にユーザのコストやパフォーマンスに影響をあたえる種々のファクタが示され，パフォーマンスを改良するためのアプローチをしている。

現在のARPANETのアーキテクチャの弱点が示され，コストとパフォーマンスの特性を改良した別のネットワーク・アーキテクチャが提案されている。

(25) Simulation of a data communication network

T. H. Beeforth, M. C. Daniels

University of Sussex, UK

この論文は，異なったルーティング方法に関して，データ交換網のシミュレーションを述べたものである。シミュレーションは，25ノードのネットワークに適用され，ルーティング法の効果は，変化するロードの下でトラフィックの要求に対して得られたシステムの応答を考慮することによって評価された。

いかなるペアの交換ノード間でも１つしかルートがないような固定ルーティングが最初にシミュレートされた。もしそのルートのどこかが稼動していないときは，代替のルートは考えられない。システムのパフォーマンスはもちろんもとのルートの特徴に従属する。

２つのタイプの適応ルーティング法がシミュレートされた。集中方式：ネットワークの１ケ所で最新のネットワークの統計が保持されており，これらの統計はルートを決める要求によって利用される。システムのコンジェスションはこの方式では大巾に低下した，そしてシステム利用率が良い値になる。

前進方式：ルートはそれぞれのノードにおいて最も良いリンクの選択によって決まる，これは一般のローカル情報によって選択される。リンクの選択は利用率と方向性によって単純になる，またすべての利用可能なリンクにウェイトをつけることにより単純にすることもできる。

適応ルーティング法で得られた結果は一般的にいって固定ルーティングのそれよりも良かったが，集中方式のルーティングより良いものはなかった。ルーティング効果はルーティング法を選ぶときに考慮される一つの要因にすぎない，そして他の最適な要因について論じられている。

(26) Queueing system with delayed feedback and computer communication networks

F. Borgonovo, L. Fratta, F. Maffioli

Istituto di Elettronica and Centro CNR Telecomunicazioni Spaziali, Politecnico di Milano, Italy

この論文においては，一定量のサービスと同期オペレーションの仮定の下で，遅延フィードバックを持った待ち行列の分析がマルコフ過程のアプローチでおこなわれている。待ち行列の中で使われる上下限の時間が引き出されている。

この研究の動機は，蓄積交換のコンピュータ通信ネットワークにおいてパケットの伝送時間のタイムアウトの影響を評価する必要性があったからである。

(27) Out-of-sequence problem in a packet switching network : a simulation approach

Mauro Maier

IBM Scientific Center, Pisa, Italy

この論文の中で，我々はパケット交換ネットワークにおける順序の乱れの問題に対するシミュレーションのアプローチを述べている。

まず第１に，要素の一般的特徴とネットワーク・インターフェース機能があたえられる。この中において，論理的通信チャネル，基本情報装置，メッセージ・ハンドラーの特徴が解析されている。それから，パケットの紛失の問題が考慮され，タイム・アウトと順序の乱れの概念にもとづいて確率的な方法が論じられている。

最後に，統計をとるために使われたシミュレーション・モデルが述べられ，順序の乱れの現象に関する結果が報告されている。

Operational Aspect

㉘ Flow control in the packet switching networks

Dag Belsnes

Ass. Professor EDB-sentret, Universitetet i Oslo, Norway

大容量ファイルを転送するのに使用されるデータ網においては，ネットワークのコンジェスションを防ぐためにフロー制御が要求される。それぞれのコネクションのend-to-endのフロー制御を考慮するWindow方式はグローバルなコンジェスションを避けるために利用される。いかにしてよいスループットが得られ，網のオーバ・ロードが避けられるようなWindowサイズを送信側がセットできるかを示す。HOSTコンピュータの中でどのくらいのバッファが使われるべきかが討論される。

㉙ The problems of connecting hosts into ARPANET

Adrian V. Stokes, Peter L. Higginson

Department of Statistics and Computer Science University College London, U. K.

University College LondonのDEC PDP-9はARPA ネットワークのコミュニケーション・コンピュータであるTerminal Interface Message Processorに結合されている。PDP-8は，賃貸回線によってRutherford Laboratory（RL）のIBM 360／195とCambrige Computer Aided Design CentreのATLASに結合されている。PDP-9を近い将来に，他のコンピュータやネットワークに結合しようとしている。

この論文において我々は，特別の説明としてRLの360をARPAネットワークに結合するときのことを参照しながらHOSTをコンピュータ・ネットワークに結合する問題に対する新しいアプローチを示す。我々のアプローチにおいては，すべてのネットワークに従属するソフトウェアはPDP-9の中に用意される，これによって大形コンピュータにおけるソフトウェアを最少にする。我々が直前した問題とそれをいかに解決したかを述べる。このアプローチは，PDP-9を通じて他のコンピュータをARPANETに結合する場合に利用できる。

㉚ An EPSS interface service for the NPL data communication network

N. W. Dawes, I. G. Dewis

National Physical Laboratory, UK

この論文は，NPLとEPSSパケット交換ネットワークをつなぐためにNPLで現在作られているシステムを記述している。この結合は，両ネットワークのユーザが他のネットワークにあるサービ

スにアクセスすることを可能にする。最初のところは，主なる用途は文字端末から会話形サービスにアクセスすることである。

(31) Maximizing system performance by optimizing scheduling and accounting algorithms

I. A. Newman, D. W. Taylor

Computer Studies Department, Loughborough University of Technology, UK

この論文は，サービスしている環境においてシステム・パフォーマンスの定義のむつかしさを調べることから始めている。この仕事に対するいくつかのアプローチが研究され，すべてのリソースのアロケーション方式にリンクしたスケジューリング・システムとアカウンティングの利用が主張される。

そのようなシステムのモデルが記述され，ユーザが満足するような効果と，バッチ処理と会話形の両者において計算機のリソースの利用が増すことを示すために実際の例が示される。そのようなシステムのネットワークに対するアプリケーションが論じられ，良い結果が得られるであろうことを示す。

## Services and Terminals

(32) The design of host and terminal interface systems for data communication networks

B. M. Wood, J. A. Bowie

Computer Analysts & Programmers Limited, UK

データ通信ネットワークは広く使われるようになってきたので，効率とそれらに対するインターフェースのコストがしだいに利用の大きな要因となってきた。

この論文は，ネットワークのオペレーションを含む制御のレベルを明解で単純な方法で分割するシステム構造によって安いコストで高いパフォーマンスを得ようとする要求を調和させるアプローチをするために，インターフェース・サブシステムのデザインの特徴のいくつかを調べている。

(33) The Impact of prospective public data network facilities on terminal design

J. A. K. Frampton

International Computers Ltd., UK

電気通信におけるデジタル技術の出現は，目的にあった公衆データ網の計画をみちびきだした。この論文は，ターミナル ── プロセッサに結合されたあるいは回線につながったターミナル ── の構造に影響をあたえたインパクトを評価する。この評価においては最初のところでは，賃貸，回線

交換，パケット交換のような新しい機能の利用を意味しないように，パイロット開発と電気通信の領域において一般的に利用できるようになるまでの時間要因を考慮している。

評価をできるだけ意味あるものにするために，提案された機能は，現存する公衆電話，電信，テレックス網と比較された，そして比較は，すでに発表されているシステム構造の概念にもとづいておこなわれた。

(34) Practical experience with the telephone used as a data terminal

G. Arndt, K. Stemberger

Research Laboratories, Siemens AG, Federal Republic of Germany

電話は，それが非常に広くいきわたっているために，データ端末として大きな可能性を持っている。それは，すべての人に対して大きな端末コストの増大をさせることなくデータ・サービスへのアクセスを可能にする。電話をデータ端末として使う人々は，コンピュータの経験者ではない，だから特別なシステムが開発され，試行プロジェクトの一部として21000の内線を持った大きなＰＡＢＸで運転中である。

問題となるアプリケーションは，コンピュータ制御の電話番号情報システムである。これは内線の利用者が，同じＰＡＢＸに結合された他の利用者の電話番号についての情報を得られるように電話からコンピュータと会話できるようにする。特別なレコード・プログラムがシステムの稼動データをとるために動作している，そしてこれらのデータの評価から，電話はこの種のコンピュータ対話を行う媒体として全く適していることが示された。

## Protocols and Languages

(35) Command language design for networks of processors

I. A. Newman, N. S. Fitzhugh

Computer Studies Department, Loughborough University of Technology, UK

この論文は，異機種のネットワークにおいて使われる統一的なコマンド言語の設計の問題に関するものである。コマンド言語の概念構造を導きだすために利用者の要求が研究された。この素地と，ネットワークが利用される方法に対する考慮から，ネットワーク・コマンドの仕様のためにちょうどよいサブ・モデルが開発された。最後に典型的なネットワーク・コマンドの例が示され，提案されたアイデアがいかにして利用者に使われるかを説明する。

(36) Network independent high level protocols

A. S. Chandler

Network Technology, Computer Aided Design Centre, UK

通信サブ・システムの効果的な利用のためには，利用者が通信できる高位のプロトコルが存在す

る必要がある。確立された通信方法――たとえば，賃貸回線を通じてコンピュータにつながるＲＪＥ端末――においては，高位のプロトコルはオペレーティング・システムのなかにうまく統合化されている。同様な状況が，ターミナルがＣＰＵにアクセスするときにも適用される。このレベルのプロトコルはオペレーティング・システムの一部であるから，異なったメーカのコンピュータをつなごうとしたときに大きな問題が起こる。

英国の郵政省が設定したプロトコルは，ネットワークに対するインターフェースを含んでいる，しかしながらシステムがどのように使われるかを決めていない。ＥＰＳＳは実験的なものであるから，英国における恒久的な公衆網のそれは多少現在の提案のものと異なるであろう，ネットワークに対するリンクの設計を可能なかぎりネットワークのメカニズムと独立にするのが賢明である。

この論文は，ＥＰＳＳで基本ファイル転送プロトコルとしてインプリメントが提案されている１つのセットのプロトコルの階層構造を記述している。ファイル転送の運用の特徴の概要と下位のプロトコルとのインターフェースが述べられている。

(37) Job control in a heterogeneous computer network

P. Schicker, W. Baechi, A. Duenki

Cost 11 (EIN) Project ETH Computer Centre, Switzerland

分散型のスタック・マシンがネットワーク・サービスの構成で作られた。このマシンは，利用者が分散形のジョブを実行することを可能にし，並行処理における同期の問題を解決している。統一的で高級なジョブ制御言語で書かれたジョブの制御は，ネットワークのどこかにある仮想的な装置の利用を可能にする。大容量転送サービスはＩ／Ｏシステムを拡張し，仮想装置に対するアクセスを可能にする。

PASCALのような記法を使ったならば，ネットワーク制御ジョブの例というものは，ユーザ・レベルの言語の必要性を説明し，ジョブが実行される環境を示し，ファイル転送，エラーのあつかい，プロセスの創成およびパラメータの受け渡しを行うための媒介物となる。

(38) Inter-mainframe communication via spooling RJE machines

P. N. Jackson

Computer Technology Limited, UK

我々はＣＰＵ間のファイルの転送に関する問題を議論し，すでに存在している機能を利用するメカニズムが必要であるという結論を出した。そのようなメカニズムは，もし我々が異なったＣＰＵに結合されたバッチの端末間でのインターフェースが存在することを仮定すれば可能である，そしてスプーリングＲＪＥ端末の存在がこのインターフェースの供給を可能にするかを示す。それから，結合を達成するソフトウェアについて述べ，そのかかわりあいについて議論する。

## 8.4 パケット交換ネットワークの設計の問題点

—— ARPAネットワークを中心として ——

### 8.4.1 はじめに

過去数年に渡って，ARPAネットワークでは運用上の問題点の分析，ネットワーク・プロトコルの評価・改定，などの作業を行って来た。

ここでは，パケット交換ネットワークの設計の問題点について述べる。

なお，ここで参照した文献は，

BBN Network Measurement Note 26, INWG Note 64, Report on Network Design Issues, October 31, 1974.

である。

### 8.4.2 設計の考慮事項

ARPAネットワークの設計で重点をおいた考慮事項を次に示す。

a. 伝送遅延を一定時間以内にする —— 伝送回線の伝送帯域，ノード中のキューイング，回線エラーなどの原因により伝送遅延時間が増加するので一定時間以内におさえる。

b. 伝送効率を上げる —— ネットワーク・オーバヘッド（ハードウェア・オーバヘッドとソフトウェア・ハードヘッダの両方），回線エラーなどの原因より伝送帯域が減少するので，できるだけオーバヘッドを省くように設計する。

c. パケット伝送エラーに対する対策 —— 回線エラーなどの結果として，パケットが損失したり，あやまってパケットを再送したりする事態が起るので，それに対処する。

d. パケットのシーケンシング —— パケットはルーティングや再送によって目的地には送信した順序では到着しないので，送信された順序に並べ換えて受信HOSTに渡す必要がある。さらに発信地ＩＭＰと目的地ＩＭＰの間では次の問題点がある。

e. バッファ容量が有限である —— ノードのバッファ容量は有限であるので，パケットの輻輳によるロックアップなどのデッドロック状態を起さないように制御する。

f. 発信地ノードと目的地ノードの処理速度（bandwidth）が異なる …… ノードに結合されているHOSTの受信能力がそれぞれ異なるため，目的地ノードにパケットが充満する可能性があるので，それに対処する。

上で列挙した６つの考慮事項を満たすため，ARPAネットワークでは次に示す基本的な技術で対処している。

a. バッファリング —— ネットワーク中では有限の遅延があり，スループットを増加させるためには，たくさんのメッセージを網中にバッファリングしておくことが望ましい。

b. パイプライン化 ── ネットワークは有限の伝送帯域であり，遅延を減少させるためには，メッセージを小さな伝送単位であるパケットに分解して，それぞれを独立に伝送する必要がある。回線の伝送帯域が小さいので，hop毎に一度にメッセージ全体を伝送すると，遅延を増加させる。メッセージをパケット化することによって，ＩＭＰはメッセージの第１パケットを他のパケットを待つことなしに独立に目的地に運ぶことができる。メッセージがＰパケットで，目的地がＨhop離れている場合，遅延はＰ×Ｈではなく Ｐ＋Ｈ－１に比例する。比例定数は１パケットを１ hop前進させるに必要な時間である。

c. エラー制御 ── 目的地では，エンド・ユーザに正しいデータを渡すためには，パケットの紛失やパケットの重複をチェックする必要がある。メッセージの再送要求は，メッセージの再送のトリガとなるので有効である。パケットの紛失や重複を検出するため，パケットにはユニークな番号が付けられている。

d. シーケンシング ── パケットは発信された順序では目的地に到着しないので，目的地ではメッセージ番号を使用してメッセージの発送された正しい順序に並び換えなければならない。

e. ストレージ・アロケーション ── 発信地と目的地には有限のバッファ容量しかないので，その使用状況を管理しなければならない。一つのアプローチは，発信地はメッセージのコピーを持ち，メッセージを送信し，一方目的地は受信用のバッファがなければそのメッセージをすてるようにする。もう一つのアプローチは発信地側が目的地へバッファ確保要求を送り，バッファを予約し，それからメッセージを送り，発信地側ではそのメッセージのコピーを持つ必要をなくす。最初のアプローチは目的地にバッファを確保するというセットアップ時間が必要でないので低遅延となる。一方，二番目のアプローチはバッファが既に予約されているので再送が不必要となり高スループットとなる。もしメッセージ伝送でエラーが起っても問題にならなければ，発信地にも目的地にもバッファを特別に用意する必要がない。しかし，遅延，スループット，そして伝送の信頼性を同時に最適化するためには多少のコストを費しても発信地と目的地の両方にバッファを用意しなければならない。

f. フロー制御 ── 発信地側と目的地側のデータ率が異なるので，特に目的地側の方が発信地側よりもデータ率が小さい場合，ネットワークが輻輳状態に移行しないようにフロー制御が必要となってくる。フロー制御の手順としては，シーケンシング機構（ある番号以上のメッセージは受信しない）とバッファ予約方式（ある個数以上のバッファが空き状態になるまでは，新しいメッセージを流さない）を採用している。

以上の６つの基本的な技術を遂行するためには，次の２つの問題を考慮しなければならない。

- 公平さ ── リソースはすべてのユーザが公平に使用できなければならない。
- デッドロックの防止 ── リソースはデッドロック状態が起らないように割り付けられなければならない。

以下では，各々の設計の考慮点について詳細に検討する。

### 8.4.3 サブネットからメッセージ処理の削除

サブネット中でメッセージ処理を行った方がよいのか検討する。ARPAネットワークでは，メッセージのシーケンシング，メッセージの伝送エラーの検出及び再送はサブネットの分担として設計してある。特に，これらの機能をサブネットよりもＨＯＳＴの分担にした方がよいのかが問題となる。ARPAネットワークの設計者達は，これらのメッセージ処理はサブネットの分担でもよいと考えている。

通信サブネットワークからメッセージ処理を削除することによって，以下に示す設計原理となることが，V. Cerf 氏等によって指摘されている。

a. 信頼性の面から，ＩＭＰが行っていたこの仕事をHOSTが分担しなければならない。

b. IMPがこの仕事を行うと，HOST間のパフォーマンスが低下するかもしれない。

c. IMPがこの仕事を行うと，ネットワークの相互接続がやりにくくなる。

d. サブネットでこの仕事を行うと，ロックアップが起りうる。

e. HOSTが分担するとIMPは簡単になり，この仕事を行わないのでバッファ容量が増える。

最後の指摘は正しいが，しかしその他の指摘は問題がある。ARPAネットワークの設計では，ロックアップが起らないようにしている。その結果としてHOSTのスループットを抑えているが，このスループット値は，ネットワークの回線とHOSTインタフェースのビット率よりも大きい。ＩＭＰに複数HOSTが接続されており，それらが同じ目的地IMPに接続されているHOSTと交信すると，パイプ管理*によりお互に干渉を受けることは事実である。これらの２点について以下で詳細に検討する。

もし，通信サブネットがメッセージ紛失の検出，メッセージのオーダリングをやらないで，かつHOSTにもこれらの対策が考慮されていなければ，HOSTはお互にデッドロック状態に陥る。何故このような状態に陥ってしまうかを，以下に示す。

- 受信側のデータ処理速度が遅く，有限のバッファ量のため，送信側が受信能力以上のデータを送り込むとオーバフローが起るので，フロー制御によってそれを抑止する。
- すべてのメッセージは受信HOSTへ行く前に必ず目的地ＩＭＰを通過する。
- 大量のバッファ量があっても，ＩＭＰバッファは時々オーバフローをする。
- HOSTがＩＭＰバッファを正確に割り当てることは本質的に不可能である。
- 目的地ＩＭＰのバッファがオーバフローした場合，発信側では再送を行わなければならず，

---

＊： 1975年１月にフロー制御機構が変更され，現在はHOST対間でパイプの大きさを８に変更している。

パフォーマンスは極度に低下する。

一般に，たくさんのHOSTを持つ大規模のネットワークでは，もし通信サブネットワークがメッセージを極度に紛失すると，HOSTの大部分は通信サブネットワークが使用に耐えないと見なすであろう。すべてのチャネルに雑音があるので，ＩＭＰは雑音を最小化しなければならない。それ故に，すべてのHOSTでこの対策を取るに必要な経費に比べ，IMPで行う方がずうっと安上りである。IMPレベルで出来ることはHOSTレベルで行うべきでない。

もし，これらの機能をHOSTレベルに移行して行ってもそれほど効率は増加せず，デッドロックの問題も解決しない。デッドロック問題は現実に起るので，HOSTにはより大量のメモリがあるので起る頻度が少なくなったにすぎない。

現在のARPAネットワークの設計は，音声伝送のような一定の低遅延を要求するような分野での使用には適してないが，この種の利用に耐えられるように，現存の機構と平行して新しい機構を追加することができる。

もし，これらの機能の大部分をIMPからHOSTへ移行した場合，HOSTの大部分はこれらの機能を実行するmini-HOSTをIMPとHOSTの間に設置するであろう。事実，現在の傾向として，コンピュータのコストが低下しており，大型のコンピュータからこれらの機能を除去し，周辺プロセッサに移行している。しかし，もしＩＭＰが現在行っているメッセージ処理を周辺プロセッサが行う場合には，順序不同で到着してくるパケットをバッファするため大きな記憶装置を必要とすることに注意しなければならない。この種のインプリメンテーションを各自が独立に行った場合，初期の開発コストがかさむばかりでなく，テストや改定が非常に困難になるであろう。

以下，メッセージ処理をネットワーク中のどこでやるかに関して，ARPAネットワークの設計者の考え方を要約する。

a. 機能や制御を階層構造で組み立てることは，ネットワークのような複雑なシステムでは常套手段である。
   - 効率の面から，IMPはサブネット・リソースを制御し，そしてHOSTはHOSTリソースを制御すべきである。
   - 信頼性の面から，ＩＭＰのプログラムがサブネット全体を効果的に制御しなければならない。
   - 保守性の面から，基本的なメッセージ処理はＩＭＰソフトウェアの分担であり，そうすることによってその処理をHOSTプログラムに分担させた場合に比べ，ずっと変更が簡単になる。
   - デバッギングの面から，手続きは階層構造にして作成するのが本質である。もしそうしないと，バグがあった場合，HOSTプログラムを含めたすべてのプログラムを検査しなければならない。

b. もし，メッセージ処理をサブネットではなくHOSTが行うことにすると，改定を各HOST

が拒否した場合，改定を行うことができなくなってくる。

c. HOSTの分担にした場合，輻輳，HOST間の相互干渉，準最適なパフォーマンスというネットワーク特有の問題を軽減しないし，事実，HOSTはIMPリソースであるバッファ，CPU処理能力（CPU bandwidth），および回線の帯域幅等を制御することができないので，もっと悪くなってしまう。

d. メッセージ処理をIMP内で行う方が費用が安いし，IMPの分担としても決して悪くはならない。

Cerfの"An Assessement of ARPANET Protocols"の論文では次の点を指摘している。

a. 多重パケット・メッセージをスループットをあまり低下させずに削除することができる。

b. 目的地IMPでオーダリングを行うとロックアップを起す要因となるので，もし，サブネットの障害によりオーダリングが必要ならば，HOSTがやるべきであり，IMPはやる必要がない。

c. もし，目的地IMPがオーダリングやリアセンブリを行わなければ，任意の長さのメッセージを送ることが可能である。

d. 再送，エンド・ツー・エンドの確認，エラー検出，そしてメッセージのオーダリングはHOSTレベルで行うべきであり，IMPサブネットでは行う必要がない。

以下では，ARPAネットワーク設計者達の反論を上のと対応させて紹介する。

a. 多重パケット・メッセージを削除しても高スループットを持続できるという点に関しては，

1) 多重パケット・メッセージの使用は有限のバッファ量に制約を与える原因となるという指摘があるが，この制約は実際にはbit coding limitであり，簡単に解除することができる。

2) HOST同士はリンク毎に一時には1個のメッセージしか送出しないという規則を守ってネットワークを有効に使っているので，多重パケット・メッセージを削除することによる損得は得であるという結論に関しては，事実，HOSTはネットワーク使用を改善することができるが，しかし，ネットワークは既に高スループットを実現しているので，多重パケット・メッセージを削除することによるネットワーク・パフォーマンスの低下を償うに充分な改善ができるのかが疑問である。

3) 多重パケット・メッセージを削除してもほんのすこししかパフォーマンスが低下しないという点に関しては，事実，サブネットで多重パケット・メッセージがない場合でも，パケットを一まとめにするという機能を実行しないと，パフォーマンスの低下が著しくなる。

b. サブネットでオーダリングを行うとロックアップを起すという点に関しては，

1) IMPのリオーダリング機構の基本的な考え方は，目的地に必要なバッファを確保してからメッセージを送るので，IMPがout-of-orderのメッセージによってロックアップを起すという指摘は誤っている。

2) HOSTレベルでリオーダリングを行うとロックアップに陥いりやすい。一番外側のレベル

でロックアップを防止するためパケットを棄却することを許したならば，より内部のレベルでもそれ以上に棄却するので，同じように非効率となってしまう。

3) 通常の加入者はメッセージやパケットを発信した順序で目的地に運ぶことを希望し，それ故に，経済性やインプリメントの点からこの仕事はサブネットで当然行わなければならない。

c. 任意の長いメッセージを伝送することができないという宿命的な欠点に関しても Cerf の指摘は充分ではない。ARPAネットワークでも任意の長いデータ・ストリュームを有限の長さのメッセージに区切って送れば可能である。

d. サブネットで行っている機能と同じ機能をHOSTでも行っているという指摘は，HOSTだけでやれば効率的であるということを意味しない。実際問題として，サブネットは自分自身をローカルな輻輳や入出力のデータ率の違いというようなことから防御しなければならない。

いずれにしろ，これらの機能分担の問題は，HOSTコンピュータの負荷の軽減と，データ交換網の処理効率，信頼性などの質とのトレードオフの問題である。

### 8.4.4 単一パケット・メッセージのみ

よく次の質問を尋ねられる。

「もし，パケット・サイズを大きくし，一方メッセージ・サイズはパケット・サイズと同じになるまで小さくしたならば，リアセンブリ問題やフロー制御問題がなくなるのではないか？」

その答は，「フロー制御やリアセンブリはメッセージ・サイズやパケット・サイズにはほとんど関係していない」である。以下に次の4点について検討する。

パケット・サイズ： ネットワーク遅延は，回線速度，伝播遅延，回線利用率，使用される優先権機構，IMPの処理時間，そしてリソースの予約確率などの複雑に組み合わさった関数である。現在のARPAネットワークでは，優先権を持つ短いメッセージに対する遅延は，ネットワーク中の他のトラフィックのパケット・サイズに正比例する。このため，短いパケットが望ましい。回線が長距離であったり，あるいは高速である場合，その結果として伝播遅延の方が伝送遅延の方より大きくなった場合には，この考え方は変えなければならない。

メッセージ・サイズ： メッセージに対するオーバヘッドの割合を減少させるためには，メッセージ・サイズを大きくする必要がある。メッセージ・サイズが短いと，それだけ多くHOSTはメッセージ割り込みを処理しなければならないので，非効率である。しかし，現在のネットワークでは，IMP中のバッファを使用してリアセンブリし，ネットワーク遅延を抑えるためにも，メッセージ・サイズには上限が存在する。

リアセンブリ： 回線の伝播遅延が大きい場合，その回線を数個のチャネルにマルチプレクシングし，回線をフル効率に使用するため，同時にデータ・ピースを数個流す必要がある。このデータ・ピースがリアセンブリを必要とするパケットか，順番に発送しなければいけないメッセージに対

応するかによってほんのすこし異ってくる。衛星リンクを使用しないARPAネットワークでは，発信地IMPと目的地IMP間にパイプを設定し，回線をフルに使用するため，同時に送出できるデータ量を30パケット分位にしている。衛星リンクの場合は，もっと多くのパケットが必要である。もし，発信地 ── 目的地間のパイプの使用率を3％（100％／30)で我慢することを望まなければ，IMP，あるいはIMPが行わなければHOSTでリアセンブリを行わなければならない。

フロー制御： 目的地が発信地よりもデータ率が小さい場合，パイプ中のトラフィックを抑止するためにフロー制御が必要である。ネットワーク中にはたくさんのパイプがあるので，フロー制御は複雑になる。ARPAネットワークでは，フロー制御は処理を簡単にするため同じ大きさのメッセージが流れるとしてスペースの管理を行っている。もちろん，このフロー制御の単位であると想定したメッセージ・サイズと実際に伝送されるメッセージ・サイズとは必ずしも一致しない。パケット・サイズはARPAネットワークでインプリメントされたフロー制御にほんのすこし関係する。

### 8.4.5 パケット・サイズ

ここでは，パケット交換ネットワークにおけるパケットの最適なサイズについて検討する。

パケット・サイズが長ければ長いほどエラーのある電話回線で伝送する場合伝送エラーの確率が大きくなり，このことはパケット・サイズをできるだけ小さくすべきことを指摘している。一方，パケット・サイズが小さければ小さいほどオーバヘッドのパーセンテージが増加することになり，このことはパケット・サイズをできるだけ大きくすべきことを指摘している。ARPAネットワークの回線エラーの特性から実効スループット（オーバヘッドがあまり大きくなく，エラーによる再送の確率を小さくする）を最適にするパケット・サイズを計算することができる。計算によると，ARPAネットワークが採用している約1000ビットのパケット・サイズは最適曲線の中に含まれており，これは準最適である。

もう一点は，蓄積転送遅延（store-and-forward delay）は，パケット・サイズが小さければ小さいほど減少する。即ち，パケット・サイズが減少すると，パイプライン効果により遅延も減少する。しかし，パケット・サイズが減少すると，オーバヘッドが増加するため，実効スループットはまた減少する。この場合，遅延と実効スループットはトレードオフ（trade-off）の関係であり，最適な値を決定することは不可能である。

DaviesとBarberは，バンキング・システムやエアライン・システムで交換されるメッセージの大部分は約2000ビットであり，このサイズが1パケットにちょうど入るようにし，パケット・サイズをすくなくとも2000ビットにすべきであることを主張している。しかし，これは，彼達がARPAネットワーク用語でメッセージ，即ちHOSTとIMP間で授受される情報の基本単位をパケットという用語を使っているので，ARPAネットワークのパケットが小さすぎるということを意味しない。事実，ARPAネットワークのメッセージはこの提案された最小のサイズをかなり越えてい

ARPAネットワークの最近の測定によると，ARPAネットワークのパケット・サイズは準最適であり，大部分のメッセージは約250ビットである。この値は，ネットワーク中で観測されたトラフィック・ミックスからIMPのバッファの使用効率を最適にするように決められた。

しかし，これには2つの重要な制約条件が無視されている。第1番目に，IMPのバッファ・ストレージと50Kbの回線の相対コストはIMPのバッファ・ストレージを最適にすべきでないことを指摘している。パケット・サイズを決定する真のトレードオフは電話回線の伝送帯域の効率的使用（パケット・サイズをより大きくする）と遅延特性（パケット・サイズをより小さくする）である。もし，バッファ・ストレージが制約条件ならば，増設すればよい。

第2番目は，IMPが電話回線を操作するに充分なバッファを備えている場合，概してこのバッファ・スペースはあまり使用されてない状態となる。事実，必要なバッファ・ストレージと使用される伝送帯域とは直接比例関係にあるので，例えばファイル転送のような高帯域伝送を使用するようなケース以外では，バッファはほとんど使用されない状態となる。しかし，このケースでは，ユーザは多重パケット・メッセージを送ることが前提となっている。要約すると，IMPバッファはファイル転送を行うために用意されているが，しかし，現在は短い会話的なメッセージがネット内をたくさん流れている。

非常にたくさんのインタラクティブ・ユーザがこのネットを同時に使用すれば，この短いパケットでIMPが飽和してしまうことが起りうる。もしそうならば，回線上のオーバヘッドを減少するため，短いメッセージをより大きな伝送単位にまとめて送り，目的地で分解すべきである。

以上のように，パケット・サイズの選択はたくさんの要因によって影響を受ける。要因の中のいくつかは本質的に相反し，最適なパケット・サイズを決定するのは困難であるので，現在のARPAネットワークのパケット・サイズは適切な妥協点である。他のパケット・サイズ（例えば2000ビット）が実際には使用されるかもしれないが，現在使用されている1000ビットのパケット・サイズよりもはっきりとすぐれているということは言えないことは確かである。

### 8.4.6 おわりに

今，パケット交換ネットワークの設計に関してかなりの経験が得られ，基本的な設計項目がわかってきた。一方，実用レベルのパケット交換ネットワークはまだ萌芽期であり，今後更に経験しなければならぬたくさんの課題がある。新しい課題としては，例えば，(a)パケット交換技術を実験的な場から大規模なコマーシャル運用の場への移行に関する技術，(b)新しく使用できるようになった衛星通信リンクをパケット交換ネットワークへ応用する技術，(c)パケット交換ネットワークを通しての音声伝送，(d)無線によるパケット伝送，(e)パケット交換ネットワークの相互接続，そして(f)パケット交換ネットワークのためのHOSTのオペレーティング・システムの設計などである。

## 8.5 コンピュータ・ネットワークの開発方法

—— CYCLADESを中心として ——

### 8.5.1 CYCLADESの概要

A. 歴史と目的

フランスにおいては，1970年以来CRI（Comité consultatif pour la Recherche en Informatique：情報処理の研究のための諮問委員会）で行われてきた研究によって，コンピュータ・ネットワークの重要性が立証された。

これをうけて，1972年の初頭より，DI（Délégation a I'nformatique：情報代表部）の指導のもとに開始されたのがCYCLADES網（Reseau CYCLADES）である。

CYCLADESプロジェクトは，軍，大蔵省，文部省，情報代表部，IRIA，P＆T（郵政省）より構成される理事会により監督されている。このメンバーの中で，情報代表部がプロジェクトの予算（1972年－1975年）である20.8百万フランの80％を用意し，軍が20％を負担する。P＆Tは，電話回線およびモデムを無料で提供する。IRIAが，このプロジェクトの実行機関としての役割をはたす。

CYCLADESの目的は，種々の新らしい分野での実験の道具となるプロトタイプのコンピュータ・ネットワークになることである。特に，つぎにあげる6つの分野が考えられている。

(1) データ・コミュニケーション
(2) コンピュータの相互会話
(3) 協同ジョブの研究
(4) 分散形データ・ベース
(5) インターフェースおよび共通言語
(6) リソースの共有

これらの目的が，次のような段階をへて実現されていった。

73.9：3台のホストと1台のノードよりなるネットワークで3つの機能をデモンストレーション。

- オペレータ間通信
- ファイル転送
- リモート・ジョブ・エントリ

74.2：4台のホストと3ノードからなるパケット交換網（CIGALE）が出来上った。CIGALEは，1日3時間ずつ稼動した。

74.6：CIGALEが7ノードとなった。

74．8： European Informatics Network（EIN）の一環として，CYCLADESとNPLネットワークが結合された。

この段階では，CYCLADES側からNPLの機能の利用は可能になったが，この逆はまだできていない。

74.11： カナダのトロントで，ＴＳＳ，リモートバッチ，ファイル転送，ローカル・ターミナルアクセスのデモンストレーションをおこなった。

75．1： CIGALEが，１日６時間稼動になった。

75．2： 予算の関係で，ノードのうち４台をターミナル・コンセントレータとして使用できるようにして，コンセントレータを新規に設置しないことにした。

IRIAにネットワーク・コントロール・センター設置（Mitra－15を使用）。

P＆Tリサーチセンター（Rennes）にネットワーク・メジャメント・センターを設置（Mitra－15を使用）。

75．7： この時点で使用可能な機能は次のとうりとなった。

CII，IRIS－80およびIBM 360／67のTSS，リモート・バッチ，ファイル転送，ローカル・ターミナル・アクセスが可能。

CII 10070のリモート・バッチ，ファイル転送，ローカル・ターミナル・アクセスが可能。

B．参加団体および規模

CYCLADESは，20の研究機関の参加を予定して作られた。このうち13機関は，ネットワーク利用の研究を主眼とする研究機関および大学であり，残りの７機関は，ネットワーク利用が軌道に乗ってから以後開発を進めて行く行政局の管理機関である。

参加団体およびホスト・コンピュータの現状については，新らたな参加団体，参加をとりやめた団体，ホストのリプレースをおこなったセンタがあり，正確にはつかめない。主なる参加団体は，表８－４に示すとうりである。

C．CYCLADESのもたらすもの

CYCLADESの開発にあたっては，官公庁，郵政省，軍，情報処理産業など，各種の分野にわたっての利益および期待がまとめられたので，これについて述べてみる。

行政機構

国の経済活動において，行政機構が負わされているサービスは数多く，このことから由来する負担は，はなはだしいものがあり，増加の一途をたどっている。行政機構は，大量に情報を収集し，処理し，普及せしめるという重要な任務を負っている。行政機構は，それの所有している情報を共有化することによって生ずる利益について認識を新たにしてきた。

いくつかの計画がすでに試みられている。人間の身体（SAFARI），又は道徳（SIRENE）に

表8－4　参加センター

- Institut de Recherche d'Informatique et d'Automatique (IRIA) 2 center
- Compagnie Internationale pour l'Informatique (CII)
- Météorologie Nationale (METEO)
- Institut de Recherche des Transports (IRT)
- Compagnie Internationale pour l'Informatique Centre Scientifique de Grenoble (CII)
- Université de Grenoble (IMAG)
- Centre Universitaire de Calcul de Lyon (CCILS)
- Ecole des Mines de St-Etienne (MINES)
- Centre d'Etudes et Recherches de Toulouse (CERT)
- Université de Toulouse (TOU)
- Université de Rennes (REN)
- Centre National d'Etudes des Télécommunications (CNET)
- Centre Commun d'Etudes de Télécommunications et Télévision (CCETT)
- Centre Electronique de l'Armement (CELAR)
- Ecole Supérieure d'Electricité (ESE)

関する共通目録の作成，諸々の活動や生産物のためのカタログ，中央官公庁（税関，国税局，厚生省等）により計画された情報電送システムの設置などである。

CYCLADESは，これら一連の活動のうちの一つであり，次にあげるようなサービスの提供を行っている。

―― データの交換

いくつかの方式が研究され，次いで提案されることになっている。たとえば，報告書，コピー等の転送，文献検索の質疑及び文献抽出，他のネットワークに蓄えられた索引プログラムへの直接のアクセスなどである。

―― プログラムの交換

行政サービスに関するいくつかのプログラムは共有することができる。それは，次の様な場合である。

- ネットワークを通じて，プログラムの普及と保守が必要な場合。その各サービスは適切な計算機の下で行われることになる。
- 遠隔地から，ネットワークを通してデータを送り処理結果を受けとるような処理が必要な時。

――リソースの共有

この領域こそ，CYCLADESが，行政の要請にもっともよく答え得る方式の選択ができる場であり，もっともよく柔軟性を示す場でもある。これらリソースのうちわけは次の通りである。

- コンピュータ
- 集中管理装置と回線
- 回線

結論として，CYCLADESプロジェクトは，行政に関して次のような利点を持っている。

- 短期的に見れば，自由な使用に供することによってデータ転送の道具となり得る。
- より長期的には，特定の外形の研究に関した行政に関する，一個又は数個の情報ネットワークの概念から一つの方法論を引き出すことができる。これは，コンピュータ・ネットワーク（ことに，データ又はプログラムの信頼度に関する問題点を整理した上で標準化されたプロセデュアの創造のことであるが）の共同開発によって生じた問題点の研究と，一連のヴァリアントを持った実験によって可能になる。

CYCLADESの設置と並行して，共同作業グループである，システム・ダンフォルマシオン（Systémes d'Information）が，異なった研究センター間のデータ・ベースへのアクセス，及び，情報の流動性の持つ問題点の研究を進めて行く。設置チームと一致協力しながら，その作業によって，ネットワークに関連した適応の諸問題を予測し，解決して行くことができる。

PetT

PetTの行政機構は，CYCLADES網の受益者であるといえる。それは，PetTが，連結されているデータ・バンクへのアクセスの可能性の恩恵を蒙っているからであり，又，PetTはデータ伝送とデータ交信に関しては，パイロット的役割りを果しているからである。

軍　隊

CYCLADESの実施によって得られる手腕や技術は必ずや軍関係のネットワークの概念に貢献することであろう。軍本部は，それ故大いにプロジェクトに興味を示すであろう。

情報処理産業

情報処理産業にとって，CYCLADESは，将来のシステムのハードウェアとソフトウェアの特性を決定するためには最良の場であると言える。

CYCLADESは，コンピュータ・ネットワークの一つの系統の原型であるといえる。その系統の特徴は，並列処理，コミュニケーション，専用化，信頼度などによって示されることになるであろう。それ故，新しいアーキテクチャ，新しい使用モード，効率の高い伝送用の機材及び技術が考案されねばならぬであろう。同時に，現在既に不足しており，その需要が将来急増すると考えられる規格品の開発もされねばならぬであろう。

コンピュータの異種性のために，CYCLADESは，互換性の無さという問題点の分析，又，そ

れを是正する手段を必要とするようになるであろう。これらの作業は，同一でなければ少なくとも相互に，機械的に翻訳可能なコンピュータの構造の問題に行きつくであろう。

ネットワークの設置に積極的に参加しているC．I．I．にとってCYCLADESは，長期的な目で見れば十分利益となるものであろう。

1972年から1975年の間に，C．I．I．の勢力の及ぶ市場で，コンピュータMitra15の引渡し台数が制限されるのが事実だとしても，このようなシステムの概念の出現によって，Mitra 15のカタログを改良し，補充する（ことに結合装置の性能の改良であるが）こともできるであろうし，また，機材の特権的状態によって，買い入れ側にしてみれば，開発を強く要求されている同種的下位ネットワークを構成することができるようになるであろう。

≪分散されたインテリジェンス≫の概念は，新しいシステムのアーキテクチャの基礎がためにもなる。このようにして，ソクラテス・プロジェクト（分散データ・バンク）は，CYCLADESに関して自由使用可能な最初のソフトウェアの一つを生みだすことになるだろう。

建造者たちがコンピュータ・ネットワークの開発に専心している時に，CYCLADESは，パイロット的な実験を行い，その成功は，その産物とC．I．I．の商標のイメージ，この両者の頭上に輝しい光を投げかけるであろう。

サービス企業と情報処理会議

これらの企業は，将来における，コンピュータ・ネットワークの存在と直接のかかわりがある。しかも，CYCLADESの実現に一役買っているのであり，これらの企業はCYCLADESの実験と能力を大いに評価することになるであろう。というのも，これら企業は，オンライン処理システムを持つことを，コミュニケーションに関するソフトウェアの実現を，汎用あるいはアプリケーション・ソフトウェアの普及等を要求されているからである。

フランスの主要な企業の大部分は，CYCLADESの実現に参加するのに必要なあらゆる能力を持ったハイレベルの人材を備えている。フランスの市場に触手を延ばそうと待ちかまえているアメリカの企業によって，これら企業が脅かされないためには，ぜひともこの能力を開発する方向にむかわなければならない。

大　学

短期的観点よりすれば，CYCLADES網の実現に伴う主要な利益の一つは，大学のパイロット的研究センタの参加にあると言える。数多くの学生達が，ネットワーク建設に関する諸問題について，また，機材や実際に働いているソフトウェアの特性について，そこから様々な研究テーマを見い出すことができるからである。また，同時に，彼らは直接役に立つ専門的な経験をも重ねて行くのである。実験段階に入ると同時に，CYCLADESは，大学の研究センタあるいは研究所のコンピュータを結合することができるであろう。各研究センタは，このようにして，システムの非常に拡張された全範囲を又ランゲッジや，データ・ベースを利用できることになる。研究者

作業，学生の養成は，彼らが適当な道具を自由に使えるだけに益々，効果的なものになるであろう。

研　究

今述べたごとく，コンピュータ・ネットワークの主要な刷新面は，その利用，ことに研究チーム自身の為の利用にあるといえる。

ところが，研究をして行く上で，様々な障害につきあたることがしばしばである。例えば，地理的条件，学閥等々の障害である。これに反して，CYCLADESは，共通の目的にそって作業を行いながら，様々なチーム間で，情報を交換したり，対話をかわしたりすることができるのである。ネットワークが機能を開始すれば，ネットワークを通して，これらチームの構成員達は共に作業をしつづけることができるのである。

さらに言えば，情報処理関係の人々のみが関係者ではないのであって，コンピュータの使用者でありさえすれば，すべての学問の研究者達も，他の研究センタの同僚と情報を容易に交換できるという恩恵に浴することができるのである。

D．開発内容

CYCLADESにおいて当初企画された作業のリストを以下に示す。

これらの作業は，73／74年に計画されていたものであり，夫々73年初頭より開始された。

① Abonné Transiris-CYCLADES（CII-Grenoble）

期限：1973年12月

CYCLADESネットワークの利用者に対して，Transirisサービス（タイム・シェアリング，リモート・バッチ，……）へのアクセスを可能にするソフトウェア。次のプロトコルに関係する。

— ST／ST

— ユーザ／サーバ・コネクション

— バーチャル・デバイス

② Siris 7／8の下でのターミナル・コンセントレータ（CEA）

期限：1974年3月

ローカル端末が，ネットワークによって他のサービスを受けることを可能にするソフトウェアCTの実現。

③ R-ファイル　Siris-7（CII-Grenoble）

期限：1973年12月

リモート・ファイルに対するVDAMアクセス法（ダイレクト・アクセス）の拡張。ネットワーク上に，ファイルが分散され得るSOCRATEシステムの直接の応用。

④ IMAGのIBM 360／67のCPサービスへのアクセス（IMAG-Grenoble）

期限 ： 1974年4月

リモート端末からCP／67の利用を可能にするようなインターフェース・ソフトウェアの実現。次のプロトコルに関係する。

── ＳＴ／ＳＴ

── ユーザ／サーバ・コネクション

── バーチャル・デバイス

⑤ ＩＢＭ 360／67のＯＳ／ＭＶＴのリモート・バッチ・サービスに対するアクセス（ IMAG ）

期限 ： 1974年5月

リモート端末からＩＢＭ 360／67においてジョブを実行できるようにするインターフェース・ソフトウェアの実現。

⑥ CPの下でのターミナル・コンセントレータ（ IMAG－Grenoble ）

期限 ： 1974年2月

ローカル・ＣＰ端末がネットワークを通じて他のサービスを受けることができる。

⑦ ＯＳ／ＭＶＴの下でのターミナル・コンセントレータ（ IMAG ）

期限 ： 1974年3月

ローカル端末が，バッチ・セッションの間に，ネットワークを通じて他のサービスを受けることができる。

⑧ タイム・シェアリングＳＡＭへのアクセス（ CERT－Toulouse ）

期限 ： 1973年12月

CERTのＣＩＩ 10070で実現されるソフトウェアで，リモート・ユーザがＳＡＭシステムへのアクセスを可能にする。次のプロトコルに関係する。

── ＳＴ／ＳＴ

── ユーザ／サーバ・コネクション

── バーチャル・デバイス

⑨ ＳＴに対するリスタートの問題の研究（ IMAG ）

期限 ： 1974年10月

いろいろな場合の故障を調べ，ＳＴがリスタートできる水準を明確にすることを可能にする。

⑩ CYCLADESのドキュメンテーション・システムの設置（ CCILS－Lyon ）

期限 ： 1974年10月あるいは1975年

完全なドキュメンテーションに対して関係者が参照できるように，ネットワーク上に4ケ所，論文に関する4つのストックを設置する。

⑪ ネットワーク・インタープリータの研究および開発（ FACULTE－Toulouse ）

期限 ： 1974年9月

インタープリータの形で定義された1つの"ネットワーク・マシン"が実際のマシン上にすべて実現される。そして，実験の範囲内で利用するもの，オペレーティング・システムおよび利用手続きの異質性による諸問題を克服するためのものである。

⑫ ネットワーク上で分散されたプログラムの実行（CERT）

期限 ： 1974年1月

いろいろなサイトに分散されたプログラムのアクティベイト，同期の問題の研究。

⑬ STの測定（FACULTE-Rennes）

期限 ： 1974年6月

STの機能の評価を可能にする"スパイ・マシン"の設計と実現。"スパイ・マシン"による測定。測定結果の利用。

⑭ シミュレーション（FACULTE-Rennes）

期限 ： 1974年6月

ST／10070のシミュレーション。"動作中のネットワーク"レベルでのセンタのシミュレーション。CIGALEのシミュレーション。（最初のアプローチ）

⑮ ファイル転送のプロトコル（ESE-Rennes）

期限 ： 1974年12月

いろいろなサイトに共通な，ファイルの転送，蓄積および交換手続きの設計。

⑯ CYCLADESのターミナル・コンセントレータ

期限 ： 1974年6月

Mitra-15の下で，ST／STおよびPAVプロトコルを利用するターミナル・コンセントレータの仕様決定と実現。これらのコンセントレータは，Siris 7／8のRB，TSサービスならびにCP，OS／MVT，SAMのサービスにアクセスすることができる。

⑰ ネットワーク言語

ネットワーク言語の研究のために，SOCネットワークと協力する。

⑱ SOCRATE（CII-Grenoble）

異機種のSOCRATEに関する問題の一般的解決の研究。

### 8.5.2 CYCLADESの開発コスト分析

A. プロジェクトの見積り

研究プロジェクトにおいて，精度の高い費用の予測は一般に困難である。この理由は，研究の課程において新らたに発生する出費があるからである。しかしながら，コンピュータ・ネットワークにおいては，すべての技術が全く新らしいものではない。それらのうちの多くは，すでに試みられ，かなり成功したものである。

CYCLADESが，プロジェクトの開発コストのチェックのために利用したのはARPAネットワークである。これは，ARPAネットワークが，CYCLADESより以前に実施された唯一の同様なシステムであったからである。もちろん，ARPAネットワークの数字を生のまま利用したわけではなく，目的，作業環境，制約などの違いが考慮された。

ネットワークの構成のために必要なコストについての考察は，1971年の初頭に，一般的な仮定をおいて実行可能かどうかということでおこなわれた。これは，費用見積りのための確かな根拠となった。

プロジェクトの主要な仕事が再評価され計画されたのち，よりつっこんだ分析が1972年の初頭におこなわれた。この中で実行すべきものは，大まかに言えば，次のように分割された。

ⓐ 調整

ⓑ ホスト・インターフェースと交換網の一般的な仕様

ⓒ プロジェクト・マネージメントのためのソフトウェア・トゥール

ⓓ 理論的な側面の一般的な研究

ⓔ パケット交換網のソフトウェア

ⓕ ターミナル・コンセントレータのソフトウェア

ⓖ パケット交換網とターミナル・コンセントレータのためのハードウェア

ⓗ ホスト・インターフェースのソフトウェア

ⓘ リモート・ジョブ・エントリ，ファイル転送，コマンド言語，会話形アクセス，データ・ベース・ハンドリングのようなソフトウェア・サポート

1つのものを，4つの機種において作成するものとされた。

ⓚ データ・ベースを含むアプリケーションの研究と委託。1つにつき，6つの機種において作成するものとされた。

ⓛ ネットワークに関する研究

全体のプロジェクトは，1972年から75年までの4年間の計画であった。それぞれの年度には，それぞれの主題があった。

— 72年 ： 設計

— 73年 ： インプリメンテーション

— 74年 ： テスト

— 75年 ： アプリケーション

仮定したネットワークの規模は，20台のホスト，6台のパケット交換ノード，7台のターミナル・コンセントレータであった。

人件費や機械使用時間の単位時間当りのコストは，正確に決めることができない。この理由は，プロジェクトの種類，使用コンピュータ，期間がそれぞれ異なっているからである。平均のレー

| | 工数（人月） | | | | | 機械時間（時間） | | | | | 賃貸／購入ハードウェア（千フラン） | | | | | ハードウェアの研究（千フラン） | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 72 | 73 | 74 | 75 | 合計 | 72 | 73 | 74 | 75 | 合計 | 72 | 73 | 74 | 75 | 合計 | 72 | 73 | 74 | 75 | 合計 |
| a 調整 | 12 | 12 | 12 | 12 | 48 | 24 | 24 | 24 | 24 | 96 | | | | | | | | | | |
| b 一般的な仕様 | 36 | 12 | 0 | 0 | 48 | 36 | 12 | 0 | 0 | 48 | | | | | | | | | | |
| c ソフトウェア・トゥール | 9 | 12 | 12 | 0 | 33 | 45 | 60 | 60 | 0 | 165 | | | | | | | | | | |
| d 一般的な研究 | 16 | 48 | 48 | 48 | 160 | 80 | 240 | 240 | 240 | 800 | | | | | | | | | | |
| 合計 | 73 | 84 | 72 | 60 | 289 | 185 | 336 | 324 | 264 | 1,109 | | | | | | | | | | |
| e パケット交換網ソフトウェア | 25 | 33 | 12 | 12 | 82 | 125 | 165 | 60 | 60 | 410 | 100 | 100 | 100 | 100 | 400 | 0 | 0 | 0 | 0 | 0 |
| f ターミナル・コンセントレータ・ソフトウェア | 10 | 46 | 14 | 12 | 82 | 50 | 230 | 70 | 60 | 410 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| g 交換網＋コンセントレータ | 18 | 22 | 13 | 12 | 65 | 0 | 0 | 0 | 0 | 0 | 0 | 800 | 1,600 | 2,600 | 5,000 | 800 | 0 | 0 | 0 | 800 |
| 合計 | 53 | 101 | 39 | 36 | 229 | 175 | 395 | 130 | 120 | 820 | 100 | 900 | 1,700 | 2,700 | 5,400 | 800 | 0 | 0 | 0 | 800 |
| h ホスト・インタフェース・ソフトウェア | 10 | 160 | 100 | 100 | 370 | | | | | 1,850 | 200 | 200 | 200 | 200 | 800 | 100 | 100 | 0 | 0 | 200 |
| j ソフトウェア・サポート | 48 | 144 | 64 | 48 | 304 | 240 | 720 | 320 | 240 | 1,520 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| k アプリケーション | 0 | 144 | 193 | 156 | 493 | 0 | 720 | 1,025 | 780 | 2,525 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 合計 | 58 | 443 | 357 | 304 | 1,167 | 240 | 1,440 | 1,345 | 1,020 | 5,895 | 200 | 200 | 200 | 200 | 800 | 100 | 100 | 0 | 0 | 200 |
| 総合計 | 184 | 633 | 468 | 400 | 1,685 | | | | | 7,824 | 300 | 1,100 | 1,900 | 2,900 | 6,200 | 900 | 100 | 0 | 0 | 1,000 |

図8－17　コスト分析の詳細

トは次のように決められた。

1人月（man・month）　　：　15KF（約102万円）

1機械時間（Computer hour）：　2KF（約13.6万円）

（KF＝1,000 Francs）

それぞれの年のプロジェクトごとの分析は，図8－17に数字で示されている。回線とモデムの評価は，はずされている。この理由は，フランスP＆Tとの間で75年末までは無料であるということになっていたからである。

種々の費用が，平均のレートによってフランに換算された。そしてデータ通信装置と関係する研究を加えるとネットワークを建設するのに必要な全費用となる。

この金額は57.5百万フラン（約39.1億円）となるが，この過程は，図8－18に示す。

最初から，参加センタは，工数，機械時間，ホストのハードウェア的な結合装置に関して何らかの費用を分担することが仮定されていた。

実際には，結合ハードウェアは小さなものであった。この理由は，CYCLADESでは広く使用されているメーカの伝送手順と，CCITTのインターフェースを採用したからである。

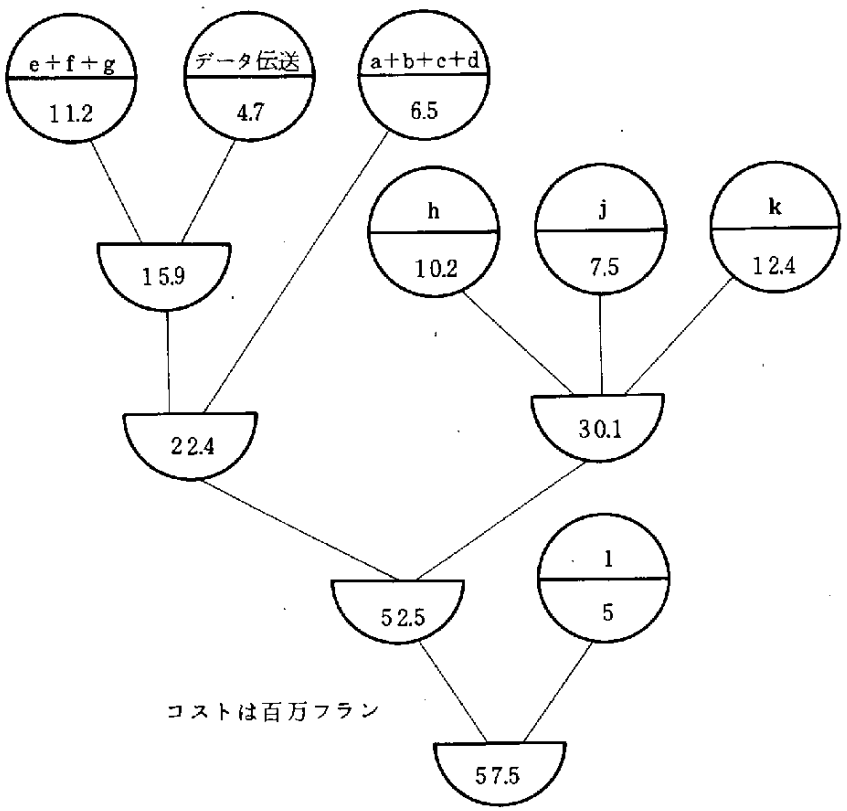



図8－18　　コ　ス　ト　構　成

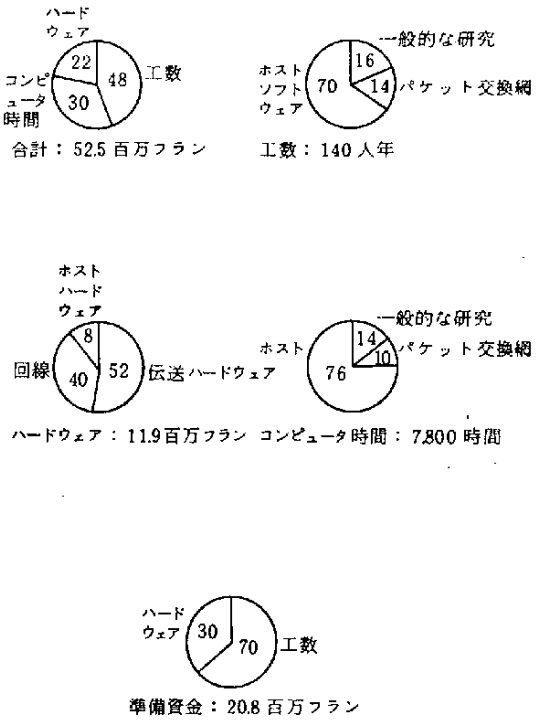



図8－19　相対コスト

参加センタでは，ホストあたり次の費用を用意することが決められた。

| | |
|---|---|
| 工数 | 4人月 |
| 機械時間 | 300時間 |
| ハードウェア | 40千フラン（約272万円） |

この結果として，残り部分は20.8 MF（約14.144億円）となった。そしてこれは，情報代表部と軍の研究部によって用意された。コスト要素に関する用途ごとのグラフは図8－19に示されている。

B. プロジェクトの運用状況

前項で述べた最初の計画は，実行においてずれが生じた。新会員の募集，組織化に関して，いくつかの活動がスケジュールよりも遅れてスタートした。そして72年と73年の出費が予定よりも少なかった。この傾向は，74年には逆になり，75年にも持続されるであろう。

最初のつまづきにもかかわらず，主な目標はスケジュールどうりであった。1974年7月の時点で，予測された予算の約半分が投ぜられたが，全体的な予測が合わないような徴候はない。

いくつかの分野において，このプロジェクトは異なった成長をした。実験であるのとは逆に，いくつかの開発は，しだいに実用化の方向に向いていった。

最初は計画されなかったが，パケット・ターミナル，無線リンクなどの新しい分野が研究された。

CYCLADESは，研究，オペレーション，実用化において，一時点な実験というよりはむしろ異なった方向で継続したと考えられており，新しいプランと適当な資金が用意されるのが期待されている。

### 8.5.3 CYCLADESの成果と今後

A. CYCLADESの利用

CYCLADESの利用に関して，基本的には2つの側面をみることができる。1つはコンピュータ・ネットワークの実験的な役割りであり，もう1つはそれの商用化である。前者をCYCLADES-0，後者をCYCLADES bisと呼んでいる。

CYCLADES-0

初期の計画から75年末までのシステムをこう呼んでいる。

ネットワークの利用における主たる目的は，大学あるいはいくつかの研究所にコンピュータを持っている文部省のために，大学間ネットワークを構築することであった。

それ故に要求されたのは，大型コンピュータでサービスされているメーカが作り上げたソフトウェアを，ネットワークを通してターミナル・コンセントレータからのアクセスも含めて多くのユーザ間で共用できるようにすることである。この種のサービスは75年1月からグルノーブルのコンピュータで開始され，満足のいくものになっている。

実際に，グルノーブルの2台とIRIAの1台のコンピュータがサービス体制にある。他のものは技術的には動いているが実験，デモンストレーション，研究などに使用されている。IRIAのコンピュータは，過負荷状態にあるので，ジョブのうちのあるものは，グルノーブルのホストで実行している。

これに加えて，CYCLADES-0は，ネットワーク言語，新型の端末，新しいオペレーティング・システムなど，ネットワークの研究のための道具として使われている。

このような分野で，75年7月現在のところ進められているプロジェクトは次のとうりである。

- TSSあるいはリモート・バッチで使用できるプロセッサ：APL，SOCRATE（汎用データ・ベース・マネージメント・システム），MISTRAL（ドキュメンテーションとインフォメーションのリトリーバル），テキスト・エディタやコンパイラなどの通常のソフトウェア開発の道具。
- これらを使った，化学関係のデータ・バンク（THERMODATA），回線シミュレータ，コンピュータ・シミュレータ，経済モデルシステムなどのプログラム。
- 社会科学，言語学，教官管理，CAI，血友病に関するデータ・バンクなどのアプリケー

ションを開発中である。

── 軍事関係の実験や，関税や財務に関するパイロット・アプリケーションが準備されつつある。

CYCLADES bis

CYCLADES bisは商用版である。フランスの官庁のうち現在2つの部門が，プライベートCYCLADESを採用した。これは，ソフトウェア会社によって設置されるだろう。

いくつかの会社では，自社デザインの自社システムを作ることを望む。この傾向は，新しい技術がひろまったときにおこる。技術スタッフは，新らたに経験することを望み，自社デザインのシステムだけが要求を満たすことをマネージャーに説得する。ある場合には，そのような投資は正しいであろう。しかし，一般的にいえば，経験あるマネージャーは，開発したところが保証するシステムを好む。

コンピュータ・ネットワークに於ても同様である。

CYCLADESは，異機種ネットワークに対して，よく決められたプロトコルを持った唯一の商用化システムである。現在のシステムは殆んどCIIの製品でサポートされているが，CYCLADESのプロトコルを種々のオペレーティング・システムにおいて商用化して使うことができるようにする研究が進められている。

B．開発状況

CYCLADESプロジェクトは，種々の分野におよんでいる。次にあげるのは，そのうちの主なものである。

分散形データ・ベース

長期的な目標は，いくつかの異なったデータ・ベースを並行的にアクセスできるようにすることである。

第一のステップはSOCRATEを利用することによってインプリメントされた。このデータ・ベース・システムは，CII，SIEMENS，IBMのコンピュータで使うことができる。

いくつかのコンピュータに分散された同機種のデータ・ベースにアクセスできるネットワーク・アクセス法がインプリメントされた。第二のステップは，2つのデータ・ベース・システムを結合することである。

科学技術情報の共有や配布のために相当の努力がされ，この10年間に特別な形のデータ・バンクを作るために相当の投資がされた。これは全世界的なものである。フランスにおける調整機関はBNIST（Bureau National d'Information Scientifique et Technique）とBAB（Bureau d'Automatisation des Bibliothéques）である。これらの機関は，全ヨーロッパの科学情報をプールしようとしている EURONET プロジェクトによってヨーロッパ共同体に含まれている。CYCLADESとSDS-ESROネットワークの相互結合は，この一部である。こ

の分野におけるフランスとカナダの間でおこなわれた協力は，75年に共同プロジェクトとなる。

全世界レベルにおいて，すべての国の科学情報の配布を計画し組織化する目標を持った長期の研究がUNESCOでおこなわれている。

ネットワーク言語

ポータビリティのあるネットワークに共通な言語を作ることの可能性に関する研究プロジェクトがある。この作業は，EIN（Europian Informatic Network）と共同でおこなわれている。

シミュレーションとモデル化

ホスト・プロトコルは，論理性と性能の評価のためにインプリメンテーションに先だってシミュレーションがおこなわれた。またルーティングとフロー制御手法をテストするためにCIGALEのモデルが開発された。後者の場合，カナダのワーテルロー大学，その他のヨーロッパの研究グループとの協力をおこなった。

端末からのアクセス

コンセントレータは，種々の構成を許したり，CIGALEによる管理ができるようにするために修正中である。標準ネットワーク・パケットを直接処理できるプロトタイプの端末とミニ・コンセントレータの研究をおこなっている。無線ターミナルも研究されている。

ネットワークの拡張

何台かのターミナル・コンセントレータが75年と76年に増設される。2ないし3台のノードが必要になってくるであろう。

CYCLADESの開発は，公衆パケット交換網のTRANSPACと緊密に協力している。CIGALEとTRANSPACのデータ・グラムのコンパティビリティは保証されている。

C．動　向

コンピュータ・ネットワークは相互に通信をおこなわなければならないので，標準化が重要である。

プロジェクトの最初の頃から，標準化は考えられてきた。そして，他のネットワーク作成者とあるレベルのコンパティビリティを可能とするための意見の一致を見い出す努力をしてきた。NPLとの相互結合はこの例である。

他の要素は，公衆網とプライベート網の問題である。CIGALEは，CCITT用語でいえば"データ・グラム"と呼ばれるサービスを行っている。それ故に，プライベートなパケット交換網を作っているユーザは，将来の拡張計画に応じて，公衆パケット交換網と入れ変えることができる。

CYCLADESの成果は，しだいに個々の企業に移っていっている。ソフトウェアとハードウェアの開発と技術は，今や商用化できるところにきた。そのレベルの開発はもはやIRIAの使命で

はない。現在IRIAに於ては典型的なネットワーク技術を必要とする軍，財務，あるいは公共のドキュメンテーション・システムのようなプロトタイプのアプリケーションに力が入れられている。

## 第8章の参考文献


- Hawaii 大学

(1) W. H. Bortels, "Simulation of Interference of Packets in The ALOHA Time-Sharing System," March 1970, B 70-2

(2) Alan C. H. Kam, "UHTSS Library Management Yesterday, Today and Tomorrow," Nov. 1970, B 70-3

(3) John M. Davidson, "UHTSS Console Control Program and OS Interface," Aug. 1971, B 71-6

(4) Joho Davidson, "The ALOHA System Interface to TSO," May 1972, B 72-3

(5) Franklin F. Kuo, "Selected Papers on Computer-Communication Nets," May 1974, B 74-4

(6) Richard Binder, "Variable-Length Packets In an Unslotted ALOHA Channel," Jan. 1975, B 75-2

(7) Michael J. Ferguson, "An Analysis of Variable Length Packets in Unsloted ALOHA," Feb. 1975, B 75-7

(8) David Wax, "Design Considerations For The ALOHA Radio Communication Sub-System," Feb. 1975, B 75-11

(9) David Wax, "A Description of The ALOHA Radio Communication Sub-System," Feb. 1975, B 75-12

(10) Franklin F. Kuo "User Standards for Computer Networks," June 1975, B 75-19

(11) Franklin F. Kuo, "Public Policy Issues Concerning ARPANET," June 1975, B 75-20

(12) Richard Binder, "A Possible Scheme for Repeater Addressing," CCG／W-41

- RAND Corporation

(1) Eric F. Harslem and Suganne D. Landa, "VIEW : A distributed system for graphical analysis of large data bases," RAND, pp. 29, P-4972, March 1973

- NBS

(1) Marshall D. Abrams, Ira W. Cotton, "The Service Concept Applied to Computer Networks," Aug. 1975, NBS Technical Note 880

(2) Robert P. Blanc, "Review of Computer Networking Technology," Jan. 1974, NBS Technical Note 804

- BBN

(1) Bolt Beranek and Newman Inc., "Consulting, Research, Development in Sience and Technology"

(2) Bolt Beranek and Newman Inc., "BBN and Computers A History"

(3) Bolt Beranek and Newman Inc., "ANNUAL REPORT 1974"

(4) Bolt Beranek and Newman Inc., "Noise and Vibration Control"

(5) "Technical Reports on the APPA Network Project Issued by BBN Computer System Division as of June 1975"

(6) F. E. Heart, et al, "The interface message processor for ARPA computer network," SJCC'70

(7) Alex Mckenzie, "Host/Host Protocol for the ARPA Network," Jan. 1972, NIC 8246

(8) S. M. Ornstein, et al, "The Terminal IMP for the ARPA computer network," SJCC'72

(9) J. M. Mcquillan, et al, "Improvements in the design and performance of the ARPA network," FJCC'72

(10) A. A. Mckenzie, et al, "The Network Control Center for The ARPA Network," ICCC'72

(11) Nancy W. Mimnt, et al, "Terminal Access to The ARPA Network : Experience and Improvements," COMPCON'73

(12) Peggy M. Karp, "Origin, Development and Current Status of The ARPA Network," COMPCON'73

(13) W. Crowther, et al, "Reliability Issues in The ARPA Network," Nov. 1973

(14) F. E. Heart, et al, "A new minicomputer/multiprocessor for the ARPA network," NCC '73

(15) S. Butterfield, et al, "The Satellite IMP for the ARPA Network," Jan. 1974

(16) "Multiple Minicomputers : Inexpensive and Reliable Computing." Jan. 1975

(17) W. R. Crowther, et al, "Issues in packet switching network design," NCC '75

(18) Bolt Beranek and Newman Inc. "Pluribus Document 1 : Overview," May 1975, Report No. 2999

(19) V. Cerf, et al, "Internetwork End-to-End Protocol," INWG General Note #96, July 1975

(20) David C. Walden, "Experiences in Building, Operating, and Using the ARPA Network," UJCC '75

- IBM-France

(1) Monique Somia-Brecket, "Etude du systéme de contrôle d'un reseau d'ordinateurs distribué et de ses relations avec les systemés de contrôl locaux," Thesis of Ph. D, pp. 306, June 1975 ( in French )

- IRIA

(1) IRIA, "d'emonstrations ou sicob, 18-26 september 1975"

(2) "CYCLADES present configuration"

(3) IRIA, "CYCLADES abstracts," BIB 507. 2, pp. 31, Sept. 1975

(4) Le Bihan, J., "A survey of problems in distributed data bases," DAT 510, pp. 6, July 1975

(5) Pouzin, L., "The CYCLADES network-present state and development trends," RES 505. 2, pp. 11, July 1975

(6) Zimmermann, H., "Insertion d'une station de transport dans un systéme d'exploitation," SCH 546, pp. 8, Jan. 1975

(7) Pouzin, L., "Standards in data communication and computer networks,"

DATACOMM '75, pp. 5, Oct. 1975, SCH 552. 1

(8) Zimmermann, H., "The CYCLADES end to end protocol," DATACOMM '75, pp. 6, Oct. 1975, SCH 565

• Grenoble 大学

(1) J. P. Ansart, R. Pournier and J. du Masle "Support for the CYCLADES network on the IBM 360/67 at the University of Grenoble," SHARE European Association (SEAS) Proceedings 75, Dublin

(2) J. du Masle, "One year of experience with CYCLADES," SEAS 75 Proceedings, Dublin

(3) J. C. Chupin, J. Seguin and G. Sergeant, "Distributed applications on heterogeneous networks," 3 rd Annual Symposium on Computer Architecture, Jan. 19−21, 1976, Florida, U. S. A.

(4) J. du Masle, "NJCL, un language de commande pour reseau d'ordindeurs," pp. 78, July 1973 (in French)

(5) Rite de Calume, "Etude dulanguage de commande et de controle pour le reseau d'ordinateurs SOC (Systéme d'Ordinateurs Connectés)," Thesis of Ph. D, pp. 134, Sept. 1973 (in French)

(6) Guy Sergeant and Mohamed Méjib Farga, "Machine interprétative pour la Mise Oeuvre d'unlanguage de commande sur le reseau CYCLADES," Thesis of Ph. D, pp. 184, Oct. 1974 (in French)

(7) CII, "Socrate, manuel d'utilisation," Reférénce document : 4336 E/FR, pp. 124, July 1973 (in French)

• CTNE

(1) C. T. N. E, "Brief description of the special data transmission and switching network of CTNE," RETD-001-2

• EUROCOMP

(1) Proceedings of European Computing Conference on Communication Networks, Sept. 1975, pp. 620

# あ と が き

ARPANETの稼働が開始されてすでに6，7年の歳月が経過し，商用のコンピュータ・ネットワーク・サービスも徐々に軌道に乗り始めた。一方，現存のシステムは真に理想とするコンピュータ・ネットワークとは程遠く，ネットワークがその真価を発揮するには，現在のシステムになお多大の論理的機能を付加する必要があり，その実用化にはなお多年にわたる研究と実証の努力が重ねられねばならぬことも指摘されている。

各種の異なったリソースの真に有効な共用を実現するには異機種間のデータやデータ・ベースの完全な互換性の確立，ハードウェアおよびソフトウェアの異機種間の自由なポータビリティやロードのシェアリングの実現等，解決せねばならぬ難題がなお多数存在している。

JIPNETの目標もこれら新技術の一角を実現し，その有効性の検証を試みることにあるが，その実験台としての基本的ネットワーク・ファシリティのインプリメントの段階に於ても幾多の問題に遭遇し，それはそれで貴重なる体験であった。

我が国に於ても近年急激にシステムのネットワーク化に対するニーズが高まりつつある。とくにリソースの分散とその共用はコンピュータ利用形態の必然的な要求として経済性，効率性，信頼性の優れた真の実用システムの出現が期待されている。この期にあたり，研究開発の貴重な機会を与えられたJIPNETに課せられた使命の重大さを痛感するとともに，この経験を踏み台として，更に新しい研究開発が継続し得ることを念ずるものである。

昭和51年3月発行

発行所　財団法人　日本情報処理開発協会
（旧 財団法人 日本情報処理開発センター）
東京都港区芝公園3丁目5番8号
機械振興会館内
TEL（434）8211（代表）

印刷所　三協印刷株式会社
東京都渋谷区渋谷3丁目11番11号
TEL（407）7316

50－S003