

48—S 001

コンピュータ ネットワーク システムの 研 究 開 発

昭和 49 年 3 月



財団法人 日本情報処理開発センター

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48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を共用する、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、現在世界的に急速に進展しつつあります。

タイムシェアリング・システムの次に来るものとして、ネットワークを通してのコンピュータの利用法は、今後の情報処理形態の一つの焦点になるものと考えられます。

我が国においては、この分野の研究開発が、欧米にくらべ、やや立ち遅れていると言われておりますが、最近になってその気運はかなり高まってまいりました。しかし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ーの複数台のコンピュータを持ち、このようなネットワークの研究開発を実施しうる環境をそなえていると判断いたし、昭和48年度より、数ヶ年の計画で、当財団内の3機種を結合したリソース・シェアリング・ネットワークの建設をおこなうことといたしました。このネットワークは将来、外部への拡張も考慮いたしておりますが、本年度は、その第一年目にあたり、ネットワークの基本構想の作成およびファシリティとしての基本的なハードウェア、ソフトウェアの設計を中心におこない、本書はシステムの間接報告としてその成果をまとめたものであります。

本報告書が、この方面に興味の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お願いいたす次第であります。

昭和 49 年 3 月

財団法人 日本情報処理開発センター

会長 難 波 捷 吾

目 次

まえがき	
1 JIPNETの概要	3
1.1 コンピュータ・ネットワークの現況	3
1.2 JIPNETの目的	10
1.3 JIPNETの機能概要	11
2 JIPNETのシステム構成	17
2.1 ホストシステムの概要	17
2.2 結合方法とシステム構成	28
2.3 IMP, TIPのハードウェアと各種端末	33
2.4 インタフェース・アダプタ	53
3 サブネットのソフトウェア	75
3.1 サブネットの基本構造	75
3.2 サブネットの位相幾何学的構造	79
3.3 IMPの機能	83
3.4 IMPのプログラム構造, データ・ストラクチャ	123
3.5 サブネットの解析, 評価に関する定量的手法	128
4 HOSTの基本ソフトウェア	139
4.1 基本プロトコル	139
4.2 F-NCP	182
4.3 H-NCP	207
参考文献	249

ま え が き

複数のコンピュータを結合し、ハードウェア、ソフトウェア、データベース等の総合したリソースを共用するリソース・シェアリング・コンピュータ・ネットワークの開発は、米国を始め、ヨーロッパ、カナダ、オーストラリア等を含め、近年全世界にわたり、非常に活発になって来た。

リソース・シェアリング・ネットワークの利点は、これらリソースの共用による重複投資の軽減による経済性のほかに、複数システムへのアクセスによるサービス性の向上、処理負荷の最適の分担によるネットワーク全リソースの有効利用、1部の障害を他のリソースでカバーし得る信頼性の向上等、多くの利点があげられている。とくに異機種コンピュータ同志の結合の場合は、ハードウェア、ソフトウェアの互換性のある程度拡張しうる手段として、標準化にかわる他のアプローチとしての効果もあげられている。

一方、このような利点の裏ハラとして、あるいは技術的な難解さとして幾多の問題点をも含んでいる。

例えば結合にともなう、個々のシステムの機能、効率の低下、従来システムとの運用上の互換性への影響、ネットワーク・システムのメリットの、コスト効率としての把握の困難さ、あるいは、現状における建設費、運用費をふくめたコスト高の問題、さらにこれに見合うニーズの把握等今後の解決に待つべきものが多々ある。

我が国におけるコンピュータ・ネットワークの開発計画は米国には勿論、欧州諸国と比べてもかなり立ち遅れの感がある。しかしながら、最近、ネットワークに寄せる関心そのものはかなり高まってきている。現在必要なのは机上での研究あるいは学習のみでなく、自ら手を下し、小規模であっても、それぞれの目的に応じて具体的なネットワーク・システムを建設し、開発段階あるいは試用段階における諸問題を体験評価し、実用システムへの技術の修得と蓄積をおこなうことではなからうか。

この様な現状をふまえ、当センターに於ては、昭和48年度より数ヶ年の計画により、現在設置されている代表的な国産三機種を相互に結合し、ハードウェア、ソフトウェアおよびデータベースをシェアリングするネットワークの建設に着手した。

このシステムは実用と実験の両目的を兼ね、2、3年後には外部への拡張も考慮している。

およそその計画としては、48年、49年の2年度で、ファシリティとしての基礎的なハードウェア、ソフトウェアのインプリメントをおこない、50年度以降引きつづき利用面に主力をおいたハイレベル・プロトコルの設定および処理方式の検討をおこなう予定である。

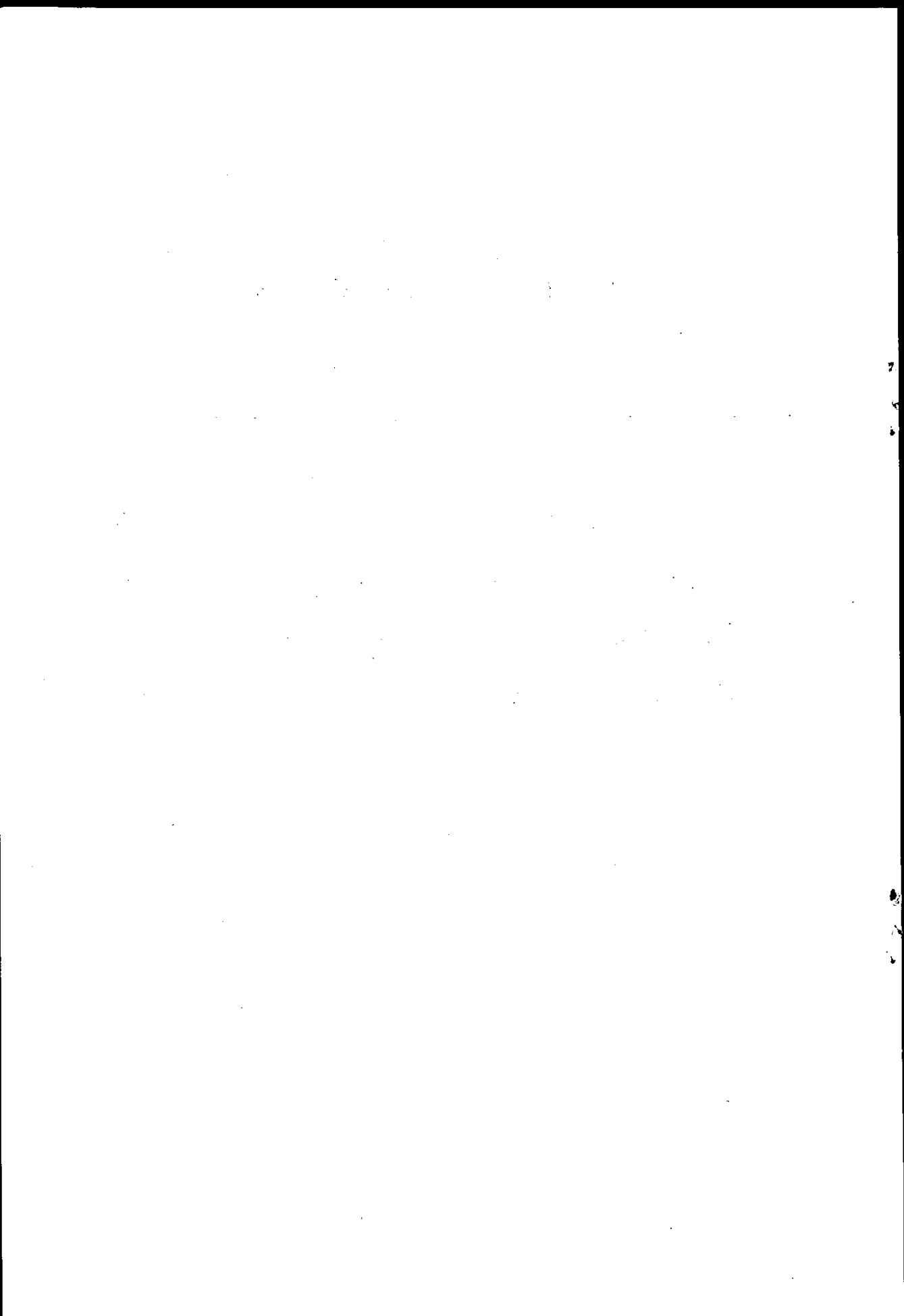
本年度は、その第一年目にあたり、このネットワークシステムの基本構想を建て、結合にともな

うハードウェアの作成，コントロール・ソフトウェアの基本設計等をおこなった。本報告書は，これらの内容をまとめたものであり，ネットワーク内のリソースの自動的な割り当て，あるいは異機種にわたる分散型データベースアクセス等を中心とするハイレベル・プロトコルの処理に関してはまだふれていない。

第1章では，当センターのネットワーク JIPNET (JIPDEC Integrated Project Network) の目的および機能の概要を，第2章ではシステム構成およびハードウェアについて，第3章ではサブネットのソフトウェアの機能を，第4章では各ホストコンピュータ側のネットワーク・コントロール・プログラムの機能につき，それぞれ述べている。

1. J I P N E T の 概 要

1.1	コンピュータ・ネットワークの現況	3
1.1.1	リソース・シェアリング・コンピュータ ・ネットワークの概念	3
1.1.2	コンピュータ・ネットワークの現況	4
1.2	JIPNETの目的	10
1.3	JIPNETの機能概要	11



1. JIPNETの概要

本章では、次章以降での詳細な説明に入るまえに、JIPNETシステムの概要を紹介する。

1.1ではまず、コンピュータ・ネットワークの概念を説明した後、内外におけるコンピュータ・ネットワークの現況を簡単に紹介する。次に1.2でJIPNETの目標あるいはそのねらいを明らかにした後、1.3でJIPNETの機能概要を紹介する。

1.1 コンピュータ・ネットワークの現況

1.1.1 リソース・シェアリング・コンピュータ・ネットワークの概念

リソース・シェアリング・コンピュータ・ネットワークとは、独立した自律系として動作する複数のコンピュータ・システムを結合し、相互にリソースの共同利用を可能にするような形態をいう。この場合のリソースというのは、単なるハードウェア機能のみではなく、ソフトウェアおよびデータを含む広範な意味でのリソースを対象にしている。

コンピュータ・ネットワークに参加するコンピュータ・システム（ホスト・コンピュータ）は地理的に離れて散在しているのが普通であり、コンピュータ・ネットワークは、これらのホスト・コンピュータ群と、ホスト・コンピュータ同志を結合するための高速通信回線からなるデータ交換網とから構成されることになる。この交換網のことを、通常サブネットと呼んでおり、その役目は、データを信頼度よく目的地まで伝送することである。このようなサブネットの交換局にあたる部分をノードと呼び、通常は交換専用の小型コンピュータ（IMP Interface Message Processor またはNode processor）で構成される。

図1-1はコンピュータ・ネットワークの構成を示している。コンピュータ・ネットワークを構成するホスト・コンピュータは、すべて同機種（オペレーティング・システムも含む）で構成される場合と、異機種のコンピュータで構成される場合とがある。同機種のコンピュータ・ネットワークはロードの分散とか、プログラムやデータの共用を比較的容易におこなうことができるが、特殊なハードウェアおよび、ソフトウェアでのみ処理できる作業の実行は、異機種コンピュータ・ネットワークによって始めて可能になる。

このようなコンピュータ・ネットワークの目標は、コンピュータ・システムの専門化、重複投資の回避、地域情報サービスの強化であると言われている。

たとえば、ある一つのコンピュータ・センターの運用ということを考えた場合、センターの利用者は夫々多種多様な要求を持っており、これらの利用者すべてに満足のいくサービスを同時に行お

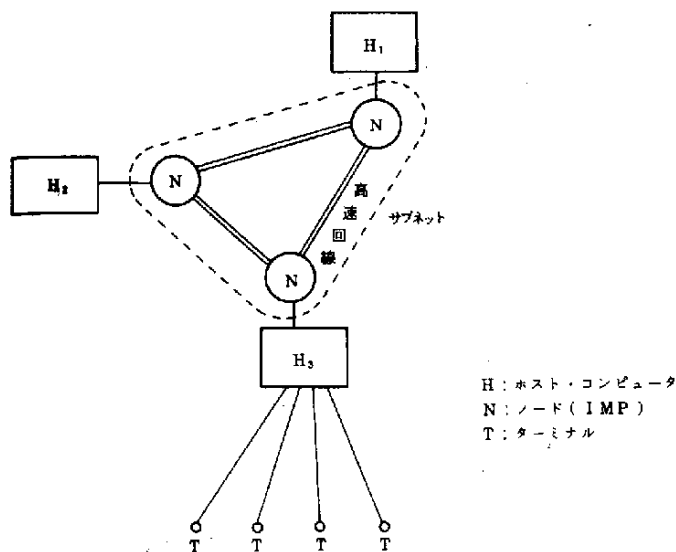


図 1-1 コンピュータ・ネットワークの構成

うとすれば、超大型のコンピュータ・センターにせざるを得ないが、その大型化には限界がある。そこで、このセンターが、コンピュータ・ネットワークに参加し、自分のセンター（ローカル・ホスト）では処理しきれない仕事は、他のセンター（リモート・ホスト）に依頼するようなことができれば、各コンピュータ・センターは、自分のセンターが最も得意とする分野に、そのサービスの力を集中することができるようになる。センターの利用者としても、自分が利用しているローカル・ホストのみではなくコンピュータ・ネットワークに参加しているすべてのホスト・コンピュータのサービスを自由に受けることができるとなれば、より広い且つ高度なアプリケーションに対処することが可能になる。

このようなリソース・シェアリング・コンピュータ・ネットワークの形成によりネットワークに参加するホスト・コンピュータは得意とする専門的サービスに専念でき、さらにそれらのサービス・ファシリティの開発に伴う総合的な費用の重複投資を回避することが可能になり、かつ、そのローカル・ホストで従来、ローカルにサービスされていた情報サービス内容は質量ともに飛躍的に向上することになる。

1.1.2 コンピュータ・ネットワークの現況

リソース・シェアリング・コンピュータ・ネットワークの最も代表的な例としては米国国防省の ARPA (Advanced Research Project Agency) による ARPANET があげられる。ARPANET は 1960 年代の後半からその開発に着手し、4 ノード結合による実験が 1969 年末に始められて以来図 1-2 に示すように、次々と拡張を重ね現在 (1973 年 9 月) は 40 ノードとなり、米国全土の主

要大学および研究機関を含み、更に米本土のみならず、ハワイ、英国、ノルウェー等にも拡張されている。図1-3に1973年9月現在におけるロジカルマップを示す。また図1-4はARPANETの地理的なMAPを示し、表1-1に各ノードの機関名を示す。

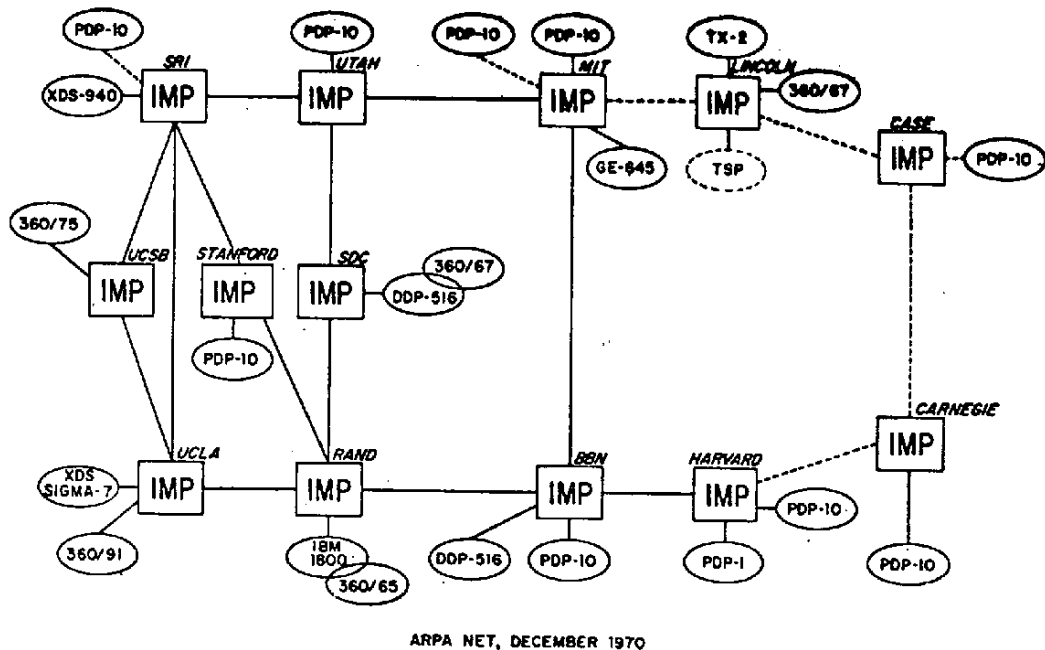


図1-2 Evolution of the ARPA Network

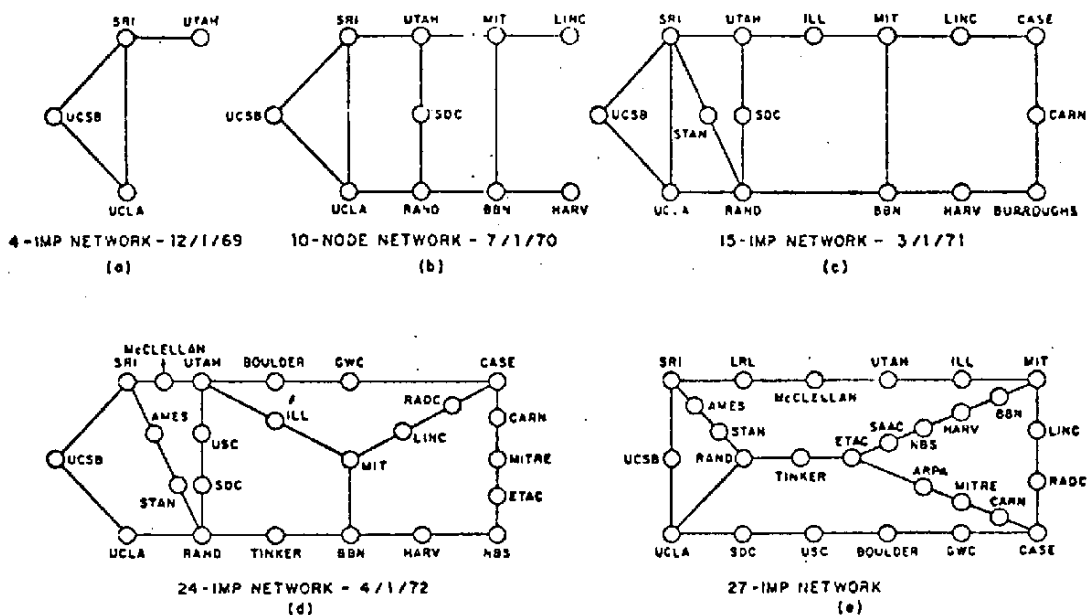


図1-3 ARPA NETWORK, LOGICAL MAP, SEPTEMBER 1973

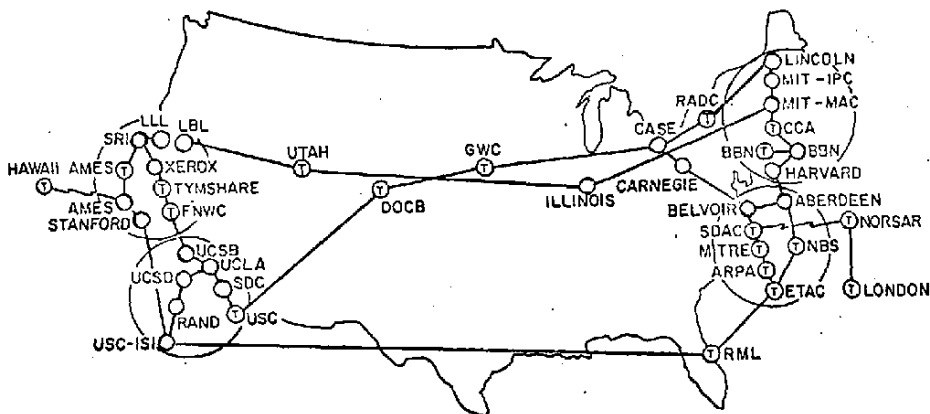


图 1-4 ARPANET, GEOGRAPHIC MAP SEPTEMBER 1973

表 1-1

ARPANET
OPERATIONAL NETWORK NODES

* Aberdeen	-- U. S. Army Materiel Command, Aberdeen R&D Center, Maryland
* AFGWC	-- Air Force Global Weather Central, Omaha, Nebraska
* Ames	-- Ames Research Center, NASA, Mountain View, California
* ARPA	-- Advanced Research Projects Agency, Arlington, Virginia
* Belvoir	-- Mobility Equipment R&D Command, Ft. Belvoir, Virginia
* BBN	-- Bolt Beranek & Newman Inc., Cambridge, Massachusetts
* Case	-- Case Western Reserve University, Cleveland, Ohio
* CCA	-- Computer Corporation of America, Cambridge, Massachusetts
* CHI	-- Culler-Harrison Inc., Goleta, California
* CMU	-- Carnegie-Mellon University, Pittsburgh, Pennsylvania
* DOCB	-- Department of Commerce, Boulder, Colorado
* ETAC	-- Air Force Environmental Technical Applications Center
** FNWC	-- Fleet Numerical Weather Central, Monterey, California
** Harvard	-- Harvard University, Cambridge, Massachusetts
** Hawaii	-- University of Hawaii, Honolulu, Hawaii
** Illinois	-- University of Illinois, Champaign-Urbana
** ISI	-- Information Sciences Institute, Marina del Rey, California
** LBL	-- Lawrence Berkeley Laboratory, Berkeley, California
** Lincoln	-- Lincoln Laboratory, MIT, Bedford, Massachusetts
** LLL	-- Lawrence Livermore Laboratory, Livermore, California
* London	-- University of London, London, England
* Michigan	-- University of Michigan, Ann Arbor, Michigan
* MIT	-- Massachusetts Institute of Technology, Cambridge, Massachusetts
* Mitre	-- Mitre Corporation, McLean, Virginia
** Moffett	-- Moffett Field, Mountain View, California
** NBS	-- National Bureau of Standards, Gaithersburg, Maryland
* NORSAR	-- Norwegian Seismic Array, Kjeller, Norway
* RADCC	-- Rome Air Development Center, Rome, New York
* Rand	-- The Rand Corporation, Santa Monica, California
* RML	-- Range Measurements Laboratory, Patrick AFB, Florida
* Rutgers	-- Rutgers University, New Brunswick, New Jersey
* SCRL	-- Speech Communications Research Lab, Santa Barbara, California
* SDAC	-- Seismic Data Analysis Center, Alexandria, Virginia
* SDC	-- System Development Corporation, Santa Monica, California
* SRI	-- Stanford Research Institute, Menlo Park, California
* SU	-- Stanford University, Palo Alto, California
* UCLA	-- University of California at Los Angeles
* UCSB	-- University of California at Santa Barbara
* UCSD	-- University of California at San Diego
* UNIVAC	-- UNIVAC, St. Paul, Minnesota
* USC	-- University of Southern California, Los Angeles
* Utah	-- University of Utah, Salt Lake City
** X-PARC	-- Palo Alto Research Center, Xerox Corporation, Palo Alto, Calif.

* Operations commence May-August 1973

** Operations commence September-December 1973

表1-2 コンピュータ・ネットワークの例

ネットワーク名	タイプ	ホスト数	機種	端末からのネットワークアクセス	回線速度	制御用マシン	説明
ARPA	分散型	55	異機種 6メーカーの 17機種	可	50KBPS	DDP316/516	全米主要大学、研究機関を結ぶ研究用ネットワーク。ハワイ、英国、ノルウェーに通信衛星により連結、米国防省ARPAプロジェクトの1つ。
CYBERNET	分散型	13	同一メーカー機種 7600 CDC 6600 6400 3300	可	40.8KBPS	Wideband Switch	CDCの商業用ネットワーク 米国東、西、中部全土にわたる。
IBM TSS	分散型	6	同機種 IBM360/67	不可	40.8KBPS	IBM2701	BTL, NASA, IBM Watson 研究所, Princeton 大学, Carnegie-Mellon 大学 等 360/67 ユーザを結ぶ研究用。
NETWORK /440	集中型	12	異機種 IBM360/91 67, 44 等 CDC 6600	不可	2.4~ 40.8KBPS	IBM 360/91	IBM Yorktown Research Center で開 発した研究用であるが、IBM 内部のみでな く Yale univ. を含むいくつかの IBM ユ ーザとも結合している。
TUCC	集中型	3	同一メーカー機種 IBM360/75 40	可	40.8KBPS	IBM 370/165	ノースキャロライナ州の3大学を結び、州 内の約50の他の機関から端末アクセスも可 能
DCS	リング型	6	異機種 PDP-10 15 その他中小機種	可	2~5MBPS (同軸ケーブル)	リング インターフェース	California 大学, Irvin で開発した実 験用、学内ネットワーク、リングインター フェースが特徴である。
MERIT	分散型	3	異機種 IBM360/67 CDC 6500	不可	2000 BPS	PDP 11	Michigan 州, Wayne 州の3大学を結ぶ、 ARPA と似た考え方による。
NPL	分散型	13	異機種 ICL ELLIOTT PDP 8 等	可	1MBPS (同軸ケーブル)	DDP 516	英国 National Physical Laboratory 内の 研究用ネットワークである。パケットスイ ッチングの新技術に特徴がある。

その他のネットワークとしては、表 1-2 に示すようなシステムがあり、さらに、フランスにおける CYCLADES、カナダの CANUNET、オーストラリアにおけるコンピュータ・センター間のネットワーク計画、スペインにおけるネットワーク、EC 8ヶ国を結ぶ COST 11 等、各国あるいは国際的な規模で、その開発が進められている。

ところで、我国におけるコンピュータ・ネットワークは、欧米の動きに比べて、著しく立遅れているというのがその実状であるが、最近になって京都大学の KUIPNET (図 1-5)、電総研のコンピュータ・コンプレックス的な EPICS システム (図 1-6) を始めとして数種のシステムが実験あるいは計画を開始した。但し、高速通信回線を使用しての全国的なコンピュータ・ネットワークは現状では皆無の状態であつて昭和 51 年頃といわれている電電公社による高速デジタル・データ網のサービスにファシリティとして期待がかけられている。

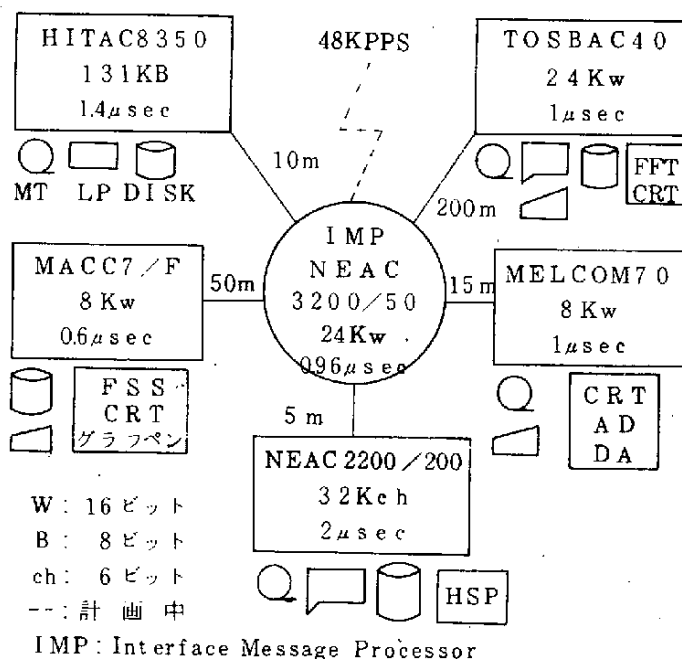
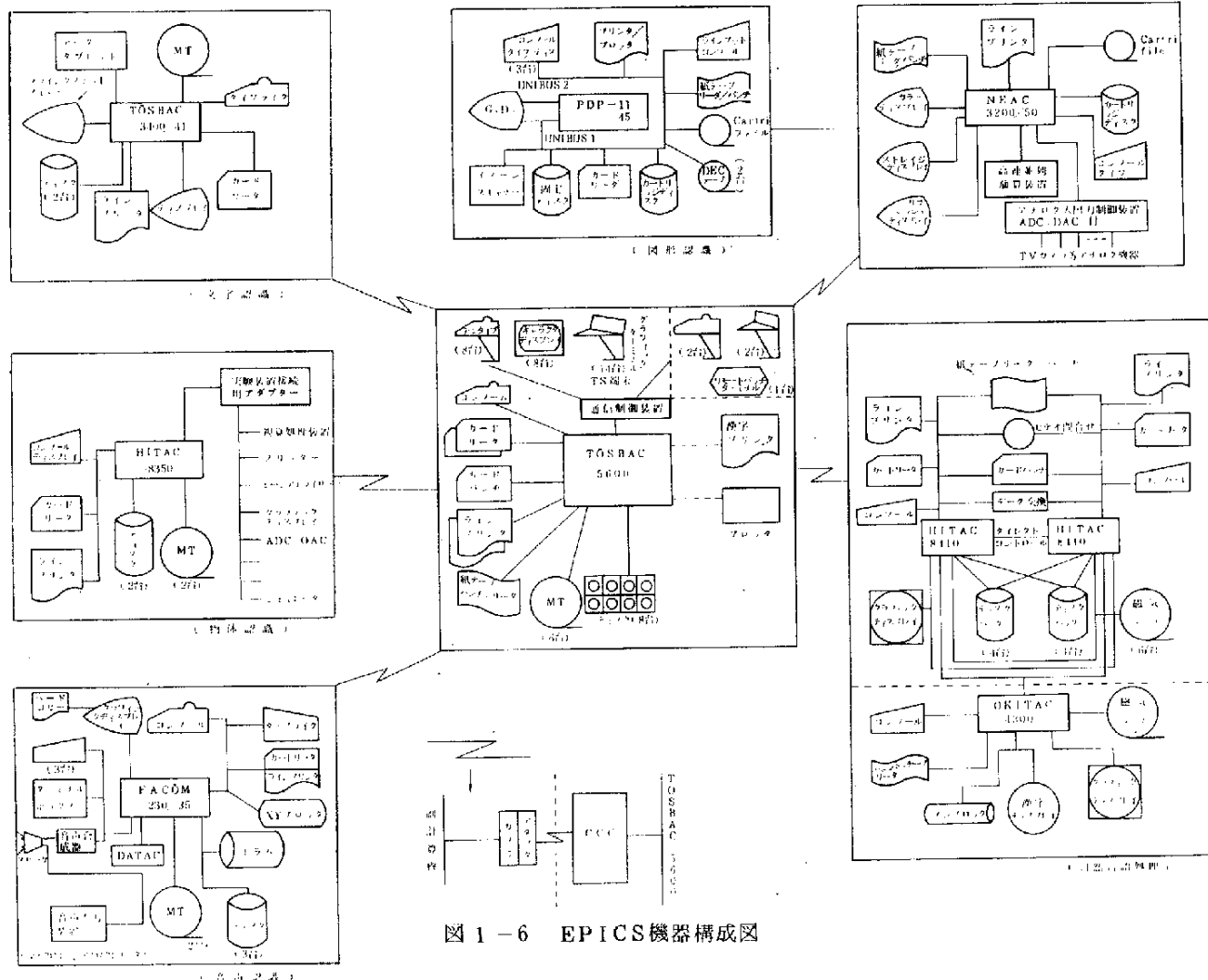


図 1-5 KUIPNETの構成図



1.2 JIPNET の目的

JIPDECにおけるコンピュータ・ネットワークは、本年度より数年計画で実現すべく着手したものであるが、JIPNETの目的について以下に記す。

JIPNETは実用と実験の両方の目的を持ったものである。ネットワークの規模は当初まずセンター内の国産3機種同志を結合したきわめてローカルなものであるが内部における2、3年の実用実験を経てTIPからの使用を含め逐次外部ネットワークあるいはシステムへの拡張も考慮している。

実用上の要求としては次の様なものがある。

- (1) JIPDECに現在設置されている異なる3機種のハードウェア、ソフトウェア、データセット等の限界を除き、総合したリソースを利用したいという希望が以前からあった。
- (2) プログラムやデータ・ファイルの互換性を標準化とは異なるアプローチにより確保したい。
- (3) 漢字入出力、図形処理装置、マークシート・リーダ等の特殊周辺装置をどのCPUからも共用したい。
- (4) 数年前から実施しているTSSに於て、端末からどのシステムでもアクセス出来るようにしたい。
- (5) 現在機種によるロードのアンバランスがかなりあるがなるべく平均化したい。
- (6) ある機種のダウンによって不可能になっていた処理を他の機種でカバーする。
- (7) コンピュータ・ネットワーク・システムの効率評価のデータを得る。

一方、実験的な目的としては次のようなものがある。

- (1) 個々のシステムの、結合したが故の負荷の増大、機能や処理効率の低下などを実測する。
- (2) 個々のシステムの結合前と結合後における運用上の互換性がどの程度保てるか。変更せねばならぬ点がどの程度であるかを体得する。
- (3) 実用上、ユーザレベルのプロトコル(protocol)としてどの様なものが必要であるか、またそれらの処理が異なる機種間でどの程度効率よくおこなわれるか。
- (4) ネットワーク利用のメリットをコスト効率という観点からとらえる。
- (5) 異機種間の分散型データベースが常識的な効率下においてマネージ出来る方式および限界を研究する。
- (6) 異機種間のオートマチックなハードウェアおよびソフトウェア・シェアリングの方法および有効性などを研究する。

以上のような実用および実験の両目的を兼ね、これを国産3機種の上で実現することにより、異機種結合のリソース・シェアリング・コンピュータ・ネットワークの我が国における実用の可能性の足がかりを得たい。

1.3 JIPNETの機能概要

前節で述べたようなJIPNETの目的を達成するために、JIPDECでは、現在設置されている国産機三機種（FACOM230/60, HITAC-8450, NEAC 2200/500）を相互に結合することにより、ローカルなコンピュータ・ネットワークを形成し、基礎的な制御用ソフトウェアを開発した上で、コンピュータ・ネットワークの新しい利用法の実験をおこなうこととなった。

現在保有の三機種の構成については2.1（17頁）で概要を紹介するが、いずれも中、大型機であり、ワード、バイト、キャラクタの夫々を単位とするハードウェア、ソフトウェアとも全く異質なマシンである。

これら三つのホスト・コンピュータでコンピュータ・ネットワークを形成するにあたっての結合方法、ネットワーク・システムの全体構成については、次章の2.2および3章で詳しく述べるが、実験的性格および拡張性を重視し、ローカルなネットワークではあるが、高速通信回線（構内専用回線：48 kbps）とIMP 3セットによるサブネットを形成することとした。また、サブネット3ノードのうち、2ノードには、各種の端末を結合したTIP（Terminal IMP）としての機能を持たせており、各種端末から各ホストへの自由なアクセスができるようになっている。このようなサブネットが持つべき機能およびソフトウェアについては3章で詳しく述べるが、ホスト間を行来するメッセージを信頼度良く送り届けるためのメッセージ交換機能を果たすMCP（Message Control Program）が主体になり、TIPにはMCPのほかに端末類をコントロールするプログラムが追加された形となる。

一方、各ホストには、ネットワーク内のプロセス同志が互いに通信するための機能をサポートするNCP（Network Control Program）が存在する。

NCPの機能については第4章で詳しく述べるが、主なものは、サブネットとホストとのインタフェース、他ホストNCPとの通信、ユーザ・プロセスの管理およびユーザ・プロセスとの通信等である。NCPはまた夫々のホストの既存OSとネットワーク・ジョブとのインタフェースの役もつかさどる。

JIPNETは互に独立した機能を果たすホストシステムと、サブネット・システムを結合することによって形成されるが、これらの独立したシステム同志が協調するためには、互いの機能を適正に分担しそれらの機能相互間で情報を伝達するための正確な約束事が必要になってくる。

関連するシステム間で正確な情報伝達をおこなうためにとり決められたデータあるいはメッセー

ジの受渡し手順、そのフォーマットおよび内容等に関するすべての約束事をまとめて、これをプロトコル (protocol) と呼んでいる。

JIPNETに必要なとなるプロトコルについては第4章で詳しく述べるが、IMP-HOST-NCP-ユーザ・プロセスの各レベルで次のようなものが必要になる。

- ① コミュニケーション・サブネットの効果的な運転を実現するためのIMP-IMPプロトコル。
- ② ホストとIMPのつなぎとして主としてデータの送受方式を規定するHOST-IMPプロトコル。
- ③ ホスト同志が交信する方式を規定するHOST-HOSTプロトコル。
- ④ HOSTがユーザ・プロセスに対してサービスするコマンドを規定したプロトコル。
- ⑤ ユーザに対して機能レベル (ユーティリティ) のサービスをおこなう高位のプロトコル。

このうちユーザ・プロセスが直接かわりあうのは、④と⑤であり、①、②、③はシステムレベルのプロトコルである。図1-7に各プロトコルの階層関係を示す。

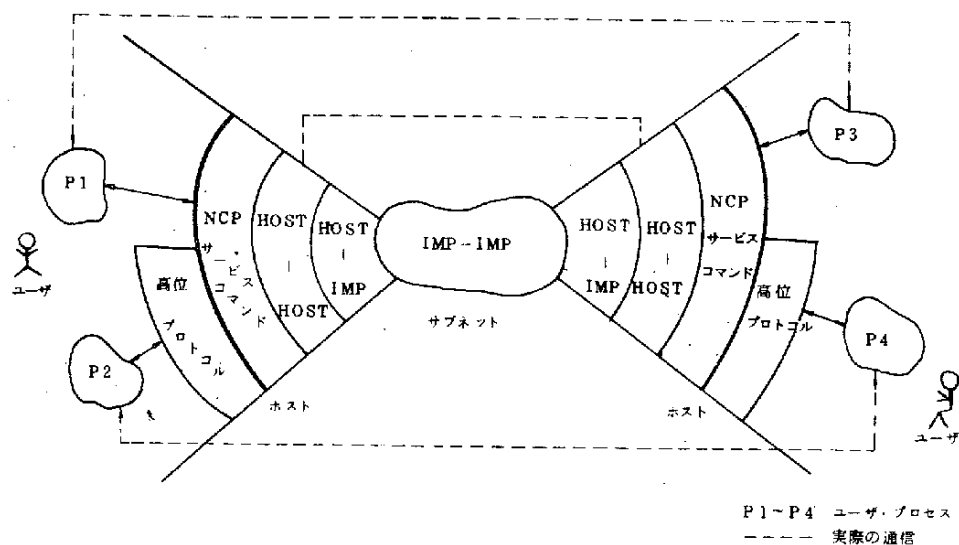


図1-7 プロトコルの階層

ユーザは④のNCPサービス・コマンドを使用することにより、コンピュータ・ネットワークのリソースをほぼすべて利用することができるわけであるが、④の機能は、ネットワーク内のユーザ・プロセスが互いに通信できるという、極めてベーシックなものであり、だれでも手軽に利用できるとは言えない。

このため一般ユーザにはより高いレベルの機能をサポートする必要がある。例えばあるホストのローカルTSS端末からネットワークを通して他ホストのプロセスへのアクセスを可能にする、ネットワーク内のホスト間でファイルを転送する、あるいは他ホストへのリモート・ジョブ・エントリを可能にする

などといった、いわゆるユーティリティ・レベルのプロトコルを用意してユーザの利用を更に容易にする必要がある。

、これらのプロトコルを、高位のプロトコルと呼び、④のNCPサービスコマンドを利用して、必要に応じて作成追加していくことが可能である。

ARPANETにおいては上記の機能を果たすものを夫々TELNETプロトコル(Telecommunication Network protocol)、File Transferプロトコル、RJEプロトコルと呼んでいる。

JIPNETではこのような基本的な機能を持つコンピュータ・ネットワークをつくりあげたうえで、更に次のような利用法の実験をおこなうことを考えている。

(1) CJP (Cooperated Job Processing)

CJPはネットワーク内のホスト・コンピュータ群を複合コンピュータとして扱い、複数のホストにユーザの指定するジョブをそれぞれ発生させ、それらのジョブ間の適当な同期をとりコントロールおよびデータの交信をおこないながら、複数ジョブの協同作業により、あるまとまった仕事を処理するものである。すなわち、これはネットワーク内の複数ホストにまたがるマルチ・タスクの実現にはかならない。このような利用法は、ARPANETにおいては、複数のホストシステムによるMcROSSという航空管制シミュレータの例があるが、これは、いずれもPDP10という同一機種、同一オペレーティング・システムTENEXのもとでのCJPである。JIPNETでは、異機種間でのCJP利用を容易にするために、基本プロトコルとしてプロセス管理に関するコマンドを設けプロセスの発生、実行指令、消去、POST-WAITによるプロセス同期などがユーザ・プロセスから利用可能のように配慮した。

実際のアプリケーション実験としてはH-8450のグラフィック・システムとF230/600の高速演算性能、大容量記憶を利用したインタラクティブ・システムが考えられている。

(2) AJD (Automatic Job Dispatching)

これは、ネットワーク・ジョブを処理するホストを自動的に選択するような機能である。ユーザがネットワークにジョブの処理依頼をおこなうと、ネットワークでは、各ホストの機能、状態、負荷等を考慮し、最適ホストを選び、ユーザには各ホスト・マシンを意識させることなく、ネットワークの総合機能の利用を可能にする。各ホストの負荷バランスを自動的に最適化し、ホスト群まで含んだネットワーク全体のコンピュータパワーの信頼性を向上させることができるように考慮する必要があるであろう。

AJDの実現のためには、各ホスト・システムのハードウェア、ソフトウェアの機能を把握しておくための、ARPANETにおけるNIC(Network Information Center)的な機能に、ダイナミックな各ホストの状態、リソースの現状、負荷等を把握しておく機能を加えたNCIC(Network Control Information Center)の機能および管理法についても検討が必要である。

(3) DDBA (Distributed Date Base Accessing)

最近我国の各省庁ではそれぞれ独自の情報ネットワーク・システムの構想が次々と計画されている。

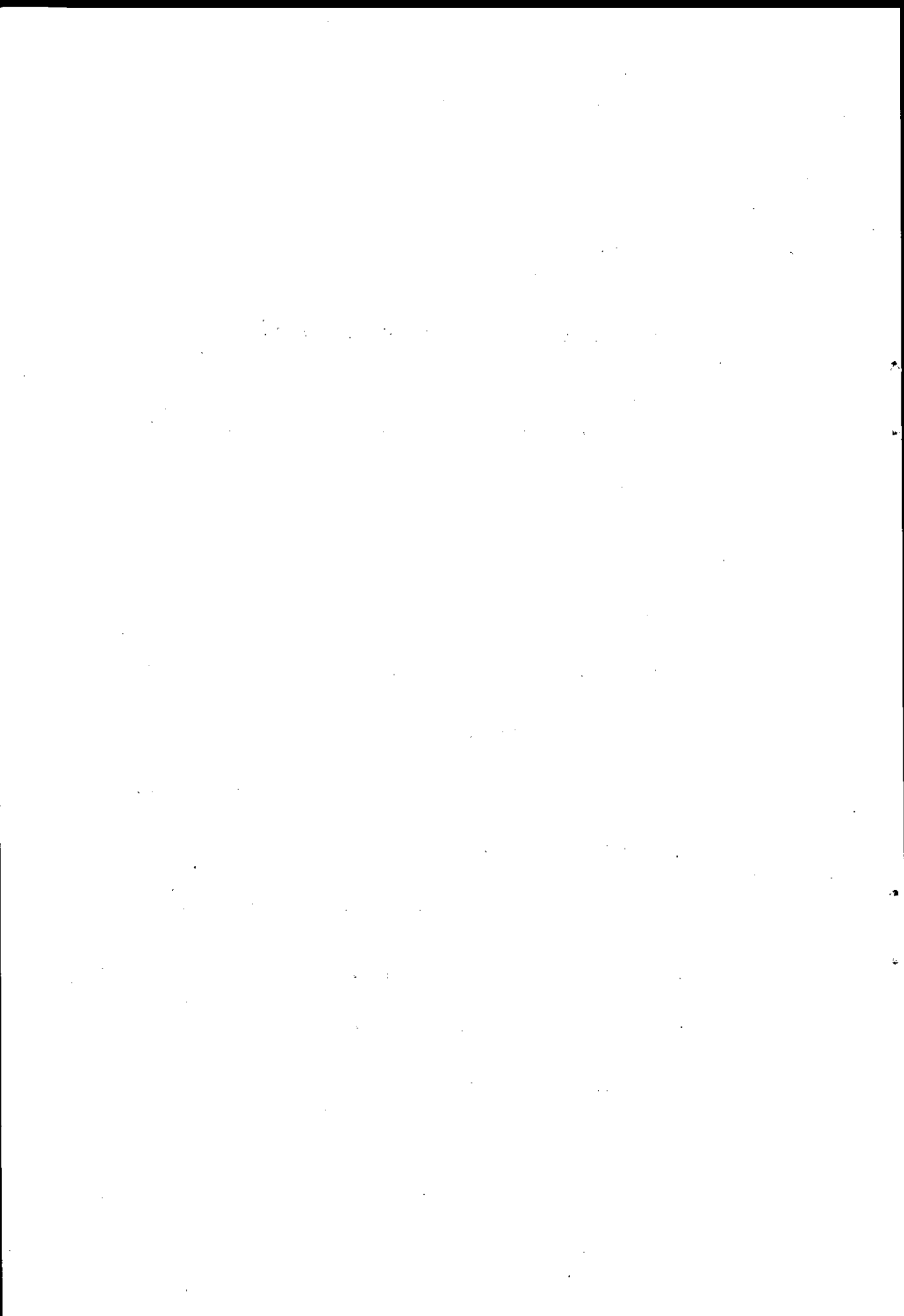
この場合、中心となる技術の一つは、ネットワーク内における各種のデータベースへのアクセスの手法であろう。

DDBAはネットワーク内における集中型、分散型のあらゆる論理および物理構造をもったデータベースへの自由なアクセスを可能とするもので、特に異機種間に分散した各種のデータベースを共通のプロトコルにより管理する機能である。

この実現のためにはデータ・ベース・マネージメントのためのハイレベル言語の開発や、その基本となる汎用データ管理方式の検討が必要である。そのためにまず、ファイル情報の標準形式および、各ホスト間におけるその情報の転送、具体的なデータ受け渡し法などを規定したハイレベル・プロトコルRDAS (Remort Data Access Support) を作成し、この機能をベースとして、DDBAを実現する。なおRDASについては4章でその詳細を述べる。

2. JIPNETのシステム構成

2.1	ホストシステムの概要	17
2.2	結合方法とシステム構成	28
2.2.1	結合方法	28
2.2.2	システムの構成	30
2.3	IMP, TIPのハードウェアと各種端末	33
2.3.1	IMP, TIPの機種選定	33
2.3.2	IMP, TIPのハードウェア構成	37
2.3.3	各種端末の概要	49
2.4	インタフェース・アダプタ	53
2.4.1	インタフェース・アダプタの機能設定	53
2.4.2	インタフェース・アダプタのコマンドと手順	57
2.4.3	IMP側アダプタの動作	66



2. JIPNETのシステム構成

2.1 ホスト・システムの概要

JIPNETには、現在、FACOM 230/60、HITAC-8450、NEAC 2200/500 の三機種が設置されており、それぞれ次のような特徴を有している。

(1) FACOM 230/60

高速演算性能、大容量記憶装置、大容量ファイルと、大型モニターMonitor Vのタイムシェアリング機能を利用することにより、外部および所内へのTSSサービス、各種オンライン・システムの開発。

49年度以降モニターはMonitor VIにバージョン・アップされ、さらにハードウェアもFACOM 230/75 にレベル・アップし、漢字入出力装置、キャラクタ・ディスプレイ端末の増設等が予定されている。

(2) HITAC - 8450

一般データ処理に適したバランスのとれた機器構成であり、さらにグラフィック・コンピュータが結合されており、マルチ・プロセッサにより汎用グラフィック・ソフトウェアを開発し、種々の図形入出力用アプリケーション・システムをサポートしている。

(3) NEAC 2200/500

一般バッチ・データ処理に使用しているが、次年度以降、新シリーズの機種に変更し、さらにデータ・ベースの充実、マーク・シート・リーダ等を結合する予定である。

以下にこれら三機種のハードウェア諸元、システム構成およびメーカー提供ソフトウェア等の一覧を示す。

FACOM-230/60 ハードウェア諸元

装 置 名		台数	性 能 ・ そ の 他	
主 記 憶 装 置	大容量記憶装置		2 6 2 K語	
	高速度記憶装置		1 9 6 K語	
外部記憶装置	集合 ディスクパック 装置	2	転 送 速 度	2 5 0 K B / 秒
			平均 アクセス	7 5 m s
			容 量	2 9.1 7 M B × 1 6 パック

装 置 名	台数	性 能 ・ そ の 他
磁気ドラム装置	3	転送速度 156KB/秒 平均アクセス 17ms 容 量 2.56MB×3
磁気テープ装置	6	転送速度 60KB/秒 記録密度 800BPI 記録型式 9トラック
	1	転送速度 60KB/秒 記録密度 800BPI 記録型式 7トラック
入出力装置	カード読取装置	1 800枚/分
	カードさん孔装置	1 250枚/分
	ラインプリンタ装置	1 62字種, 136字/行 500行~1,500行/分
		1 109字種, 136字/行 500行~1,000行/分
	タイプライタ装置	2 128字種, 120字/行, 900字/分
	紙テープさん孔装置	1 1,200字/秒
	紙テープ読取装置	1 200字/秒
	キャラクターディスプレイ装置	2 128字種, 50字×20行
	XYプロッタ装置	1
端 末 装 置	リモートパッチ 端末	6 伝 送 速 度 1,200ボー 主 記 憶 装 置 8K語 カード読取装置 100枚/分 タイプライタ装置 128字種, 120字/行, 1,200字/分 ラインプリンタ装置 120行/分, 136字/行 設 置 先 工技院機械技術研究所 当センタ内…4台

装 置 名		台数	性 能 ・ そ の 他
	デ マ ン ド 端 末	6	伝 送 速 度 200ボー タイプライタ 装置 128字種, 120字/行, 1,200字/分 設 置 先 機振協, 当センタ内
		1	伝 送 速 度 50ボー タイプライタ 装置 126字種, 80字/行, 375字/分 設 置 先 当センタ内

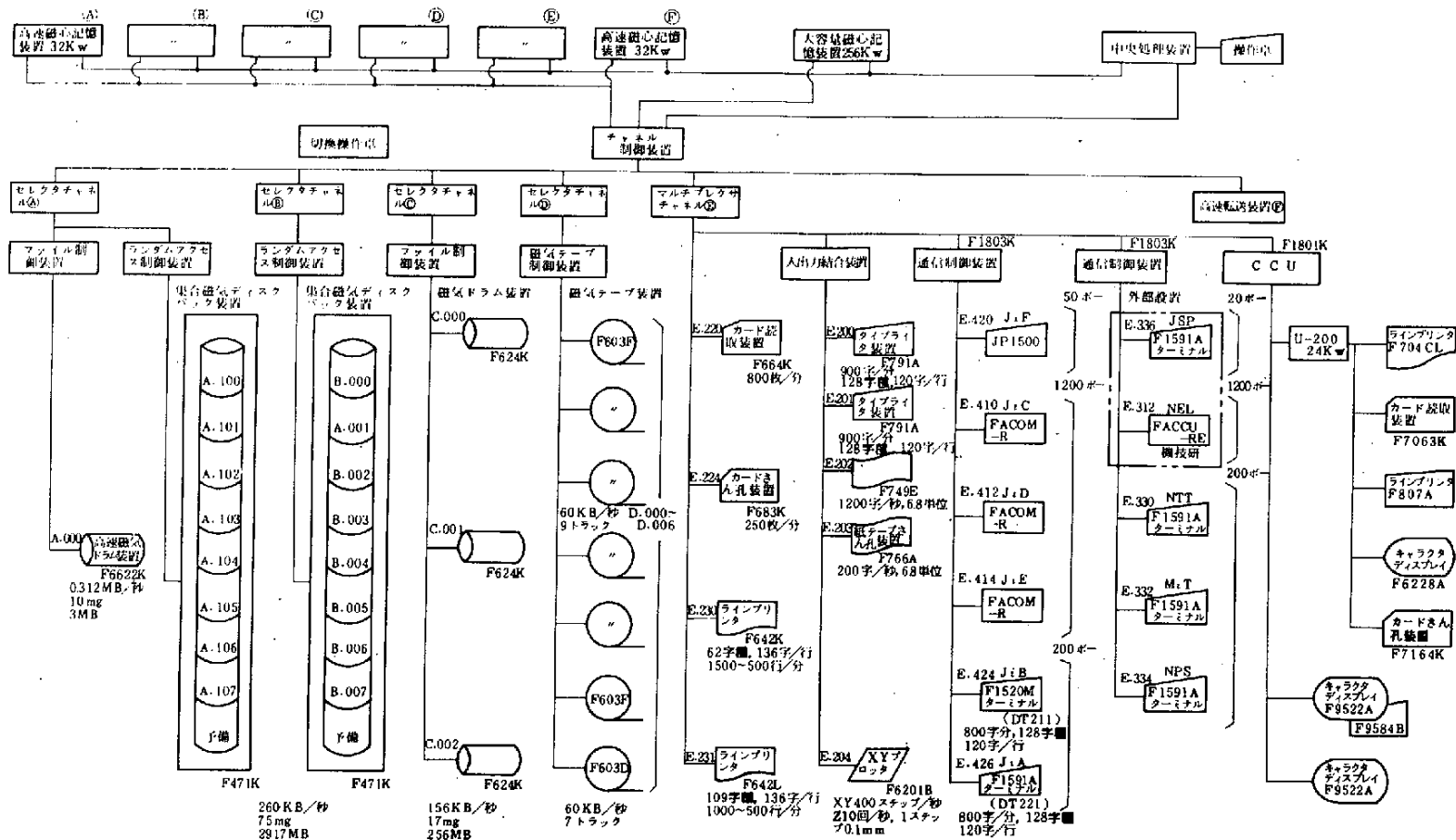
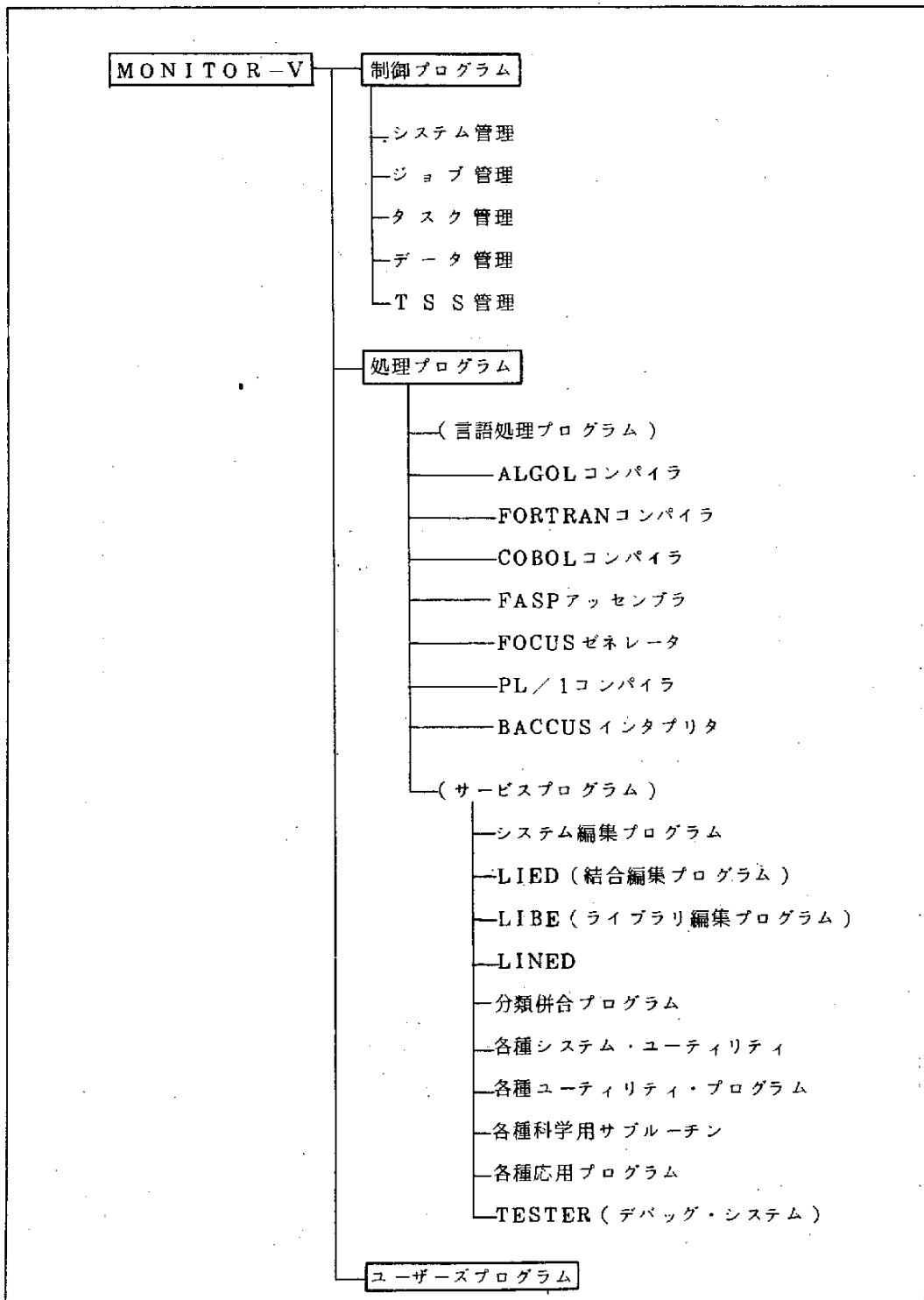


図 2-1 FACOM 230-60 ハードウェア構成

表2-1 FACOM230-60 ソフトウェア構成



(2) HITAC-8450ハードウェア諸元

装 置 名		台数	性 能 ・ そ の 他
主記憶装置	主 記 憶 装 置		393KB
外部記憶装置	集合ディスクパック装置	1	転送速度 312KB/秒 平均アクセス 60ms 容 量 29.17MB×8
	磁気テープ装置	6	転送速度 60KB/秒 記録密度 1,600BPI 記録形式 9トラック
	〃	1	転送速度 60KB/秒 記録密度 800BPI 記録形式 9トラック
入出力装置	カード読取装置	1	1,600枚/1,000枚/分
	(マークリーダー付)	1	1,470枚/750枚/分
	カードさん孔装置	1	100枚/分
	ラインプリンタ装置	1	16/110字種, 132字/行, 600/300行/分
	〃	1	16/110字種, 132字/行, 1,250/625行/分
	コンソールディスプレイ	1	112字種, 50字×20行
H-8811	グラフィックコンピュータ	1	ディスク, プロッタ付

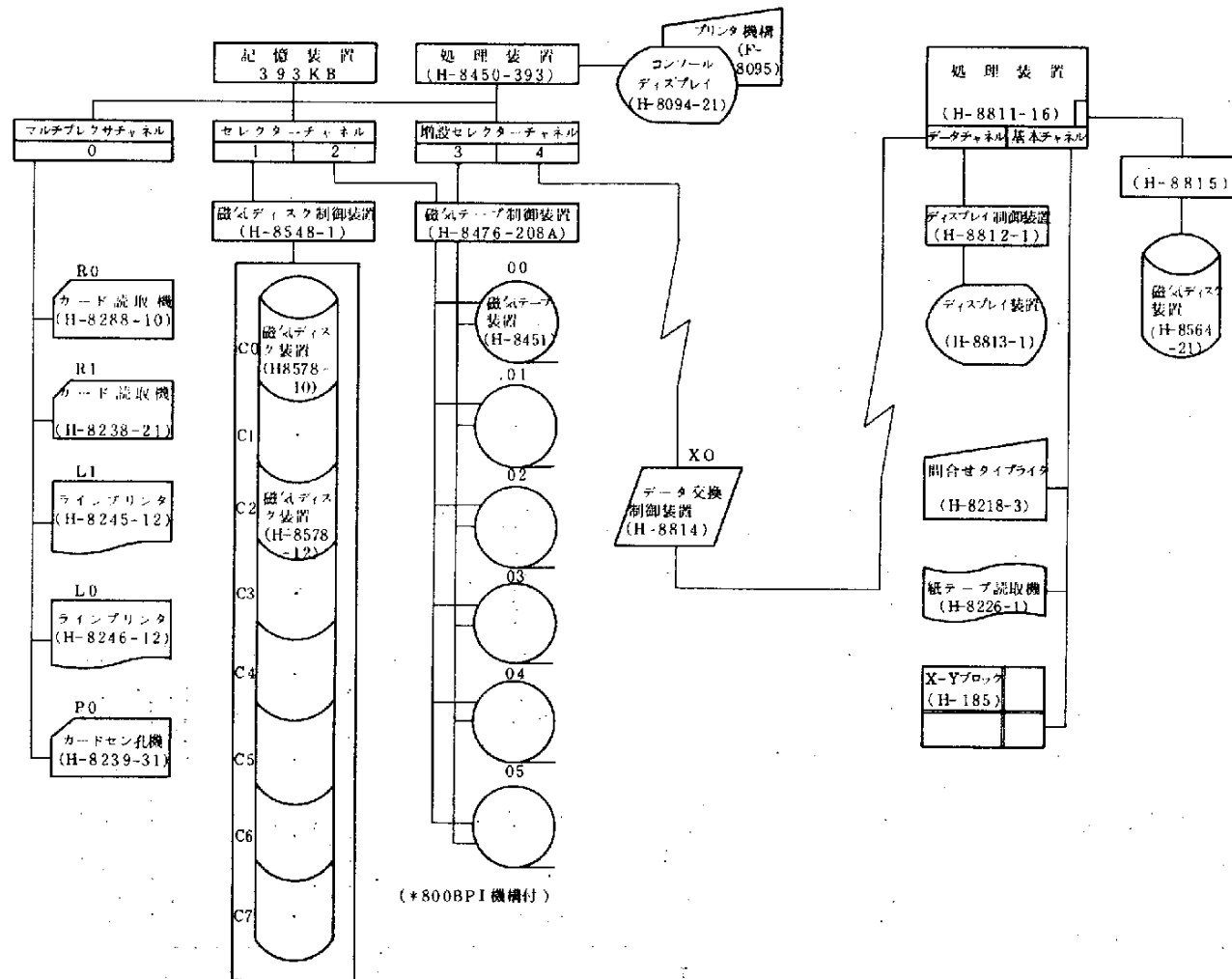
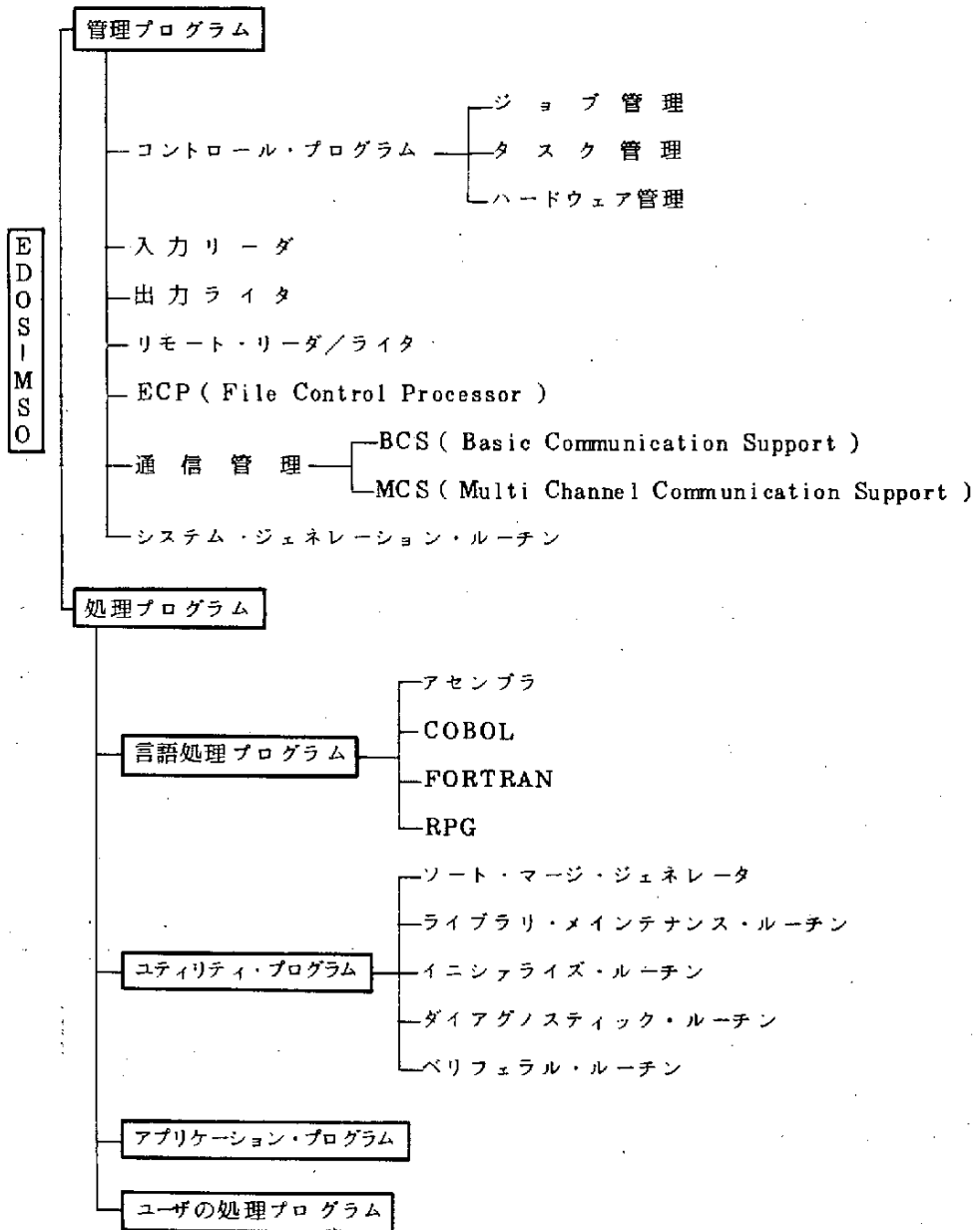


表 2-2 HITAC-8450ソフトウェア構成



(3) NEAC-2200-500ハードウェア諸元

装 置 名		台数	性 能 ・ そ の 他
主記憶装置			524KC
外部記憶装置	ディスク・パック装置	5	転送速度 203KC/秒 平均アクセス 8.5ms 容 量 9.6MC×5
	磁気テープ装置	6	転送速度 64KC/秒 記録密度 800/556CPI 記録形式 7トラック
入出力装置	カード読取装置	1	800枚/分
	カードさん孔装置	1	100枚/分
	ラインプリンタ装置	2	109字種, 132字/行, 420行/分
	タイプライター装置	1	64字種, 580字/分
端 末 装 置	学 習 端 末 装 置	1	
	NEACライター装置	1	96字種, 560字/分

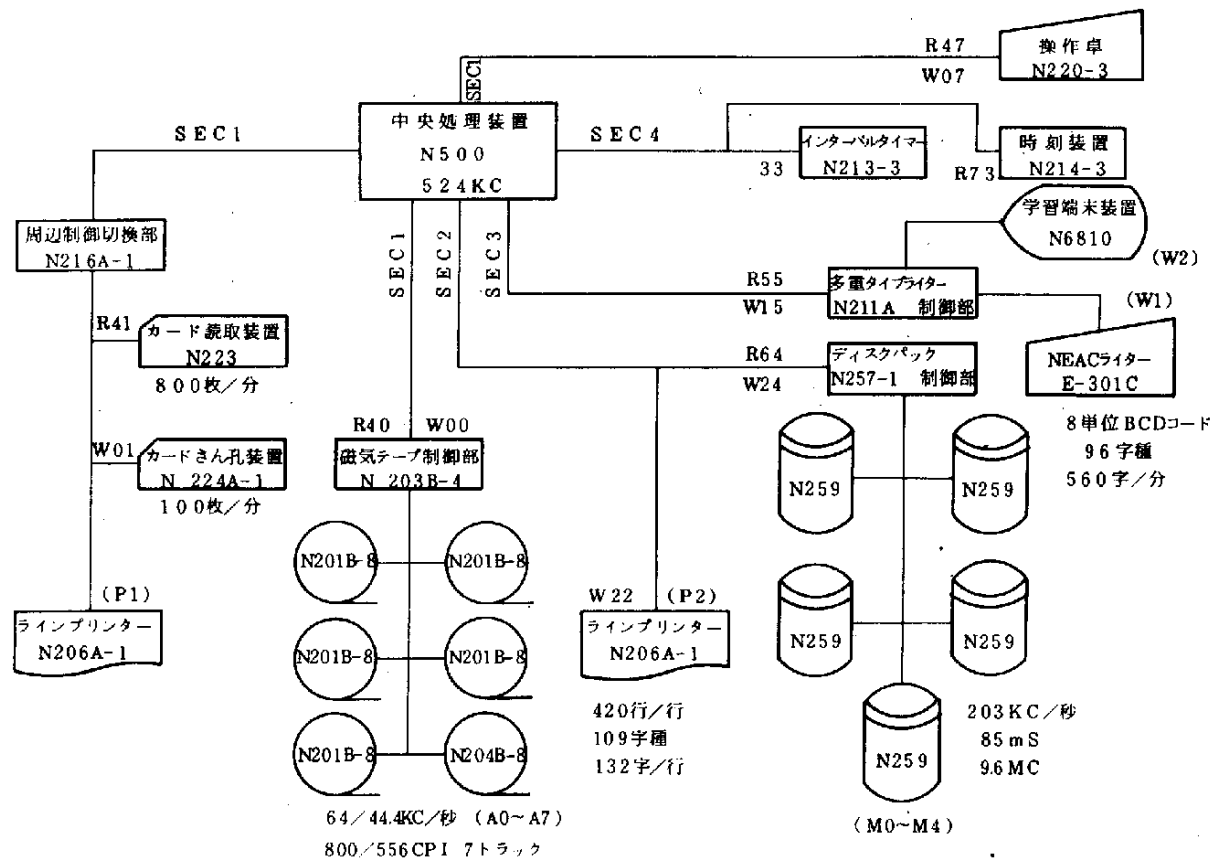
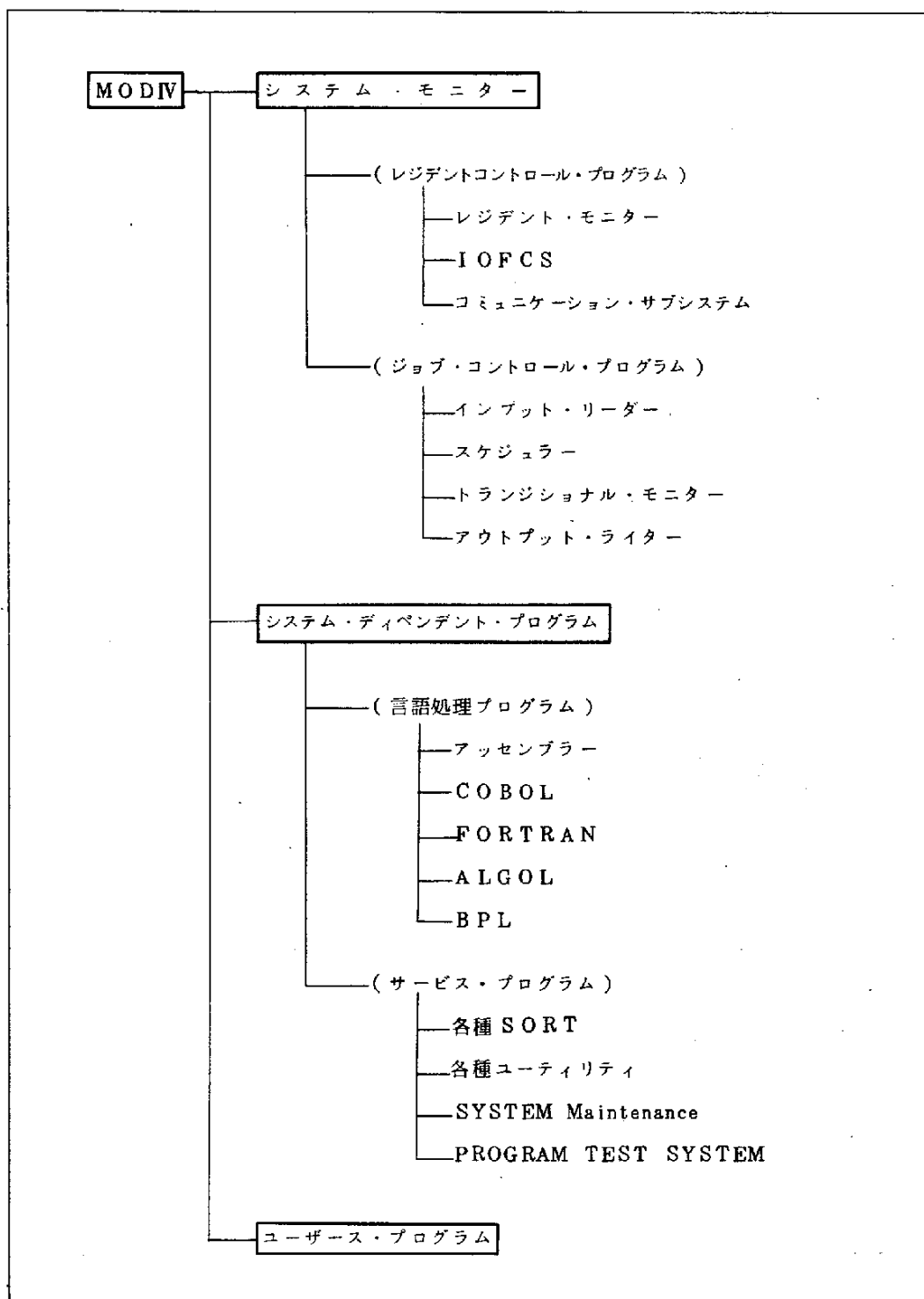


図2-3 NEAC-2200-500ハードウェア構成

表2-3 NEAC-2200-500ソフトウェア構成



2.2 結合方法とシステム構成

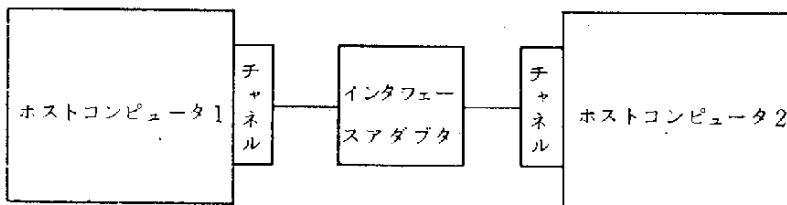
2.2.1 結合方法

前節で述べた三機種をホスト・マシンとして位置づけ、コンピュータ・ネットワークを実現するには、何らかの形でこれらを結合しなければならない。計算機間を結合する方法としては、以下の3種が考えられる。

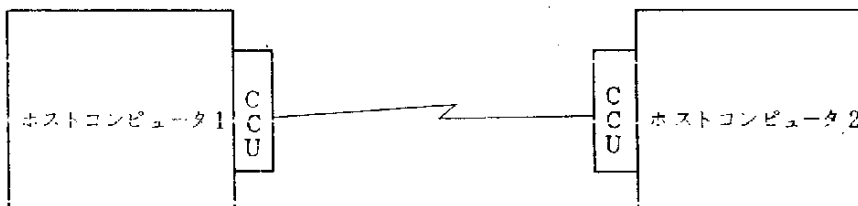
- ① ホスト・マシン間のチャネル同志を適当なアダプタを介し、直接結合する。
- ② ホスト・マシン間を高速通信回線により直接結合する。
- ③ メッセージの交換機能をホスト・マシンから切離し、専用の小型コンピュータと高速回線によりサブネットを構成しホスト・マシンはサブネット・ノードに結合する。

上記の各形態を図2-4に示す。

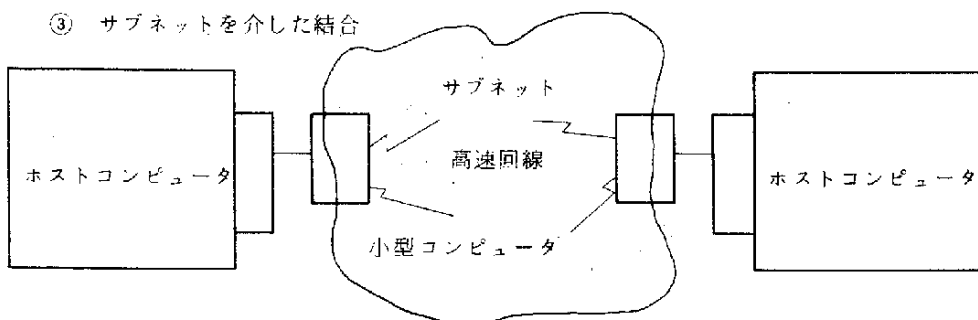
① チャネルによる直接結合



② 高速通信回線による直接結合



③ サブネットを介した結合



・図2-4 結合の形態

当センターのコンピュータ・ネットワークは以下のような理由から③の結合方法をとることとした。

(1) 異機種間インタフェースの調整

各ホストは独立した機能を果たす異機種のコンピュータであり、それらを直接結合するのは、ハード・ソフト両面のインタフェース上問題が多い。たとえば、ハードウェア上の問題として、各機種のチャンネル・インタフェースは、そのコントロール方式、信号線等にかなり大巾な差異があり、異なる二機種を結合するには、複雑な変換機構によるチャンネル・インタフェース・アダプタを必要とする。この場合、2コンピュータ間の許される距離は、通常50～60m程度以内であり、特殊なドライバ・レシーバを使用しても数百メートルが結合できる最大距離である。当センター内の三機種をコンピュータ・コンプレックスとして結合するだけが目的であれば、距離的には三機種とも同一フロアにあり問題はない。また、距離が相当ある場合でも、結合する事だけを考えた場合は高速の通信回線による直接結合の形態をとれば良いが、一方ソフトウェア上の問題を考えると、このようなチャンネル・インタフェース・アダプタに対するソフト・サポート、そのメッセージ・フォーマットのホスト毎の変換等が必要となり、(図2-5)先に述べたハード上の問題ともからんで、多数の異機種コンピュータを結合するのに直接結合の形態をとるのは望ましくない。

このような異機種コンピュータ間のインタフェース上の問題を解決するための一つの手段として図2-6に示すように、結合部に標準インタフェースを持つ小型コンピュータを配置し、各ホスト・コンピュータは単にその標準インタフェースとのインタフェースのみを配慮するようにすれば、ハード・ソフト上のインタフェースはとりやすくなるが、この結合に於て、インタフェース・プロセッサがダウンするとネット全体が動作不能となり信頼性がいちじるしく低下する。

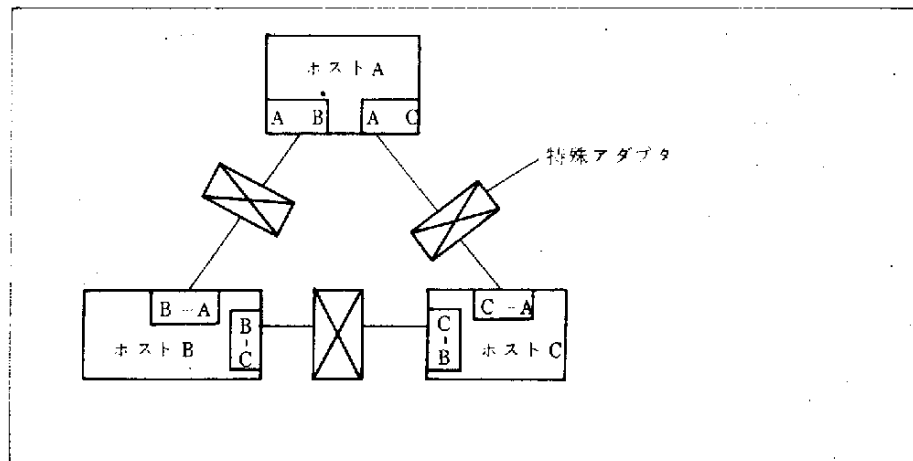


図2-5 直接結合

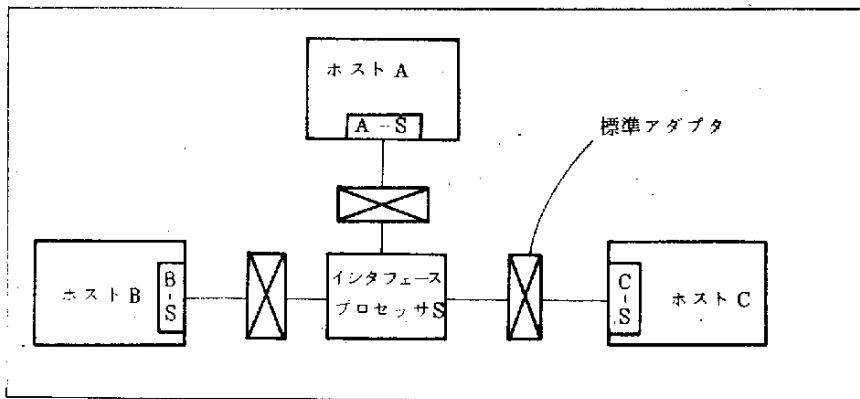


図 2-6 標準インタフェースを介した結合

(2) ホスト負荷の軽減

ホスト間を往来するメッセージの交換機能を各ホストが受持つことにすると、ホストは本来のジョブ処理以外にメッセージのハンドリング、交換という仕事を背負うことになる。トラフィック量が増大すればホストの負荷が大きくなり、ホスト内のジョブ処理に悪影響をおよぼすことになる。このため、メッセージの交換機能を専門に受持つ小型コンピュータを設け、そのメッセージ交換用コンピュータ群からなるネットワーク（サブネット）と、本来のジョブ処理に専念するホスト・コンピュータを独立させるよう機能分担をおこなえば、ホスト負荷は軽減されることになる。

(3) ネットワークの拡張性

前記(1)、(2)のように、ホスト・コンピュータとサブネットの機能を分離しておき、互いに独立性を保つことにより、新しいノードの追加によるサブネットの拡張および新しくホスト・コンピュータの追加あるいは他のネットワークとの結合等の拡張が比較的容易になる。

(4) 信頼性の確保

サブネットの結合に於て、いづれのノード間にも2つ以上のルートをもうける様結合することにより、ネットワーク全体の信頼性を確保することが出来る。ホスト同志の直接結合の場合に多くのルートをもうける事は経済性の面からもホストの負荷の面からも望ましくない。

以上の様ないくつかの問題を検討した結果 ARPA 型の IMP 群によるサブネット構成をとることとした。

2.2.2 システムの構成

前項で述べたように、当センターのコンピュータ・ネットワークは ARPA 型の IMP 群と高速回線からなるサブネットを介してホスト・コンピュータ群を結合することとした。

1.2 (頁) において、本プロジェクトは異機種コンピュータを結合しコンピュータ・ネットワークを形成する事、そのコンピュータ・ネットワークを用いて各種の新しい利用法を実験する事、実用化システムを目指し改良・拡張をおこなう事を基本的なねらいとしてしていると述べたが、この目標を達成するための最適なシステム構成を決定するにあたっては、以下のような諸点を考慮した。

(1) コンピュータ・ネットワークとしての基本的機能を有すること。

前述の基本的なねらいを満足させるためには、コンピュータ・ネットワーク・システムとしての基本的な機能が、ほぼ全て実験可能なシステム構成でなければならない。

たとえば、ホストとIMP間、およびIMP-IMP間のメッセージ・ハンドリング、IMPにおけるメッセージの交換機能、IMP間的高速伝送制御、各種ターミナルを結合したTIPとしての制御機能、ネットワークのパフォーマンスの測定機能などである。

(2) 当センター独特の利用法が実現可能であること。

1章で述べたCJPのように、複数ホストをコンピュータ・コンプレックス・システムとして使用するような当センター特有の利用法の実験に適したシステム構成でなければならない。

(3) 情報ネットワーク・システムのモデルとしての実験が可能であること。

1章で述べたように最近、各省庁で計画されている情報ネットワーク・システムの中核を占めるであろう。コンピュータ・ネットワーク下での集中型、分散型のデータベース・マネジメント・システムの実験が可能であること。

(4) ネットワークとして拡張性を有すること。

当センターの三機種のみを結合するのでは、実用効果を評価するには不十分であり、やはり外部機関のコンピュータをホスト・マシンとして追加接続、またはTIP端末、TIPの追加により外部からの利用を容易にするなどして、当然ネットワークを拡張し新しい利用法の実験をしていく事を前提としておく必要がある。

また、将来他のネットワークと結合するということも十分考えられるので、異なるサブネットワーク間でのインタフェースがとりやすくなるよう考慮しておく必要があろう。このようなことから、外部へ拡張する上での問題点もあらかじめ実験可能なシステム構成としておくことが望ましく、且つ拡張が容易で、拡張時のコストも比較的低いことが必要である。

(5) 従来、各機種で行なってきた作業が、何等の変更なしに引き続き稼動可能であること。

このような諸点を考慮して、最適として採用したシステム構成は図2-7に示すようなものである。

図2-7に示すように、JIPNETの構成要素はかなり多彩である。

まずサブネットは、メッセージ交換用コンピュータとしてNEAC 3200/50 小型コンピュータ3セットから成る3ノードの完全結合であり、通信回線は現在利用可能な最高スピードの全二重48 Kbps の高速ラインを使用している。

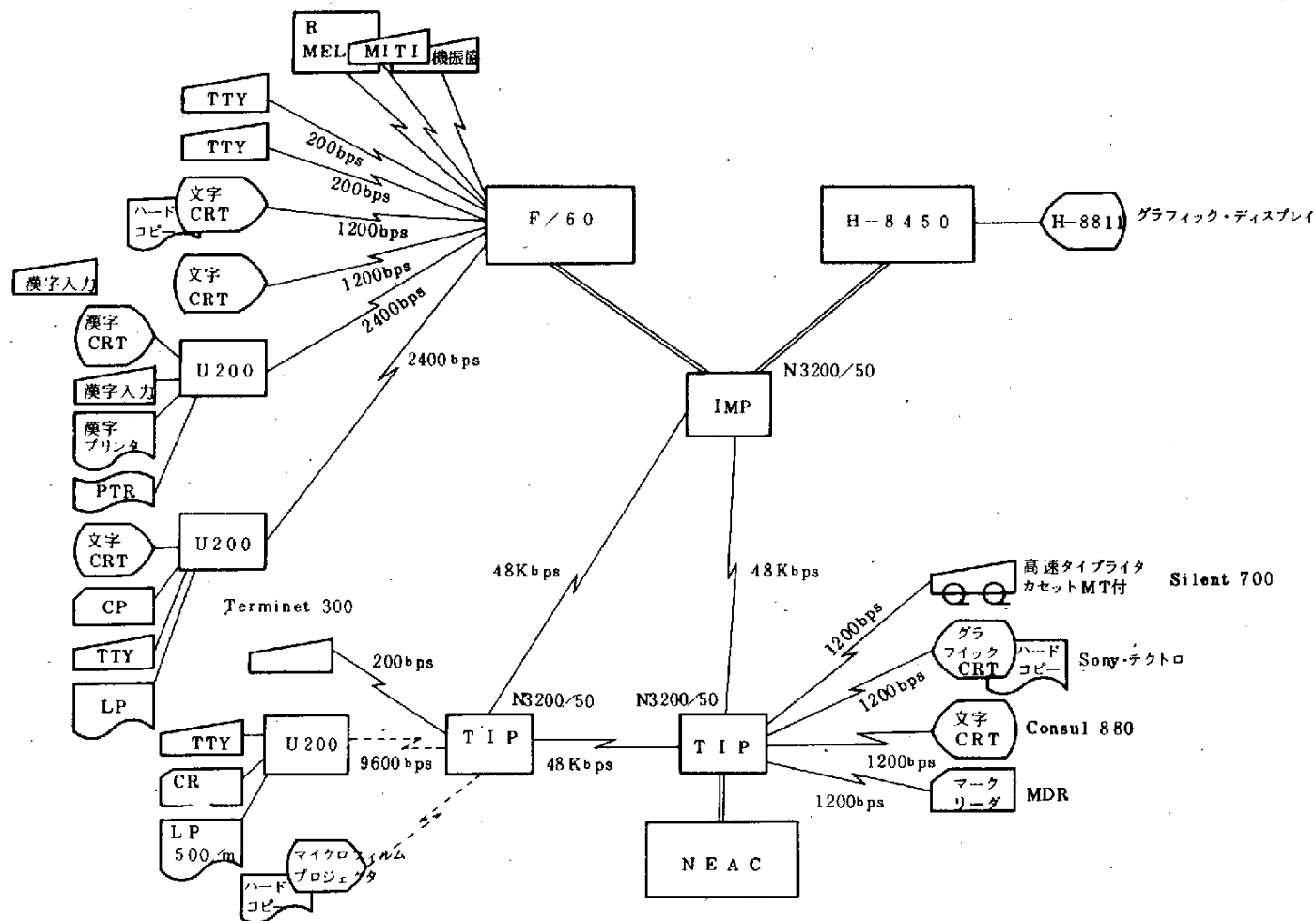


図 2-7 JIPNET コンピュータ・ネットワークの構成

各ホスト・コンピュータとIMP間はセレクターチャネルによる直接結合でありチャネル・インタフェース・アダプタを介して8 bitパラレルで1 Mbpsの半二重ラインで結合しており、F/60、H-8450間は大量の図形情報の転送を必要とするCJPジョブがアプリケーションとして予想されるので同一IMPに接続し、コンピュータ・コンプレックスとして効率良く働き得るよう配慮した。

また、残る2セットの小型コンピュータにはTIPとしての機能を持たせ、現時点で最新かつ低コストの多種の端末類を結合し、全ホストへの自由なアクセスを許し、多彩な実験的利用を計画している。TIPの1セットはNEACホスト・コンピュータにも接続されており、端末制御とホスト接続のIMP機能を同時に果すTIP、ホスト・コンピュータを接続しないTIP、2ホストを接続しているIMPと各種の結合形態を実現し、各状態での実験が可能になっている。

なおインハウスの結合のみを目的とするならば、IMP間を48Kの通信回線で結合するよりも、これもまたチャネル結合による、より高速な結合が可能であるが、外部への拡張も考慮し、IMP間の距離がより長い場合のモデルとして構成したため、あえて48.Kbpsの通信回線による結合方式をとった。

2.3 IMP, TIPのハードウェアと各種端末

2.3.1 IMP, TIPの機種選定

コンピュータ・ネットワークのサブネットの能力に最も影響をおよぼすのは、IMP用コンピュータの性能と回線スピードである。本項では、まずJIPNETのIMP, TIPとしての機種選定にあたって考慮したハードウェア性能について述べる。

(1) サイクル・タイムが速いこと。

サイクル・タイムが各種CPU命令の実行スピードに与える影響の度合は、機械語命令の実行制御方式によって異なるが、ごく基本的なCPU性能評価尺度としては、やはり有力な比較項目である。

(2) 実効命令の実行スピードが速いこと。

ハードウェア性能を考える場合は、前記のサイクルタイムの速さ、命令の実行スピードの比較ということでは不十分であり、そのハードウェアの使用状態に最も近い状態での命令の組合せによる比較が必要なのは当然である。

このための評価尺度として表2-4に示すような各種命令への荷重をかけたリアルタイム・ミックス値を計算して比較した。このリアルタイム・ミックスはオンライン・リアルタイム用のコンピュータの性能評価に、最近使用されているものである。

表 2-4 リアルタイム・ミックスの荷重

	Instructions	Weight
1	load	22%
2	store	22
3	add / sub.	11
4	conditional jump	9
5	unconditional jump	8
6	compare	6
7	logical	6
8	shift	6
9	index increment and test	5
10	loader modify index	5

(3) メモリ容量が大きく、拡張性があること。

3章で述べるように、IMPは蓄積交換の機能を果たすもので、出入回線用のバッファ、処理待のペケット用のバッファ、ホストとのメッセージのやり取りのためのバッファ等各種のバッファ・エリア用としてかなりの容量のメモリ・スペースが必要となる。補助記憶装置を利用するのがこの問題の解決法の一つではあるが、信頼性の低下につながるような要因はできるだけ排除しなければならず、高速の蓄積交換機能を果たすためにも主記憶ですべて処理できるように考えるのが望ましい。また、TIPに接続する端末はできるだけ多種多様なものを随時増設して、新しい利用法を実験していく予定であり、それらの端末制御用プログラム等の拡張スペースを考慮せねばならず、かなりの余裕を持ったメモリー容量でなければならない。

ARPAの例でも Honeywell 516, 316 バージョンのサブネットでは、サブネットのスループットは結局バッファ用メモリー・スペースが最も大きい制限要素であると報告されており、充分のスペースを得ておく必要がある。

(4) プログラム割込をはじめ、多様の割込機能を有すること。

IMPのプログラム構造は、高速回線、低速回線、ホストとのインタフェース・アダプタ、タイマ等からの割込によって起動される各種のタスク群からなり、これらのタスク群が互いにかみ合いながら処理が進行するような構造になる。したがってこのような割込の制御機構はハードウェアでかなりの機構を有しているとともに、たとえば割込の許可・禁止、優先順位の変更等がプログラムによって柔軟にコントロールできることが必要である。さらに、プログラムによるタスク・コントロールの切換をハードウェア割込と同様に、プログラム割込として備えている事が、タスク・コントロールを容易にする為には是非必要である。

- (5) チャネルによるメモリ・サイクル・スティーラが、CPU 実行スピードに与える影響が小さいこと。

通常、入出力チャネルは、CPU 命令のメモリ・サイクルをスティーラする形で動作して並行処理をおこなっているため、チャネルのサイクル・スティーラの割合が大きいと、それだけ CPU 命令の実行が遅らされることになる。入出力トラフィック量がそれほど多くない場合は、さほど問題にはならないが、本システムのように、IMP に接続されている 2 台のホスト・コンピュータが、2 チャネルを専有して高速大量データを転送しあっている状態がある場合には、CPU 命令実行スピードが極端に低下し、たとえば高速回線からの割込に対し、バッファを割付けるタイミングが間にあわず、メッセージを失なうようなことにもなりかねない。このようなことから、チャネルによるサイクル・スティーラの割合が極力小さいことが要求される。

- (6) 高速回線、低速多重回線の制御機能を合わせ持つこと。

本項の冒頭でも述べたように、サブネットの性能を左右する主要な要素の一つは、IMP 間通信回線のスピードである。現時点では日本の場合 4800 bps が公社回線としてサービスされている最高スピードであるが、国産機各メーカでも、来る 50 年度からの公社の高速データ網計画に間に合うべく 48 Kbps の通信制御装置、モデムを開発しており、本ネットワークでも当初は構内の専用回線としてこの 48 Kbps 高速回線を利用できることが必要である。

また、TIP の各種端末をコントロールするための通信制御装置は、多種類のスピードをコントロールでき、一括管理できる多重回線用通信制御装置であることが必要である。さらに TIP の端末制御プログラムの処理を簡略化すると共に、メッセージ・ハンドリングの為に CPU 負荷を軽くするために、メッセージ組立を通信制御装置が受け持つ、いわゆるメッセージ・モードの制御形態であることが望ましい。

- (7) 信頼性が高いこと。

すべてに優先する条件としてまず、サブネットの高信頼性を保つために IMP 用コンピュータの信頼性はできるだけ高い必要がある。

以上のようなハードウェア性能を評価項目として、国産の数種の小型コンピュータを比較した結果の一部を表 2-5 に示す。

表 2-5 各社 IMP 用小型コンピュータ性能比較表

機 種 比較項目	機 種 A	機 種 B	機 種 C	機 種 D
サイクル・タイム	0.96 μ s	1.6 μ s	0.65 μ s	0.6 μ s
リアルタイム・ミックス	1.92	3.2	2.99	1.46
アダプタ・スピード	1 M bps DMA	1 M bps 高速DMC	2.88 M bps CCA	4.76 M bps セクタ・チャンネル
C P U 効 率 (実効リアルタイムミックス)	87.5% (2.18)	60% (5.33)	64% (4.67)	64% (2.28)
各アダプタを1M bps にした C P U 効 率 (実効リアルタイムミックス)	87.5% (2.18)	60% (5.33)	87% (3.43)	92% (1.59)
最大メモリ・サイズ	32 KW	16 KW (32 KW)	32 KW	65 KW
プログラム割込	△ 改造にて可	△ 改造にて可	○	○

以上のようなハードウェア性能のはかに、実際の機種選定にあたっては、次のような項目を考慮しておく必要がある。

(1) 各ホスト・コンピュータとのインタフェース

各ホスト・コンピュータと IMP 間は適当なアダプタを介しチャンネル間接続を行うのが適当であるとしたがホストと IMP 間の標準インタフェースを実現して、各ホストのチャンネル部分までのインタフェース機器を IMP 用コンピュータ・メーカーが併せて開発して提供することが望ましい。

(2) コスト

この種のネットワークのコストを軽減する1つの要因として IMP コンピュータが安価である事が望ましいのは当然である。特に、近い将来、外部システムとの接続をおこなう際に、接続用の機器が低コストであることは、拡張の可能性のかかなり大きな要因となる。

(3) 協力、支援体制

本プロジェクトのように複数のメーカーが対象となるシステムを開発していく際は、メーカーのハード、ソフト両面での当センターに対する協力、支援体制、およびメーカー間での協力体制が重要になる。

IMPの機種選定の結果が他のメーカーの協力度にどれだけ影響するかを事前に察知してトータルでの協力支援体制が最大の効果を生むように考慮することが必要である。

(4) 納 期

当センターの場合は48年度中に設置、結合テストが完了することが必要である。

(5) 実 績

すでに他社コンピュータとチャネル結合の経験を持ち、ネットワークに関しても実績があるメーカーが望ましい。

このような総合的な評価により、IMP、TIP用コンピュータとして、日本電気のNEAC 3200/50システムを採用した。

なはこの機種の原型はHoneywell DDP 516であり、ARPAネットワークのIMPとして採用されている。

2.3.2 IMP、TIPのハードウェア構成

前項で述べたように、JIPNETのIMP、TIP用コンピュータとしては、NEAC 3200/50を採用することとしたが、本項ではIMP、TIPのハードウェア構成について簡単な説明を加え、その中でハードウェア構成を決定するに際し問題となった点についてはやや詳細に述べることとする。

図2-8は、NEACホストに接続されているTIPのハードウェア構成を示したものである。以下に2-8図内の番号にしたがって説明する。

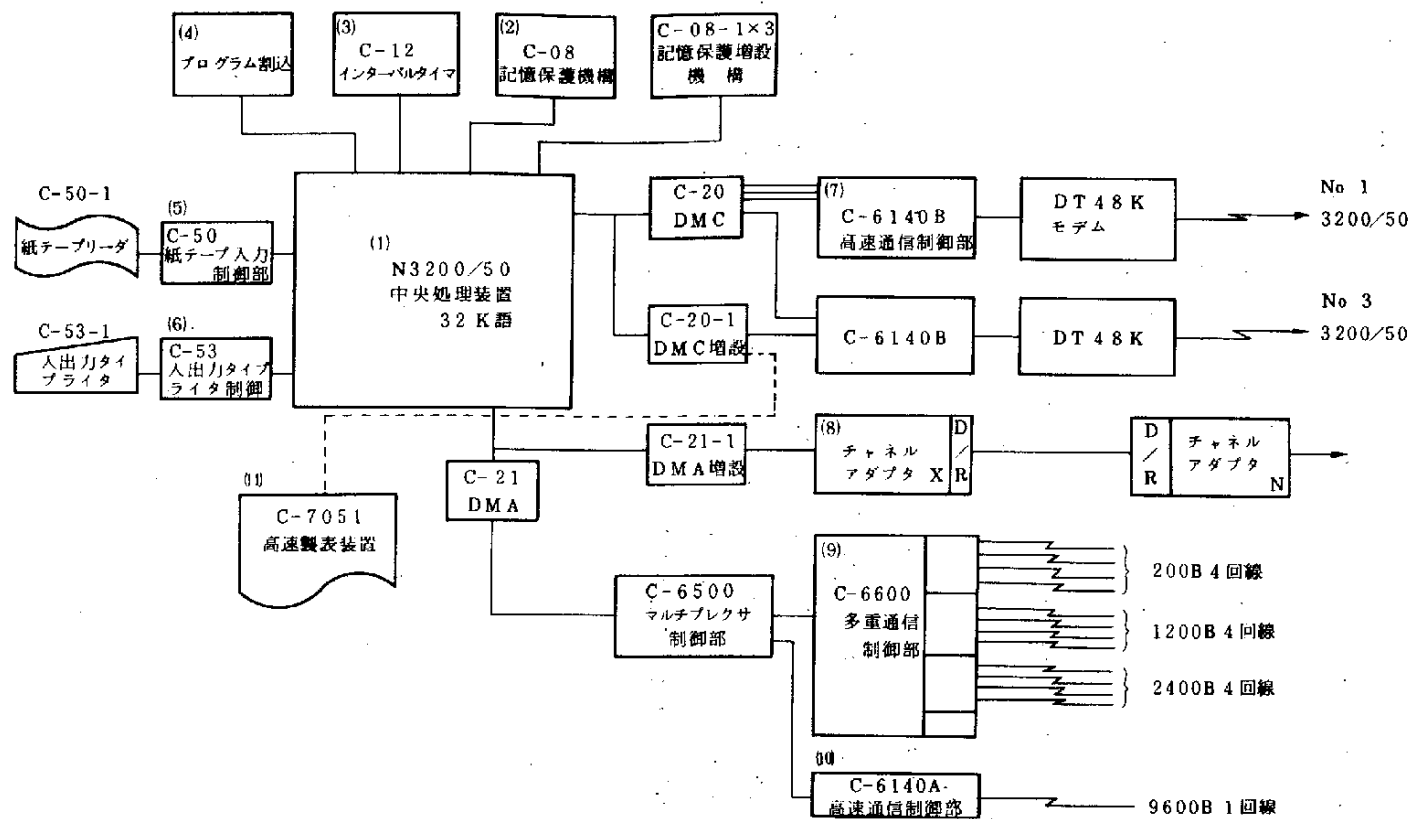


図 2-8 TIP のハードウェア構成

(1) 中央処理装置

NEAC 3200/50は、Honeywell社のDDP 516 とほぼ同様のマシンであり、中央処理装置の性能は表2-5に示すようなものである。

表2-5 中央処理装置の性能

項 目	NEAC-シリーズ3200モデル50 中央処理装置
語 長	16ビット+1パリティ・ビット
データ形式	命令語：1語 データ語：1語、2語
命令形式	1アドレス方式
命令数	基本命令72種 付加命令16種
数 値	2進、負数は2の補数形式
主記憶容量	8K語、16K語、24K語、32K語
記憶速度	サイクリタイム 0.96 μ s/語
パリティ・チェック方式	奇数パリティ チェック
アドレス方式	1アドレス、多重レベル間接アドレスおよびインデックス修飾
演算の種類	2進演算(単精度、倍精度)、論理演算、 入出力制御、プログラム制御、割込制御
演算速度	加減算 1.92 μ s 乗 算 5.28 μ s ※ 除 算 10.56 μ s ※
入出力制御	基本チャンネル 1個 DMCチャンネル※最大16個 DMAチャンネル※最大 4個 入出力制御部接続台数 最大14個
割込み機能	基本割込 1レベル(16ライン) 優先割込 4~48レベル※
インターバル タイマ	タイムベース 250 μ s, 500 μ s, 1ms, 16.7ms(60Hz), 20ms(50Hz) ※
記憶保護機構	停電に対する保護 プログラム エラーに対する保護

※ 付加機構

JIPNETのTIPの場合前項のハードウェア条件でも述べたように主記憶容量がかなり大きいことが望まれるので、NEAC 3200/50としては最大容量である32K語の主記憶を最初から装備しておくこととした。

乗算、除算の機構はオプションとなっており、IMPのプログラムの性格上、乗除算の使用頻度はあまり多くなるとは思われないのでつけていない。

(2) 記憶保護機構

記憶保護機構は、N 3200の場合セクター(512w)単位にメモリ内容をプロテクトする機構で、メモリ・プロテクションとしては、かなり柔軟性に富んだ指定をおこなうことができるようになっている。IMPプログラムのように多数の小タスクに分割されたルーチン群が複雑にからみあって実行されるような場合のディバグ時には非常に有効であろう。また、電源異常が検出されたら既定番地に割込む電源異常割込の機能もあるのでこの機能を利用し、IMPプログラムを破壊せずクリーン・ストップさせることができるし、適当な中断処理をある程度おこなうことができる。

(3) インターバル・タイマ

インターバル・タイマは、特定番地の内容を自動的に+1するもので、タイム・ベースとして250 μ s, 500 μ s, 1ms, 16.7ms, 20msのいずれかが指定できるようになっている。タイマーによるインクリメントが16ビットをオーバーフローすると、インターバル・タイマー機構はプログラムに割込をかけるようになっている。

IMPのようなリアルタイム・システムの場合、各タスクの進行制御にはこのインターバル・タイマーの機構は是非必要となるものである。タイム・ベースは小さい程良いというのではなく、メモリへの書込にメモリ・サイクルをスティールするので、適当な値を選定する必要がある。たとえば250 μ sのタイム・ベースの場合のサイクル・スティールの割合は
$$\frac{3(\text{サイクル})}{250} = 1.2\%$$
となる。

IMPとしてのタイマーの使用はARPAの例では以下のような場合であり、いずれもそれほどの精度は必要としない。

- fast time out (25ms)

ルーチン再起動用

- middle time out (100ms)

ACK(確認信号)の返ってこないパケットを再送する。

- slow time out (500ms)

回線の状態監視、ルーティング・テーブルの更新等

また、このインターバル・タイマを、IMPプログラムのタスク走行時間の測定に利用し、IMPのパフォーマンス向上の役に立てようとも考えたが、各タスクは非常に小さいルーチン

からなっており、その走行時間を $250\mu\text{s}$ でも正確に把握するのは難しく、そのための使用をやめた。

このような理由からタイム・ベースは 1ms としている。

(4) プログラム割込

IMPプログラムは、各種の割込によって起動されるタスクの集りとして作成されるが、これらの実行はあるタスクの終了後、他のタスクにコントロールを移すような場合が多い。

このようなタスク・スイッチのためには、プログラムで、ある命令を実行すると特定番地に割込むようないわゆるプログラム割込による割込命令があると便利である。N3200/50には、このような機能が装備されておらず、メーカーの技術者とも相談して、次のような形でこのプログラム割込を実現した。

N3200には標準I/O以外の外部機器を接続し、その外部機器を制御するためにOCP (Output Control Pulse) 命令およびSKS (Sense 状態検出) 命令を付加することができるようにになっている。

OCP命令は指定されたOCPラインに対し、CPUから信号を送り出すもので、SKSは指定された要因のステイタス状態に応じてCPUに信号を送るもので、SKS命令によりステイタスを検知することができるものである。

これらのOCP/SKSは16種ずつ増設可能で、ファンクション・コードの指定によりOCP00~OCP17₈、SKS00~SKS17₈が選択できるようになっている。

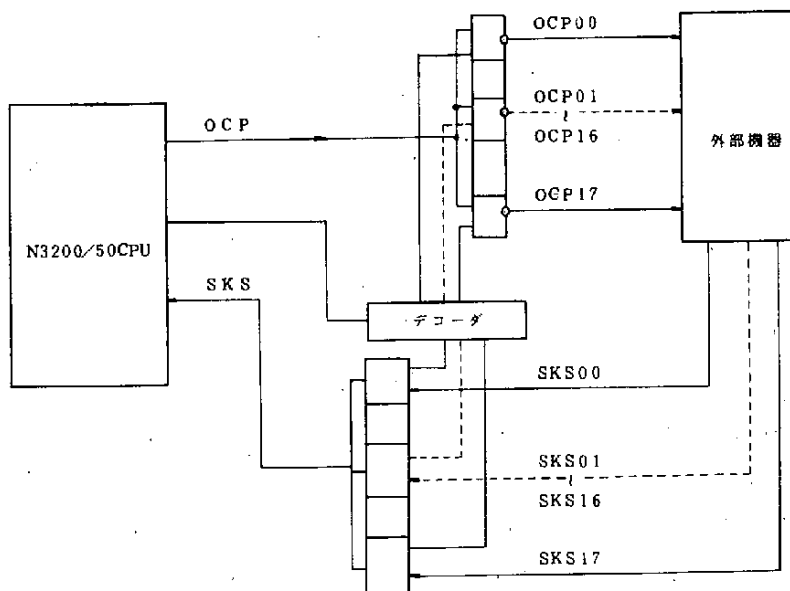


図 2-9 OCP/SKS 付加機構略図

また、N3200/50 には最高 48 レベルまでの優先割込ラインを付加し、外部からの信号によって CPU に優先割込を起し、各ライン毎に割当てられた特定番地へジャンプすることができるようになっている。(図 2-10 参照)

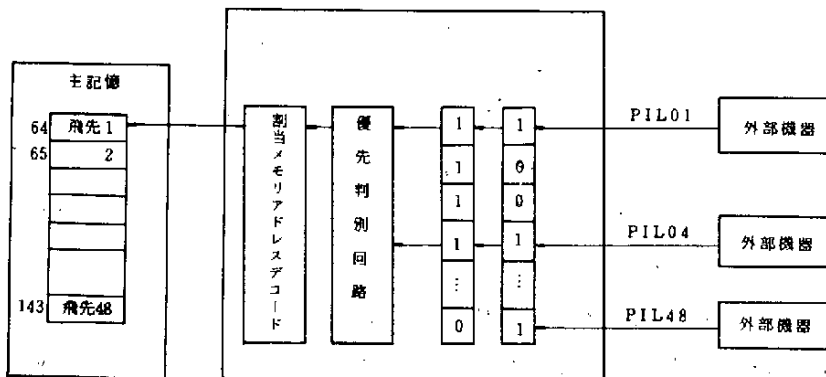


図 2-10 優先割込付加機構略図

これらの二つの付加機構を利用し、CPU から OCP 命令を発することにより、あたかも外部機器に何等かの指令を出し、その外部機器制御部が CPU に対し優先割込を起すような機能を持つ、CPU に対する割込命令 (CPU 命令) を実現することができる。このための改造は、単純に言えば、OCP ラインと PIL ラインを結び、その OCP ラインにより PIL ラインに信号がでるようにすれば良く、非常に簡単に済むことになる。

実際の改造では、OCP00~OCP03, PIL01~04 と各々 4 本分のラインを接続し、対応するラインをループ結合し OCP00~OCP03 を使い分けることにより 4 レベルのプログラム割込が可能のように考慮されている。

(5) 紙テープ・リーダー

プログラム作成時および、IMP プログラム完成後も最初のプログラムのローディング用に高速紙テープ・リーダーを備えた。読取速度は 300 字/秒である。

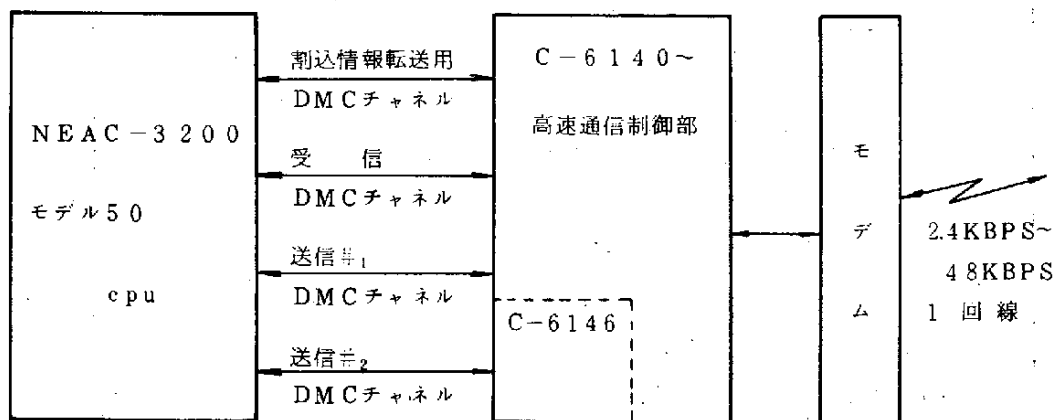
(6) 入出力タイプライタ

IMP オペレーションのためにコンソールとして入出力タイプライタを備えた。印字速度は 10 字/秒で、同速度の紙テープ・リーダー、パンチがついている。

(7) 高速通信制御部

IMP 間を結ぶ高速通信回線の通信制御をおこなう機構であり、DMC チャネルに接続され、ハイレベル・データリンク通信制御手順に沿った通信を一回線収容することができる。

高速通信制御部の構成を図 2-11 に示す。



注、破線部はオプションを示す。

図2-11 C-6140~高速通信制御部構成図

表2-6は高速通信制御部の機能仕様の主なものをまとめたものである。

表2-6 高速通信制御機能仕様

種 別	機 能
① 収容回線数	全二重一回線
② 回線種類	電話専用回線 電話交換回線
③ 通信速度	2.4 Kbps ~ 48 Kbps $2.4 \text{ Kbps} \leq C-6140A \leq 20 \text{ Kbps}$ $20 \text{ Kbps} < C-6140B \leq 48 \text{ Kbps}$
④ 通信方式	全二重または半二重 (ハードウェア機能は全二重)
⑤ 同期方式	同期式 (1個以上のフラグ・シーケンスによる)
⑥ 使用符号単位	8単位符号
⑦ ビット送出順序	低位ビットより送出
⑧ 誤り検出	受信BCSチェック・エラー 受信オードフロー・エラー 受信フォーマット・エラー 受信アドレス不連続(バッファ・チェイン時) 送信アンダフロー・エラー

種 別	機 能
⑨ 制御符号	制御符号としては指定せず Fシーケンス, 放棄シーケンス受信で割込む
⑩ CPUとの接続	DMCチャネル3本使用 割込情報転送用(1本) 受信情報転送用(1本) 送信情報転送用(1本) 他に送信側に1本追加可能
⑪ 回線の制御	モデム信号CD(キャリア検出), DR(データ・セットレディ)の監視 CS(送信可)によるデータ送出の制御 CI(呼出表示), DRオンによる着信検出 ER(端末レディ), RS(送信要求)のオンオフ

対象とする情報のフレームの基本構成は以下のようになっている。

F, A, C, (情報フィールド), BCS, F

F: フラグ・シーケンス 8 bit (01111110)

A: アドレス・フィールド 8 bit

C: 制御フィールド 8 bit

情報フィールド $8 \times n$ bit

BCS: ブロック・チェック・シーケンス 16 bit

情報フィールドは必要な場合のみ使用され $8 \times n$ ビットより成る (n : 任意の正整数)

BCSは, A, C, (情報フィールド) から成るビット列を生成多項式 $X^{16} + X^{12} + X^5$

+ 1 で割った剰余で BCS の付加, チェックはハードで自動的におこなう。

放棄シーケンスは 01111111 の送信, 受信によって表わされる。

情報フィールド内のビット・レベルのトランスペアレンシーを保障するため A, C, (情報フィールド), BCS 内のビットについて次のような操作をハード的に施している。

送信: 5 個連続した "1" を送出した直後には必ず "0" を 1 個挿入してデータを送出する。

(Fシーケンス, 放棄シーケンス・パターン送出時は除く)

受信: 5 個連続した "1" を受信し, 次に "0" を受信した場合その "0" を削除し, その結果の情報に対して基本フレームの構成を適用する。

このほか, 送信については送信チャネルの切換により, 送信バッファのチェイニングが自動的に行われ, 受信についても 2 バッファまでのバッファ・チェインが使用可能である。

(8) チャネル・アダプタ

IMPとホスト・コンピュータを結合するためには、IMPのチャネルと、ホストのチャネルのインタフェースをとるためのチャネル・インタフェース・アダプタが必要となる。インタフェース・アダプタの機能に関しては次節で詳しく述べるので、ここではIMP、CPUとの結合部について考慮した点を記しておく。

インタフェース・アダプタの平均データ転送スピードは、次節で述べるようにF60ホスト、H-8450ホスト間でのコンピュータ・コンプレックス的な使用に耐え得るよう1Mbps程度とした。

このスピードでインタフェース・アダプタが動作している時のメモリ・サイクル・スティールの割合は、DMCチャネルを使用すると1w転送に4メモリ・サイクル要するのでN3200/50のメモリ・アクセス時間を約1μ秒として

$\frac{1 \text{ Mbps}}{16 \text{ Mbps}} \times 4 = \frac{1}{4}$ で1チャネルあたり25%となる。これでは、IMPホスト間の転送によるサイクル・スティールの割合が大きすぎ、2.3.1項(4)(頁)で述べたように他への影響が大になるので、1wの転送に要するメモリ・サイクルが2メモリ・サイクルであるDMAチャネルを使用することとした。しかし、DMAチャネルのメモリ・アクセス優先度はハードウェア的な固定結合ではDMCのそれよりも高位にあり、IMP・ホスト間の転送中48Kbps 高速通信回線モデムからのデータが確実に受信できるよう注意する必要がある。

たとえば、割込要因の分析、モデムからの割込に対する前準備(バッファのアロケート等)はできるだけ短いステップでコーディングする必要があり、その処理順序等にも十分な配慮が必要である。

(9) 多重通信制御部(MLC: Multi Line Controller)

TIPで各種端末を制御するために、下図のような構成の多重通信制御部(MM-MLC: Message Mode Multi-Line Controller)を用いる。

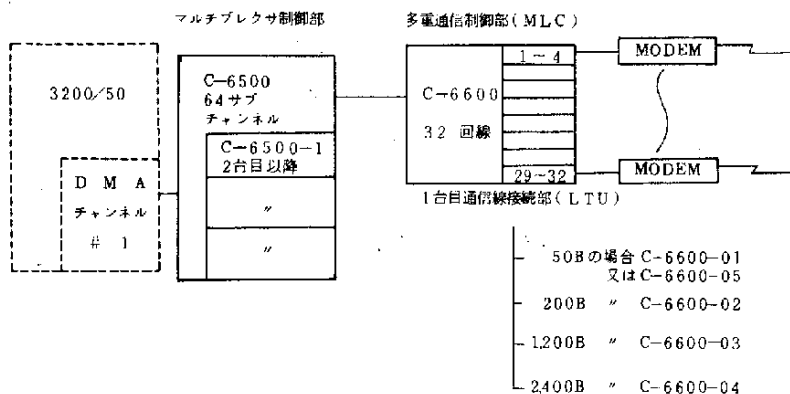


図2-12 MLCの構成路図

MLCのハードウェアとしては全二重128回線までを制御することが可能であるが、本TIPには200 bps 4回線、1200 bps 4回線、2400 bps 4回線の12回線を装備している。

表2-7にMLC部の機能仕様の一部を記す。

表 2-7 MLC部機能仕様

種 別	機 能
① 収容回線数	最大全二重32回線 MLC部は32回線ごと、LTU部は4回線ごとに増設でき、フルシステムで128回線まで収容できる。ただし、1200 bps、2400 bpsはあわせて32回線までしかとれない。
② 回線種類	電話品質専用線 オプションで電信回線も収容可
③ 通信速度	50 bps、200 bps、1200 bps、2400 bps
④ 通信方式	全二重または半二重方式 (ハードウェア機能としては全二重である)
⑤ 同期方式	調歩同期式(50 bps、200 bps、1200 bps) 同期式(2400 bps)
⑥ 使用符号単位	5、6、8単位符号 (2400 bpsは8単位のみ)
⑦ ビット送出順序	低位ビットより送出
⑧ 誤り検出	調歩同期エラー 受信オーバーラン 奇偶垂直パリティ・チェック
⑨ 制御符号	5、6、8単位符号をMLC部1台につき最大10種類まで、4種の組合せが任意に指定可能
⑩ CPUとの接続	N3200/50 DMA NO1 チャンネル
⑪ 回線の監視	CS(送信)によるデータ送出の制御のみ
⑫ 収容タイプ数	制御符号、同期方式、使用符号単位からなる回線のタイプがMLC部一台につき4タイプまで選択可

次にMLCの動作概要を簡単に説明する。

MLCはDMAのNO1チャンネルを介し、直接N3200の主記憶とデータ伝送を行なう。あらかじめ指定された主記憶内には、データ入力用・開始アドレスと終了アドレス格納エリア、およびデータ出力用・開始アドレスと終了アドレス格納エリアの4種類のメモリ・セルが1回線ごとに指定される。

MLCでは、CPUのプログラムより指定されたこのアドレスに従ってデータの入出力動作を行なう。開始アドレスは、1語(2キャラクタ)のデータを入出力するごとに更新され、開始アドレスと終了アドレスとが一致した時に中央処理装置へエンド・オブ・レンジを示すステータス・ワードを転送後、割込を起し、新しいバッファ・エリアの指定を要求する。

中央処理装置とのデータの流れを図2-13に示し、送受信動作の概略図を図2-14に示す。

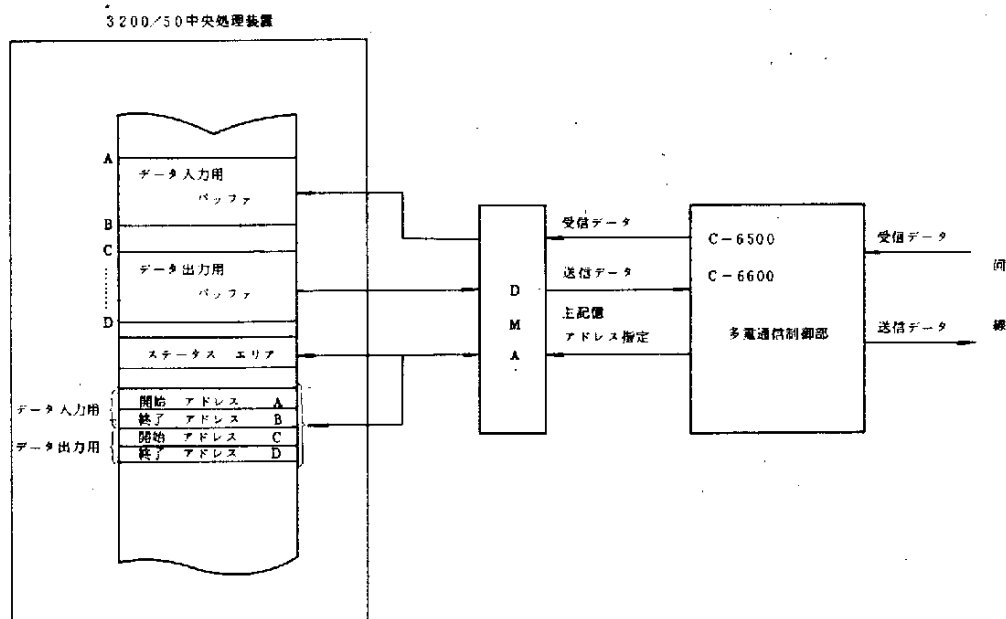
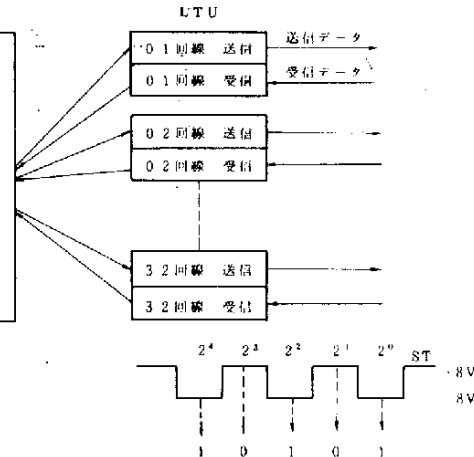
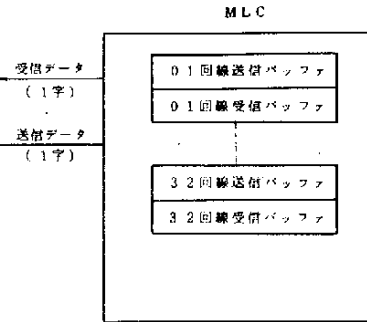
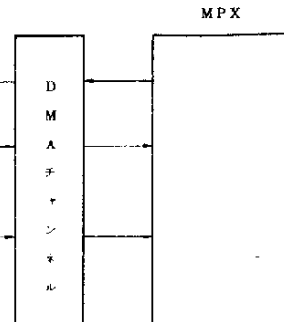
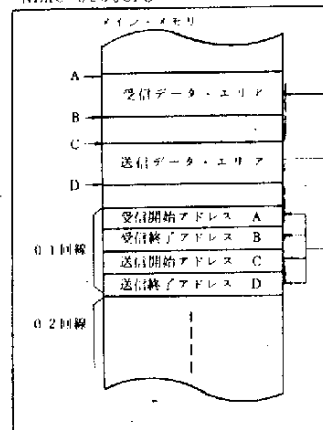


図2-13 中央処理装置とのデータ流れ図

NEAC 3200CPU



	MPX	MLC	LTU	回線の状態
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 MLCは1200ビット/秒の1ビットタイムの間隔(833 μ s)で0.1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833 μ s)で回線へ送出される。	LTU側 回線側 "0" +8V "1" -8V
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。	⑤ MLCは1200ビット/秒の1タイムビットの間隔(833 μ s)で0.1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。 ⑥ 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査が行われMPXへ転送される。	⑦ 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。	LTU側 回線側 "0" +8V "1" -8V

注. 回線の状態は、専用電話回線の場合(モデミング方式)を示し、電信回線の場合

C 5610ヘルツ変換装置で1.20mAのリレーインタフェースとなる。

図2-14 送受信動作概略図

MLCは他の周辺制御部と異なり、回線の接続方法、使用方法等によって動作が変わってくる。このため、MLCとCPUとの制御情報の為のハードウェア規定番地や、回線からMLCを通してCPUへ割込む制御符号等をシステムごとに定める必要がある。

ハードウェア規定番地は、IMP、TIPプログラムの共通部は同一性を保つ必要があり、またIMPプログラムの大きさを最大16 kwでおさえることに考え、さらに拡張用IMPの最低メモリ容量を16 kwとすることにして、17 kw目に指定することとした。

制御符号は各種ターミナルの機能により大巾に異なってくるため、ASCIIタイプの端末とそれ以外に分け、当初の端末5セットは会話型標準端末として使用方法を定め、CR、ETX、longspace を制御符号とすることとした。したがって通常の会話型で端末を利用する際はメッセージの終わりにCRあるいはETXをキーインすればMLCからCPUに割り込むことになる。

また、long spaceはブレーク信号としてQuitの機能用に使用する。

(10) 高速通信制御部 (9600 bps)

9600 bpsの高速通信制御部は、インテリジェント・ターミナル、高速データ・ステーション、遠距離ホストとの結合用に利用するつもりであり、実験段階では、IMP間の通信回線として使用し各種の実験をおこなう予定である。

(11) 高速製表装置

IMP、TIP用プログラムのディバッグ時にのみ用いる予定で、420行/分の印字スピードである。

2.3.3 各種端末の概要

前節のシステム構成でも述べたように、JIPNETでは、TIPに各種の最新の端末を接続し、コンピュータ・ネットワークの広範な利用実験をおこなうよう計画している。

JIPNETのTIPに当初の段階で接続する端末は以下の5種であり、実験が進むにつれて随時拡張していく予定である。

タイプライタ端末として

Terminet 300 (Honeywell Bull 製)

Silent 700 (Texas Instruments 製)

文字、グラフィック・ディスプレイとして

Consul 880 (ADDS 製)

Computer Display Terminal 4010-1 (Tektronix 製)

特殊入出力機器として

MDR 4000 (マーク・リーダー) (Bell & Howell 製)

AUTO Display * (マイクロ・フィルム・リーダー・プリンタ) (Image Systems 製)

高速データ・ステーションとして

U 200+CR, LP*

(富士通+VERSATEC)

[*は接続予定のもの]

これらの端末をJIPNETの実験端末装置として選定するにあたっては以下の諸点を考慮した。

- ① N 3200 MLC とモデム・インタフェースで接続可能のこと。
- ② 限られた予算で多様な実験をおこなうために低価格であること。
- ③ 新しい利用法を実験するにふさわしい最新の機能を有すること。
- ④ 実用化実験にも耐えうるよう、実用的機能も十分有すること。

次にこれらの端末の機能概要を簡単に紹介する。

(1) Terminet 300

Terminet 300 はGE製のタイプライタ端末である。インパクト・プリンタではあるが、タイプ・ベルト方式を採用しており、非常に静かで高速の印字ができるようになっている。ハードウェア性能上は10, 15, 30字/秒をセレクトできるようになっているが、MLCの伝送スピードが300 bpsがなく、200 bpsで接続せざるを得ないため、実質上の印字スピードは20字/秒に抑えて使用することになる。

印字位置を示すデジタル・カウンタがパネル前面についており、キーを押すたびにブザー音が出る点などが通常の端末とは変っている。

オプションとしてアンサー・バック・メモリ(端末IDをCPUからの要求により送出する機能)、モータのオン・オフの制御、ヴァーティカル・タブ(ホリゾンタル・タブは通常の端末にもあるが)等を装備した。

Terminet 300はJIPNETの標準的な汎用タイプライタ端末として使用するつもりである。

(2) Silent 700

Silent 700はTexas Instruments社製のタイプライタ端末である。感熱式のノンインパクト・プリンタであるため、ほとんど音がせず高速の印字ができるようになっている。ハードウェア性能上は10, 15, 30字/秒の印字スピードをセレクトできるようになっており、カセット磁気テープがついているので、旧来のASR紙テープ・テレタイプライタと同じように利用できる。

MLCとの接続は1200 bpsの伝送制御オプションを介しておこなっているので、カセットMTによるデータの送受信は120字/秒という高速のデータ伝送が可能である。さらに、プログラムからカセットMTのオペレーションの制御をおこなうためにRDC(Remote Device Controller)オプションを装備した。これによりたとえばカセットの巻戻し、ロード・ポイントへのセット、カセットへの記録、カセットからの印字、カセットMTデータの送出、状

態データの読出し等のカセットMTオペレーションのほぼすべてをCPUからの制御コードにより制御可能である。

Silent 700はJIPNETの高速タイプライタ端末として使用するとともに、RDC機能を生かしたアプリケーション用の高速データ自動送受信装置としても利用し得る。

(3) Consul 880

Consul 880はApplied Digital Data Systems社製のキャラクタ・ディスプレイ端末である。Consul 880には3種のオペレーション・モードがあり、夫々次のような特徴を有する。

① 会話モード

会話モードは、完全にテレタイプ・コンパティブルなモードであり、一字ずつキー・インするごとに伝送されるようになっていいる。さらにこのモードでは、最下段の行にカーソルが位置すると、一行分キー・インする毎に画面全体が上に一行シフトされるような、いわゆるスクロール機能が働くようになっていいる。

② メッセージ・モード

メッセージ・モードは、伝送が一行分毎におこなわれるもので、ハードウェア機能による種々のエディット機能を利用しながら一行分の正しいメッセージをつくりあげ、トランスミット・キーを押すことにより一行分のメッセージがブロック転送される。

③ ページ・モード

ページ・モードは伝送が画面全部一度におこなわれるもので、画面全体のデータ入力、編集が済んだ後にトランスミット・キーを押すことにより、一画面分の内容がブロック転送される。

メッセージあるいはページ・モードでは、固定データ（灰色表示）からなるフォーマットを表示し、この固定データの間に可変データ（黒色表示）を入力し埋めていくというフォーマット・コントロールが可能になっている。このフォーマット・コントロールによりデータ入力の信頼性をあげ、入力が容易になるとともに、可変データのみ送出できるようになっているので伝送時間も大巾に短縮される。

このほか、Consul 880には、グラフィック・オプションがついており、72×160のドットマトリックスで、適当なグラフの表示ができるようになっていいる。この端末は、1200bpsで接続していいるので、伝送・表示スピードは120字/秒である。

Consul 880は、JIPNETのパフォーマンス・モニタリング、および、オペレーション・コントロール用の表示・指令用装置として使用する予定であり、もちろん、一般のTIP端末としても利用する予定である。

(4) Computer Display Terminal 4010-1

4010はTEKTRONIX社製のグラフィック・ディスプレイ端末である。この装置は、日本でもかなり以前から使用されており、蓄積管型のグラフィック・ディスプレイで、リフレッシュが不用なので、オンラインのグラフィック端末として適しており安価で手軽に利用できるものの一つであろう。JIPNETでは、この装置を1200 bpsで接続し、ハード・コピー装置も準備しており、図形情報の入出力が必要なアプリケーションのための端末として使用する予定である。

(5) MDR 4000

MDR 4000 (Mark Document Reader)はBell & Howell社製のマーク・ドキュメント・リーダである。この装置は、パンチ・カード幅で任意の長さの読取エリアを設定することができるデータ伝送用のターミナルであり、マーク・ポジションのレイアウトは他機種と比べてかなり自由度がある。JIPNETではMDRを特殊なマーク・データ入力を必要とするようなアプリケーションの入力装置として利用するつもりである。またTerminet 300で必要項目を印字した後、必要な入力データをマークして入力するような、いわゆるターン・アラウンド入力システムの応用実験用の端末としても利用が可能になっている。

(6) Auto Display

Auto DisplayはImage Systems社製のマイクロ・フィルム・リーダ・プリンタである。収容コマ数は45,000あるいは60,000コマで、平均アクセス・タイムは3〜4秒程度である。現在はまだオフラインになっているが、次年度以降TIPに接続してタイプライタ端末と組合せて写真情報表示をコントロールするような利用法も予定している。

(7) 高速データ・ステーション

旧来のリモート・バッチ・ステーションは100枚/分のカード・リーダ、120行/分程度の入出力しか利用できず、比較的大量のデータ入出力を必要とする場合はこの低速I/Oがネックになっている。このため、ミニコン(U 200)に高速カード・リーダ(600枚/分)、高速ライン・プリンタ(500行/分)を接続し、高速リモート・データ・ステーションの結合を考えている。このステーションはTIPに9600 bpsで接続し、TIPからはディスタント・ホストとして扱うこともできる。

2.4 インタフェース・アダプタ

本節では、サブネットのIMPと各ホスト・コンピュータを結合するためのインタフェース・アダプタに関して、その機能設定上留意した点、伝送手順と機能レベル・コマンド、各アダプタにおける実現方法などについて述べる。

2.4.1 インタフェース・アダプタの機能設定

JIPNETは21, 22でも述べたように、FACOM 230/60, HITAC-8450, NEAC 2200/500という異機種コンピュータをN3200/50小型コンピュータ(IMP)を介し結合するもので、各ホスト・コンピュータとIMP間は両コンピュータのデータチャンネルをインタフェース・アダプタにより接続するものである。

このような異機種間インタフェース・アダプタは、両側のデータ・チャンネルの差異に対し次のような変換をおこなわなければならない。

① 機械的変換

両データ・チャンネルのコネクタ、ケーブル等は異機種の場合、その形状、ピンの配列などが異なっておりこのような機械的な差異を変換する必要がある。

② 電氣的変換

両データ・チャンネル間の信号の電氣的な差異、たとえば電圧、電流、インピーダンス等を変換する必要がある。

③ 論理的変換

両データ・チャンネルの入出力信号の論理的意味の差異を変換することで、たとえば片側のデータ・チャンネルからa, bという信号が同時に出力された場合、あるいは、cという信号が出力された場合に他方のデータ・チャンネルへxという信号を伝えるために $x = a \otimes b \oplus c$ ($\otimes$: AND, $\oplus$: OR)というような論理的変換をおこなうことである。この論理的変換はハードウェアとしては論理回路で実現されるが、この部分をどこまでハードウェアで持ち、あとはソフトウェアにまかせるか、すなわちハードウェアとソフトウェアとの機能分担をいかにすべきかが、インタフェース機能設定上もっとも重要なポイントとなる。

この場合、ホスト・コンピュータ側のソフトは、通常、既存のOSでサポートしている範囲で使用せざるを得ないため、ホスト・コンピュータに対する差異は、インタフェース・アダプタで吸収することになる。

しかし、IMP側のインタフェースについては、IOCSからすべて新規に作成することになっているので、ハード・ソフトを含めて全体システムとしての最適な機能分担を図ることが必要である。

次に、JIPNETのインタフェース・アダプタの各種機能を設定した際の基本的な方針について述べる。

(1) 標準インタフェースの実現

JIPNETのように、異機種コンピュータによるコンピュータ・ネットワークを構成する際には、ホスト・コンピュータをサブネットのノード（IMP）に接続するためのインタフェースは、ハードウェア・ソフトウェアともに標準化された規約にしたがうことが必要である。

この標準インタフェースと各ホスト・コンピュータの差異は、ホスト側のインタフェース・アダプタ、あるいは、ホスト内のソフトウェア的変換によって変換することになる。このような形で、JIPNETサブネットとホスト・マシン間の標準インタフェースを明確に規定しておけば、全体システムのインタフェースが簡略化されるばかりでなく、新たに他のホスト・コンピュータが、JIPNETに参加していく際にも、その新規ホスト・マシン特有のインタフェース部分のみを変換するようなインタフェース・アダプタを介して結合すれば良い。

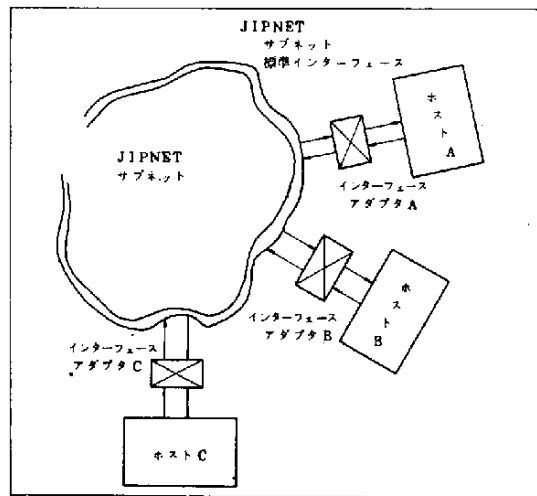


図2-15 標準インタフェースとホストとの結合

(2) 単純な転送手順

サブネットIMPのデータ・チャネルとホスト・コンピュータのデータ・チャネル間に位置するインタフェース・アダプタは、基本的には両コンピュータ間のデータを転送する機能を有すれば良いわけで、その転送の為の手順が、いたずらに複雑である必要はなく、ソフトからの利用上、できるだけ単純であることが望ましい。

あるホスト・コンピュータ側でのハードウェア機構あるいはソフトウェア（OS）の入出力制御機構がもともと複雑で、そのホスト側規約にしたがわなければ入出力装置が使用できない場合があっても、ある一機種の複雑なインタフェースの為に標準インタフェースそのものが複雑化することは避けなければならない。このような場合は、単純化した標準インタフェースとの

ギャップを、できるだけそのホスト特有のアダプタが埋めるべきである。

また、IMP側からの転送制御も、小型コンピュータであるが故に、ソフトでかなり複雑な制御を必要とし、かつそれが可能な場合でも、論理的にまとめる事が可能な信号はできるだけまとめるなどして転送手順そのものを単純化すれば、IMPプログラムのパフォーマンス向上にもつながる。この場合問題になるのは、先に述べたハードウェアとソフトウェアとの機能分担で、どちらか一方に片より過ぎると、ハードウェアあるいはソフトウェアの大巾な効率低下およびコスト・アップにつながる。

(3) テスト機能の充実

我国における異機種コンピュータ結合に関する経験・実績はコンピュータ・メーカ、あるいはユーザとも必ずしもまだ十分であるとは言えず、技術的にもまだかなりの困難性を残している。このような時点で、異機種コンピュータをチャネル経由で結合するには、その結合に際してのテスト機能を、あらかじめ充分考慮しておくことが不可欠である。この為の方法としては、折返しテスト機能を持つとか、各信号の状態表示を可能にしておくとか、他系のチャネルのシミュレータを作成し、シミュレーション・テストをおこなうなど、予め、トラブルの解消を出来るだけ容易にする手段を考慮しておくことが望ましい。

また、結合した後発生する障害についても、異機種間のどこでその障害がでているのか、できるだけ速やかに発見できるような障害診断機能が必要である。

この障害診断機能は、オンラインでの利用が可能にしておくことにより、IMP、およびホストからの他系への障害を発見を容易にすることができ、サブネット運転中のホスト切離し、またはその再生が可能になる。

このような方針に基づいてJIPNETのインタフェース・アダプタの機能検討に入り、三ホストへのインタフェース・アダプタの構成を図2-16のように定めた。

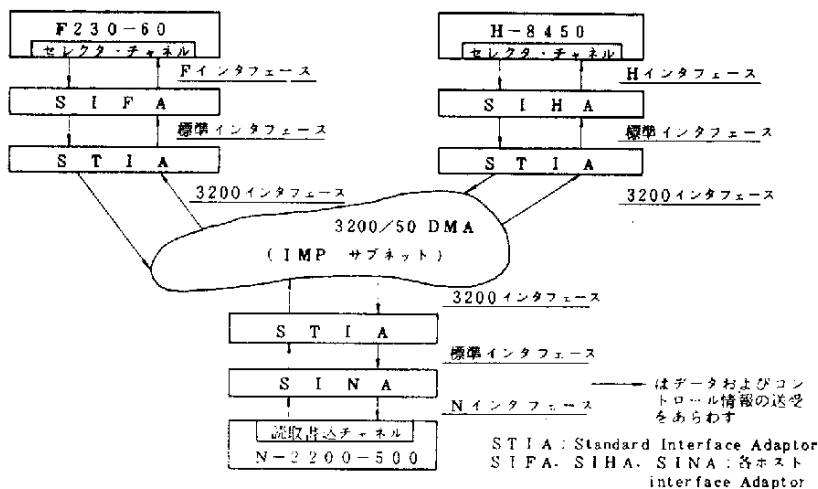


図2-16 JIPNETインタフェース構成

次に、これらのインタフェース・アダプタについて設定した機能を以下に示す。

- (1) 通信方式は半二重方式とする。
- (2) 転送方式はアダプタ間 8 ビット並列転送、ブロック転送方式とする。
- (3) 転送速度は両 CPU 間で 1 M bps とする。
- (4) 転送制御用コマンドは、データの入力、出力、状態の読取、状態制御用コマンドを基本とする。
- (5) 転送要求の優先順位は送信要求を早く出した側を高くし、送信要求が衝突した場合はその旨両 CPU に通知して、ソフトウェアで再開するものとする。
- (6) 障害検出のため、小量データの折返しテスト機能をつける。
- (7) ホスト・IMP 間のフロー・コントロールのため、プログラム・スイッチを設け、転送の抑止、再開を可能にする。

これらの結合の機能図は次のようになる。

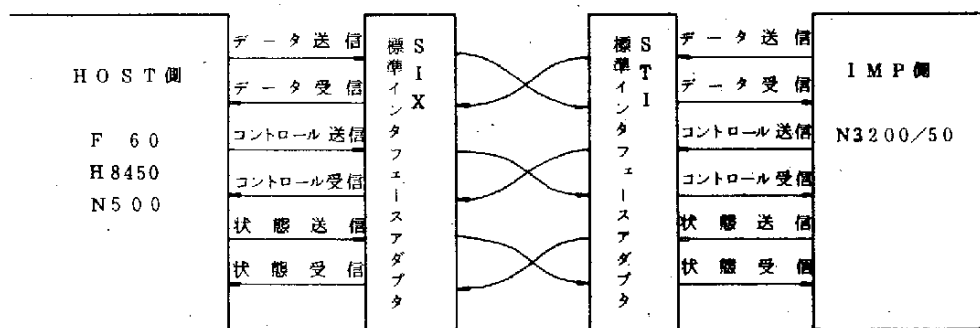


図 2-17 インタフェース・アダプタ機能図

2.4.2 インタフェース・アダプタのコマンドと手順

本項では、FACOM 230/60のF adaptor, HITAC-8450のH adaptor を例にとり、JIPNETで採用した標準インタフェースに対するコマンドの機能と使用方法を説明したい。

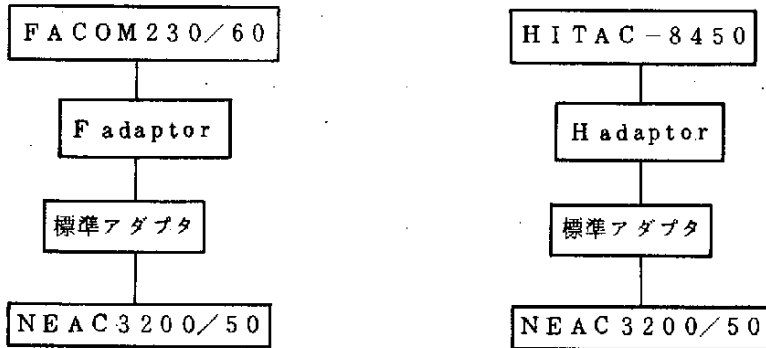


図 2-18

A. 機能概要

F 60, H 8450 と N 3200 間のデータの送受には出力用コマンド Write Data with Interrupt (WDI), Write Data (WD) と入力用コマンド Read Data (RD) を使用し、両 CPU からの入出力要求がマッチしたらデータ転送を開始し、いずれかの CPU のデータ・カウンタが 0 になった時点で転送を完了する。WDI と WD の違いは他系が RD をまだ発していない場合に割込をかけるかどうかという点である。

両 CPU で発したコマンドがマッチしない要求 (WD/WDI-WD/WDI, RD-RD) の場合は incompatible の状態で異常終了する。WDI or WD のデータ・カウンタが RD のデータ・カウンタより大きい場合は転送終了時に incorrect length のセンス情報を表示する。

また、両 CPU のプログラムがデータ転送可能状態にあるかどうかの状態表示用として、センス情報内に Program Switch (PSW) を設け、Sense command により相手 CPU のプログラム状態を検知し、かつ psw の lock, unlock をコマンドにより制御できるものにする。

B. コマンドの種類および動作概要

アダプタ用コマンドの一覧表を以下に示す。

コ マ ン ド 名	F-60 コマンド・コード	H-8450 コマンド・コード
	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7
1. Write Data with Interrupt	0 M M M M 0 0 1	M M M 0 0 0 1 1
2. Write Data	0 M M M M 1 0 1	M M M 1 0 0 1 1
3. Read Data	M M M M M 0 1 0	M M M M 0 1 0 1
4. Sense	M M M 0 0 1 0 0	M M M M 0 0 0 1
5. No Operation	M M M 0 0 0 1 1	M M 1 1 0 1 1 1
6. Psw lock	M M M 0 1 1 1 1	M M 0 1 0 1 1 1
7. Psw unlock	M M M 0 0 1 1 1	M M 0 0 0 1 1 1
8. Who Are You		1 0 0 0 0 0 0 0
9. Send Status		M M M M 0 1 1 0
10. Set Interrupt		0 0 1 0 0 0 0 0
11. NOOP		M M M M 1 0 0 0

(M: Modifier)

次にこれらの各コマンドの動作概要を述べる。

(1) White Data with interrupt (WDI)

WDIコマンドは、他系にデータを送出するために使用するもので、送出要求は割込で他系に知らされる。実際のデータ転送は、他系がRead Data (RD) コマンドを発信して初めて可能になる。

他系がRDを先に発信し読み込み待ち中の時は割込まずただちにデータ転送を開始する。

他系がWDIを発した時の割込は、F-60においてはAttention割込、H-8450においては装置要求割込として自系CPUに知らされる。

自系がWDIを発した時のデータ転送終了は、F-60においてはCHE、DVEで終了割込、H-8450ではTIで終了割込にて完了する。

但し incompatible なコマンドを他系が発した場合、他系のコマンドがWDIの時F-60では

CHE, DVE, UCK, UEX およびセンス情報の incompatible ビットをオンにして自系の WDI コマンドを異常終了させ、H-8450 においては TI, SI およびセンス情報の incompatible ビットと WDI ビットをオンにして自系の WDI コマンドを異常終了させ、自系 CPU に割込む。他系のコマンドが WD の時は F 60 では CHE, DVE, UCK, incompatible bit をオンにして自系の WDI コマンドを異常終了させ、H-8450 においては TI, SI およびセンス情報の incompatible bit をオンにして自系の WDI コマンドを異常終了させ自系の CPU に割込む。

(2) Write Data (WD)

WD コマンドは他系にデータを送出するために使用するものであり、WDI コマンドと異なり、送要求の割込は起さない。実際のデータ転送は他系が Read Data コマンドを発信して始めて可能になる。自系が WD を発した時のデータ転送の終了は、F 60 においては CHE, DVE, H-8450 においては TI にて自系 CPU に割込んで知らされ完了する。

他系が incompatible なコマンドを発した時の処置は WDI の場合と全く同じである。

(3) Read Data (RD)

RD コマンドは他系からデータを受信するために使用する。実際のデータ転送は他系が WDI あるいは WD コマンドを発信して始めて可能になる。他系に対する転送要求の割込は起さない。

自系が RD を発した時のデータ転送の終了は、他系の発したコマンドが WDI, WD のいずれであるかにより、以下のような完了状態をとり自系 CPU に割込む。

① WDI - RD の場合

(他系) (自系)

F 60 CHE, DVE, UEX

H-8450 TI, SI, センス情報の WDI ビット

② WD - RD の場合

(他系) (自系)

F 60 CHE, DVE

H-8450 TI

(4) Sense

Sense コマンドはアダプタの詳細な状態を知るために使用し、通常は I/O 動作の異常終了時にこのコマンドが出されアダプタに保持されているセンス情報が CPU に読込まれる。

自系 CPU が自系のアダプタとの間でデータ転送結合中でなければ、すなわち、自系が WDI, WD, RD コマンドの実行中でなければ、センス・コマンドをいつでも発することができる。

(5) No Operation (NOP)

NOP コマンドはハードウェアの動作は何もせず、プログラムが任意に使用して良いコマンドである。このコマンドの実行によりステータス情報は CPU にとり込まれるので Test I/O として利用することもできる。

(6) Lock Psw (LK)

LKコマンドは、アダプタに保持されているセンス情報のProgram Switchをlock するために使用する。lockするPswは自分側のアダプタのものであり、Senseコマンドで知られるのは相手側アダプタのPswの状態である。

自系のPswがlockされている間は他系からのI/Oコマンドは拒否され異常終了 (F 60 はCHE, DVE, UCK, Psw, H-8450はTI, SI, Psw)させられる。

(7) Unlock Psw (ULK)

ULKコマンドは、アダプタに保持されているセンス情報のPswをunlockするために使用する。

UnlockするPswは自分側のアダプタのものであり、Senseコマンドで知られるのは相手側アダプタのPswの状態である。電源が投入され初期状態でのPswはlock状態になっている。

(8) その他のコマンド

この他、H-8450はchannelから出されるコマンドとしてWho Are You, Send Status, Set Interrupt, Noopの4種のコマンドがある。

このほか、コマンドではないが、I/O動作を強制的に終了させるCPU命令としてF-60ではHalt I/O, H-8450ではHalt Deviceがある。

この命令が実行されるとチャネルはI/O動作を中止し終結のシーケンスに移るので、アダプタ側では、このシーケンスになったらI/O動作を中止し、レディ状態になる。この時、他系と結合状態であれば他系I/O動作を強制終了させ、完了情報としてHalt指令があった事を他系に報告する。

しかしH-8450ではHalt Deviceと通常の終結シーケンスが全く同じで区別できないのでアダプタでは検知不可能である。

C. ステータス情報

アダプタからCPUに報告するステータス情報は以下の8ビットの情報である。

(1) F 60ステータス情報

ビット位置	名 称	略 称
0	Attention	A T T
1	_____ (0)	_____
2	_____ (0)	_____
3	Busy	B S Y
4	Channel End	C H E
5	Device End	D V E
6	Unit check	U C K
7	Unit Exception	U E X

ATT : 他系からWDI コマンドを受け取った時に、自系側アダプタが Read 状態にない場合にセットされる。ATT 割込が受付けられるとリセットされる。

BSY : アダプタが何らかのコマンドを受取った時にセットされ、アダプタが現在動作中であることを示す。終了割込でリセットされる。

CHE : このビットは DVE と常に同時にセットされるもので、コマンドの実行が終了した事を示す。終了割込が受付けられるとリセットされる。

DVE : このビットは普通 CHE と同時にセットされるもので、コマンドの実行が終了した事を示す。

終了割込が受付けられるとリセットされる。

UCK : コマンド実行中に異常状態が発生した事を示す。

詳細な内容はセンス情報にて表示される。

Sense コマンド実行後リセットされる。

UEX : このビットは他系が WDI を発して自系に対するデータ転送が終了した時、あるいは他系の WDI に対し、自系が incompatible なコマンドを発し、異常終了した時にオンになる。

(2) H-8450 ステータス情報 (SDB)

ビット位置	名 称	略 称
0	装置要求割込	MRI
1	終了割込	TI
2	Device busy	DBY
3	control busy	CBY
4	————(1)	
5	Secondary Indicator	SI
6	inoperable	IOP
7	————(0)	

MRI : 他系から WDI コマンドを受け取った時に自系側アダプタが Read 状態にない場合にセットされる。MRI 割込が受付けられるとリセットされる。

TI : コマンドの実行が終了した事を示す。

終了割込が受付けられるとリセットされる。

DBY : アダプタが何らかのコマンドを受取った時にセットされ、アダプタが現在動作中であることを示す。終了割込でリセットされる。

CBY : このビットは常に DBY と同じにセット、リセットされる。

SI : センス情報として報告すべき内容がある場合にセットされ、senseコマンド終了後リセットされる。

IOP : このビットは他系の処理装置が動作不能の時にセットされる (power off , general reset etc) 動作付の状態になったらリセットされる。

結合時、結合中に他系の inoperable 状態が検出されたら IOP をオンにして異常終了する。

D. センス情報

センス情報は Sense コマンドの実行により知らされる 8 bit の情報で Psw, loop test bit (F 60 では inoperable も含む) 以外は Sense コマンドの終了時、その他のコマンドの起動時、System reset, selective reset によってクリアされる。

(1) F-60 のセンス情報

ビット位置	内 容
0	invalid command
1	inoperable
2	psw lock
3	loop test (test 中 1)
4	incorrect length
5	incompatible
6	Halt I/O
7	_____ (0)

○ invalid command

許されたコード以外のコマンドが発信された。

○ inoperable

H-8450 Status 情報の inoperable とほぼ同じ。

○ psw lock

相手側アダプタで、初期状態後あるいは、LK 発信後 ULK コマンドがまだ発信されていない。

ULK コマンドによりクリアされる。

○ loop test 中

他系が loop test 中である。test 終了後クリアされ、loop test 中の F, H からの I/O コマンドは拒否される。

○ incorrect length

WDIあるいはWDで指定した送出バイト数がRDで指定した読込バイト数より大きい。

○ incompatible

両系のコマンドがマッチングしない。

WDI/WD — WDI/WD RD—RD の時にセットされる。他系がWDIとWD のいずれを出して incompatibleになったかは F 60においては UEX, H-8450 には WDI ビットのオンにより区別する。

○ Halt I/O

他系と結合中に、他系から Halt I/O が出された。

(2) H-8450のセンス情報

ビット位置	内 容
0	(0)
1	Halt I/O
2	Psw off
3	loop test
4	WDI
5	incorrect length
6	incompatible
7	invalid command

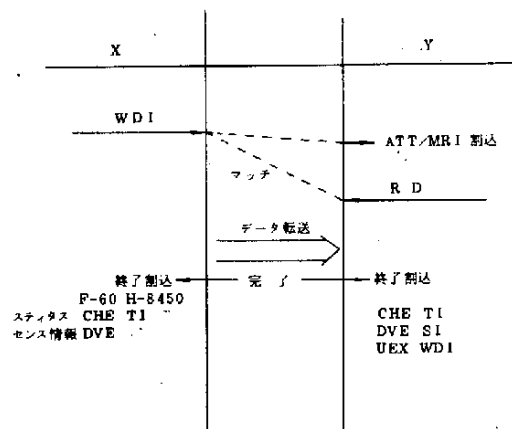
○ WDI

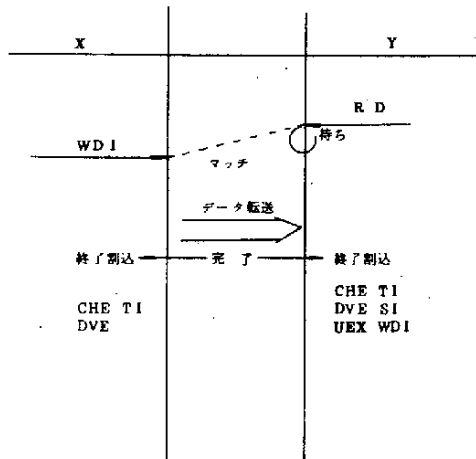
F-60のUEXビットと同じ意味を持つ。

○その他のビットはF-60と同様

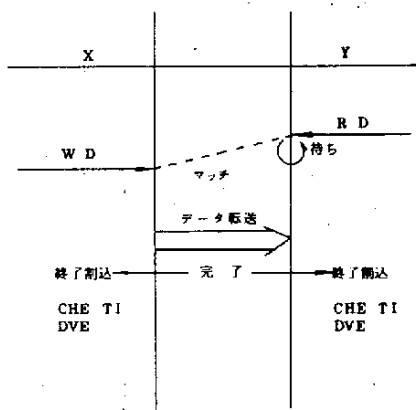
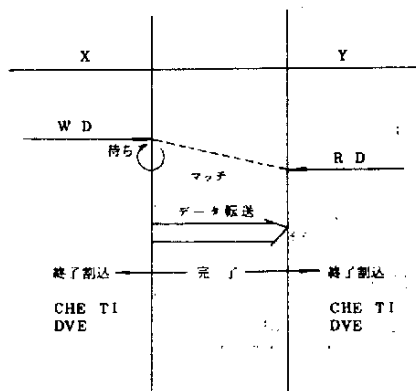
E. コマンドの使用法

(1) WDIとRD

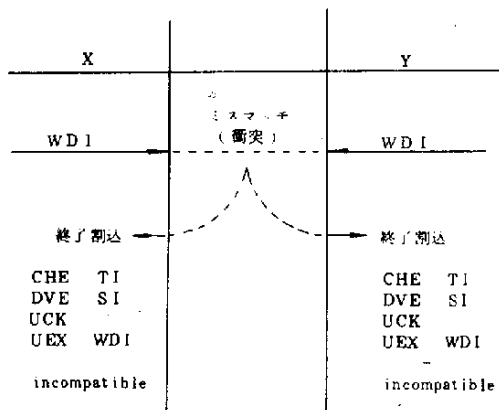




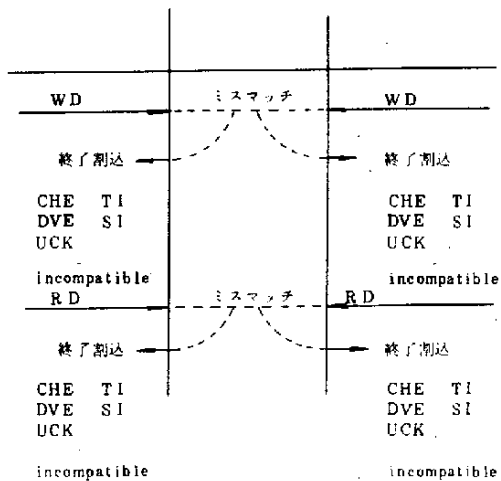
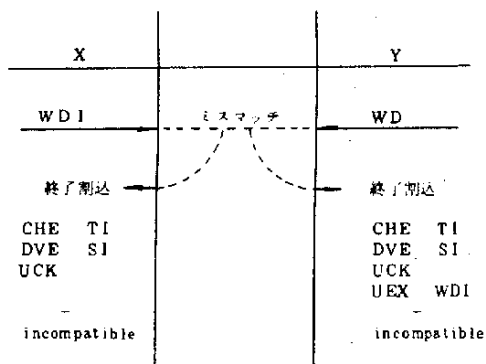
(2) WDとRD



(3) incompatible なコマンド

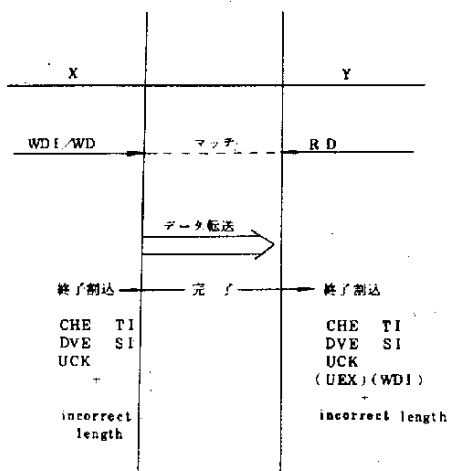


WDI の衝突も上記と同じ処理



(4) incorrect length

WDI or WDのデータ・カウント > RDのデータ・カウント



2.4.3 IMP側アダプタの動作

前項では、ホスト・コンピュータからのコマンドを中心にインタフェース・アダプタを説明したが、本項では、IMP側アダプタの動作概略、インタフェースの構成について説明する。

A. N 3200 アダプタ (STIA) 動作概略

- (1) データ転送は両CPUがプログラムにより任意に発する事が可能なWDI (WRITE Data with Interrupt Command) 又はWD (WRITE DATA Command) とRD (READ DATA Command) の条件が整合する事により開始される。

NEAC-3200とIOC間は16BIT並列 (word単位) で転送されるが、IOC — D/R — D/R — 他のインタフェース間は8BIT並列で転送される。又転送されるDATA数はプログラムにより任意に指定出来る。

転送されるDATA (8BIT DATA) のうち奇数番目はNEAC-3200, 16BIT DATAの1~8BIT (UP 8BIT) に対応し偶数番目は9~16BIT (LOW 8BIT) に対応する。

尚WDIを発すると相手側アダプタが先にRDを発していなければ相手側CPUに割込を発する。これは相手側よりWDIが発せられた場合も同様にNEAC-3200 CPUに対し割込を発する。

データ転送の正常な終了はどちらかのCPUのデータ・カウンタが終了を示した時である。(NEAC-3200はレンジカウンタより分けられるEND OF RANGE 信号により終了する)

但しREAD側CPUが先に終了を発した場合はWRITE側アダプタにIncorrect LengthのSTATUSを残す。

NEAC-3800ではSKS××××命令 これを判別する事が出来る。

尚 IOCは送受信データブロックのデータ数が奇数の時は次に示す処理を行う(ハードウェア)

① IOC READ MODEの場合

最終データの9~16 BITを " zero " としてCPUに送る。又SKS (センス命令)で聞く事が可能なODDREAD プラグを立てる。

② IOC WRITE MODEの場合

WDI(又はWD), 実行以前にODW(奇数WRITE) 命令を実行して置く事により最終データ(16 BIT DATA)の9~16 BIT を送出せずに終了する。

データ転送が正常に終了するとIOCには必ずWCE(WDI END)又はWDE(WD END)のSTATUSが残る。これにより相手側のCommandが一連の手順の途中変更されても転送終了後知る事が可能になる。

(2) データ転送の異常終了は次の場合がある。

① 両CPUのCommand が不整合の場合。

② 何らかの原因でハードウェア又はソフトウェアがストール、データ転送が中断した場合又は相手側プログラム上の都合で一定時間内に整合するCommandが発せられなかった場合。Command不整合を検出したIOC及び他インタフェースは両CPUに割込を発して終了する。

この場合IOCにはINCP(Incompatible)WCE(WDI From HOST), WDE(WD From HOST)等のSTATUSが残るのでプログラムにより不整合状態を知る事が出来る。

データ転送の中断及び未開始はIOCでは検出機能を持たないのでプログラムは、Timer 等を用いてこれを検出しなければならない。プログラムがこの状態を検出した後HALT Command (IOCではイニシャライズ指令)を発する事により終了させる事が出来る。

相手側より、HALT Commandを受けたインタフェースにはHALTのフラグが残るのでプログラムにより検出可能である。

(3) 割込機能

IOCはMASKFLOPがSET されていて以下に示す各状態が発生するとCPUに対し標準割込(PILOO)を発する。

① NOT BUSY状態の時相手側よりWDIを受信した場合。

NOT BUSY状態とは両方のインタフェースが正常でかつIOCはCPU(NEAC-3200)より何のCommandも受付ていない事を示す。

② IOCがBUSYからNOT BUSYに変化した場合。

IOCはNOT BUSY状態でCPUよりCommandを受けるとたゞちにBUSYとなる。

NOT BUSY割込はIOCがCPUより受けたCommandを終了した事を意味する。

たゞし正常終了か異常終了かわ後迷するセンス命令によりプログラムで判別する必要がある。

- ③ IOCが、I/Oモードに設定されており"READY"になった場合I/Oモードとプログラムモードによるデータ転送の事であり"READY"とはCPUとIOCデータ転送可能な状態を意味する。

(4) 各種STATUS

IOCは何らかの動作を終了するとNOT BUSY割込を発生すると共に動作上の各種STATUS FLOPを残しその動作状態をCPUに知らせる事が出来る、プログラムはSKS命令を用いてこれらのSTATUSを知り、処理を行う。

STATUS FLOPには以下に示すものがある。

① WCE FF

IOC又は相手側よりのWDIが正常終了した場合又はINCPで終了した時相手側のCommandがWDIであった事を示す。

② WDE FF

IOC又は相手側よりのWDが正常終了した場合又はINCPで終了した時相手側のCommandがWDであった事を示す。

③ ODDR FF

IOCのFEAD DATAが奇数であった事を示す。

④ ICL FF

IOCがWRITE又はREADの時READ側か先に終了した事を示す。

⑤ INOP FF

Commandが不整合で異常終了した事を示す。

IOC WRITE MODEでのINCPの場合①②のチェックにより相手側のCommandを知る事が出来る。

⑥ INOPH FF

BUSY状態の時のみ、発生するもので、相手側インタフェース(D/Rも含む)の電源断を示す。

IOCは相手側電源断又は次項のpsw Lockの時はBUSYとなる。

⑦ PRSWH

BUSY状態の時のみ発生するもので相手側インタフェースのpswがOFF(Lock)であることを示す。

⑧ HALT

相手側よりHALT Commandが発せられた事を示す、但しIOC側が先にHALT Commandを発した場合はこのSTATUS FLOPは残らない。

SKS 命令は以上の状態を個々に判別出来ると共に IOCの動作状態を知るためのものがいくつか用意されている。

B. N 3200 アダプタ (STIA) のインタフェースの構成

N 3200 アダプタ (STIA) の構成を図 2-19 に示す。

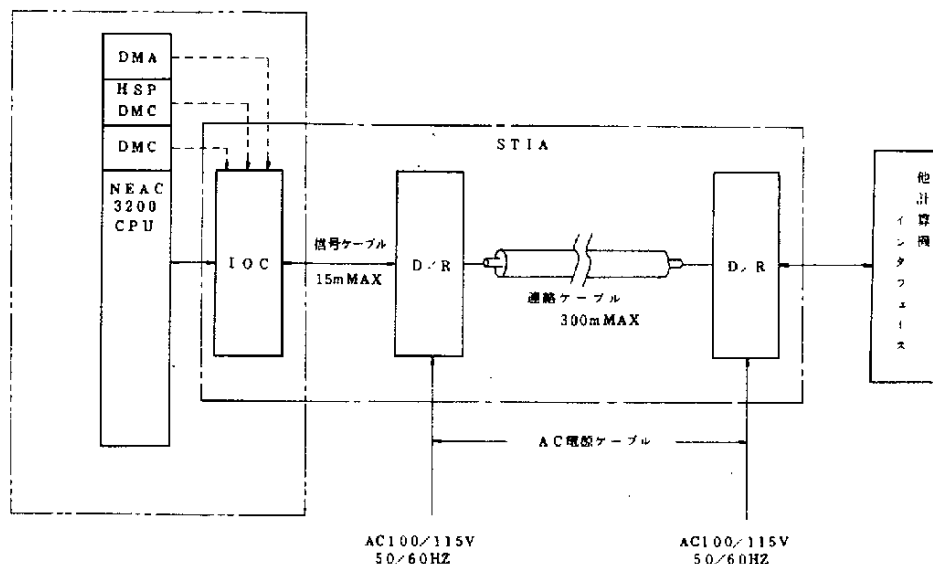


図 2-19 N 3200 アダプタの構成

N 3200 とホスト・インタフェース・アダプタ間のインタフェース接続信号は図 2-20 の通りである。

各信号の意味を以下に示す。

(1) IMP からホストへの信号線

WCIX	WDI 送出ライン
WDIX	WD 送出ライン
RDIX	RD 送出ライン
DTI01	8 bit 並列データ送出ライン
}	
DTI08	
STIX	データ STROB 信号ライン
CLLIX	Clear (HALT) 信号ライン
POWIX	電源 ON 信号ライン
PLTIX	ループテスト制御ライン

(2) ホストから IMP への信号線

上記信号のうち I を H に変更したもので機能は同じである。

(3) プログラム・スイッチ ON 信号

プログラム・スイッチが OFF となった場合は DTX01 ラインが ON (" 1 ") となり、
POWXX ラインが OFF (" 0 ") となる事により判別する。

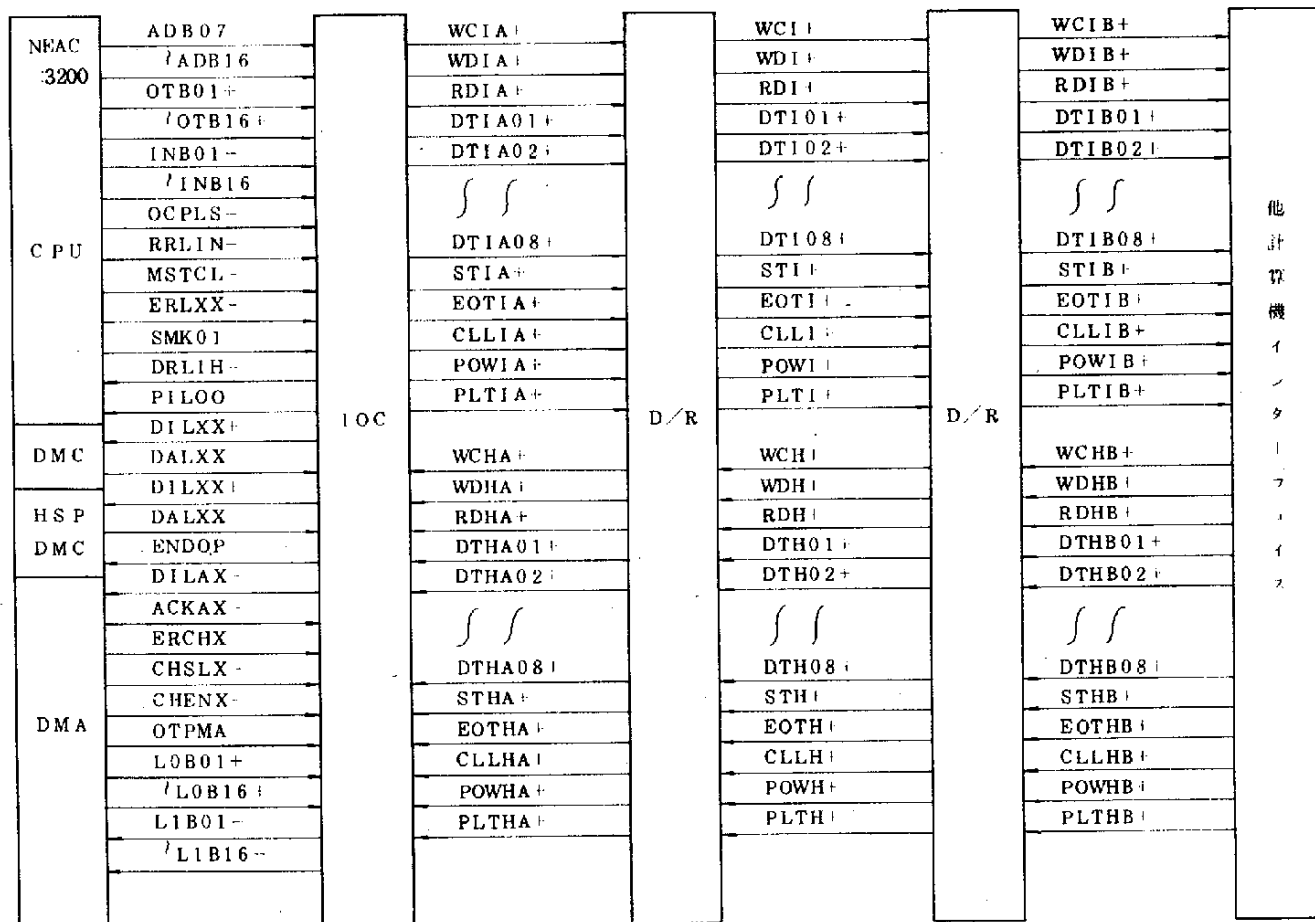
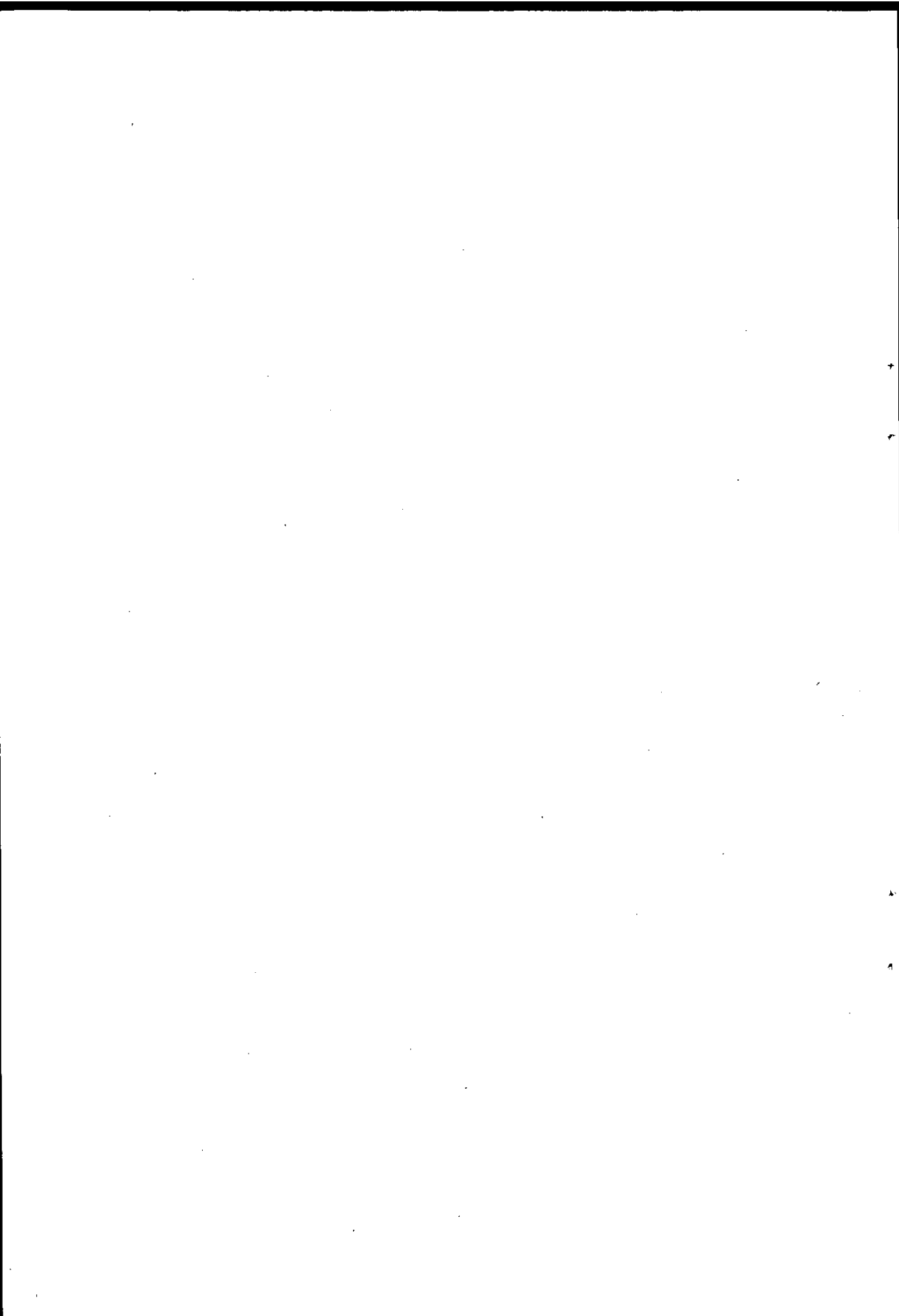


図 2-20 インタフェース信号

3. サブネットのソフトウェア

3.1	サブネットの基本構造	75
3.1.1	結合方式	75
3.1.2	伝送方式	77
3.1.3	制御方式	79
3.2	サブネットの位相幾何学的構造	79
3.3	I M Pの機能	83
3.3.1	蓄積交換	84
3.3.2	パケットの伝送確認手続き, 再送手続き	88
3.3.3	ルーティング	92
3.3.4	フロー・コントロール	105
3.3.5	シーケンス・コントロール	112
3.3.6	障害対策機能	115
3.3.7	デバッグ, モニタリング機能	120
3.4	I M Pのプログラム構造, データ・ストラクチャ	123
3.5	サブネットの解析, 評価に関する定量的手法	128
3.5.1	I M Pの最大スループット率	129
3.5.2	ラウンド・トリップタイムとメッセージ長の関係	131
3.5.3	回線利用率とバッファ個数	132
3.5.4	メッセージ・ディレイ・モデル	133



3. サブネットのソフトウェア

コンピュータ・ネットワークの形成に於ては、その最も基本に於て、計算機間通信 (Computer to Computer Communication) を可能にする機構が確立されていなければならない。

この章では、コンピュータ・ネットワークにあって、コミュニケーション・システムとしての役割を果たすサブネットについて、その機能を明らかにする。

初めに 3.1 では、サブネットの基本的な性格を決定する結合方式、伝送方式、制御方式について説明を加え、現在、我々が指向している分散型蓄積交換網 (Distributed Store and Forward) についてその基本概念を明確にする。

3.2 ではネットワークの骨格とも云うべき位相構造 (Topological Structure) について分散網を採用したいきさつを述べ、ネットワーク・トポロジーの最適化の問題について簡単な考察を加える。

3.3 は本章の中心とも云える部分で、サブネットの動作を決定する IMP の機能について詳細な説明を与える。これはネットワーク・ソフトウェアと云う観点からは IMP-IMP プロトコル、Host-IMP プロトコル、発信地 IMP-目的地 IMP プロトコルと云う形式で要約されるものであるが、ここでは機能面からの把握を重視し、各機能についてその動作機構を中心として話をまとめ、規約本位な話の展開を避けた。

3.4 では 3.3 で述べられた各種のサブネット機能を実際に遂行する IMP プログラムに関して、そのプログラム構造、データ構造を解説し、その動作機構を明確にする。

3.5 ではネットワークのパフォーマンスの解析及び評価に関する基本的なアプローチを示す目的から、ARPA ネットワークの研究を中心に、その定量的手法のいくつかを紹介する。

3.1 サブネットの基本構造

3.1.1 結合方式

異機種計算機を相互に接続し、ネットワークを形成する方法として最も直接的な方法は、図 3-1 に示すようにネットワーク・コンピュータ同志をチャネルを介して直接結合する方法である。

この方法によれば、通信設備は回線のみで構成され一見経済性に優れているように思われる。しかし実際には、異機種間ではチャネルのデータ伝送手順、語長等に相異があるためこれらを相互に結合するためには両者のインタフェースを合わせるための特別なハード要素 (インタフェース・アダプタ) を作成することが必要になる。

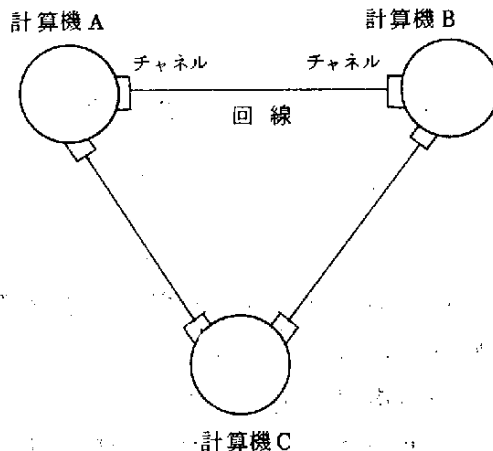


図 3-1 ホスト-ホスト直接結合

このための労力、コストは、ネットワーク規模が大きくなるにつれて、大きなものになる。又ネットワークの拡張や修正、(例えばネットワーク・コンピュータを一つ追加したり、ラインを再配置したりする場合)に関しても余り自由度が望めない。

又、機能的には、各々の計算機システムは本来の処理に加えて、データの送受に関する各種の雑務を行わなければならない、そのための計算負荷が増大し本来の処理に関して、機能低下を招くこととなる。

この様な理由から、コンピュータ・ネットワークの形成に於ては図 3-1 の様な直接結合の形態はほとんど採用されていない。

現在一般に行われている方法は、図 3-2 に示すように、小型計算機を介してネットワークを形成する方法である。

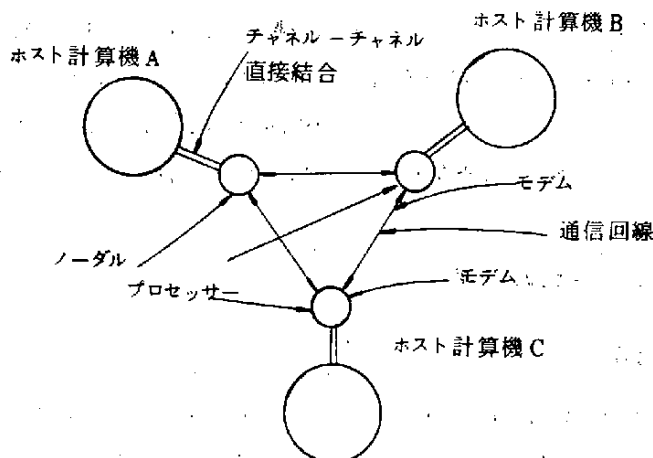


図 3-2 ミニコンを介した結合方式

即ち大型計算機システム (Host Computer) の近くに小型計算機 (Nodal Processor) を配置しこの間をチャンネル直接結合又はモデム経由で接続し、小型計算機同志をデータ回線で相互に接続することによりネットワークを形成する。

この場合、小型計算機を同一機種で統一することにより、各ホスト計算機と同一の小型計算機の間のインタフェースのみを問題にすればよいことになり、接続上のインタフェース整合の問題はかなり楽になる。

又小型計算機群と回線によって構成されるネットワーク(サブネット)にデータ交換機能を専門に行わせることによりホスト計算機からデータ交換に関わる負荷の多くを開放することができる。

その他このタイプの結合方法によるネットワークの利点として次の様な点があげられる。

- ホスト・コンピュータとサブネットの動作を独立にすることができるからホスト計算機システムの障害によって通信機構が影響を受けることが無い。即ち通信設備としての信頼度を高めることができる。
- 同一機種で構成されているから、ハード、ソフト共に保守が容易である。
- 同一機種であれば、サブネットのソフトウェアは全く同じものを準備すればよいからソフトウェアの開発コストという点からも非常に有利である。

現在このような形式の結合方法をとっている代表的なコンピュータ・ネットワークとしてARPAネットワークがある。我々の場合も、以上に述べた事柄を考慮した結果、拡張性、信頼性、効率等の面からこの形式の結合形態が最も優れていると結論されたため小型計算機をノードル・プロセッサとして、ネットワークを構成する方式を採用することにした。現在ノードル・プロセッサとしてはN3200/50を使用している。又回線は50KB、全二重回線でネットワークは3ノード3回線から構成されている。

図3-3はこの様な結合形式によるネットワークの一例を示したものであるが、この構成に於てデータ交換を専門に行うノードル・プロセッサをIMP(Interface Message Processor)と呼び、大型計算機の方はホスト・コンピュータと呼んで区別している。又IMPと回線から構成される通信システムは、ホストと独立した通信機構と考えこれをサブネットと呼ぶ。又従来のTSS回線網と区別してサブネットをハイレベル・ネット、TSS網をローレベル・ネット(ローカル・ネット)等と呼ぶこともある。

3.1.2 伝送方式

あるソース・ホストで発生したデータはソースIMPからサブネットに流入し回線を通して直接に又はいくつかのIMPと回線を経由して目的地IMPに到達し目的地ホストに受け入れられることになる。この時データ伝送のやり方としてサブネット・リソース(IMPと回線)をスタティックに割り当てるかダイナミックに割り当てるかという違いから2つの伝送方式が可能である。前者は電話交換技術の分野で用いられている回線交換方式であり後者は、メッセージ交換方式と呼ばれる方法である。

回線交換方式では、データ伝送に先だってまずコントロール・メッセージを送り目的地との間に

フィジカルなデータ・パスを確立する。データは設定されたパスを經由して目的地に送られるが、その間このデータ・パスは伝送が終了するまで専有されることになる。

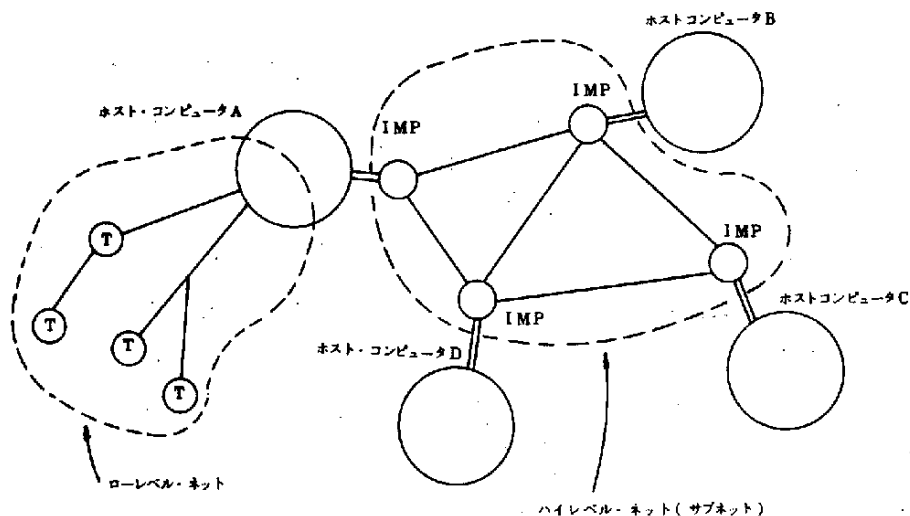


図 3-3 サブネットの概念図

この方法によれば最初のパスの設定に少々時間を要しても長大なメッセージを連続的に伝送するような場合にはかなり効果的なデータ伝送が期待できる。しかし会話型処理にみられるような短いメッセージを頻繁にやりとりするような場合は余り有利ではない。又回線障害が発生した時の致命度も大である。その他負荷変動やネットワークの利用効率と云う観点からも余り優れているとは云えない。

以上の様な理由から伝送方式は基本的にメッセージ交換方式を採用することを決定した。

メッセージ交換方式に於ては発信地、目的地の間に前もって特定のデータ・パスは確立しない。メッセージは行き先に関する情報(リーダー)を持ってサブネットに流入する。各 IMP は受け取ったメッセージの行き先を調べその目的地に最短時間で到達できるような径路を選択してこれを送り出す。この様にメッセージがどのノードのどの回線を利用して目的地に到達するかはあらかじめ定められておらず、その時の状況に応じた適切な割り当てが行われることになるが、回線交換方式と比較するとノード・プロセッサの負荷は多少増大することが考えられる。しかし短いインタラクティブ・メッセージの扱いやネットワークの利用効率、回線障害等に関しては優れた性格を備えていると云える。

サブネット内に流入したメッセージは厳密には、発信地 IMP に於てパケットと呼ばれるサブネット内の伝送の単位に分割され、1つ又は複数個のパケットがそれぞれ独立に目的地に向って伝送される。目的地 IMP に於ては、順次到着してくるこれらのパケットを再びもとのメッセージに編集し直し目的地ホストに渡す。この様なメッセージ交換方式を特にパケット交換方式と呼んでいる。

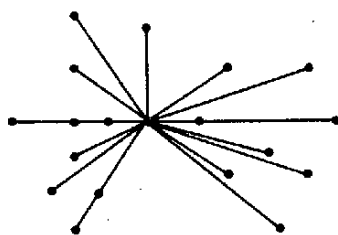
3.1.3 制御方式

以上、サブネットの基本構造に関わる結合方式、データ伝送方式について述べてきたが、もう一つサブネットの性格を決定する重要な要素としてサブネットの制御方式がある。これには、中央の管理局がサブネット動作を規制する中央制御方式と各々が自律して動作する分散制御方式があるが、中央制御方式ではサブネットの生命は中央の制御コンピュータにかかることになるため、この部分に障害が発生すると他の要素コンピュータが健全であってもサブネット全体の運転を停止しなければならないことになる。分散制御方式では各々が自律であることからいくつかの要素コンピュータが故障してもその部分を除けば他は従来通りの運転が可能である。この様な事情からサブネットの全体としての信頼度を保つことを重要視しサブネットの制御方式としては分散制御方式を採用することにした。

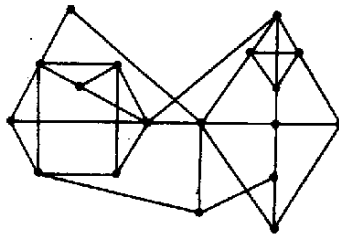
3.2 サブネットの位相幾何学的構造

ネットワークを形態的に分類すると図3-4の様な各種の形態が考えられる。一般にこれらのネットワーク形態は使用目的、地理的条件、コスト等の立場から選択されることになる。この中でもメッセージ交換方式で一般にとりあげられるものは、星状網、完全結合網、分散網である。

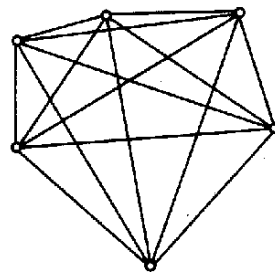
完全結合網はすべてのノード・ポイントが回線で完全結合されているため伝送速度、信頼性の点で最も優れたものになっている。しかしネットワーク規模が拡大した時の回配コストが莫大なものとなるため実現が困難である。星状網では、回線コストはそれほど問題にならないが中央のノード・プロセッサにトラフィックが集中するため、ネットワーク・トラフィックが大きくなるとすぐ飽和してしまうと云う難点がある。又中央のノードに障害が発生するとすべての交信が途絶えてしまう。回線障害に関しても、一地点に一回線しか割り当てられていないから回線障害が発生すると必ずどこかのノードが通信不能状態になる。分散網は完全結合網と比較して伝送速度に劣るが経済性に優れたものであり又2点間に複数の径路が設定できると云う点で信頼性に於ても満足できるものと云える。



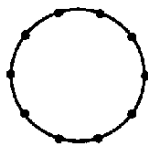
星 状 網



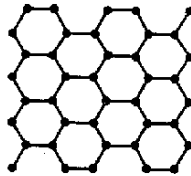
分 散 網



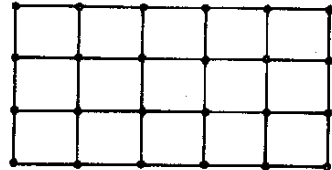
完全結合網



環状網



蜂巣状網



格子状網

図3-4 各種ネットワークの形態

図3-5は、現在、我々が採用しているネットワークの形態を示したものである。現在はまだ要素コンピュータの数が少く、又距離的にも接近しているため2点間完全結合の形式をとっているが、将来の拡張性と云う観点からは基本的に分散網を指向したものである。

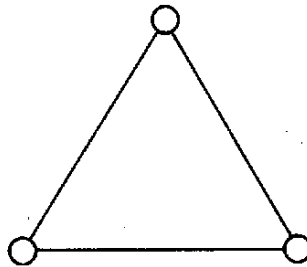


図3-5 JIPDECのネットワーク・トポロジー

ここで、将来の拡張性と云う事から一般の分散網の位相形態に関する最適化手法について考案を加えておく必要がある。

始めに設計目標として、ネットワークの良きの基準をどこにおくかを考えなくてはならない。これには一般に次の様なクライテリアが挙げられている。

- ① ビットあたりの転送コストが安価であること。
- ② 信頼性が高いこと。(例えば2点間に必ず2つの経路が保証されていること)
- ③ メッセージの平均伝送遅れが短いこと。
- ④ スループットが高いこと。
- ⑤ 負荷変動にも耐えるようにある程度容量に余裕を持たせること。
- ⑥ 新たなノード、回線の追加などの拡張が容易であり、かつ低コストであること。

ネットワークの最適化の問題はこれらの各種のパラメータで規制される条件下に於て、与えられたノードの間に回線をどのように配置し、その容量の配分をどのように決定するかと云う形で定式化できる。しかしこの問題を扱う過程にはいくつかの困難がある。例えば、上記の各種パラメータの間のトレード・オフをどのように考えるか、即ちネットワークの良きそのものに対する定義をどう考えるかと云うこと自体が出発点の問題としてある。具体的に云えば、同じコスト条件の下で、Aは高いスループット・レートを保証するが短いメッセージに対する応答速度がやや劣る又BはA

と逆に最大スループット・レートはそれほど高くないが、短いメッセージに対する応答特性が優れている、このような場合我々は A、B どちらのネットワークを優れているとすればよいだろうか？ 又、ほんのわずかなコスト上昇を認めれば極端に特性の優れたネットワークが実現可能であるとすれば、このような場合一体どちらのネットワークを採用すべきであろうか？

このようにネットワークの良さに関する定義自体が現在のところ余り明確でない分である。又数学的に扱う上での困難な問題として、与えられたノード数、リンク数を作って構成されるネットワークの種類はノード数、リンク数が増すと非常に大きな数となることが指摘される。今、NノードにM回線を割り当ててネットワークを構成する時を考えてみる。この時その場合の数は次式のようになる。

$$\frac{(N(N-1)/2)!}{(N(N-1)/2-M)!M!}$$

仮りにノード数が9個で回線の数11の場合を考えてみると、構成可能なネットワークの種類は実に600805296通りとなることが分る。

このように、ネットワーク規模がある程度大きくなるとそのすべての場合について検証を試みることは現在の計算機をもってしても不可能と云わねばならない。

その他の困難な問題として要素のディスクリート性（例えばチャネル容量は12KB, 24KB, 48KBと云う風に変化する）があるがこれは整数プログラミングによる解法に頼らねばならない。又コスト関数、時間遅れ関数、信頼性関数等はすべて非線型関数であり、これも解法を困難にする一因となっている。

以上述べた様な事情から、最適化に関する完全な解析的手法は現在のところまだ得られていない。

現在、行われているアプローチは、いくつかの前提のもとに各変数間の関係を解析的に、又はシミュレーションにより把握し、これらの解析的手法の集りにヒューリスティック・アルゴリズムを適用して効率的に解を求めてゆこうとするやり方である。例えば、ARPAネットワークのトポロジー設計を担当したNAC社（Network Analysis Company）では信頼性、遅延時間等のパフォーマンス・パラメータで制約されるネットワークから低コスト・ネットワークを見つけ出すヒューリスティック・プログラムを開発している。このプログラムでは、入力として①既存のネットワーク構成、②ノード間のトラフィック見積り、③短いメッセージに対する最大の平均遅延時間、④最大コストを与えこれらの条件を満たすネットワークについて①ネットワーク・トポロジー②月額コスト③最大スループット④平均トラフィック⑤ビット当りコスト⑥平均メッセージ遅延時間等が出力されるようになっていいると云う。

これに関連して、H・FRANKは、実行可能解（与えられた制約条件を満足するネットワークの集合）を求め、これに局部変換と呼ばれる操作を施して最適化を計る効果的な手法を発表している。

部分的な解析の面では、メッセージ遅延時間のモデル及び各種コスト関数を用いたチャネル容量

の最適配分の問題がKLEINROCKによって解析的に導かれている。

これらの手法のいくつかに関してはその詳細を3-4にゆずり、ここでは信頼性の評価に関する1つの手法と、それを現在の我々のネットワークに適用した結果について簡単にふれておこう。

ネットワークの信頼性を問題にする時、まずそれをどのような尺度で表現すればよいかと云う点から考えてみる必要があるだろう。初めにどのような状態をもってネットワークを故障とみなせばよいかを考えてみる。例えば図3-6の様なネットワーク構成に於て回線2, 3が切断した時、これはネットワークの故障と云えるだろうか。この様な場合には多少はパフォーマンスが低下するがA, B, C, D間の交信には何等支障が無い筈である。一方1, 2が切断した時はどうであろう。

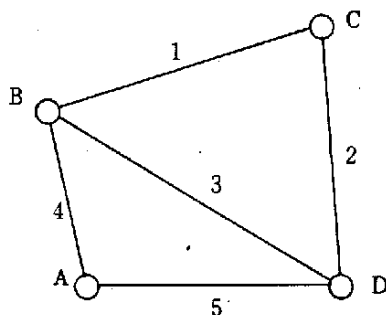


図3-6 ネットワークの例

この場合には明らかにCがあらゆるノードと交信不能におちいる。このように考えるとネットワークの故障としては後者の場合をとりあげるのが妥当であると云えよう。

このような観点から現在一般に用いられている信頼性の尺度に平均交信不能ノード対の数 $N(P)$ 及びディコネクションの発生する確立 $h(P)$ がある。

即ちネットワークの構成要素(回線, IMP)の故障率を P とした時に交信不能なIMPの対が平均的に何個あるかと云う事、又は交信不能なIMPの対が発生する確率はいくらになるかと云う事でそのネットワークの信頼性を計ろうとする方法である。

任意の構成のネットワークについて $h(P)$, $n(P)$ を求める手法として良く知られたものにMOORE-SHANNONの方法があるが、ここでは最も単純な方法としてモンテカルロ・シミュレーションによる解法について説明しよう。

今、ネットワークを構成するノード・コンピュータの故障率を α 、回線の故障率を β とする。まず、ネットワークの各要素 $n_1, n_2, n_3, \dots, L_1, L_2, \dots, L_3, \dots$ に対して各々対応する乱数を発生させ、ノードについてはもし対応する乱数 r_i が $0 \leq r_i < \alpha$ であればそのノードは故障とみなす。又回線については $0 \leq r_i < \beta$ であればそのラインは故障とみなす。この様にして各ノード、各ラインの故障を吟味し、故障した要素を取り除いたネットワークについてその交信不能ノード対を調べる。以下この手順を十分な精度が得られるまで繰り返せばよいわけである。

図3-7は、現在の我々のネットワーク構成についてこの手法によって求めた信頼性カーブを示

したものである。

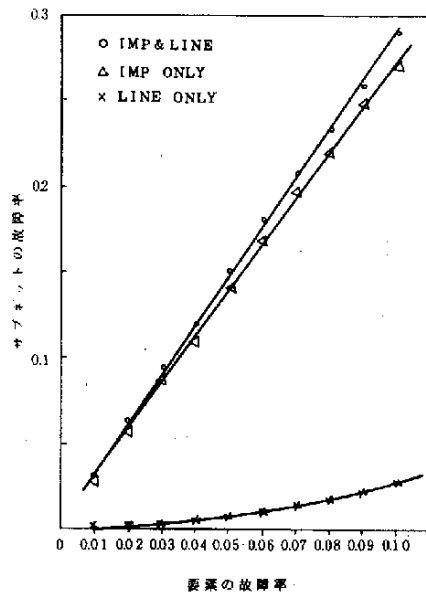


図 3-7 サブネットの信頼性カーブ

横軸は要素の故障率、縦軸はサブネットの故障率を表わしたものである。又図中*、△、○のマークはそれぞれ回線の故障のみを考えた時のカーブ、IMPの故障だけを考えた時のカーブ、両者の故障を考慮した時のカーブをあらわしている。例えばIMP、回線の故障率を共に0.01とするとサブネットの故障率は約0.05となることが分る。

3.3 IMPの機能

サブネットは発信地ホストから目的地ホストに信頼度よくビットを伝送するコミュニケーション・システムとして動作する。この様な高信頼度、高能率なデータ伝送機構を実現するために、我々はサブネットワークのソフトウェアに関して、その設計目標として次の様な基本的なねらいを設定した。

- ① サブネットの動作はホスト・コンピュータの動作と独立であること。

サブネットとホスト・コンピュータとの間に論理的な支援関係があってはならない、即ちホスト・コンピュータがサブネットの論理的動作を変える様なことがあってはならない。このようにすれば、ホスト・コンピュータに発生した障害が原因でサブネット機能が停止するような事態が避けられることになる。

- ② IMPは相互に依存関係を持たないこと。

一つのIMPのダウンがサブネット全体の機能を停止させることが無いようにするためには、IMPは各々自律な系として動作することが望ましい。

- ③ 会話型処理を可能にするためには、短いメッセージに対して十分な応答特性が得られること。会話型処理に於ては、人間の我慢できるレスポンスの限度は約2秒であると云われている。

ARPAでは、この内1秒が低レベル・ネットで費やされるとして、サブネットの平均応答特性は0.5秒を目標値としている。現在達成されている値は約0.2秒であると言う。

- ④ ファイル・トランスファー等の利用形態に対しても十分耐え得るように、大量なデータ伝送に対しては十分なスループット・レイトを保証すること。

- ⑤ ネットワークの拡張性を考慮したソフトウェアであること。

サブネットの拡張時に、新たなソフトウェアを制作したり、従来のソフトウェアを大々的に修正したりすることが無いようにするために、IMPのソフトウェアは全て同一にする。又、動作アルゴリズムがネットワーク・トポロジーに依存するようなことがなるべく無いようにする。現在我々がIMPとして採用している機種はN-3200/50であるが、ノード・プロセッサを同一機種で統一することはソフトウェアの統一化と云う観点からも必要である。

以上述べたような目標を達成するために、IMPは本来の蓄積交換機能に加えて、効率、信頼性を支えるための各種の機能を遂行しなければならない。これには、パケットを、最適径路を選んで送り出すためのルーティング機能、ネットワークの混雑（Congestion）を緩和するためのフロー・コントロール機能、メッセージのシーケンスを保証するシーケンス・コントロール機能、サブネットの障害に対処するための障害対策機能、IMPの自己保存機能、パフォーマンスに関する有用なデータを収集するモニタリング機能等の各種の機能が含まれている。以下ではこれらの機能の各々に関して、その動作機構を説明するが、これはネットワーク・ソフトウェアと云う観点からは、HOST-IMPプロトコル、IMP-IMPプロトコル、発信地IMP-目的地IMPプロトコルに関する記述にあたるものである。

3.3.1 蓄積交換（Store and Forward）

サブネットはホストから受け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IMPに送る。目的地IMPではこれをもとのメッセージにリアセンブリ（再編集）して目的地ホストに渡す。以下では、メッセージが発信されてから目的地に受け入れられるまでの過程について、蓄積交換機能を中心としたややマクロな解説を行う。その他関連した項目としてメッセージ、パケット、パケット化、ACK、リアセンブリ、RECEIVE等についても説明を加えるが、話を分り易くするために簡略化した部分があり厳密さに欠けるところがある。これらの点については、後半でその厳密な規定を与えることになる。

A メッセージ

ホスト-IMP間でやりとりされる情報の単位をメッセージと呼ぶ。メッセージは行き先ホストに関する情報を含んだリーダーと呼ばれる部分とメッセージ本体とから総成される。そのフォーマットは図3-8のようになっている。

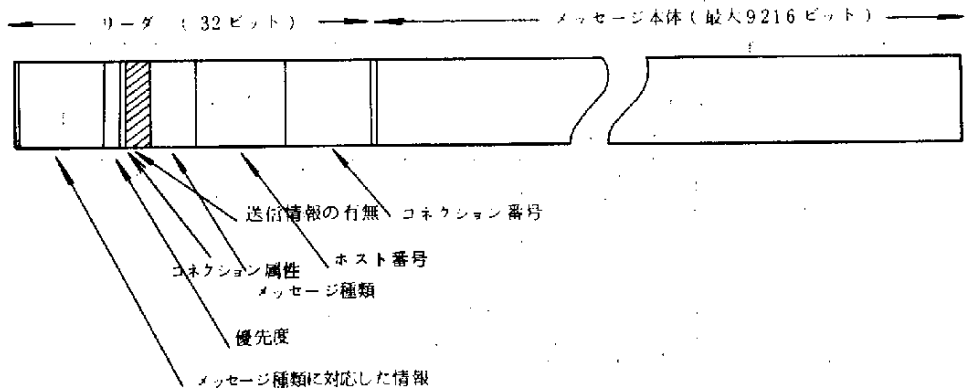


図3-8 HOST→IMPメッセージ・フォーマット

メッセージ長の最大は主としてホスト・バッファ、IMPバッファの制約から決定されるが、現在のところ9216ビットに設定してある。

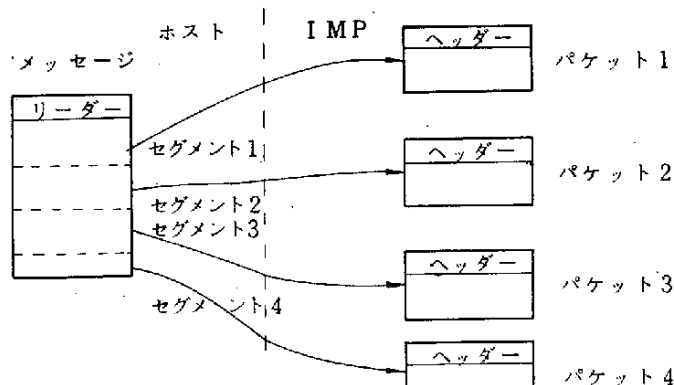
メッセージ長はこの範囲で可変長である。

B パケットおよびパケット化

IMPはホストからセグメント化された形式でメッセージを受けとりこれをサブネット内の伝送単位であるパケットに分割編成する。

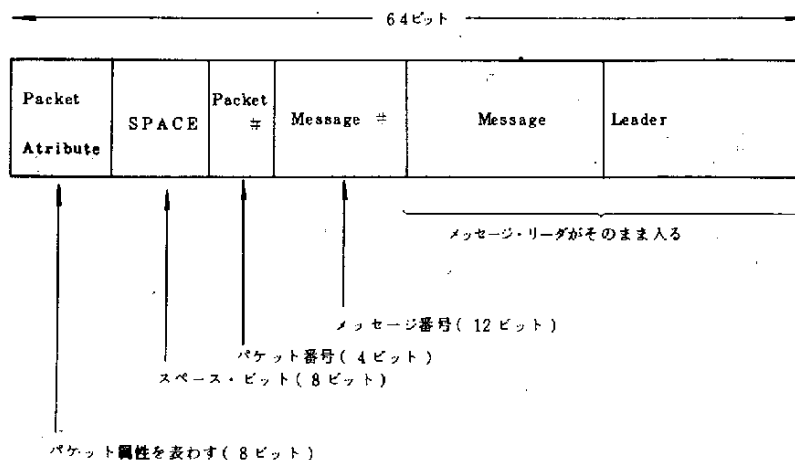
セグメントとはホスト-IMP間のフィジカルな伝送の単位である。即わち、IMPはホストから受けとったセグメントにヘッダーと呼ぶ制御情報を付加してパケット化する。

図3-9はメッセージがパケット化される過程を図示したものである。



パケットのヘッダは64ビットから構成され、これにはメッセージのリーダ情報の他、パケット種類に関する情報、パイプ番号、パケット番号、メッセージ番号、ACK情報等が含まれている。

図3-10はパケット・ヘッダのフォーマットを示したものである。



ACK語はヘッダの直後の16ビットを占めその内容は次のようになっている。

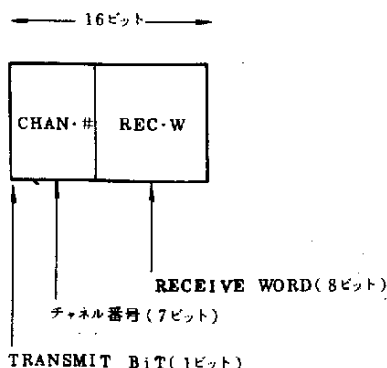


図3-10 パケット・ヘッダ・フォーマット

パケット本体はメッセージの有効情報ビットから構成され、セグメントそのものに対応する。パケット長は、1152ビットの固定長であるが、これに満たないメッセージや、最終パケットは可変長で扱う。

C 受信確認情報

メッセージを受け入れた発信地IMPはこれをパケット化し、目的地IMPに向けて送り出すが、パケットがサブネット内でIMPからIMPに手渡される手順は蓄積交換と呼ばれる伝送手順に従う。蓄積交換では、送信側IMPは送信バッファに入っているパケットを隣接IMPに送り出すと、そのパケットを直ちに棄却せず、隣接IMPから正しく受取ったと云う確認情報ACK

(Acknowledge) が返ってくるまで、保持している。従ってパケットを受け取った IMP は必ず受け取った回線を通して確認情報 ACK を返さなければならない。このように IMP-IMP 間の 1 回のパケット伝送は ACK の受信によって完了する。もし伝送の過程でエラーが検出されると ACK が返送されない。このような場合は送信側 IMP は同じパケットを再送することになる。又伝送エラー以外の場合にも、例えば受信側 IMP のバッファが満杯の時は受信側 IMP は受け取ったパケットを棄却し故意に ACK を返さない。この場合も送信 IMP はバッファが空くまで何度も再送を試みることになる。ここでは蓄積交換の基本動作を分かり易く説明する意味で、1 パケットに対して対応する ACK が 1 つ返送されることを前提として話を進めてきた。しかし実際に我々が採用している方法は、数パケットまとめて ACK を返信する、(しかも通常の逆方向トラフィックに便乗させる) より効果的な方法である。

この手順の厳密な規定は「(2)パケットの確認、再送手続き」の項で述べる。

D リアセンブリと RECEIVE

発信地ホストから出されたメッセージは発信地 IMP でパケット化されいくつかの IMP で蓄積交換を受けて最終的に目的地 IMP に到着する。目的地 IMP では次々に到着してくるパケットをリアセンブリ・バッファに格納し、これをパケット番号に従って順序をそろえていく。時間的に前に出されたパケットが必ずしも先に到着すると云う保証は無いから、目的地 IMP に到着してくるパケットの順番は全くランダムであると考えなくてはならない。このようにしてすべてのパケットのシーケンスがそろってこれをもとのメッセージ・フォーマットに変換してホストに伝送する。

これを受け取ったホストはメッセージ受信の確認メッセージ RECEIVE を発信地ホストに向けて発信する。発信地ホストは RECEIVE を受信して 1 回のメッセージの転送の終了を知る。

ここでも話を簡単にするためリアセンブリの終わったメッセージが無条件でホストに送られると述べたが実際には、メッセージのシーケンス・コントロールの指示に従った順序でホストに送られる。この規定については「3.3.5 メッセージのシーケンス・コントロール」の項で述べる。

図 3-11 は ACK, RECEIVE の関係を図示したものである。

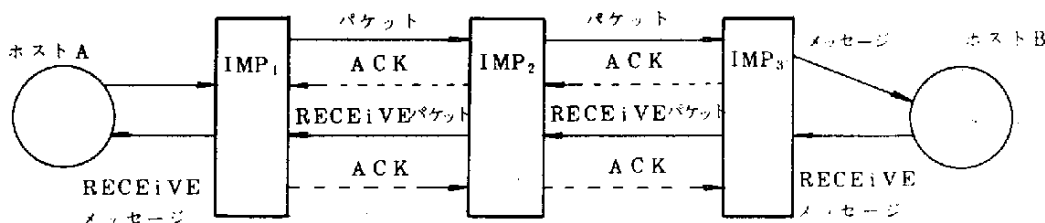


図 3-11 ACK, RECEIVE の関係

3.3.2 パケットの伝送確認手続き、再送手続

前節では蓄積交換機能を説明する便宜上パケットが送信されると、受信 IMP が直ちに対応する受信確認信号 ACK をパケットとして返信すると述べた。この手法は初期の ARPA ネットワークで採用されていた方法であるが、オーバーヘッド・トラフィックが多すぎると云う理由から現在は使用されていない。

ACK は明らかにデータ伝送上の有効情報ではないから、このようなネットワークの制御情報がネットワーク負荷としてかかることはネットワーク効率上、好ましくない分である。

これを改善するアプローチとして次のような方法が考えられる。

○パケットを受信しても直ちに ACK を返さず数パケットまとめてから 1 つの ACK として返す。

○ACK を逆方向のトラフィックに便乗させる。

現在の ARPA ネットワークでは、この両方の手法をとり入れて効率の改善を計っている。

我々の採用している方法もほぼ ARPA の手法に従ったものである。以下、この手順について説明を加える。

まず、IMP と IMP を結ぶ 1 つの回線を 8 つのロジカルなパス (チャンネル) に分割し、その回線を利用するパケットをその空いているチャンネルの一つを使って伝送すると云う機構を実現する。

8 つのチャンネルには、図 3-12 に示すように、両方向に、それぞれそのチャンネルの使用状態を示すチャンネル・ユース・ワード (Channel Use Word) と送信状態を記録するトランスミット・ワード (Transmit Word) と受信状態を記録するレシーブ・ワード (Receive Word) が対応している。(図では、簡略化のため 4 チャンネルで説明をする)

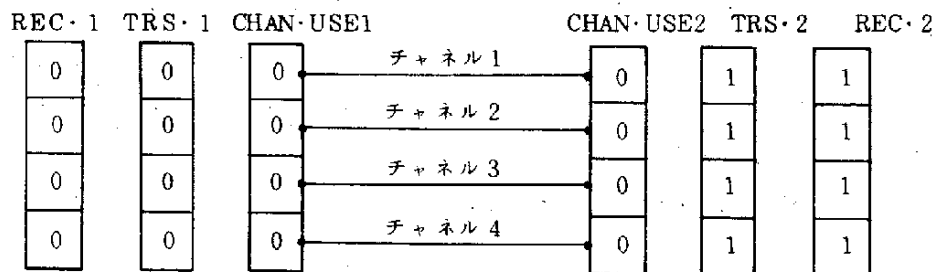


図 3-12 チャンネル

初期状態に於て各々のワードの対応するビットは図の様に初期化されている。即わちチャンネル・ユース・ワードはそのチャンネルが全て未使用状態であるから、この状態を全ビットオフで表わしている。又トランスミット・ワード、レシーブ・ワードは双方の対応するビットが反転するように初期化されている。

今 IMP 1 から IMP 2 に向けてパケット転送が要求されると、まず IMP 1 はそのチャンネル・ユース

ス・ワード CHAN・USE 1 を調べて空いているチャンネルがあればそのチャンネルを割り当て、対応するビットをオンにする。又パケット・ヘッダにトランスミット・ワード TRS 1 の対応するビットを記録し又レシーブ・ワード REC・1の全ビットと使用したチャンネル番号もパケット・ヘッダに記録してそのパケットを IMP 2 に向けて送りそのパケットをチャンネル 1 に関して ACK 待ちの状態に入れる。ここで、ヘッダに記録する情報を chan#, TRS・1, REC・2 の形式で表わすことにすると、今チャンネル 1 を使ってパケットを送出すると図 3-13 のような状態になる。

IMP・1

IMP・2

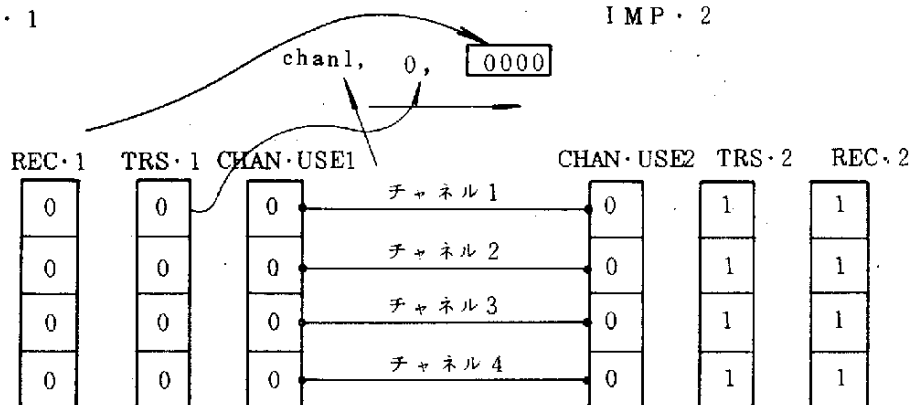


図 3-13 IMP 1 からチャンネル 1 を使って IMP 2 にパケットを送り出した時 (パケットは未だ到着していない)

このパケットが IMP 2 に到着すると IMP 2 はまずパケット・ヘッダから TRS・1 の情報を取り出し、レシーブ・ワード REC・2 の対応するビットと比較する。もし両ビットのマッチングがとれれば、送られてきたパケットは、既に受信済みの重複パケットとみなし何もしないでそのパケットを棄却する。もし両ビットが逆の関係にあればこのパケットを受け入れ REC・2 の対応するビットを反転させる。次に、パケット・ヘッダに記録されている REC・1 を取り出し、トランスミット・ワード TRS・2 と全ビットの比較を行う。もしマッチングのとれたビットがあればそのビットに対応するチャンネルに関する ACK 信号とみてまず TRS・2 のそのビットを反転させそのチャンネルを通して IMP 2 → IMP 1 に送られた ACK 待ちのパケットを解放し又チャンネルを解放する。即ちチャンネル・ユース・ワード CHAN・USE 2 の対応するビットをオフにする。図 3-14 は前の例で、パケットが IMP 2 に到着した時の状態を示したものである。

IMP・1

IMP・2

chan1, 0, 0000

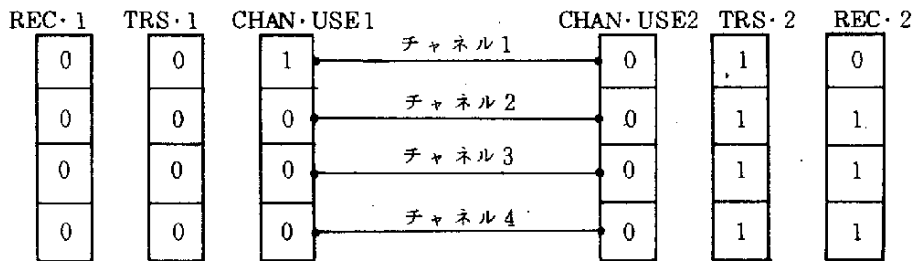


図3-14 パケットが到着した後の状態

同様にして、IMP 1からチャンネル 2を使って次のパケットをIMP 2 に送った時の状態を図3-14'に示す。

IMP・1

IMP・2

chan2, 0, 0000

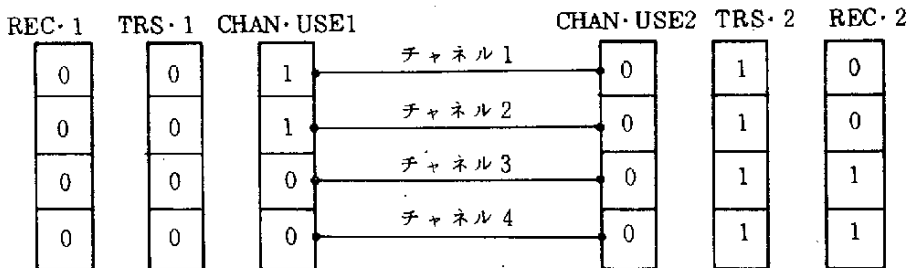


図3-14' IMP1からチャンネル 2を使って次のパケットをIMP 2
に送った時の状態

又図3-15, 図3-16はこの状態でIMP 2からチャンネル 1を使ってIMP 1にパケットを送った時及び到着した時の状態を示したものである。図中・印が付けられたものはACKを受けたことを意味している。

chan1, 1, 0011

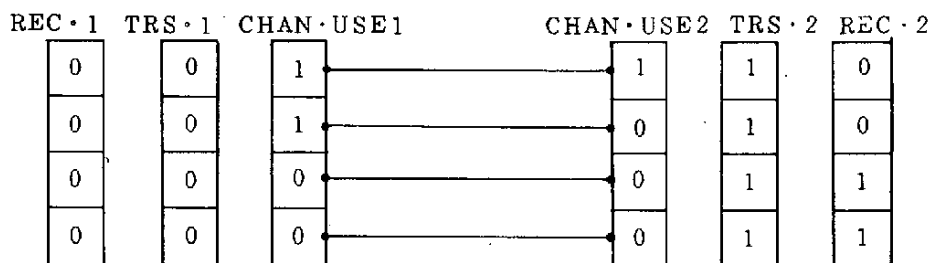


図3-15 IMP2からチャンネル1を使ってIMP1にパケットを送り出した時(パケットは未だ到着していない)

IMP · 1

IMP · 2

chan1, 1, 0011

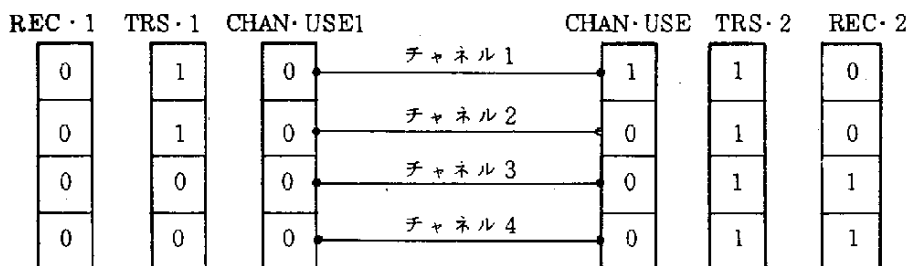


図3-16 パケットが到着した後の状態

このようにして、ACKはパケット単位で返信せず数パケットまとめて、逆方向のノーマル・トラフィックに便乗させて扱うことができる。

一方向的な流れしかないような場合、例えばIMP1からIMP2に向うパケットはあるが、IMP2からIMP1に向うパケットが無い場合は、IMP2はその回線のトラフィックを監視していても、送信キューも未確認パケット(ACKを受信していないパケット)も無くなった場合は、ACK情報のみをヌルパケット(パケット・ヘッダのみ)として送出する。

又IMPは送信キューからパケットを取り出してそれをチャンネルに割り当てる時、1, 2, 3, 4, …… 8, 1, 2, 3, …と云う風に必ずサイクリックにその使用状態を調べ割り当ててゆくようにする。もしこの過程で利用状態にあるチャンネルにぶつかった時は、それは1周期前に発信されたパケットであるにもかかわらず、ACKを受信していないと云う事になるから、そのパケットを再送してから次のチャンネルを割り当てることを試みるようにする。

送信キューが無くなった状態に於てもチャンネルをサイクリックにチェックし未確認のパケットがあればこれを再送する。この手順を、送信キュー、未確認パケットが無くなるまで続ける。

3.3.3 ルーティング

サブネットに流入したメッセージは発信地 IMP でパケット化され、各々のパケットは独立に目的地 IMP に向けて送られる。これらのパケットを受け取った中間ノードはそれが自己のホストに向けられたパケットでなければ、次の IMP にそれを伝えるために適当な回線を選んでこれを送出しなければならない。パケットを目的地にできる限り早く届けるためには、最適な経路を選ぶための適当な決定機構が必要になる。このための機能をルーティングと呼ぶ。ルーティングでは、現在地から目的地までの物理的な経路長による時間見積りだけではなく、中間ノードに於ける現時点のトラフィック状態の見積り及び、中間ノードの将来のトラフィック状態の見積り等に基いた決定機構が必要である。しかし将来のトラフィックの状態は統計的な形でしか把握できないからある時点での判断はあくまで予測に過ぎず、決定論的な判断は下せないと言うことになる。

ルーティングの手法に関しては、これまでも多くの方法が提案されているが、これは一般に、次のような要求にかなうものが望ましい。

- 最短時間でメッセージを目的地に送ることを達成すること。
- ネットワーク・トポロジーが変更されてもアルゴリズムの修正を加える必要が無いこと。
- これはネットワークの一部に障害が発生してもうまくこれに適応してゆくことも含んでいる。
- ネットワークのトラフィック変動にもうまく適応してゆくこと。
- 一時的に混雑状態にあるノードはなるべく迂回してやること。
- 処理アルゴリズムがなるべく簡単で計算負荷を要しないこと。

ルーティング手法に関しては、Boem, Baran, Kleinrock 等の研究があるが、ここでは Fultz の行った研究をもとにして、従来の手法を簡単に紹介し、各手法の比較、評価を行ってから現在我々が採用している手法について説明することにする。

現在、提案されているルーティング手法は、決定論的な手法 (Deterministic Routing Technique) と確率論的な手法 (Stochastic Routing Technique) の2つのカテゴリに分けられる。図3-17は各種の手法をこのカテゴリに従って分類したものである。以下この中の主なものについて説明を加える。

① Flooding (波紋法)

IMP はパケットを受けるとそのすべての出力回線にそのパケットのコピーを送り出す。それを受け取った隣接 IMP も、それが以前に受け取ったものでなければ、同様にしてそのすべての出力回線にそのコピーを送り出す。それが以前に受け取ったものであればそのパケットを棄却する。このようにして1つのパケットは次々と波紋状に目的地に送られてゆく。目的地 IMP では最も早く到着したパケットを取り入れ、後から到着してくる重複パケットは棄却してゆく。(All はすべての出力回線にパケットのコピーを送る方法、Selective は限られ

た数の出力回線に送る方法)

決定論的手法 (Deterministic Algorithm)

- ① Flooding
 - all
 - Selective
- ② Fixed Routing
- ③ Split Traffic
- ④ Ideal Observer

確率論的手法 (Stochastic Algorithm)

- ⑤ Random Routing
- ⑥ Isolated
 - Local Estimate
 - Shortest Queue + Bias
- ⑦ Distributed
 - Periodic Updating
 - Asynchronous Updating

図 3-17 ルーティング手法の分類

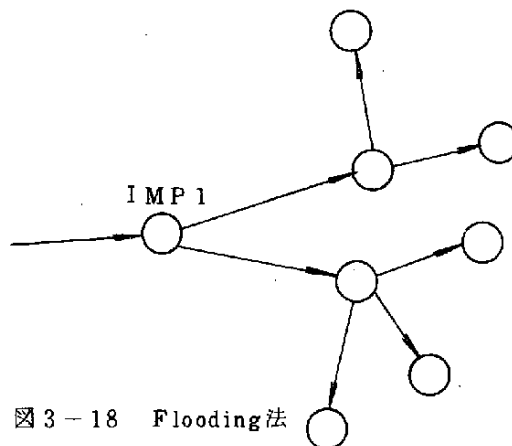


図 3-18 Flooding法

この方法の長所としては、処理アルゴリズムが単純なこと、トポロジーの変化に対して安定なこと、信頼性が高いこと(途中でパケットが失われても重複パケットがある)などの点あげられる。しかし無駄なトラフィックが多いためそれによるネットワーク負荷が増大し効率が著しく低下する。又一度送ったパケットを記憶しておかなければならないと云う問題もある。

② Fixed Routing 法 (固定ルーティング法)

固定ルーティング法では、あらかじめ、ある地点からある目的地にゆくための最短経路を静的に見積っておき、各 IMP ではそれに従ったユニークなパスを選択してパケットを送り出す。最短経路の見積りには例えば目的地に最小ホップ数で行く経路等が用いられる。その他回線容量等も考慮に入れなくてははいけない。

各 IMP は図 3-19 の様なルーティング・テーブルを用意していて、パケットが到着するとその目的地を調べて、テーブルの対応する回線にパケットを送り出す。例えば図の例ではホスト 5 に向うパケットを受けとったら、このパケットは、回線 1 を通して送出されることになる。

目 的 地	回 線 番 号
ホ ス ト 1	2
ホ ス ト 2	1
ホ ス ト 3	1
ホ ス ト 4	3
ホ ス ト 5	1
ホ ス ト 6	2
ホ ス ト 7	3
ホ ス ト 8	3

図 3-19 固定ルーティング法のルーティング・テーブル

この方式の長所は処理アルゴリズムが単純であることと、動作が安定していることである。欠点としてはネットワーク・トポロジーの変化に対して弱いこと及び、一時的なトラフィック変動に対して適応性が無いことがあげられる。

③ Split Traffic Routing

固定ルーティングでは目的地に対して 1 本の回線がユニークに対応するため、その径路にパケットが集中すると送信キューが大きくなりキューイング・ディレイによる遅れが大きくなってくる。この効果を緩和するために目的地に対して何本かの回線を候補として選びその回線に適当にトラフィックを分散させる方法がある。これを Split Traffic Routing と呼ぶ。

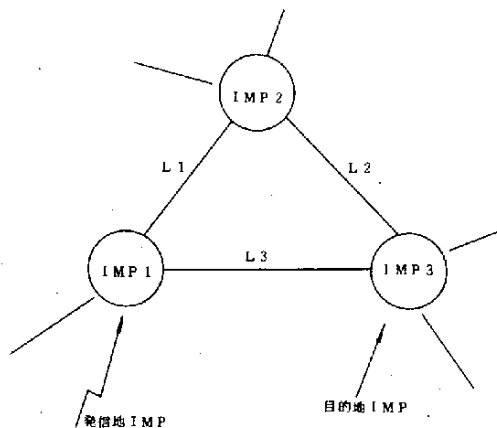


図 3-20 Split traffic

例えば図3-20の様なネットワークの場合 IMP1からIMP3にパケットを送る時、その経路としてL1 — IMP2 — L3 とL3の2通りを候補と考え、IMP1では回線L1、回線L2にそれぞれトラフィックを1:2の割り合いで分流させる。

この方法は固定ルーティングの短所を一部補うものであるが、基本的には同じような問題点が指摘される。

④ Ideal Observer

この方法はホストから新たなパケットがノードに流入するたびに、ネットワークの現在のトラフィック分布に関する完全な情報とネットワークのトポロジーに関する情報をもとにしてそのパケットが目的地に到達するまでの時間を最小にするようなルートを計算して求める。しかしある時点でのネットワークの状態に関する情報を完全に把握することは不可能だから、この方法は実際には実現できない。他の方法を評価するデータを得る目的から理論的に求めるだけの意味しか無い。

⑤ Random Routing

ランダム・ルーティングとは、出力回線を選ぶ時に、ある確率分布に基いてこれを決定する方法を云う。例えば最も単純な方法では、目的地に無関係に全くでたらめに回線を選んでそこに送り出すやり方がある。この方法は、ネットワークのトポロジーの変化に対して安定であること又処理アルゴリズムが単純であること等のメリットがあるが、パケットの時間遅れが大きく余り効果的でない。

⑥ Isolated法

Isolated法と次に述べるDistributed法は、その動作機構に於て非常に類似している。IMPは図3-21に示される様な形式のディレイ・テーブルとルーティング・テーブルを備えている。ディレイ・テーブルはある目的地に行くために、それぞれの回線を使った場合どれだけ時間を要するかと云う遅れの見積りを記録している。図では行が目的地を表わし列が回線を表わしている。表の中の要素 $D(n, \ell)$ は目的地 n に行くために回線 ℓ を利用した時の遅れを意味している。

例えば目的地3に行くために回線2を使うと $D(3, 2) = 0.3$ となることが分る。パケットをルーティングする時はこの表を使って目的地に最短時間で到着するための回線を選んでよいが、パケット毎にこの操作を繰り返すのは無駄なのであらかじめ各目的地ごとに最短時間で到着できるような回線を抜き出してまとめておく。これがルーティング・テーブルである。即ちルーティング・テーブルの各要素 $R(n)$ は次式で決定される。 $R(n) = \min_{\ell} (D(n, \ell))$ ($D(n, \ell)$ を最小にする時の ℓ の意味) 例えば目的地1に関してこの行を横にみてゆくとL4の0.1が最小であるから $R(1) = L4$ となる。

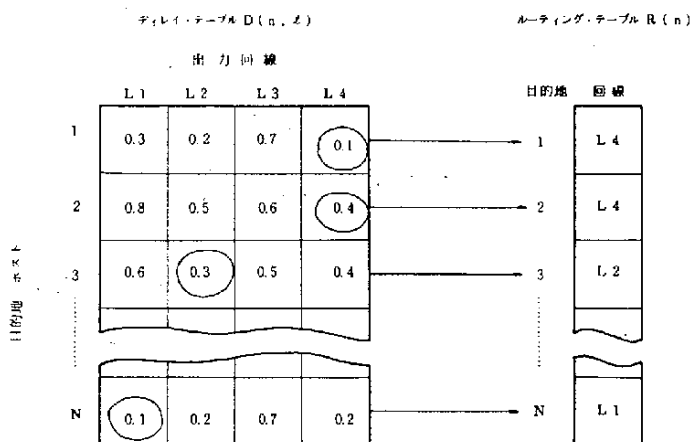


図3-21 ルーティング・テーブルとディレイ・テーブル

このように実際のパケットのルーティングは、ルーティング・テーブルを使って効率よく行われることになる。以上述べたことがIsolated法、Distributed法の基本動作にあたる。両者の違いはディレイ・テーブルのアップデート・メカニズムの違いにある。即ち、Isolated法ではディレイ・テーブルを修正する情報を自己の内部にある情報又は外部からきたノーマル・トラフィックのみに頼ることを原則とするがDistributed法ではIMP相互がディレイ・テーブルを修正するための特別な情報をやりとりする。

Local Delay estimateはBoem等の考えたバックワード・ラーニング(backward learning)を基本とした方法である。即ちすべてのパケットに発信地を出た時から現在に到る経過時間を記録させることにより、IMPは受信したパケットの経過時間を調べて逆にその発信地に行くにはどの程度時間がかかるかを知ることができる。この機構を式で表わすと次のようになる。 $D(n, \ell)_{NEW} = k_1 \cdot D(n, \ell)_{OLD} + k_2 \cdot T(n)$

k_1, k_2 : は重みを与える係数

$T(n)$: 発信地 n を出たパケットが現在までに過した時間

このようにIMPはディレイの見積りを、通過するパケットから教えてもらい、最新情報を保つことができる。実際には例えばトラフィックの局所的な集中や、ネットワーク・トポロジーの関係等から、条件が悪い時は正確な情報をつかめないような場合もあるし又パケットの経過時刻を管理するのが大変だと云った欠点もある。又パケット毎にアップデートを行うとすればこのための計算負荷も問題になる。

Shortest queue + Bias アルゴリズムでは外部から全く情報を取り入れずに、IMP内部に於ける送信キューの長さやバイアス項の和を回線ごとと比較し、その最小のものを選んでパケットを送り出す。この操作は1パケットごとに行われる。バイアス項を決定するのはバイア

ス・テーブルによるが、この構成はディレイ・テーブルと全く同じである。

		出力回線			
		L 1	L 2	L 3	L 4
目的地	1	0.3	0.2	0.7	0.1
	2	0.8	0.5	0.6	0.4
	3	0.6	0.3	0.4	0.5
	⋮				
	N	0.1	0.7	0.2	0.2

図 3-22 バイアス・テーブル $B(n, \ell)$

バイアス・テーブルの要素を $B(n, \ell)$ で表わすと、ルート選択を表わす式は次の様に表現できる。

$$L(n) = \min_{\{\ell\}} [B(n, \ell) + Q(\ell)]$$

$L(n)$: 目的地 n にゆくための出力回線

$Q(\ell)$: 回線 ℓ の送信キューのキューイング・ディレイ

バイアス・テーブルの各要素の見積り、即わら $B(n, \ell)$ は通常、無負荷状態のネットワークを考えて、その状態で回線 ℓ を通って目的地 n に到着するための最小時間として求めている。これは例えば Floyd のアルゴリズム等を用いて簡単に計算できる。その他の方法としてはネットワークを初期化する時（ネットワークが運転状態に入る時）に同期更新の手法を用いて IMP から IMP に情報を伝え合うことによって創成してもよい。ネットワーク・オペレーションの立場からは後者のやりの方が实际的であろう。

一段（IMP から次の IMP まで）あたりの消費時間 D_p は、IMP のパケット処理時間は一定と考えてよいから、回線容量、回線長がすべて同じであれば一定と考えて扱ってもさしつかえない以下ではこの場合を仮定して話をすすめる。

例えば図 3-23 では IMP1 のバイアス・テーブルの $B(3, L_3)$ は次のようになる。

$$B(3, L_3) = 2 D_p$$

ルーティング法に於て注意しなければならないことの 1 つにループの問題があるが、このアルゴリズムに関しても例外ではない。（固定ルーティング法以外のものはすべてのループの可能性はある）但しここで云うループとは一時的なループであってネットワークの状態が少し変

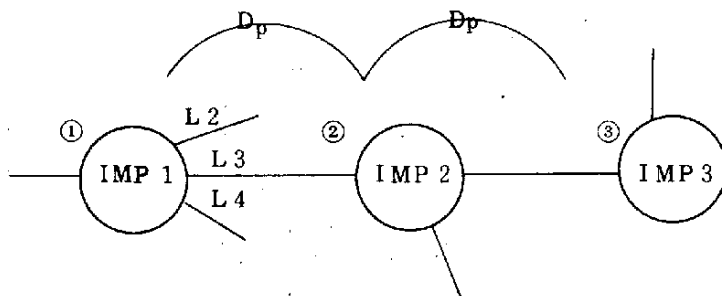


図3-23 バイアス項の見積り

化すれば解消する性格のものである。しかしループは明らかにパケットの到着時間遅れに関して悪影響を与えるものであるから、その防止対策を考えることは必要である。

SQ + BIASアルゴリズムでは特に D_p の値の設定がループに関して大きな意味を持つ。

図3-24はループの発生するケースを例示したものである。

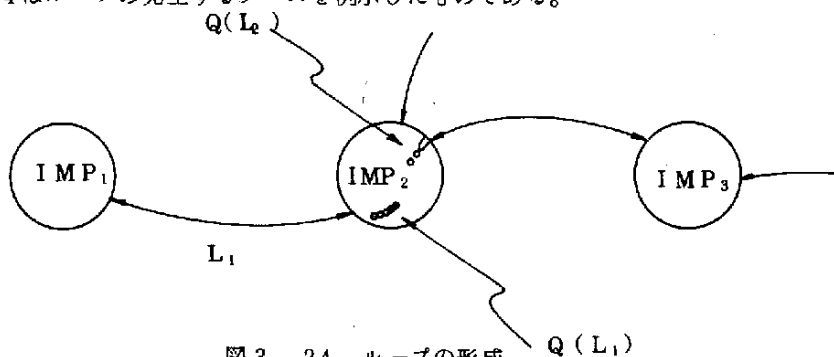


図3-24 ループの形成

図に於て、今パケット P_1 が IMP3 から目的地 IMP1 へ向けられて L_2 を通って IMP2 に入ってきたとすると、IMP2は出力回線として L_1 か L_2 を選択することになる。

これは次の2つの場合の時間の比較に於て決定される。

① L_1 を通って IMP1 へゆく場合の時間 $t_1 = D_p + Q(L_1)$

② L_2 を通って IMP1 へゆく場合の時間 $t_2 = 3D_p + Q(L_2)$

今仮りに $Q(L_1)$ が非常に大であった場合を相定すると $t_1 > t_2$ となる場合が生じる。この時そのパケットは再び元の回線 L_2 を通って IMP3 にもどることになる。そして再び IMP2 にやってきた時 $Q(L_1)$ が減少していれば $t_1 < t_2$ となり IMP1 に送られるが、意地の悪いケースを考えると他から流入したトラフィックで相変らず $t_1 > t_2$ が保たれている可能性があるから、この場合は再び L_2 を通って IMP3 にもどされることになる。この様に、悪条件が重なりと一時的なループを発生させる可能性がある。

これを抑制するためには D_p が大きな役割を果たす。

D_p が非常に小さい場合を考えると t_1 , t_2 の関係は $Q(L_1)$ の大小に対して大変敏感に

なる。即ち $Q(L1)$ が少し大きくなるとループを起す可能性が非常に大となる。

一方 D_p が大である場合は $Q(L1)$ の大きさは、 t_1 、 t_2 の関係に余り影響を与えなくなる。つまりバイアス項が支配的となりループを起す可能性は低くなる。しかしだからと云って D_p をどんどん大きくしてゆくとキューレングスは $t_1 - t_2$ の関係に対してほとんど意味を持たなくなりバイアス項だけで決定されることになるからこれは結局、固定ルーティング法に変わってしまう。

以上の様な事情を考慮した上で D_p の最適値を決定する必要がある。Fultz はキューイングモデルを使って D_p の最適値を解析的に導いているがこれは次式のような形で与えられる。

$$D_p = \frac{K_1}{1 - RE/K_2} \quad (RE \text{ はスループットを表わす})$$

又図 3-25 は実験的に相定したネットワークについて、シミュレーションを行って求めた D_p とメッセージ・ディレイの関係を示したものである。

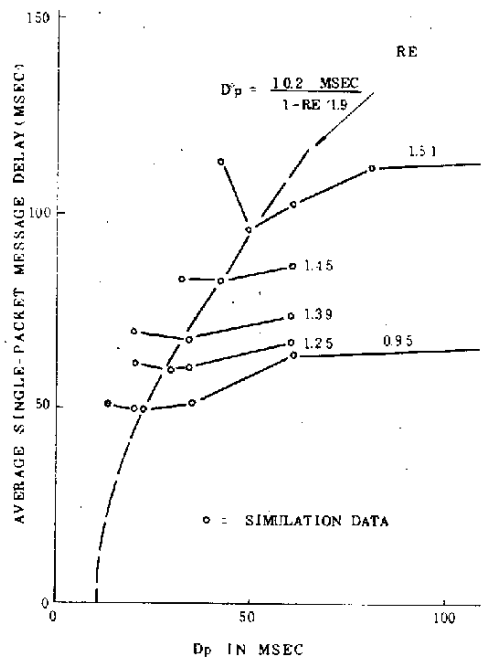


図 3-25 Fultz の求めた D_p とディレイの関係
(RE はスループットを表わしている)

SQ + BIAS アルゴリズムは、パフォーマンスは優れているが、パケット単位に処理を行わなければならないから計算負荷が大きくなるきらいがある。又バイアス・テーブルは固定であるからネットワーク・トポロジーの変化に非常に弱い。

⑦ Distributed 法

すでに述べたように Distributed 法も基本動作に於ては Isolated 法と変るところはない。即ち各回線を使って目的地に行くのに要する時間を記録したディレイ・テーブルとルーティング・テーブルを使ってパケット・ルーティングを行う。ディレイ・テーブルの更新に関しては、Isolated 法では自己内部にある情報だけをもとにして決定したのに対し Distributed 法では IMP 相互が更新情報をやりとりすることによって最新の情報を保持する。

この時更新情報をやりとりするきっかけが問題になるが、この方法の違いから Periodic Updating と Asynchronous Updating の 2 つの方法に分類される。Periodic Updating 法は初期の ARPA ネットワークで採用されていた方法である。

この方法では、各 IMP は、一定の時間間隔を置いて周期的に、ディレイ・テーブル・アップデート・ベクトルを隣接 IMP に送る。(ARPA ネットワークではこの時間を約 625 ms においている) アップデート・ベクトルは各 IMP が同期して送っても良いし又各々が独立に送っても良い。IMP はアップデート・ベクトルを受けとるまではルーティング・テーブルを使って固定ルーティングと同じ手順でパケット・ルーティングを行っている。

アップデート・ベクトルは図 3-26 に示されるような形式をしている。即ちある IMP i は、自分を通して各目的地に行くとき、最低どれだけの時間がかかるかという情報をアップデート・ベクトルにまとめる。

								From IMP i
目的地	1	2	3	...			N	
所要時間の見振り	0.5	0.7	0.8	...			0.4	

図 3-26 アップデート・ベクトル $U(n)$

アップデート・ベクトルの各要素を $U(N)$ とすれば $U(N)$ は次式で決定されることになる。

$$U(N) = \min_{\{i\}} D(n, \ell) \quad \left| \quad D \quad \left(D(n, \ell) \text{ の最小値の意味} \right) \right.$$

例えば IMP i のディレイ・テーブルを図 3-27 のように仮定すると、IMP i は図 3-28 に示されたようなアップデート・ベクトルを作成することになる。一方他の IMP からアップデート・ベクトルを受け取った IMP は、この各要素に、その回線のキューの長さと、IMP 一段あたりの処理遅れ D_p を加えてそれをその回線に関する新たな時間遅れの見振り値としディレイ・テーブルのその回線に対応するコラムをアップデートする。この関係を式で表わすと次の様になる。

出力回線

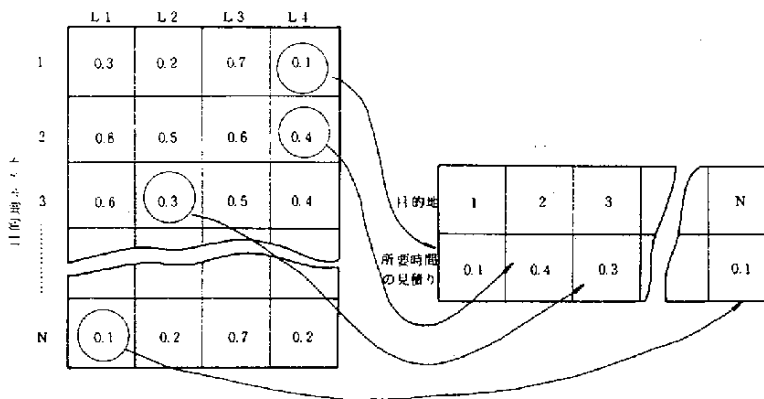


図 3-28 IMP_iの作成するアップ・

図 3-27 IMP_iのディレイ・テーブル $D(n, l)$

データ・ベクトル $U(N)$

$$D(n, l)^{\text{NEW}} = U(N) + Q(l) + D_p \quad (Q(l) \text{は回線 } l \text{ のキューイング・ディレイを表わす。図 3-29 参照})$$

この様にして各回線を通して受け取ったアップデート・ベクトルを用いてディレイ・テーブルは周期的に修正されてゆくことになる。

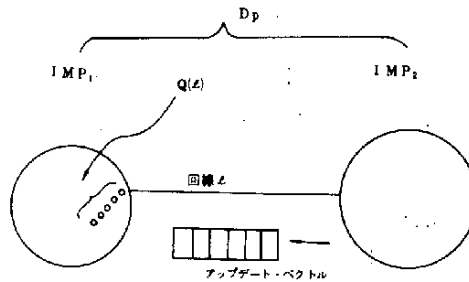


図 3-29 アップデート・ベクトルの通知

今図に於てアップデート・ベクトルの計算後の値を $U(n)$, 又回線を $l = L2$ とすれば, IMP₁ のディレイ・テーブルは図 3-30 のようにして修正される。

ルーティング・テーブルのアップデートの方法にはアップデート・ベクトルを受け取るたびにディレイ・テーブルを評価して作成する方法又はアップデート・ベクトルを受け取ってもすぐに修正を行わず, すべての回線からのアップデート・ベクトルがそろってからディレイ・テーブルとルーティング・テーブルをアップデートする方法等がある。

Periodic Updating法では, このようにして, 自己の内部の事情だけでなくネットワーク全体のトラフィック状態を考慮した決定を下すことが可能になる。又サブネットに障害が発生しても, この情報はルーティング情報としてネットワーク全体に侵透してゆくから, ネットワーク・トポロジーの変化に対しても強い。

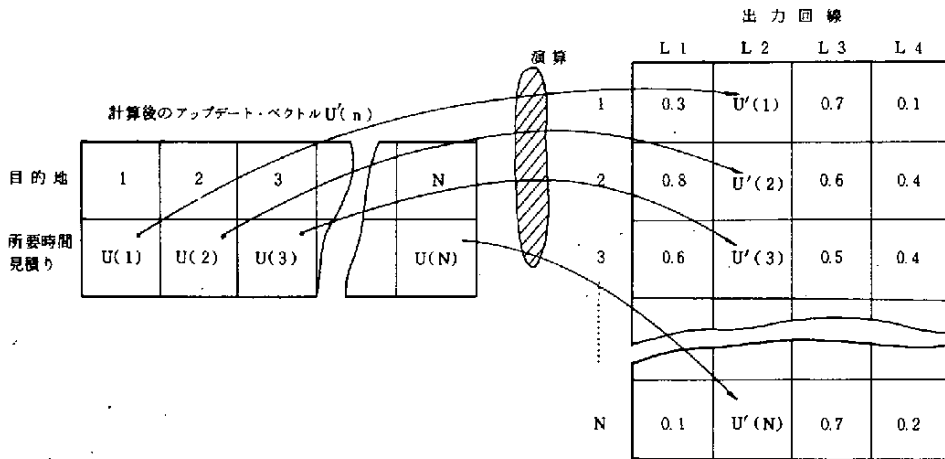


図 3-30 ディレイ・テーブルの修正

しかし、アップデート・ベクトルを周期的に作成して送り出す操作とそのためのトラフィックは余分なネットワーク負荷となるからその間隔の設定には十分な注意を要する。余り短いとオーバーヘッドが大きくなるし又余り長くすると原理的に固定ルーティングと同じことになってしまう。

又外部のトラフィック状態（特に遠方の状態）は時間遅れを伴って伝えられるからトラフィック状態の変化がゆっくりしている場合は良いが急激な変化が頻繁に起るような場合には余り適した方法とは云えない。

Asynchronous Updating 法ではディレイ・テーブルのアップデートを周期的に行わずに、ディレイの見積りが以前の見積りから大幅に変化した時のみ通知する。又パケットのルーティング法は Shorterst queue + Bias 法と全く同じ方法で行う。アップデート・ベクトルを作り出す機構は次のようになっている。

IMP は常に 1 回前に送出したアップデート・ベクトルを記憶している。そして 1 つのパケットをある目的地 n にルーティングする時に、選択された回線のディレイの見積り値 $T(n)$ と以前に作成したアップデート・ベクトルの対応する目的地への見積り値 $U(n)$ を比較する。

($T(n)$ は次式で表わされる。

$$T(n) = \min_{\{L\}} [B(n, L) + Q(L)] \Big|_t$$

もしこの差 $\Delta T(n) = T(n) - U(n)$ があるしきい値を超えたら以前の見積り値はかなり現在の状態からずれが生じていることになるから新たにアップデート・ベクトルを作成して隣接する IMP に知らせる。もし越えていなければアップデート・ベクトルは作成しない。このしきい

値は、スレッシュホールドと呼ばれる。

このように、Asynchronous Updating 法は Synchronous Updating 法に於けるオーバーヘッド・トラフィックを改善したものであるがパケットベースのルーティングを行うことになるためそのための計算負荷が増大することが考えられる。

以上、各種のルーティング・テクニックに関してその手法、問題点について述べたが最後にこれらのパフォーマンス上の性能比較を行っておく。

図 3-31 は Fultz がシミュレーションにより得たパフォーマンスに関する比較データである。

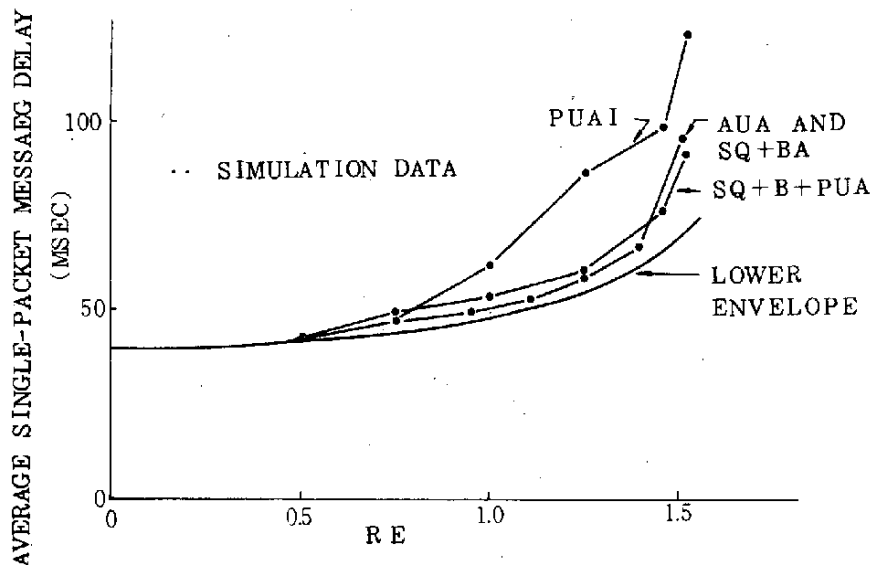


図 3-31 各種ルーティング種法のパフォーマンス比較

横軸 RE はネットワーク・スループットを表わし、縦軸は、シングル・パケットの平均時間遅れを表わしている。

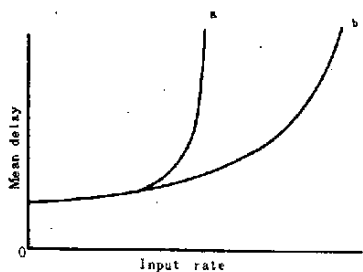
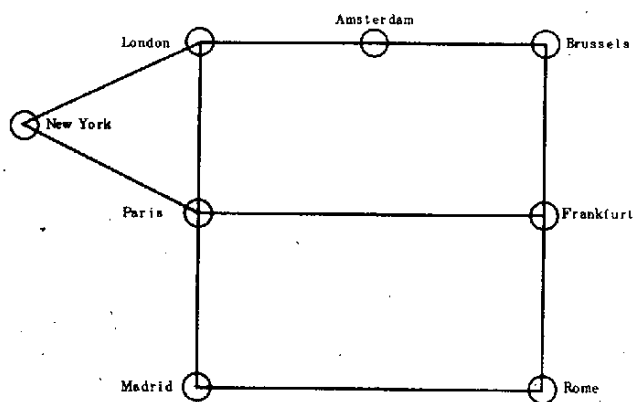
PUA1 は Periodic Updating, AUA は Asynchronous Updating, SQ+BA は Shortest queue + Bias, SQ+B+PUA は Shortest queue + Bias + Periodic Updating を表わしている。

この結果からだと、概して、パフォーマンス上の大きな差は無いと云える。あえて云えば SQ+BA が最も優れていて PUA が最も悪いことになるが、これは、このシミュレーションを行った時の環境下に於て云えることであって、例えばネットワーク・トポロジー、各ノードのトラフィック状態等が変わった時に全く同じ結論が導かれるかどうかは疑問である。

例えば NPL の W・L・PRICE の行った実験では固定ルーティングと適応方式ルーティングではむしろ固定ルーティングの方が優れていると云う結論に達したと云う。

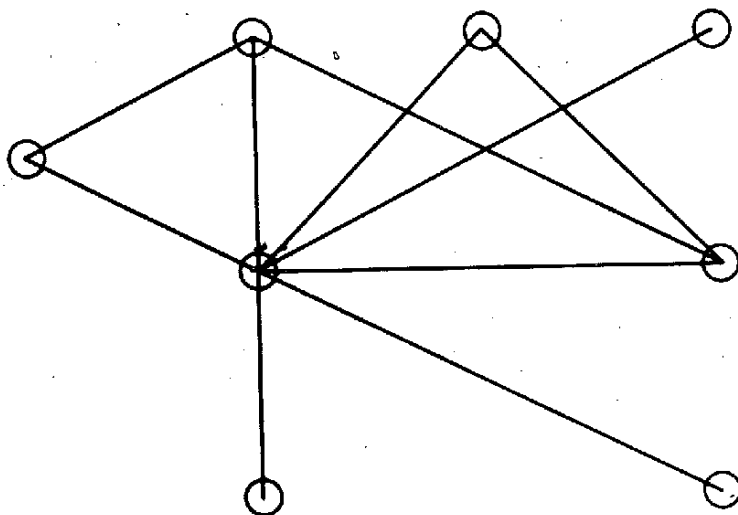
又 SILK が SITA ネットワークについて行った実験ではネットワーク・トポロジーを変えると固定ルーティングと適応方式のパフォーマンスの差が大きく変化することを認めている。図 3-32,

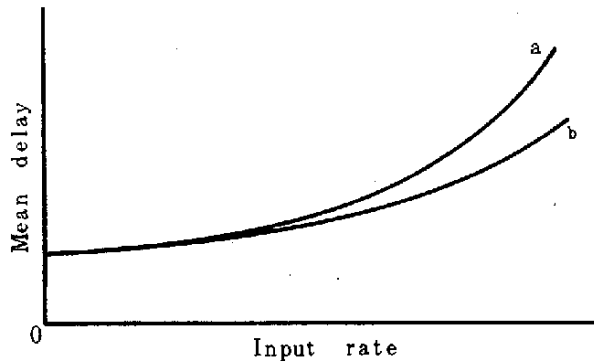
図3-33は2つのトポロジについて比較した結果を示したものである。



a - fixed shortest-distance routing doctrine
b - adaptive routing doctrine

図3-32 SITAネットワーク(I)





a - fixed shortest distance routing doctrine

b - adaptive routing doctrine

図 3-33 SITA ネットワーク(II)

このように考えると、最適な手法の選択は、ネットワーク・トポロジー、運転環境等の要素を考慮した上で為されなくてはならない。

現在我々が採用している方法は Shortest Queue + Bias アルゴリズムを基本とする方法であるが、この手法に関する欠点を補うために次の様な修正機構を取り入れている。

- ① 計算負荷を減すために一部、固定ルーティングのテクニックを取り入れる。

即わち、SQ+BIAS の評価を 1 パケット単位で行わず数パケットは固定ルーティングを行いその後ルーティング・テーブルをアップデートする。

- ② ネットワーク・トポロジーの変化に追従するようにディレイ・テーブルの非同期更新を行う。

即わちネットワーク・トポロジーの変化に関する重要なイベントが発生するとそれを検知した IMP はディレイ・テーブル修正のための情報をサブネット全体に伝える。

イベントの具体的な内容は、回線障害、IMP 障害、回線修復、IMP 回復である。(IMP のスタート・アップ、オフもこれに含まれる) 修正情報をどのようにして送るかと言うことに関しては 3.3.6 の障害対策機能の項で述べることにする。

3.3.4 フロー・コントロール

サブネットは、発信地ホストからメッセージを受け取り、目的地ホストにそれを届ける通信システムとして、ネットワークに接続されているすべてのホストに均等にサービスをしなくてはならない。しかしホストからのメッセージの流入を無制限に許すと、サブネット内のトラフィック密度が高くなってサブネット内に於けるキューイング・ディレイが大きくなるため、メッセージ・ディレイが許容値を超えたり又目的地に於てリアセンブリー・ロックアップと呼ばれる一種のデッドロックが発生し、トラフィックの動きがとれないような状態におちいたりする。このような事態を回避するためには、サブネットはホストからのトラフィック・フローを何等かの形で規制しなくてはならない。このための機能をフロー・コントロールと呼ぶ。フロー・コントロールはサブネット内部

だけの問題ではなく、ホスト-ホスト間に於ても同様な問題としてある。例えば図3-34に於てホストAからホストBに向けて連続的にメッセージを送る場合を考えると、

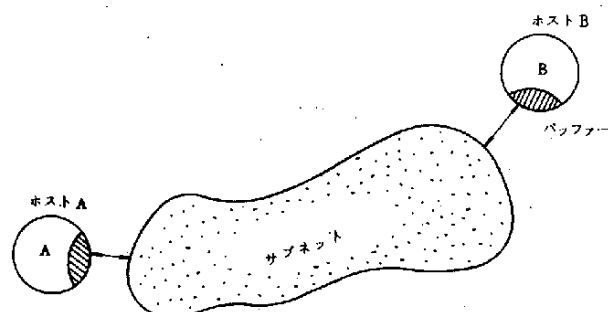


図3-34 ホスト-ホスト・フロー・コントロール

もしホストBがメッセージの流入速度に十分見合うだけのスピードでメッセージを処理しなければ、到着したメッセージは次々とホストBの受信バッファに蓄積されてゆき、最終的に受け取りバッファの余裕がなくなると、受け取ったメッセージを棄却しなくてはならなくなる。この様な場合一般にそのメッセージは再送されることになるがこれは云わば通信システムを無駄使いしているに等しい。従って、この様な事態を無くすためには、ホストA、ホストB間でフローを抑制するための約束を決めておく必要がある。

この様にフロー・コントロールはネットワーク全体として考えなくてはならない性格の問題である。サブネットに於けるフロー・コントロールをサブネット・レベルのフロー・コントロールと呼び、ホスト-ホストに於けるフロー・コントロールをホスト・レベルのフロー・コントロールと呼ぶことにする。ホスト・レベルのフロー・コントロールの規約はサブネットにとっては無関係であり、これはホスト-ホスト・プロトコルとして規定されるので、これについては4章で述べることにする。ここではサブネット・レベルのフロー・コントロールについて述べるがこれは、ホスト-IMPプロトコル、発信地IMP-目的地IMPプロトコル、IMP-IMPプロトコルとして規定される。

フロー・コントロールはデッド・ロック(リアセンブリ・ロックアップ)を防止し、短いメッセージに対する応答時間の制約と大量データの移送に関する高伝送効率とを保証することを前提に考えなくては行けない。現在、我々が採用しているフロー・コントロールの技術はARPAネットワークで実施されている方法とかなり共通点があるが、以下では、初めに、フロー・コントロールの手法を考える上で重要な問題となるリアセンブリ・ロックアップとストア・アンド・フォワード・ロックアップの発生機構について説明してから、従来のいくつかのフロー・コントロールの手法と現在我々の採用している手法について説明を加える。

A リアセンブリ・ロック・アップ

マルチパケット・メッセージが目的地でリアセンブルされる時残りのパケットが到着しないと、

目的地 IMP はそのためのリアセンブリ・スペースを確保したままそのパケットの到着を待つ。このようにしてすべてのリアセンブリ・バッファが、未完成のマルチ・パケット・メッセージで占められてしまうと、この目的地 IMP に向けてやってきた他のパケットは目的地 IMP に入れないため、隣接 IMP にとどまることになる。このため今度は隣接 IMP の蓄積交換バッファがつまり、目的地 IMP への径路を遮断することになる。こうなると、目的地 IMP で、リアセンブリを待っている未完成メッセージの残りのパケットは、その隣接 IMP で拒否されるため目的地に行くことができないので、リアセンブリ・スペースはいくら待っても解放されず、膠着状態が続くことになる。これをリアセンブリ・ロックアップと呼んでいる。

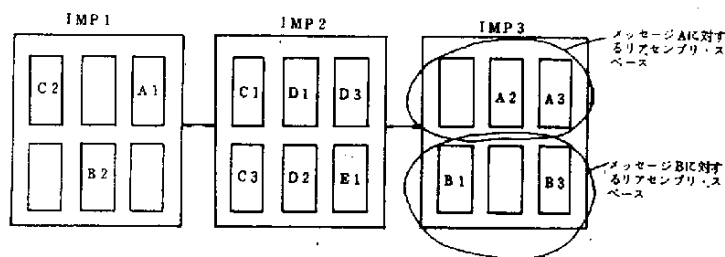


図 3-35 リアセンブリ・ロックアップ

図 3-35 はリアセンブリ・ロックアップの発生する一つの状態を示したものである。

図で、IMP3 は A1, A2, A3 からなるマルチ・パケット・メッセージ A と B1, B2, B3 からなるマルチ・パケット・メッセージ B によってそのすべてのリアセンブリ・バッファを費されている。

一方 IMP2 には、やはり IMP3 を目的地とするパケット C1, D1 …等が到着してきているがこれらは IMP3 にリアセンブリ・スペースがとれないので A, B のリアセンブリが終ってバッファが空くのを待っている。しかし肝心の A1, B2 は IMP2 がつまっているため IMP3 に行くことができない。従ってこの状態はいつまでもたっても解けないことになる。

B ストア・アンド・フォワード・ロックアップ (Store and Forward Lockup)

ストア・アンド・フォワード・ロックアップは 1 本の回線に、大量の両方向トラフィックが集中した時その両端の IMP に於てしばしば発生する。これを説明するために図 3-36 のような構成がよく例にあげられる。図に於て IMP A, B はそれぞれ IMP A', B' に大量のデータを送ろうとしている。この時 C は D に行く大量なトラフィックでそのバッファ・スペースをすべて D に向う送信キューに費やしてしまうことが考えられる。又 D も C に行く大量なトラフィックによってそのバッファ・スペースを C に向う送信キューにとられてしまうことがあり得る。このような状態では C から D に行くパケットは、D では受け入れスペースが無いため棄却せざるを得ない。又 D から C に向うパケットも C に於て同じ扱いを受けるから、ノード C, D ではお互いに受け取ったパケットを棄却し合うばかりでトラフィックの流れは完全に停止し C, D はバッファを満杯

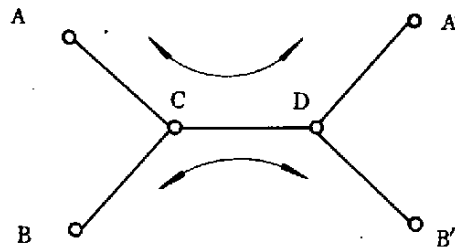


図3-36 Store & Forward Lockup

にしたままロックアップ状態に陥ることになる。これをストア・アンド・フォワード・ロックアップと呼ぶ。ストア・アンド・フォワード・ロックアップの防止は本質的にそれ程難しい問題ではない。これは、IMPのバッファ・スペースを1つの回線の送信キューにすべて費すため受信バッファのスペースが無くなることによって発生する分であるから、この原因をとり除いてやれば簡単に防止することができる。現在、我々の場合は各回線の入出力バッファとして、一定数の専用バッファを割り当て、それ以上必要な時は、バッファ・プールから確保してくると云う形式でバッファ管理を行っているからストア・アンド・フォワード・ロックアップが発生することはない。専用バッファの大きさは、蓄積交換が円滑に行われるのに十分のように確保しなければならない。

C フロー・コントロールの手法

フロー・コントロールの手法は、先に述べたリセンプリ・ロックアップの問題も含めて、スループット、レスポンスを保証する様な機構を考えなくてはならない。初期のARPAネットワークではリンクと云う概念を取り入れてフロー制御を行うことを考えた。リンクとは発信地、目的地に張られたロジカルなデータ・パスでありメッセージはリンクを通して送られると考える。一本のリンクにメッセージが送出されると発信地IMPはこのリンクをブロックし、次のメッセージを受けつけない。メッセージが目的地に到着してRFNM(Ready For Next Message)が返信されると発信地IMPはリンクのブロックをとき次のメッセージを受けつける。この様に1本のリンクを通して複数個のメッセージを同時に送ることはできないから、発信地IMPに於てリンクの最大数を制限すれば、(64本)発信地に於けるフローの流入を制限できると考えたわけである。しかし実際には、この方法には、以下に述べるような多くの問題が含まれていた。

- リアセンプリ・ロックアップの防止に対して効力を持たない。
- 一つの目的地に多くのリンクが集中すると目的地は必然的に混雑状態におちいる。
- 全部のホストから出されるリンクの総数が大きくなるとやはりサブネットは過負荷状態におちいる。

○リンクはコネクションに対応して用いられているから、1つのプロセスが1つのコネクションを使ってメッセージ多重通信を行うことができない。

以上に指摘された問題点を考えると結局リンク・メカニズムは本質的にフロー・コントロールとしての役割を果し得ないと云うことが結論される。

このような理由から、現在の ARPA ネットワークでは完全にこの機構を廃止して、後述する全く別のメカニズムを採用している。

NPLの DAVIES は "Isarithmic" と呼ばれる独想的な機構を提案した。DAVIES はサブネットの混雑は、サブネット内部のトラフィック総量の絶対量を規制しない限り本質的に避けられないと考え、サブネット内部のトラフィックを一定に保つようなフロー・コントロールの機構を考えた。

これは次の様にして実現される。(以下では1メッセージ1パケットとして話を進める)

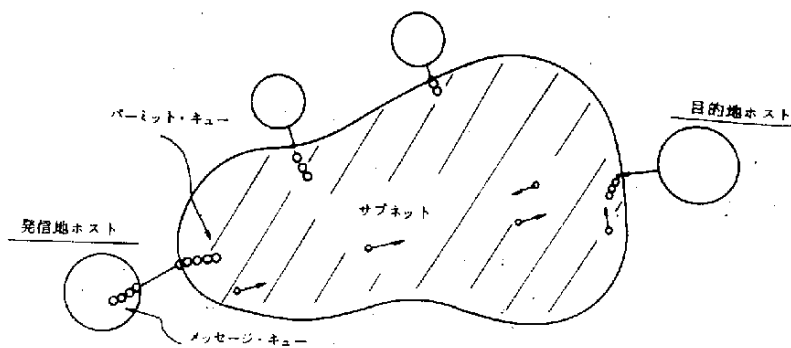


図 3-37 Isarithmic の概念図

パケットがサブネットを通して運ばれる時、パケットは必ずパーミット (permit) と呼ばれる乗り物に乗せられて運ばれることにする。従ってメッセージが発信地からサブネットに入る時にはまずパーミットを確保しなければならない。これは発信地 IMP のパーミット・キューを調べて、取り出されるがもしパーミットがキューにあればこれを1つ確保しそのパーミット・キューの長さは1つ減らされる。パーミット・キューが0の場合はメッセージはサブネットに入ることができない。パーミットを確保したパケットは目的地に向かって出発するが目的地に到着するとパーミットを返却し目的地のパーミット・キューの長さを1つふやしてサブネットの外に出る。この時目的地のパーミット・キューの長さがある一定値を超えると、過剰なパーミットはそれを要求している他の発信地に送り、すべてのノードに於ける流入トラフィックのバランスを保つ。この機構ではサブネット内のパーミットの総数を一定にして於けばサブネット内のトラフィックがある一定の許容値を超えないようにすることが簡単に実現できる。又局部的に集中してパケットが流れても、送信側はパーミットがついてくるからそれ以上送り出すことができず混雑状態は持続しないことになる。

DAVIES はこのようなフロー・コントロールの手法を "Isarithmic" と名付けた。(ギリシャ語で equal number の意味) しかし、この方法も完全な形で実現されるためには、次の様な問題が検討されなくてはならない。

○メッセージがマルチパケットで構成される時にも、等しく有効な手段として適用されるかどうか。

○過剰パーミットの配分アルゴリズムをどうするか。

現在、我々が採用しているフロー・コントロールの手法は ARPA の技術に基くものであるが、これは基本的に、次の様なねらいを持っている。サブネットのフロー・コントロールは発信地 IMP、目的地 IMP の間に規約を設けることにより実現する。即ち送信要求のある発信地 IMP は各々目的地を混雑させないように、原則として目的地の混雑度を調べた上でホストからメッセージを受け入れるようにする。このようにすればネットワークを混雑させる、フローの流入を止めることができるし又、目的地に於てリアセンブリ・ロックアップが発生することも無い。厳密には、マルチ・パケット・メッセージとシングル・パケット・メッセージは別の形でフローの抑制を受ける。以下ではこの機構についてもう少し詳細な説明を加える。

① マルチ・パケット・メッセージのフロー・コントロール

発信地 IMP がホストからマルチ・パケット・メッセージを受けとる時、初めにそのリーダを受けとって、残りのセグメントに関しては、目的地の様子が分るまでその受信を見合わせる。そしてリーダを受けとるとまずその目的地を知り、その目的地に対して、マルチ・パケット・メッセージのリアセンブリ・バッファを確保するためのリザーブ要求を出す。これは RFA (Request For Allocation) と呼ばれる特別なコントロール・パケットを送信することによって行われる。一方 RFA を受け取った目的地 IMP は現在のバッファの空き具合をから、リアセンブリ用の 8 個のバッファが確保できるかどうかを調べる。もしバッファが確保できれば、そのバッファを発信地に対して割り当て、他の使用を禁止して、発信地に対して要求を受け入れたと云う返答を返す。これは ALLOC (Ready For Allocation) と呼ばれる特別なコントロール・パケットの返信によって行われる。発信地 IMP は ALLOC を受けとると、それまで見合わせていた残りのセグメントをすべて読み込み目的地に送り出す。目的地 IMP でバッファが確保できなかった場合は RFA はリクエスト・キューと呼ばれる、キューに入れられてバッファが空くのを待ち、バッファが空くと直ちに ALLOC が返信される。

一般に、その目的地に対していくつかのホストが発信を行っていると考えられるから、RFA はリクエスト・キューを形成し、順番にサービスを待っていると考える方が自然である。

この機構では、発信地 IMP はマルチ・パケット・メッセージを受け取るたびに目的地 IMP に問い合わせをしなければならない。これはオーバ・ヘッド・トラフィックとなるから連続的にマルチ・パケット・メッセージを送るような場合には余り効率的ではない。この点を

補うために次の様な約束を設ける。目的地 IMP は目的地ホストからマルチ・パケット・メッセージに対する RECEIVE を受け取ると、その返信先 IMP に対する RFA を自動的に創成し RECEIVE メッセージと共にリクエスト・キューにキューイングする。目的地 IMP はリクエスト・キューを処理する時 RECEIVE が付いた RFA に対しては ALLOC を RECEIVE 情報に乗便させて発信地に返すようにする。一方、発信地 IMP では、ALLOC の受け取りに関するインディケータを設け、ALLOC が返されるとこのインディケータを設定するようにする。この様にすれば RFA を出すのは最初のメッセージだけで引き続いて、発信地 IMP に入ってきたマルチ・パケット・メッセージに関しては RFA を出すことなく、このインディケータの設定を調べてもしセッット状態にあれば残りのパケットを読み込み目的地に伝送することができる。又、マルチ・パケット・メッセージの流入が途絶えると発信地 IMP はタイム・アウトをかけてこのインディケータをリセットし目的地ホストに対してバッファが不要になったことを知らせる GIVE BACK メッセージを送る。ALLOC インディケータは後述する。パイプ単位で管理されることになる。図 3-38 はこの過程を図示したものである。

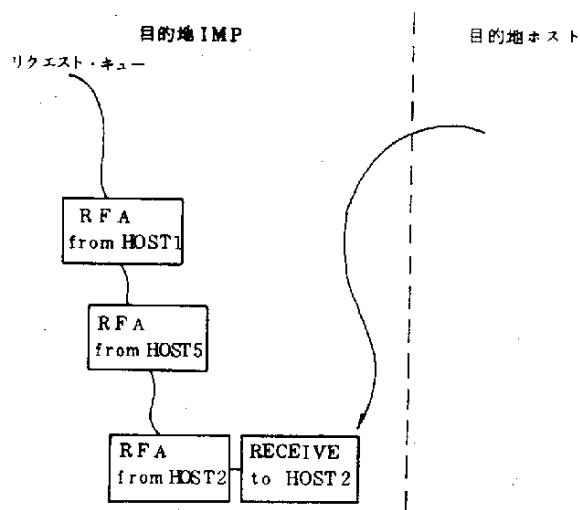


図 3-38 リクエスト・キューと RECEIVE

② シングル・パケット・メッセージのフロー・コントロール

シングル・パケット・メッセージの場合は、目的地 IMP はそのまま無条件に読み込み、目的地に向けて送り出す（実際には後述するパイプ機構から、発信の許可を得る）。

この時発信地 IMP にはそのコピーが保存される。このパケットが目的地ホストに、受けとられ RECEIVE が返されると発信地 IMP のコピーが棄却されるようにする。シングル・パケットが目的地 IMP に到着した時に、これを受け入れるためのバッファが無い時は、目的地 IMP はこのパケットを棄却し、その発信地から送信要求のあったことを記憶しておく。これ

は別のリクエスト・キューにALLOC.S(Ready For Allocation For Single Packet)をキューイングすることにより行われる。

目的地IMPのバッファに空きができると、このキューを見て次々に送信要求のあった発信地IMPにALLOC.Sを返す。発信地IMPはALLOC.Sを受けると残してあったパケットのコピーを再び目的地に送る。

目的地IMPは2つのリクエスト・キューにサービスするが、この時ALLOC.Sキューの方を優先してサービスするようにする。

3.3.5 シーケンス・コントロール

発信地IMPは、ホスト・チャネルを通して次々と流入してくるメッセージ・ストリームをそのホスト入力用バッファの最大限(基底となる個数とダイナミックに拡張可能な個数の和からなるから、その時のIMPの状態で異なる)まで受け入れることができる。(マルチ・パケット・メッセージの場合は、メッセージは受け入れるが、その実体の読み込みはフロー・コントロールの規制を受ける)IMPはこれらのメッセージをパケット化しフロー・コントロールの規制の下に次々と目的地に向けて送り出してゆく。各目的地ではこれらのメッセージを受けとってホストに渡さなければならないが、各々のメッセージがサブネットを通過するために要する時間は、その時のサブネットの状態によって決まるので、必ずしも発信された順番で目的地に到着すると云う保証はない。このため目的地では、メッセージ・シーケンスを整えてからホストに渡す手順が必要になる。この機能をシーケンス・コントロールと呼ぶ。

初期のARPAネットワークで採用されていたリンク機構では、リンクをコネクションに対応させて用いていたので、1つのコネクションは1メッセージ毎にRFNMで同期をとって送ることになり、コネクション単位にメッセージ・シーケンスを保証することができた。

現在、我々が採用しているフロー・コントロール機構には、このような機構が含まれていないためこれは別の機構として実現しなければならない。

現在のARPAネットワークでは、パイプ機構と呼ばれる手法を用いてシーケンス・コントロールを行っている。パイプ機構では、1つの発信地IMPと1つの目的地IMP間に、パイプと呼ばれる1本の論理的なデータ・パスを仮定し、パイプに同時に流れるメッセージ数を制限することによって、目的地IMPがシーケンシングを行う上での効率化をはかっている。仮りにこのような機構を設けないと、ある発信地IMPからある目的地IMPにフロー・コントロールの許す範囲で無制限にメッセージが流れることになり次のような問題が発生する。今①から⑩までの順序付けられたシングル・パケット・メッセージが同時にある目的地に送られたが運悪く①の到着が最も遅れたとすると、②から⑩までの9個のメッセージは到着しているにもかかわらず①のメッセージがくるまでは目的地に送り出すことができない。(図3-39参照)

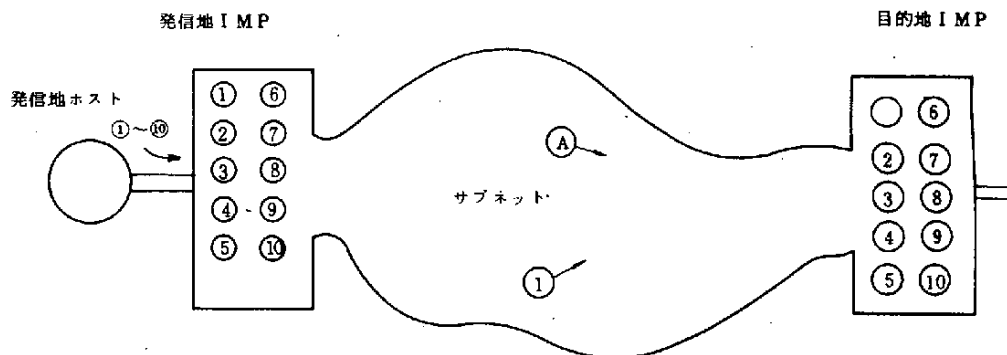


図 3-39 パイプ機構を持たないサブネット

この過程には次の様な問題が含まれている。

- 目的地に於て②から⑩までのパケットがバッファを占有する。
- 発信地に於ても RECEIVE が返されないためバッファが占有されたままになる。
- シーケンシング待ちのパケットに対してバッファを割り当てるアルゴリズムをうまく考えないと、メッセージ・シーケンシングのためのロックアップ現象が発生する可能性がある。

(例えば、図の空いている場所に A が入って A がやはりシーケンシング待ちに入った場合)

この様にパイプ機構の設定はシーケンシングに重要な意味を持つ。又パイプの許容メッセージ数を大きくするとネットワーク効率を低下させる可能性がある。しかしこれを余り減すと今度はパイプあたりの最大スループットが小さくなってしまう。現在の ARPA ではこれを 4 つに設定している。

又、ARPA ではノーマル・メッセージを運ぶパイプとプライオリティー・メッセージを運ぶパイプを区別してプライオリティー・メッセージの扱いに優先権を与えることによりプライオリティー・メッセージとノーマル・メッセージの干渉を無くしている。しかしノーマル・メッセージ用のパイプに於けるシングルパケット・メッセージとマルチパケット・メッセージの干渉の問題については考慮していない。

例えば図 3-40 に示すように NCP が短いメッセージを発生する会話型プロセスと長いメッセージを発生するファイルを扱うプロセスに同時にサービスしている時、もし両者が同一の目的地を利用すると、一本のパイプには図の様に長いメッセージと短いメッセージが混入することになる。このため、本来は独立にシーケンシングされ得る性格のものであるにもかかわらず、短いメッセージは長いメッセージの伝送遅れの影響をそのまま受けるような事態が発生する。

我々はこの問題を解決するためにノーマル・メッセージ・パイプをさらに 2 つに分けて扱うことにした。以下ではこの機構について解説する。

パイプ：発信地 IMP と目的地 IMP の間に対応する 1 本のロジカルなデータ・パスを考え、これをパイプと呼ぶ。従ってすべての IMP は各目的地に対して 1 本ずつ、全部で (ノードの数 - 1) 本の

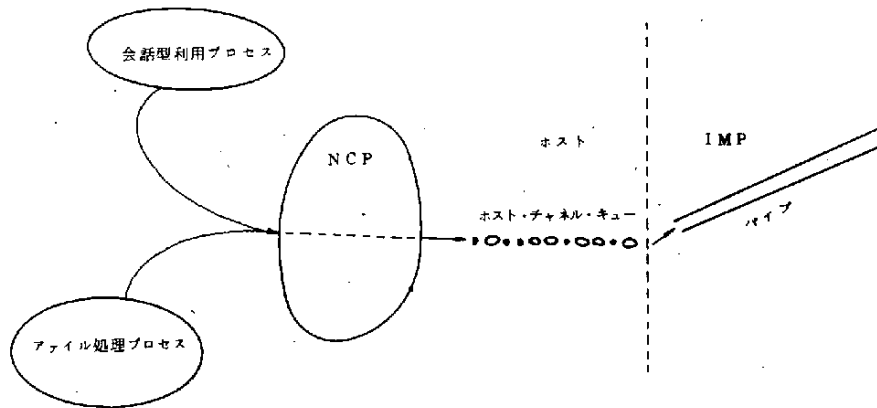


図3-40

パイプを備えていることになる。1本のパイプは、プライオリティー・パイプ、ショート・メッセージ・パイプ、ロング・メッセージ・パイプの3本のパイプから構成される。プライオリティー・パイプには最も高速性の要求されるプライオリティー・メッセージが送られる。ショート・メッセージ・パイプとロング・メッセージ・パイプにはそれぞれ1パケット・メッセージ、マルチ・パケット・メッセージが送られる。(図3-41参照)

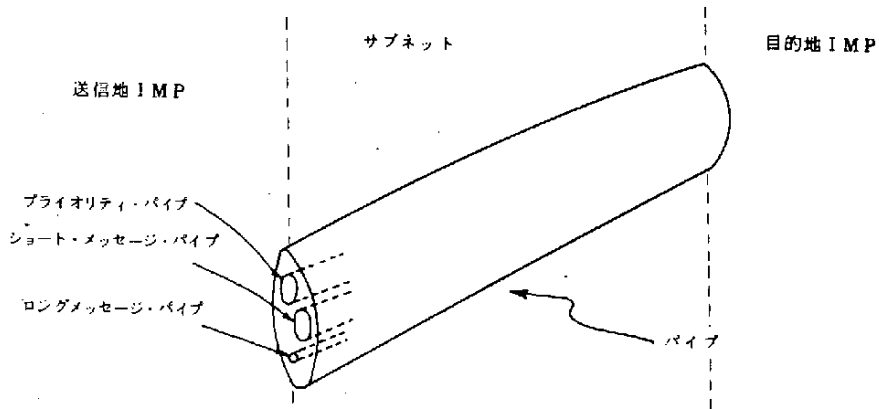


図3-41 パイプの概念図

同時にパイプに流し得るメッセージの最大値は、プライオリティー・パイプ、ショート・メッセージ・パイプは各々4つ、ロング・メッセージ・パイプは1つである。

発信地IMPは各パイプに流入するメッセージ個数が許容値を超えないように監視していて、これを超えるような場合はパイプに入れない。一つのメッセージがパイプの使用を終えるのは、そのメッセージに対するRECEIVEが発信地に返送された時である。マルチ・パケット・メッセージに関しては、メッセージはパイプに入っているが目的地にバッファが確保されないため送出できない場合も十分あり得る。

ナンバリング：各メッセージは、パイプに入れられる時に、目的地でこの順番が分るように一連の

メッセージ番号が付けられる。メッセージ番号はこの他に重複メッセージの検出のためにも使われる。メッセージ番号にはパケット・ヘッダーの12ビットを割り当て、1つのパイプの中にある3種類のパイプに関しては各々独立にナンバリングを行う。割り当てられたメッセージ番号フィールドを使い果した時には再び0から出発する。

シーケンシング：目的地 IMP は各パイプごとに、メッセージ番号を管理している。即ちあるパイプに関して、最後にホストに送り出したメッセージのメッセージ番号を記憶している。これをそのパイプの有効番号と呼ぶ。到着してきたメッセージが、この番号のすぐ次の番号の時は無条件にホストに送り出す。そうでない時は、このメッセージをシーケンシング待ち状態におく。

例えばあるパイプの有効番号が5の時に、メッセージ番号6のメッセージが到着したとするとこのメッセージはそのままホストへ送られ有効番号は6に変わる。もしメッセージ番号7のものが先に到着した場合は、このメッセージはシーケンシング待ち状態におかれ、メッセージ6の到着を待つことになる。やがてメッセージ6が到着すると6、7はこの順でホストに送られ有効番号は7に変わる。

又、有効番号が m の時、 $m < n < m + 4$ を満たさ無いような番号 n のメッセージが到着した時は、これを重複メッセージとみなし棄却する。

ロング・メッセージ・パイプに関しては、一度に1つのメッセージしか送られないからシーケンシングの必要は無い、しかし有効メッセージ番号の管理は、重複メッセージの防止と云う立場から必要である。

3.3.6 サブネットの障害対策機能

サブネットが、データ通信システムとしての信頼性を保つためには、サブネット内に発生した障害を検知しそれに対して適切な処置をとるための障害対策機能が考慮されなければならない。

サブネットの障害には、その構成要素（IMP、回線）の完全なダウンばかりでなくノイズ、電源異状等による一時的な障害、又原因不明のソフトウェア異状等も含まれる。これらのサブネット障害を扱う上での基本的な態度として次の様な点に注意を払う必要がある。

- サブネット内の一部の構成要素が障害によって失われた時にもそれを取り除いた環境の下で、可能な限りのリソースを使って正常な運転が続行されなくてはならない。
- 障害の発生及びその回復手続きの過程に於てサブネット全体の運転が一時的にも中断されるようなことがあってはならない。
- 障害回復はできる限り自動的に行われることが望ましい。

以下では、サブネットの障害対策に関する機能について、現在我々が採用している方法を説明するが、その多くはARPAネットワークで実現された方法を基本としたものである。

A 伝送エラー

サブネット内部を流れる情報は、単にホスト-ホスト間でやりとりされるメッセージ情報ばかりでなく、サブネット内の各種プロトコルに基づくコントロール情報(RFA, RECEIVE, etc)も含まれる。従ってサブネット内に於けるパケットの紛失は、単にホスト-ホスト間でやりとりされるメッセージの紛失と云う問題にとどまらず、それによってサブネット・プロトコル体系の動作に異状をもたらし、サブネットの運転に支障をきたす可能性すらある。基本的には、パケット交換は(2)で述べた伝送確認、再送手続きを通じてその伝送の信頼性に関する配慮が為されている。しかし、データ伝送の信頼性は高めることはできても完全に伝送エラーを取り除くことは不可能であるから、むしろここを出発点として考えなくてはいけない。即わち、これまでに述べた各種サブネット・プロトコルの設計の中には、この様な障害の発生を前提としたアルゴリズムが含まれていなくてはならない。ここでは、特に情報のミス又は重複があった時に各プロトコルがどのように応待するかと云う点に焦点をしばり改めてその動作を明確にしておく。

(1) ノーマル・パケットの紛失：シングル・パケット・メッセージが失われた場合は発信地

IMPはいつまでもたってもそのRECEIVEを受けとらないことになる。発信地IMPは送り出したメッセージごとに関信してから経過した時間を管理していて、もしこれがある一定値(30秒)を超えてもRECEIVEを受けとらない時は目的地に対して問い合わせコントロール・パケットINQ(inquiry)を送る。これを受け取った目的地IMPは、そのシーケンス・コントロール機構を通じて、問い合わせのメッセージを以前に受け取ったかどうかを調べ、答をコントロール・パケットANS(ANSWER)に乗せて発信地IMPに返す。発信地IMPはこれを見て、もし受け取られていれば、パケットのコピーを棄却しRECEIVEを創成して発信地ホストに通知する。又もし受けとられていなければ、パケットのコピーを再送する。この場合に、INQ又はANSが紛失したらどうなるであろうか？ この場合を考慮して、上で述べた操作はINQの発信後、そのパケットを再びもとのRECEIVE待ち状態におくことにより、解答が得られるまで繰り返されるようになっている。

マルチ・パケット・メッセージの場合は、目的地IMPに於て管理される。目的地IMPはマルチ・パケット・メッセージに対してバッファを割り当ててから一定の時間(30秒)を経過してもリアセンブリが終らない時は、再送を要求するコントロール・パケットを発信地に対して送る。(そのパイプの1つ前のメッセージのRECEIVE情報をそのまま送る。これにはALLOCも含まれている) これを受け取った発信地IMPはホストに対してそのメッセージの再送要求を出し再び同じメッセージを目的地に送る。この手順も、成功するまで繰り返される。重複パケットはリアンブリ過程又はシーケンス・コントロールによって検出され、不要の場合は棄却される。

(2) RECEIVEの紛失：(1)で述べた話で尽きている。

(3) RFAの紛失：発信地 IMPではRFAを出してから一定時間経過しても、ALLOCを受け取らない時はRFAが失われたものとして再びRFAを発信する。目的地 IMPでは、現在バッファがアロケートされているか又は前に受けたRFAがリクエスト・キューにのってあればこれを棄却してしまう。もしどちらにもあてはまらない時はリクエスト・キューに新たにつなぐ。

(4) ALLOCの紛失：(3)で述べた話で尽きている。

(5) ALLOC・Sの紛失：(1)で述べた話で尽きている。

(6) GIVE BACKの紛失：(1)で述べた話で尽きている。

B 回線障害の検出

伝送確認手続きの項で述べたように、ある回線に関して送出すべきトラフィックが完全に途絶えるとIMPはマルパケットを送出する。マルパケットも流れない状態になると受信側IMPでは0.5秒間々隔ごとにhelloパケットと呼ばれる特別のパケットを相手側IMPに送る。これを受け取った相手側IMPは確認信号としてI heard youパケットを返信する。このマッチングが約2.5秒間連続して行われないとその回線は故障したとみなされIMPは次の手続きに移る。

- 自己のルーティング情報（バイアス・テーブル）を修正し、周辺IMPにこれを伝える。

- それまでにその回線の出力キューに並んでいたパケットを再びルーティングし直す。

- ディスコネクションの吟味を行う。

又回復手続きを始動し、引き続きhelloパケットを送り出す。helloパケットを受け取ったIMPではやはりこれに対する確認I heard youパケットを返信する。このマッチングが30回行われると回線は回復したとみなされ回復手続きを始める。これはルーティング情報の修正を通じて行われることになる。

C IMP障害の検出

周辺IMPはこれを回線障害として検知し扱う。

D ホスト障害の検出

目的地IMPは、目的地ホストがある一定時間経過してもメッセージを受け取らない時はこれをホスト障害とみなしサブネット全体にこれを通知する。ホスト障害情報を受け取った各IMPはそれをルーティング・テーブルに記録しもし障害ホストに向けられてきたパケットを受け取ったらそれを棄却する。

E ディス・コネクションの検出

あらゆるノードと交信不能なノード又は交信不能なサブネット同志のことをディス・コネクションの状態にあると云う。例えば図3-42ではIMP3はサブネットから完全に孤立している。

又図3-43ではサブネットが2つのサブネットに分断され、お互いに交信不能の状態にある。サブネット内に発生した、ディス・コネクションを放置すると孤立した目的地IMPに向けられ

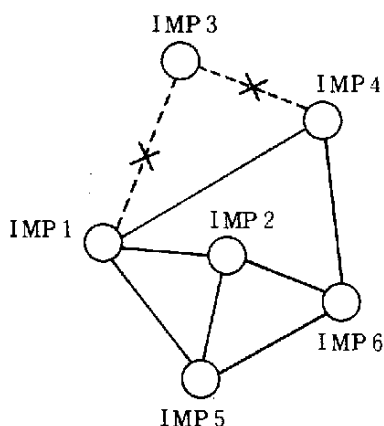


図3-42 ディスコネクション状態になったノード

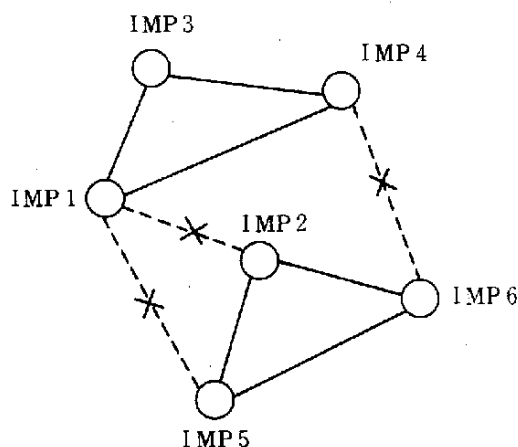


図3-43 サブネット同志のディスコネクション

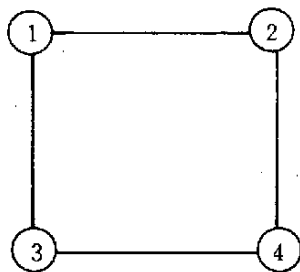
たパケットは行き先が無いため永久にサブネット内部を廻り続ける。そしてサブネットはそのようなパケットで充満することになるだろう。この様な事態を無くするためにはディスコネクションをできるだけ早期に発見しサブネット全体に伝えてやらなければならない。ARPA ネットワークでは、これを次のような方法で検出している。まず、ルーティング情報の周期更新を行う時に、各 IMP は自分を通して目的地に行くための最小ホップ数をルーティング情報にのせて周辺 IMP に知らせるようにする。

周辺 IMP から目的地に行くためのホップ情報を得た IMP はそれにさらに 1 を加えてその IMP から見たホップ数に変換する。このようにしてあらゆる IMP のホップ情報は常に最新の情報が保たれている。従ってサブネット内にディスコネクションが発生していなければ各 IMP の保有するホップに関する情報は常に一定値を保っているはずである。もしディスコネクションが発生するとその目的地に関するホップ数は際限なく大きくなってゆくからこれが（全ノード数 - 1）を超えたら、そのノードはディスコネクションの状態にあると判断する。

我々の場合はルーティング情報の周期的な更新を行っていないからこのような方法は通用しない。又上に述べた方法では更新周期が約 625 ms であるからディスコネクション情報がサブネット全体に行きわたるにはかなり時間がかかることが予想される。

現在、我々は次の様な機構でディスコネクションの検出を行っている。

各 IMP は、サブネットのトポロジーに関する情報を蓄えたルート・マトリックスを備えている。ルート・マトリックスは行、列が各々目的地をあらわしその要素 r_{ij} は、もし目的地 i と目的地 j の間に回線が張られていれば 1 そうでなければ 0 と置くことによりネットワークの構成を表現している。例えば図 3-44 の例ではルート・マトリックス R は次のようになる。



$$R = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \end{matrix}$$

図 3-44 ルート・マトリックス

グラフ理論によれば、 R^n の各要素 $rij^{(n)}$ は i から j に n ホップでいく道が何通りあるかを表わしている。今 R^n の各要素についてそれが 0 でなければこれをことごとく 1 に置き換えて得られたマトリックスを $A^{(n)}$ とすれば、 $A^{(n)}$ の各要素 $A^{(n)}_{ij}$ は i から j に n ホップでゆく道があるかどうかを表わしていることになる。今サブネットのノード数を N とすれば、ディスコネクションは次のようにして発見できる。はじめに $A^{(1)}, A^{(2)}, A^{(3)}, \dots, A^{(N)}$ を求めて

$$D = A^{(1)} \cdot \text{OR} \cdot A^{(2)} \cdot \text{OR} \cdot A^{(3)} \cdot \dots \cdot \text{OR} \cdot A^{(N)} \quad (\cdot \text{OR} \cdot \text{ は各対応する要素の OR をとることを意味する })$$

を求めれば D の各要素 D_{ij} は i から j に N ホップ以内で行くルートがあるかないかを示していることになる。従って D を計算した IMP では自分に対応する D の行 (又は列) を見て、0 である要素が含まれていれば、その対応する列 (又は行) の目的地がディスコネクション状態にあることを知ることができる。

D は回線障害が発生する度に次のような手順で各 IMP で計算される。又この時、同時にルーティング・テーブル (バイアス・テーブル) の修正も行う。

今図 3-45 で示されるネットワークで IMP3 と IMP8 の間の回線が故障したとすると、IMP 3, IMP8 は自己のルート・マトリックス及びルーティング・テーブルに修正を加え、さきに

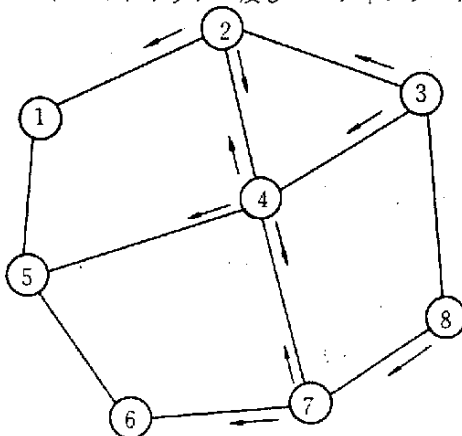


図 3-45 回線エラー情報の伝え方

述べた方法でディスコネクションのチェックを行ってから回線障害情報をアップデート・ベクトルと共に周辺 IMP に伝える。又アップデート・ベクトルを受け取った周辺 IMP はその情報が、ルーティング・テーブル又はルート・マトリックスに修正を加えるものであれば同様な手続きを踏んでそれを再び周辺に伝える。もしそれが以前に受け取った情報と全く同じものであれば棄却する。このようにしてサブネットに完全に情報が行きわたれば、障害情報、アップデート・ベクトルは完全にサブネットから姿を消し、ネットワークはその環境の下で最適な運転状態に入る。又、この過程でディスコネクションが発見されるとそれを発見した IMP は自己のルート・マトリックスに修正を加え、ルーティング・テーブル・アップデートの収集を早めるようにする。ルート・マトリックス D は、行に IMP の 1 語を対応させ全部で 16 w で構成されている。一見マトリックス演算に計算時間がかかりそうであるが、実際のオペレーションはすべて論理演算命令で行われること又ルート・マトリックスが対称行列であること等からかなり計算時間を短縮することが可能である。(1 回のチェックに要する時間は約 10 ms 程度)

F IMP の自己保存機能

電力異状に陥った時のプログラムの暴走を未然に防ぐ目的から、IMP は電力異状割り込みが発生すると保護メモリー領域におかれた自動停止プログラムに制御をわたし、プログラムが暴走を始める前に、IMP を正常な停止状態におく。又電力が回復すると、やはり保護メモリー内にある回復プログラムを起動し、IMP プログラムの初期設定手続きを行って再スタートさせる。この時障害の発生する直前に扱った IMP 内のバケットはすべて棄却される。又ソフトウェアの異状を監視する目的から "Watch Dog Timer" と呼ぶ監視機構を持っている。プログラムはその正常な動作の過程に於ては必ずある時間内にこのタイマーを設定するような機構をプログラム構造上実現する。このようにすればプログラム異状時(例えば特定の部分でループを発生したような場合)にはこのタイマーの設定が行われないためタイム・アウトが検出され、プログラム異状が発見できる。プログラム異状を検出した IMP は保護メモリー領域におかれたオート・ローディング・プログラムに制御を渡す。オート・ローディング・プログラムはそのどれかの回線を使って隣接 IMP からプログラムのコピーを送ってもらう。又この過程で IMP は保護メモリー内におかれたその IMP 個有の情報をプログラムに設定する。このようにしてプログラム・ロードが成功すると再び初期設定手続きを行ってプログラムを始動する。

3.3.7 デバッグ、モニタリング機能

IMP プログラムはその通信システムとしての本来の動作を司るオペレーショナル・プログラムとオペレーショナル・プログラムの動作が空いた時に、モニタリング等の副次的な仕事をするバック・グラウンド・プログラムから構成される。

バック・グラウンド・プログラムには、モニタリング機能の他、IMP テレタイプによる IMP -

IMP間の交信機能、スナップ・ショット等のデバックに関する機能、擬似ホストとしての機能、人工負荷の発生機能等を持たせ、ネットワークのパフォーマンスを測定したり、デバッグの便宜を計ることを可能にしている。

以下ではバック・グラウンド・プログラムの主な機能について解説する。

A TTY

IMPテレタイプから入力されたテキストをメッセージに変換し、指定されたホスト又はIMPに向けて送り出す。又そのIMPに向けられてきたIMPメッセージを受け取り適当な変換を行ってテレタイプにプリントする。又他のバック・グラウンド・プログラムに向けられたものは対応するプログラムに渡す。

テレタイプからはリーダ情報及びメッセージ・テキストを入力するがこれは文字入力、バイナリ入力のいずれも可能である。

B DISCARD

そのIMPに向けられたIMPメッセージがDISCARDを指定していた場合は、DISCARDルーティンはこのメッセージを消滅し発信地に対してRECEIVEを送る。擬似トラフィックを発生させてパフォーマンス測定を行う時などにメッセージの後始末をするために使う。

C TRACE

ホストメッセージもIMPメッセージもそのリーダにトレース・ビットと呼ばれるフラグを持っている。任意のIMPは受け取ったパケットのトレース・ビットが設定されていた時にはそのパケットが出てゆくまでにそれをどの様に扱ったかを記録しその情報をメッセージにまとめて特定のホストに送る。この機能はサブネット内をメッセージが通過する過程で各IMPでどのような処理を受けたかのヒストリーを知る上で大変便利である。

TTYで収集する情報は次の通りである。

- 通過したパケットのヘッダー
- 到着した時の時間
- 出力回線の番号
- 回線キューに置かれた時の時間
- 送出した時の時間
- ACKを受け取った時の時間
- 再送表示

D SNAPSHOT

IMPの保有するパラメータ・ブロック（バックグラウンド・プログラムの動作を指示するパラメータの集り）に於てSNAPSHOTパラメータが設定されると、IMPは指示された時間々隔ごとに、IMPの現在の状態を表わす（キューリングス、ルーティング・テーブル等）情報を集め

て特定のホストに向けて送り出す。SNAPSHOT が各 IMP に於てとられる時、そのタイミングは同期をとってやる必要がある。時間々隔の指定はパラメータで指示されるが、最低は 0.5 SEC で約 25 ms きざみで可変である。

SNAPSHOTによって収集される主な情報は以下のとおりである。

〔キュー・レングスに関して〕

- 各ホストへの出力キュー
- 各回線への出力キュー
- 各種コントロール情報のキュー
- フリーストレージの数
- 現在使われている蓄積交換バッファの個数
- 現在使われているリアセンブリ・バッファの個数

〔ルーティング・テーブルに関して〕

- 各目的地に行くための出力回線
- 各目的地に行くためのディレイ見積り

その他サブネット障害に関する情報（ホスト障害、IMP障害、回線障害）も記録している。

E 統計収集

パラメータ・ブロックに於て統計収集を指示するパラメータが設定されると IMP は、受け取ったメッセージの個数や種類、行き先、IMPの状態等に関する統計をとり、これを一定の時間々隔ごとに特定のホストへ送る。

これらの情報は、ネットワークの稼動状態を把握したり、ネットワーク・ロードの分析を行ったりする目的で使用されることになる。

収集される主な情報は次のとおりである。

- メッセージ長に関する統計（パケット数の分布でとらえる）
- 1パケット・メッセージの長さに関する統計
- ホストから流入したメッセージの総数
- ホストに向けて出したコントロール・メッセージの総数
- 各目的地に向けられたメッセージの往復時間の総和
- 各目的地ごとのメッセージの総数
- 各回線に出された hello パケットの個数
- RECEIVEを出した回数
- 各回線を通して送られたトラフィック量
- リアセン・ブリ時間に関する統計
- CPUの利用率

○各キューに関する統計

F 人工負荷の発生

人工負荷の発生を指示するパラメータが指定されると、IMPは指定された長さを持つメッセージを指定された時間々隔をおいて発生する。その目的地はパラメータ・ブロックで定義される。発生したトラフィックは前に述べたDISCARDにより目的地で棄却される。

時間々隔を指定する最小ユニットは約 25 ms である。

G パラメータ・チェインジ

パラメータ・チェインジに向けられたメッセージを受け取りその内容に従ってパラメータ・ブロックの各パラメータを設定する。

3.4 IMP のプログラム構造

3.4.1 概 要

IMP中のプログラムは、前節で述べたような各種の機能を効果的にはたすような構造を持たなければならない。IMPにはメッセージ交換を高い信頼性でもって効率よく実行するMessage Control Program(MCP) が主役として存在するがここで注意しなければならないのは、TIP の機能とのソフトウェア上のインタフェースを明確にしておかなければならないことである。

MCP自身は、対象となる装置が明確になっているので規格化をおこない易く、すべてのIMPにおいて同一プログラムで目的を達することができる。

しかしながらTIPは、標準端末に対するサポートばかりでなくたとえばTIPをremote data station としてあるいはリフレッシュ形のCRTなどを接続して使用したいなどの要求がユーザからでてくることが予想される。

このような各種の要求をすべて同一プログラムでサービスすることは困難である。このために、TIPのプログラムに関しては、それを設置したセンターの要求に合わせてプログラムを追加してゆく方針をとることとし、当初は標準的かつ基本的なルーチンのみをシステムとしてサービスすることとした。

N/3200においてMCPをインプリメントする場合に最初に直面する問題はプログラムをどのようなモードで動作させるかである。すなわちN/3200にはノーマル・モード(スーパーバイザ・モード)と制限モードがありこのいずれを使って動作させるかということである。

ノーマル・モードにおいてはすべての命令および記憶領域の参照が可能であるが、制限モードにおいては、入出力命令、記憶保護領域の参照をすることができない。

MCPは実際の使用段階ではデバッグをしつくされたプログラムであるはずであり、ノーマル・モードで動作させてもよいかも知れないがもし何らかの障害が起これば、これによって障害対策機能

プログラムあるいは各IMP固有のパラメータ領域が破壊された場合には障害対策が無意味になってしまう。

MCPを制限モードで動作させた場合には上記のような問題点は解決されるが、MCPとTIPプログラムが制限モードで同時に動作すると、領域の管理がめんどろになること、MCPの各処理タスクへコントロールを渡す場合にオーバーヘッドが大きくなること、OCP命令で実現したプログラマブル割り込みが全く使用できないこと等の欠点が出てくる。

以上のような理由から、サブネットのノードとして本来はたさなければならない機能を実行するMCPはノーマルモードで動作させ、ユーザが作成する可能性のあるTIPプログラムは制限モードで動作させることにした。

IMPプログラムは以上のような形態で動作するマルチタスクプログラム構造を持っている。

3.4.2 プログラムの構造

A 基本構造

IMPプログラムは(表3-1)に示すような割り込みを主たる要因として動作するマルチタスクプログラムである。これらの要因はマスク命令によって割り込みを禁止することができるので、それぞれの処理タスクごとにマスクパターンが決められており、そのタスク処理中にはより緊急度を要する割り込みだけがはいるようになっている。この処理レベルは0~3まで4種類ありレベル0はどのような割り込みも可能な状態をいい、1, 2, 3と順次緊急度のレベルが高くなっている。

(図3-36)はIMPプログラムにおける主要タスクとそれらのタスクがどのような要因により起動されるかを示したものである。

すなわち、④~⑥のタスクは割り込み処理ルーチンによって起動され、⑩~⑫のタスクはCPUディスパッチャーにより起動される。このうち⑩, ⑫(あるいは⑪, ⑬)はどちらの原因によっても起動される。

表 3 - 1 割り込み原因と割り込みマスク

マスク命令	マスクビット	割り込み原因となる装置
SMK'0020	1, 2	磁気テープ*
	5, 6, 7	ホストとのアダプタ
	9	紙テープ入力装置
	11	T T Y
	12	カードリーダー*
	14	ライン・プリンタ
	15	メモリ・パリティ
	16	タイマー
SMK'0420	9	多重通信制御装置 (M L C)
SMK'0520	9 ~ 12	高速通信制御装置

* 現在は利用しないが将来利用する可能性があるもの。

割り込み処理ルーティン (処理レベル 3)

② モデム入力・タスク (処理レベル 3)	
① モデム出力・タスク (処理レベル 1)	
④ ホストインタフェース・タスク (処理レベル 1)	
④ タイマータスク (処理レベル 1)	
④ M L C 入力タスク (処理レベル 2)	
① 他の割り込み処理タスク (処理レベル 1)	
② CPU ディスパッチャー (処理レベル 0)	① パイプ管理タスク (処理レベル 0)
	③ ルーティングタスク (処理レベル 0)
	③ コントロールパケット処理タスク (処理レベル 0)
	④ メッセージ組み立てタスク (処理レベル 0)
	① ホストインタフェースタスク
	④ モデム出力タスク
	④ T I P プログラムディスパッチャー (処理レベル 0)
	④ バックグラウンド (処理レベル 0)

図 3 - 36 IMP プログラム中のタスク

B パケットの流れとタスクの機能

ここでHOSTからはいってきたメッセージがMCP内においてどのように処理されて目的地HOSTに到着するかをMCPのタスク機能とあわせて説明しよう。

(図3-37)はパケットの流れを示したもので、このように、MCPはキューと割り込みを主体として動作するものである。以下にこれら各タスクの説明をおこなう。

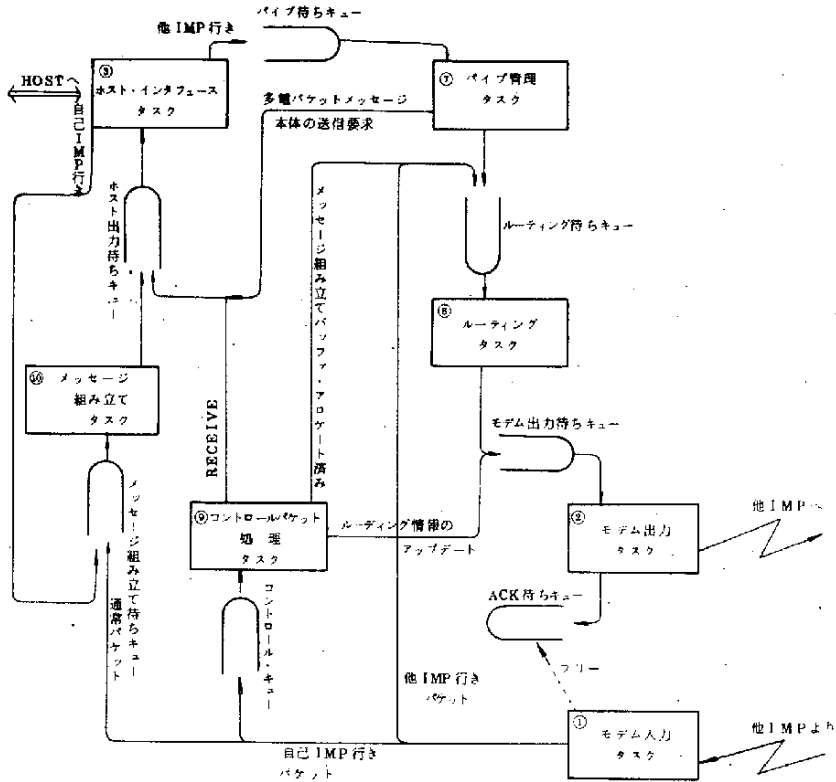


図3-37 パケットの流れ

① モデム入力タスク

48 K bit/sec の高速通信回線を通じてパケットを入力するタスクであり、この回線を接続しているIMPチャネルからの終了割り込みをトリガにして起動される。

このタスクが起動されると入力パケットにエラーがなければ、次のパケットの読み込み準備をしたのちACK待ちキューにあるパケットをフリーにする。

そして対応するIMPに対するモデム出力キューにパケットがなければACKパケットを作り出し、モデム出力キューにのせる。受け取ったパケットはその種類により、ルーティング待ちキュー、コントロール・パケット・キュー、メッセージ組み立て待ちキューなどにのせられる。

② モデム出力タスク

48 K bit/sec の高速通信回線を通じて他IMPに対してパケットを送出するタスクであ

り、この回線に接続している IMP のチャンネルからの終了割り込みおよび CPU ディスパッチャー（⑥参照）により起動される。

このタスクはモデム出力待ちキューにのっているパケットを順次当該回線に出力し、ACK 待ちキューにのせ変えるのが主たる仕事である。

③ ホスト・インタフェース・タスク

ホスト・アダプタに対する入出力を次の章で述べる HOST-IMP プロトコルにしたがっておこなうタスクであり、チャンネルからの終了割り込み、WDI 割り込みおよび CPU ディスパッチャー（⑥参照）により起動される。

このタスクは、入力データに対してはその行き先が自己 IMP、他 IMP の区別によりそれぞれメッセージ組み立てキュー、パイプ待ちキューにのせる。出力データはホスト出力待ちキューよりとり出して HOST に転送する仕事を受け持つ。

④ タイマー・タスク

IMP においてタイマー割り込みを処理するものであり、実時間の管理および ACK 待ちキューにのっているが ACK が返送されないパケットを再送のためモデム出力待ちにキューにのせ変える仕事を受け持つ。

⑤ MLC 入力タスク

MLC に接続している通信回線を通じてパケットを入出するタスクであり、TIP の場合に使用されるものである。

このタスクは TIP の端末用回線を接続しているチャンネルからの終了割り込みをトリガにして起動される。

⑥ CPU ディスパッチャー

割り込みに起因する処理が終了するとコントロールが渡されるタスクであって、すべてのキューを調べ起動すべきタスクがあるとそれを起動する。

⑦ パイプ管理タスク

サブネットにおけるメッセージフローおよびシーケンスを管理するタスクであり、CPU ディスパッチャーにより起動される。

このタスクはパイプ待ちキューにのっているパケットをルーティング待ちキューにのせ変える。また RECEIVE パケットに対しては、バッファ・アロケートを便乗（Piggy Back）させ、多重パケットメッセージに対しては、ホストからのデータ入力を促進する情報を作り出してホスト出力待ちキューにのせることなどをおこなう。

⑧ ルーティング・タスク

前節で述べた手法によりパケットが目的地に最短時間で着くような経路選択をおこなうタスクであり、CPU ディスパッチャーにより起動される。

このタスクが起動されるとルーティング待ちキューからパケットをとり出し最適ルートと思われる経路にある回線に対応するモデム出力待ちキューにのせ変える。

⑨ コントロールパケット処理タスク

ALLOC, RFA, line down RECEIVE (これらに関しては、3.3 参照のこと) など IMP-IMP プロトコルにおいて定められたコントロールパケットを処理するタスクであり、CPU ディスパッチャーによって起動される。

このタスクはそのパケットの種類によってそれぞれ異なった処理がなされる。

ALLOC パケットの場合には、メッセージ組み立てバッファのアロケートをして RFA をルーティング待ちキューにのせる。

line down の場合には、ディレイテーブルなどを書き換え必要があれば他の IMP に同様の情報を流すべく line down 情報をモデム出力待ちキューにのせる。

RECEIVE の場合には、パイプの管理テーブルのアップデート、バッファアロケートなどをおこなう。

⑩ メッセージ組み立てタスク

自己 IMP あてに転送されたパケットを reassemble してメッセージに組み立てるタスクであり、CPU ディスパッチャーにより起動される。

このタスクは、多重パケットメッセージの場合には、すべてのパケットが到着してメッセージができあがるまでパケットを保持しメッセージができ上がったならばホスト出力待ちキューにのせる。単一パケットメッセージの場合にはたどちにホスト出力待ちキューにのせる。

3.5 サブネットの解析、評価に関する定量的手法

この節ではネットワークの解析、評価に関するいくつかの定量的手法について簡単な紹介を試みる。

ネットワークの設計目標は、支えられたコスト条件、トラフィック条件等の下でメッセージ・ディレイを最小化するように各種デザイン・パラメータを決定することにある。

これは一般に次のように定式化される。

目 標： メッセージ・ディレイを最小化する

拘束条件： $\left\{ \begin{array}{l} \text{コスト一定} \\ \text{トラフィックに対する要求 (各ノード間でやりとりされるデータ量)} \\ \text{信頼性に関する要求 (例えば 2 点間に必ず 2 パスが保証されていること)} \end{array} \right.$

デザイン・パラメータ： $\left\{ \begin{array}{l} \text{メッセージ長} \\ \text{メッセージ優先処理に関する規定} \end{array} \right.$

- バッファ・サイズ
- リンク容量の割り当て
- メッセージ・ルーティングの規定
- ネットワーク・トポロジー

ここに見るように、最適化の過程にはいくつかの問題が含まれている。中でも中心的な問題としてとりあげられているものは、パフォーマンスを予測するためのメッセージ・ディレイ・モデルの作成、ルーティング法の選択、チャネル容量の最適配分問題、ネットワークのトポロジー設計等の問題である。これらの問題は様々な立場からアプローチされ解析が試みられているが、その背景には待ち行列理論、ネットワーク・フロー理論、リニア・プログラミング、インテジャ・プログラミング、ダイナミック・プログラミング等の従来の数学理論が重要な役割を果たしている。

一方、アナリティック・モデルの検証とか、解析的手法の適用が困難な場面ではシミュレーションによる解析が功を奏している。又IMPのモニタリング機能を使った実測データの収集も解析、シミュレーションで必要になる基本データ（例えばメッセージ長の分布、発生時間々隔の分布等）を集めたり、又解析結果を検証したりする目的から、非常に重要な役割を果たしている。

3-2でも述べたようにネットワークの最適化は、これらの各種手法による色々な角度からの分析結果をもとにそれを統合化し、結論が導き出される性格のものである。この過程では適切なヒューリスティック・プロセデュアが大きな効果を発揮する。

ここでは、初めにネットワークの静的なパフォーマンスを把握する目的から、BBNで検討された結果をもとにIMPスループットとメッセージ長の関係、ラウンド・トリップ・タイムとメッセージ長の関係、回線利用率を最大にするためのバッファ個数等に関してその定量的な扱いを述べる。

又一定の負荷条件の下でのメッセージ・ディレイを予測する数式モデルとして、KLEINROCKが求めたネットワーク・ディレイ・モデルを紹介する。

3.5.1 IMPの最大スループット率

IMPの最大スループット率とは、IMPが単位時間に処理することができるホストからの最大流入ビット数のことを云う。

これは、メッセージあたりのIMPの処理時間を求め、メッセージ・ビット数をこの値で割れば求めることができる。しかしIMPの処理時間を算定するには、単にメッセージの有効ビット数ばかりでなく、パケット数、ヘッダー、ACK、RFNM等のオーバーヘッド・トラフィックも考慮しなくてはならない。以下ではまずIMPのメッセージあたりの処理時間の算定法について説明し、それを用いて最大スループット、最大回線トラフィックがどのように表現されるかを示す。

始めに I_m , I_r , C , P , V を各々次の様に定義する。

I_m : Bビット(有効ビット数)からなるメッセージを入/出力するのに要する時間(メモリからモデムに伝送する時間)

I_r : RFNMを入/出力するのに要する時間

C : パケットあたりのプログラムの処理時間

P : Bビットからなるメッセージがパケット化された時のパケット偶数

V : IMPが定常的に動作している時のオーバーヘッド・タイムをパーセントで表現したもの

これらの量が求まればIMPのメッセージあたりの処理時間 L は次のように表現されることになる。

$$L = (P \times C + I_m) \times (1 + V) + (C + I_r) \times (1 + V)$$

メッセージの処理時間 RFNMの処理時間

ここで I_m 、 I_r 等は次の様にして求めることができる。

今パケットあたりのオーバーヘッド・ビットを S 又単位時間あたりの入出力伝送率を R とすれば I_m 、 I_r は次式のように表現されていることが分る。

$$I_m = \frac{2(B + P \times S)}{R}$$

$$I_r = \frac{2S}{R}$$

又、オーバーヘッド・ビット S はパケット・ヘッダのビット数とACKのビット数の和として求めることができる。ARPAの場合は $S = 144$ ビットとなる。

又 R はDDP-516の場合は約4200 KB/Sである。次にパケットあたりのプログラムの処理時間 C を求めなければならないが、これは実際のプログラム処理の内容から決定しなくてはならない。

ARPAの場合その内訳を次のように見積っている。

パケットの送信 (100 サイクル)

パケットの受信 (100 サイクル)

受信パケットの処理 (150 サイクル)

ACKの送信 (100 サイクル)

ACKの受信 (100 サイクル)

ACKの処理 (150 サイクル)

1パケットあたりのプログラム処理時間 C はこれらの総和として求めることができる。この場合だと $C = 700$ サイクルとなる。

IMPオーバーヘッド V はIMP内の周期的な処理に要するオーバーヘッド・タイムをパーセントで表

わしたせで ARPA では約 5% と見積っている。

この様にして求めた L を使って、IMP の最大スループット T は次の様に表現することができる。

$$T = \frac{B}{L}$$

又回線トラフィック率 R は

$$R = (\text{スループット率}) + \frac{(1 \text{ メッセージあたりの全オーバーヘッド・ビット})}{(1 \text{ メッセージあたりの IMP の計算時間})}$$

であるから

$$R = T + \frac{(P+1) \times (S+H)}{L} = \frac{B + (P+1) \times (S+H)}{L}$$

となる。但し H はパケットと ACK に付加されるハードウェアのオーバーヘッド・ビットの和を表わす。ARPA ではパケット、ACK の両方を加えると $H=144$ (ビット) になる。図 3-46 は BBN で求められた DDP-516 に関する結果を示したものである。

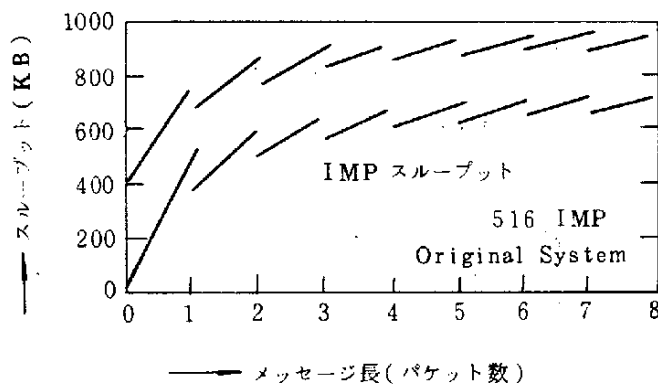


図 3-46 メッセージ長とスループットの関係

3.5.2 ラウンド・トリップ・タイムとメッセージ長の関係

ラウンド・トリップ・タイムとはメッセージを発信してから RFNM を受け取るまでの時間を云う。ここでは、無負荷状態のネットワークでのラウンド・トリップ・タイムを求める方法を述べる。

パケットを遅延させる時間要素は IMP の処理時間と、伝送遅延時間 T_p 及び回線の伝播遅れ L である。従って最初のパケットによる遅延時間は $(C + T_p + L)$ で与えられる。

最初のパケットに続く次からのパケットは $(C + T_p)$ 時間ずつずれて送られることになるから、今 P 個のパケットからなるメッセージが H ホップ先の目的に送られる時のラウンド・トリップ・タイム R_t は次のように表わされる。

$$R_t = H \times (C + T_p + L) + (P - 1) \times (C + T_p) + H \times (C + T_r + L)$$

最初のパケットによる遅れ 残りのパケットによる遅れ RFNMによる遅れ

但し T_r は RFNM の伝送遅延時間を表す。

図3-47はARPAネットワークに関して、BBNで求められたメッセージ長とラウンド・トリップの関係を図示したものである。

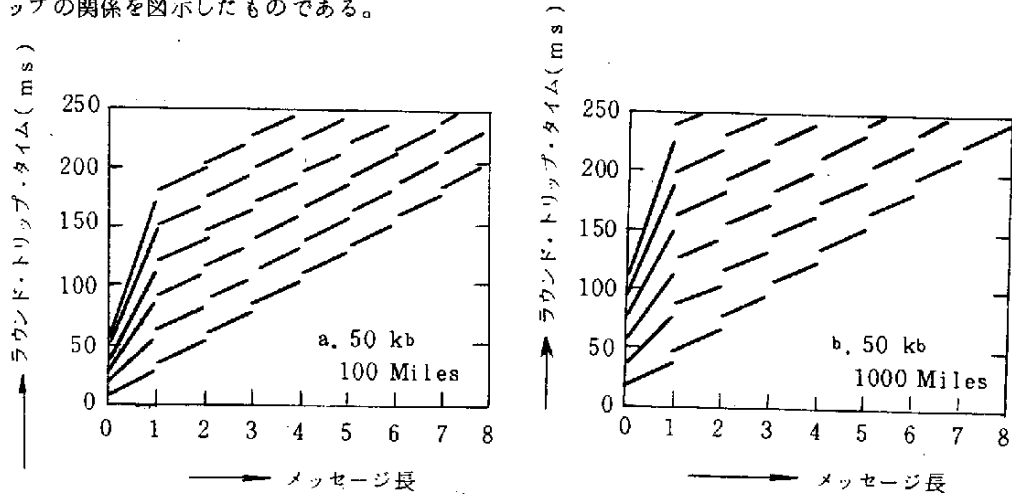


図3-47 1から6ホップまでのメッセージ長とラウンド・トリップ・タイムの関係

3.5.3 回線利用率とバッファ個数

蓄積交換では、パケットを送出してからACKを受信するまではその送信バッファを解放することができないから、回線能力の許す限り、連続的にパケットを送信するような場合は、送信バッファをいくつか用意しておかないと、次のパケットが送れる状態になっても送信バッファがまだ使用可能な状態にならないような事態が起り、パケットと次のパケットの間に無駄なタイム・ギャップがとられ回線利用率が悪くなる。

ここでは回線利用率を100%にするためには最低限バッファをいくつ用意すればよいかと云う問題を検討してみる。

今 P_p , P_a , T_p , T_a , C を各々次のように定義する。

P_p : パケットの伝播遅れ

P_a : ACKの伝播遅れ

T_p : パケットの伝送遅延時間

T_a : ACKの伝送遅延時間

L : ACKを伝送する前のIMPの処理時間

始めのパケットを送出して次のパケットが送出可能な状態から、始めのパケットのACK

を受け取るまでの時間はこれらの総和として表わされる。従って、この時間内に IMP が処理可能なパケット数が必要なバッファ個数となる。バッファ個数を N とすればこの関係は次式で表わされる。

$$N = (P_p + T_p + L + P_a + T_a) / T_p$$

ここで $P_p = P_a$ であるから N は次の様に書き換えることができる。

$$N = \frac{2 P_p}{T_p} + 1 + \frac{L + T_a}{T_p}$$

この式から分るように N は回線の長さに比例しパケット長さに反比例する。

T_p の計算には、長いメッセージと、短いメッセージの混合比を考慮しなくてはならない。最小パケット (152 ビット) に関する伝送遅延時間を T_s 、最大パケット (1160 ビット) に対する伝送遅延時間を T_L 、又その混合比を $x : y$ と仮定すると T_p は T_s 、 T_L の重み付き平均として次式で求めることができる。

$$T_p = \frac{x T_s + y T_L}{x + y}$$

又 T_a は簡単に T_s 、 T_L の平均値として $T_a = \frac{T_s + T_L}{2}$ として求める。図 3-48 は各種のメッセージ・ミックに対する回線長とバッファ個数の関係を示したものである。

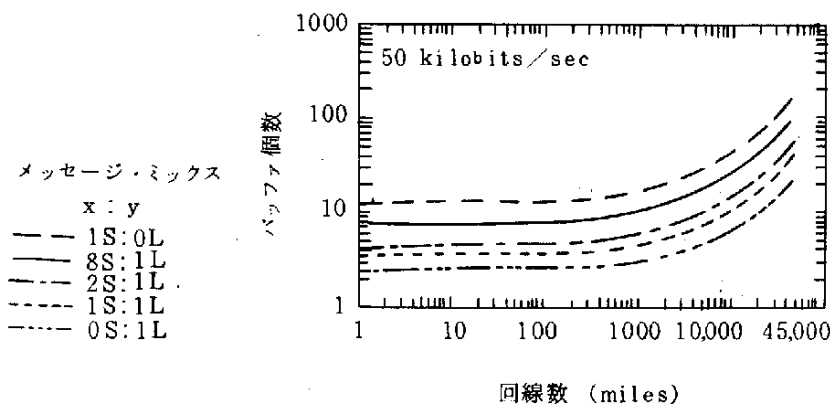


図 3-48 回線長とバッファ個数の関係

3.5.4 メッセージ・ディレイ・モデル

ネットワーク・パフォーマンス・メジャの最も重要なものの 1 つとして考えられているものにメッセージ・ディレイがある。以下ではメッセージ・ディレイを解析的に求める手法について考案を加える。始めにネットワークに於けるメッセージ・ディレイの原因となる要素を考えなくてはならないが、これには一般に次の様な要素がその原因としてあげられる。

- IMP内に於けるキューイング・ディレイ
- IMPの処理時間による遅れ
- IMPのストレージ・ブロック
- ACKやルーティング情報を送るため、チャネルの実効容量が少くなる。
- チャネル・エラーによるパケットの再送
- ルーティングによるパケットのループ現象

ネットワークに於けるメッセージ・ディレイを解析する方法としては、IMP間の依存関係を排し、各段が独立であると云う仮定のもとに待ち行列理論を適用することが一般に行われている。即ち、各段が独立であると云う仮定のもとに各段に於ける遅延時間を求め、これをネットワーク全体について、トラフィック条件を考慮して平均化することにより平均メッセージ・ディレイを求める方法である。

しかし上に述べたような各種のディレイ要素を考慮すると、各段独立の扱いは厳密には成立しない。即ちIMPのバッファ容量は有限であるからこれが完全に一杯になってしまうと前段からの出力を受けとることが出来なくなり、IMP相互間に依存関係が生じることになる。このような問題をネットワーク全体について考慮して扱うことは現在の待ち行列理論では不可能な範囲になっている。

しかし我々が解析の対象としている問題は正常の負荷状態に於けるネットワークのメッセージ・ディレイであるから、このような状態に於てはさきに述べたようなバッファのブロック現象はめったに発生しないと考えてもさしつかえない。従ってこのような状況では各段独立の扱いは十分成立し得る分である。

KLEINROCK は以上の様な仮定の下に、ネットワーク・トポロジー、チャネル容量、ノード・トラフィック、が与えられた時のメッセージ・ディレイについて解析を行っている。(但し、ルーティング法は簡単のため固定ルーティング法を仮定して行っている。)

以下ではKLEINROCK の考えたネットワーク・ディレイ・モデルについて簡単に解説しておく。

初めに解析の基本となる、待ち行列モデルとしてM/M/1系の解析結果を以下に引用しておく。定常な単一サービス系では到着時間々隔がポアソン分布に、又サービス時間が指数分布に従うと仮定すれば平均待ち時間Tは次式で与えられる。

$$T = \frac{\bar{t}}{1 - \rho}$$

但し $\bar{t}$ はサービス時間の分布に関する一次のモーメント又は ρ は平均到着率を λ とした時 $\rho = \lambda \bar{t}$ で与えられる。

この結果をネットワークの解析に適用するためには、ストカスティック・ネットワークを単一サー

ビス系に分解することを考えなくてはならない。以下ではこの一連の過程について説明する。

今、パケット長は平均 $1/\mu$ の指数分布に従い IMP J から IMP K に向うパケットの到着が、平均 γ_{jk} のポアソン分布に従うとする。又これらのパケットが IMP J から IMP K に行くのに要する時間遅れの平均を z_{jk} とすると、パケットの平均時間遅れは次式で与えられる。

$$T = \sum_{j, k} \frac{\gamma_{jk}}{\gamma} z_{jk}$$

これを単一チャネルに関する遅れと云う観点から書き改めると、次のようになる。

$$T = \sum_i \frac{\lambda_i}{\gamma} T_i$$

(但し λ_i は i 番目のチャネルを通過する平均トラフィック量 (パケット/秒))

T_i は M/M/1 系の解析結果から、 $T_i = \frac{1}{\mu C_i - \lambda_i}$ として求めることができるから

基本的にはこの 2 つの式から完全に T を求めることができる。

実際には、伝播遅れ、オーバーヘッド・トラフィック等の影響も考えなければならないから、これらを考慮した結果として最終的に次式が得られる。

$$T = K + \sum_i \frac{\lambda_i}{\gamma} \left[\frac{1}{\mu' C_i} + \frac{\lambda_i / \mu C_i}{\mu C_i - \lambda_i} + P_i + K \right]$$

ここで $1/\mu'$ はホストからのパケットの平均長、 $1/\mu$ は ACK, RFNM 等を含めたパケットの平均長、 K はパケットあたりの IMP の処理時間を表す。又 P_i は i 番目のチャネルの伝播遅れである。

図 3-49 はこの式から求めたシングル・パケットに関するスループットとメッセージ・ディレイの関係を表わしている。プロットはシミュレーションによる結果を示したものであるが両者は非常によく一致していることが分る。

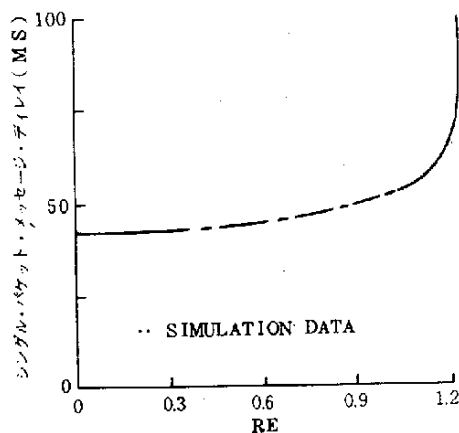
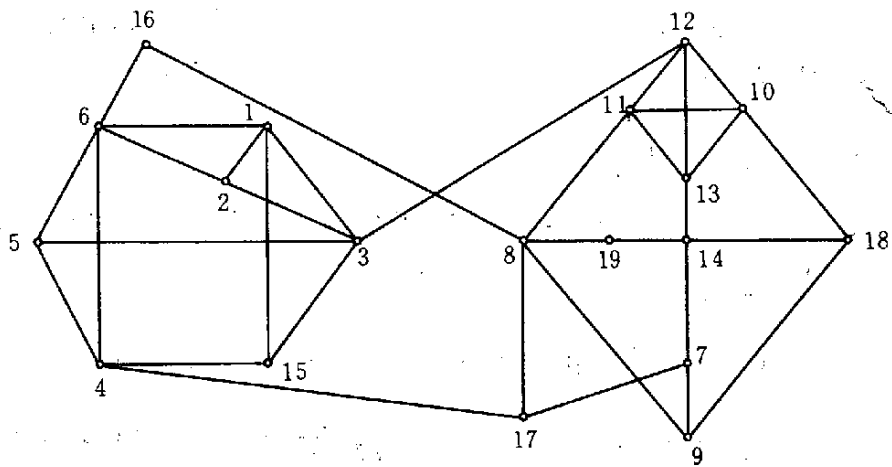


図 3-49 スループットとメッセージ・ディレイの関係

又図3-50はこの解析に用いられたネットワーク構成と、トラフィック・マトリックスを示したものである。



		DESTINATION NODE																		
SOURCE NODE		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
	2	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	0
	3	1	0	0	0	0	1	0	0	0	0	1	0	1	0	1	0	0	0	2
	4	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0
	5	1	1	1	1	0	3	0	0	3	3	1	0	6	1	1	0	0	0	0
	6	1	1	1	2	2	0	1	1	1	1	1	1	2	1	1	1	1	1	1
	7	1	0	0	1	0	2	0	0	2	0	1	0	1	0	1	1	0	0	0
	8	1	10	1	2	1	2	1	0	1	0	2	5	1	1	1	20	0	0	1
	9	0	1	1	0	0	1	2	1	0	0	1	1	0	0	0	0	1	0	0
	10	0	0	0	0	0	1	0	0	0	0	1	1	1	0	1	1	0	0	0
	11	1	1	1	2	1	1	1	1	1	2	0	2	1	1	1	1	1	1	1
	12	1	0	0	0	0	1	0	1	1	0	1	0	0	1	1	2	0	0	1
	13	0	0	0	0	2	1	1	0	0	1	1	0	0	0	0	1	0	0	0
	14	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	2	0
	15	1	1	1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	2
	16	1	5	0	0	0	1	2	8	1	0	1	5	0	0	1	0	0	0	1
	17	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
	18	0	1	0	0	0	1	0	0	0	1	2	0	0	2	0	0	0	0	1
	19	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

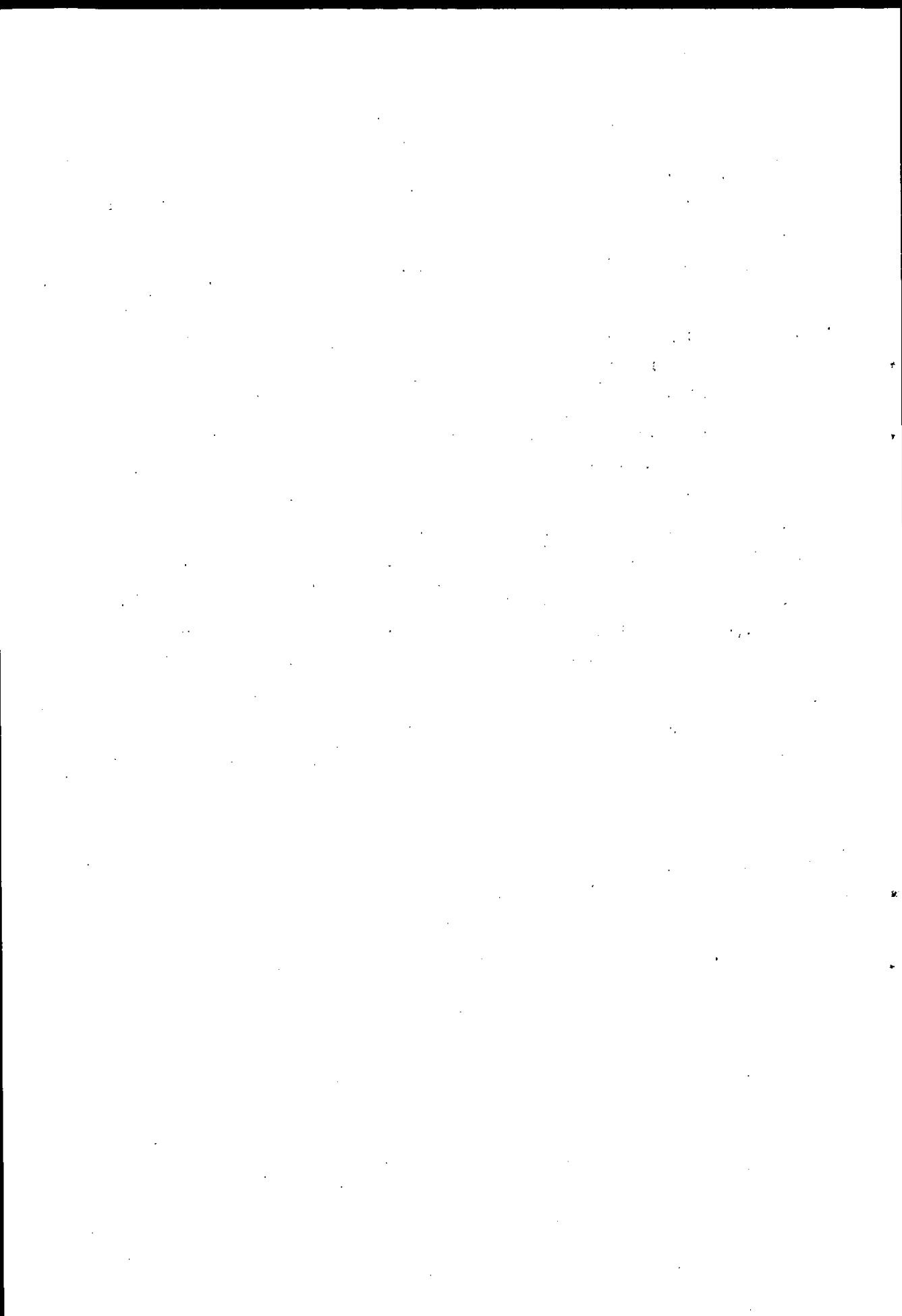
図3-50 ネットワーク構成及びトラフィック・マトリックス

4. HOSTの基本ソフトウェア

4 HOSTの基本ソフトウェア	139
4.1 基本プロトコル	139
4.1.1 基本プロトコルの概要	139
4.1.2 プロトコルの基本的性格	140
A 超高性能コンピュータの共用	141
B 特殊な入出力装置の共用	141
C プログラムシステムの共用	142
D 協用ジョブの遂行	142
E 共用データファイルの利用	143
F 結 論	143
4.1.3 NCPサービスコマンド	144
A プロセス間コミュニケーション	144
a. プロセス間コミュニケーションに関するコマンド	146
b. コネクションの確立	149
c. プロセス間コミュニケーションの問題点	150
d. プロセス間コミュニケーションのコマンド体系	154
B プロセスの発生	156
C プロセス間の同期	159
4.1.4 HOST-HOSTプロトコル	160
A フローコントロール	160
B コントロールコマンド	162
4.1.5 HOST-IMP プロトコル	169
A チャネルアダプターのコマンド	169
B データフォーマット	170
C HOST-IMP のデータ転送制御	175
D 初期状態手順	178
E データの正規転送手順	178
F 衝突手順	181
G 緊急処理手順	181

H 終了処理手順	181
4.2 F-NCP	182
4.2.1 FACOM 230-60/75 MONITOR VI システムの概要	182
A ジョブの処理形態	185
B 領域管理とファイル管理	189
4.2.2 F-NCP基本構想	192
A F-NCP概要	192
B F-NCP機能	193
C サブモニタ	193
D 検討事項	194
4.2.3 CCAインターフェース	195
A アテンション・アペンデージ・インターフェース	195
B 衝突	195
C モニターNCP間の衝突	196
4.2.4 ETAMインターフェース	196
A NCP-ETAM インターフェースの考え方	196
B 問題点	197
4.2.5 プロセス間コミュニケーション	197
A 第1案	198
B 第2案	198
C 検討結果	198
4.2.6 プロセスの起動と監視	199
A イニシェータの機能	199
B ターミネータの機能	200
C プロセスの監視	201
4.2.7 F-NCPに対するサービス要求	201
4.2.8 会計情報	201
A NCP会計ルーチン	202
B ネットワーク使用会計情報	203
4.2.9 高位プロトコル	205
A RDASプロトコル	205
B TELNETプロトコル	206
4.2.10 NCIC情報採集	207

4.3	H-NCP	207
4.3.1	EDOS-MSO の概要	208
A	ジョブ管理	208
B	タスク管理	211
C	主記憶装置の割当て	212
D	FCP	213
4.3.2	H-NCPの基本構想	216
A	H-NCPの機能	217
B	H-NCPの構成	219
C	コントロール及びデータ・フロー	222
4.3.3	H-NCP機能の実現方法	226
A	プロセス間コミュニケーション	226
B	プロセス管理	229
C	HOST-IMP インターフェース	233
D	コンピュータ・リソースの把握	235
E	高位のプロトコル処理	236
4.3.4	ネットワーク・プロセス	239
A	ネットワーク・プロセス間の会話	239
B	ディスパッチ・モジュール	242
C	ユーザレベル・プロトコル	244



4. HOSTの基本ソフトウェア

コンピュータ・ネットワークにおいて基本的にサービスしなければならないのはHOST間の通信である。

JIPNETにおいて、このレベルの通信を保障するためには、サブネット間、HOSTとサブネット間およびHOSTとHOST間のソフトウェア的結合が必要である。

このうち、HOSTとサブネットのハードウェア的結合、サブネットのソフトウェアに関してはそれぞれ第2章、第3章に述べた。

この章においては、HOST中であってネットワークに関するサービスを行なうNCP (Network Control Program) と、NCPとサブネットのソフトウェア的結合方法について述べる。

このうち、4.1においては、それぞれ固有のOSの下で動作しているHOSTを結合する分散型のOSにおいて協同して目的を遂行するための必須条件となるOS間の通信規約を決めるとともに何故そのような形をとったのかについて述べる。

4.2、4.3においては4.1に決められた通信規約を実現させるためには、それぞれFACOM 230/60 Monitor VI, HITAC 8450 EDOS-MSOにおいてどのようなOSとのインターフェース上の問題点があるかについて述べる。

4.1 基本プロトコル

4.1.1 基本プロトコルの概要

JIPDECのコンピュータ・ネットワークJIPNETは、たがいに独立した機能をはたす各HOSTと、IMPおよび回線からなるサブネットという系を結合することにより作成されるものである。

このようにそれぞれ独立した系により共同した目的を達成させるためには、それぞれの系の間での機能分担と情報伝達のための正確な取り決めが必要となってくる。

プロトコルとは、ある系と他の系の間でデータの受け渡し手順、あるいはタイミング、内容などを取り決めたものである。

HOSTからみた場合サブネットは1つの系であり、この系との結合のためにHOST-IMPプロトコルがあり、そして他HOSTに対する結合のためにHOST-HOSTプロトコルがある。

HOSTが関係する基本プロトコルはこの2つであるが、さらに機能レベルのものを任意のHOST間で定義することができる。

基本プロトコルは(図4-1)に示すような構成になっている。すなわちサブネットとHOSTをソフトウェア的に結合し、その間でのデータ授受などの規約を定めたHOST-IMPプロトコルおよ

び、相対したHOSTとの結合をするためにコネクションの確立方法、プロセスの起動などを定めたHOST-HOSTプロトコルである。

実際にユーザが利用するのはHOST-HOSTプロトコルではなく、Network Control Program (NCP) がそのHOSTのOS特有の使い方にあわせてサービスしているNCPサービスコマンドである。

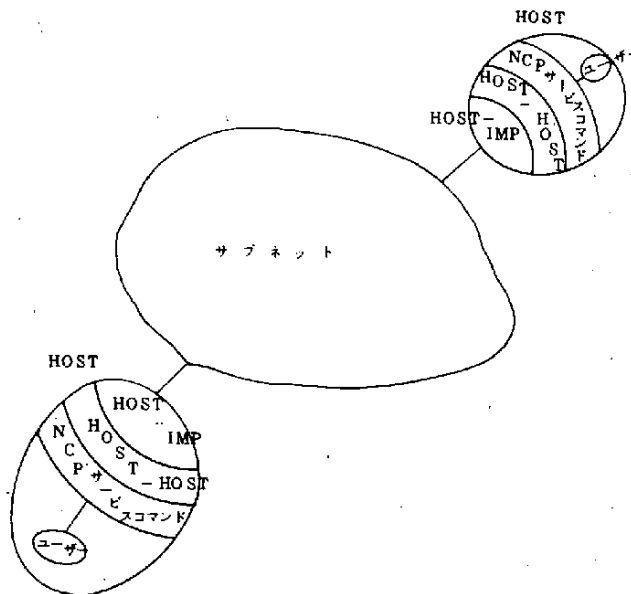


図4-1 基本プロトコルのレベル

4.1.2 プロトコルの基本的性格

JIPNET コンピュータ・ネットワークは、利用範囲を明確に規定したものではなく、むしろコンピュータ・ネットワークを作成することにより、ハードウェア、ソフトウェア、データベースなどのコンピュータ・リソースを共用し、より高度な利用を可能にすることを目標としているものである。

このために、利用あるいは使用形態を制限しない、汎用性、拡張性を持つことが、プロトコルの機能を決定するための主たる目標となった。

一般にリソース・シェアリングとは次の3つの観点から考えることができる。

① ハードウェアの観点

- 超高性能コンピュータの共用。
- 特殊な入出力装置の共用。

② ソフトウェアの観点

- プログラム・システムの共用。

- 協同ジョブの遂行

- ③ データベースの観点

- 集中形データベースの共用

- 分散形データベースの共用

これらの利用を可能にするためにはどのような方法があるかをそれぞれ検討してみよう。

- A 超高性能コンピュータの共用

超高性能コンピュータは、コンピュータ・ネットワークにとって非常に重要なリソースである。このようなコンピュータは高価であること、処理能力が大きいために多くの所に設置しても無駄が多いなどの理由により限られた場所にしか設置されない可能性が強い。

コンピュータ・ネットワークにこのような超高性能のHOSTを結合し、そのリソースを共用することは非常に有効なことである。

ここで超高性能コンピュータを他のHOSTから利用する方法を考えてみよう。

基本的には、1個のジョブが複数個のHOSTを利用することになるのでこれを有効に利用する方法を検討することになる。

複数個のCPUを持つコンピュータを有効利用する方法として両方でメモリをシェアする方法があるが、コンピュータ・ネットワークにおいてはこれが実現できたとしても、メモリの参照要求があってから実際に参照できるまでには数百ms~1secかかり、CPU機能を殺してしまうことになり意味がない。

このために利用するブロック単位にメモリを転送する方式を考えなければならないが、このようにすると一方のCPUだけがそのメモリの内容を参照することになり同時並行的なCPUの利用が不可能となる。

これを避けるため、サブルーチンを他のコンピュータにおいておき、必要な時点で制御をそのルーチンに渡し、同時にパラメータの値を転送する。こうしてその処理が終わったならば処理結果の値を返送してくるような利用方法を考える方がすっきりしてくる。

また、高性能コンピュータを利用しなければ不可能な処理量を考えれば、1つのジョブステップとしての処理を依頼し結果をファイルという形で転送しても問題はないであろう。

いずれにしろ、このようなコンピュータの利用のためには、ジョブステップあるいはサブルーチンに相当するものを他のホストにおいて発生させ、それにデータを送信あるいは返送する機能をサービスする必要がある。

- B 特殊な入出力装置の共用

漢字入出力装置、手書き文字入力装置、図形入出力装置あるいは実時間応答を要求されるような入出力装置などの共同的な利用を考えてみる。

これらの装置はそのアクセス方法、データ形式が一致しているとはいえない。このために共

通なアクセスメソッドを設定してサービスをおこなわないと、アクセスのサービスプロトコルだけで膨大な量となり実用にならない。

このような場合には、ファイルあるいはレコードレベルのアクセス法で統一し装置ごとのインターフェースの違いは各HOSTが責任を持っておこなわざるを得ない。

C プログラム・システムの利用

各ホスト用にそれぞれ開発されたソフトウェアを他ホストから利用するのが目的である。

そのソフトウェアがバッチ処理を対象としたものであるか、TSS利用のものであるかによって異なるが、コンピュータの処理は終局的には(図4-2)のように、あるデータを加工して出力することである。

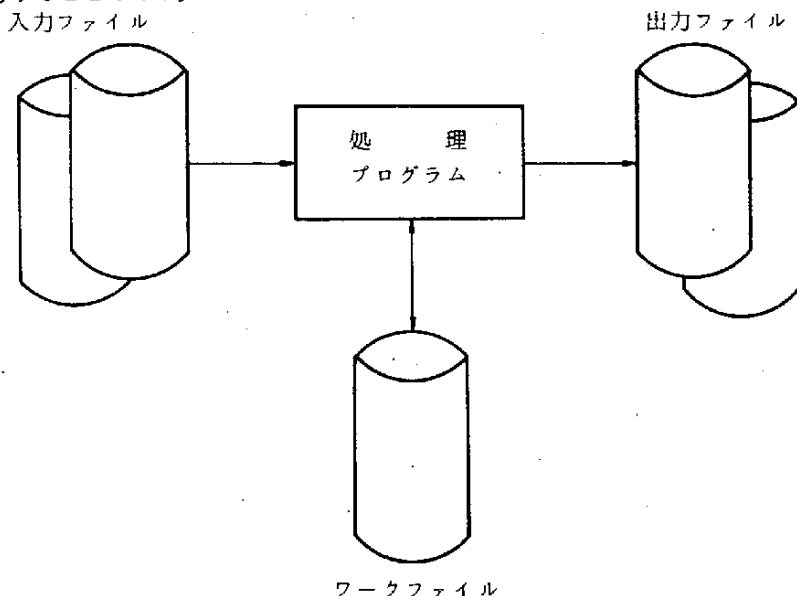


図4-2 コンピュータの処理

他ホストのソフトウェアの利用というのは2つの方法がある。すなわち入力ファイルをそのホストに転送して処理をしてもらい、処理の終了後に出力ファイルを返送してくる方法と入力ファイル、出力ファイルの両者を自己ホスト内に持ち処理プログラムの入出力指令に対応してファイルに対して入出力をおこないデータを送受信する方法である。

TSSの場合には、入出力ファイルの一つが端末になったと考えればよい。

これらの機能を実現するためには、入出力の命令部分を他ホストにおいてシミュレートし、実行する方法を講ずればよい。

D 協同ジョブの遂行

協同ジョブは、複数個のコンピュータが協同して遂行する1個のジョブをいう。これはコンピュータの性能あるいは機能をフルに生かすようなジョブの分割をすることにより、より高度

の利用が可能となる。

たとえば、高速演算機能を持つHOSTにおいては演算だけをおこないデータ・ベースの存在するHOSTにおいてはデータベースの処理をおこなうなどのジョブの分割方法がある。

これが実際に各HOSTにおいて実現されるためには、それぞれのHOSTにプロセスが存在し、それらのプロセス間のコミュニケーション、データの授受、および同期の調整などをおこなうことが必要となる。

E データベースの共用

データベースの共用はそれぞれのHOSTに分散しているファイルへのアクセスを可能にすることである。

これにはレコードのレベルでおこなうか、ファイル単位でおこなうかという問題があるが、レコードのレベルの方がより汎用性を持つのでこれの実現のために必要なプロトコルを考えてみる。

これにはデータベース・アクセス法を基本プロトコルとして用意する方法と、プロセス間コミュニケーションを基本プロトコルとして用意し、データベースへのアクセスは高位のプロトコルとしてサービスする方法がある。

前者の方法は、共用データファイルのファイル形式が一定である場合には、そのアクセス法だけをサービスすればよいので、OSのオーバーヘッドも少く実現も容易である。しかしながら一般的にいてファイルの形式やアクセス法が各データファイルおよび特定HOSTによって異なっており、これを基本プロトコルの段階でサービスするのは困難である。

後者の方法は共用データ・ファイルをアクセスするプロセスを高位プロトコルのサービスとして用意し、このプロセスとコミュニケーションすることにより実際のデータ・ファイルをアクセスすることになる。この方法では、共用ファイルのファイル形式、ファイルに対する各HOSTにおけるアクセス方式の違いなどはすべて、ファイルをアクセスするプロセスで処理でき単純かつ汎用性、拡張性を持っている。この場合に一番問題となるのは、使い易く、汎用性を持った高位プロトコルを決定することである。

F 結 論

以上に述べた要求をすべて満足するような基本プロトコル体系を作らなければならない。

ここで現在までに報告されているコンピュータ・ネットワークのプロトコルに関して検討をくわえてみよう。

これらのプロトコルはネットワークの利用目的から分類することができる。

第1のグループは、Network/440、SOCネットワークにみられるように、バッチジョブを他のコンピュータに処理してもらい、結果を返送してもらうことを主体として考えられたものである。

このプロトコルは、ジョブデック、ファイルを送信するコマンド、ジョブの実行を指令するコマンド、出力ファイルを返送するコマンド等よりなっている。

第2のグループは、第1グループのものをTSSを対象とするものに代えた形でCYBERNETがこれにあたる。

この場合にはファイルのことを無視すれば、他HOSTは複合端末を持ったフロントエンドプロセッサ(FEP)という形で実現される。

第3のグループはバッチ、TSSを問わずどのような形でも実現できるよう汎用性、拡張性を持たせたものであり、ARPA, MERITなどのプロトコルがこれにあたる。

これは、プロセス間コミュニケーションを基本とするものである。HOST間のコミュニケーションはすべてこれによっておこなわれ、TSS端末、ジョブの受け付けなどの機能はそれぞれのプロセスの責任においてローカルな問題として処理される。

前にも述べたようにJIPNETは、特定の目的を持った専用ネットワークではなく、汎用性を持つことが目標である。

このような理由から、JIPNETのプロトコルは、プロセス間コミュニケーションを基本プロトコルとし、それぞれの機能本位の利用に関しては高位のプロトコルとしてサービスすることにした。

基本プロトコルの内容については4.1.4で詳しく述べる。

4.1.3 NCPサービスコマンド

NCPサービスコマンドは、ネットワークを利用しようとするユーザに対して公開された最低レベルの機能をサービスするものである。実際にユーザが利用するのはこのコマンド以外の高位プロトコルであることが多い。

このコマンドは次の節で規定するHOST-HOSTプロトコルに対応する機能をユーザが使い易いように各HOSTのNCP作成者が自由に決めてよい性質のものである。

ここでNCPサービスコマンドを説明する理由はHOST-HOSTプロトコルを決めるにあたって、どのような使い方を想定していたかを明確にするためである。

前節で述べたように、JIPNETの基本プロトコルはプロセス間コミュニケーションにプロセスの発生などの機能を追加したものである。

以下にそれぞれの基本機能についての考え方を述べる。

A プロセス間コミュニケーション

プロセス間コミュニケーションとは、2つのプロセスの間での情報の授受をいうが、ここではNCPを通して2つのプロセス間においてコネクションの確立、コネクションを通じたデータの授受、コネクションの解放を意味するものとする。

コネクションとは2つのプロセス間で、一方のプロセスのデータ出力が他方のプロセスのデータ入力となるような関係をいう。

コネクションは(図4-3)に示されるように、ユーザ・プロセス中にある入出力に関する情報を記述したもの……これをPort という……に対してユーザが全ネットワークに対してグローバルな名前……これをPort-idという……をつけ、これが他のプロセスのport-idとNCPによって論理的に結合されることによって確立されるものである。

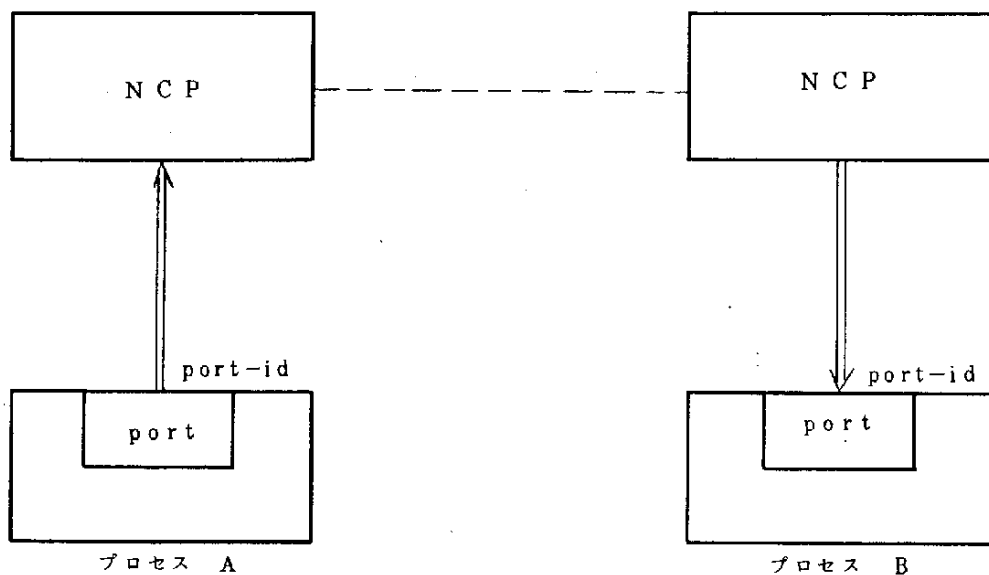


図4-3 コネクション

Port-idは(図4-4)に示されるようにユーザごとにつけられるものである。このためユーザは他のユーザのことを意識することなく連番をつければよい。ただNCPはユーザが正しく自分のユーザ番号を使用しているかどうかをチェックする必要がある。こうしないと、あるユーザのプロセスは知らないうちに間違って他のユーザのプロセスと結ばれることがあるからである。

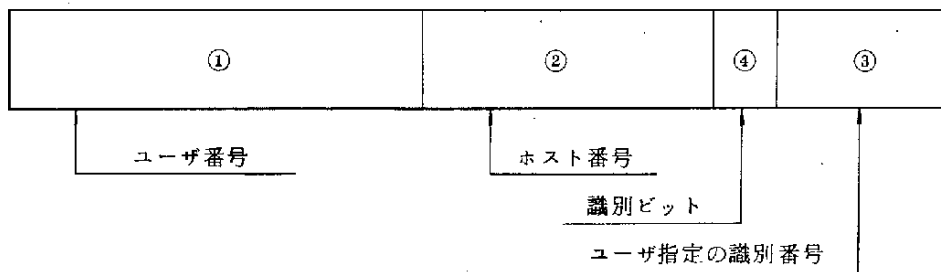


図4-4 port-idおよびecd-idの形

① ユーザ番号 (24 ビット)

ユーザを識別するために使用される。最初の 8 ビットはユーザが所属するホスト番号のために使用する。

残りの 16 ビットは各ホストにおいて自由に使用してよい。

② ホスト番号 (8 ビット)

Port あるいは ecb が属するプロセスの存在ホストを示す。

③ ユーザ指定の識別番号 (7 ビット)

Port-id として使用する場合には、7 ビット目が 1 のとき、Write port-id であり、0 のとき Read port-id となる。

④ 識別ビット (1 ビット)

ecb-id ならば 1, port-id ならば 0

ecb-id とは event control block-id でプロセスの同期をとるときに使用する。

a プロセス間コミュニケーションに関するコマンド

プロセス間コミュニケーションをユーザに対してサービスするためには、コネクションの確立、データの授受、コネクションの解放を要求するサービスコマンドを用意すればよい。

ここにおいてコネクションは、その両端となるプロセスの合意の上で確立されるものである。

通常ユーザが 2 つの HOST でプログラムを作成しそれをプロセスとして両方で実行させ、両者の間にコネクションを確立する場合にはこれでよい。

しかしながらユーティリティ・プロセスたとえばファイル・トランスファーのようにユーザプロセスからの要求によりコネクションを確立するようなものに関しては、ユーザ・プロセス側は相手側の port-id を知っているけれどもユーティリティ・プロセスの方は相手側の port-id を知らない。このようなものに対してサービスをするためには NCP はコネクションの確立要求を 1 つのイベントとしてこれに対する待ち状態をポストするコマンドを用意しなければならない。

以下にプロセス間コミュニケーションをおこなうために必要なコマンドを説明する。

(1) コネクション確立要求 (Open) コマンド

一般形

OPEN <port address>

パラメータ

<port address> : <port> の先頭番地を示す。

機 能

コネクションの確立を要求するコマンドである。

(2) READ コマンド

一般形

READ <読み込み情報番地>

パラメータ

<読み込み情報番地>: 読み込み情報の先頭番地を指す。読み込み情報は、<ecd 領域>、<port の先頭番地>、<読み込み領域の先頭番地>、<読み込み長>より成る。

<ecb 領域>は、READ コマンドの完了情報が post される。

<port の先頭番地>は、READ の対象となる port を指す。

<読み込み領域の先頭番地>は実際にデータが読み込まれる領域である。

<読み込み長>は、ユーザが読み込みたい長さを指定する READ コマンドの実行が完了すると実際の読み込み長がセットされる。

機 能

このコマンドを実行すると制御は次の命令にもどされるので、コマンドの終了はイベント待ちコマンド (wait) により知ることができる。

(3) WRITE コマンド

一般形

WRITE <書き込み情報番地>

パラメータ

<書き込み情報番地>: 書き込み情報の先頭番地を指す。書き込み情報は、<ecb 領域>、<port の先頭番地>、<書き込み領域の先頭番地>、<書き込み長>より成る。

<ecb 領域>はWRITE コマンドの完了情報が post される。

<port の先頭番地>は、WRITE の対象となる port を指す。

<書き込み領域の先頭番地>は実際に書き出されるデータの格納領域である。

<書き込み長>は、ユーザが書き出したいデータの長さを示す。この長さは port で指定した長さ以下でなければならない。

機 能

このコマンドを実行すると制御は次の命令にもどされるので、コマンドの終了はイベ

ント待ちコマンド (wait) により知ることができる。

このコマンドを同一コネクションに対して、イベント待ちをしないで連続的に発信すると NCP は強制的にそのプロセスを待ち状態にすることがある。

(4) CLOSE コマンド

一般形

CLOSE < port の先頭番地 >

パラメータ

< port の先頭番地 > : 解放するコネクションの一端となっている port を指す。

機能

確立されているコネクションの解放をする。

解放の結果は、port の中にセットされる。

(5) コネクション待ち (LISTEN) コマンド

一般形

LISTEN < my port-id >, < ecb 領域 >

パラメータ

< my port-id > : Listen の対象となる < port-id > を指す。

< ecb 領域 > : Listen により通知される < port-id > がはいる < ecb 領域 > を示す。

この ecb 領域は 40 ビットである。

機能

このコマンドは、他プロセスから Listen で指定した < my port-id > に対して Open コマンドが発信されたかどうかを調べるものである。

NCP が < my port-id > に対しての RFC マクロを受信すると < ecb 領域 > に他プロセスの < port-id > を POST する。

一度 POST されると Listen は無効になるので、再び RFC マクロの受信を聞くためには Listen を再発行しなければならない。

(6) イベント待ち (WAIT) コマンド

一般形

WAIT < ecb 個数 >, < ecb 指定 >, ………, < ecb 指定 >

パラメータ

< ecb 個数 > : この WAIT コマンドで < ecb 指定 > をいくつするかを書く。

< ecb 指定 > : < ecb-id > (< ecb 領域 >) あるいは < ecb 領域 > のいずれかである。他のプロセスから POST コマンドによるイベント通知を待つ場合には前者を指定する。

機 能

指定された<ecb>に対してすべて情報がPOSTされるまで待ち状態のままになる。

b コネクションの確立

以上のように設定したNCPサービスコマンドを利用してどのようにコネクションを確立するかを説明しよう。

この説明で、NCP間の会話に使用するコマンドはHOST-HOSTプロトコルで説明しているのをそれを参照されたい。

(1) 両方のプロセスが相手側のport-idを知っている場合

通常のコネクションの確立パターンであり、(図4-5)に示されるようにHOST X 中のプロセスUが、自分側のport-idがAeであり、確立するコネクションの相手側のport-idがBoであるのを知っている。また、HOST Y 中のプロセスVが自分側のport-idがBoであり相手側のport-idがAoであるのを知っていて、この2つのport-id Ao, Bo間でコネクションを確立する場合である。

この場合には、プロセスUおよびVがそれぞれ自分と相手のport-idをパラメータとしてOPENをかけることにより確立される。

OPENを受け取ったNCPは、相手側HOSTからRFCを受け取っていなければRFCを、受け取っていればCOKコマンドを返す。

この場合、RFCが両プロセスから発信されるタイミングが存在する。このときは発信用のport-idはCOKはださないで受信用のport-idを持っているNCPがCOKを発信する。

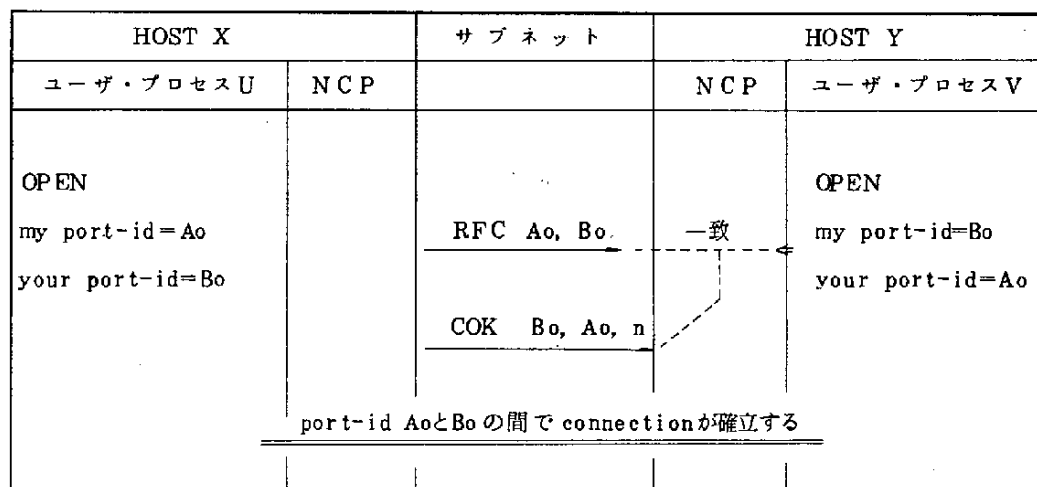


図4-5 コネクションの確立過程(1)

(2) 一方のプロセスだけが相手側の port-idを知っている場合

この確立手順は、あるプロセスが不特定のプロセスに対して確立する場合に使用することができる。すなわち、ユーティリティ・プロセスなどの高位のプロトコルを実現する場合にはこの手順によっておこなう。

(図4-6)においてHOST Y 中のプロセスVはコネクションを確立する相手を知らない。

プロセスVはLISTENマクロにより、NCP に対して自分側の port-id(図ではBo)を宣言したのち、イベント待ちマクロを発して待ち状態にしておく。他プロセスから(図ではプロセスU)コネクション要求があった場合に、相手側 port-id(図ではAo)を通知されて待ち状態が解かれる。このAo を使用して、コネクションの要求をするOPENを発信する。

この場合に、LISTEN により宣言した port-idと相手側 port-idとの間にコネクションを確立してもよいが、他の port-id と確立してもよい。

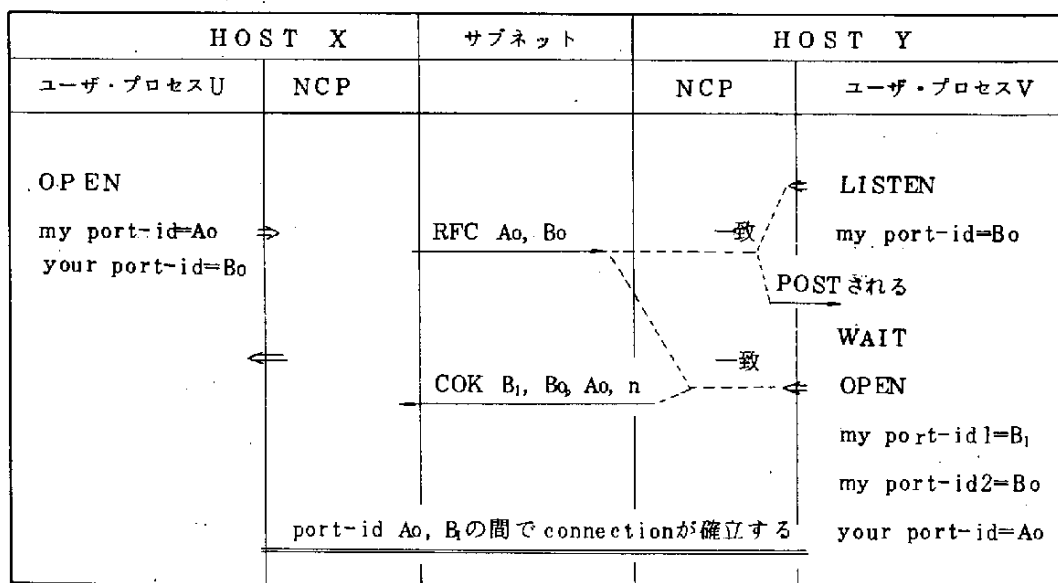


図4-6 コネクションの確立過程(2)

c プロセス間コミュニケーションの問題点

JIPNETにおいて、プロセス間コミュニケーションは基本的に重要なプロトコルであることは既に述べたとうりである。これの決定にあたって特に考慮したのはコネクションの切り換えをうまくおこなうことである。これはAJD, NCICを実現するために必要なものである。コネクションの切り換えとは、次のようなものをいう。

- (1) プロセスAとプロセスBのコネクションをBの要求によりプロセスAとプロセスCのコネクションにする。すなわち下図のようなものである。

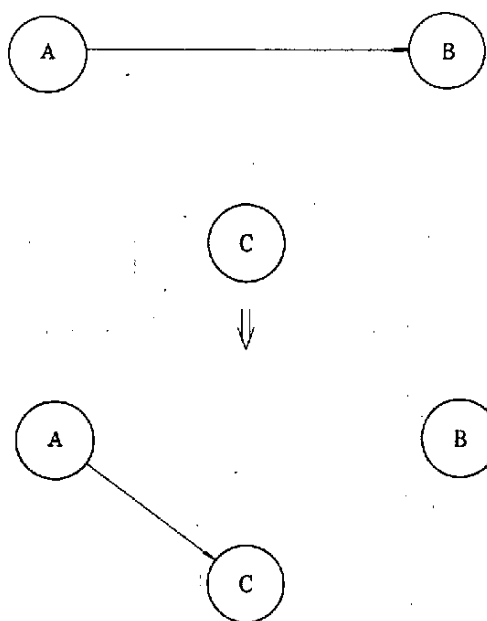


図 4-7 コネクションの切り換え(1)

- (2) プロセスAとB, プロセスBとCの2つのコネクションをBの要求によりプロセスAとCのコネクションにする。すなわち下図のようにすることである。

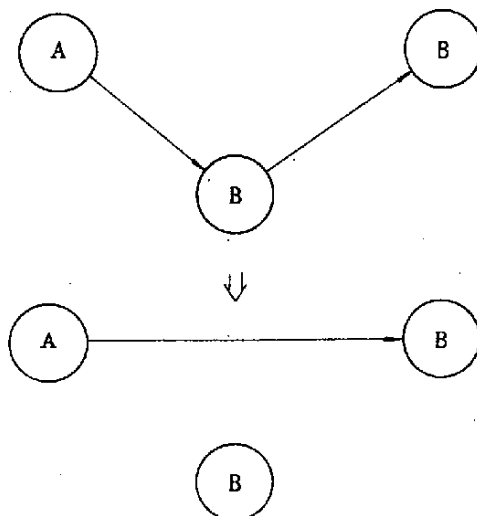


図 4-8 コネクションの切り換え(2)

コネクションの切り換えにおいて発生する問題点は次のものである。

データのフローコントロールとして応答性、伝送容量の向上のためにJ1PNETで採用した方法はHOST-HOSTプロトコルで述べている。この方法では、出力プロセスの動作が入力プロセスのそれよりも速ければ、入力プロセス側のNCPにデータが先取りされた形で蓄積されることになる。

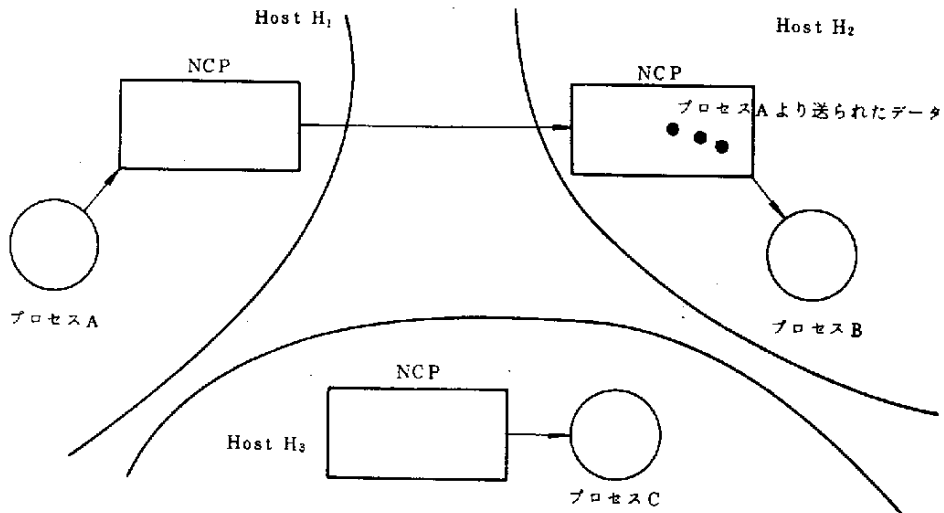


図4-9 データの先取り

このような状態のもとでコネクションを切り換えるとすると解決しておかなければならない問題がある。

すなわち(図4-9)において、プロセスBがプロセスAとCとのコネクションに切り換えるとすれば、前もってプロセスAから送り込まれたデータをHOST H₂のNCPからHOST H₃のNCPに送り込んでおいてから切り換えるかあるいはHOST H₁のNCPがHOST H₂のNCPには送られたが、プロセスBに受け取られなかったデータを切り換えられたコネクションを通じて再送するかのいずれかをおこなわなければならない。

まずHOST H₁のNCPが再送する場合を考えてみよう。

NCPが再送するためには送信データを保持しておかねばならないが、保持しておいたデータをいつの時点で捨てていいかということになる。このための一番簡単な方法はNCPがプロセスにデータを渡す時点でRECEIVEを発信することである。このようにすれば発信側NCPは対応するコネクションのRECEIVEを受け取った時点までデータを保持すればよいことになる。

RECEIVEをだすタイミングを上記のようにユーザプロセスにデータを渡すタイミング

でだと次のような問題点がある。

- ① サブネットのパイプがつまる。
- ② RECEIVEにバッファ・アロケートの情報をのせることが困難。
- ③ コネクション当りの最大伝送容量が減少する。

ここで1番問題になるのは①である。パイプとは、第3章で述べたように発信地IMPと目的地IMP間に存在するものであり、発信地、目的地が同一のコネクションに対して共用されるために①の現象がおこる。

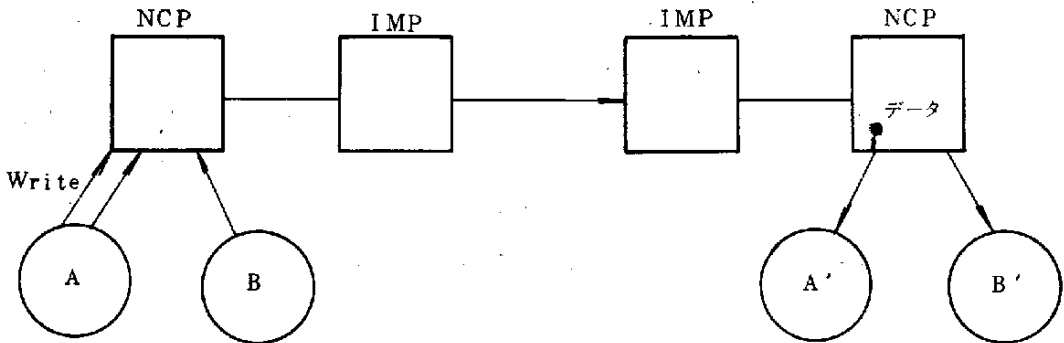


図 4-10 パイプの lock up

(図4-10)に示した例では、プロセスAとA'、プロセスBとB'にコネクションが確立されており、これは多重パケットメッセージを伝送するものであるとする。ここでAがデータの発信をしたにもかかわらず、A'が長期間入力命令を発しないと、当該パイプには1個のメッセージしか流入できないのでB→B'という流れがとまってしまうことになる。これはパイプ当りのメッセージ多重度をいくら大きくしてもそれ以上にコネクションがそのパイプを利用すればパイプがつまることをさけることができない。

この理由によりRECEIVEをだすタイミングをユーザプロセスにデータを渡すタイミングでだすことはできない。

このため発信NCPが再送するとすれば、RECEIVEは目的地NCPが発信し、ユーザプロセスにデータを渡すときには発信地NCPの再送用に保持されているメッセージを捨ててもよいというコントロールメッセージを発することになる。

次にHOST H₂のNCPがHOST H₃のNCPの間にコネクションを確立して先取りしたデータをこのコネクションを通じて送信し、このデータが終ったら新たにプロセスAとプロセスCの間にコネクションを設定する方法を考えてみる。

これで問題となるのは、先取りデータがある場合と無い場合ではHOST H₂のNCPの動作が違ふことである。このためにH₂のNCPでは先取りメッセージを持っているかどうか調べ持っていないければその時点以降にそのコネクションを通してはいつてくるメッセージを

捨てて、捨てたメッセージはHOST H_1 のNCPがコネクションが切り換えられたのち、プロセスCに向けて再送する方法をとらなければならない。

このため、RECEIVEはNCPがデータをIMPから受けた時点で出す。これにはコネクションの切り換え待ちなどの理由により再送を要求するむねをのせることができるようにする必要がある。

JIPNETで採用したコネクションの切り換え手順は、後者の手法である。

d プロセス間コミュニケーションのコマンド体系

プロセス間コミュニケーションを実現するコマンド体系は種々考えられるが、JANETにおいては、それぞれのHOSTにおける通常のファイルをアクセスする手法をそのまま流用した。これはプロセス間コミュニケーションが通常の順編成ファイルと概念的に同じであるため新しいアクセス法を作り上げるよりユーザにとってなじみ易いことにあった。

また、このようなアクセス法をうまく利用すると順編成ファイルが自己ホストに存在するかあるいは他ホストに存在するかを全く意識することなく、同一プログラムでアクセスできるという利点もある。

ここで他HOSTのファイルをアクセスする高位プロトコルを検討してみよう。このプロトコルをRemote Data Access Support (RDAS)と呼ぶことにする。

RDASは、あるホストにおいて、そのホスト内にあるファイルと他のホストにあるファイルを全く区別することなく、同一アクセス法で入出力することをサポートする高位プロトコルである。

通常そのホストにあるファイルのアクセス法はほぼ次のとおりである。

```
OPEN      port

READ

      ⋮
      ⋮
READ

CROSE     port
```

このアクセス法を他HOSTのファイルにまで拡張するためには、ネットワークの基本プロトコルがプロセス間コミュニケーションを基準としているために次のような形にしなければならない。

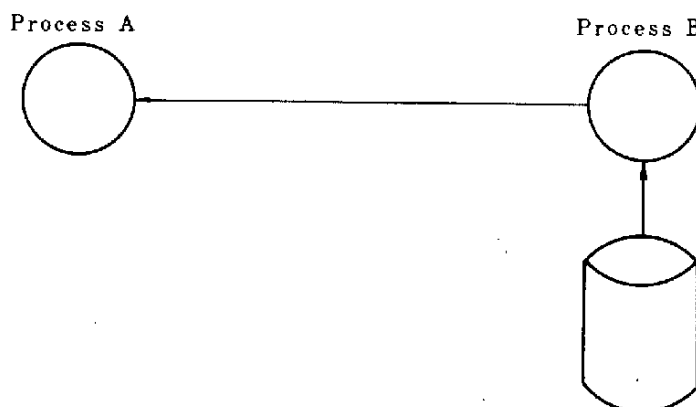


図 4-11 プロセス間コミュニケーションによるファイルの入力

ここでProcess AはProcess B とのコミュニケーションだけをおこない実際のファイルへのアクセスはProcess Bが他ホストの手順によっておこなう。そしてそのデータをProcess A, Bの両方のプログラムを作成しなければならない。このような手間をはぶき、ユーザはProcess Aだけを作成し、それが自ホストにあるファイルと全く同一アクセス法により、他ホストのファイルアクセスができるようにしたのがRDAS プロトコルである。

RDAS プロトコルは、Process B をシステム側で用意して次のような手順で処理をおこなう。

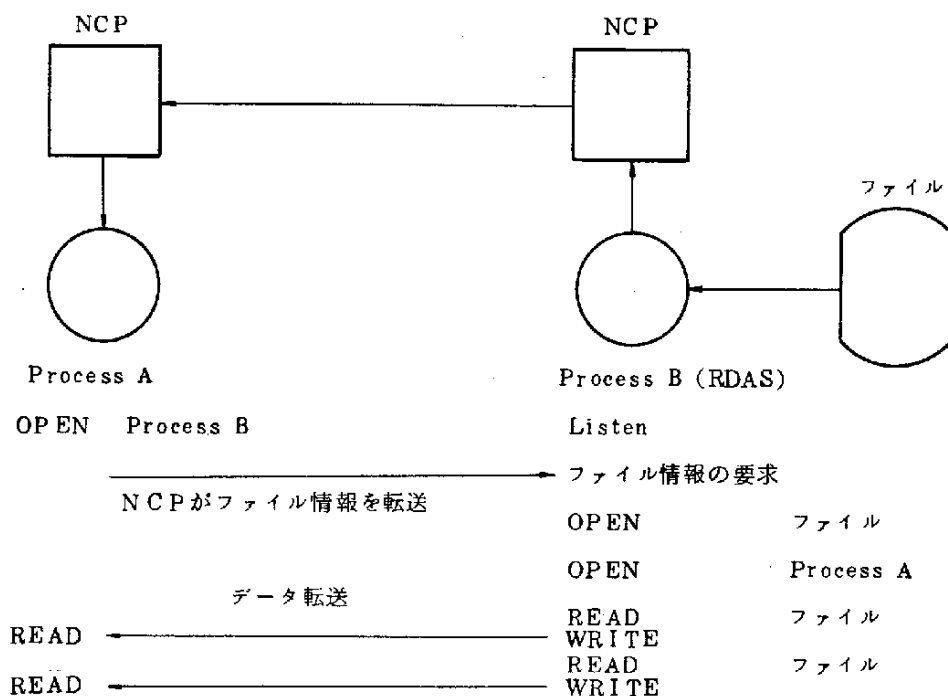


図 4-1.2 RDAS

このように、RDASによりユーザは、自ホスト内ファイルと他ホストにあるファイルを全く同一手順によりアクセスが可能になる。

B プロセスの発生

プロセス間コミュニケーションは、その両端となるプロセスが存在することを前提としてサービスするものである。

通常ユーザは一つのHOSTにおいてカードデックあるいはTSS端末によってプロセスを発生させる。ここでそのユーザがプロセス間コミュニケーションにより他ホストのプロセスと共同してジョブの実行をしようとしたとする。この場合、他ホストに自分の意図するプロセスを発生しなければならなくなる。このためにプロセスを発生するプロトコルが必要となる。

これをするためには、基本プロトコルの中にプロセスを発生する機能をサービスする方法と、プロセス間コミュニケーションを利用して高位のプロトコルとしてサービスする方法の2種類ある。

高位プロトコルによるサービス方法は、次のとおりである。特定のport-idがプロセスを発生するプロセスに割り当てられ、そのプロセスはそれぞれのHOSTに存在している。プロセスを発生しようとするユーザは、このport-idに対してコネクションを確立し、これを通じて発生すべきプロセス名、その他発生のために必要な情報を送信することによって目的を達することができる。

高位プロトコルによる方法は、プロセス間コミュニケーションだけを基本プロトコルとすればよいことになり、プロトコルが単純になるが次のような欠点を持っている。

- ① プロセス発生用のデータを送るコネクションを確立しなければならないためにオーバーヘッドが増加する。
- ② プロセスの状態監視をNCPにおいてするのが困難である。

ここで②の状態監視をする理由はネットワークにアクセスしようとするジョブが異常状態で終了したかどうか等を監視する必要があるからである。このような方法をとらないとデータ出力プロセスが異常終了した場合に対応するコネクションの一端となっている入力側プロセスは待ち状態からぬけられない。

この監視の機能を利用すれば、プロセスが異常終了した時点においてNCPがこれを知り、コネクションの解放などの処置によって上記のような状態をさけることができる。

以上の理由によりJIPNETにおいてはプロセス発生プロトコルを基本プロトコルとした。

これをNCPサービスコマンドとした場合の例を次に示す。

(1) プロセス発生 (LOADP) コマンド

一般形

LOADP <実行プログラム名>, <発生情報番地>, <プロセス情報番地>

パラメータ

＜実行プログラム名＞：実行形式プログラムの名前を示す。この名前は8文字以下でなければならない。

＜発生情報番地＞：＜発生情報＞の先頭番地をさす。

＜発生情報＞は＜ホスト番号＞、＜プロセス番号＞、＜ecb領域1＞、〔＜ecb領域2＞〕

構成される。ここで＜ホスト番号＞はプロセスの発生ホストを示す。＜プロセス番号＞はLOADP コマンドを発した時NCPが発生するプロセスにつける番号でありプロセスが発生した時NCPによりポストされDELP コマンドを使う場合これにより指定する。

＜ecb領域1＞は、プロセスの発生が成功したか不成功であったかをpostする場所である。

＜ecb領域2＞は、プロセスの実行が終了した時その完了情報がポストされる場所である。

＜プロセス情報番地＞：＜プロセス情報＞の先頭番地をさす。

＜プロセス情報＞は、＜ファイル個数＞、＜ジョブ情報＞、＜ファイル情報＞より構成される。

ジョブ情報および＜ファイル情報＞はそれぞれ（表4-1）、（表4-2）のとおりである。最初のファイル情報は実行プログラムが存在するファイルのそれではない。

表 4-1 ジョブ情報

項 目	長さ(バイト)	内 容
ユーザ番号	6	ネットワークユーザ番号
パスワード	6	ネットワーク・パスワード
コア容量	2	1024 バイト単位で指定
CPU使用時間	2	秒単位で指定
優先度	1	0：通常 1：優先
実行時パラメータ長	1	バイト数
実行時パラメータ	36	パラメータ、そのものを入れる

表 4-2 ファイル情報

	長さ (バイト)	
○ ファイル定義名	8	プログラムで定義した名前と実際のファイル名との つき合わせに使用。
○ ファイル名	16	ファイル名として使用。プロセス間コミュニケーション の場合には汎用データ管理とのインターフェー スの時のみ意味を持つ。
○ デバイスタイプ	1	Y00 ₈ : ディスクファイル Y01 ₈ : 磁気テープファイル Y02 ₈ ~ Y77 ₈ : 特殊デバイス Y=0 : プロセスの存在ホストにあるローカルフ ァイル =1 : プロセス間コミュニケーション
○ ファイルタイプ	1	01 ₈ : 順編成ファイル 02 ₈ : 分割形順編成ファイル 04 ₈ : 直接編成ファイル 10 ₈ : 順インデックスファイル
○ ファイル存在場所	6	
○ ファイル属性		
ファイルグループ	1	0 1 : システムファイル 0 2 : パブリックファイル 0 4 : コモンファイル 1 0 : ユーザファイル
○ ファイル層	1	01 ₈ : ワークファイル 02 ₈ : 新ファイル作成 04 ₈ : 旧ファイル参照
ジョブステップ終了 後の処置	1	01 ₈ : 保 存 } 新, 旧ファイルのとき有効 02 ₈ : 消 去
○ パスワード	6	
○ その他		
⊗ プロセス間コミュニ ケーション		
my port-id	5	
your port-id	5	

(2) プロセス消去 (DELP) コマンド

一般形

DELP <プロセス番号>, <ecb領域>

パラメータ

<プロセス番号>: 消去したいプロセスの番号がはいっている番地を示す。この番号は LOADP コマンドにより付けられたものである。

<ecb領域>: このコマンドの完了情報がセットされる。

機能

LOADP コマンドにより発生したプロセスを消去するのにもちいる。従属プロセスの場合にこのコマンドにより消去された時は LOADP コマンドの<ecb領域 2>には POST されない。

C プロセス間の同期

同一のコンピュータ内であってもマルチ・タスクプログラムにおいては、タスク間で同期をとりながら処理を進めてゆく。複数コンピュータにまたがるジョブにおいても同様のことがいえる。

プロセス間の同期は、情報のポスト、情報のポスト待ちという形をとるので、プロセス間コミュニケーションにおいて2つのプロセス間でコネクションを確立し、この間でREAD/WRITEの対応によって同期をとることもできる。JIPNETにおいてこれを基本プロトコルにいたしたのは、プロセス間コミュニケーションで代用する場合にはコネクションの確立をおこなったのちにしかREAD/WRITEにより同期をとらなければならないが、このような情報は非常に短くより単純な方法をサービスしてもNCPにとって負担にならないと考えたからである。

これをサービスするNCPサービスコマンドはPOSTとWaitであるがWaitはすでにプロセス間コミュニケーションにおいて説明したのでPOSTだけをのせた。

(1) イベント通知 (POST) コマンド

一般形

POST <ecb-id>, <イベント情報>

パラメータ

<ecb-id>: 通知をする相手を指す。

<イベント情報>: 32ビットの情報

機能

他プロセスと同期などをとるためにイベント情報を転送する。

4.1.4 HOST-HOSTプロトコル

HOST-HOSTプロトコルはHOST間のデータの転送、NCP間の会話方法を規定するものである。

HOST間のデータの転送はコネクションを通じておこなわれ、次節で述べるHOST-IMPプロトコルに従ってサブネットに渡された形そのままに相手HOSTに手渡される。

A フローコントロール

コネクションを通してプロセス間でデータの授受をおこなう場合にNCPにおいてはデータの流れをコントロールしなければならない。

データの送信に関するフローコントロールは、サブネットにおいてそのような機構がありそこでおこなっている。NCPにおけるフローコントロールは、NCPのバッファの利用状況の管理という形でおこなわれる。

プロセス間コミュニケーションにおいて、NCPは受信側プロセスに対して必要となるデータをできるだけ速やかに渡すことが役目となる。

しかしながら発信側プロセスがそれと同期をとってデータ送信をしているという保障がないために、両者の間の同期をうまくとり、高速な伝送を実現しなければならない。

フローコントロールをする手法として次のものがある。

- ① ユーザ・プロセスがデータの受信要求を出したとき、この要求を送信側NCPに知らせ、対応する発信データを送る。
 - ㉑ コネクションの両端のプロセスは、データの伝送が完了するまで待ち状態となる。
 - ㉒ コネクションの両端のプロセスは、データ伝送の命令を出すと伝送が終了しないうちに制御がもどってくるので伝送が終了したかどうかをチェックしなければならない。
- ② 送信側プロセスで送信要求がでた時点でデータを受信側NCPに転送する。
 - ㉓ データ伝送が完了するまで両端のプロセスは待ち状態となる。
 - ㉔ データの受信側NCPにおいて先取りしたデータが一定の個数を越えた場合、送信側NCPに転送中止を出し、これによって送信側プロセスを待ち状態にする。...

①の手法は基本的に、受信要求がきた時点で目的地に向ってデータを送信する方法をとるために(図4-13)のように送信要求、データ、確認というメッセージが往来していることになる。

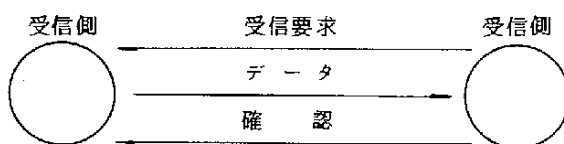


図4-13 データの授受

受信要求、確認の伝送に t_r 、データの伝送に t_d の時間がかかるものとすれば、単位時間に伝送できるデータ量は

④においては、メッセージ長 / $(2t_r + t_d)$ となる。⑤においては、受信要求が十分にでていれば、サブネットが許す量まで可能である。

この方法では、受信要求がでてからデータが流れるために NCP において中間的なバッファを用意する必要がないという長所を持つが 1 つのコネクションが大量のデータを転送要求した場合には、それに見合うだけの多重バッファリングをユーザがしなければならない。また、受信要求のコントロール・メッセージが増加すること、受信要求がでてから受信までに $(t_r + t_d)$ の時間がかかるという短所を持っている。

②の手法は、データが発生した時点で目的地に対して送信する方法をとっている。

この手法における最大伝送データ量は、受信要求の時間がないため

④においては、メッセージ長 / $(t_r + t_d)$ となり、⑤においては、コネクションにおけるメッセージ多重個数を n とすれば④の量の n 倍だけ伝送できることになる。

この方法では、ユーザは受信側において多重バッファを考慮することなく、高速伝送を利用できる長所を持つ。しかしながら NCP において必ずバッファを用意しておかなければならないという短所を持っている。

ここで t_r , t_d がどのくらいの値になるかということは非常に興味があるが、残念ながら現時点ではこれを予測するデータを持っていない。このため参考のために ARPANET において予測したデータをみってみる。発信地から目的地に行くまでに最少何個の IMP を経由しなければならないかによって変化するが、これが 1 の時と 5 個の時は、 t_r はそれぞれ 5, 50 msec であり、 t_d は 8 パケットすなわち 8000 ビットメッセージの場合、それぞれ 195, 320 msec である。この直を①の④、②の④にあてはめてみるとコネクション当りの最大伝送量はそれぞれ 40Kbit/sec ~ 19Kbit/sec, 40Kbit/sec ~ 21Kbit/sec となる。

JANET においては、ネットワーク自体は小さいために発信地—目的地間にただ 1 個のコネクションしか設定されていない確率はかなり高いこと、ファイル転送、インタラクティブ・グラフィックなどのように大容量のデータが対象となる場合がかなりあることからたとえ 1 つだけのコネクションに対しても最大の伝送量を出すことができるように設計すべきである。

このため、1 コネクション当り最大 100Kbit/sec 以上だせるようにしなければならないために①の⑤あるいは②の⑤のいずれかを採用することになる。そして、ユーザの負担を軽くするために基本的には②の⑤を採用した。

発信側プロセスは、WRITE 命令により port, ecb およびバッファなどを指定する。出力の完了は wait 命令によって ecb およびバッファなどを指定する。入力完了は wait 命令によって ecb にポストされるのを待つ。

このようなユーザコマンドを効果的に動作させるために、NCPは次のようになっていればよい。

コネクションが確立する時点でNCPはバッファプールよりバッファを1個取り出して基底バッファとする。基底バッファは、そのコネクションが解放されるまで保持しておくものである。

データが伝送されてくると、それを基底バッファに入れる。次のデータが伝送されてきた時点で基底バッファが使用中ならば、バッファ・プールからバッファを取り出して、そのコネクションに対するデータを先取りしておく。

バッファ・プールからバッファを取り出した場合には、ユーザ・プロセスがREAD命令を出したならばデータを渡した後、必ずバッファ・プールに返さなければならない。

この方法において、発信側プロセスのデータ伝送よりも、受信側プロセスの要求の時間間隔が長いと、受信側NCPにおいてそのコネクションのバッファ占有量が多くなり、他のプロセスへのサービスに支障をきたすことになるので、バッファプールに残っているバッファ個数が一定値以下になった時に、一定個数以上のバッファを占有しているコネクションに対してデータの伝送を一時停止させる。このコネクションに対しては基底バッファのデータがユーザ・プロセスに渡された時点で送信再開をさせる。

このようにすれば、コネクションが少ない時にはバッファを多く利用できるために、1コネクション当りの同時伝送度が高くなり大容量のデータを高速に伝送できるばかりでなく、混雑時にも少なくとも1個のメッセージの受け取りを保障されることになる。

B コントロール・コマンド

コントロール・コマンドは、各ホスト中に存在するNCP間で会話し、ネットワークジョブに対してサービスをするために使用されるものであって、コントロール・コネクションを通じて通常メッセージ(これをコマンドという)としてNCPに転送される。

NCPにおいてはこれらのコマンドを発信、受信、解読、実行できなければならない。

コントロール・コネクションは、NCPが初期設定を完了したときに確立されているものである。

コマンドは(図4-14)に示すように、8ビットのコマンドコードと568ビット以下のコマンドパラメータより構成されている。

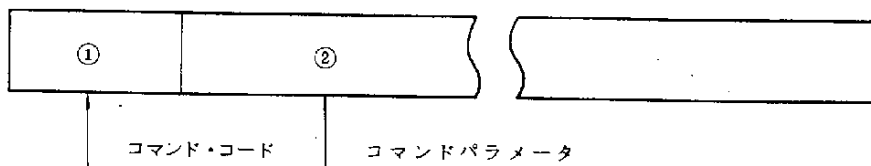


図4-14 コントロールコマンドの形式

① コマンド・コード (8 ビット)

HOST - HOST プロトコルで何のコマンドであるかを示す。

② コマンド・パラメータ (0 ~ 568 ビット)

各コマンドによって決められたパラメータ

コントロールコマンドは、次の 4 種類の機能をはたすものに大別できる。

① プロセス間コミュニケーション

② プロセスの起動

③ イベント情報の通知

④ ファイル情報の通知

以下にその内容を述べる。

(1) コネクションの確立

① コネクション確立要求 (Request for Connection) コマンド

一般形

RFC <my port-id>, <your port-id>, <ecb-id> ,

$\left\{ \begin{array}{l} \text{<コネクション番号>} \\ \text{<メッセージ最大長>} \end{array} \right\}, \text{ <pass word>}$

コマンドコード = 0 1

パラメータ

<my port-id> 4 0 ビット : 確立を要求するコネクションの一端を示す <port-id> である。

<your port-id> 4 0 ビット : RFC コマンドの受信側に存在する <port-id> である。

<ecb-id> 4 0 ビット : RFC コマンドの結果がポストされる。

<コネクション番号> 8 ビット : <my port-id> が受信 port の場合に NCP が割り当てる。コネクションの確立後は、この番号により、プロセス間のデータ授受を行なう。

<メッセージ最大長> 8 ビット : <my port-id> が送信 port の場合に、このコネクションを通る最大メッセージ長を 144 ビットの倍数であたえる。

この項に (その倍数 - 1) を入れる。

このため、メッセージの最大長は 36864 ビット (4008 バイト) 以下となる。

<Pass word> 4 8 ビット : ユーザのパスワードを指定する。

機 能

プロセス間コミュニケーションを実現するコネクションの確立をはかる。

このコマンドに対する応答は、C O KあるいはC L Sコマンドである。

② コネクション確立応答 (Connection OK) コマンド

一般形

C O K <my port-id 1>, <your port-id>, $\left\{ \begin{array}{l} \text{<コネクション番号>} \\ \text{<メッセージ最大長>} \end{array} \right\}$,
[<my port-id 2>]

コマンドコード=02

パラメータ

<my port-id 1> 40ビット：確立されるコネクションの一端となる自分側の
<port-id>である。

<your port-id> 40ビット：確立されるコネクションの他端となる相手側の
<port-id>である。

<コネクション番号> 8ビット：<my port-id 1>が受信portの場合にNCPが
割り当てる。プロセス間コミュニケーションを実現するデータの授受は、この番
号によっておこなわれる。

<メッセージ最大長> 8ビット：<my port-id 1>が送信portの場合に、このコ
ネクションを通る最大メッセージ長を144ビットの倍数であたえる。この項に
（その倍数-1）を入れる。

<my port-id 2>：RFCで指定してきた<your port-id>とC O Kコマンドで
指定する<my port-id 1>が違っている場合に、RFCで指定してきた<your
port-id>を入れる。

機 能

RFCコマンドに対してコネクションの確立ができたことを応答するコマンドである。

<my port-id 2>を指定するのは、ユーザが相手プロセスからの<port-id>の通
知により、新たに自分の<port-id>を変えてコネクションを確立するとき使用する。
る。

③ コネクション解放 (Close) コマンド

一般形

C L S <my port-id>, <your port-id>, <last message no>

コマンドコード=03

パラメータ

<my port-id> 40ビット：closeしたいコネクションの自分側の一端を指す。

<your port-id> 40ビット：closeしたいコネクションの相手側の一端を指す。

<last message no> 32ビット：プロセスに（あるいはプロセスから）渡された

最終メッセージの番号を入れる。

機 能

コネクションを解放する。このコマンドは、確立されていないコネクションの確立失敗にも利用される。

(注)

送信側の<port-id>を持つNCPは、最終メッセージに対応するRFNを受け取ってからこのコマンドを発信しなければならない。

④ メッセージ送信一時停止 (Cease) コマンド

一般形

CEASE <コネクション名>

コマンドコード=04

パラメータ

<コネクション名> 16ビット: メッセージの送信を一時停止するコネクションを示す。

機 能

このコマンドは、受信側プロセスのあるNCPが発信するものであり、このコマンドの対象となったコネクションは、RESUMEコマンドが発信されるまでメッセージの転送はおこなわない。

⑤ メッセージ送信再開 (RESUME) コマンド

一般形

RESUME <コネクション名>

コマンドコード=05

パラメータ

<コネクション名> 16ビット: メッセージの送信を再開するコネクションを示す。

機 能

メッセージ送信の一時停止をされたコネクションに対して、最後に受け取られたメッセージの次の番号のメッセージから送信を再開する。

⑥ コネクション切り換え待ち (Wait for Reconnection) コマンド

一般形

WFR <コネクション名>, <メッセージ番号>

コマンドコード=06

パラメータ

<コネクション名> 16ビット: 切り替えをするコネクションを指定する。

<メッセージ番号> 3 2ビット：プロセスに（あるいはプロセスから）渡された最終メッセージの番号を入れる。

機 能

コネクションの切り替えをおこないたい時には、このコマンドを前もって発信しなければならない。このコマンドをNCPが受け付けると当該コネクションは一時停止状態におかれ、COKコマンドあるいはRESUMEコマンドにより、この状態がとかれる。

（注）

発信側 port を持つNCPがこのコマンドを発信するためにはそのコネクションに対して最後に発信したメッセージに対するRFNMが返ってきてからでなければならない。

発信側と受信側が同時にWFRコマンドを発信した時には、受信側のNCPが出したWFRコマンドが有効となり発信側のそれは無効となる。

⑦ コネクション切り換え (Exchange Connection) コマンド

一般形

ECON <your port-id 1>, <your port-id 2>, <ecb-id>

コマンドコード = 07

パラメータ

<your port-id 1>：相手が受信 port になっている port-id を示す。

<your port-id 2>：相手が送信 port になっている port-id を示す。

<ecb-id>：このコマンドの結果がポストされる ecb を示す。

機 能

あるプロセスから、自分のプロセスを経由して別のプロセスにつながっているような2つのコネクションを、つなぎ合わせて1つのコネクションを作りあげ、自分のプロセスは、このコネクション関係から切り放される。

このコマンドは受信側プロセスのNCPに向って発信される。

このコマンドを発信する前に送信側プロセスはWFRコマンドによって一時停止状態においておかなければならない。

(2) 情報の通知

① POST コマンド

一般形

POST <ecb-id>, <通知情報>

コマンドコード = 10

パラメータ

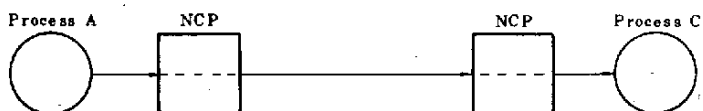
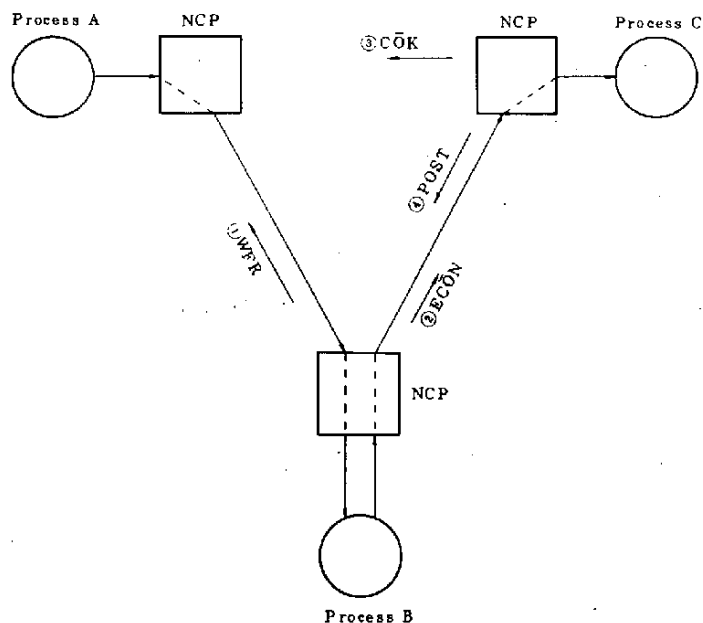


図 4-15 コネクション切り換え

< ecb-id > 40 ビット: POSTすべき ecb を示す。

< 通知情報 > 32 ビット: POSTすべき情報を示す。

機 能

情報を通知する。ユーザプロセスが待ち状態にあれば待ちが解かれる要因となる。

(3) プロセスの制御

① プロセス発生, 実行 (Load and Execute) コマンド

一般形

LOADE < プロセス名 >, < プロセス情報ヘッダー >, < ecb-id 1 > ,
[< ecb-id 2 >]

コマンドコード = 20

パラメータ

<プロセス名> 56ビット：発信ホストでつけたプロセス名で始めの24ビットはユーザ名である。

<プロセス情報ヘッダー>：<実行プログラム名>，<ジョブ情報>，<ファイル個数>より成り，それぞれ64ビット，432ビット，16ビットの長さである。

<ファイル個数>は，その個数だけCNT情報が追加される。

<ジョブ情報>は（表4-1）を参照のこと。

<ecb-id 1> 40ビット：プロセスのload 結果がPOSTされるecbを示す。

<ecb-id 2> 40ビット：プロセスの実行結果がPOSTされるecbを示す。

機能

プロセスを発生させ実行させる。

② プロセス消去 (Delete Process) コマンド

一般形

DELP <プロセス名>，<ecb-id>

コマンドコード= 21

パラメータ

<プロセス名> 56ビット：LOADPコマンドにより指定したプロセス。

<ecb-id> 40ビット：このコマンドの実行結果がPOSTされる。

機能

LOADP で発生したプロセスを消去する。

(4) ファイル情報転送

① ファイル情報要求コマンド

一般形

RFF <your port-id>

コマンドコード= 50

パラメータ

<your port-id> 40ビット：要求するファイル情報を持つPORTを指定する。

機能

ファイル情報を送るように要求する。これを受け取ったNCPはportとport 定議のコントロールカードより，標準FDR情報を作成して送らねばならない。

② 追加情報

一般形

CNT <情報コード>，<情報>

コマンドコード= 40

パラメータ

<情報コード> 8ビット：追加情報が何を意味するかを示す。

ファイル情報の時は01であり、標準FDR情報のときは02である。

<情報> 640ビット以下：情報コードごとに自由に使用する。

機能

コマンドの追加情報を送信するために利用する。

現時点では、ファイル情報を送るのに使用している。

表 4-3 標準FDR情報

項 目	長さ(バイト)	
○ ファイル名	24	
○ デバイスタイプ	1	00 ₈ : ディスクファイル 01 ₈ : 磁気テープファイル 02 ₈ ~ 77 : 特殊デバイス
○ ファイル存在場所	6	
○ ファイル属性		
ファイル	1	01 ₈ : Temporary ファイル 02 ₈ : 新ファイル作成 04 ₈ : 旧ファイル参照
ジョブステップ終了 後の処理	1	01 ₈ : 保 存 02 ₈ : 消 去 } 新, 旧ファイルのとき有効
ブロック長	2	バイトで示す。
レコード長	2	バイトで示す。

4.1.5 HOST-IMPプロトコル

HOST-IMPプロトコルは、高い信頼性と高速伝送の役割をはたすサブネットのノードとなるIMPとHOSTとの間でのデータの受け渡し、障害時の処理、初期会話などを取り決めたものである。

HOSTとIMPの関係はここで決められたものだけであって、それ以外何の関係も持たないで動作しなければならない。

A チャンネル・アダプターのコマンド

IMPのチャンネルとHOST Computerのチャンネルを結合するチャンネル・アダプターには次のコマンドを解説、実行する機能を持っている。

これらのコマンドのコマンドコード、表現形式はそれぞれのHOSTおよびIMPにおいて違っているので詳しくは「

① READ DATA (RDと略称する)

相手側の発したWDI, WDコマンドによるデータを読み取る。

相手側の発したコマンドがWDIあるいはWDのいずれであるかを知ることができる。

② WRITE DATA with INTERRUPT(WDI と略称する)

相手側CPUに対して割り込みをおこなう。RDコマンドが相手側から発せられるとデータを転送する。

WDIを発する以前に相手側がRDを発していた場合には割り込みはおこさない。

③ WRITE DATA (WDと略称する)

RDコマンドが相手側で発せられるとデータを転送する。このコマンドは相手CPUに割り込みをおこさない。

④ NO OPERATION (NOPと略称する)

何もおこなわない。

⑤ SENSE

アダプターの状態を聞くコマンドである。

⑥ LOCK PSW (LKと略称する)

自分側アダプターのプログラム・スイッチをlockする。

プログラム・スイッチは、チャンネル・アダプターには2個あり、この2個がともにunlock状態のときデータを転送することができる。いずれかがlockされているときには、データ転送をしないで異常終了する。

SENSE コマンド聞くことのできるプログラム・スイッチは相手側のそれである。

電源が投入された初期状態では、このスイッチはlock状態となっている。

⑦ UNLOCK PSW (ULKと略称する)

自分側アダプターのプログラム・スイッチをunlockする。

B データ・フォーマット

HOST - IMP間において授受されるデータは、制御情報と一般データの2種類ある。

制御情報は(図4-17)に示すように32ビットで構成され、WDIコマンドにより転送される。

一般データは初期設定によって決められたセグメントサイズ以下の不定長でありWDコマンドにより転送される。

ユーザ・プロセスが送信したデータは、メッセージごとに制御情報とメッセージを分割して1個以上のセグメントを作りサブネットに投入される。

目的地IMPにおいては到着したメッセージを送信側NCPがサブネットに投入した時と全く同じ様式になおして受信側NCPに渡し、それを送信側プロセスが渡したメッセージと全く同じ様式にして受信側プロセスに渡す。

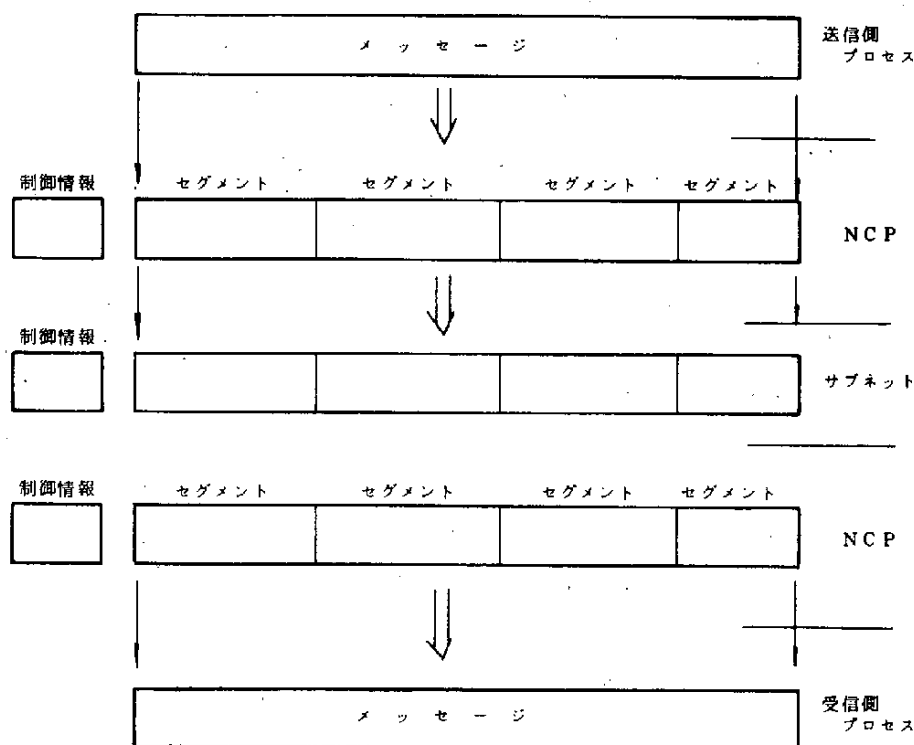


図4-16 メッセージの様式

以下に制御情報の形式を示す。

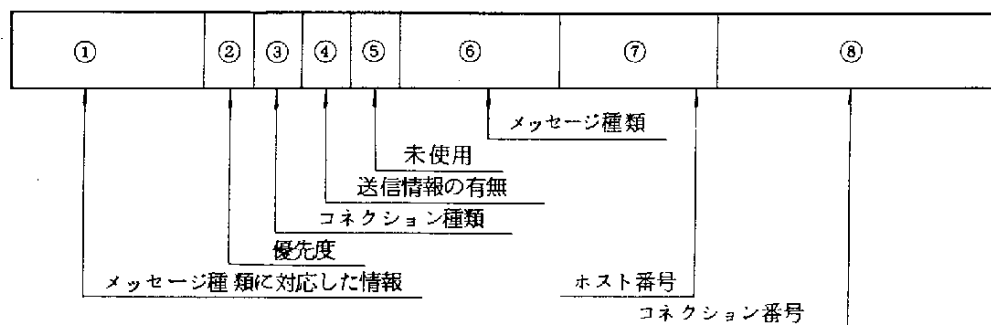


図4-17 ホスト-IMP間の制御情報

- ① メッセージ種類に対応した情報（８ビット）
 - ④のメッセージ種類により使用方が異なる。（表４－６）参照のこと。
- ② 優先度（１ビット）

１のとき優先度指定あり。

優先度指定されたデータはサブネットワークにおいては優先的に伝送される。

この指定は、１メッセージが１パケットより構成されるもの以外には使用できない。
- ③ コネクション種類（１ビット）

１のときこのコネクションは多重パケットメッセージを送り出すことを示す。
- ④ 送信情報の有無（１ビット）

この情報を送ったあとに、次に相手側に送る情報の有無を示す。１のとき送信情報があるのを示す。
- ⑤ 未使用（１ビット）

常に０とする。
- ⑥ メッセージ種類（４ビット）

制御情報が何を意味するかを示す。（表４－４），（表４－５）を参照のこと。
- ⑦ ホスト番号（８ビット）

ホスト→IMP制御情報の場合には、目的地ホスト番号を示す。

IMP→ホスト制御情報の場合には、発信地ホスト番号を示す。
- ⑧ コネクション番号（８ビット）

プロセス間コミュニケーションにおいてNCPがプロセス中のportを識別するための番号。

表４－４ IMP→ホスト制御情報

メッセージ種類	意味
0	通常メッセージ送信要求。（Regular message） 他ホストより、そのホスト宛の一般データを転送する要求があった時、プレフィックスとして使用される。
1	セグメント長変更指令 サブネットワークの初期設定の結果、セグメントサイズが標準値でない場合にIMP↔ホストの初期設定時に発信される。
2	初期設定指令 サブネットワークの初期設定の結果、セグメントサイズが標準値の場合

- に発信される。
- 3 データ受信完了 (RECEIVE)
他ホストに発信したメッセージが正常に送られたことをIMPが確認したとき発信する。
- 4 接続中止要求
IMPがネットワーク・サービスを中止したい時にホストに対して発信する。
- 5 切断完了
ホストがネットワークサービスの中止要求を発信したのに対する応答。
- 6 無用情報 (NOP)
- 7 ホストへのテストデータ
ネットワークの伝送能力を測定するために使用する。これにより伝送されたメッセージは目的地ホストにおいて捨てられる。
メッセージ番号 = 0 と同じようにプレフィックスとして使用される。
- 8 未使用
- 9 稼働テスト
ホストが稼働中かどうかテストする。稼働中であれば、稼働応答が返ってくる。
- 10 交信一時停止
- 11 交信再開
交信一時停止をIMPが発信した時これを再開するために使用する。
- 12 複数セグメント受け入れ準備完了
ホスト側から複数セグメントの転送要求があった場合に受け入れ準備ができたなら、これを発信する。これによりホスト側から第1セグメントから最後のセグメントまで送られてくる。
- 13 メッセージ再要求
サブネットの障害などで、パケットが消えた場合にHOSTに対してメッセージの要求をする。
- 14～15 未使用

表 4-5 ホスト→IMP 制御情報

メッセージ種類	意 味
0	通常メッセージ送信要求。(Regular message) 他ホストに一般データを送信したい場合にプレフィックスとして使用される。
1	初期設定完了表示。(Ready)
2	未使用
3	データ受信完了 (RECEIVE) 目的地ホストがメッセージを受け取った時に発信する。
4	接続中止要求 ホストがネットワークとの結合を切断したい時に発信する。
5	切断完了 IMP がネットワーク・サービスの中止要求を受信したのに対しての応答。
6	無用情報 (NOP)
7	ホストへのテストデータ ネットワークの伝送能力を測定するために使用する。これにより伝送されたメッセージは目的地ホストにおいて捨てられる。 メッセージ番号 = 0 と同じようにプレフィックスとして使用される。
8	IMP へのテストデータ サブネットワークの伝送能力を測定するために使用する。これにより伝送されたメッセージは、目的地 IMP において捨てられる。 メッセージ番号 = 0 と同じようにプレフィックスとして使用される。
9	稼働応答 IMP が動作確認をしてきた時に応答する。
10	交信一時停止
11	交信再開 交信一時停止をホストが発信した時、これを再開するために使用する。
12~15	未使用

表 4-6 制御情報のメッセージ種類に対応した情報部分の用途

メッセージ種類	メッセージ種類に対応した情報の用途
0	送信メッセージのセグメント数 (1 ~)。
1	初期設定するセグメント長をユニット数で指定する (1 ユニット = 144 ビット)。 IMP → ホスト制御情報のときのみ使用
2	未使用
3	RECEIVE の状態を数字で入れる。 = 1 のとき 正常受信 = 2 のとき バッファ余裕なし, discard = 3 のとき コネクション切り換え中 discard
4 ~ 6	未使用
7	送信メッセージのセグメント数 (1 ~)。
8	送信メッセージのセグメント数 (1 ~)。 ホスト → IMP 制御情報のときのみ使用。
9 ~ 15	未使用

C HOST - IMP のデータ転送制御

HOST, IMP のそれぞれが相手側にデータを転送されるタイミングは非同期に起こるが、アダプタは半二重であり同時に一方向だけにデータを流すことができるだけである。このために、両側からアダプタ利用権の取り合いがおこり、これをうまく制御してやらなければならない。この方報として次のものを考えることができる。

- ① アダプタの利用権を時間帯を分けて、時間帯ごとに HOST, IMP のいずれか一方が利用する。
- ② 初期状態として一方に利用権をあたえ、データの転送により相手側に利用権が移行する。
- ③ データの転送により一時的に利用権は相手側に移行するが、相手側が利用する必要がなければ再び自分側に利用権が返る。

アダプタの利用権を取り合いを制御する目的は、両側からのデータ転送をうまく制御するためであるが、実際に発生する事象というのはハードウェア、ソフトウェア (OS の内部) におけるデータ転送コマンドの衝突、データ転送コマンド発信のタイミング調整という形である。

①の方法は HOST, IMP の両側でタイマーが正確に働いていれば、コマンドの衝突は全くないので有効であるといえる。しかしながら、この時間帯の設定に問題がある。すなわち 1000

ビットのメッセージの転送に必要なアダプタ利用時間はたかだか数m secであるが、この程度の時間ごとにタイマ割り込みをかけて利用権の移行をさせるのはオーバーヘッドが非常に大きいばかりでなく、HOSTのOSによっては全く不可能である。たとえば、HITAC 8450のタイマ割り込みは秒単位であり、しかも±20m secの誤差があり、これを利用するのは不可能である。

②の方法は、一方からのデータの転送後に必ず他方からのデータの転送があり場合には有効な方法であるが、今回はこのような形になっていないのでこれを採用することはできない。

③の方法は、②の改良形といえるものである。ここで問題になるのは初期状態をどうするか、利用する必要のないのをどのような形で知らせるかということである。

初期状態どちらかにコマンド発信権を持たせるのは不都合がおこるので両方に持たせておきコマンドの衝突が起った場合だけどちらかが優先するようにする。

利用権の移行の制御の方法としては次のものが考えられる。

- ④ データを受け取ったならば必ずACKを返し、この情報中に相手側に利用権を渡すかどうかを指定する。
- ⑤ 転送データの中に利用権を自分側に要求する情報を入れておく。相手側が同様に要求したならば交互にデータ利用権に移るようにする。
- ⑥ データ転送後、一定時間待って相手側からデータがこなかったならば自分側に利用権があるものとする。

このうち、⑥はタイマーとの関係で利用できない。また④はACKを必ず返す手順であれば有効利用できるが、今回のシステムの場合はACKを返す必要がないことからACKを追加すればそれだけ情報の転送が増える。このため、⑤を採用することにした。⑤は④にくらべて、衝突回数は増えるおそれがあるがACKを追加するよりは有効であると考えられるからである。

次に、実際HOST-IMPでおこなっているデータ転送のタイミング調整について述べる。

同期をとって転送されるデータがあるのでまず最初にこれについて述べる。

- ① HOSTは、一般データを正常に受信したならば、ただちにRECEIVEを返送する。
- ② HOSTは、IMP側から多重パケットメッセージの転送要求があったならば、ただちにそのパケットを連続して送らなければならない。
- ③ 単一パケットメッセージおよびIMPからHOSTに送る多重パケットメッセージは、制御情報につづいて転送される。

上記の条件を満たしてかつコマンドの衝突をできるだけ起こさないように下記の手順を使用した。

- ① 現在送信しようとする制御情報のあとに送るべき制御情報の有無を示すビットをもうける。
- ② データ送信後に相手側から次の送信情報ありの状態になっていれば受信準備をする。

③ コマンドの衝突がおこった場合には、優先された方はその転送がすんでから、受信準備をしなければならない。

④ RECEIVE制御情報の発信の場合には、次の情報をただちに発信してよい。……ただし、コマンド衝突の場合にはコマンド衝突のあと処理を優先させる。

すなわち、JIPNET で採用した手順は、メッセージの往来が疎な時には何らのメリットもないが非常に多量の往来がある場合に効果が表われる。これはメッセージの往来が疎な場合には、次に相手側に送るべきメッセージが無いためである。一万多重のメッセージの往来がある場合には、ある一時点をとらえたとき、次に送信するメッセージが有りこの場合にはうまく制御が働く。

(図4-18)はその例を示している。いま、IMPからHOSTに対して①～④までのメッセージがあり、そのうち②、④はRECEIVEである。またHOSTからIMPに対して1'～4'までのメッセージを送信するとして。

こと1と1'が衝突したとする(衝突がおこらなくても1がHOST側に送られれば同じ手順をとることになる)と、図中の順序で以後メッセージは衝突なく転送されることになる。

すなわち、一度衝突が起るかあるいはたがいに相手から自分側に対するデータの転送があることことを知らせあうことによって、その時点で待ち行列にあるものまたは交互に転送しあっている最中に来たものについては衝突およびデータ交換の制御をうまくおこなうことができる。

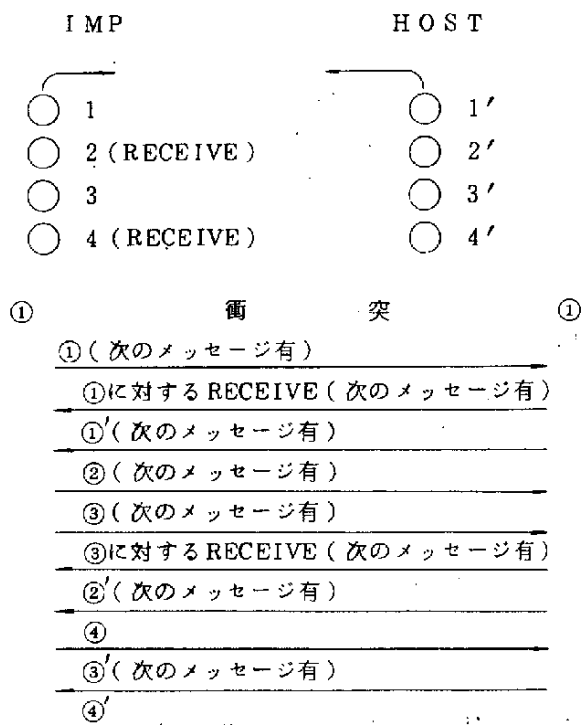


図4-18 データ転送の制御

D 初期状態処理手順

ネットワークのサービス開始時には、自分側アダプターの電源が on の状態になっているのを確認したのを自分側 P S W を unlock し相手側アダプターの状態をさく。

相手側が inoperable あるいは P S W が lock のいずれかの状態であれば相手側はまだ動作可能状態にないものとしてその状態を覚えておき、相手側からの連絡を待つ。

相手側が inoperable, P S W のいずれでもなければ、相手側に対して初期連絡をおこなわなければならない。

この連絡方法およびその応答手順は次のとおりである。

○ H O S T 手順

初期連絡

N O P 制御情報 (メッセージ種類 = 6) を発信する。

応答手順

初期設定指令の場合には、標準セグメントサイズ (1152 ビット) で初期設定をおこない、これが終了したら初期設定完了制御情報 (メッセージ種類 = 1) を発信し初期手続きを完了する。

セグメント長変更指令の場合には指定されたセグメント長で初期設定をおこないこれが終了したら初期設定完了情報を発信して初期手続きを終了する。

○ I M P 手順

初期連絡

セグメントサイズが標準ならば初期設定指令制御情報 (メッセージ種類 = 2) を発信する。

セグメントサイズが非標準ならば、セグメント長変更指令制御情報 (メッセージ種類 = 1) を発信する。

○ 応答手順

初期連絡に同じ。

E データの正規転送手順

H O S T - I M P 間における情報の授受はすべてこの手順によっておこなわれる。

この手順によって転送される情報には、N C P - I M P 間であらかじめ決められた制御情報とホスト中のプロセスの要求により作られる一般データである。

制御情報は 32 ビットで構成され、W D I コマンドにより転送される一般データは初期状態処理手順により決定されたセグメントサイズ (これを n ビットとする) によって、(n) ビットのセグメントに分割されて (最終セグメントは (n) ビット以下の不定) W D コマンドにより転送される。

制御情報には、それだけで意味を持つものと、一般データのプレフィックスとして使用され

るものの2種類がある。

データの正規転送手順において、ハードウェアエラーが検出された時は、両側の処理系に知られるので送り側が再送する。

○ 一般データが1セグメントの場合

データを送信する側は、WDIコマンドにより制御情報(メッセージ種類コード=0: Regular messageを示す)を送り、これが終了したのちWDコマンドにより1セグメント転送する。これを図示すれば次のようになる。

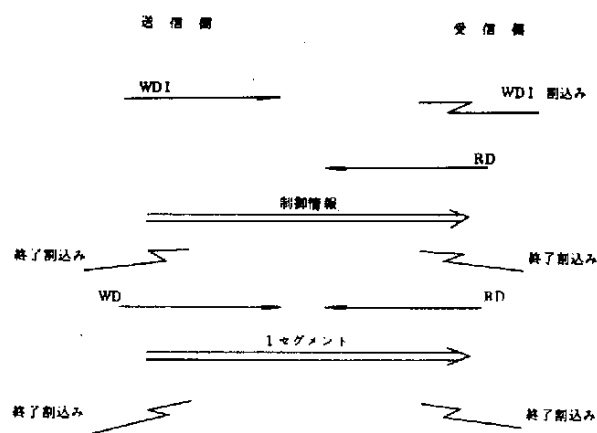


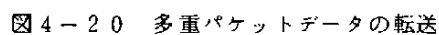
図 4-19 データの転送

○ 一般データが複数セグメントになる場合

データを送信するHOSTは、WDIコマンドにより制御情報(メッセージ種類コード=0: Regular messageを示す)を送り、IMPから対応するコネクション番号を持つものに対する制御情報(メッセージ種類コード=12: 複数セグメント受入れ準備完了)が送られてくるのを待つ。この間に他の情報ならば任意のものを送ることができる。

IMPから制御情報が送られてきたならば、ただちにWDコマンドにより一般データを構成する全セグメントを送らなければならない。

IMPからHOSTに送る場合にはWDIコマンドによって制御情報(メッセージ種類コード=0: Regular message)を送り、これにつづいて第1セグメントから最終セグメントまでをWDコマンドにより送る。



F 衝突手順

アダプタの両側から同時にコマンドが発信された場合に、アダプタあるいはチャネルなどの場所でコマンドの衝突がおこる。これは正常な場合に起こるとすれば、WDIとWDI、あるいはWDとWDIのいずれかである。

① WDとWDIの衝突

この場合にはWDIを発した方を優先させる。すなわちWDIを発した方は再びWDIを発することができる。WDを発した方は相手側からの割り込みを待たなければならない。

② WDIとWDIの衝突

直前にRDコマンドを発信していた方を優先させる。

いずれの場合にも、コマンド衝突で優先した方は、次のデータ転送タイミングにおいて相手側からデータが送れるように準備しなければならない。

G 緊急処理手順

緊急処理とは、HOSTあるいはIMPの間で相手との交信が不可能となった場合の処理をいう。

この処理には、ハードウェアの故障によるなどの回復不能の場合の処理とハードウェア、ソフトウェアのいずれかが原因となって一時的に交信不能となる場合の処理の2種類がある。

① 交信不能

コマンドの終了時に inoperable の情報が返えされた場合およびチャネルコマンドが timeout をされた場合に検知することができる。

この場合には、たがいに相手側と交信が不能になったむねのメッセージをコンソールに出力し、HOSTをサブネットから切はなす。

IMPは、アダプタのデータ折返し機能を使用して障害場所の明確化をおこなう。

② 交信一時中断

PSWがlock状態になることによりソフトウェアで検知できる。これは、IMP側がHOSTからのデータ読み込みバッファが無くなった時点での手段である。

この場合にはHOSTはIMP側から情報がくるまでIMP側にデータを転送してはならない。

H 終了処理手順

終了処理手順は通常の状態においてHOSTあるいはIMPが正常に処理を終了する手順を決めたものである。

終了処理の開始は一方が他方に対して、接待中止要求制御情報(メッセージ種類=4)を発信することにより、双方においておこなう。これを受けた処理系は、同様のヘッダーを確認の為に送らなければならない。

終了処理は、開始してから5分以内に終わらなければならない。

終了処理が終ると切断完了(メッセージ種類=5)を発信して、相手が死んだことにし、初期状態待ちにはいる。

4.2 F-NCP

コンピュータ・ネットワークに加入するHOSTには、必ずNCP(Network Control Program)と呼ばれるネットワークコントロール用のプログラムがある。FACOM 230-60/75のNCPをF-NCP(FACOM-NCP)と呼び、MONITOR VIシステムの前で稼動するように作成されている。

まず4.2.1ではMONITOR VIシステムの概要とF-NCPの作成に直接関係する事項について述べる。4.2.2ではMONITOR VIシステムの前で稼動するF-NCPを作成する場合の検討事項をあげる。4.2.3以降は各々の検討事項について詳細に述べる。

4.2.1 FACOM 230-60/75 MONITOR VI システムの概要

FACOM 230-60/75用のオペレーティング・システムはMONITOR VI(以降M-VIと略す)と呼ばれ、M-VIシステムは、バッチ処理、リモートバッチ処理、会話形処理、リアルタイム処理を同時平行に運用するシステムである。

M-VIの構造は、中核となる制御プログラム(スーパーバイザ)とその前で稼動する処理形態に対応したサブモニタからなる。

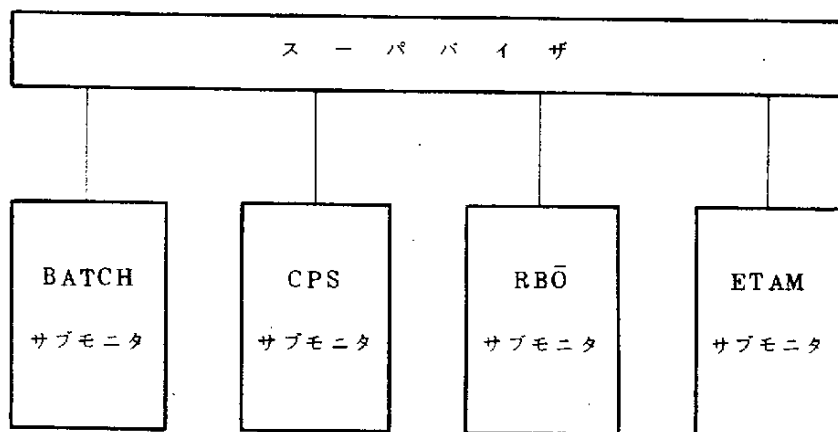


図 4-21 M-VIシステムの構造図

各サブモニタを紹介する。

(1) BATCH

通常のバッチ処理を行う。

(2) C P S

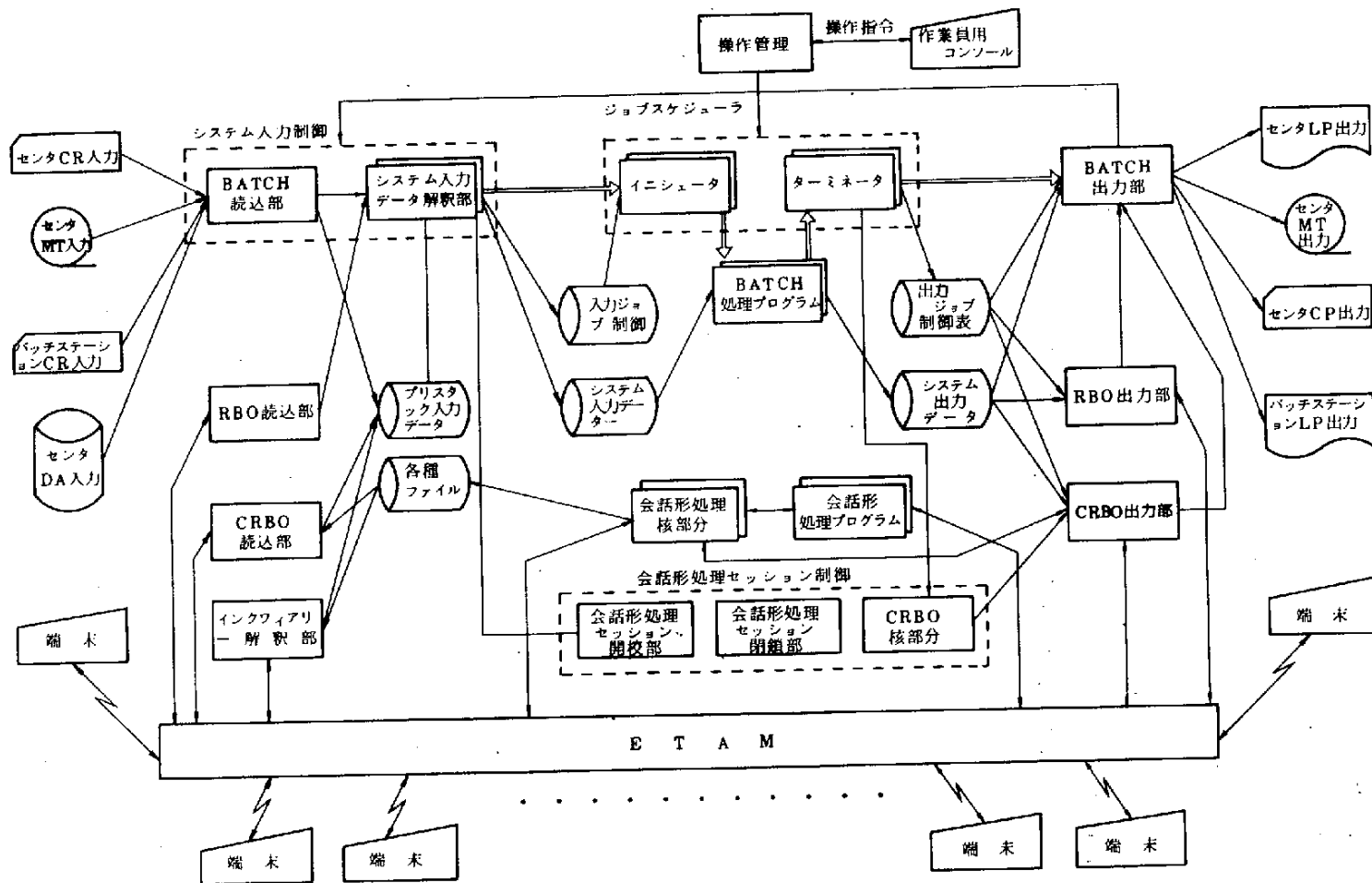
会話形処理と会話形リモート・バッチ処理を行う。

(3) R B O

リモート・バッチ処理を行う。

(4) E T A M

オンライン処理における端末の入出力サポートを行う。



以下、各処理形態のジョブ処理について説明する。

A ジョブの処理形態

(1) バッチ処理

ジョブの流れは次のとおりである。

a システム入力

ジョブは、次のいずれかの装置からシステムに投入される。

- センタ内のカード読取装置
- センタ内のディスクパック駆動装置
- センタ内の磁気テープ装置
- バッチステーションのカード読取装置
- リモートステーション

b ジョブ優先権

システムはジョブの処理順序をジョブ優先権によって定めるシステムでシステムに投入されたジョブは、その処理が開始されるまでは記憶上に貯えられる。(Stacked Job)

ジョブ優先権には、16のレベルがあり、各レベルは1～16の整数で表わされる(数の大きい方が優先権が高い)。ある時点において、システム内にスタックされている複数のジョブがあり、しかもそれらの中にジョブ優先権の異なるものがあれば、ジョブ優先権の高いジョブは、低いジョブよりも先に処理される。つまり、ジョブの処理順序はシステムに投入された順序ではなく、ジョブ優先権により定められる。

c ジョブの処理開始

処理開始の順番がまわってきたジョブは、イニシエータにより、主記憶にロードされ、実行を開始する。

取外し可能なボリュームを使用することをFD文で指定したジョブの場合には、その処理開始に先立って、イニシエータがオペレータにボリュームの取付けを依頼する。このメッセージはコンソールタイプライタ装置かまたは文字ディスプレイ装置に表示され、ジョブ番号、ボリューム通番、装着すべき機番等が指示される。

磁気テープ装置やディスクパック装置でAVR (Automatic Volume Recognition) 機能を持つものを使用している場合には、オペレータは装着完了を通知する必要はない。

d ジョブの多重処理

同時に処理されるジョブの数をジョブ多重度という。多重度の値はSTART INIT (INITIATE)指令により与える。

e ジョブの実行

イニシエートの処理が完了すると、そのジョブを構成する最初のジョブステップの実行が開始される。一般的には、処理中のジョブステップの数は処理装置の数よりも多いので、システムは実行優先権に従ってCPUを与える。

f システム出力

ジョブの処理結果は、通常は大記憶装置上にスタックされ、システムが自動的に利用者が指定した装置に出力する。

このような装置としては、

- ラインプリンタ装置
- カードせん孔装置
- 磁気テープ装置
- バッチステーションのプリンタ装置
- リモートステーション

などがある。

(2) 会話形処理、リモートバッチ処理、会話形リモートバッチ処理

M-VIのタイムシェアリングの利用形態としては、次の三つのものがあり、これらは一つの管理プログラムで統一的に制御されており、これをCPS (Conversational Programming System)

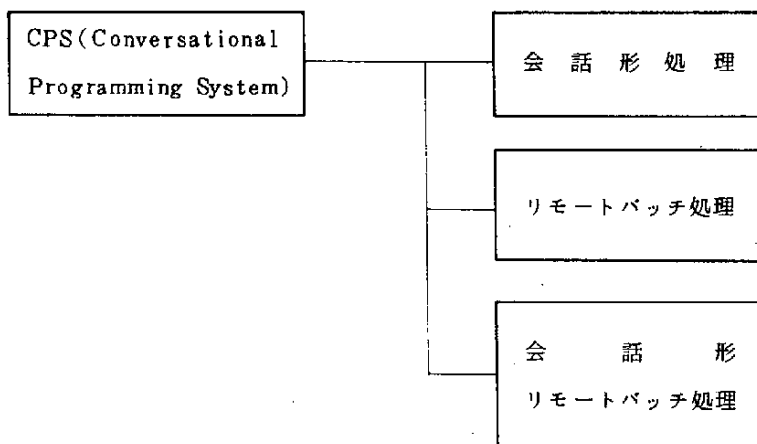


図 4-23 タイムシェアリングの利用形態

会話形処理では、タイプライタ装置、文字ディスプレイ装置のような端末からCPSと会話しながらプログラムを作成し、実行させることができる。

リモートバッチ処理は、カード読取装置とラインプリンタ装置が組になって計算機と回線

を経由して接続され、カード読取装置からは、センタでバッチ処理を行なうと同じようにカードを入れて計算機に処理を依頼する。回線をとおして計算機に送られたジョブは通常のバッチジョブと同様に処理されて、その結果が回線を経由して端末のラインプリンタ装置に出力される。

会話形リモートバッチ処理はタイプライタ装置のような端末から会話的に入力し、翻訳実行といった処理そのものはリモートバッチで行なうものである。

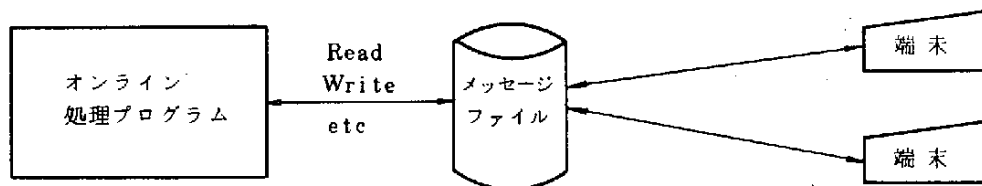
(3) ETAM (Extended Telecommunication Access Method)

ETAM は遠隔地とのデータの送受を通信回線を経由して行なう際の回線制御に関するオンライン処理サポートモジュールである。

オンライン処理において端末と通信を行なう上で、必要な回線ネットワーク制御、端末制御はETAMが吸収している。

ETAMの最大の特長は、オンライン処理プログラムがETAMの用意する各種マクロ命令、サービス機能を使用することによって一般バッチ処理の順編成ブロックレベルおよびレコードレベルのアクセス法と同一インターフェースで端末とのデータの送受信を行なうことができる点である。

① 端末とオンライン処理プログラムの結合



オンライン処理プログラムと端末との結合はメッセージファイルという中間ファイルを経由して行なわれる。

メッセージファイルはオンライン処理プログラムのFD文によって定義される。

¥FD FILE=(NEW, メッセージ・ファイル名),

CTERM=((端末名, 端末定義名),)

上記したFD文の例では、このオンライン処理プログラムが起動されるとオンライン処理プログラム—メッセージファイルという接続が出来る。

一方、端末から端末操作員指令

¥¥CONNECT M=メッセージファイル名

を投入することによって、当該端末とメッセージファイルの接続が実現し、

オンライン処理プログラム—メッセージファイル—端末

という接続が出来る。

以後、端末からの入力メッセージは接続中のメッセージファイルに格納され、オンライン処理プログラムからの READ 要求によって処理プログラム域へ転送される。またオンライン処理プログラムから端末への送出データは処理プログラム域から、接続中のメッセージファイル経由で宛先端末へ出力される。

端末とメッセージファイルの解除は、端末操作員指令、またはジョブの終了時点に行なわれる。また、メッセージファイルとオンライン処理プログラムの接続の解除は、CLOSE マクロ命令、またはジョブの終了時点に行なわれる。

② アクセス・マクロ命令

アクセス・マクロ命令には次の4つがある。

READ/LGETS

WRITE/LPUTS

またこの他に、OPEN/CLOSE/CHECK等のマクロ命令があり、アクセス・マクロ命令を含め、これらのマクロ命令は全て順編成ブロックレベルおよびレコードレベルのアクセス法と同一インターフェースである。

③ 端末操作員指令

端末から ETAMあるいは実行中のオンライン処理プログラムに対する指示を行なう指令である。

$$1) \begin{matrix} \text{¥¥} \\ \left\{ \begin{array}{l} \text{CONNECT} \\ \text{CON} \\ \text{C} \end{array} \right\} \end{matrix} \left\{ \begin{array}{l} \text{J=ジョブ名〔(メッセージ・ファイル名)〕} \\ \text{M=メッセージ・ファイル名} \end{array} \right\}$$

機 能

この指令を発信した端末と指定されたメッセージファイルを接続する。または指定されたジョブのジョブ制御文を^{注)} T J C ファイルから読み込みジョブを起動する。ジョブが起動された後、そのジョブで定義したメッセージファイルと端末を接続する。

注) T J C ファイル……端末起動ジョブのジョブ制御文を格納するファイルであり、

T J C ファイル作成ユーティリティによって作成する。

ii) その他

- ¥¥ DISCONNECT, ¥¥ DCON, または ¥¥ D……メッセージファイルと端末との接続を切る。
- ¥¥ INTRPT, ¥¥ I……割り込み指令
- ¥¥ STOP, ¥¥ S……ストップ指令
- ¥¥ EOF……EOF 指令
- ¥¥ TEST, ¥¥ T……テスト指令

- ¥¥ RECOVER, ¥¥ R..... 端末障害の場合の代行端末宣言指令

B 領域管理とファイル管理

(1) 領域管理

実行プログラム単位は、1個以上の領域 (domain) から構成されている。ひとつの領域は、主記憶空間中の連続した番地を占めるが、領域と領域との間の距離関係は不定である。すなわち、番地割当てに関して領域は互に独立である。また領域の主記憶空間中の位置は、ジョブステップの実行中にロールイン、ロールアウトや、主記憶リアロケーションによって変化することがある。

領域はまた記憶保護の単位でもある。記憶保護に関して開設ずみの領域は次の5つの状態をとる。

- N (参照不可) 状態
- R (読出し可) 状態
- RW (読書き可) 状態
- RX (読み実行可) 状態
- RWX (読書き実行ともに可) 状態

主記憶は

- HCM
- LCM

の2種に分かれる。M-VIでは256 K (ただしK=1024語) より小さい番地をもつ主記憶をHCMと呼び、256 K以上の番地をもつ主記憶をLCMと呼ぶ。

HCM上の1 K語境界からはじまる1 K語を1ブロックと呼ぶ。

HCM上に配置される領域は連続した整数個のHCMブロックが割当てられる。すなわち1 K語境界からはじまり1 K語の整数倍の大きさをもつ。

LCM上の32 K語境界からはじまる32 K語を1ブロックと呼ぶ。LCM上に配置される領域は連続した整数個のLCMブロックが割当てられる。すなわち32 K語境界からはじまり32 K語の整数倍の大きさをもつ。

(2) ファイル管理

データ管理でいうファイルとはCPUに対して同じ処理目的で入出力されるレコード群またはあるきまった形式をしているレコード群のことをいう。

ファイルには、処理のためのデータの集まりであるもの、システム運営のための制御情報の集まりであるもの、処理プログラムの集まりであるもの等いろいろな内容のものが存在する。

データ管理は、ファイルをつくるときに使用する外部媒体の性質と前記のファイルの内容

の違い、処理する目的の違いに応じて、それぞれ違った処理手段を提供している。

ファイルは処理手段の違いを識別するために、いくつかのファイル編成に区分される。

- ① 順編成 (Sequential Access Method)
- ② 順インデックス編成 (Indexed Sequential Access Method)
- ③ 直接編成 (Basic Direct Access Method)
- ④ 分割型順編成 (Basic Partitioned Access Method)

a 装置の割当て

ジョブ制御文のFD文でファイルが必要とする装置の割当てを受けておく必要がある。媒体が大記憶または磁気テープである場合には、装置上に適切なボリュームが装着されているかどうかモニターによって確認される。

b 領域の割当て

大記憶を媒体として新しいファイルを創るときには、ファイルに必要な領域の割当てを受けなければならない。

領域の割当てはFD文で必要な領域を記述することでジョブ管理が、領域割当て要求を代行してくれる。

① ファイルの定義

a FCB

プログラマは、ファイルの処理を始める前にファイルの実体についての情報を整理してシステムに提供しなければならない。この制御情報の集まりがFCBと呼ばれるものである。

FCBはFile Control Blockの略で、ファイル制御ブロックのことである。

FCBは、アセンブラであるFASPプログラムの中に記述される命令のひとつであるFCBマクロ命令に関連しており、FASPプログラムを翻訳した段階で、実行時にファイルを制御するために必要な情報を記憶する領域として確保される。

プログラマがFCBマクロ命令のなかに記述すべき主な情報として

ファイルの特徴

ファイル処理のしかた

があるが、FCBに十分な情報が書き込まれるのはファイル処理を開始する (OPEN処理) 時である。ファイル処理に必要な情報の源は4種あって、優先順位はFCB修正プログラムがもっとも高く、

FCBマクロ命令

FD文

ファイルラベル

F C B修正プログラム（後述のマクロ命令 PREOPEN と OPEN との間での使用者の
F C B書換えの部分をいう。）

の順である。つまり、ある制御情報を F C Bマクロ命令と F D文の両者が指定しているような場合は F D文中の指定が採用され F C Bマクロ命令中の指定は無視される。

F C Bはファイルを代表するもので、ファイルを OPENすること、F C Bを OPENすることとは、プログラマからみればまったく同義である。また、ファイル処理を終結させるには CLOSE処理があり、CLOSE によってファイルが切り離され、F C Bは意味を失う。

F C Bとファイルとは 1対1に対応するのが普通であるが、同じような特徴をもついくつかのファイルを続けて処理する場合には、F C Bの領域と情報を物理的に流用することができる。しかし論理的には、F C Bは常にファイルと 1対1にしか存在しえない。

b F D文

F D文はジョブ制御文の 1つである。

F C Bの項でも触れたように F C Bを構成する情報は F D文中の記述から与えることもできる。しかしその主目的は F C Bマクロ命令中に記述がなかった部分を充足することである。F D文から F C Bに提供しうる情報としては

レコード形式

バッファの数、大きさ

使用者が任意に選択できるような付加機能

がある。

F C Bマクロ命令と F D文のファイル処理に関する地位は、F C Bマクロ命令中の記述が処理に共通な事項を概括的に定義するのが本来の目的であるのに対して、F D文中の記述は処理を実行するのに必要な条件を具体的に定義するのが、本来の目的であることから必然的にきまってくる。

c ファイルラベル

媒体上のラベルの項で述べたように、ファイル・ラベルは 1個以上のブロックから構成されるファイル識別用情報の集まりである。

大記憶上に記録されているファイルのファイル・ラベルはひとまとめにされて VTOC ファイルとして存在し、磁気テープ・ボリューム上のファイル・ラベルは、ファイル毎の前書きラベルおよび後書きラベルとして存在している。

ファイル・ラベルに含まれているファイル仕様に関する情報は、F D文や F C Bマクロ命令で記述する必要がない。

4.2.2 F-NCP基本構想

FACOM 230-60/75 は高速演算性能, 大容量記憶装置, 大容量ファイル, リモートバッチおよびTSSサービス, 各種オンライン・ソフトのサポート等豊富な資源を所有している。これらの資源を管理しているのがM-VIシステムである。我々はこの資源をより有効的にかつより広範囲に利用させるためにネットワークとM-VIシステムの橋渡をするF-NCPの実現性を検討してみた。

M-VIシステムを考慮してみると, 会話形処理形態に於いてコンピュータ・ネットワークの概念を取り入れていないので仮想端末用のTCPを作成し, 会話形処理のインターフェースについて検討した。

また, 自系のユーザには, 他系プロセスとコミュニケーションを行う場合あたかもFile Accessを行う場合と同じ手法でそれを可能にする。その手段としてM-VIシステムのデータ管理(その中でもSAM)インターフェースを取り入れることを検討した。

A F-NCP概要

概要を述べる前にいくつかの用語を定義しておく。

NCP独立プロセス

NCP独立プロセスとは, Batch, あるいはTSSモニタにより起動されたプロセスであって, 通常NCPはこれを監視, 強制終了などを行うことができないプロセスを呼ぶ。

NCP従属プロセス

NCP従属プロセスとは, ユーザがLOADP コマンドをNCPに発することにより作り出されるプロセスを呼ぶ。NCPはこのプロセスに対して監視, 強制終了など管理を行う。

F-NCPのM-VIにおける位置を図4-24に示す。

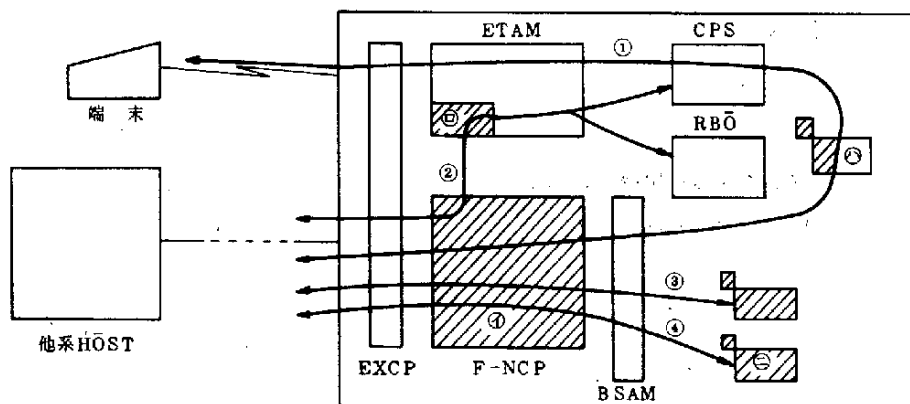


図4-24 F-NCPのM-VIにおける位置

データの流は、前図 — 線によって示す。  は当センタ作成のモジュールである。

①は端末から他系HOSTのプロセスを使用するルートである。

②は端末と他系HOSTのプロセスの間でデータの交換を行なうプロセス(以後TELNET と呼ぶ)であり、端末よりのCALLコマンド(CPS)によって起動される。

③はネットワークからCPS又はRBOを使用するルートである。③はその際のF-NCPとETAM間のインターフェースをとるモジュール(以後NTCPと呼ぶ)である。

NTCPはETAM内でネットワークの最大コネクション数だけの仮想の端末を制御するモジュールとして動作すると共にF-NCPとETAM間のデータの受け渡しを行なう。

③はBatch プロセスと他系HOSTのプロセスのコミュニケーションである。

④はF-NCPが他系HOSTよりの依頼で起動したプロセス(NCP従属プロセス)とネットワークとの間のコミュニケーションである。

B F-NCP機能

F-NCPの基本的機能は、集約すると次の5つからなる。

- ① プロセスの起動および監視
- ② プロセス間のコネクションの確立及びデータの授受
- ③ プロセス間の情報の授受
- ④ コンピュータ・リソースの使用状況の把握
- ⑤ 使用者ごとのサブネット利用の会計情報のセット

①、②、③はHOST-HOSTプロトコルおよびHOST-IMPプロトコルのため、④はAJDのため、⑤はサブネットを利用するにあたり、料金徴収のため夫々必要な機能である。

F-NCPはサブモニタで作成する。まずはサブモニタの必要な機能を簡単に述べ、次に実現にあたり検討すべき事項をあげる。

C サブモニタ

M-VIでは、モニタは1つのスーパーバイザと、いくつかのサブモニタから構成される。そしてこれらはジョブステップという形をとる。これらのジョブステップは処理プログラムのジョブステップと同様に、いくつかの領域をもつことができ、いくつかのプログラムから構成される。ジョブステップには必ずジョブステップごとのジョブステップ作業域がある。ジョブステップ作業域には、スーパーバイザ作業域、サブモニタ作業域、処理プログラム作業域の3種がある。それぞれの作業域は、直接参照部分と間接参照部分にわかれている。ジョブステップ作業は、制御表という単位で管理する。サブモニタとしては、BATCH, ETAM, CPS…… などがある。F-NCPもサブモニタで作成することにし、ここでは実現するにあたり、特に必要な機能をあげる。

- (1) サブモニタは、センタ操作員がSTART指令を待ちこむと、スーパーバイザがイニジェート

し、STOP 指定でターミネートする。(常駐する)

- (2) サブモニタは、タスク領域、GETDM 領域、ジョブステップ作業域を使用できる。
- (3) INIT マクロ、IJEXECマクロを発信させて、ジョブ・ステップを発生できる。(イニシエータ機能)
- (4) 発生させたジョブ・ステップが正常終了、異常終了しても制御はサブモニタにかえる。また、サブモニターでこのプロセスを強制終了(IJABORT マクロを発信)させることができる。(ターミネータ機能)

D 検討事項

検討事項としては以下に述べる様に 8 個ある。

(1) CCAインターフェース

JIPNETのCCA (Channel to Channel Adaptor)の制御はM-VIのブロックレベル/レコードレベル・サービスでは制御できない。このような装置に対するサービスはM-VIにおいてはEXCP (EXECUTE CHANNEL PROGRAM) マクロ命令及びアペンデージインターフェースを使用することになっているのでこれを利用する。その際のコマンドの衝突などをいかにするかについて「4.2.3 CCAインターフェース」として述べる。

(2) ETAMインターフェース

M-VIはTSSサービスとしてCPSサブモニターを用意している。TSSを利用するには通常、端末装置が窓口となるが、この端末装置を一括管理しているのがETAMである。このETAMといかにしてインターフェースを持たせるかについて「4.2.4 ETAM インターフェース」として述べる。

このインターフェースを持たせる目的は、他系ユーザにネットワーク経由でM-VIのTSSサービスを利用させることにある。

(3) プロセス間コミュニケーションをいかに実現するか「4.2.5 プロセス間コミュニケーション」として述べる。

(4) NCP従属プロセスに対してプロセスの起動、監視を行うがそれについていかに行うかを「4.2.6 プロセスの起動と監視」として述べる。

(5) F-NCPに対するサービスの要求

F-NCPでは、種々のネットワーク・サービス・コマンド(いわゆるマクロ)を用意し、ユーザプロセスにサービスを行なう。この場合、モニタ・レベルのサービスをいかにして行なうかを、「4.2.7 F-NCPに対するサービス要求」で述べる。

(6) 会計情報

プロセスがネットワークを利用する場合は、ジョブアカウンティングの必要が生じる。例えばNCP独立プロセスの場合でもNCPとは独立にM-VIのアカウンティング規則により

情報はとれるがそれは別にネットワークの利用に関する情報をとらねばならないが、これをいかにするかを「4.2.8 会計情報」で述べる。

(7) 高位プロトコル

高位のプロトコルをいかにして実現するか「4.2.9 高位プロトコル」として述べる。

(8) NCICの情報採集のための

- Jobの個数に関して
- コア領域の使用状況
- CPUの使用状況

等の採集について、いかにして採集するか「4.2.10 NCIC情報採集」として述べる。

以下、各検討事項について、各項目に分けて検討する。

4.2.3 CCAインターフェース

IMPからの割込みはアペンデージ・インターフェースで処理しコマンドの発信にはEXCPを使用して制御する。

A アテンション・アペンデージ・インターフェース

NCPがCCAからのアテンションを直接受取る事を可能にする。

このルーチンは、処理プログラム内の1つのルーチンとして作成され、その起動は所定ecbに対するpostによる。

アテンション—— Read/Write の間隔に時間的制限がある場合、EXCPの発信が間に合わないことがない様に次の様な点に考慮をしておかねばならない。

- EXCP ルーチンをRRとする(システム編集時)
- アペンデージ・ルーチンを含む処理プログラムの主記憶実行優先度を最高にする。

B 衝突

コマンドの送信・受信時においてタイミング的なエラーとしてコマンドの衝突の問題があげられる。

コマンドの衝突が起る可能性のある箇所としては

- ① アダプタ上での衝突
- ② データチャネル上での衝突
- ③ モニターNCP間の衝突

があげられる。各々について以下に説明する。

(1) アダプタ上での衝突

アダプタ上でWDI/WD-WDI/WD の衝突が起る可能性がある。これについては4.1.4で詳述してあるのでここでは省略する。

(2) データチャネル上での衝突

コマンド発信とWDIによる割込みがチャネル上で衝突した場合で、次の2つの場合がある。

- a コマンド発信が割込みよりもわずかに早かった場合
- b 割込みの方がコマンド発信よりもわずかに早かった場合

各々異なるステータス情報でコマンドがリジェクトされるが(WDIを発した系ではコマンド未終了なので)、NCPではリジェクトされたコマンドを再び発信する。その結果 incompatible (コマンドミスマッチ) のセンス情報でコマンドが異常終了した場合は、①と全く同じなのでNCPでは①の処理を行なう。

C モニターNCP間の衝突

NCPがEXCPに対して発信したWDまたはWDIがEXCPで受けられる前に、NCPのアペンデージャーにIMPからのWDIによりアテンション割込みが通知された場合である。

この時NCPでは発信したWDまたはWDIがEXCPに受けられIOSPVから incompatible の通知があるのを待って、その通知が来た段階で①の処理を行なう。

4.2.4 ETAMインターフェース

M-VIでは、既存のTSSモニタ(会話形処理、リモートバッチ処理)を他系ユーザより使用できる機能を持たせる予定であるが、Bで述べる様な問題について未解決である為、単に方針を述べるにとどめる。

A NCP-ETAM インターフェースの考え方

既存のTSSモニタ(CPSサブモニタ)はETAMと直接結合している。この為、他系ユーザがCPSを使用する為にはNCPとETAMとの間に何らかのインターフェースを設ける必要がある。

ETAMはオンライン端末との通信を制御するプログラムであるが、NCPとのインターフェースを設ける為に次の様な方法をとる。

NCP-ETAM 間に複数のインターフェースモジュール(これをSERVERと仮称する)とETAM内にインターフェース用のTCP(これをNTCPと仮称する。NTCP……NCP interface Terminal Control Program)を作成する。

NTCPは一般の端末制御プログラムとETAMとのインターフェースに合わせる為、SERVERを端末(仮想端末)とみなして制御を行なう。

一方、SERVERはRead/WriteのPortを持ち、NTCPからのRead, Write依頼をNCPのインターフェースに変換して依頼する。NTCPはどのコネクションを使用している

かは全然、意識せず、単にどのSERVERを使用しているかということのみ意識している。一方NCPは他系ユーザプロセスとのコネクションを確立する度に空いているSERVER(端末)を依頼元ユーザプロセスにダイナミックに割り付けていく。同時にコネクション番号とSERVER名(端末名)との対応表を作成する。そして以後closeしないかぎりはそのユーザプロセスには必ず同一SERVER名(端末名)を用いる。

CPSはETAMとインターフェースを持っているので以上の方法によって、セッション——ユーザ名——端末名のリンクが確立し、CPSはネットワーク経由のサービスを一般オンライン端末に対するサービスと同じインターフェースで行なうことができる。

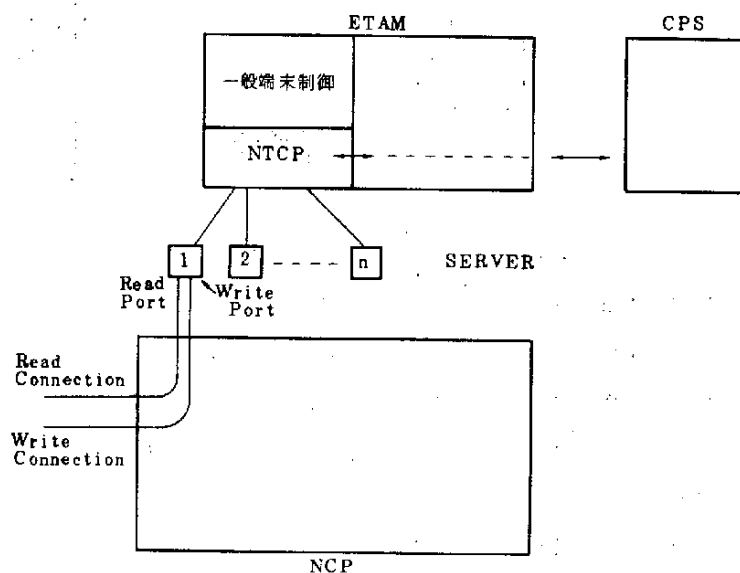


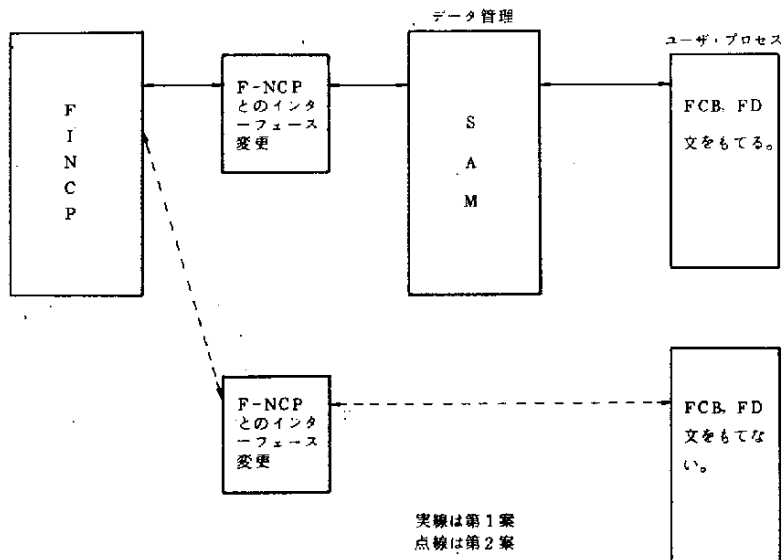
図 4-25 NCPとETAMの関連

B 問題点

既存のOS(M-VI)には、仮想端末という概念がない為NTCPの作成に当ってはOSの改造が必要となる。この改造がどの程度のものになるかは現在検討中である。

4.2.5 プロセス間コミュニケーション

プロセス間コミュニケーションを実現する為に、2つの案を計画した。第1案はユーザ・インターフェースをM-VIのデータ管理(その中でもSAM)インターフェースと合わせ、データ管理を経由してF-NCPとインターフェースをもつ案、第2案は、システム・サービスのために直接F-NCPとインターフェースをもつ案である。



A 第1案

F-NCPでは、ファイル管理の項で説明した手順と同じ手順を採用する。すなわち、コネクションの確立要求は、ユーザ・プロセスが通常のOPENコマンドを出すことによって行なわれ、解放要求は、CLOSEコマンドを出すことによって行なわれる。また、コネクション確立後のデータの送信はWRITEコマンド、受信はREADコマンドが使われる。

これらのコマンドが出されると、制御は、M-VIのデータ管理（その中でもSAM）を経由し、F-NCPに渡る。

F-NCPの主な機能は次の3つである。

- ① FCBとFDBの結合により port-idを知り、コネクションを確立する。
- ② コネクションの解放。
- ③ データの送信・受信。

B 第2案

第2案では、あるプロセスがプロセス間コミュニケーションを行なう場合に、データ管理を経由して呼ぶことができないとき、すなわちFCB、FDBをもてない時のために、直接F-NCPにサービスを要求するルートをもうける。第2案では、第1案の機能の代わりにユーザプロセスとF-NCPのインターフェースの変更処理のみを行なえばよい。

C 検討結果

第1案を採用した場合、次の様な修正、追加が必要である。

a 修正箇所

- (1) データ管理に於いて、コミュニケーション判別する箇所

例えばOPENモジュール、CLOSEモジュール、READモジュール、WRITEモジュール

ル、CHECK モジュール、WAIT モジュール、イベント通知モジュール等

(2) 言語処理プロセッサがコミュニケーションを行なう場合、そのプロセスが所持している I/O ライブラリーを修正しなければならない。

(3) バッチ、あるいは TSS モニタで起動されたプロセスが異常終了した場合、F-NCP に制御がとないため、それぞれのターミネータを修正しなければならない。

b 追加箇所

(1) コミュニケーション用の FDB 文 (FDB 作成マクロ)

(2) FDB と FCB の結合を行なうモジュールの作成

c その他

(1) M-VI システムを修正する場合、修正そのものが困難であること。修正した場合、エラーの追求が大変である。

(2) M-VI システムが拡張中なのでデータ管理の各モジュールの制御表、インターフェースが変更される可能性がある。

(3) F-NCP を作成するために必要な作業量が多い。

これらのことを考慮して、第 2 案をまず作成し、完成次第に第 1 案に取りかかることとした。

4.2.6 プロセスの起動と監視

ここでは、NCP 従属プロセスの起動、終了について述べる。起動は F-NCP のジョブ・ステップ・イニシュエータが行ない、終了は F-NCP のジョブ・ステップ・ターミネータが行なう。

イニシュエータとターミネータの起動は F-NCP が行なう。起動としてはまず INIT マクロによって、ジョブ・ステップを発生させる。(ジョブ・ステップの発生とは、ジョブ・ステップのジョブ・ステップ番号を割当て、その作業域を主記憶上に獲得して、その初期化および、いくつかの制御表の関連づけを行ない、そのジョブ・ステップを、主記憶優先行列に組み入れることをいう)

次に IJEXEC マクロによって領域を開設し、制御を移行する。

A イニシュエータの機能

イニシュエータの主な機能は次の 3 つである。

- ① FDB の作成
- ② ファイルの割当て
- ③ プロセスの起動

(1) FDB の作成

イニシュエータは、従属プロセスが必要とする FDB を作成する。FDB には、プロセス間コミュニケーション用の FDB と自系 File 用の FDB の 2 種類がある。

しかし、現在前者の FDB は作成することができないので、後者の FDB についてのみ述

べる。

(2) ファイルの割当て

イニシエータは、プロセスが使うファイルを割当てる。NEWファイルに対しては、S・ALCTDマクロを発信させてファイルの創成を行ない、OLDファイルはS・ATCHDマクロを発信させて、OLDファイルにファイル定義名を与える。

(3) プロセスの起動

イニシエータは、起動すべきプロセス名をパラメータとしてXCTRLを出す。このときXCTRLは領域を開設し、プロセスをローディングし、制御をわたす。

図は従属プロセスの起動概略である。

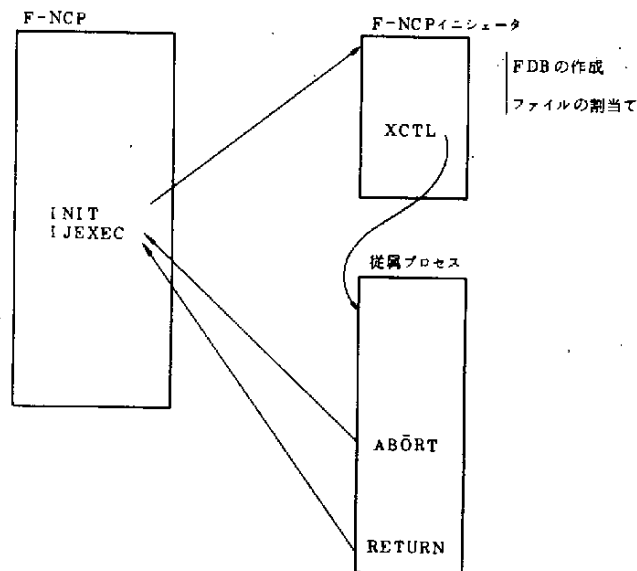


図 4-26 従属プロセスの起動概略

B ターミネータの機能

ターミネータの主な機能は、次の3つある。

- ① コマンド／マクロの終始末
- ② スペースの終始末
- ③ プロセスの終始末

(1) ユーザ・プロセスがシステム(M-VIシステムとF-NCP)に依頼した処理で、まだ完了していないものに対して、その依頼をキャンセルする。

(2) スペースの終始末

プロセス終了時、それらの中で定義されたファイルで、未スクラッチのものを必要に応じ

て S・SCRCHD マクロでスクラッチ処理する。

(3) プロセスの終始末

- ① S CRM (scratch memory)マクロを発信させて、開設されているプロセスを閉鎖し、開設されているすべての領域の閉鎖を行なう。
- ② TERM (terminate)マクロを発信させて、ジョブステップ作業域を返却し、このジョブステップを主記憶優先行列から、切り離す。

C プロセスの監視

プロセスの有無を監視しなければならないが、NCP 従属プロセスについては行なえる。プロセスが正常終了のときも異常終了のときも必ずNCPに制御が移行される。また、従属プロセスを起動要求したプロセスから「終了要求」がきたときや、従属プロセスがループ状態になったときも IJABORTマクロによって終了(結果的には異常終了)させることができる。ただしNCP独立プロセスのときは上記の様に制御が移行されないので Batch, CPSのターミナタを修正するか常にジョブ制御表を監視しなければならない。

4.2.7 F-NCPに対するサービス要求

F-NCPでは、ユーザプロセスのネットワークに関するサービス要求に対して種々のネットワーク・サービス・コマンド(マクロ)を提供する。サービスコマンドのマクロ命令文はアセンブルで展開されTRP命令を発信し、ネットワークサービス用サービスモジュールを起動する。

サービスモジュールを使用するのは、次の様な理由による。

- ① 異なるモジュール間の連絡、制御を容易に行なうことができる。
- ② モニタレベルのサービスが可能である。
- ③ NCPがSAMインターフェースをとる為。

サービスモジュールは基本的にはTRP命令(スーパーバイザ・コール)によって呼出される。

処理プログラムから呼出されるサービスモジュール(ここではネットワークサービスコマンド)の識別のための番号としてSF番号がある。あるSF番号を持つサービスモジュールは同一システム内で同時に2個以上登録されることはなく、SF番号を持つ登録されたサービスモジュールと、使用中のSF番号は1対1に対応する。

ネットワークサービスコマンドの登録は、システム編集時、SFCODE文でSF番号、マクロ名を登録する事によって行なう。

4.2.8 会計情報

本システムではFACOM 230-60/75の標準会計ルーチンで収集する会計情報に加えて以下の会計情報の収集を行なう。

- ① NCPジョブ(NCP従属プロセス)会計情報
- ② ネットワーク使用会計情報

A NCP会計ルーチン

ネットワークジョブサービスの運用において使用される会計ルーチンは、次の様に4個の小ルーチンより構成される。

- ① 初期設定センタルーチン
- ② NCP・ジョブ開設センタルーチン
- ③ NCP・ジョブ閉設センタルーチン
- ④ 終了処理センタルーチン

これらのルーチンは、NCPサブモニタのプロセス管理ルーチンより呼出される。

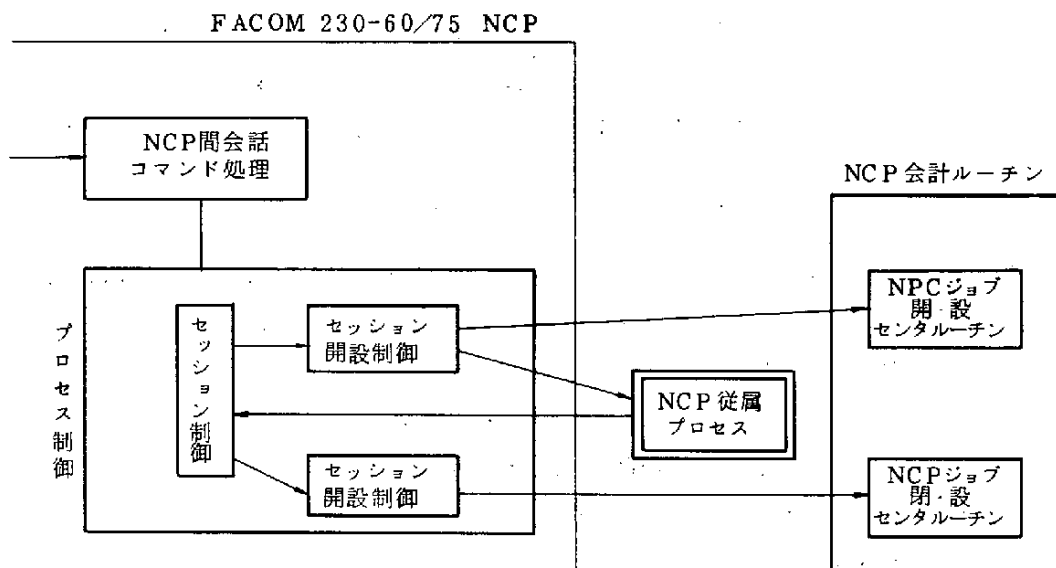


図4-27 NCPとNCP会計ルーチンの関連

(1) 初期設定センタルーチン

本ルーチンは操作員より START NCP指令を受けた時、NCP初期設定ルーチンより呼び出される。

処理内容は、NCP会計ルーチンの初期設定及び会計ファイルの作成を行なう。

(2) NCPジョブ開設センタルーチン

このルーチンは、NCPのセッション開設制御から EXEC形式で呼ばれ、ネットワークユーザの資格審査など、NCPのセッションを開始できるかどうかのチェックを行なうルーチンである。

セッション開設制御から呼び出された場合、パラメータとして会計表がレジスタ経由で渡

される。

NCPでは本ルーチンと呼ぶ前に会計表にユーザ名、パスワードをセットしておかねばならない。本ルーチンでN.で始まる8文字のジョブ番号、現時刻等を会計表にセットする。

(3) NCPジョブ開設センタルーチン

本ルーチンはNCPのセッション閉設制御からEXEC形式で呼ばれ、レジスタ経由で渡された会計表に主記憶占有時間、ファイル・アクセス回数等をセットし、会計ファイルへロギングする。この時後述するネットワーク使用料金表をその後に続けてロギングする。この後会計情報をネットワークへ出力する。

(4) 終了処理センタルーチン

本ルーチンは、操作員よりSTOP NCP指令を受けた時NCP終了処理ルーチンより呼び出される。

NCP会計ルーチンの終了処理及び会計ファイルのクローズを行なう。

(5) 会計表

会計表に収録する内容としては、次のものがある。

- ① 主記憶占有時間
- ② 主記憶占有量
- ③ ファイルアクセス回数
- ④ チャネル占有時間

B ネットワーク使用会計情報

商用コンピュータネットワークシステムでは、ネットワーク使用に対して課金を行なうことが必要となる。

ネットワーク使用料課金に関しては次の様な問題がある。

- ① 既存の会計ルーチンとのインターフェース
- ② 会計情報の送信
- ③ ネットワーク使用料金の二重取り

本システムでは、以下に述べる様にこの問題を処理している。

(1) 既存の会計ルーチンとのインターフェース

本システムではネットワーク使用料金の課金は、現在の会計ルーチンのインターフェースに合わせることを原則としている。この為、ネットワーク使用会計情報は1ジョブ単位に行ない、且つその会計情報は既存の会計ルーチンの会計表の後部に追記し、1個の会計表として、会計ファイルに書き込む方法をとる。

具体的には次の様になる。

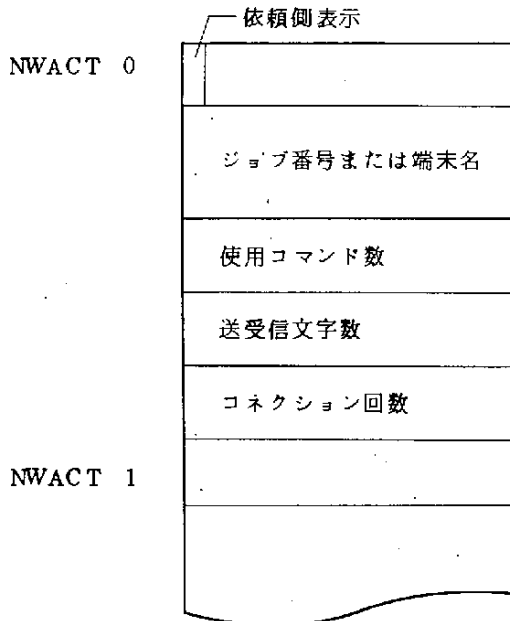


図 4-28 NCP 会計表

NCP は図の様な会計表を用いてネットワーク使用料の課金を行なう。会計表の個数は port の数だけ用意され、NCP 領域中に連続に並べられている。

会計表にはジョブ単位の情報記録される。

① 会計表の初期設定

NCP はユーザプロセスよりコマンド依頼を受けるとユーザプロセスのジョブ番号をキーとして全会計表をサーチする。その結果、ユーザプロセスのジョブ番号と同じジョブ番号を持つ会計表が見つからない場合は、そのユーザプロセスからの初めての依頼とみなし、空きの会計表に

依頼元ユーザプロセスのジョブ番号を記録する。

② 会計表の更新

①の会計表サーチにおいて目的会計表が見つかった場合は一般のユーザコマンドの依頼なら「使用コマンド数」を+1 更新する。CLOSE 依頼であれば「コネクション回数」を+1 更新する。

またデータを送受信したときは「送受信文字数」に受信したデータ長を加算する。

③ 会計表の内容消去

ユーザプロセスが終了すると、会計ルーチンは独自の会計表を作成した後、その会計表の後にネットワーク使用会計表をつけ加えて会計ファイルに書き込む。会計ルーチンでは、センタ固有の会計表の後にネットワーク使用会計表がある場合は、一般の TSS とは異なり、会計情報のネットワークへの出力はしないこととする。

以上の処理が終了したら、会計ルーチンは該当するネットワーク使用会計表の内容を消去する。

(2) 会計情報の送信

一般の TSS においては会計情報の送り先は端末だが、コンピュータネットワークの場合はそれがユーザプロセス（一般ジョブ）だという点で相違がある。本システムでは NCP でこの相違を吸収する事は拡張性のうえから問題があり且つ処理が複雑となる為、ネットワーク間では会計情報の送信は行なわないこととする。各センタでは各自のリソースの使用状態

に従って使用ユーザ毎の請求書を作成する。

(3) ネットワーク使用料金の二重取り

ネットワーク使用料を自系プロセス、他系プロセス各々課金すると料金の二重取りになってしまう。これを避ける為に依頼側のみネットワーク使用料をとることとする。つまりNCPが自系のユーザプロセスAよりオープンコマンドを受けた時、未だ目的のプロセスBからオープンコマンドを受けていないなら、ユーザプロセスAを依頼側として、ネットワーク使用会計表に依頼側表示をたて、以後CLOSEコマンドを受けるまでネットワーク使用の度にネットワーク使用会計表の更新を行なう。

クローズコマンドを受けた時点でNCPは依頼側表示をクリアする。一方、依頼側でないプロセスからのコマンド依頼に対してはネットワーク使用会計表の更新は行なわない。

4.2.9 高位プロトコル

高位プロトコルとして、種々のものを考えることができるがその実現方法は全く同じであるので次の2つについてのみ述べる。

- ① RDASプロトコル
- ② TELNETプロトコル

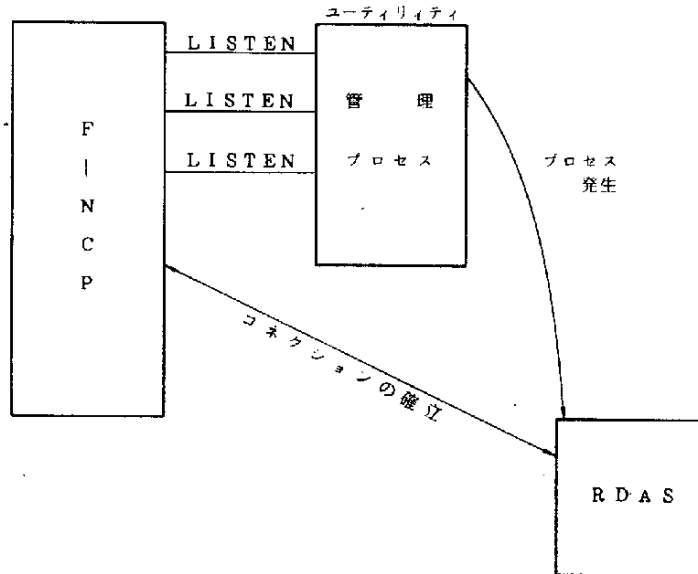
A RDASプロトコル

RDASプロトコルは、ユーザが使用しようと思うファイルがどこにあるか(自系、他系)を意識せずにファイル・アクセスができるようにするプロトコルである。このプロトコルは基本的なプロトコルを使用することにより実現することができる。

ユーザ・プロセスのプロセス間コミュニケーションは、SAMインターフェースをもっている。すなわちF-NCPで提供したFCB、FD文を用意し、OPENコマンド、CLOSEコマンド、READコマンド、WRITEコマンド等を使用する。この場合FCB、FD文にはコネクション情報の他にFile名、VOL、DISP、NEW/OLD等の項目を指定する。これらを用いてFile情報を作成し、相手プロセスに送る機能をF-NCPが持っている。File情報を受け取ったプロセスでは、Fileをアサインし、Fileに対してRead/Writeをだしてデータを読む。このデータは、コネクションを通して送ることにより実現することができる。

このプロトコルはユーザ・プロセスとして用意され、きまったPort-idがつけられている。

このようなユーティリティ的な機能を持つプロトコルを一括して管理するプロセスを作り、F-NCPに対してコネクション確立要求をだしておき、その要求がくると必要なプロセスを発生する。



このように管理プロセスに port-id とそれに対するプロセスを登録することにより、少量の記憶領域で多数のプロトコルをサービスすることができる。

B TELNET プロトコル

TELNET (Telecommunication Network) プロトコルは端末より他系 HOST にあるプロセスとコミュニケーションが行なえるプロトコルである。

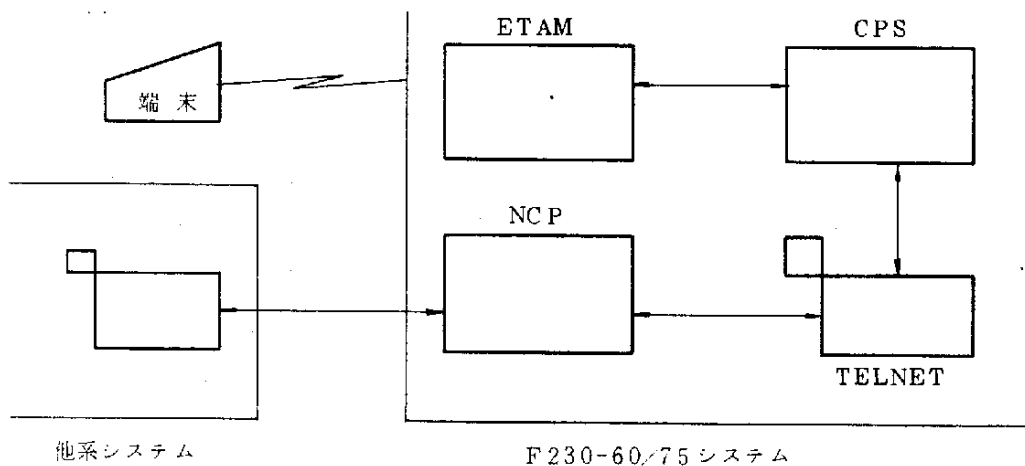


図 4-29

ここでは端末から、他系プロセスをアクセスする手順について述べる。

- (1) まず端末から ¥ Y C P S コマンドを投入して C P S (会話形処理モジュール) を起動する。
- (2) 次に端末より CALL TELNET を投入する事によってネットワークとのメッセージ交換を

行なう処理プロセス TELNET を呼出す事が出来る。

(3) TELNET からの Open 依頼によって NCP は宛先プロセスとの間にコネクションを確立する。

(4) 以後端末操作員は TELNET 経由で宛先プロセスとの会話を行なう。処理終了すると TELNET より Close 依頼を発信してコネクションを切断する。

その後端末より BYE コマンドを送信して CPS を終了させる。

4.2.10 NCIC 情報採集

F-NCP では NCIC の為の情報採集について現段階としては、次の情報を収集しておくこととする。

- ① ジョブの個数に関して
 - 実行中のジョブ個数
 - 実行待ちのジョブの個数
- ② memory の使用状況
 - 現在使用中の主記憶量
- ③ CPU の使用状況
 - idle Time

上記の項目について情報の採集の方法を述べる。

- (1) バッチジョブの実行待ちジョブ数及び実行中ジョブ数はバッチジョブ多重度管理表より採集する。
- (2) CPS ジョブの実行中ジョブの個数は、CPS サブモニタ固有直接参照作業域より採集する。
- (3) 現在使用中または空の主記憶量については、モニタの主記憶管理表より採集する。
- (4) CPU の idle time は M-V で使用した効率測定用プログラムによる idle time の測定結果が良好であったので、M-VI でも同じ方法で行なう予定である。

4.3 H-NCP

H-NCP (HITAC-Network Control Program) は、HITAC8450 処理装置内にあって、EDOS-MSO の助けをかりる事により、自己ホスト内に存在するプロセスの管理及び他 HOST の NCP との連絡をとる事を目的とする。

コンピュータ・ネットワークを通して他のコンピュータ・リソースの利用を要望する HITAC ユーザにとって、H-NCP はネットワーク・モニタとして存在し、ネットワークに結合された他の

プロセスと有機的に結びつく事が可能となる。

この目的を果たすために、H-NCPは、リアル・タイム・ジョブとしてHITAC 8450 内に常駐し、プロセス間コミュニケーションを基本とする各種の機能を持ちコンピュータ・ネットワークを達成する事を目標としている。

H-NCPとユーザ・プロセスの関係は、モニタと処理プログラムのな関係を保つ事が望ましい。すなわち、H-NCPは複数のユーザ・プロセスを一括管理し、そのプロセス間の相互関係の調和を保ち、さらにEDOS-MSOと各種の連絡をとる。これに対して、ユーザプロセスはあまりH-NCPの存在を考える事なくネットワークにアクセスする事ができなければならない。これらの事から、HITACにネットワークを実現する為には、H-NCPの設定及び、ユーザレベルのアクセス・マクロのサポートを為さなければならない。

4.3.1 EDOS-MSOの概要

HITAC 8450のEDOS-MSO(Extended Disc Operating System with Multi Stage Operations)オペレーティング・システムは、コントロール・プログラム、通信管理、FCP (File Control Processor)、システム・ジェネレーション・ルーチンより構成される。本項ではネットワーク・システムの実現上特に重要となるコントロール・プログラム及びFCPについて述べる。

コントロール・プログラム(エグゼクティブ)は、ジョブの進行を制御するジョブ管理、タスク及びプログラムの制御をするタスク管理、入出力装置のハードウェア制御を行なうハードウェア管理から構成され、FCPは、ユーザ・プログラムに組込まれて使用され、コントロール・プログラムの管理のもとで汎用の入出力データ・ファイルの処理を行なう。

以下にH-NCP及びネットワーク・プロセスの作成上必要とされるジョブ管理、タスク管理、主記憶装置の割当て及びFCPについて概説する。

A ジョブ管理

ジョブ管理は計算機システムに行なわせる仕事の開始から終了に至るまでの一連の手続きを統一して管理するために、ジョブの連続処理の制御、ジョブ・スタックとジョブ・スケジューリング、システム資源の割当て、ジョブ・アカウント、オペレータとの連絡、入力リーダの制御、出力ライタの制御、リモート・リーダ/ライタの制御、オペレータ・パーティションの制御及びプログラム・デバックの制御の機能を持つ。

EDOS-MSOにおけるジョブの性格はJOBステートメントによって記述され、1つのジョブは複数のジョブ・ステップより構成され、ジョブの処理される形態は、バッチ処理、リアルタイム処理、リモート・バッチ処理のいずれかである。ネットワークにおけるH-NCPはリアル・タイム処理のジョブとし、ネットワーク・プロセスはバッチ処理のジョブとする。

ジョブの入力は、カード読取り装置、磁気テープ装置及び磁気ディスク装置より行なわれ、この装置の事をシステム・リーダーと呼ぶ。システム・リーダーから入力される一連のジョブは実行される段階ではジョブ・ストリームを形成する。EDOS-MSOでは最高12本までのジョブ・ストリームを形成することができる。ジョブ・ストリームとは、ジョブの連続的な流れを言う。ジョブ単位で登録されたジョブが各ジョブ・クラスごとに実行され、1つのジョブが終了すれば次のジョブをジョブコントロールが取出し実行する。

EDOS-MSOは最大4個のステージより構成され、各ステージはR/A, B, Cの3個のジョブ・クラスで構成され、このジョブ・クラスに対してシステム資源が割当てられる。

(1) ジョブの実行過程

EDOS-MSOではジョブの実行に必要なジョブ制御情報や入力データ及びジョブの実行結果を一時的にディスクに格納する。これをジョブ・スタックと言い、ジョブ・スタック・エリア上でのジョブの状態は、実行中の状態、実行待ちの状態、実行抑止の状態に分けられる。

ジョブ・スケジューリングは、ジョブ・コントロールが行なうが、あるジョブを資源不足のために削除したり、抑止状態にした時には次の高い優先権を持つジョブを実行する。又特権エリアで主記憶装置の不足が生じた時の非特権エリアの一部分をロール・アウトする際の制御も行なう。

図4-30にジョブの実行過程を示す。

(2) 入力リーダーと出力ライター

ジョブ実行に必要なジョブ・コントロール・ステートメント群と、カード形式のデータ群をシステムに登録するプログラムが入力リーダーであり、入力リーダーはこれらを入力ジョブ待ちセキューに一時的にスタックする。入力リーダーはJOBステートメントが現われて次のJOBステートメント又はENDステートメントが現われるまでを1つのジョブとして、ジョブ単位に一度に多数のジョブを入力ジョブ・スタック・エリアに登録する。一度のジョブ・スタック・エリアの入力ジョブ待ちセキューは各ステージの各ジョブ・クラスごと、オペレータ・パーティション、開始時刻待ち、ホールド・ジョブおよびリモート・ジョブ用の待ちセキューのために最大16個作成される。

出力ライターはジョブの実行結果のデータと、システム出力データを出力ジョブの待ちセキューから取り出してラインプリンタ及びカード・セン孔機に出力するプログラムで、ライター・パーティション・エリアで実行される。出力ジョブ・スタック・エリア上では、それぞれの出力データは指定された出力タイプごとに出力ジョブの待ちセキューをつくる。出力ライターはそれら出力タイプのいずれかに属し、そのキュー上に登録されている出力データを順次出力する。

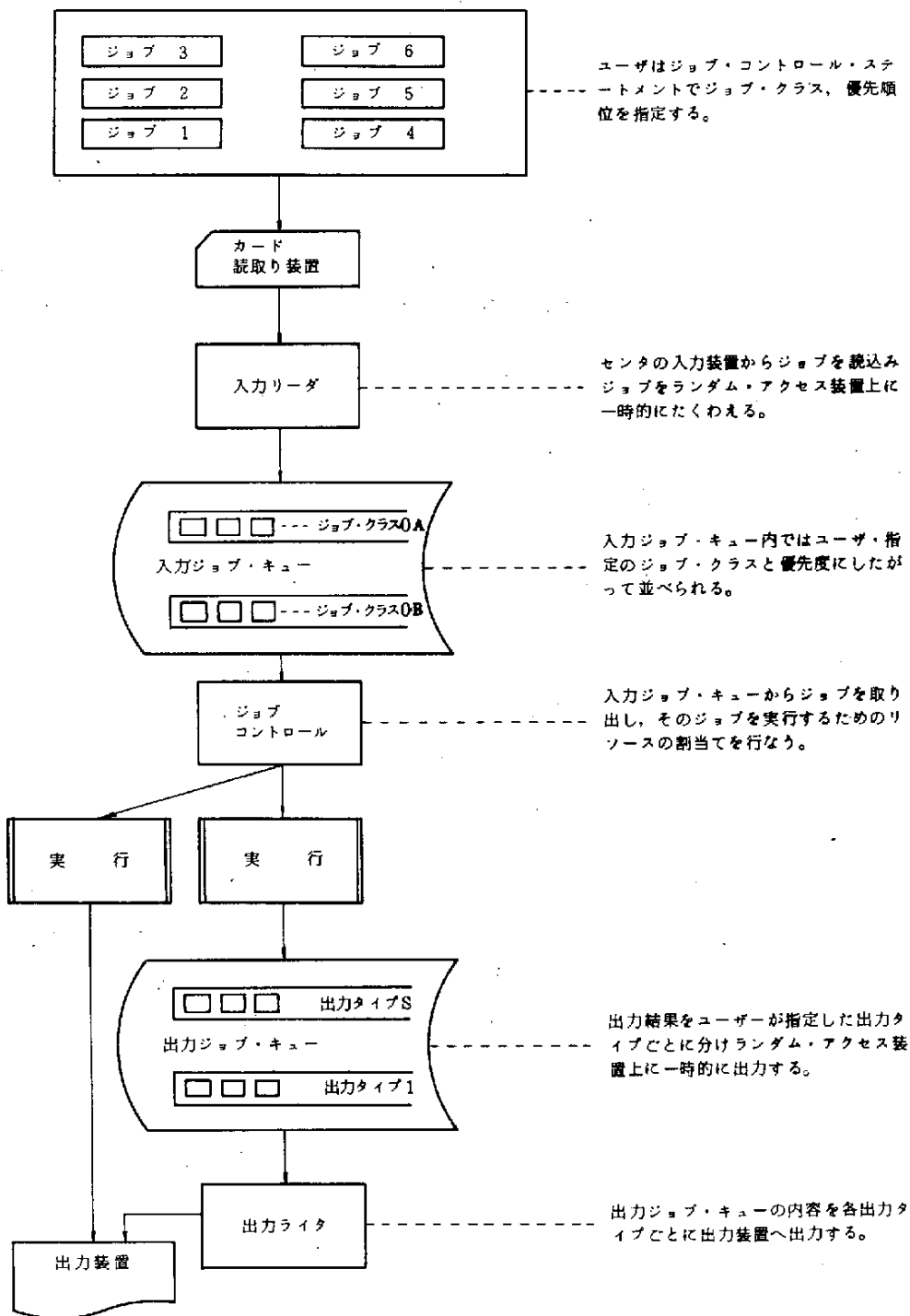


図 4-30 ジョブの実行過程

(3) オペレータ・パーティション

マルチ・ステージ・エリアで実行されるジョブに加えて、オペレータ・コマンドによりオペレータ・パーティション・グループで実行されるジョブがある。これをオペレータ・パーティション・プログラムといい、このプログラムはジョブ・ストリームを形成しない。オペレータ・パーティション・プログラムは最大4個であり、特徴はコンソール装置からの装置割当て、コンソール装置からのメモリ・ダンプ等が可能な事である。

B タスク管理

タスク管理は、ジョブ管理によって実行されたジョブを構成するプログラム及びタスクを制御し、全体として効率的に仕事を遂行してゆくための管理を行なうものであり次の役割を果たす。

(1) タスク制御

システム資源はジョブを構成するものとしてプログラム全体に割当てられ、タスクはプログラムの構成要素としてシステム資源を利用する。

EDOS-MSO のジョブ・ステップは一般にはシングル・タスク・プログラムであり、ネットワーク・プロセスは全てこの形態をとるが、マルチ・タスク・プログラムもリアルタイム処理のジョブにのみ許されていて、H-NCPはマルチ・タスク・プログラムである。マルチ・タスクではメイン・タスクと呼ばれるタスクとそのメイン・タスクによって起動されるサブ・タスク（最大63個）があり、メイン・タスクはサブ・タスクよりプライオリティが高く、全サブ・タスクの起動・停止にともなう処理を管理している。

マルチ・タスク・プログラムの特長は、処理中の待ち時間を他のタスクが利用でき、多重処理を同一プログラム内で可能にする事である。

(2) オーバレイ制御

EDOS-MSO コントロール・プログラムはシステム区域に存在し、システム区域は常駐エリアとオーバレイ・エリアに分けられる。コントロール・プログラムのうち頻繁に用いられる部分は常駐エリアにあるが、減多に使用されない機能はシステム・レジデンスに存在し必要な時にオーバレイ・エリアに呼ばれて実行される。又、ユーザ処理プログラムでもオーバレイ・ルーチンを要求する事があり、これらの制御を行なうのがオーバレイ制御である。

(3) プログラム制御

プログラム制御はユーザ・プログラムの制御に用いられ、それはユーザの処理プログラムの作成、実行についてのコントロール及びユーザ・プログラム間の通信機能の制御を行なう。H-NCPとネットワーク・プロセス間の会話はこのプログラム制御によって行なわれる。すなわちプログラム制御はプログラムの作成、オーバレイ構造の制御、オーバレイ・ロードの制御、共通データ・エリア方式によるプログラム間通信、割込み転送方式によるプログラ

ム間通信、ロード・ディレクトリーの常駐化機能などの制御を行なう。

(4) ロール・アウト／ロール・イン制御

EDOS-MSO では、特権エリアのジョブ（A/Rクラスジョブ）が、プログラム開始時やプログラム実行時に非特権エリア（B, Cクラス）にまで境界を越えて主記憶装置を要求した場合、非特権エリアのジョブは一時的にディスクにロール・アウトされる。ロール・アウトの原因が解除されるとロール・インされ処理が実行される。これらの制御を行なうのがロール・アウト／ロール・イン制御である。

(5) タイマの制御

時刻のサービス、処理装置使用時間のサービス、プログラムへの割込みサービス、ジョブ・アカウントのサービス、ジョブ・ステップの打ち切り、ジョブ開始・終了時刻指定のサービス、タスクへの割込みサービスなどを制御する。

C 主記憶装置の割当て

EDOS-MSO の主記憶装置は、システム区域とユーザ区域に大別され、システム区域は、コントロール・プログラムの内頻繁に使用される基本的なルーチンが占める常駐処理エリア、プログラムが必要とされるとき実行されるオーバレイ・ルーチン用のオーバレイ処理エリア、コントロール・プログラムの付加機能が処理される付加機能処理エリア及びコントロール・プログラムの用いるテーブルの4つのエリアから構成され、大きさは、常駐処理エリアとオーバレイ処理エリアを合わせて40KB（1KB=1024B）であり、付加機能処理エリアとテーブルの両エリアは、ユーザのシステム構成により異なる。ネットワーク実施時のシステム区域の大きさは110KBとする。

ユーザ区域の入力リーダー・パーティション、出力ライター・パーティション、ユーザ・OWNコード・パーティションおよびメモリ・シェア・パーティションはシステム・ジェネレーション時にエリアの大きさが決定される。またマルチ・ステージの各ステージとオペレータ・パーティション・グループの各オペレータ・パーティションもシステム・ジェネレーション時にエリアの大きさが決められる。各ステージの非特権エリアとオペレータ・パーティションはジョブを実行するときにそのジョブの要求メモリサイズ分だけ割当てられ、特権エリアのサイズは実行時に動的に変化する事が可能である。

ユーザ区域の1つのパーティションの主記憶装置割当てを考えると、必須エリアとオプション・エリアがある。主記憶装置上の実行中のプログラムの割当てを図4-31に示す。

サイズ(バイト)

200	コントロール・プログラムとのコミュニケーション・エリア	必須
20n+4	デバイス・リスト・エクステンション・エリア	
224	マルチタスク・コントロール・エリア	
1000	SYSOUT エリア	必須
1240	SYS IPT エリア	
104	MONTBL	必須
184	TCA	必須
184	ESA	
JOB ステートメントでの指定サイズ		必須
	ジョブ・コントロール・ステートメント・エリア	

n: ランダム・デバイスの個数

図4-31 プログラムのメモリ割当て

D FCP

FCPはEDOS-MSO コントロール・システムの下で入出力装置を制御する。入出力装置の操作を実行しようとするユーザは、アセンブラのソース・コーディング中にマクロ命令を記述する。アセンブル中にマクロ命令に出会うとマクロに対応する必要なコーディングと定数をジェネレートし、このマクロを処理するFCPルーチンを作り出す。FCPルーチンはファイ

ル定義に基づいて要求されたルーチンのみがオブジェクト・プログラムに含まれるようにモジュール形式に設計されている。

プログラマは入出力装置の制御に次の3つのレベルで対処する事ができる。

(1) ロジカル FCP レベル

入出力データをハードウェアのブロック単位でなく、論理的なレコードの単位で処理する。このレベルで処理されるファイルには、シーケンシャル・アクセスファイル、ダイレクト・アクセスファイル、インデックスド・シーケンシャル・アクセスファイルがある。

(2) フィジカル FCP レベル

入出力データの取扱いは、ハードウェアのブロック単位であるが、複雑なファイルのオープン、クローズ処理をファイル定義マクロと入出力マクロを用いる事により簡単に行なうことができる。

(3) フィジカル I/O レベル

入出力データの取扱いは、ハードウェアのブロック単位であり、ファイルのアクセス方法その他の制御は全てユーザの処理プログラムにまかされる。

ネットワーク・プロセスの扱うレベルとして特に重要視されるのはロジカル FCP であるので、以下にロジカル FCP について述べる。

ロジカル FCP はロジカル・レコードの処理を制御する。プログラマは、2つの形のマクロ命令（ファイル定義マクロと入出力制御マクロ）と、DTF ステートメントによってロジカル FCP を使用する。このロジカル FCP の機能は、ラベルの自動的処理、ファイル処理、レコード処理、入出力動作のオーバーラップ等がある。

a ファイル定義マクロ

ファイル定義マクロは、ロジカルファイルの特徴及びフィジカル入出力機器の指定と、実行される機能を定義し、マクロの種類は次の3種類である。

① DTF SR

レコードを物理的に配列されている順序に従ってアクセスするシリアル・ファイルを定義するマクロで、磁気テープ、磁気ディスク、カード読取機、ラインプリンタ等の機器の指定ができる。

② DTF DA

レコードを物理的な配列でなく、一定のアドレス変換方法に従って直接にアクセスできるダイレクト・アクセス・ファイルを定義するマクロで、磁気ディスクの指定ができる。

③ DTF IS

1つのファイルをキーの順序に従ってシーケンシャルにレコードをアクセスできると

同時に、ファイル中に設定されているインデックスを用いる事によってダイレクト・アクセスも可能な、インデックス・シーケンシャル・アクセスファイルを定義するマクロで磁気ディスクの指定ができる。

b 入出力制御マクロ

ロジカルFCPに用いられる入出力制御マクロは、ファイル定義マクロで定義したファイルの実際の入出力を指示するマクロで、ファイルを処理できる状態にしたり、処理できない状態にしたり、ブロッキング・デブロッキング、入出力エリアの交互利用、ロジカルレコードの読み書き、ファイルの始め及び終りの処理等の一連の処理を行なう。以下に主な入出力制御マクロについて述べる。

① OPENマクロ

このマクロはプログラム中の各ファイルを処理可能にする。このマクロは、ファイルがロジカルFCPによって処理されるときに使用し、DTFSR, DTFDA, DTFISのいずれかで定義されたファイルに適用される。

② CLOSEマクロ

このマクロはシステム中の入出力装置において以前にOPENマクロによって処理が開始されたファイルの処理を終了する。

③ GETマクロ

このマクロは入力エリアまたは特定のワーク・エリアのどちらかを処理されるロジカル・レコードを入力ファイルから取り出す。DTFSR, DTFISのいずれかで定義されたファイルに適用される。

④ PUTマクロ

このマクロは出力エリアまたは特定のワーク・エリア内で直接作成されるロジカル・レコードの書き込みを行なう。DTFSR, DTFISのいずれかで定義されたファイルに適用される。

⑤ READマクロ

DTFDAで定義されたダイレクト・アクセス・タイプのファイルからフィジカル・レコードを読み込んだり、DTFISで定義されたファイルの指定されたキーから、データレコードを読み込む時使用される。

⑥ WRITEマクロ

フィジカル・レコードをDTFDAで定義されたダイレクト・アクセス・タイプのファイルに書き込んだり、DTFISで定義されたファイルの指定されたキーを参照してデータ・レコードを書き込む時使用される。

c FCP留意点

ロジカルFCPを用いるユーザは、次の事に留意しなければならない。

- ① 読み書きの際固定長あるいは可変長レコードを自動的にブロックしたりデブロックしたりするので、入力ブロック・エリアの全てのレコードが処理された時、ロジカルFCPは自動的に次のブロックを読み込み、出力ブロック・エリアが一杯になった時書出す。
- ② 4つの汎用レジスタがFCPによって使われる。汎用レジスタ0と1はプログラムとFCPとの連絡情報を引渡すために使用され、汎用レジスタ14と15はジャンプ・アドレスとリターン・アドレスのために用いられる。
- ③ DTF SRで作成されたファイルはDTF DAで読むことができず、DTF DAで作成されたファイルはDTF SRで読むことができない。
- ④ 入出力制御マクロは、ファイルをオープン後、クローズするまでの間でしか使用できない。
- ⑤ ロジカルFCPを用いるとき複数個のファイル定義マクロでファイル定義をすることができ、DTFマクロはすべてジェネレータ形式であるので、リンケージの時外部参照名に注意を要する。

4.3.2 H-NCPの基本構想

HITAC8450は、データ処理に適合するようなバランスのとれた周辺機器構成と、グラフィックディスプレイ装置を結合したシステム構成を持ち、一般のデータ処理及びグラフィック処理に有効に利用されている。またソフトウェア・ファシリティとして、EDOS-MSOをオペレーティング・システムとしている。

ネットワーク・ユーザにとってモニタ的役割を持つH-NCPは、HITAC8450の特質を考慮して設定されていて、リアルタイム・ジョブとしてHITAC内に常駐する。

EDOS-MSOの特質を考えると、基本的にはTSS機能を果たすようには設計されていない。しかしながらネットワークの他HOSTから依頼のあったジョブは、他HOSTにおいてはTSSモードで動作している可能性がある。この為にH-NCPはEDOS-MSOのもとで擬似TSSを実現する必要が生じる。この擬似TSSは、キー・イン/アウト・シミュレーション機能を利用して行なわれ、バッチ・スケジュールとは無関係にジョブ・スケジューリングをする事ができる。すなわちH-NCPはHITAC8450のEDOS-MSOのもとで、基本的にはモニタの改造なしに、ユーザ・モニタとして作成される。

H-NCPとEDOS-MSOのエグゼクティブ(モニタ)及びネットワーク・ジョブ(ネットワーク・プロセス)との関係を図4-32に示す。

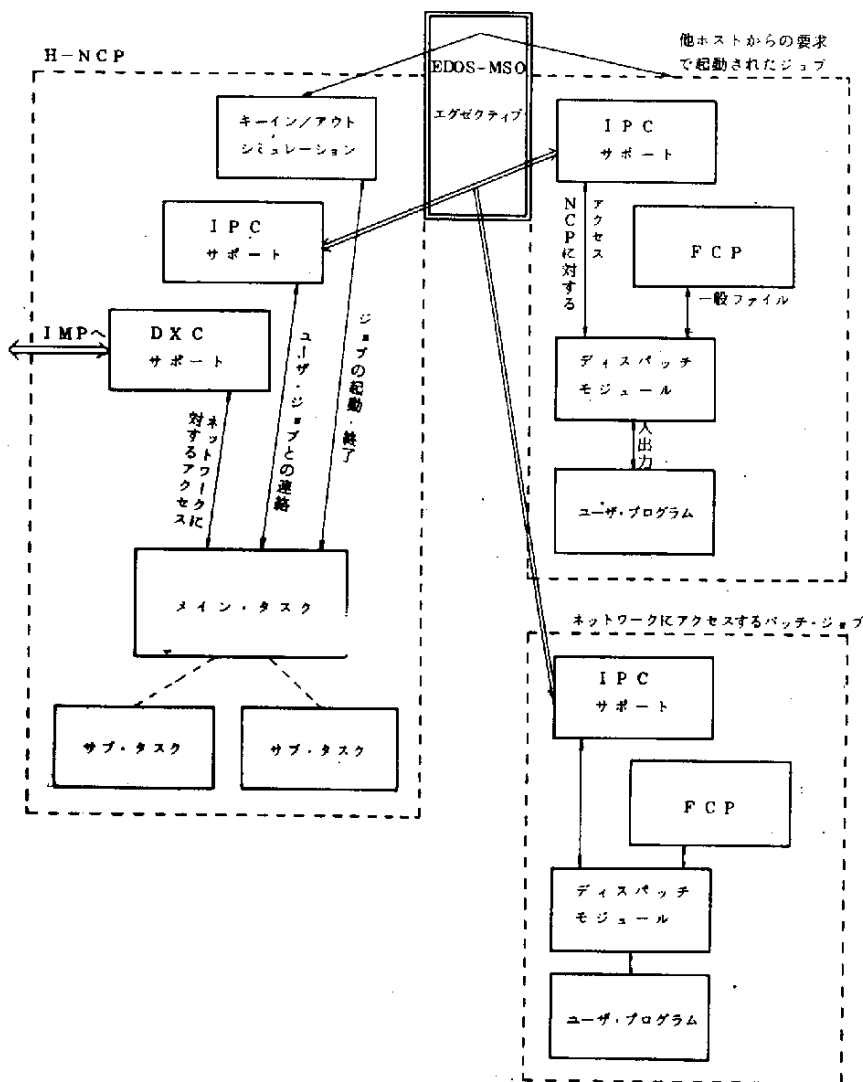


図 4-32 H-NCPとネットワーク・ジョブ

A H-NCPの機能

EDOS-MSOのもとで動作するH-NCPの機能は、次の通りである。

(1) NCPに関するプロセスとの連絡 (プロセス間コミュニケーション)

ネットワークを利用したいと考えるHITAC内のネットワーク・プロセスと、H-NCPとの会話機能及び、H-NCPが起動したネットワーク・プロセスとの会話機能を果す。

(2) ネットワーク・プロセスの起動、監視、終了の制御 (プロセス管理)

H-NCPはネットワーク・プロセスの要求に従ってHITAC内にネットワーク・プロセスを起動し、起動したプロセスの管理及び終了の制御をおこなう。

(3) IMPとのメッセージ交換 (HOST-IMPインターフェース)

HITAC 8450 にセレクト・チャンネルで結合されたチャンネル・アダプタを通じてIMPとのメッセージ交換をおこなう。

(4) コンピュータ・リソースの把握

HITACの使用状況を把握する機能で、ジョブの個数、メモリの使用状況、ファイルの使用状況等の把握を行ない、ネットワーク全体の動的管理資料とされ、AJDを実現するために必ず必要となる。

(5) 高位のプロトコル処理

ネットワークの特定な利用 (ファイル・ムーブ、グラフィックス) を大幅に推進するための高位のプロトコル (ハイレベル・プロトコル) の処理を行なう。

(6) その他の機能

H-NCP自身の制御のために、割込み処理、フロー・コントロール、コネクション管理等の機能を果す。

これらの各機能実現のために、H-NCPはEDOS-MSOがサポートするエグゼクティブ連絡マクロを使用する。このエグゼクティブ・連絡マクロは、H-NCPの作成時にアセンブラソース・コーディング上に記述され、EDOS-MSOとの連絡をとる事が可能となる。

H-NCPが機能実現のために使用するエグゼクティブ連絡マクロと機能の関係を表4-7に示す。

表4-7 エグゼクティブ連絡マクロとH-NCP機能

機 能	マクロの分類	マ ク ロ 名		
プロセス間 コミュニケーション	プログラム間連絡 マクロ	PCSET	PCWRT	PCRD
		PCWT	PCXT	PCCB
プロセス管理	キー・イン/アウト シミュレーション・マクロ	STKEY	KEYSM	
		KEYXT	KEYCB	
HOST-IMPの インターフェース	DXC装置 制御マクロ	STDXC	DXCXT	
		EXCPW	CCB	
高位のプロトコル 処理	マルチ・タスク 制御マクロ	SETSK	ALTSK	SETPL
		XITSK	WTSKT	DTSK
		GETWA	FREWA	EXTSK
		TTSK	FTTSK	TFEW
		PST	PSTR	TSGS
		RSGS	DWAM	

次にネットワーク・プロセスとH-NCPのコミュニケーションについて考えると、HITAC内で動作するネットワーク・プロセスは、H-NCPと連絡をとってネットワークの資源を利用する。すなわちネットワーク・プロセス内に何らかの手段でH-NCPとの連絡をとるための機能が必要となる。これを実現するためにユーザ・プロセスにプロセス間コミュニケーション用のマクロを準備して、ユーザが簡単にネットワークを利用できる様に考えている。このマクロは、ユーザレベルのプロトコルとして定義され、ユーザがネットワーク・プロセスを作成する時に用いる。

ユーザ・レベルのプロトコルは、他のネットワーク・プロセスとの会話に使用されるが、一般のファイルの感覚で使用するマクロとネットワーク専用のマクロがある。これらのマクロを用いる事によって、H-NCPとの会話に必要なプログラム間連絡マクロが、ユーザ・プロセスに附加される。

B H-NCPの構成

H-NCPはEDOS-MSOのもとで動作するスレーブ・モードのタスクであり、リアルタイム・クラスのジョブとしてHITAC内に常駐し、ネットワークの運営上必要とされるコントロール・プログラムである。このリアルタイム・クラスのH-NCPは、複数個のサブ・タスクを持ち、HITAC内に存在するネットワーク・プロセスとの連絡、他HOSTのNCPとの連絡を行なう。

H-NCPは機能実現のために図4-33に示される構造を持つ。

各機能は互いに関連を持つが、H-NCPの円滑な動作をさまたげるものではない。各機能における処理内容は以下の通りである。

(1) DXC管理

IMPとのデータの授受に関する管理を行なう。このDXC管理は、アダプタを通じてIMPから送信されてきたデータの処理及び、H-NCPからIMPへ送信するデータの処理を主として行なう。IMPから送信されてきたデータの処理は、DXC割込み処理ルーチンで受信され処理される。その処理が終了した時点でフロー管理に制御を渡す。IMPへの送信は、フロー管理より送られてきたデータをもとに行なう。

(2) フロー管理

IMPからDXC管理を通じて送られてきたデータに対して、その受信結果についての応答が必要とされる。このフロー管理では、IMPへの応答用のメッセージを作成し、DXC管理を通じてIMPへのデータ転送を行なう。また、IMPから送られてきたデータをコネクション管理に渡す役割を持つ。コネクション管理から送られてきたデータに対しては、DXC管理に送るデータ群を作成し、DXC管理に制御を渡す。又、フロー管理の処理として、H-NCPの状態把握、IMP、アダプタの状態の把握及び、他HOSTに存在するNCP

の状況把握等も行ない、ネットワークにおけるIMPとH-NCP間のデータ・フローを制御する。

(3) コネクション管理

他HOSTに存在するネットワーク・プロセスと、HITAC内に存在するネットワーク・プロセスとの間のコネクションの状態の管理及び、コネクション待ちの管理を行なう。このコネクション管理は、フロー管理を経て送られてきたコネクション要求のデータの解析、プロセス管理、データ管理に渡すべきデータの解析及び、H-NCP内で処理するデータの解析等を行なう。又、他の機能から送られてきたデータが、IMPへの送信データの時には、フロー管理への制御の移行も同時に行なう。このコネクション管理は、H-NCPの中核となる機能で、ネットワークのHITACにおける状態を全てコントロールするものである。

(4) データ管理

H-NCPとHITAC内のネットワーク・プロセス間のデータ転送を管理する。H-NCPとネットワーク・プロセス間の連絡はプログラム間連絡マクロ(IPCマクロ)によって行なわれる。このデータ管理では、コネクション管理を通して送られてきたデータをネットワーク・プロセスに送信したり、ネットワーク・プロセスから送られてきたデータをコネクション管理に送るために、データの解析、作成等を行なう。HITAC内のネットワーク・プロセスが複数個存在する時には、イベント管理と連絡をとり、データ転送の制御を行なう。

(5) イベント管理

HITAC内に存在するネットワーク・プロセスの管理を行なう。このイベント管理では、データ管理より送られてきた情報をもとにして、複数個のプロセスのイベント状況、待ち状況、プロセスの状態を知り、同時に各プロセスのリソース・シェアの状況を判断し、データ管理にその情報を渡す。

(6) プロセス管理

コネクション管理の要求によって、HITAC内のネットワーク・プロセスの起動、終了の管理及び、ネットワーク・プロセスがコンソールに出したメッセージの解析、状態把握等を行なう。このプロセス管理は、キー・イン/アウト・シミュレーション・マクロによってEDOS-MSOのエグゼクティブと連絡をとる事によりなされる。プロセス管理が起動したネットワーク・プロセスの管理は全てこの管理ルーチンで処理され、コネクション管理への送信データが作成される。データ管理におけるネットワーク・プロセスとの連絡は、コネクションが確立されているものに限られていたが、このプロセス管理では、コネクション確立の前段階が存在する。又、この管理ルーチンでは必要に応じて、キー・イン/アウト・シミュレーションにより、EDOS-MSOのエグゼクティブと連絡をとり、エグゼクティブが持つ各種の情報を入手する事ができる。

(7) 割込み管理

H-NCPは各機能実現のためEDOS-MSOのエグゼクティブと連絡をとるが、DXC管理、データ管理、プロセス管理、リソース管理のある部分は割込み処理ルーチンを持ち、割込みモードで動作する。この割込みモードは非同期に起こるために、これらを統括管理してH-NCPの動作を円滑化させる事が必要となる。この割込み管理では、各種割込みモードで動作する割込み処理ルーチンの制御、待ち合せキューの処理、アドレス変換等の処理を行なう。

(8) リソース管理

HITAC内で処理中のジョブの個数、記憶領域の状態、入出力装置の使用状況などを管理する。他のNCPからの連絡又はH-NCP自身が必要とする時に、コネクション管理、フロー管理より連絡がされ、各種情報を入手する。リソース管理ではこの情報の入手のために、キー・イン/アウト・シミュレーション・マクロを実施したり、他のマクロを使用してエグゼクティブと連絡をとる。又このルーチンでは、ある一定時間間隔でHITACの使用状況を管理し、会計情報の把握を行なう。

(9) コマンド管理

他NCPから送信されてきたデータのコマンド解析、他NCPへ送信するためのコマンド作成のために、コネクション管理、フロー管理が利用する。すなわちこのコマンド管理は、IMP-HOSTプロトコル、HOST-HOSTプロトコルの管理をするものである。

(10) NCPタスク管理

H-NCP内のサブ・タスクの管理を行なう。H-NCPではハイレベル・プロトコルの処理のために複数個のサブ・タスクを起動する。このサブ・タスクの起動要求は、コネクション管理より出され、NCPタスク管理では、サブ・タスクと同期をとりながら処理を行なう。

(11) ハイレベル・プロトコル処理

ハイレベル・プロトコル処理は、H-NCPのメインタスクから呼び出されるサブ・タスク又は、H-NCPが起動するプロセスで、このハイレベル・プロトコル処理には、グラフィック・コミュニケーション、ファイル・トランスファ等が考えられる。サブ・タスクはNCPタスク管理によって管理され、プロセスは、プロセス管理によってOPEN-ENDEDに管理される。

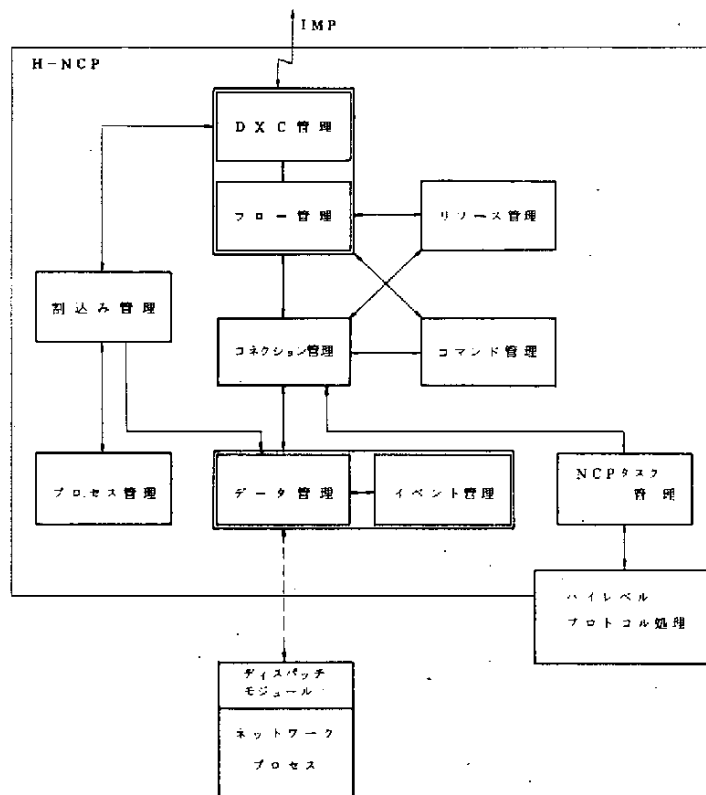


図 4-33 H-NCPの構造

C コントロール及びデータ・フロー

H-NCP内のコントロール及びデータの流れは、非同期に生じる割込みによって複雑な形態をとるが、H-NCPの各処理ルーチンは原則として、独立に処理される。

各機能のコントロールは、入力されてくるデータの種類によって異なり、入力されるデータには、HITAC内のネットワーク・プロセスからの連絡によるデータと、アダプタで結合されているIMPからの連絡によるデータとがある。またデータの流れは、H-NCP内での流れと、ネットワーク・プロセスとの間の流れ及びIMPとの間の流れに大きく分けられる。

次にコントロール及びデータの流れを、IMPからの連絡によるデータ処理と、ネットワーク・プロセスからの連絡によるデータ処理に分けて説明する。

(1) IMPからの連絡によるデータ処理

IMPから何らかの連絡があった時、H-NCPはそのデータが何であるかを判断しなければならない。連絡されたデータは、プロセスの起動、終了に関するデータ、コネクションの確立、開放に関するデータ、HITAC内のネットワーク・プロセスへの連絡データ、H-

NCPへの連絡データ、IMPの状態を知らせるデータに分けられる。これらのデータの種類に従ってH-NCPは各種の機能モジュールを実行し、データがその実行に従って流れる。この時モジュール内でテーブルの検索、マクロの実行等が行なわれる。次に連絡されたデータ別の処理を述べ図4-34にその処理過程を示す。

a プロセスの起動、終了処理

DXC管理によって受け付けられたデータは、プロセスの起動又は終了に関する情報である事が識別されると、フロー管理に引渡される。フロー管理では追加情報が必要であるかどうかの判断を行ない、追加情報の必要な時には、H-NCPは待ち状態となる。全てのデータがそろった所で、コネクション管理を通じて、プロセス管理に制御が渡る。プロセス管理では、キー・イン/アウト・シミュレーションによってプロセスの起動又は終了の処理を行ない、その結果に従って、プロセス管理テーブルの書替えが行なわれる。最後に、処理の完了情報を相手NCPへ連絡する。

b コネクションに関する処理

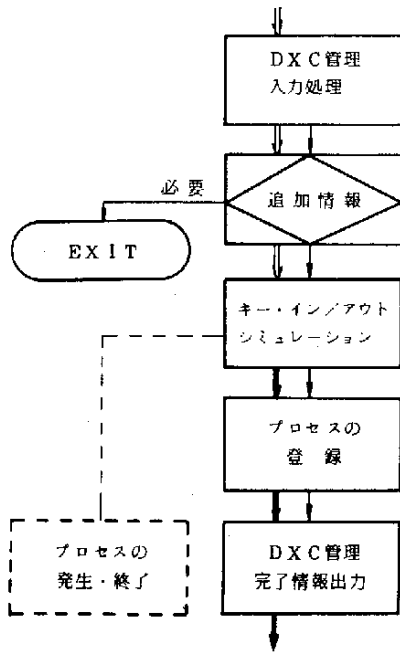
コネクションの確立、解放に関するデータは、DXC管理、フロー管理を通じて判断され、コネクション管理で、コネクション・テーブルの検索、コネクション待ち状態の判断がなされ、コネクション確立又は開放の処理がなされる。その後完了情報の通知を相手NCPに対して行なう。コネクション・テーブルにはコネクション待ち、確立されたコネクション等が登録されている。

c プロセスへの連絡通知

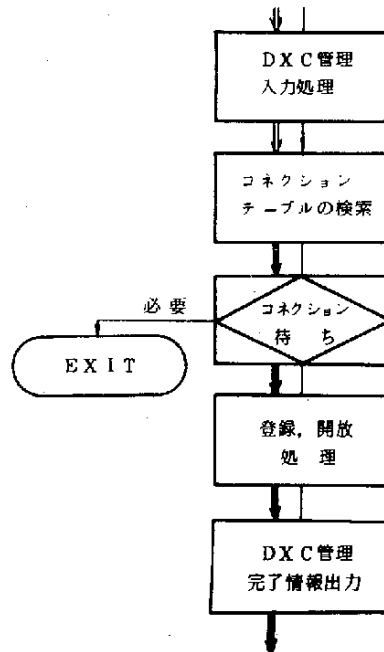
プロセスへの連絡処理は、すでに確立しているコネクションを持つネットワーク・プロセス間でのデータ転送を意味する。DXC管理、フロー管理を経てコネクション管理に渡されたデータは、コネクション管理において、コネクションが確立しているかどうか調べられる。その後データ管理に制御が渡り、対応するネットワーク・プロセスにデータを転送する。この時データ管理は、イベント状況を知るために、イベント管理と連絡をとる。HITAC内のネットワーク・プロセスにデータを転送した後、完了情報を相手NCPに連絡する。

d H-NCPへの連絡処理

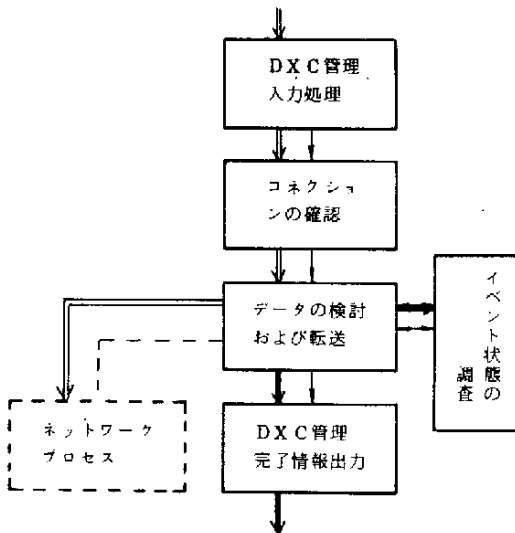
他HOSTのNCPからH-NCPに対してHITACの状態を聞いたり、ハイレベル・プロトコルの処理を要求する場合に、NCP間での連絡がなされる。HITACの状態を調べる場合には、DXC管理、フロー管理を経たデータが、リソース管理にわたり、ジョブの個数、主記憶装置の状態、ファイルの状況が検討され、その結果が相手NCPに連絡される。ハイレベル・プロトコルの処理要求があると、コネクション管理を経てNCPタスク管理に制御が渡る。NCPタスク管理では要求されたハイレベル・プロトコル処理を含む



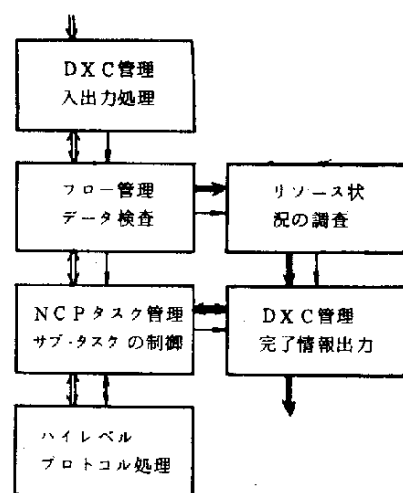
プロセスの起動・終了処理



コネクションに関する処理



プロセスへの連絡処理



H-NCPへの連絡処理

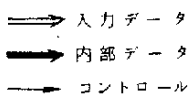


図4-34 IMPからの連絡によるデータ処理

サブ・タスクを発生させ処理をする。各ハイレベル・プロトコルの処理中には相手NCPとのデータのやりとりが頻繁に行なわれる。この処理が終了した場合完了情報が相手NCPに連絡される。

e IMPの状態を知らせるデータの処理

HITACに接続されているIMPの状態が、H-NCPに知らされると、フロー管理がその状態を確認し、データの内容によって各種の処理をおこなう。

(2) ネットワーク・プロセスからの連絡によるデータ処理

HITAC内にあるネットワーク・プロセスが、H-NCPに対して何らかの連絡をする時データ管理あるいはプロセス管理がその窓口となってH-NCPが動作する。この時のH-NCPの動作を、データ管理からの連絡と、プロセス管理からの連絡に分けて説明する。

a データ管理からの連絡処理

HITAC内にあるネットワーク・プロセスが、IPCマクロを利用してH-NCPへ連絡してきた時、データ管理ではそのデータを受け取りH-NCPの各機能へ連絡する。データの種類としては、他HOSTのNCP又はH-NCPに対してのネットワーク・プロセスの起動、終了要求のデータ、コネクションに関するデータ、他のネットワーク・プロセスへの転送データがある。データ管理によって受け取られたデータはその内容が解析され、イベント管理によって、イベント待ちであるかどうかの判断がされる。イベント待ちのデータであれば、イベント待ちテーブルに登録される。他のNCPに対するネットワーク・プロセスの起動、終了の要求であれば、フロー管理、DXC管理を通じて相手NCPにデータを転送する。H-NCPに対するネットワーク・プロセスの起動要求の場合には、プロセス管理に制御を渡す。コネクションに関するデータ及び他のネットワーク・プロセスへの転送データの場合には、コネクション管理に制御が渡り、コネクション管理で、コネクションテーブルの検索を行ない、フロー管理、DXC管理を通じて他NCPにデータが転送される。コネクション管理、フロー管理では、他NCPに転送する際に必要とされるコマンドを調べ、コマンド管理に依頼してコマンド群を作成し、DXC管理では転送の制御を行なう。以上の過程を図4-35に示す。

b プロセス管理からの連絡処理

H-NCPが起動したネットワーク・プロセスがHITACのコンソール装置に対して何らかの連絡をした時、プロセス管理でその連絡メッセージを知る事ができる。このメッセージに対する応答が必要とされる時には応答をする事も可能である。これらの情報は、H-NCP内でリソース管理、プロセス管理のために利用される。

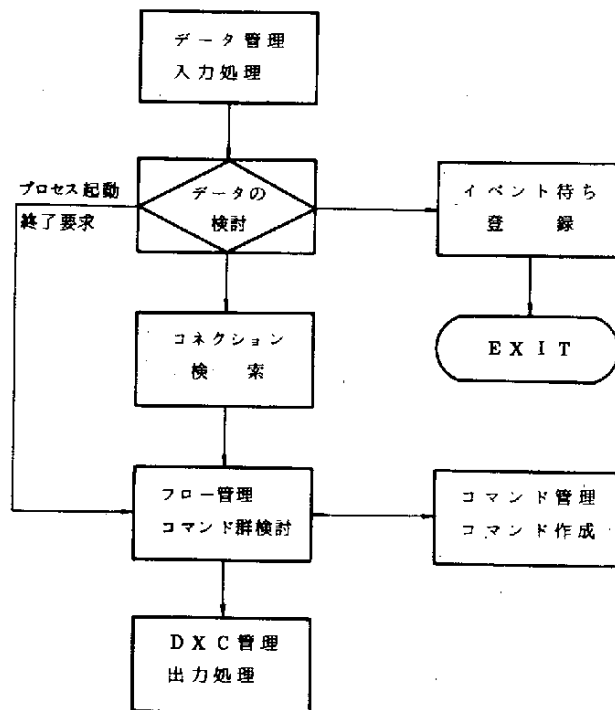


図 4-35 データ管理からのデータの処理

4.3.3 H-NCP機能の実現方法

H-NCPの機能実現のために、EDOS-MSOのエグゼクティブと連絡をとる事が必要となる事は前項で述べた。ここでH-NCPの主な機能とエグゼクティブ連絡マクロについてさらに詳しく述べる。

エグゼクティブ連絡マクロのうちで特に機能と密接に関係があるのは、DXC装置制御マクロ、プログラム間連絡マクロ、キー・イン/アウト・シミュレーション・マクロ、マルチ・タスクサポート・マクロの4種類であり、H-NCPはこのマクロを利用して、IMPとの連絡、HITAC内のネットワーク・プロセスとの会話、ネットワーク・プロセスの起動、ハイレベル・プロトコルの処理等の機能を実現する。以下にH-NCPの機能別にその処理内容、エグゼクティブ連絡マクロとの関係、機能実現のために必要とされる情報等について述べる。

A プロセス間コミュニケーション

H-NCPとHITAC内のネットワーク・プロセスとの連絡は、プログラム間連絡マクロ(IPCマクロ)を用いて行なわれ、連絡の始まりは、ネットワーク・プロセスがH-NCPに対してP C Write (ネットワーク・プロセスからH-NCPへのデータ転送)要求を発した時である。すなわち、H-NCPとネットワーク・プロセスとの会話は、ネットワーク・プロセスからの積極的アクションによって確立される。

H-NCPは、起動した時の処理として、ネットワーク・プロセスからの連絡が受け付け可能な状態に初期設定され、ネットワーク・プロセスがH-NCPに始めてアクションをするためのユーザ・レベルのマクロは、OPENマクロ、LOADPマクロ、LISTNマクロ、POSTマクロ、FDRマクロの5種類のマクロである。これらのマクロが、ネットワーク・プロセス内で実行された時、ネットワーク・プロセスに追加されているディスパッチ・モジュールがH-NCPに連絡を取り、その後は、両者間で自由にデータの受け渡しが可能となる。図4-36に会話の確立例を示す。

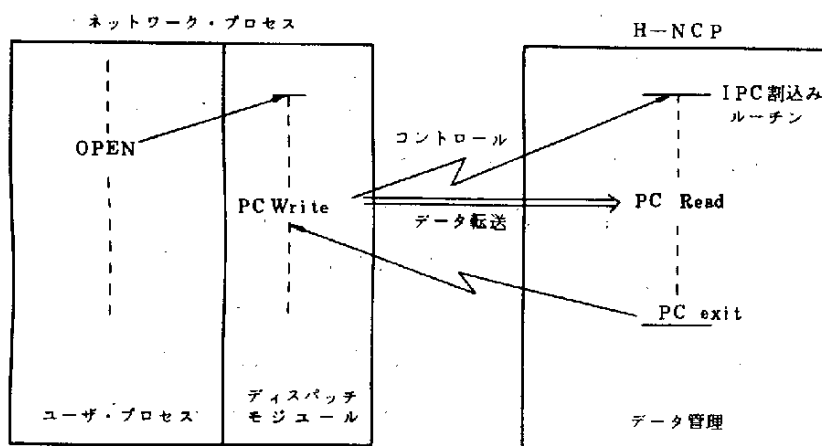


図4-36 IPCマクロによるプロセス間コミュニケーションの確立

以下にIPCマクロの概念、H-NCPのIPC処理、転送情報について述べる。

(1) IPCマクロ

IPC (Inter Program Communication) は、緊急のデータ転送、データの保護を行ないたい時、データを送信したい相手のプログラムに割込み信号を送り、相手の受信体制と同期をとってデータを転送する方式である。

IPCマクロは6種類のマクロで構成され、メイン・タスクのみがこれらのマクロを出す事ができる。各マクロの機能を以下に示す。

- a PCSET: エグゼクティブに対してプログラム間通信コントロール・ブロック (PCCB) のアドレスを知らせる。このマクロはデータ連絡を行なうとき両方のプログラムで出されなければならない。
- b PCWRT: 転送したいデータが生じた時出すマクロで、PCCBにセットされている情報に従って他のプログラムにデータを転送する。このマクロを実行すると相手のプログラムにIPC割込みを起す。
- c PCRD: IPC割込み処理ルーチンで出し、このマクロが実行されて始めてデータの

転送が行なわれる。PCRD マクロが完了すると割込みをかけてきた相手のプログラムが PCWTを出していれば待ち状態を解除する。

d PCWT: PCWRT マクロの後で出され、データを受取るプログラムとの同期をとる。データ転送が実際に終了するまでは送信側のプログラムはこのマクロで待たされる。

e PCXT: このマクロにより IPC 割込み処理ルーチンから出て割込まれたものとのアドレスへコントロールを戻す。

f PCCB: このマクロは PCCB を定義するエリア定義マクロであり、このマクロによってプログラム間通信コントロール・ブロック (48 バイト) が作成される。

(2) H-NCP の IPC 処理

H-NCP の IPC 処理はデータ管理に含まれ次の通り行なわれる。

a H-NCP が起動されると PCSET マクロを実行し、IPC 割込み処理ルーチンのアドレスをエグゼクティブに知らせ、IPC 処理を可能な状態にする。

b ネットワーク・プロセスからのデータ転送要求が、割込みをとまって非同期に生じると、H-NCP 内の IPC 割込み処理ルーチンに制御がわたされる。この割込み処理ルーチンで PCRD マクロを実行し、データを読み込む。読込んだデータは、初めてのアクションであるかどうかの判断及び登録、コマンドの判別等が行なわれ、コネクション管理に制御が渡される。その後 PCXT マクロを実行して、IPC 割込み処理ルーチンから抜け出る。

c コネクション管理からの要求により、ネットワーク・プロセスとの会話が必要となると、連絡すべきネットワーク・プロセスからすでにアクションがあったか否かを調べる。アクションがなされている事を確認するとネットワーク・プロセスに PCWRT マクロを実行する。アクションがなされていない場合には、アクション待ちの状態となる。PCWRT マクロの実行完了は、PCWT マクロを用いて同期をとる。

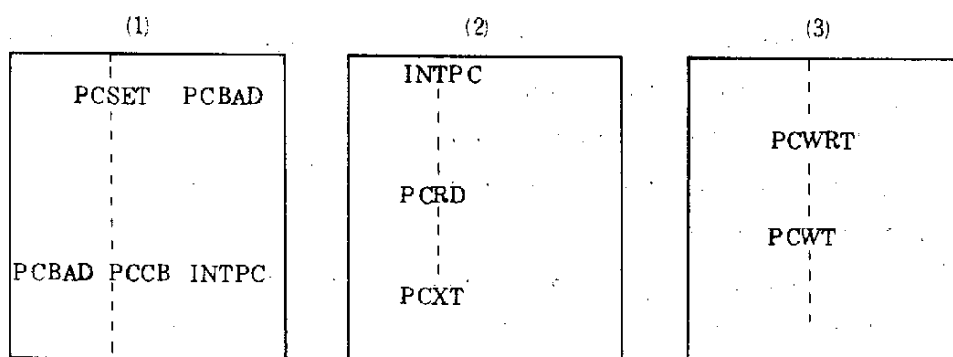


図 4-37 H-NCP の IPC 処理

(3) 転送データ

H-NCPとネットワーク・プロセスとのデータの転送は、1回の転送長が最大256バイト迄であるので、それ以上のデータに関しては複数回に分けて転送される。H-NCPとネットワーク・プロセス間で転送されるデータ長の最大は4608バイトまでとし、このデータを送り終るまでに18回の転送が必要とされる。1回目のデータはヘッダ部を持ち図4-38に示されるフォーマットを持つが、2回目以降のデータはデータ本文だけで構成される。

データ長	CC	COP	データ本文
4	2	2	248以下

図4-38 転送データ・フォーマット

a データ長(4バイト)

全データ長をバイトで表わし8~4608迄許される。

b CC(2バイト)

H-NCPからネットワーク・プロセス又は、ネットワーク・プロセスからH-NCPへのコマンドの種類を示すコマンド・コードを表わす。

c OP(2バイト)

コマンド・コードに附加される情報を示すオプション・エリアで、コマンド・オプションを表わす。

d データ本文

H-NCPとネットワーク・プロセスの間で授受するデータの本文で、1回目は248バイト以下とし、2回目以降は256バイト以下とする。

B プロセス管理

他HOSTのNCP又はHITAC内のネットワーク・プロセスの要求によって、H-NCPのプロセス管理はネットワーク・プロセスの生成、消去、及び監視の機能を果たす。この機能は、キー・イン/アウト・シミュレーション・マクロを用いて行なわれ、キー・イン/アウト・シミュレーションによって生成されたネットワーク・プロセスはH-NCPと親子関係を持ち、H-NCPによって制御される。

H-NCPは起動された時、キー・イン/アウト・シミュレーション機能の準備を行ない、HITAC内にネットワーク・プロセスを生成させたり消去させる事が可能な状態におかれる。実際の生成、消去は、コネクション管理から送られてきたデータをもとにキー・イン/アウト・シミュレーションを実行する事によってなされる。

ネットワーク・プロセスが他のネットワーク・プロセスの生成を要求するネットワーク・ユーザ用のマクロは LOADP マクロであり、消去を要求するマクロは DELP マクロである。このいずれかのマクロがネットワーク・プロセスにおいて実行されると、H-NCP のプロセス管理に、プロセスの生成、消去に必要なデータが渡され、プロセス管理は必要に応じて、キー・イン/アウト・シミュレーションを実行する。又このプロセス管理は、キー・イン/アウト・シミュレーション用の割込み処理ルーチンを持ち、生成したネットワーク・プロセスとの会話をエグゼクティブを通じて可能にしている。以下にキー・イン/アウト・シミュレーション・マクロの振まい及び、H-NCP のプロセスの発生方法、必要とされる情報について述べる。

(1) キー・イン/アウト・シミュレーション・マクロ

キー・イン/アウト・シミュレーションは、あるプログラムのコンソール・オペレーションを他のプログラムで制御しようとする機能を果す。キー・イン/アウト・シミュレーションを行なうプログラムをイニシエータと呼びイニシエータによってコントロールされるプログラムをサボーディネータと呼ぶ。サボーディネータは必ずイニシエータによってロードされなければならない。

コンソール・オペレータによるコマンドをイニシエータが行なうと、エグゼクティブと各種の連絡をとる事が可能であり、各種のオペレータ・コマンドをイニシエータから模擬的に実行したり、模擬オペレータ・コマンドによってロードされたサボーディネータから出されたメッセージを受け取り、それに対する応答を模擬的に実行することができる。

この機能を実行するためにイニシエータ中で、エグゼクティブと連絡するためのマクロは次の4種類である。

- a STKEY: このマクロはエグゼクティブに対しキー・イン/アウト・シミュレーション・コントロール・ブロック (KEYCB) のアドレスを知らせ、イニシエータとなることを宣言する。
- b KEYSM: エグゼクティブに対して、オペレータ・コマンドのシミュレーションを要求する。メッセージ・エリアにメッセージをセットしておく事が必要である。
- c KEYXT: イニシエータのキー・イン/アウト・シミュレーション割込みルーチン (Key-Sim 割込み処理ルーチン) から出て、割込まれたアドレスに制御をもどす。
- d KEYCB: キー・イン/アウト・シミュレーション・コントロール・ブロックを定義する。この定義によって24バイトのエリアが確保される。

(2) ネットワーク・プロセスの発生、消去方法

H-NCP は、HITAC 内のネットワーク・プロセス又は他のNCPの要求によって従属プロセスを発生させ、そのプロセスをH-NCPの管理下におくが、この時に必要なデータ処理を図4-39に示す。

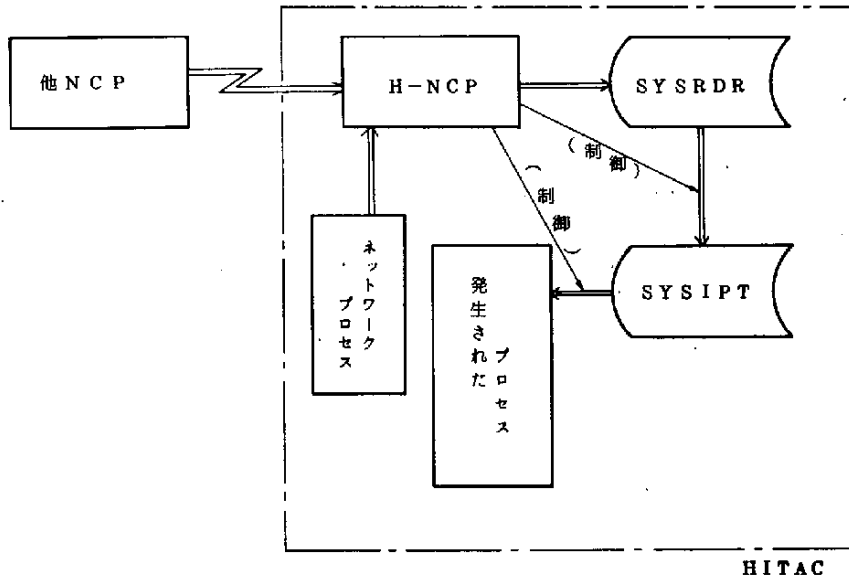


図 4-39 プロセス発生概念

HITAC 内のネットワーク・プロセス又は、他のNCPから送られて来たジョブ起動に関する情報は、H-NCP内でジョブ・コントロール・ステートメント（//JOBから//ENDまで）に変換されて、非常駐入力リーダーSYSRDR（ディスク）に蓄えられる。その後、非常駐入力リーダーの起動を行ない入力ジョブ・ストリーム（SYSIPT）に登録され、コア上にロードされる。ここで図4-39における（制御）は全て、キー・イン／アウト・シミュレーションによる模擬オペレータ・コマンドによってなされる。

ネットワーク・プロセスの発生要求は非同期になされ、H-NCPが判断して正当なものであればプロセスの発生を行なう。この判断形態と、H-NCPが発する模擬オペレータ・コマンドを図4-40に示す。

ネットワーク・プロセスの消去は、そのプロセスが実行中である事を確認した後、キー・イン／アウト・シミュレーションにより、ジョブの強制終了を指示する事によってなされる。

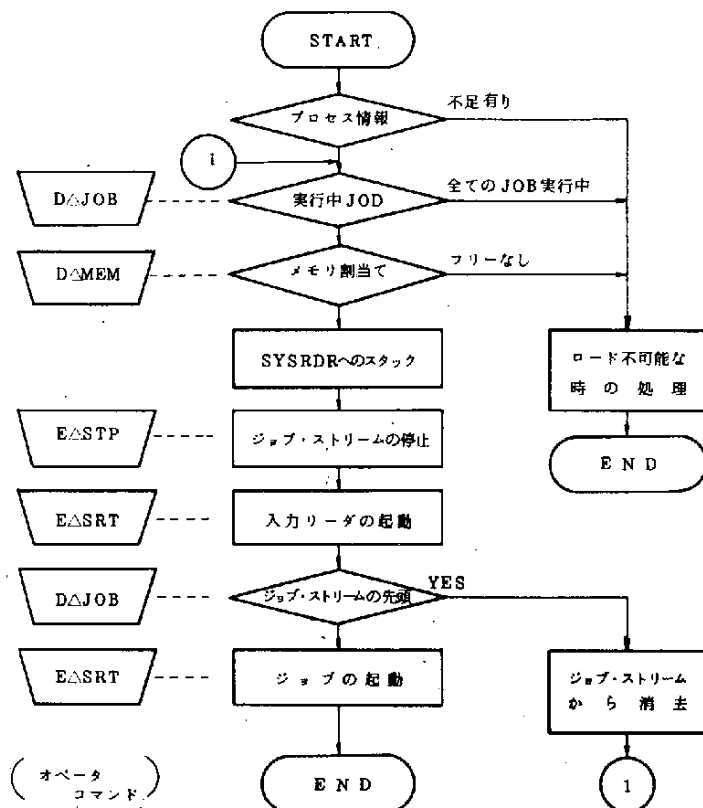


図 4-40 プロセスの発生方法

(3) プロセス発生、消去に必要とされる情報

ネットワーク・プロセスを発生する時に必要とされる情報は次の通りである。

a ジョブの情報

ジョブ名称、ジョブ・プライオリティ、開始時間、メモリ・サイズ等の情報で、JOB ステートメントを作成するのに用いられる。ジョブの情報の指定がなされない時にはH-NCP がインプリシットに情報をセットする。

b プロセス情報

発生するプロセスの名称、プロセスのレジデンスについての情報であり、この情報は必ず必要とされる。発生するプロセスの名称から EXEC ステートメントを作成し、プロセスのレジデンス情報（ボリューム・シリアル番号、ファイルID）から JOBLIB ステートメントを作成する。

c ファイル情報

プロセスが必要とするファイルの情報で、DTF ステートメントの作成に用いられる。その内容はファイル定義名、デバイス・タイプ、ファイル・タイプ、ファイルの存在場所、

ファイル属性、プロセス終了後の処理等である。

d データ

プロセスの実行に必要とされるインプット・データ

また、ネットワーク・プロセスを消去する時に必要とされる情報は、プロセス名称だけである。

C HOST-IMPインターフェース

HITACのH-NCPとIMP間のデータの授受は、セクタ・チャンネルを介して行なわれ、H-NCPはDXC装置制御マクロによってIMPとの間のデータ転送の制御を行なう。

DXC装置制御マクロはH-NCPのDXC管理で使用され、H-NCPが起動された時点でエグゼクティブに連絡をとり、データ交換制御装置からの割込みの受けつけ準備を行なう。この準備処理によってIMPとのメッセージ交換が可能となる。

H-NCPとIMP間のデータ転送は一時点では一方向のみであり、EDOS-MSOのコントロール・システムはDXC(チャンネル・アダプタ)を1つの入出力装置として取り扱うが、一般のノン・ランダム・アクセス装置の扱いと異なる点は、割込み処理ルーチンが必要となる事である。

以下に、DXC装置制御マクロ、H-NCPのDXC処理、データ転送上の留意点について述べる。

(1) DXC装置制御マクロ

DXC装置と一般のノン・ランダム・アクセス装置との違いは、DXC装置の場合には、HITAC内の演算とは非同期に他の処理装置(IMP)側でDXC装置への書込みコマンドが発せられ、コマンドの実行によって、HITAC内に割込みがかかってくる事である。そのためHITACでは割込み処理ルーチンを登録しておかなければならない。この割込み処理ルーチンはH-NCPの中にあり、その指定はDXC用のCCB(Command Control Block)を作成する事によってなされる。DXC用のCCBは、一般の入出力装置用のCCBに16バイトのエクステンションが付加されたものである。

次にDXC装置制御のためのオプション・マクロについて説明する。

- a STDXC: データ交換制御装置からの割込み処理の準備を行なう。このマクロの実行によりエグゼクティブにDXC用のCCBアドレスが知らされ、他の処理装置からのDXC経由のデータを受け取る事が可能になる。
- b DXCXT: このマクロはSTDXCマクロで準備されたDXC割込み処理ルーチンから割込まれた元のルーチンに制御を戻すために、DXC割込み処理ルーチンの出口で用いられる。

図4-41にDXCマクロの使用例を示す。

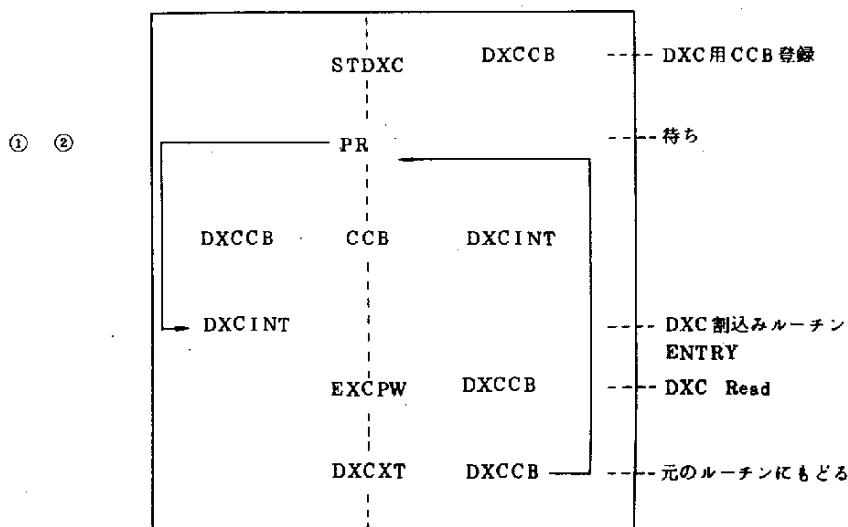


図 4-41 DXC マクロ使用例

(2) H-NCPのDXC処理

H-NCPがIMPとメッセージ交要をする時のDXC処理手順は次の通りである。

a H-NCPの起動時

H-NCPが起動されると、先ずSTDXCマクロを実行し、エグゼクティブにDXC割込み処理ルーチンの登録を行なう。次にNCP-IMP間の初期状態処理を行ない、IMPとの間のメッセージ交換を可能とする。

b IMPにメッセージを転送する時

データの正規転送手順により、IMPに対しWriteコマンドを出す。

c IMPからのメッセージが到着した時

STDXCマクロで登録したDXC割込み処理ルーチンに制御が渡され、そのルーチン内で正規転送手順によりReadコマンドを出す。メッセージが複数セグメントで構成されている時には、メッセージの読み込みが終了すると同時に再びReadコマンドを出す。

d メッセージが衝突した時

アダプタの両サイドから同時にコマンドが出された時、衝突手順に従って処理を行なう。

e 緊急事態発生の時

ハードウェア、電源断により相手と交信することが不可能になったり、ハードウェア、ソフトウェアのいずれかの原因で一時的に交信不能となった場合には、緊急処理手順に従って処理を行なう。

(3) データ転送上の留意点

H-NCPとIMPはそれぞれ全く非同期並行に進行するための両方ではほぼ同時にDXC Write要求を出す場合がある。この場合に、両方からのWrite命令が衝突するという事が考えられる。この種の処理は、データ転送上必ず考えなければならない。両方からのWrite命令が衝突した時には、誤応答 (incompatible) のステータス情報がH-NCPにかえされるので、DXCにWrite命令を出したDXC管理は、そのステータス情報を見て、誤応答が生じていれば、自分の側の要求を待ち合わせるか、再度要求を出すかを判断し、処理を行なう。待ち合わせる場合には、Read コマンドを出し、相手のWriteコマンドを受け付け、その処理を行なった後Writeコマンドを出す。自分側の要求に優先権を持たせるか、相手側のコマンドに優先権を持たせるかは、その時出されたコマンドの種類及び、直前に出されたコマンドの種類によって決定される。

D コンピュータ・リソースの把握

H-NCPはHITACの使用状況を把握し、他のNCPに連絡することができる。このコンピュータ・リソースの把握はリソース管理が、キー・イン/アウト・シミュレーションを用いてEDOS-MSOのコントロール・システムと会話をする事によって行なう。又、特殊な情報については、他のエグゼクティブ・連絡マクロを用いて把握する事もある。

H-NCPのリソース管理が入手する情報を以下に述べる。

(1) ジョブの個数

HITAC 8450において同時に実行可能なユーザ・ジョブの個数は最大16個迄であるが、ネットワークの実施時に実行可能なジョブの個数は、ステージ・ジョブが6個 (A0, B0, C0, R1, B1, C1) と、オペレータ・パーティション・ジョブが2個 (N0, N1) である。このうちR1ジョブとしてH-NCPが位置するので、H-NCPが管理できるジョブの個数は7個である。これらのジョブは全てバッチ・ジョブであり、H-NCPによって起動されたジョブと、ローカルジョブに分けられる。

実行中のジョブ数、実行待ちのジョブ数及び特定のジョブの状態は、キー・イン/アウト・シミュレーションで、DΔJOBの模擬オペレータ・コマンドを出す事によって知る事ができる。

(2) コア領域の使用状況

システムのメモリ割当て状況、現在実行中のジョブが使用しているコア領域の利用度を求める。コア領域はパーティションに分割されているので、各実行中のジョブはパーティションで管理されている。ステージ・パーティションは各クラスに分割されているので、各ステージ毎のコアの利用状況の把握も可能である。又、オペレータ・パーティション・プログラムで使用しているエリアのサイズ、指定ジョブ・クラスで使用しているエリアのサイズ、未使用エリアのサイズ等を知る事もできる。このコア領域の状況を知るためには、

キー・イン/アウト・シミュレーションで D Δ MEM 模擬オペレータ・コマンドを用いる。

(3) ファイルの使用状況

ファイルの使用状況を知るためには、次の2つの方法がある。

- ① 各ジョブで割当てている入出力装置の確認
- ② ディスク・ファイルの未割当てエリアの検索

磁気テープファイルは、あるジョブに割当てると、他のジョブが使用できないので①の方法で使用状況を把握する。

ディスク・ファイルの使用状況で特に必要となるのは、空きエリアの状態を知る事であるが、これは②の方法で検査する。

これらの状況把握には、キー・イン/アウト・シミュレーションにより、それぞれ D Δ DEV, D Δ FEX 模擬オペレータ・コマンドを用いる。

又、ディスク・ファイルの割当て状況を把握したい時には、H-NCPのリソース管理が指定されたボリュームの VTOC を検索する事によって行なう。

(4) その他の状況

その他に必要とされる情報は、キー・イン/アウト・シミュレーションによって模擬オペレータ・コマンドを用いて得たり(システムの登録日付と時間、オンライン状態のボリューム等)、他のエグゼクティブ連絡マクロを用いて得る(オンライン・カタログの検査、エグゼクティブ・ストレージ・エリアの参照、モニタ・テーブル、タスク・コントロール・テーブルの検索)事が可能である。

E 高位のプロトコル処理

ネットワークの特定の機能を効果的に利用しようとするネットワーク・ユーザは、機能指向的な高位のプロトコルを使う事ができる。

ネットワーク・プロセスの要求による高位のプロトコルの処理は、H-NCPがプロセスを発生して、コネクションをその発生したプロセスに渡す方法と、H-NCP内でサブ・タスクを発生して処理を行なう方法のいずれの方法でも可能である。前者は一般のネットワーク・プロセスの扱いと同様であり、たとえばファイル・トランスファー・プロトコルのようなものは1つの独立したプロセスとしてこの形態を持つ。後者は、H-NCPの下でサブ・タスクとして動作する高位のプロトコルでグラフィック・コミュニケーション・プロトコルはこの形態を持つ。これらの高位のプロトコル処理は、1つのプロセス又はサブ・タスクとして OPEN-ENDED に処理されるので、各種のファンクション・オリエンティッドなプロトコルをプロセスまたはサブ・タスクとして作成する事が可能である。

プロセスの形態を持つ高位のプロトコルの処理は、H-NCPが起動するネットワーク・プロセスであり、一般のプロセス間コミュニケーションの扱いをうける。ここではサブ・タスク

の形態を持つ高位のプロトコル処理について述べる。高位のプロトコル処理用のサブ・タスクは、H-NCPのタスク管理によって起動され、制御される。高位のプロトコル処理が必要となった時点で、NCPタスク管理はサブ・タスクの起動を行ない、サブ・タスクでは要求された処理を行なう。サブ・タスクの処理が終了すると、NCPタスク管理に制御がわたり、NCPタスク管理では、そのサブ・タスクの必要性を考慮に入れて待ち状態にするか停止状態にするかを決定する。このようにサブ・タスクによる高位のプロトコル処理は常にNCPタスク管理に制御される。

以下にサブ・タスクの形態を持つグラフィック・コミュニケーション・プロトコルについてさらに詳しく述べる。

HITACにおける機能指向的なプロトコルとして先ず考えられるのは、グラフィック・ディスプレイの利用に関するプロトコルである。グラフィック・ディスプレイは、インタラクティブ・システムとして各種のアプリケーション・システムに使われていて、特殊入出力装置(CRT, ライトペン等)を用いるマン・マシン・コミュニケーションの接点となっている。すなわち図形出力装置及びインタラクティブ入力端末としてその効果を発揮する。

このグラフィック・ディスプレイのネットワーク上での利用形態を考えると、他のHOSTに存在するネットワーク・プロセス又はTSS端末から容易にHITACのCRTが使え端末間の交信を可能にする事が要求される。またCRTから他のHOSTのネットワーク・プロセスを起動したり、TSS端末を利用する事も可能である事が望ましい。これらの事を考えると、グラフィック・ディスプレイ利用の為の高位のプロトコルが必要となり、グラフィック・コミュニケーション・プロトコルが考えられた。

グラフィック・コミュニケーション・プロトコルの機能は次の通りである。

(1) H-NCPとの連絡

H-NCPと連絡をとる事によって他HOSTのNCPとの連絡も可能となり、ネットワーク・プロセス間のコミュニケーションが可能となる。

(2) グラフィック・システムとの連絡

グラフィック・システムが持つ特殊入出力装置との連絡を行なう。この機能はTCP (Terminal Control Portion)を含んでいる。

(3) H-NCPからのメッセージ解析

H-NCPを通して送られてくるメッセージは、グラフィック・システムの使用要求、表示用図形データ、インタラクティブ・レスポンスメッセージ等があるのでそれらのメッセージ解析が必要となる。

(4) グラフィック・システムからのメッセージ解析

グラフィック・システムが端末として動作する時、メッセージが送られてくると、他HOST

のNCPに対するメッセージか、HITAC内のネットワーク・プロセスに対するメッセージかを判断しなければならない。また、ネットワーク・プロセスの起動、TSSの利用要求メッセージの解析を行ない、それに対応するコネクションの確立も行なう。

ネットワーク・ユーザがCRTを用いて他HOSTのプロセス又はTSS端末利用を考えると、次の様な処理手順が実施される。

- ① CRT(ライトペン、キー・ボード)より、プロセス起動、TSS端末利用の要求が出され、グラフィック・システムよりH-NCPに連絡が生じる。
- ② H-NCPではNCPタスク・コントロールがグラフィック・コミュニケーション・プロトコル処理用のサブ・タスクを起動する。
- ③ サブ・タスクがグラフィック・システムからのメッセージを解析し、プロセス起動コマンド又はTSS利用コマンドを作成し、NCPタスク管理に制御を返す。
- ④ H-NCPは他のHOSTのNCPと会話を行なう。
- ⑤ H-NCPで受け取られたメッセージが、起動したプロセス又は、TSS端末であると、タスク管理よりグラフィック・コミュニケーション処理用のサブ・タスクに制御が渡る。
- ⑥ サブ・タスクでは、図形データ、レスポンスメッセージ等を解析し、CRTに対して連絡を行なう。

このフロー・コントロールを図4-42に示す。

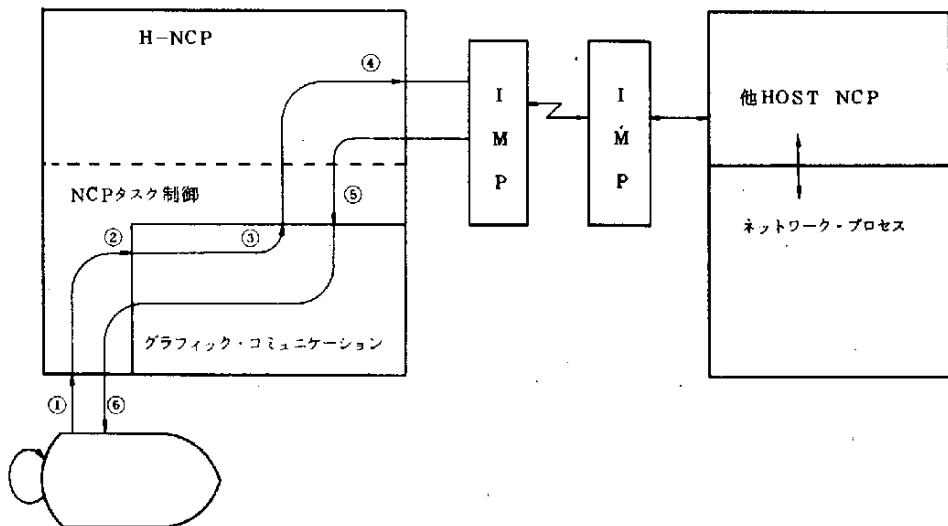


図4-42 グラフィック・コミュニケーション処理

4.3.4 ネットワーク・プロセス

HITAC におけるネットワーク・プロセス (ユーザ・ジョブ) は、バッチ・ジョブの形態を持ち、H-NCP と連絡をとりながらネットワークの資源を利用する事ができる。

ネットワーク・プロセスが、H-NCP を通じてネットワークへのアクセスを要求する時には、各種のネットワーク・ユーザ用のマクロ (ユーザレベル・プロトコル) を利用する事ができる。このユーザ用のマクロは、ネットワーク・プロセスの作成時に、アセンブラ・ソース・コーディング上に記述する。

ネットワーク・ユーザ用のマクロを用いたプロセスは、ネットワーク・プロセスとしての権限を持ち、H-NCP との会話とファイル管理のための、ディスパッチ・モジュールが附加される。このディスパッチ・モジュールとユーザの処理モジュール、H-NCP との関係は図 4-43 の通りである。

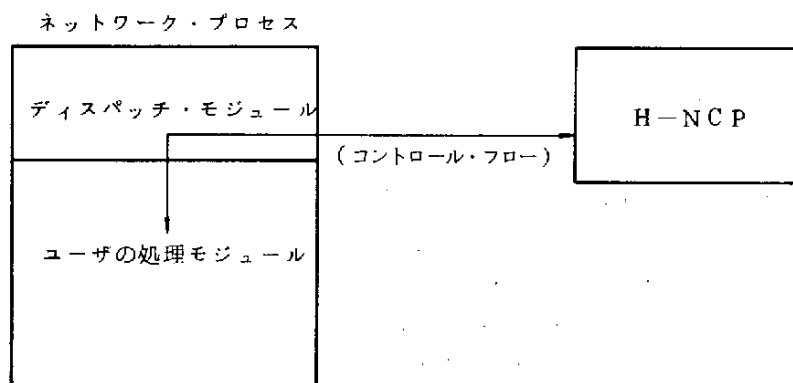


図 4-43 ネットワーク・プロセスと H-NCP

ネットワーク・プロセスが、ネットワーク資源を利用する時の形態として先ず考えられるのは、他のネットワーク・プロセスとの会話である。この会話は、ファイルの扱いの拡張された概念としてとらえられる。他の形態としては、他系 NCP, T I P との単独会話が考えられる。以下にネットワーク・プロセス間の会話、ディスパッチ・モジュール等について述べる。

A ネットワーク・プロセス間の会話

他のネットワーク・プロセスに対して何らかのアクセスを要求する HITAC 内のプロセスは、シーケンシャル・ファイルに与える D T F ステートメント (ジョブ・コントロール・ステートメントの一種) の記述方法によってネットワーク・プロセス間の会話を可能とする。この時ネ

ネットワーク・プロセスはDTFNT ファイル定義マクロによって、シーケンシャル・ファイルの定義を行なっている必要がある。

DTFNT ファイル定義マクロによって定義されたシーケンシャル・ファイルは、ネットワーク・プロセス間の会話に使用する事が可能であるが、それ以外に、DTFステートメントの記述方法によっては、HITAC のシーケンシャル・ファイル(ローカル・ファイル)として使用する事もできる。

(1) DTFNT ファイル定義マクロ

DTFNT ファイル定義マクロは、ロジカルファイルを定義し、シーケンシャル・デバイス及びランダム・デバイスに適用する事が可能であり、処理のタイプはシリアル処理である。このマクロによって定義されたファイルはネットワーク・プロセス間の会話ファイルとして適用される。すなわち既存のDTFSR ファイル定義マクロで指定されたファイルに、ネットワーク利用の権限を与えたものであり、ネットワーク・プロセス間の会話に必要とされる情報(PORT-ID, ファイル情報)は、DTFNT ファイル定義マクロで定義されるか、DTFステートメントで与えられる。

DTFNT ファイル定義マクロによって作成される制御ブロックEDCB (Extended Data access Control Block)は、既存のDTFSR ファイル定義マクロによって作成される制御ブロックにネットワーク・プロセス間の会話の為の制御ブロックNFCB (Network File Control Block)を附加したものである。図4-44にEDCBを示す。

F ~ F					
FCB (File Control Block) 既存のDTFSRで展開されるコントロール ブロック					
MY PORT-ID1					
MY PORT-ID2					
YOUR PORT-ID					
OPEN	LIS TN	CONN CT	FDR	PASS WORD	
EVENT CONTROL					R.LENGTH
FILE ID					
		DEV	FIL	VSN	
STA TUS	END	BLOCK	RECORD	OPTION	

図4-44 EDCB

(2) DTFステートメント

ネットワークを利用しようとするユーザは、ファイルと装置の関係を定義するDTFステートメントに対して、ネットワーク特有の記述を行なう。但し、DTFNTファイル定義マクロで定義したファイルをローカル・ファイルとして扱う場合には今迄の通りでよい。

ネットワークを利用する時のDTFステートメントは次の記述による。図4-45にその例を示す。

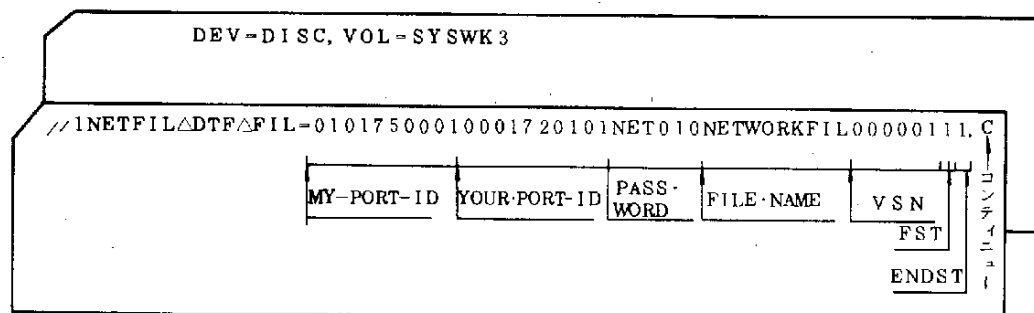


図4-45 DTFステートメント例

// DTF 名称△DTF△FIL =ネットワーク指定, DEV=DISC, VOL=VSN

DTF 名称: DTFNTファイル定義マクロで指定したファイル名称(PORT名称)で、
7文字以内で指定する。

DEV = DISC: この項は必ず指定

VOL = VSN : VSNにオンライン状態になっているディスクのボリューム・シリアル番号を6文字で指定する。

FIL=ネットワーク指定: プロセス間の会話のための情報を44文字で記述する。詳細は以下の通り。

1. MY-PORT-ID(必須)

16進10桁の数字で、自分のPORT-IDを指定する。この時第1桁目は0~9迄の数字とする。

2. YOUR-PORT-ID(必須)

16進10桁の数字で、相手のPORT-IDを指定する。この項を必要としない時は全て0を指定する。

3. PASS-WORD(必須)

6文字の英数字で自分のパスワードを記述する。

4. FILE-NAME(オプション)

10文字の英数字でファイル名を指定する。このファイル名は、他HOSTで使用されるファイルの名前である。

5. VSN

4で与えたファイル名の存在するボリューム・シリアル番号を、6桁の英数字で指定する。

6. FST

4のファイルのステータス情報を1～3の数字で与える。

7. ENDST

4のファイルの終了処理の方法を1～3の数字で与える。

B ディスパッチ・モジュール

ディスパッチ・モジュールは、HITAC内のユーザ処理モジュールに附加されて、ネットワーク・プロセスが用いるネットワーク・ファイルの管理及び、H-NCPとの会話の管理を行なう。

ディスパッチ・モジュールがネットワーク・プロセスに附加されるのは、ユーザの処理モジュールがネットワーク・ユーザ用の実行マクロを用いた時である。

ディスパッチ・モジュールの構成要素は、DTFNTファイル定義マクロで定義されたネットワーク・ファイルに対するアクセスを管理するファイル管理と、H-NCPとの会話を管理するIPC (Inter Program Communication) 管理からなり、相互に密接な関係を持つ。図4-46にディスパッチ・モジュールの構成を示す。

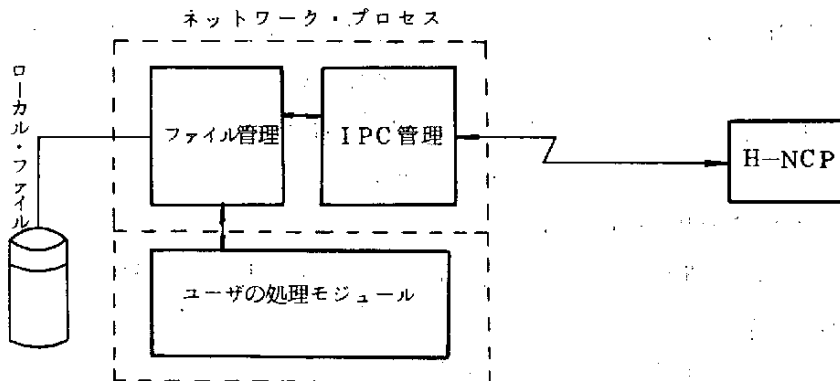


図4-46 ディスパッチ・モジュールの構成

ディスパッチ・モジュールの機能を要素別に示す。

(1) ファイル管理

DTFNTファイル定義マクロで定義されたファイルの管理及び、ファイルに関する入出力

マクロが実行された時の制御を行ない、次の機能を持つ。

a EDCB管理

ネットワーク・ファイル用の制御ブロックの管理は、大きくPORT-IDの管理、パスワードの管理、他のHOSTで使用するファイルの管理に分けられる。

b DTFステートメント解析

DTFNT ファイル定義マクロで定義されたファイルに対するDTFステートメントが、ネットワーク・ファイル用か、ローカル・ファイル用かを調べ、ネットワーク・ファイル用であれば、DTFステートメントのFIL項の内容解析及びEDCBへの登録を行なう。

c 入出力マクロの制御

シーケンシャル・ファイルに対するOPEN, CLOSE, GET, PUT 等の入出力マクロが、ネットワーク・アクセスを意味するかどうかを調べ、ネットワーク・アクセスであれば、IPC管理に制御を渡し、ローカル・ファイルに対するアクセスであればHITACのFCPに制御を渡す。

(2) IPC管理

HITAC 内のネットワーク・モニタであるH-NCPとの会話を管理し、ネットワーク・プロセス間の会話を可能にする。IPC管理は、割込み処理を含み、次の様な機能を持つ。

a IPC初期設定

IPC割込み処理ルーチンの登録を行なう。この登録は、ファイル管理の入出力マクロ制御により 初めて制御が渡された時と、ネットワークに特有なユーザ用のマクロを実行した時である。

b IPC割込み処理

H-NCPからネットワーク・プロセスに何らかの連絡があると、IPC割込み処理ルーチンが、データの読み込み及び解析を行なう。処理の終了後この割込み処理ルーチンから抜け出るが、この時のタイミングの制御も行なう。

IPC割込み処理ルーチンは、1プロセスに対して1エントリのみとされている。すなわちネットワーク・プロセスが、H-NCP以外のHITAC内のプロセスと会話しようとする場合でも、ネットワーク・プロセス中には1つのIPC割込み処理ルーチンが存在するだけである。すなわち、ネットワーク・プロセス中のIPC割込み処理ルーチンは、H-NCPとの会話に用いられるだけでなく、他のHITAC内のプロセスとの会話も可能としなければならない。この為ディスパッチ・モジュールのIPC割込み処理ルーチンは、他のプロセスとの会話に支障のないように、特殊なインターフェース部分を持っている。

c H-NCPへの連絡

ネットワーク・プロセスからH-NCPへ連絡する時の制御を行なう。この連絡はデー

タ転送を主目的とし、H-NCPへのIPC割込みとして実現される。

図4-47にディスパッチ・モジュール内のIPC処理手順を示す。

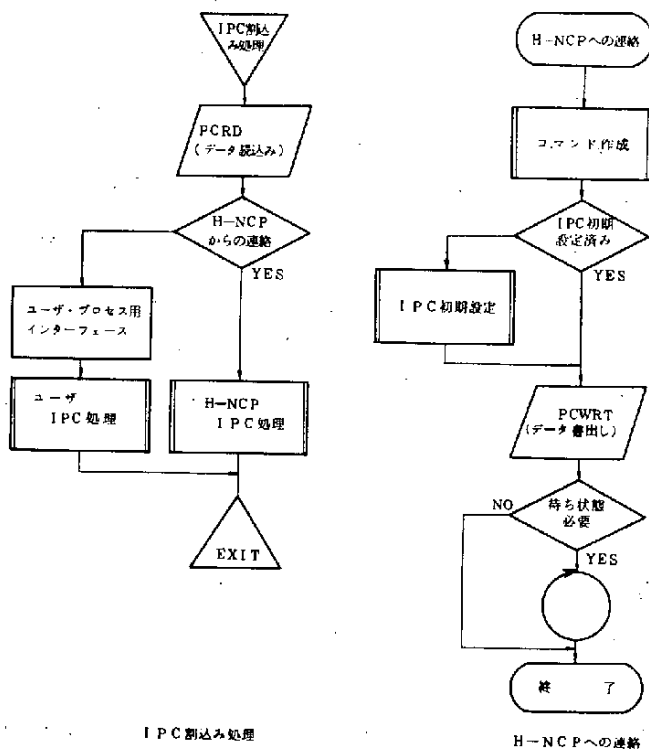


図4-47 IPC処理手順

C ユーザレベル・プロトコルの処理

ネットワーク・ユーザ用の基本的なマクロをユーザレベル・プロトコルとするが、HITAC内のネットワーク・プロセスは、以下に述べる各種のユーザ用のマクロを使う事によって容易にネットワーク資源を利用することができる。このユーザ用のマクロ及びその処理は次の通りである。

(1) コネクションとデータ授受に関するマクロ

他のネットワーク・プロセスとの会話に用いられるマクロで、ネットワーク・プロセス間のコネクションの確立、解放のためのOPEN、CLOSEマクロ、データの授受のためのGET、PUT及びGETN、PUTNマクロ、コネクション切換えのためのCONCTマクロからなる。これらのマクロのうち、OPEN、CLOSE、GET、PUTマクロは、一般のシーケンシャル・ファイルに使用されるマクロをそのまま適用し、ネットワーク・ファイルがローカル・ファイルとしても利用可能なようにしている。

ネットワーク・プロセスとの会話は、EDCBを参照して行なわれ、マクロの種類によって各種のデータを作成し、IPC管理によって、H-NCPへデータの転送を行なう。データ転送終了後、待ち状態を必要とするマクロ（OPEN, CLOSE, GET, PUT, CONCT）は、ディスパッチ・モジュール内で完了情報の到着まで待つ。待ち状態を必要としないマクロ（GETN, PUTN）では、次の命令に制御を渡す。

これらのマクロの処理方式を以下に示す。

- a ローカル・ファイルに対するアクセスか、ネットワーク・ファイルに対するアクセスかを調べる。ローカル・ファイルに対するアクセスの場合FCPに制御を渡す。ネットワーク・ファイルであれば以下の処理を行なう。
 - b 各マクロ毎のH-NCPへの連絡データを作成する。
 - c H-NCPへのデータ転送を行なう。
 - d 待ち状態が必要であれば、待ち状態を作成する。待ち状態が解かれるのは、H-NCPからの完了情報が知らされてきた時である。その後、実行されたマクロの次に制御を渡す。
- (2) 事象待ち・通知に関するマクロ

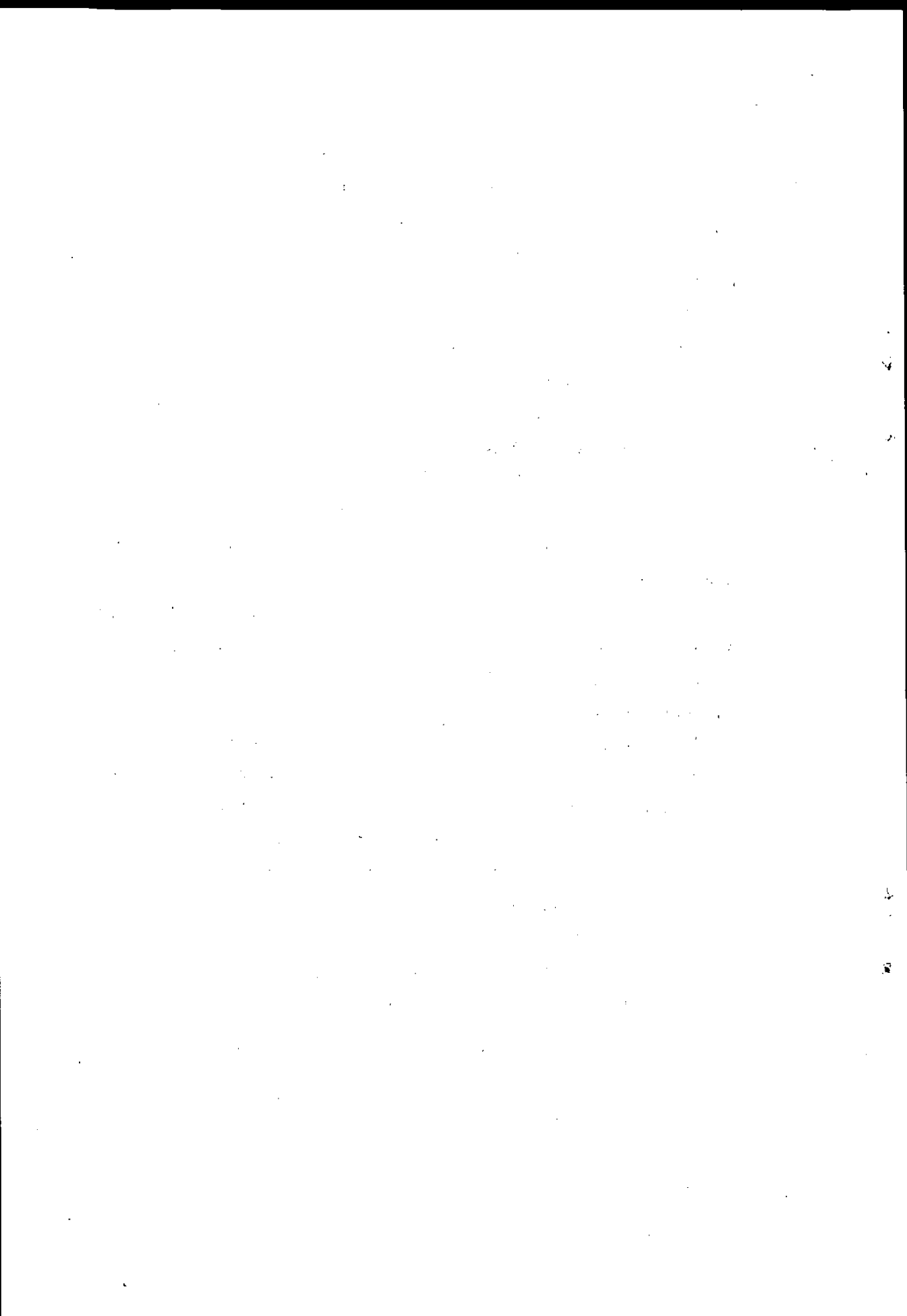
ネットワークにおいては複数個のネットワーク・プロセスが互にある種の関連を持ちながら処理を進める。この過程においてネットワーク・プロセス間の同期あるいは情報の通知が必要となる。この機能を満足させるために、事象待ち用のLISTN, NWAIT, AWAITマクロと、通知用のPOSTマクロがある。

これらのマクロは、2つのネットワーク・プロセス間で、POST-NWAIT/AWAITマクロによるイベント情報の相互連絡に用いられる他、LISTN/GETN/PUTN-NWAIT/AWAITの関係で、完了情報の確認にも使用される。すなわち、これらのマクロは、ネットワーク・プロセスのイベント・コントロールのために特に設定されたものである。

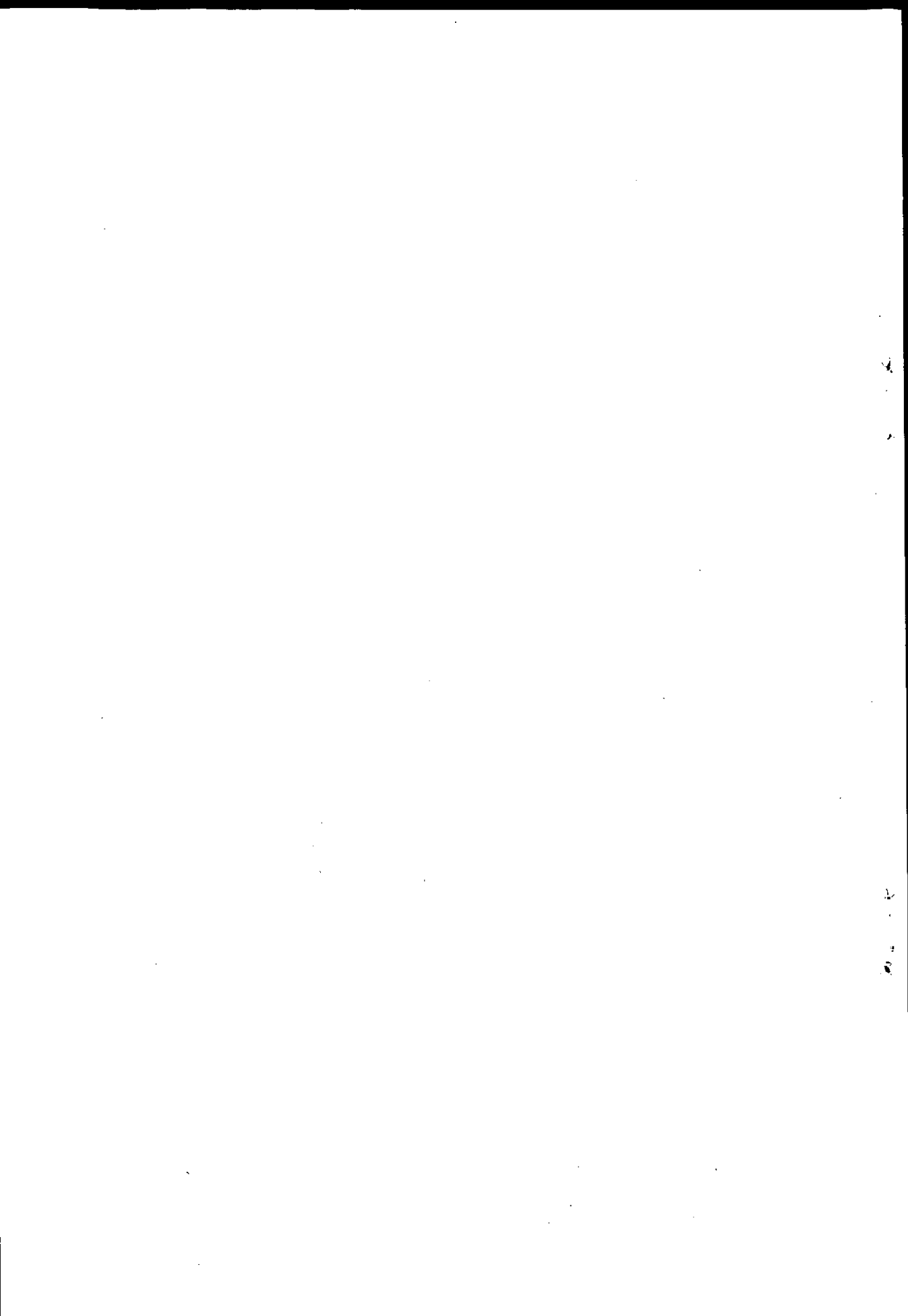
各マクロの処理において、事象待ちのためのマクロは、ディスパッチ・モジュール内で待ち状態を生じ、通知マクロは、待ち状態を生じない。

- (3) プロセスの起動・終了に関するマクロ

HITAC内のネットワーク・プロセスが他のプロセスを発生させたり、終了させるために、LOADP, DELPマクロがある。これらのマクロを実施しようとするネットワーク・プロセスは、プロセスの起動、終了に関する情報を用意して、H-NCPにプロセスの発生、終了の要求を出す。これらの情報の妥当性は、ディスパッチ・モジュールが判断を下す。ディスパッチ・モジュールでは、H-NCPに連絡した後待ち状態を作成し、完了情報を検査する。



参 考 文 献



1. ネットワーク開発概況

- 1) Abramson, N.
The ALOHA system--another alternative for computer communications. Proceedings of AFIPS 1970 Fall Joint Computer Conference, Vol.37, pp.281-285.
- 2) McKay, D.B. and Karp, D.P.
IBM Computer Network/440. Courant Computer Science Symposium 3, December 1970, pp.27-44.
- 3) Herzog, B.
MERIT Computer Network. Courant Computer Science Symposium 3, December 1970, pp.46-47.
- 4) Aupperle, E.
MERIT Computer Network: Hardware Considerations. Courant Computer Science Symposium 3, December 1970, pp.49-64.
- 5) Cocanower, A.
MERIT Computer Network: Software Considerations. Courant Computer Science Symposium 3, December 1970, pp.66-77.
- 6) Aupperle, E.M.
MERIT Network Re-examined. Comp. Con., 1973, pp.25-30.
- 7) Kirstein, P.T.
The future network development in Europe and the USA. International Computer State of the Art Report: Computer Networks Infotech Information, 1971, Vol.6, pp.347-363.
- 8) Somia, M.
THE approach of software problems in the SOC experimental computer network. ICCO, 1972, pp.390-396.
- 9) Roberts, L.G. and Wessler, B.D.
Computer network development to achieve resource sharing. SJCC, 1970.
- 10) Frank, H., Kahn, R.E. and Kleinrock, L.
Computer communication network design — Experience with theory and practice. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol.40, pp.255-270.

- 11) 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. Proceedings of AFIPS 1972 Fall Joint Computer Conference, Vol.41, pp.741-755.

2. ネットワーク・デザイン

○ IMPに関して

- 1) Heart, F.E., Kahn, R.E., Ornstein, S.M., Crowther, W.R. and Walden D.C.
The interface message processor for the ARPA computer network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol.36, pp.551-567.
- 2) Roberts, L.G.
Initial design for interface message processors for the ARPA computer network. BBN Report No.1763, 1969.
- 3) Carr, S., Crocker, S. and Cerf, V.
Specifications for the interconnection of a Host and an IMP. BBN Report No.1822, 1971 revision.

○ TIPに関して

- 4) Ornstein, S.M., Heart, F.E., Crowther, W.R., Rising, H.K., Russell, S.B. and Michel, A.
The terminal IMP for the ARPA computer network. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol. 40, pp.243-254.
- 5) Kahn, R.E.
Terminal access to the ARPA computer network. Courant Computer Symposium 3 Computer Networks. Courant Institute, New York, November 1970 — Proceedings to be published by Prentice Hall, Englewood Cliffs, New Jersey, in preparation.
- 6) User's guide to the terminal IMP.
Bolt Beranck and Newman Inc. Report No.2183.
The BBN terminal interface message processor BBN Report 2184.

○ メッセージ・スイッチング, パケットティング方式に関して

- 7) 岡田, 宮原, 真田, 手塚,
"メッセージ交換処理網におけるパケットィング方式について" 信学会, 交換研資,
(1971-10)

o フロー・コントロールに関して

- 8) Davies, D.W.
The control of congestion in packet-switching networks.
Trans. IEEE, Vol.COM-20, No.3, p.546, June 1972.
- 9) Kahn, R.E. and Crowther, W.R.
Flow control in a resource sharing computer network.
Proc. of the Second ACM IEEE Symposium on Problems in the
Optimization of Data Communications Systems, Palo Alto,
California, October 1971, pp.108-116.

o ルーティング (Routing に関して)

- 10) Frank, H. and Chou, W.
Routing in computer networks. Networks John Wiley, 1971,
Vol.1, No.2, pp.99-112.
- 11) Fultz G.L. and Kleinrock, L.
Adaptive routing techniques for store-and-forward computer
communication networks. 1971 International Conference on
Communication, Montreal, Canada, p.39-1, (June 1971).
- 12) Fultz, G.L.
Adaptive routing technique for message switching computer
communication networks. UCLA-ENG-7252, July 1972.

o ネットワーク・トポロジーに関して

- 13) Frank, H., Frisch, I.T. and Chou, W.
Topological considerations in the design of the ARPA com-
puter network. Proceedings of AFIPS 1970 Spring Joint
Computer Conference, Vol.36, pp.581-587.
- 14) Frank, H. and Chou, W.
Topological optimization of computer network. Proc. of
IEEE, Vol.60, No.11, November 1972.

o プロトコルに関して

- 15) Carr, S., Crocker, S. and Cerf, V.
Host/host protocol in the ARPA network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol.36, pp589-597.
- 16) Crocker, S., Heafner, J., Metcalfe, R. and Postel, J.
Function-oriented protocols for the ARPA network. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol.40, pp.271-280.
- 17) ARPA network current network protocols. August 1971
Available from the Network Information Center as NIC #7104 at Stanford Research Institute, Menlo Park, California 94025.
- 18) TELNET Protocol Specification, NIC #15372.
- 19) Mckenzie, A.
Host/host protocol for the ARPA network, NIC #8246.

3. パフォーマンス評価, 解析, シミュレーション

- 1) Kleinrock, L.
Analytic and simulation methods in computer network design. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol.36, pp.569-579.
- 2) 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, Palo Alto, California, October 1971, pp.39-45.
- 3) Kleinrock, L.
Models for computer networks. Proceedings of the International Conference on Communications, pp.21.9-21.16, June 1969.
- 4) Kleinrock, L.
Performance models and measurements of the ARPA computer network. ONLINE 72 Conference Proceedings, Vol.2.
- 5) Price, W.L.
Simulation of data transit networks. National Physical Laboratory, Com. Sci. 56, April 1972.

- 6) Kleinrock, L.
Comparison of solutions methods for computer network models.
Proc. of the Computers and Communications Conference, Rome,
New York, September 30 - October 2, 1969.

4. その他

- 1) Thomas, R.H.
A resource sharing executive for the ARPANET. National
Computer Conference 1973, pp.155-163.
- 2) Walden, D.C.
A system for interprocess communication in a resource
sharing computer network. C.ACM, April 1972, Vol.15, No.4,
PP.221-230.
- 3) Thomas, R. and Henderson, D.A.
McROSS—A multi-computer programming system. Proceedings
of AFIPS 1972 Spring Joint Computer Conference, Vol.40,
pp.281-294.
- 4) Kahn, R.E.
Resource-sharing computer communications networks. Pro-
ceedings of IEEE, Vol.60, No.11, November 1972.

—— 禁 無 断 転 載 ——

昭和 49 年 3 月 発行

発行所 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 3-11-11

TEL (407) 7316

