



計算機網向き OS, NOS の開発

——開発上の仕様とその一実現法——*

林 恒 俊**

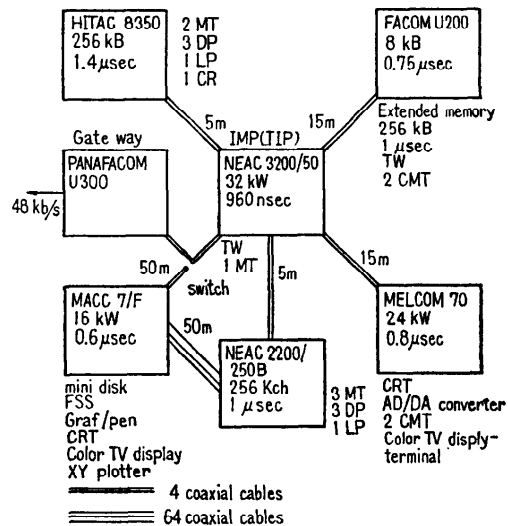
Abstract

A new operating system NOS is developed for the medium size host computer NEAC 2200/250B to promote efficient and useful resource sharing among users of the computer network KUIPNET. Some uniqueness is required for NOS because it is designed and developed to meet the requirements as an operating system of a host computer of a resource sharing computer network. The requirements for NOS by the KUIPNET users are relatively hard for this medium size computer. However, the careful design of NOS makes it possible to meet them. The design considerations and specifications of the NOS operating system are described in detail.

1. ま え が き

京都大学工学部情報工学教室において、高度な情報処理の研究のための手段としてインハウス資源共有計算機網 KUIPNET が開発された^{1),10)}。Fig. 1 に示された構成から判断できるように、KUIPNET の中で汎用的な資源をもっとも豊富に備えたホスト計算機は NEAC 2200/250B である。計算機網向き OS, NOS がこのホスト計算機の資源を計算機網の他のホスト計算機から有効に利用できるようにするため、新規に開発された。もともと NEAC 2200/250B に対して提供されていた OS は、(1)多重プログラミングの機能がなくそのままでは網からのジョブの入力ができない、(2)資源を有効に共有するために課せられた条件 (例えばデータ転送の遅延時間の制限) が非常にきびしく、既存の管理プログラムに単に網制御プログラムを組み込んだもののみでは、到底この要求をみたしえないなどの理由で不十分であり、NOS が著者によって新規に開発された。NOS は 3. に述べるように、計算機という環境から要求される条件を本質的に満足するよう

に (個々の条件の対策をばらばらに積みあげるのではなく)、その構成が設計されている点の特徴となっている。この条件、すなわち設計の目標はこのホスト計算機のもつディスク装置、あるいは言語処理プログラ



MT: tape unit, DP: disk pack, CMT: cassette tape unit, LP: line printer, TW: typewriter, FSS: flying spot scanner, CR: card reader, 1 k=1024.

Fig. 1 Configuration of KUIPNET (Since Nov. 1976).

* The Development of the computer-network-oriented operating system NOS—Specifications and Implementation— by Tsunetoshi HAYASHI (Department of Information Science, Faculty of Engineering, Kyoto University).

** 京都大学工学部情報工学教室

ムなどの資源の共有をこの計算機のみで実行されるジョブ間のみでなく、網全体にわたって最大限に可能にすることである。そのためには処理されるデータの高速度転送、特に音声信号などの生データの実時間転送が可能でなければならない。

2. 計算機網という環境の特殊性

資源共有計算機網 KUIPNET のホスト計算機上で運転される NOS の環境の特殊性（すなわち従来の OS の環境としてみられなかった点）及びそれから要請される条件は次のようなものである。

(1) ホスト計算機間及びサブネットの間の相互作用

従来の周辺装置の制御は特殊な装置を除いて中央処理装置が制御の主導権をもち、その支配のもとに出入力がおこなわれるのが普通である。一方計算機網のサブネットとホスト計算機の場合は共に知能（制御能力）をもつ自律（オートノマス）系であり、その関係は対等であって、どちらか一方が単独で主導権をもつことはない。ホスト計算機間でも同様である。すなわち、OS が常に単独で（他のシステムとは無関係に）動作しているのではなく対等な相手を持つことを前提としている。この点は従来の、OS が常に主導権をもって処理をおこなえばよいという前提とは大きく異なっている。これに対応するには従来の OS の構造では不充分であり、OS 内部でも複数の主導権による分散処理が可能な構造を採用する必要がある。

(2) 資源の共有が目標

計算機網の各ホスト計算機から NOS の管理下にある多くの資源を、何時でも、何処からでも利用可能でなければ資源共有のためには有効でない。また当然、このホスト計算機単独のローカルな処理も同時に実行可能でなければならない。したがって NOS は多重プログラミング方式で、かつ計算機網からジョブを開始するためには、一括処理方式のみでなく TSS もジョブ・スケジュール方式として同時に含まなくてはならない。

この点はすでに、大型計算機システムの OS では実現されているが、中型計算機システム上に作成される NOS でこれを実現することは、普通は非常に困難である。

(3) 原始データの処理

さらにこのホスト計算機のすべての資源を有効に共有するためには、上述の TSS 方式によるメッセージ

単位（文字データ単位）の処理のみでなく、デジタル音声信号データ、画像データなどの生データも NOS の下で処理可能なように、サブネットの高速転送能力とバランスした高速処理が要求される。それ故 NOS はその下で運転されるプログラムに対してデータを高速に、特に音声データの場合には一定の遅延時間内で、転送する能力が必要である（10 bit/サンプル、20 k サンプル/秒の音声信号の速度は 200 kbps）。

(4) プロトコルの概念

計算機網に加入しているホスト計算機相互間、あるいは各ホスト計算機の管理プログラムの下で動作している（相異なるホスト計算機中の）プロセス相互間の通信は、一般に一定の規約にしたがっておこなわれる。従来の計算機間通信が（計算機複合体も含めて）個々の組み合わせに応じて方式を定めていた点と比して非常に異なっている。この規約、すなわちプロトコルの処理が NOS 通信制御（データ転送処理及び網制御）の中心になる。

KUIPNET のプロトコルには文献 3) に与えられるようにホスト・ホスト間、ホスト・サブネット間及びユーザ・ユーザ間などの機能のレベルに対応して階層的に設計されている。NOS の管理プログラムで網制御を実現する場合に、網制御の処理の各階層が自然に対応づけられるように、その構造を設計しなければならない。網向き OS である NOS は、4. で述べるように、それを構成する各モジュール間の独立性が高くプロトコルの階層構造に容易に対処しうる構造になっている。いいかえるとプロトコルにみられる機能の分離、独立性、モジュラリティなどの概念²⁾ がその設計の基本原則となっている。

(5) NOS 導入ホスト計算機固有の特殊性

一般にすべての計算機網についてこれらの点が問題であるが、さらに NOS の場合には KUIPNET 固有の問題が課せられている。KUIPNET は音声理解システム、音声合成システム、画像処理、人工知能などの高度情報処理の研究をサポートすることを目的としている。これらの研究では NEAC 2200/250 B は計算機網上でコンピュータ・コンプレックスを作ることが多いため NOS は次のような能力が必要とされる。

i) デジタル音声の実時間転送

ホスト計算機 MELCOM 70 は音声研究に用いられているが、そこでは音声信号データを実時間で NEAC 2200/250 B のディスク・ファイルとの間で転送可能でなければならない。そのためには NOS の管理プログ

ラム内部の処理が充分高速でなければならない。

ii) 画像処理用コンピュータ・コンプレックスのサポート

KUIPNET を開発するより以前から NEAC 計算機と MACC 7/F 計算機はチャンネル・アダプタにより結合されコンピュータ・コンプレックスとして主に画像処理の研究に使用されてきた。NOS はこのコンピュータ・コンプレックスの在来通りの運転を保障し、サポートする機能が必要である。

iii) 既存システム（既存プログラム）との両立性

NOS の開発より以前に作成され、使用されてきたプログラムは大きな資源であり、これらのプログラムもまた NOS の下で、本質的な修正なしに使用可能でなければならない。ところがこれらのプログラムは、作成されたときには適当な管理プログラムが使用されていなかったため、多重プログラミングの管理プログラム下で運転することを全く考慮していないので、入出力命令・停止命令などの特権命令を直接使用しているものがすくなくない。又、計算機会社から提供されている言語処理プログラム、ユーティリティ・プログラムに関しても同様である。したがって、これらの特権命令についても、それが他のプログラムを破壊したり、あるいは管理プログラムの実行を妨害したりなどしないかぎり、シミュレーションなどの適当な手段によってその実行を保証しなければならない。

iv) 旧操作法との両立性

NOS の開発される以前は簡単なプログラム・ローダを中心として運転していたため、その操作は通常単独・占有状態でおこなわれていた。既存のプログラムはこの状態を前提としている。したがって NOS の下での計算機オペレーションもある程度旧操作法に近いものを可能にする必要がある。プログラムの実行の中断・再開、レジスタ・主記憶の表示・修正などである。

NOS はこれらの中型計算機システムのオペレーティング・システムに対するものとしては非常にきびしく多岐にわたる条件をみたすため、次に述べるような手法によって設計された。

3. NOS オペレーティング・システムの設計

NOS オペレーティング・システムの設計では、2. で述べた条件をみたすのに必要な機能、及びその機能を実現する各モジュールへの分割を、次に述べるように相互間の依存ができるだけ少なくなるように考慮した。

(1) ジョブ・スケジューリング

計算機網において資源の共有を一般的に実現するためには、前述のように多岐にわたるジョブ・スケジューリング方式を実現しなければならない。通常大型の OS では、ジョブ管理は管理方式に対応した論理にしたがって制御プログラムにその一部分として組み込まれる。中規模オペレーティング・システムの NOS では、この方式はプログラムの規模及び常駐主記憶容量の制限から不適当であり、個別のジョブ・スケジューラを管理プログラムに組み込むかわりに、管理プログラムとは独立させユーザ・プログラムとして実行させる方針を採用した。管理プログラムはジョブ・スケジュールに必要な同期、相互排除、資源管理、プロセス生成などの機能を十分に用意するのみであり、固有のジョブの概念、ジョブ・スケジューラをもたない。

ジョブ・スケジューラはユーザのジョブ・スケジューリング・アルゴリズムに基づいて、ユーザ・プログラムとして管理プログラムとは独立に作成される。

この方針によると、(i)常駐管理プログラムがジョブ管理を含まないため、その主記憶に占める大きさが小さくなる、(ii)新しいジョブ・スケジュール方式を簡単に導入することが可能であり open-ended なシステムになっている、(iii)管理プログラム固有のジョブ制御文あるいはジョブ制御コマンドをもたないため利用しやすいジョブ制御文を設計しうる、などの利点がある。

(2) ファイル・システム

NOS の管理プログラムが固有のジョブ・スケジューラをもたないのと同じ理由により、NOS の管理プログラムは固有のファイル・システムを持たない。ただボリューム単位による占有・排除使用、共有使用などの管理のみをおこなっている。ファイルの定義やアクセスもすべてユーザ・プログラムによっておこなわれる。ただし言語処理プログラムとして計算機会社から供給されたものを使用するため、プログラム・ファイルはそれらによって定義されているものと同一の形式を採用し両立性をはかった。

(3) 管理プログラム

前述のように KUIPNET の目標の一つは原始データの実時間転送であり、NOS オペレーティング・システムもこの実時間に充分対応できるようにそのデータ転送処理の遅時間を小さくすることが要求される。

したがって NOS の中核の管理プログラム、NOS モニタは高速転送処理を可能にするため、次のような

構成で設計されている。

(i) 高速送受信に関連したデータ処理をおこなうプロセスに対して、そのプロセスから呼びだされるモニタ・ルーチンも含めて、徹底した優先制御が必要である。本質的にモニタ・ルーチンをも含む優先制御をおこなうプログラム構造が、モニタ・プログラムに採用された。すなわち **Fig. 2** に示すように従来の制御プログラムの設計では、プロセス及びモニタ・ルーチンはモニタ・プログラム中核部の制御下でスケジュールされ実行される。いいかえるとこの中核部が制御の主体であり、主導権を持っている。NOS の管理プログラムでは逆に制御の主体はプロセスにあり、各々のプロセスは課せられた機能やジョブを自律的に処理する。管理プログラム中核部はこれらのプロセスがその与えられた優先度にしたがって並列的に動作するように支えるのみであり、従来の制御プログラムの意味での制御の主体となることはない。モニタ・ルーチン(モニタ・マクロ命令)はプロセス(正確にはプロセスとして動作するプログラム)から呼び出されるサブルーチンとみなされ、呼び出したプロセスの優先度に応じて実行される。当然これらのモニタ・ルーチンは再入可能共有手続きとして作成される。NOS モニタは上に述べた構成を 4. に述べるように具体的に実現

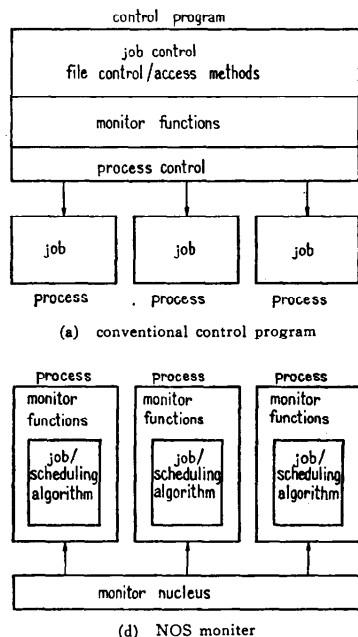


Fig. 2 Structures of a conventional control program and the NOS monitor.

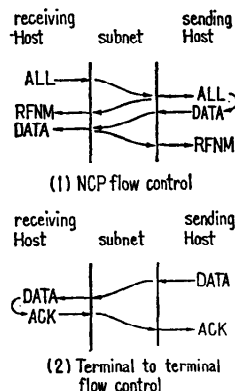
することによって作成された。

又、このような構造はモニタ・プログラムを構成する各モジュール(中核部、プロセス、モニタ・ルーチン)の機能の相互の独立性の非常に高い分割になっている。

(ii) 網上のデータ・フロー制御を高速転送の必要なデータの通るコネクションに対して、ホスト・ホスト(網制御プログラム・網制御プログラム)間のみでなく、コネクションのターミナルであるプロセス・プロセス間でもおこなえるようにする。この機能は次の場合に必要である。(a)網中のホスト計算機の網制御プログラムの中にはフロー制御機能をもたないものがあり、そのようなホスト計算機に対するデータ転送ではユーザ・レベルでフロー制御をおこなわなければならないため、(b) NEAC 2200/250 B とサブネットの間が半二重で結合されているため、高速転送には有効データ・メッセージに対する制御メッセージの回数(オーバ・ヘッド)をできるだけすくなくするため。

KUIPNET のプロトコルに基づいてフロー制御をおこなうと、**Fig. 3** に示すように1回の有効データ転送に対してすくなくとも3回の制御コマンドを含めた送受信が必要であるのに対し、ターミナル間では2回でよい³⁾。

フロー制御をターミナル間でおこなった場合に 10 bit, 10 k サンプル/秒のデジタル音声信号の連続転送が充分可能であり、NOS 実現後実際に音声信号の収集に用いられている。当然この場合でも、コネクションの確立、解放に関してホスト・ホスト・プロトコ



ALL: ALLOCate command, RFNM: Request For Next Message, DATA: regular data message, ACK: ACKnowledge message

Fig. 3 Flow control of the KUIPNET.

ルは有効である。

4. NOS モニタの設計

NOS オペレーティングシステムの中心は、NOS モニタと呼ばれる管理プログラムで、3. に述べた構成にしたがって作成された。以下でこの NOS モニタの具体的な構造を説明する。

(1) NOS モニタの構造

NOS モニタはモニタ中核部、いくつかのモニタ・ルーチンの集り及びいくつかのプロセスから構成されている。この構成は各部分の機能の独立性が最も強くなるように、以下のように考慮されている。

モニタ中核部は、いくつかのプロセスの並列動作とモニタ・ルーチンの呼び出しを確立している。中核部は、モニタ共通データ・ベース、ショート・ターム・プロセス・スケジューラ、外部割り込み処理部及びコール・リターン機構から構成される。モニタ共通データ・ベースは、プロセス、周辺装置、網のコネクション、作業領域などを、主記憶上のコントロール・ブロックとして定義する。プロセス・スケジューラはレディ・プロセス待ち行列から優先度にしたがってプロセスを選び、その実行を開始させる。コール・リターン機構は、プロセスがモニタ・ルーチンの呼び出しをおこなう時に、制御の移行を次のように管理する。各プロセスはこの呼び出しのため、制御スタックをそのデータ・ベース中に備えている。コール・リターン機構は、中央処理装置の状態を、呼び出しをおこなったプロセスの制御スタックに退避させ、呼び出された機能に対応する入口番地及び中央処理装置の状態をスタックする。プロセスの実行は常に制御スタックのトップの状態から再開されるので、次にこのプロセスがプロセス・スケジューラによって選ばれると、呼ばれた機能が実行される。呼び出されたルーチンの処理が完了すると、コール・リターン機構は、スタックをポップ・アップして制御が呼び出し元へ戻るようにする。

外部割り込み処理部は割り込みを受けつけて、その原因に対応するプロセスに通知する。

モニタ・ルーチンは、中核部によって確立されたプロセスが、その与えられた役割を果たすために必要な環境をつくっている。その機能にはプロセス同期、相互排除、プロセス生成・終了、資源管理、入出力制御、及び網制御などを含み、前述のジョブ・スケジューラに必要な機能がこのモニタ・ルーチンとして用意され、プロセスからサブルーチンとして呼び出し可能で

ある。

プロセスはユーザ・プログラム、ジョブなどを実行するユーザ・プロセスと、特定の機能のみを取扱うシステム・プロセスに分けられ、後者はモニタ内部で特定の処理をおこなうために必要である。

(2) システム・プロセス

プロセスの進行と非同期に進行する入出力、メッセージ転送などを集中して処理すること及び自律的な処理が必要な場合、その主体とすることを目標としてシステム・プロセスが導入された。自律的な処理とは、例えばホスト・ホスト・プロトコルに關した網制御で、他ホストからのコネクション確立要求に対する拒否応答、あるいはオペレータ・インタフェース（コンソール・コマンド等の処理）で、オペレータの意志を代表する場合などであり、これらは全てプロセスから呼び出されるサブルーチン型式（モニタ・ルーチン）では実現不可能である。逆にこの構成によってプロトコルを実現するためには、確立されたコネクションはモニタ・ルーチンで処理すればよく、このように各部分の独立性を大きくとった構成が、網制御の階層によく対応しうことは明らかである。

システム・プロセスはプロセス・スケジューラによってスケジュールされて動作する点からいえばプロセスであるが、NOS モニタが計算機システムに格納されて動作を開始した時点から存在し、NOS モニタが停止しない限り消滅しないこと、及び一定の機能のみを処理するなどの点から NOS モニタの本質的な部分である。

(3) ユーザ・プログラム制御

他プロセスのプロセス生成機能の呼び出しによって生成されるユーザ・プロセスはユーザ・プログラムを実行する。この場合に、NOS オペレーティング・システム開発以前におこなわれていた操作卓による操作法、すなわちプログラムの停止、主記憶・制御レジスタの内容の表示・変更などは、既存のプログラムによって欠かせないものであり、又新しいプログラムのデバッグにも必要である。NOS モニタにはこれらの操作卓の機能を代行するためのユーザ・プログラム制御コマンドが用意され、モニタ機能として呼び出されて実行される。これらのコマンドは、特権命令の一部を代行するコンパチブル機能と相まって、NOS 以前に開発されたプログラムをも NOS モニタの下で実行することを可能にしている。

既存のプログラムとの両立性は NOS オペレーティ

ング・システムの要請の中でも重要なものであり、上述のように NOS モニタの機能として解決した。

(4) コンピュータ・コンプレックスのサポート

画像処理用コンピュータ・コンプレックスは画像データ入力用 FSS や、カラー・ディスプレイ装置が接続された MACC 7/F ミニコンピュータと NEAC 2200/250 B によって構成され、通常画像向き周辺装置のついた MACC 7/F 側をマスタとして操作し、NEAC 側はスレーブになっている場合が多い。したがって NOS モニタの下でこのコンピュータ・コンプレックスを動作させるプロセスは、割り込み起動型のリアル・タイム・プロセスとしてサポートされる。生成されたユーザ・プロセスが資源としてチャンネル・アダプタを確保することによって、コンピュータ・コンプレックスと対応づけられる。MACC 7/F からの割り込み信号によって、対応したプロセスは再起動 (wake up) され、その信号源に対応した処理をおこなったのち、再び割り込み待ち状態をとる。もともと画像処理用プログラムは割り込みによって動作するものが多いためこの設計は妥当であった。

(5) 既存プログラムとの両立性及び計算機網下での使用

NOS の導入以前に開発された既存のプログラム及び言語処理プログラムやユーティリティ・プログラムも、一切の修正を加えずに NOS モニタの下で動作させるのみでなく、さらに他のホスト計算機から網を経由してネットワーク・モードで使用可能にするためコンパチブル機能を拡張してユーザ・トラップ機能が導入された。前述のように実行が許される特権命令 (入出力命令) は内部割り込みによって呼び出されるコンパチブル機能によって代行される。この場合にあらかじめ入出力命令の種類 (トランク・アドレス) 及び入口を宣言しておくことによって、ユーザ・プログラムに制御を戻しうるようにするのがトラップ機能である。

既存のプログラム中の操作卓あるいはカード・リーダー、ライン・プリンタに対する入出力命令に対してトラップさせ、あらかじめユーザ・プログラム領域に (ジョブ・スケジューラなどが) 用意したトラップ・プログラムが、データをユーザ・レベル・プロトコルにしたがってデータ変換し、網に転送することによって、網から既存プログラムがそのまま使用可能になる。

従来のユーザ・プログラム中の内部割り込みでは異常状態の処理が主であるのに対し、ユーザ・トラップ

ではむしろ積極的に利用して旧プログラムとの両立を計っている。この機能は NOS の下で作成されたサーバ TELNET に実際に使用されている⁹⁾。

(6) ジョブ・スケジューラ

現在までに一括処理及び TSS 型スケジューリング・プログラムが開発されて標準的に使用されている。一括処理スケジューラは制御プロセス、リーダー・プロセス、いくつかのジョブ・プロセス及びライター・プロセスから構成される。これらのすべてのプロセスはディスク上のファイルにとられたジョブ待ち行列を介してジョブを管理している。リーダー・プロセス及びジョブ・プロセスは、必要に応じて制御プロセスによって生成され、単位ジョブの処理が完了すると消滅する。

ディスク・ファイルにはハードウェアのトラック・リンキング・レコード機能を利用したブロック・チェイン・データ構造を採用した。

TSS 型ジョブ・スケジューリングはユーザ・レベル・プロトコルの TELNET プロトコルを実現して作成された。NEAC 2200/250 B は固有の通信制御装置及び端末装置をもたないのでこの方法が採用された。TELNET システムは TELNET ロガ・プロセスとサーバ・プロセスから構成され、サーバ・プロセスはロガ・プロセスによってログインに対応して生成される。主なジョブはサーバ・プロセスによって処理される。

5. 結 論

NOS オペレーティング・システムの特徴は、その設計に際し要請される条件が、計算機網という環境によって従来考えられなかったほど多岐にわたり、実現するのが困難であったにもかかわらず、NOS モニタという比較的コンパクトな管理プログラムを中心に中規模オペレーティング・システムとして実現した点である。そしてその設計は固有のジョブ・スケジューラ、固有のファイル・システムを持たず open-ended であり、又優先制御をモニタ・ルーチンまで徹底しておこない、高速のデータ転送を可能にして、大幅な資源共有を可能にしている。

いいかえれば、資源共有計算機網という環境をもっとも重要な因子として意識して、はじめて設計されたオペレーティング・システムである点がもっとも特徴となっている。

なお、NOS モニタはアセンブリ言語によって作成され、主記憶容量 44 k 文字を占めている。そして、NOS モニタの下で運転されるジョブ・スケジューラ

として、一括処理、独立ジョブ・スケジューラ及び TSS 形式の TELNET システム⁵⁾が作成されている。

謝辞 NOS の開発中、終始有益な助言を与えてくださいました田畑孝一助教授及び計算機使用の便宜を計っていただきました坂井研究室の皆様には感謝いたします。

参 考 文 献

- 1) 坂井, 田畑, 大西, 北沢: インハウス・コンピュータ・ネットワークと HOST コンピュータ, 情報処理, Vol. 15, No. 12, pp. 948~954 (1974).
- 2) 坂井利之: コンピュータ・ネットワークと情報処理, 情報処理, Vol. 16, No. 3, pp. 170~178 (1975).
- 3) 坂井, 北沢: KUIPNET の解説, KUIPNET マニュアル 1 (1976).
- 4) 坂井, 林: ホスト計算機のネットワーク向モニタ, 第 15 回情報処理学会全国大会 (1974).
- 5) 坂井, 林: ネットワーク向モニタ NOS の TELNET システム, 第 16 回情報処理学会全国大会 (1975).
- 6) Brinch Hansen, P.: The Nucleus of a Multiprogramming System, Comm. of ACM, Vol. 13, No. 4, pp. 238~250 (1970).
- 7) 日本電気: NEAC シリーズ 2200 プログラミング説明書モデル 250, EOI-140026.
- 8) Metcalf, R. M.: Strategies for Operating System in Computer Networks, Proc. of ACM National Conference. (1972).
- 9) 坂井, 林: NEAC 2200/250, NOS オペレーティングシステム, KUIPNET マニュアル 2 (1976).
- 10) T. Sakai, T. Hayashi, S. Kitazawa, K. Tabata and T. Kanade: Inhouse Computer Network KUIPNET, in B. Gilchrist (ed.): Information Processing 77, North-Holland, pp. 161~166 (1977).

(昭和 51 年 11 月 8 日受付)

(昭和 52 年 11 月 11 日再受付)