Ｓ＆Ｔｎｅｔ上に構築された分散型オペレーティング・システムの基本機能

砂原　秀樹，　所　真理雄，　中島　達夫
慶應義塾大学　理工学部

第１章　はじめに

近年、計算機資源の低価格化により、計算機の利用形態は、１箇所に集中して設置されている大型機の共同利用から、分散して設置された中・小型の計算機の専用使用へと変化してきている。そして、共有の対象がハードウェア・リソースからソフトウェアやデータ（情報）へと変化するにしたがって、それら分散した資源の有効利用は、更に重要な問題となってくる。特に、ワークステーションのような小型で高機能な計算機が各所に分散した環境の場合、この資源有効利用の問題は今後更に顕著になり、重要になると考えられる。

分散環境における資源有効利用の第一歩はネットワークであると考えられる。ローカル・ネットワーク、広域ネットワークのどちらも欠くことのできない通信媒体である。この要求を受けて本大学理工学部においてはローカル・ネットワークの研究が1976年より始められ、Keio Ｓ＆Ｔｎｅｔ第一版が81年[Tokoro 81]、第三版が83年より[Murai 83]より稼動している。このネットワークは、Acknowledging Ethernet [Tokoro 77,Tokoro 83] を利用し、Unix*を基本にしたシステムである。現在開発中の、Ｓ＆Ｔｎｅｔでは、ＶＡＸ**をAcknowledging Ethernet でつなぐとともに、電々公社のＤＤＸを介して、他の場所にあるUnix システムとの接続もできるようになっている。

しかし、ただ単にネットワークで計算機を接続するだけでは、分散した資源の有効利用は完全に実現したとはいえない。そこで我々はこの問題を解決するために、Ｓ＆Ｔｎｅｔ上にユーザがネットワークを使っているかどうかを意識しないで使える環境（ネットワーク・トランスペアレンシィ）を持った分散型オペレーティング・システムを構築している。ここでは、このオペレーティング・システムの基本機能について述べる。

本論文では、第２章でＳ＆Ｔｎｅｔシステム全体の概要について説明し、第３章において、本オペレーティング・システムの目標とそれを実現するために必要な機能について述べる。さらに第４章ではこれらの機能を実現するためのソフトウェアの構成について説明をし、最後に、第５章で、本分散型オペレーティング・システムの将来像について述べ全体をまとめる。

第２章　Ｓ＆Ｔｎｅｔシステムの概要

Ｓ＆Ｔｎｅｔ第三版はUnix 4.1BSD上にインターネットを含むまないネットワークを構築した。第三版に関しては、実装及び評価を終えている[Murai 84]。　Ｓ＆Ｔｎｅｔ第四版では、この第三版を受継ぎUnix 4.2BSD上にインターネットを含む様々な分散システムの構築のために強力な通信機能を提供することを目標としている。そのために、次のような５つの目標が設定された。

１）　柔軟性及びモジュール性のためにレイヤ化された通信機能を提供する。

２）　リモート・サイト・プロセスとの通信手段と同様のそれを、ローカル・サイト・プロセスに対しても提供する。

３）　ユーザ・プロセスに様々な形式のプロセス間通信機能を提供する。

４）　ネットワーク間接続機能を提供すること。

５）　高速、高効率、高信頼性の通信を提供する。

１）を達成するために、６－レイヤ・プロトコルによるネットワーク・アーキテクチャが設計され、ネットワーク内トランスポート・サブレイヤ及び下位レイヤを実装するために、バーチャル・サーキットとデータグラムの両方のサービスを組み込んだAcknowledging Ethernet Interface Unit （ＡＥＩＵ）を採用している。２）及び３）を達成するために、セッション・レイヤでプロセス間通信プリミティブをサポートしている。さらに４）を達成するために、ネットワーク間トランスポート・サブレイヤがネットワーク・アーキテクチャに加えられ、柔軟で効率のよいネットワーク間通信を提供している。このネットワーク間通信の対象の一つとして、ＣＣＩＴＴのＸ．２５を採用している電々公社のＤＤＸとのインターフェースを実装中である。最後の５）を達成するために、レイヤに基いた方法と同様にデータに基いた方法がネットワーク・ソフトウェアの設計及び実装に採用された。

第３章　分散型オペレーティング・システム

３．１．　他のシステムの現状と問題点

ここではUnix 4.2BSD(TCP/IP), UCLAのLOCUS 及びXerox のXNS を例にとり、本システムとの比較を行ってみる。

３．１．１　Unix 4.2BSD (TCP/IP)

Unix 4.2BSDのネットワーク・ソフトウェアの構成を図3.1 に示す。

図のsocketがIPC をサポートするためのシステム・コールである。これには、Unix ドメインとINETドメインという２種類のドメインが用意されており、ネットワークを利用する場合には、後者を用いる。INETドメインではプロトコルとしてDARPA のTCP/IP を採用している。IPによってネットワーク間通信が実現されてい

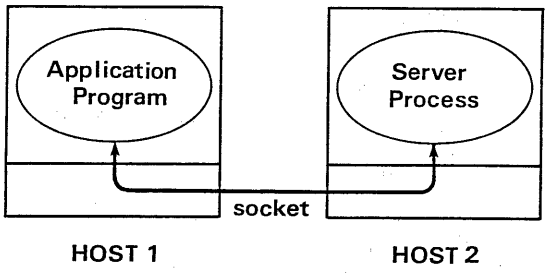


図3.1 Unix 4.2BSD ネットワーク・ソフトウェアの基本構成

*　Unixは米国におけるAT&T Bell 研究所の商標です。

**　VAX はDigital Equipment Corporation の登録商標です。

るため、Unix 4.2BSDではネットワーク間通信をも行うことができる。

Unix 4.2BSDにおいて各アプリケーションは、基本的にServer & client のメカニズムに基いて作られている。そのため、各アプリケーションごとにサーバが作られており、各アプリケーション・プログラムはsocketを通じて目的ホストのサーバと交信することにより実行を進める(rwho-remote whoやruptime-remote uptime では別の構成を取っている)。

1) DARPA のプロトコルを採用しているためUnix 以外のシステムでもDARPA のプロトコルを採用していれば交信できる。
2) ネットワーク間通信を実現している。

などの利点があるが、

1) アプリケーションごとにサーバを必要とするため、各ホストの負荷が大きくなる。
2) どのアプリケーションでも、相手とするホストをいつでも意識していなければならない。
3) ホスト内通信用のUnix ドメインとネットワーク間通信用のINETドメインを使い分ける必要がある。

などの欠点を有する。

3.1.2. LOCUS (UCLA)

LOCUS は、ネットワークワイドなファイルシステムを持ち、記憶の自動複製、分散プロセスの実行、各種高信頼化の機能を備えた、Unix 上位互換な分散型オペレーティング・システムである。このシステムはUCLAにおいて3年間稼動している。

LOCUS は現在16台のＶＡＸ11/750と１台のＶＡＸ11/780をＥthernet で結合したシステム上で稼動している。設計の最重要項目にネットワーク・トランスペアレンシを挙げており、通常の操作ではユーザからは全体が１台の計算機に見えるようになっている。

LOCUS は、

1) ファイルやプロセスは自由に移動ができ、リモートとローカルの区別はない。
2) ファイルは自動的に複製を作るため信頼性が高い。
3) ファイルやプロセスの基本構造はUnix のそれを継承している。
4) プロセス間交信のために（一般化されたメッセージ・パッシング・プリミティブを使うのではなく）目的別に誂えられた50～60のIPC プリミティブが用意されている。
5) heterogeneous なcpu でのシステムの構築が可能になっている。

など数々の優れた機能を持っているが、

1) 今のところネットワーク間通信機能は実現されていない。
2) ほとんどの機能がシステム・コールとして実現されているため、高速な反面、柔軟性に弱く、システムの変更が困難である。
3) ローカルにあるファイルをアクセスする場合でも、１度CSS(Current Syncronization Site)に問合せをする必要があり、ファイルのある場所に関するロケーション・トランスペアレンシが弱い。

など克服されるべき問題も多い。

3.1.3. XNS (Xerox)

Xerox Network System* (ＸＮＳ) はオペレーティング・システムの立場から見ると、図3.2 に示すようにインターネット・トランスポート、クーリエ、クリアリングハウスの三層から成ると見なすことができる。

インターネット・トランスポートでは、ネットワークを越えたホスト間の通信機能を提供する。但しここで提供される通信機能はトランザクション指向のデータグラムのみである。

次のクーリエは、ユーザーにRPC(Remote Procesure Call)を提供している。これは、各プログラムにユニークナンバーをつけ、各プログラム中のプロシジャをそのユニークナンバーとプロシジャー名で呼びだすようになっている。つまり、プロシジャのパラメータが相手ホストに転送され、その相手ホストでリモートプロシジャを実行し、そのリターン値をもとのホストに返すようになっている。これにより、各プログラムにおいては、高レベルな通信が可能となっている。

クリアリングハウスは計算機システムでユーザが名前で参照できるすべてのものをネームドオブジェクトととらえ、その名前とその名前が指すネームドオブジェクトの属性のマッピングをするシステムである。そのオブジェクトの属性は、そのネームドオブジェクトの存在する位置（アドレス）や、ネームがユーザ名のときはユーザプロファイルやそのユーザが使用可能なサーバ名など、グループ名のときは、ユーザ名リストやサーバ名リストなどからなっている。またクリアリングハウスでは、全体でユニークな名前をつけるのを容易にするため、３段階の階層ネーミングを用いている。これは下のように、

<オブジェクト名>@< ドメイン名>@< オーガニゼーション名>

---

* Xerox Network Systems, NS はXerox Corporation の登録商標です。

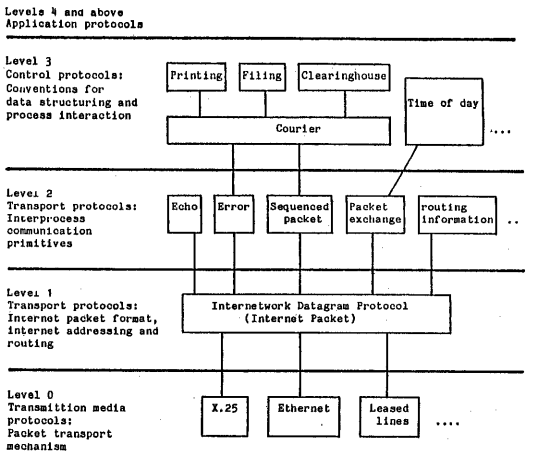


図3.2 XNS のソフトウェア構成

から成り、全体をオーガニゼーションとドメインの２つのレベルで分けている。その分けかたは、論理的なものでネットワークのような物理的なものと関係なく分けることが可能である。また各クリアリングハウスサーバは自分のドメイン内の情報だけを持っており、他ドメインの情報が必要なときはそのドメイン内のクリアリングハウスサーバに問い合わせにいくように設計されている。また各ドメインには、複数のクリアリングハウスサーバが存在することも可能で、そのときはそれぞれのクリアリングハウスサーバは内容の同じデータベースを持つことになる。そのためもし一つのクリアリングハウスサーバが故障しても同じデータが他のクリアリングハウスサーバにも存在するためユーザは故障を気にせず作業を続けることができる。

ＸＮＳでは、このクリアリングハウスサーバによりユーザに高度の分散環境を提供している。しかし、ＸＮＳは基本的にRPC を中心にしたシステムである。RPC を中心にしたシステムは、アプリケーション・レベルに限ればUnix 上にも構築は可能であるが、並列性のレベルやＳ＆Ｔｎｅｔの IPC との整合性の点から、Unix に実装するには問題点がある。

３．２．　本システムの目標

Unix 4.2BSDでは、分散型のオペレーティングシステムとしてはまだ未完成である。またLOCUS はかなり完成されたシステムであるが、まだ解決されるべき問題点も多い。本システムではすでに述べたＳ＆Ｔｎｅｔ上の構築を前提とし、通常の操作ではユーザからはシステム全体が１台の計算機に見える、いわゆるネットワーク・トランスペアレンシを持ったUnix (4.2BSD)上位互換な分散型オペレーティング・システムを構築することにある。これらを達成するためには、以下に挙げるような機能が必要であろう。

1）　ネットワークを意識する必要のないプロセス間通信機能。
2）　プロセスを必要なホスト上に起動する機能。
3）　ネットワーク・ワイドなネームのサービス機能。
4）　ネットワーク・ワイドなファイル・システム。
5）　ネットワークを意識する必要のないユーザ・インタフェースの提供。

１）は、Ｓ＆Ｔｎｅｔ第四版が持つＩＰＣ(Inter Process Comunication)を用いることにより実現している。Ｓ&Ｔｎｅｔ第四版ではネットワーク間交信機能をサポートしているため、複数のネットワークに接続されている場合でも同様にネットワーク・トランスペアレンシを持ったシステムを構築できる。また、２）及び３）については、それぞれ後述するプロセス・サーバ、ネーム・サーバにより実現されている。さらに、４）については、ネットワーク全体で１つの階層構造を持つファイル・システムを構築している。最後の５）については、stshと呼ばれるネットワーク・ワイドな機能を持つシェルによって実現する。この他、ネットワーク間交信をサポートした場合、そのネットワークシステムを使用するユーザがすべてのホスト上にlogin アカウントを持っているとは限らない、そこで我々のシステムではUMT(User Mapping Table) と呼ばれるテーブルを新たに作り、これによりユーザの管理を行っている。

第４章　システムの構成

４．１．　全体構成

本システムの全体構成を図4.1 に示す。

network manager はＳ＆Ｔｎｅｔのプロセスで、ネットワーク間交信機能を含むIPC をサポートしている。UMT(User Mapping Table)は本システム中に存在するすべてのユーザを管理するためのものである。また、プロセス・サーバ(Process Server :ps)は必要なホスト上へのプロセスの生成、ネーム・サーバ(Name Server : ns)はネットワーク・ワイドなネームの管理を行っている。さらに、ファイル・アクセス・ドライバ(File Access Driver : fad) は、ネットワーク・ワイドにファイルにアクセスするためのもので、必要な時にプロセス・サーバにより起動される。そして、これらを利用してstshなどのアプリケーション・プログラム(AP)が作られている。

４．２．　ＩＰＣ(Inter Process Comunication)

これは、すでに述べたように、Ｓ＆Ｔｎｅｔ第四版のIPC を利用している。Ｓ＆Ｔｎｅｔ第四版ではIPC として、コネクション指向のバーチャル・サーキット、コネクションを設定せずに通信を行うデータグラム、サーバ・プロセスなどのためのウェルノウン・チャネルの３種類のものを提供している。

現在、第四版は設計と並行して実装を行っている。[鈴木 84]

４．３．　ＵＭＴ(User Mapping Table)とプロテクション機能

Unix では、もともとユーザの識別子としてuid というものを持っていた。しかし、これらは各ホストごとにつけられており、複数のホスト上にlogin アカウントを持つユーザがネットワークを介してそれらのホストをアクセスしようとしたときに問題が生じてくる（特にホストごとにlogin 名が違うとき）。この他、ネットワーク間交信をサポートした場合、そのネットワークシステムを使用するユーザがすべてのホスト上にlogin アカウントを持っているとは限らない。そのために、我々のシステムでは、新たにnuid(Network uid) と呼ばれるユーザidを導入し、全システム上でユニークなidを各ユーザに割当ることにした。

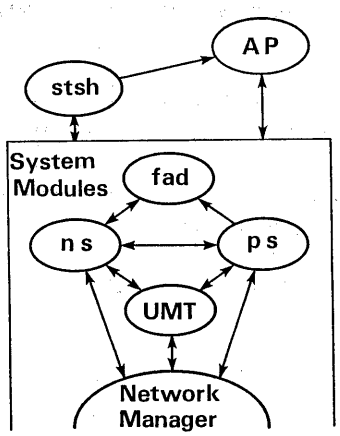


図4.1 本システムの基本構成

前述したように、各ホストでのuid(nuidと区別するためにlnuid - local nuidと呼んでいる) はそれぞれ独立してつけられているために一致していることはまず考えられない、そこでこれら各ホストでのlnuid とnuidのマッピングをとる必要が出てくる。また、外来ユーザ（そのホストにlogin アカウントがないユーザ）に対してもある程度のサービスを提供するために、外来ユーザ用のlnuid を用意し、これらのlnuid とのマッピングを行うことも必要になってくる。図4.2 にnuidとlnuid の構成を示す。

こういったlnuid とnuidとのマッピングをとるためのテーブルがUMT(User Mapping Table) である。さらに、新しい外来ユーザがそのホストにアクセスしてくるごとに新しいlnuid を割当てる機能も持っている。

ところで、従来Unix ではファイルそれぞれにowner,group, othersの３レベルのプロテクションがついていた。しかし、このようにあるホスト上にlogin アカウントのないユーザに対してもネットワークからの侵入を許したために、これだけのプロテクションでは十分とはいえなくなってきた。また、従来のプロテクションの機能では、十分とはいえない部分（特にgroup のプロテクション）があるため、我々はファイルシステムの拡張を行い、各ファイルごとに従来の３つのプロテクションに加えて、network(外来ユーザのパーミッション）と各ユーザごとのパーミッションがつけられるようになっている。これにより、複雑なファイル管理も自由に行えるようになっている。

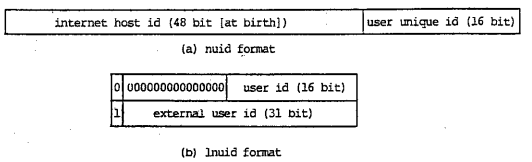


図4.2 nuidとlnuid の構成

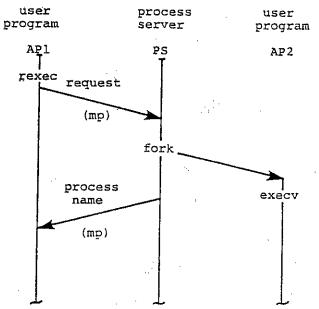
４．４．　プロセス・サーバ

プロセス・サーバはリモート・ホスト上にプロセスを生成したり、リモート・ホスト上のプロセスにシグナルをかけるためのサーバ・プロセスである。それぞれユーザからはrexec,rkill というファンクションによって呼ばれる。これらのファンクションはリモートとローカルの区別なく使うことができる。

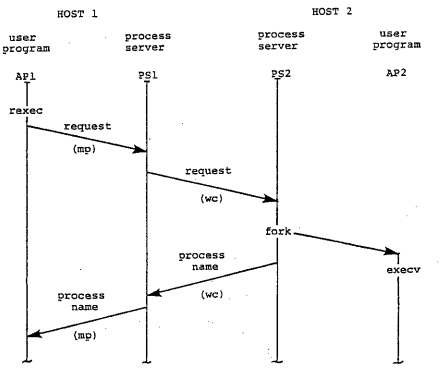
図4.3 に、プロセス・サーバによるプロセスの起動のメカニズムを示す。

リモート・プロセスを起動しようとするときには、まずネットワーク・ライブラリ関数であるrexec を呼ぶ。rexec では、プロセス・サーバへのプロセス起動要求パケットを作成し、mp* を通じてローカル・ホストのプロセス・サーバに対して要求を送る。要求を受けたプロセス・サーバでは、プロセスを起動しようとするホストが自分自身（ローカル・ホスト）かそうでない（リモート・ホスト）かを判断し、ローカルであれば自分自身がforkしてローカル・ホストに必要なプロセスを起動する。そして、起動されたプロセス名を要求してきたプロセスにmpを通じて返す。リモート・ホストへの起動要求であれば、ローカルのプロセス・サーバは、プロセス起動要求パケットをプロセス・サーバ用のウェルノウン・チャネルを通じてプロセスを起動したいホストのプロセス・サーバへ送る。要求を受けたリモートのプロセス・サーバはローカルの場合と同様にforkして必要なプロセスを起動し、起動されたプロセス名を要求のあったホストのプロセス・サーバにウェルノウン・チャネルを通じて返す。プロセス名を受け取ったプロセス・サーバは、mpで起動要求をしてきたプロセスに受け取ったプロセス名を返す。起動要求をしたプロセスではmpから起動されたプロセス名を受け取ると、必要な引数のセットをしてrexec を終了する。また、途中でエラーが起きたときはその時点でエラー・コードが返されるようになっている。

この他、プロセス・サーバで実現されるrkill なども同様にして実行される。



(a) ローカル・プロセスの起動



(b) リモート・プロセスの起動

図4.3 プロセス起動のメカニズム

４．５．　ネーム・サーバ

本システムにおけるネーム・サーバは、システム中の各ホスト名の管理とともに、Unix におけるオブジェクトであるユーザ，プロセス，ファイルにユーザがネットワークトランスペアレントにアクセスする機能のためのネーム・サービスを提供する。本

---

* カーネル内に実装されたメッセージ・パッシング・プリミティブでローカル・ホスト内での簡単なIPC を提供する。4.2BSDオリジナルのIPC と比較して、利用方法が簡単であり、高速である。

ネーム・サーバは基本的には、Ｘｅｒｏｘのclearing houseと同じ機能を提供するものである、clearing houseは分散環境をサポートする実用的なネームサーバとしては、最も高度だと思われるため、本システムにおいても基本的に同じ機能を持つように設計されている。

ネーム・サーバは２種類のネーム管理をおこなっており、１つはネーム付きオブジェクト（ユーザ，プロセス，ファイルをまとめての総称）のネームの管理、もう１つは機能によるネームの管理である。ネーム付きオブジェクトのネームの管理とは、それぞれのネーム付きオブジェクトを識別するのに用いるもので、各ユーザとか、各プロセスとかの識別に用いる。次の機能によるネームの管理とは、各ユーザにプリンタ・サーバや各ユーザのメイルボックスなどの各オブジェクトを機能により分類するときに用いる。

ネームは２つの種類にわけられていて、１つはユーザが直接用いるシンボリック名、もう１つはシステムが用いるidである。シンボリック名はXNS と同様に３段階の階層ネーミングを用いている。

idは２種類にわけられていて、１つは論理id、もう１つは物理idである。論理idはシステムが各オブジェクトを識別するために用いるもので、オブジェクトの位置と関係なくつけられる。また、これは複数のオブジェクトのグルーピングをおこなうことを可能としている。このため、複数の中味の同じオブジェクトの物理idと１つの論理idとをマッピングすることで多重コピー管理を行ったり、同じ機能のオブジェクトに１つの論理idをつけ、１番近くにいる、正常に動いているなどの要素でそのうちの１つを選ぶなど、高信頼性化、最適化の機能を実現することもできる。物理idとは、各オブジェクトの位置を示しており、すべてのオブジェクト（実体）に一意な名前がつけられる。このような機能を実現しようとすると、ネームサーバでは分散データベースと同様の問題を生じてくる。この問題については現在検討中で、今後の課題である。

現在実装中のネーム・サーバは以上の機能のうち比較的静的なホスト・アドレスの管理を提供しており、その他についてはアルゴリズムなどの検討を行っている段階である。

４．６．　その他

ファイル・アクセス・ドライバは、ネットワーク・ワイドなファイル・アクセスを提供するための機能である。これは、open,close,read,write などのファイルをアクセスするためのファンクションが呼ばれたときに、ネーム・サーバに問合せを行い、リモート・ホスト上のファイルであればプロセス・サーバによって必要なホストに起動される。こうすることにより、ファイルのアクセスはネットワークを意識することなく行えるようになっている。特にアクセスしたいファイルがローカル・ホスト上にあるときには従来のシステム・コールが呼ばれるためLOCUS に比べて効率は良くなっている。

ところで、ユーザがネットワークを意識することなくシステムを利用するためには、そのシステムのユーザ・インターフェイスが重要な位置を占めてくる。そのためのよいユーザ・インターフェイスを提供するのが分散環境用シェル(stsh)である。この上でシステムを使う限り１台の計算機を使用しているように見える。また、これにより従来のUnix のコマンド群は１部の変更だけでほとんどのものが使えるようになっている。

また、もう１つstshの重要な機能として負荷分散機能が挙げられる。これは、各ホストのload averageにしたがって実行されたコマンドを起動するホストを決める機能で、システム内の各ホストの負荷を一様にするためのものである。例えば、複数のコマンドをパイプで接続したような場合でも各コマンドを別々のホストに割り当てることで簡単な負荷分散機能を実現している。

その他、ユーザ・アプリケーション・プログラムで複数プロセスを数台のホストに起動して行う並列処理や複数のＣＰＵがあることを利用した障害回復機能などを実現するための基本機能なども提供することを考えている。

第５章　今後の展望

現在、以上に述べた機能のほとんどが実装中であり今後解決しなければならない問題は数多くある。特に、ネーム・サーバにおけるネームの分散管理のアルゴリズムやheterogeneous なcpu でのシステムの構築、ファイルの多重コピー管理などは、充分な検討が必要と考えられる。

謝辞

ネットワーク・ソフトウェアの実装に御協力戴いた村井　純、寺岡　文男両氏に感謝いたします。また、ファイル・システムの改造を行って下さった河村　元夫氏に感謝いたします。最後に、このシステムを構築するにあたり一緒に研究を進めている所研究室ネットワーク・グループの各メンバーに感謝いたします。

<<参考文献>>

[Murai 83]　Jun Murai, Mario Tokoro, Fumio Teraoka:"Keio S&Tnet: A Unix Campus Network," Proc. of the 8th Conf. on Local Computer Networks, pp.14-23, Oct. 1983.

[Murai 84]　Jun Murai, Fumio Teraoka , Mario Tokoro: "Measured Performance of a Unix-based Local Area Network -Keio S&Tnet-," to appear in COMPCON '84 fall, Sep. 1984.

[Tokoro 77]　Mario Tokoro and Kiichiro Tamaru: "Acknowledging Ethernet,"COMPCON '77, pp.320-325, Sep. 1977.

[Tokoro 81]　Mario Tokoro, Jun Murai, Masahiro Ikeda: "UNIX Campus Network on Acknowledging Ethernet," Proc. of the DECUS, Vol.2, No.6, pp.214-221, 1981.

[Tokoro 83]　Mario Tokoro, Koichi Tanaka, Fumio Teraika, Toshitada Saito: "The Development of the Acknowledging Ethernet System",COMPCON FALL, Sep. 1983.

[Ucb 82] the CSRG, UC Berkeley, "4.2BSD System Manual ; Draft of September 1, 1982," 1982.

[Popek 81] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin,G. Thiel:"LOCUS A Network-Transparent, High Reliability, Distributed System," Proc. of the 8th Symp. on Operating Systems Principles, Vol.15, No.5, pp.169-177, Dec. 1981.

[Walker 83] Bruce Walker, Gerald Popek, Robert English, Charles Kline, Gerg Thiel:"The LOCUS Distributed Operating System," Proc. of the 9th Symp. on Operating Systems Principles, Vol.17, No.5, pp.49-70, Oct. 1983.

[Mueller 83] Erik T. Mueller, Johanna D. Moore, Gerald J. Popek:"A Nested Transaction Mechanism for LOCUS," Proc. of the 9th Symp. on Operating Systems Principles, Vol.17, No.5, pp.71-89, Oct. 1983.

[Xerox 81] Xerox Corporation: "Internet Transport Protocols," Xerox System Integration Standard:XSIS 028112, Dec. 1981.

[Oppen 81] Derek C. Oppen, Yogen K. Dalal: "The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment," OPD-T8103,Oct. 1981.

[鈴木 84] 鈴木 洋一，中島 達夫，砂原 秀樹，所 真理雄:"Keio Campus Network S&Tnetにおけるネットワークインタコネクションの方式," 「LAN/マルチメディアの応用と分散処理」シンポジウム，情報処理学会，Oct. 1984.