マルチメディア通信と分散処理 35－9
（１９８７・９・１７）

仮想ネットワーク

― ネットワークにおける資源管理方式 ―

貫井春美　　三原幸博

（株式会社東芝　システム・ソフトウェア技術研究所）

ソフトウェアの分散開発環境において、ネットワーク上の開発装置（ワークステーション、パーソナルコンピュータ、ミニコンピュータ等）が、通信を意識せず利用できるネットワーク機能について考察する。このようなネットワークを仮想ネットワークと呼び、ネットワーク上の開発装置全体を仮想的にひとつの開発装置として利用できるソフトウェアの分散開発環境の構築を目指す。仮想ネットワークは、ネットワーク上の全資源のアクセス方法をあたえるネットワークファイルシステム、ネットワーク上でのプロセス起動の制御と負荷分散を行なうネットワークプロセス制御、そして、ネットワーク上のユーザ情報、ノード情報等を一元管理するネットワークアカウント管理の３つの機能をもつ。

本稿では、仮想ネットワークの資源管理において、主機能であるネットワークファイルシステムとプロセス制御について報告する。

Virtual Network

-- resource management arround the network --

Harumi Nukui , Yukihiro Mihara

System & Software Engineering Laboratory , Toshiba Corporation

70 , Yanagi-cho , Saiwai-ku , Kawasaki , Kanagawa , 210 Japan

On the distributed software development environment, it is necessary that the users can access to the remote resources and execute process arround the network without conscious of using the network.

We call such network "Virtual Network" which has following three major functions ;

(1)Network File System : provides access methods to remote resources(files).

(2)Network Process Control : provides process control methods arround the network.

(3)Network Account Management : provides account management arround the network.

In this papar, we described the basic specifications of Network File System and part of Network Process Control(remote process execution system).

1．まえがき

ソフトウェアの開発・管理には、開発対象となるソフトウェアの種類・特性、開発の各段階に於ける支援ツールの種類により、また管理目的により様々な設備が必要となる。従来、それぞれの設備の中だけで効率を追及するあまり、全体を通してみるとかえって効率の良くないソフトウェア開発を行っていることがあった。ソフト生産システムを考える場合、個々のフェーズでの効率だけでなく、ライフサイクル全体を通した一貫性の実現と開発・管理の統合を目指さなくてはならない。

このように、様々な設備を利用してソフトウェアの開発・管理を行なう場合、それ等の設備間の情報の流れをスムーズに行ない、かつ限られた資源を有効に使えるようなネットワークが必要である。

このような問題を解決するためには、個々の設備をネットワーク結合するという考え方をさらに発展させ、ネットワーク結合された個々の設備全体を仮想的な一台の設備として使用できるという考え方に基づいたネットワーク機能が有効であると考える。

このような考え方に基づいたネットワークを仮想ネットワークとよび、利用者がネットワークの存在を意識することなく、開発および運用できる環境の実現をめざす。

2．仮想ネットワークの概要

2－1．仮想ネットワークの基本方針

仮想ネットワークは、以下に示す４つの基本方針に従う。

(1)自立性

ネットワーク上の各ノードの資源は、ノード単位に管理し、ネットワーク上の任意のノードの障害が他ノードに影響を及ぼさない。

(2)透過性

ネットワーク上の全資源がネットワークの形態を意識することなく利用でき、ネットワーク上の他ノードの資源が機種・ＯＳを意識することなく、自ノードの資源と同様な方法で利用できる。

(3)継続性

ユーザインタフェース、ＯＳインターフェースが従来と同様である。

(4)移植性

異機種・異ＯＳへの移植や、ＯＳのバージョンアップへの対応が容易である。

2－2．仮想ネットワークの構成

仮想ネットワークは、他ノードの持つ資源と自ノードの資源の利用において差（性能）がないことが条件となる。

したがって、仮想ネットワークは高速な通信媒体上に構築する必要がある。そして、論理的には、ネットワークが１つの計算機として存在する。

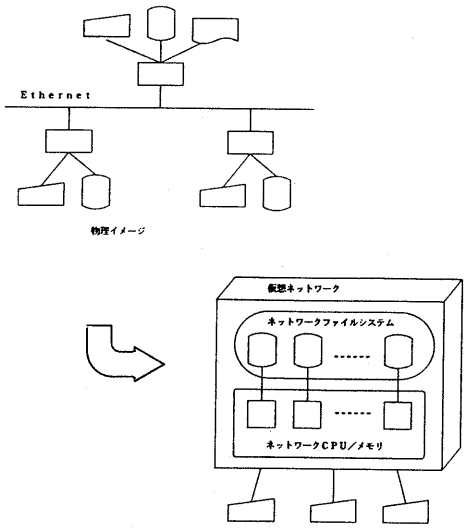


図１．仮想ネットワークのイメージ

図１に仮想ネットワークの物理イメージと論理イメージを示す。

2－3．仮想ネットワークの機能

仮想ネットワークには以下の機能が必要と考える。

(1)ネットワークファイルシステム

ネットワーク上のすべての資源（ファイル）を、ネットワークでひとつの階層木構造（Network tree）で表現する。

これにより、ネットワーク上での名前管理方式とアクセスメソッドを与える。

(2)ネットワークプロセス制御機能

ネットワーク上でのプロセスの実行制御と、プロセス実行によるネットワーク上での負荷分散を行なう。

(3)ネットワークアカウント管理機能

ネットワーク上でのアカウント情報を一元管理する。ユーザ情報、ノード情報等が管理対象になる。

3．ネットワークファイルシステム

ネットワーク上のファイル（資源）を一意に識別するためには、ファイルをネットワーク上で一元管理する名前管理方式とそれに対するアクセスメソッドが必要となる。

仮想ネットワークでは、ＵＮＩＸファイルシステムをベースとしたネッツトワーク上でのファイルシステムについて考案した。

3－1．ファイル名管理

3－1－1．ネットワークｔｒｅｅ構造

従来のＵＮＩＸのファイルシステムは、最上位のディレクトリをルートディレクトリとよび、その下位に、階層木

構造のファイルシステムが配され、ファイルは、ルートディレクトリから、ディレクトリを順次下位にたどっていくことにより、一意に決定される。

ネットワークファイルシステムでは、ＵＮＩＸ本来のファイルシステムの概念をネットワークに拡張する。すなわち、ネットワーク全体で１つの階層木構造を成すファイルシステムを構築する。

従来のルートディレクトリの上位にネットワーク全体のルートディレクトリを設定する。その下位に各ノード（マシン）を識別するノードディレクトリを設定し、以下従来のファイルシステムが配置され、ネットワーク全体でひとつのｔｒｅｅ構造（ネットワークｔｒｅｅ）を構築する。

最上位のディレクトリを、ネットワークルートディレクトリと呼び、"／／"と表現する。

図２に、ネットワークtree構造を示す。

ここでノードN2上のファイル/d0/f0は、//N2/d0/f0と記述する。また、ノードN2上のユーザーに対しては、省略記法として/d0/f0または//d0/f0 と記述することを許す。

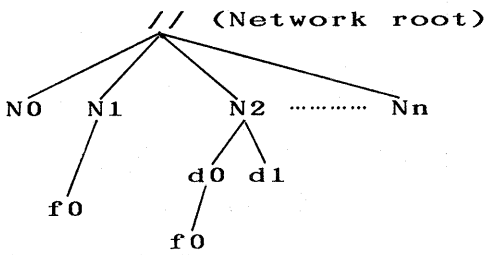


図２．ネットワークｔｒｅｅ構造

3－1－2．ノードの識別

ユーザーから指定されたファイルが、リモートファイルであるか、またリモートファイルである場合どのノードにファイルが実存（物理的に存在する）するかをシステムが自動的に識別しなければならない。

（1）識別方法

以下手順によりノードの識別を行なう。

①先頭が"／／"の時

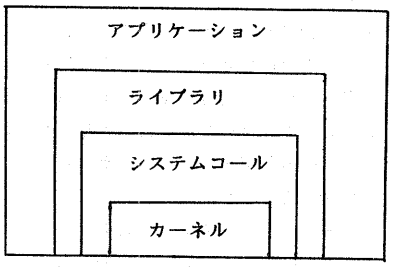
```
if（先頭ディレクトリ名が自ノード名）
  then  自ノードのファイル
  else  if（先頭ディレクトリ名が
              ネットワークに接続されている
                  ノード名と一致しない）
          then  エラー
          else  先頭ディレクトリ名が実存す
                るノードを表わす
        endif
endif
```

②先頭が"／／"でない時

すべて自ノードのファイルとして扱う。自ノードのファイルとして扱う場合は、従来の処理に従う。

（2）識別レベル

ＵＮＩＸ上で動作するプロセスは、次に示す４階層に分かれる。



どのレベルにおいてリモートまたはローカルであるかの識別を行なうかを考えると、

①カーネルまたはシステムコールで識別する。

②ライブラリで識別する。

③アプリケーションで識別する。

の３方式が考えられる。

前提として考えなければならないことは

・移植性を考慮する

・従来とインターフェースを同一にする

ということである。そして、これらの前提条件をもとに、考慮すべき点として以下の５点を考える。

・移植性

異機種への移植が容易である（優）か容易でない（劣）か

・インターフェース

従来と同一であるか異なるか

・応答性

応答が速い（優）か遅い（劣）か

・実現性

実現が容易か容易でないか

・他への影響

他のユーティリティ、ライブラリ等への影響があるかどうか

以上５点から各方式を比較・検討すると、表１に示す結果を得る。この結果から、仮想ネットワークではライブラリレベルで識別している。

3－1－3．ディレクトリ管理

ネットワーク上のファイルの情報をどこでどのように管理するかを考察する。

管理対象となる情報には、以下のものがある。

①ファイルの型（ディレクトリorファイルor特殊　ファイルetc ）

②ｉノード番号，機器番号
③ファイルのリンク数，サイズ
④ユーザーid，グループid
⑤参照日付，変更日付　etc

ここでは、これらの情報をネットワーク上で管理する４つの方式について比較・検討する。

（１）集中管理方式

ネットワーク上の全ファイルの情報を１つのノードで管理する。ネットワーク上にマスターノードを設ける。したがって、自ノードのファイル情報を参照する時もマスターノードに対しディレクトリ情報を問い合わせなければならない。そのため、通信の負荷が増加し、従来のローカルファイルアクセスに対し影響を与える。

ただし、リモートファイルのアクセスも、ローカルファイルのアクセスと同程度の性能が期待できる。

（２）自己管理方式

自ノードに実存するファイルに関しては、自ノードで管理する。したがってリモートノードのディレクトリ情報は一切関知しない。

基本的には、従来のＵＮＩＸのディレクトリ管理とまったく同様であるため、ファイルの作成・削除におけるディレクトリ情報の変更は容易に行なえる。

ただし、リモートノードのファイルに関してはノードの存在場所を識別し、通信によりファイルアクセスを行なう。ローカルアクセスが従来と同等の性能で行なえるが、リモートファイルアクセスの負荷は、やや増加すると思われる。

（３）複製管理方式

各ノードは、ネットワーク上のすべてのファイルのディレクトリ情報を持つ。リモートファイルアクセスの際のディレクトリ参照はすべて自ノードにおいて行うことができる。したがって、リモートファイルアクセスの応答性に関しては、高性能が期待される。ただし、ファイルの作成・削除に関する管理情報の更新の際に、ネットワーク全体でその同期をとらなければならないために、各ノードおよびネットワーク全体の負荷が増加する。また、各ノードが持つ管理情報量も増加する。

（４）特定管理方式

基本的には自己管理方式であるが、ネットワーク上の複数の特定ノードのみが全ディレクトリの管理を行なう。したがって複製管理方式と自己管理方式の中間方式と考えられる。

リモートファイルアクセスに対してのみ、複製を持つ特定ノードに問い合わせを行ない、ローカルファイルに対しては自ノードで識別する。リモートファイルアクセスの応答性は複製管理方式には劣ると思われるが、その他の方式に対しては優れたものが期待できると考える。　ただし、ディレクトリ情報の更新に関してはネットワーク上で同期をとる必要があり、それに対する負荷は増加する。

以上述べた４つの方式を

①　ファイルの作成・削除その他のアクセスによるディレクトリ情報変更時の負荷
②　従来のローカルファイルアクセス時の負荷
③　リモートファイルアクセス時の負荷
④　管理情報量

の４点に着目して比較すると表２に示す結果を得る。

これらの中で特に次の２点に重点をおく。

- 従来行っているローカルファイルアクセスへの影響を最小にする。
- ファイルアクセスにともなうディレクトリ情報の変更が容易にできる。

以上の点を考慮し、仮想ネットワークでは自己管理方式を用いる。

	移植性	インターフェース	応答性	実現性	他への影響
①	劣	同　一	優	難	カーネルの再ジェネレーション
②	優	同　一	優	容易	アプリケーションの再コンパイル
③	優	異なる	劣	容易	な　し

表１．資源識別レベルの比較

方　式	更　新	ローカルアクセスの負荷	リモートアクセスの負荷	管理情報量
集中管理	簡　単	増　加	やや増加	変化なし
自己管理	簡　単	変化なし	増　加	変化なし
複製管理	複　雑	変化なし	変化なし	増　加
特定管理	複　雑	変化なし	やや増加	やや増加

表２．ディレクトリ管理方式の比較

３－２．リモートファイルアクセス

リモートノードに存在するファイルをアクセスする。ファイルをアクセスするとは、以下の処理を行なうことと考える。

①ファイルをオープンする（open）
②ファイルを作成する（creat）
③ファイルからデータを読み出す（read）
④ファイルへデータを書き込む（write）
⑤ファイルのアクセスポインタを移動する（seek）

⑥ファイルのリンクを作成する（link）
⑦ファイルを削除する（unkink）
⑧ファイルをクローズする（close）

上記の処理を従来のインターフェースと同様に行なう。すなわち、自ノードのファイルと同様のオペレーションまたは、関数へのパラメータの渡しによって、リモートファイルアクセスが行なえる環境を作り出す。

ファイルアクセスは、ファイルの実存するノードで行なう。ファイルアクセス要求が発生したノードでは、上記に示したファイルアクセス要求に必要な情報（パラメータ）を実存するノードに対して転送する。

本方式をパラメータ転送方式と呼ぶ。

パラメータ転送は、プロセスの階層構造から考えて、ファイルの実存ノードの識別で述べたように４つのレベルで行なうことが考えられる。ファイルの実存ノードの識別をライブラリで行なうことから、パラメータ転送もライブラリで行なう。この時。ファイルアクセス要求の処理性能（ターンアラウンド）を向上させるための処理方式が必要になる。

（１）構成要素

パラメータ転送方式の構成要素として以下のものが考えられる。

・ユーザープロセス（user）
　リモートファイルアクセス要求を発生させるプロセスで、ファイルアクセスライブラリを含む。
・アクセス要求プロセス（requester）
　ユーザープロセスからのリモートファイルアクセス要求を受付け、ファイルが実存するノードに対してパラメータ転送のサービスを行なう。
・アクセス実行プロセス（server）
　リモートノードからのファイルアクセス要求を受付け、ファイルアクセスのサービスを行なう。

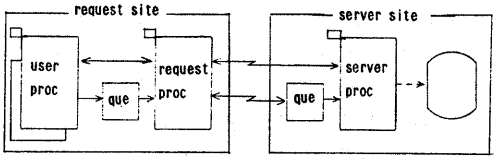


図３．単純client-server モデル

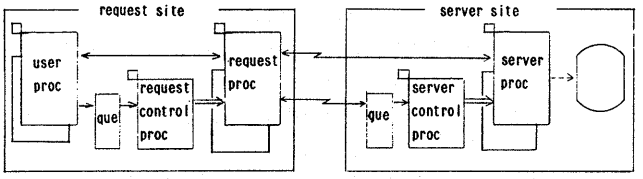


図４．二重client-server モデル

（２）単純Client-server モデル

requester 、serverがシステムに１プロセス存在しファイルアクセスのサービスを行なう。

処理方式を図３に示す。

（３）二重Client-Server モデル

(1) でのべた構成要素に以下の２プロセスを構成要素に加える。

・アクセス要求管理プロセス
　userからのアクセス要求を受付けuserに対応するrequester を起動する。
・アクセス実行管理プロセス
　requester からのアクセス要求を受付けrequester に対応するserverを起動する。

したがって、user-requester-server という１対１のサービスが可能になる。

処理方式を図４に示す。

（４）比較・検討

上記の２方式をターンアラウンド（１プロセスあたりの平均処理時間）に着目して比較すると、実用域において、二重client-server モデルのが優れる。

したがって、仮想ネットワークのリモートファイルアクセス方式は二重client-server モデルにより行なう。

４．ネットワークプロセス制御

４－１．プロセスと資源

ディスクファイルとして存在するロードモジュールが、メモリ上にローディングされ、プロセスが生成される。生成されたプロセスは、ネットワーク上に存在する他のディスクファイルや周辺機器のような資源を利用しながら動作する。

仮想ネットワークにおいて、プロセスが生成されるために必要な資源は、以下の３つにまとめられる。

①	ＣＰＵ	プロセスの走行制御
②	メモリ	プロセスの走行空間。
③	ファイル	ロードモジュールファイル、データファイル、また周辺機器もファイルとして扱う

図５にプロセスと資源の関係について示す。

ここで、ロードモジュールファイルの実存するノードを「存在ノード」、プロセスとして走行するためのＣＰＵ、メモリが存在するノードを「走行ノード」と定義する。

４－２　プロセスの分類

前節で述べたプロセスと資源の関係から、ネットワーク上のプロセスを分類すると、図６に示す５つの場合に分類される。

① ローカルノードのロードモジュールをローカルノードで起動（存在ノードと、走行ノードが同一で、ローカルノードの場合）

従来のプロセス起動である。利用する資源（ＣＰＵ，メモリ）も、すべてローカルのものを利用する。

② ローカルノードのロードモジュールをリモートノードで起動（存在ノードがローカルノードで、走行ノードがリモートノードの場合）

ローカルノードのロードモジュールファイルをリモートノードの資源（ｃｐｕ、メモリｅｔｃ）を利用し、プロセスとして生成させる。

③ リモートノードのロードモジュールをローカルノードで起動（存在ノードがリモートノードで、走行ノードがローカルノードの場合）

リモートノードのロードモジュールファイルをローカルノードの資源（ＣＰＵ，メモリ）を利用し、プロセスとして生成させる。上記(2) と逆である。

④ リモートノードのロードモジュールをリモートノードで起動（存在ノード、走行ノードが同一で、リモートノードの場合）

リモートノードのロードモジュールファイルをリモートノードの資源（ＣＰＵ，メモリ）を利用し、プロセスとして生成させる。上記(1) において、利用する資源がすべてリモートノードとなった場合である。

⑤ リモートノードのロードモジュールを異なるリモートノードで起動（存在ノード、走行ノードがリモートノードで、それぞれ異なる場合）

リモートノードのロードモジュールをさらに異なるリモートノードの資源（ＣＰＵ，メモリ）を使用してプロセスとして生成させる。

以上のように分類されたネットワーク上のプロセスは、

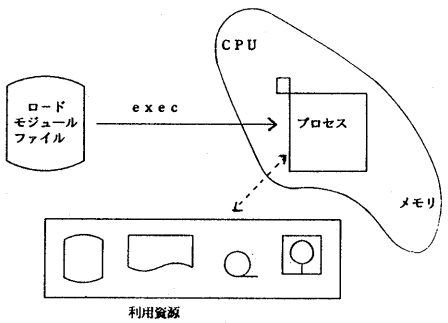


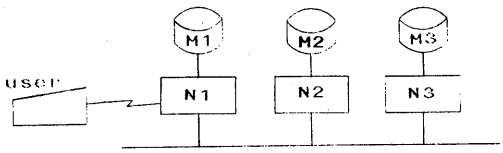
図５．プロセスと資源

存在ノードと走行ノードに着目すると、以下の２つに大別される。

・存在ノードと走行ノードが同一（①，④）

・存在ノードと走行ノードが異なる（②，③，⑤）

以下にそれぞれの問題点について考察する。



	①	②	③	④	⑤
ロード モジュール	M１	M１	M２	M２	M２
走行ノード	N１	N２	N１	N２	N３

図６．プロセスの分類

４－３．プロセス起動の問題点

(1) 存在ノードと走行ノードが同一（①，④）

①は従来のプロセス起動なので問題はない。

また、④はリモートノードでのプロセス起動であるが、プロセス起動要求のメカニズムがあれば、基本的に①つまりローカルプロセスの起動と同様である。

プロセス起動のメカニズムとは、

- ロードモジュールファイルの存在ノードの検索
- プロセス起動のパラメータの転送
- プロセス実行結果の分配

の処理を行なうことである。

④の起動形態は、リモートプロセス起動の基本形態と考えられる。

(2) 存在ノードと走行ノードが異なる（②，③，⑤）

リモートノードとローカルノードのＣＰＵのアーキテクチャが異なる時には、プロセスの生成は不可能である。ロードモジュールがサービスプログラム（コマンド）のように、同じ動作をするものが、他のリモートノードに存在する場合は、そのリモートノードでプロセスを起動（プロセス起動の代行）することも考えられる。しかしそうでない時、ユーザ作成のプログラムのような場合にはこのような対応は不可能である。

そのような場合には、ロードモジュールファイルの属性に実行ＣＰＵを持たせ、実行ＣＰＵを検索することによりプロセスの生成は可能になる。しかし、これはカーネルに依存する機能拡張である。

また、利用資源の不足という問題も生じる。例えば、メモリの不足、データファイルの有無等の問題が考えら

れる。これに対応するには、実行環境を模作する必要がある。

以上述べてきたプロセス起動方式に関し、仮想ネットワークの基本概念を考えると以下のように結論する。

- 起動方式①は、従来のプロセス起動なので、考慮しなくともよい。
- 起動方式②，③，⑤は、リモートノードの資源を分割して利用する。（ロードモジュールファイル、ｃｐｕ、メモリ）このため、ノードの障害が影響する恐れがある。また、ｃｐｕタイプの異なるノード間では、本形態によるプロセス起動が不可能であるため同様のｃｐｕを選択する必要がある。また、利用資源の不足の問題もある。
- 起動方式④は、リモートノードでのプロセス起動であるが、起動自体に関しては、起動方式①と同様であり、リモートプロセス起動（②，③，⑤）の基本形態と考えられる。

したがって、以上の観点から、仮想ネットワークにおけるリモートプロセス起動として、まず起動方式④すなわち、リモートノードのロードモジュールをリモートノードでプロセスして生成する方式について考察した。

４－３．プロセス起動方式

リモートノードでのプロセス起動を行なうには、大きく以下の２つの場合が考えられる。

(1)環境移動方式

リモートノードに、ユーザ環境を移動させ、そこでプロセスを起動する。

(2)要求転送方式

リモートノードでのプロセス起動要求を識別し、リモートノードに対し、その転送に必要なパラメータを転送し、プロセス起動を行なう。

以降この２つの方式について述べる。

４－３－１．環境移動によるプロセス起動

従来は、仮想端末によりリモートノードにlogin し、リモートノードのプロセスを起動する手法がとられていた。

仮想ネットワークではこの環境の移動を仮想的に行ない、リモートノードでのプロセス起動を実現する。

(1)ｃｄ（チェンジディレクトリ）

ユーザの作業環境は、ネットワークファイルシステム上の任意の位置（ワーキングディレクトリ）に存在する。

ｃｄによりリモートノードのディレクトリへの移動を可能にすることにより、ワーキングディレクトリをリモートノードに移す。これは、仮想的にリモートノードに接続することである。

これによりリモートノード上でのプロセス起動が可能になる。

(2)ｌｏｇｉｎ

ユーザの利用開始手続き時（login ）に、作業環境が設定される。その中にホームディレクトリがあり、ユーザの作業場所の基点を表わし、作業環境はここに設定される。

仮想ネットワークでは、ホームディレクトリの概念をネットワークに拡張する。物理的に端末が接続されているノードにホームディレクトリが存在しなくとも、リモートノードに対し、仮想的にｌｏｇｉｎを行ない、作業環境を設定する。

これにより、リモートノードでのプロセス起動が可能となる。

４－３－２．要求転送によるプロセス起動

４－３－２－１．起動方式

要求転送方式は、リモートファイルアクセスでも述べたように、プロセス起動に必要なパラメータをロードモジュールの実存するノードに転送することにより、リモートノードでプロセスを起動させる。

以下の手順により本方式を実現する。

(1)ロードモジュールファイルの検索

ロードモジュールファイルの実存するノードを検索する。存在ノードは、ネットワークファイルシステムの名前管理により識別され検索される。ローカル側にて処理される。

(2)存在ノードへの実行（exec）要求

存在ノードに対してプロセスのexec要求を行なう。プロセス起動に必要なパラメータを存在ノードに転送する。起動要求を行なうレベルはアプリケーション、ライブラリ、システムコール、カーネルの４つの階層にて行なうことが可能である。

(3)プロセスの起動

転送されたパラメータに従い、プロセスを起動し、その実行結果を要求ノードに転送する。

本機能をサービスするプロセスをネットワークｓｈｅｌｌ（ｎｓｈ）と呼ぶ。

図７にｎｓｈのプロセス起動方式を図示する。

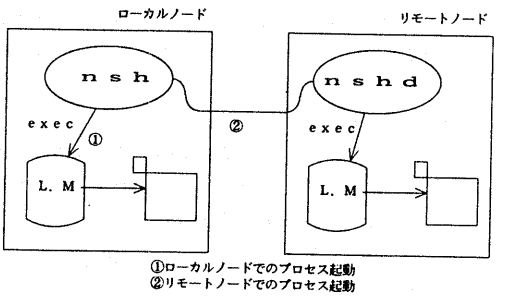


図７．リモートプロセス起動方式

本方式の特徴を以下に示す。

①カーネルに依存しないシステムコールのインターフェースレベルで実現できる。したがって移植性に優れている。

②構造がシンプルであり通信レベルでの信頼性に優れている。

③二重 Client-Serverモデルを使用しているため、実用域において効率的な処理が可能である。

4－3－2－2．コマンドインタプリタnsh

従来のコマンドインタプリタ（ｓｈ，ｃｓｈ等）を仮想ネットワークのユーザインターフェースとすると、前述したリモートプロセス起動に関して以下の問題点が存在する。

①リモートのロードモジュールはネットワークルートからのフルパス指定でないと識別できない。

②プロセス間のデータの引き渡しはファイル経由となってしまう。

③プロセスからリモート資源に対する出力が容易にできない。

これらの問題を解決するために、ｎｓｈ自体にコマンドインタプリタの機能を持たせ、いくつかの機能をネットワークに拡張している。

以下にｎｓｈのネットワーク機能を示す。

(1) パイプ

ネットワーク上のプロセス（ローカルプロセス、リモートプロセス）の出力と入力を連結し、ネットワーク上での簡単なプロセス間通信を可能とする。

(2) リダイレクション

プロセスの出力をネットワークを通してリモート資源へ切り換える。プロセスからリモート資源に対する入出力が容易となる。

(3) ワーキングディレクトリ

ワーキングディレクトリをネットワークに拡張することにより、資源の相対指定が可能となる。環境移動方式のcdと連動する。

(4) ｓｈｅｌｌ変数

①ＰＡＴＨ

コマンドの検索をネットワークを通して行なう。ＰＡＴＨにリモートノードが指定されている時は、該当ノードに対して検索する。

②ＨＯＭＥ

ホームデイレクトリをネットワークに拡張する。リモートノードにホームデイレクトリが存在する場合は、環境移動方式のｌｏｇｉｎ により、存在するノードに対しログインを行う。

5．実　現

以上述べてきた仮想ネットワークは、現在、ＵＮＩＸ Ｓｙｓｔｅｍ Ｖをベースとした０Ｓを持つ当社製ミニコンピュータＵＸ－７００上にインプリメントされている。

6．効　果

仮想ネットワークの実現により以下の効果が挙げられる。

(1) ネットワーク上の資源が、ネットワークの存在を意識することなく利用可能となった。

(2) ネットワーク上での作業場所を意識する必要がなくなった。

(3) ネットワーク上の資源の共有が容易になった。

以上から、ネットワークを意識しない開発環境の構築が可能となり、効率的な資源利用が可能となる。

7．今後の課題

仮想ネットワークは、現在インプリメント中であり、以下の問題点の解決が今後の課題となる。

(1) ネッワーク上の資源のセキュリティを考慮する必要がある。現状、１６ビットで表現されるユーザＩＤにより資源アクセス許可を行っている。これは、ローカルシステムでのメカニズムのネットワークへの拡張であり、ノード間でユーザＩＤが同一でなければならない。さらに拡張された管理機能が必要と考える。

(2) 存在ノードと走行ノードが異なる場合でのプロセス起動方式に関し検討し実現する。これによりネットワークプロセスの負荷分散へと拡張することが可能となる。

(3) 現状の仮想ネットワークは、移植性を考慮しながらもＵＮＩＸを前提としている。本方式を異ＯＳ（ＭＳ－ＤＯＳ）に適用し評価する。

今後、以上を検討しながら、さらに仮想ネットワークの研究開発に取り組んでいく。

＜参考文献＞

[1] 貫井他「ソフトウェア生産ネットワーク　－ネットワークファイルシステムの構想と実現－」　情処学会第32回全国大会 1D-2

[2] 貫井他「ネットワークファイルシステム　－ネットワークにおけるプロセスの動作環境の継承－」　情処学会第33回全国大会 3U-5

[3] 貫井他「仮想ネットワーク　－ネットワークにおける機器管理方式－」　情処学会第35回全国大会

[4] B.Walker,G.Popek etc "The LOCUS Distributed Operating System"　1983.ACM Symposium

[5] L.A.Rowe,K.P.Birman "A Local Network Based on the UNIX Operating System"　IEEE SE-8,NO2, MARCH 1982

◎UNIXは米国ベル研究所が開発し、所有権を有するOSです。

◎MS-DOSは米国マイクロソフト社が開発し、所有権を有するOSです。