マルチメディア通信と分散処理　34－8
（１９８７・７・２４）

論理型言語を用いたサービスベースシステムの実装

荻野　正　　田中　英彦

東京大学　工学部

我々は、計算機網内に分散して存在するさまざまサービス（プログラム、データ等）を扱う時に、ユーザには、その分散性・異種性を意識せずに自由に組み合わせて利用できる環境を提供し、一方各ノードでは独立にサービスの拡張ができるようなシステムについて研究を行なっている。このシステムはサービスベースシステム（ＳＢＳ）と呼ばれる。ＳＢＳでは、各計算機資源について、その外部仕様等についての記述が必要になる。仕様の記述には、適当な構造を持たせる必要があり、ここではリストを用いた方法を提案した。現在、この記述を扱う言語として論理型言語を使用したシステムの実装を行なっている。実験システムは、２台のＶＡＸ－１１／７３０からなり、ＯＳはＵＮＩＸ、システム記述言語としてｐｒｏｌｏｇとＣを使っている。実験システム上での簡単なサービス記述例及び実行例も示す。

AN INPLEMENTATION OF SERVICE BASE SYSTEM USING LOGIC PROGRAMMING LANGUAGE

Tadashi OGINO　　Hidehiko TANAKA

Department of Electrical Engineering, University of Tokyo

7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan

Abstract

In this paper, we describe the basic concept of the Service Base System (SBS) and explain the experimental system using logic programming language (prolog).

Using the Service Base System, a user can integrate and use services distributed in computer networks without knowing where the services actually exist. In the Service Base System, the specification of necessary services must be described beforehand. We also propose a new method to describe computer resources such as program and data.

1．はじめに

最近の計算機技術の発達により、複数の計算機をネットワークを介して接続し、全体として一つのシステムとして使用することは、もはや常識となりつつある。

この分野の研究は、大きく広域網に関する研究と、ＬＡＮ等の局所網に関する研究とに分類することができる。広域網に関する研究としては、異なる機種の計算機同士を接続する場合のプロトコルの変換、ファイル転送、リモートジョブエントリー等、計算機同志を接続するというレベルの研究が主であり、また、最近脚光を浴びているＬＡＮに関する研究では、その構成法や通信方法に関する研究が主流となっている。[1] しかし、様々な計算機が数十台、数百台と接続された時に、その巨大なシステムをユーザがどのように使用するかについての研究はあまり行なわれていない。

本稿では、複数台の計算機をネットワークで接続したという環境の下で、

・ユーザは、各計算機の提供する機能を、その分散性にわずらわされずに、自由に組み合わせて使用する事ができる。

・網中の各ノードでは、他の計算機とは独立に機能の拡張を行なうことができる。

等を目標として開発しているサービスベースシステム（ＳＢＳ）について発表する。

2．サービスベースシステムの概要[2]

2.1 サービスベースシステムの目標

複数台の計算機がネットワークで接続されているという状況で解決しなければならない問題は、大きく次の２つに分けることができる。

・分散性

　　計算機資源が網中に分散して存在する。

・異種性

　　各ノードの計算機に相違がある。

ＳＢＳは、

①計算機－計算機間及びユーザ－計算機間の論理的な通信をすべてサービスの要求と応答というシンプルなモデルに統一する。

②サービスに関する情報を三層ビューという構成で管理する。

事によって上記の問題を解決しようというものである。

2.2 サービスの定義

ＳＢＳでは、計算機がユーザに提供する機能をすべてサービスと呼ぶことにする。サービスは模式的に、「作用と、作用の対象となるデータを与えることによって提供される機能」ととらえることができる。（図１）

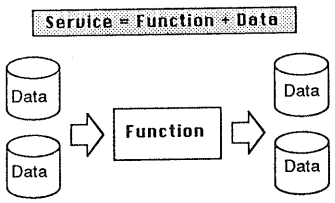

図１　サービスのモデル

ＳＢＳでは、「データ」とは、構造と静的な性質を持ったものとして定義する。構造としては、Relationとstreamに限定して考える。Relationの場合は、attribute, domain, key等を決定する事で構造を定義することができる。streamの場合は、読み書きする単位となるデータの集合や、並び方に対する規則を決定することで構造を定義する。静的な性質とは、そのデータの名前とか所有者、アクセス権、データの内容等構造に反映されない性質を指す。

「作用」とは、０個以上のデータを入力として与えた時に、１個以上のデータを出力として得るものとし、入力データとして許されるデータの記述の集合と、出力データの記述の集合及び、静的な性質に関する記述で定義する。静的な性質に関する記述とは、名前、実行環境、意味の記述等を指す。通常我々が使用している計算機では、作用はコマンドに、データはファイルに対応している。

サービスの要求とは、「作用」と「（対象となる）データの集合」を指定する事である。すべての計算機の機能が、このサービスの要求のみで実現できれば、細かい使い方の差異は吸収することが可能である。各ノードは、自計算機に存在するサービス及び自計算機を介して使用することのできるサービスに関する情報を予め持っていなければならない。そして、ユーザ或いは他の計算機からのサービス要求があった時、自計算機に存在するサービスであれば自計算機で実行し、別の計算機に存在するサービスであれば、新たにサービス要求を行なう。この時、ユーザがサービスの実際の存在場所を知らなくていいのと同様に、各計算機はどの計算機にサービス要求をすればよいかを知っているだけでよく、サービスの存在場所は知らなくてもよい（図２）。

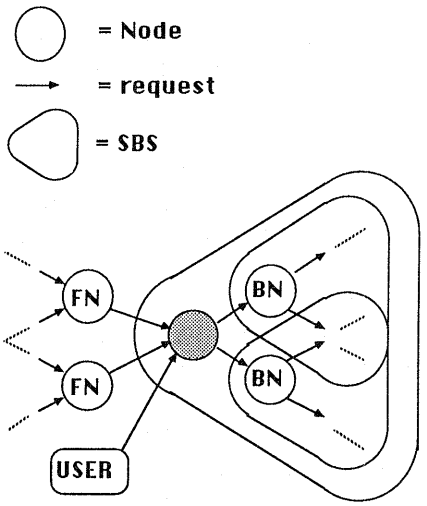

図2　SBSの網の論理モデル

2.3 三層ビュー

各計算機では、サービスに関する情報を次の3つのレベルに分けて記述、管理する（図3）。

①外部ビュー

②概念ビュー

③内部ビュー

内部ビューは、各計算機が独立に提供するサービスに関するビューであり、各計算機に1つだけ存在する。

概念ビューは、自計算機の内部ビューと他の計算機の外部ビューを統合したビューであり、分散性をここで吸収する。

外部ビューは、その計算機を使用するユーザあるいは他の計算機に見せるビューであり、それぞれのユーザあるいは計算機ごとに存在する。

この様に、サービスを三層ビューという形で管理している為、各計算機では他の計算機とは独立にサービスの拡張を行なうことができる。

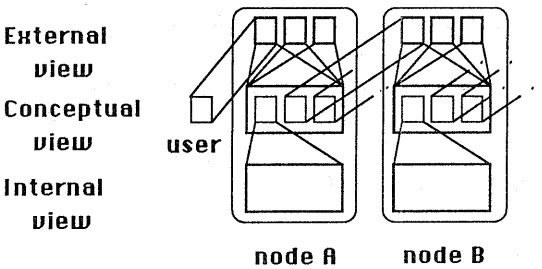

図3　3層ビュー

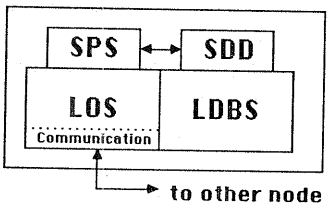

図4　SBSのノード構成

3．サービスベースシステムの構成

サービスベースシステムの各ノードの計算機の構成は図4のようになっている。各計算機には、その計算機固有のOS、及びDBSが予め存在する。これらは、その計算機に局所的な機能という意味でローカルOS（LOS）、ローカルDBS（LDBS）と呼ぶ。しかし、LOS、LDBSが分散環境を意識した機能を提供することを否定するわけではない。また、他の計算機との通信を行う為の通信機構をCMSと呼ぶ。異るコード体系の変換や、ファイル転送等の機能はCMSが提供する。

SBSでは以上の部分は予めそれぞれの計算機に存在するという仮定をおく。これらの上にSBSを構築するために、サービスの処理系とサービスの記述管理部を設ける。

サービスの処理系は、

・サービスの要求を受けとり、

・サービスの要求を分析し、

・自計算機に存在するサービスであれば、環境を設定し、

・サービスを実行する。

・他の計算機に存在するサービスであれば、サービスの要求を行ない、

・サービスの応答を待つ。

という処理を行なう。

サービスの記述管理部は、

・三層ビューの管理（サービスの検索、追加、変更、削除）

・網に関する情報（ノード名、ノードID、コストなど）の管理

・サービスの組み合わせに関する記述の管理

等を行なう。実際のデータは、LDBSに存在する。

サービスの記述管理部を記述する言語に特に制限はなく、関数型言語を用いた実験も行なった。本研究では、

「サービス記述＝計算機の機能に関する知識」ととらえ、知識処理に適している論理型言語を用いて記述するものとする。また、サービスの処理系も、記述管理部とのインタフェースを容易にするため同じ言語で記述する。

4．サービス記述[3]

4.1 サービスの記述内容

サービス記述の方法について検討する前に、どのような情報を持っていなければならないかについて考える。前述のようにサービスは作用とデータに分けて考えることができる。

まず、データについて知らなければならない情報とは、それが「どのようなデータ」であるかという情報であり、「どのような」とは、ユーザ側からみれば実験結果であるとか、英語で書いた論文であるとかいう『データの意味』を指し、計算機側からみれば、『物理的な存在場所』とか『データのフォーマット』などを指す。また作用について知らなければならない情報とは、「どのような入力データに対して、どのような処理を施して、どのような出力データを得るか」という情報であり、ユーザ側からみれば実験データの平均と分散を求めるとかソースプログムをコンパイルして実行可能なモジュールにするとかいう『作用の意味』に関する情報であり、計算機に必要なのは『入力ファイルのフォーマット』とか『実行の方法』に関する情報である。この様にユーザに提供する情報と計算機が使用する情報があり、前者をdictionary、後者をdirectory と呼ぶ。dictionaryとdirectory は厳密に区別することは困難であるが、ここではサービスに関する情報としてはdictionaryとdirectory があるものとする。

4.2 サービスの記述方法

次にdictionaryとdirectory の記述方法について考察する。サービスの記述の方法については、属性の名前とその値という表の形式で記述、管理する方法がまず考えられる。

例　Cコンパイラ

入力ファイルの形式	ソースファイル
入力ファイルの言語	C
出力ファイルの形式	オブジェクトファイル
端末との入出力	なし
存在場所	計算機１
：	：

この方法は、サービスの記述が表の形式になり扱いやすいという特徴があるが、実際には次にあげるような点で不十分であることがわかる。

・ある属性に対して複数の値をとる場合がある。
（Cコンパイラの入力として許されるのはCで書かれたテキストファイルだけでなく、例えばアセンブラのファイルでもよい。）
・属性と属性の間になんらかの関係がある場合がある。
（「入力ファイルの言語」という属性が必要になるのは「入力ファイルの形式」という属性の値がソースファイルの時だけである。）
・注目しなければならない属性の数は少ないが、どこに注目するかはサービスによって大きく異なる。（システム全体で定義しなければならない属性の数に比べて、ある特定のサービスを記述する時に必要な属性の数は少ない。）

以上の欠点を克服するために、次に示すようなサービス記述手法を考える。

4.3 リストを用いたサービス記述手法

この方法では、属性名と、その値のリストをconsしたリストでサービスを記述するものとする。

（属性名　値１　値２　…　値Ｎ）　…　（＊）

属性としては、その値としてatomをとるものと、再び（＊）の形式のリストをその値とするものの２種類を考える。これによって、ある程度複雑な構造を持った記述ができ、また不必要な属性に対してはその記述を省略できるようになる。

例１　属性Ａの値として、a1またはa2が許される場合
（Ａ　a1　a2）

例２　属性Ａの値がa1の時のみ属性Ｂの値が必要になる場合　→　ＡとＢをまとめたＣという属性を考える。
（Ｃ　（Ａ　a1）　（Ｂ　b））
（Ｃ　（Ａ　a2））

現在考えている属性の主なものを図５に示す。

例としてＵＮＩＸのＣコンパイラを起動するコマンドｃｃを考える。ｃｃを実行する時に与える入力データは、Ｃまたはアセンブラで書かれたテキストファイルであるか、すでにコンパイルされているオブジェクトファイルのどれかである。この、入力データに関する情報を上の記法で記述すると、

(arg-infs (arg-inf①) (type text) (lang c asm②))
(arg-inf③) (type object)))

属性名	属性値	内容
data-description	自然言語	データの意味
func-description	自然言語	作用の機能
arg-infs	リスト	入出力データの内容
e-name	サービス名	外部ビューでの名前
c-name	サービス名	概念ビューでの名前
i-name	サービス名	内部ビューでの名前
combination	ｐｒｏｌｏｇ	組み合わせ方
data-structure	リスト	データの構造
att-name	属性名	属性名
domain	ドメイン名	ドメイン名
type	ｆｉｌｅ属性	テキスト、オブジェクト等
lang	記述言語	Ｃ、ｆｏｒｔｒａｎ等
permission		パーミッション
format	リスト	ｆｉｌｅのフォーマット
owner	ｕｓｅｒ名	所有者
update-time	時間	変更時間
environment	リスト	環境
execute-way		実行方法

図５　主な属性

となる。②の部分は、「Ｃまたはアセンブラ」のＯＲ関係を示しており、①と③で「～テキストファイル」と「～オブジェクトファイル」のＯＲの関係を示している。

５．属性に関する記述

5.1 システムに関する知識

サービスに関する記述を上で示した方法で行なう場合、対象となるシステムで扱うすべての属性について、属性名、そのとる値、属性間の関係等を、システム管理者が予め記述しておく必要がある。この記述はそのシステムに固有のものであり、そのシステムに関する知識の一部であると考えることができる。

個々のサービスに対する記述を行なう時は、そのシステムの属性に関する記述を参照しながら操作を進めることになる。また、実際のサービスの実行の際、そのサービスの記述を調べる時にも、この属性に関する記述を参照する必要が生ずる。よって、属性に関する記述と、それを扱う部分を切り話しておくことにより、この属性に関する記述の部分を変更するだけで、異なる計算機に対するサービスの記述処理部を実装することが可能になると思われる。

5.2 属性間の関係

属性間の関係として次の２通りの関係を考える。

1. 属性間の親子関係

属性の値として（＊）の形式のリストをとる場合に、その属性と、値のリストの属性との関係を親子関係と呼ぶ。

2. 属性間の従属関係

ある属性の値が特殊な値をとる時のみ定義できる別の属性がある時、これらの属性の関係を従属関係と呼ぶ。従属関係にある属性同志は、リストの中では同じ深さに存在する。

サービス記述に使われるすべての属性の属性名、とる値、親子関係にある属性名、従属関係にある属性名をまとめたものが属性に関する記述になる。

６．実験システム

6.1 実験システムの構成

以上述べてきた方法に基ずいた実験システムを構築し、ＳＢＳの有効性の確認等を行なっている。実験システムは、当研究室の２台のＶＡＸ－１１／７３０から構成されている（図６）。このうちの１台は、必要があれば東京大学大型計算機センターのＶＡＸ８６００、Ｍ６８０Ｈとも接続することが可能である。ＯＳは、ＵＮＩＸ４．３ＢＳＤである。４．３ＢＳＤは、ネットワークに関する機能が強化されており、ｓｏｃｋｅｔという通信プリミティブを使用することにより、分散したノード間のプロセス間通信を比較的容易に行なうことができる。処理系の記述言語としては、Ｃ－ｐｒｏｌｏｇにｓｏｃｋｅｔの機能の一部を追加したｄｐｒｏｌｏｇを使用している。但し、ｄｐｒｏｌｏｇでは、ノード内のプロセス間通信しかサポートしていないので、ノード間の通信用プロセスはＣ言語で記述する。

6.2 ＬＯＳ部

ＬＯＳ部は、ＵＮＩＸをそのまま利用しているが、ｄｐｒｏｌｏｇとのインタフェースとしては、コマンドを実行するｓｙｓｔｅｍというシステム述語だけであり、それ以外の処理は行なっていない。ｓｙｓｔｅｍは、コマンドが正常に終了するとｓｕｃｃｅｓｓする述語であ

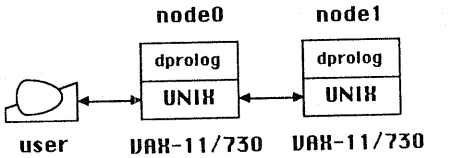

図６　実験システムの構成

る。ｓｏｃｋｅｔを使ったプロセス間通信機能は、他ノードのｐｒｏｌｏｇとの通信にのみ使われている。

6.3 ＬＤＢ部[4]

３章でも示したようにＳＢＳでは、そのノードにあらかじめＤＢＳ（ＬＤＢＳ）が存在していることを仮定しているが、実験システム上には適当なＤＢＳが存在しないので、ここでは、ＬＤＢＳをｐｒｏｌｏｇで作成した。このＬＤＢＳは最低限の機能を持つ関係データベースであり、効率等については考慮していない。

このＤＢＳではリレーションの１つのタプルが次の様な１つのファクトの形でｐｒｏｌｏｇの内部データベースに格納されている。

Ｒｅｌａｔｉｏｎ名（値１、… 値Ｎ）

各リレーションは、リレーション名と同じ名前を持つファイルにｓａｖｅ、ｌｏａｄできる。

このＤＢＳでは以下の１０個の辞書を持っている。

1. ｄｄｄ　　資源辞書
2. ｄｒｅｌ　　ｒｅｌａｔｉｏｎの定義
3. ｄｏｍａｉｎ　　ｄｏｍａｉｎの定義
4. ｓｙｓｔｅｍ－ｒｅｌａｔｉｏｎ　　ｒｅｌａｔｉｏｎ名の定義
5. ｆ－ｄ　　関数資源の定義
6. ｔ－ｄ　　テキスト資源の定義
7. ｅ－ｉｎｆ　　外部ビューの定義
8. ｃ－ｉｎｆ　　概念ビューの定義
9. ｉ－ｉｎｆ　　内部ビューの定義
10. ｎｏｄｅ　　網情報の定義

７～１０がＳＢＳの三層ビューを実現するためのリレーションであり、図７の様な形になっている。図で、

ｅ－ｎａｍｅ　　外部ビューでの名前
ｃ－ｎａｍｅ　　概念ビューでの名前
ｉ－ｎａｍｅ　　内部ビューでの名前
ｐｌａｃｅ　　存在場所
ｆ／ｄ　　関数、データ、サービスあるいはそれらの組み合わせの区別
ｄｉｃ　　dictionary記述へのポインタ（ここではファイル名）
ｄｉｒ　　directory 記述へのポインタ（ここではファイル名）

である。また、図８には実装した主なコマンドについて示す。

リレーション名：e _inf
（外部ビュー）

e _name	f/d	dictionary	directory
quicksort	f	text_e _dic1	text_e _dir1
e_data_1	d	text _e _dica	text_e _dira

リレーション名：c _inf
（概念ビュー）

c _name	f/d	place	dictionary	directory
sprit	f	vax_11a	text_c _dic1	text_c _dir1
c_data_1	d	vax_11b	text_c _dica	text_c _dirb

リレーション名：i _inf
（内部ビュー）

i _name	f/d	directory
order	f	comparison
data_1	d	data_1

リレーション名：node

node_name	socket_id
vax_11a	12
vax_11b	13

図７　３層ビューのリレーション

辞書定義	２６種類	
内訳		
リレーション定義／削除	５種類	
テキスト定義	４種類	
関数定義	３種類	
ビュー定義	８種類	
その他	６種類	
タプル操作	６種類	
例 insert		タプルの挿入
delete		タプルの削除
表示検索	４種類	
例 show-db		リレーションの表示
retrieval-key		キーによる検索
関係代数	１８種類	
例 union		和
intersection		積
join		結合
その他	８種類	
例 load-db		ファイルへのロード
save-db		ファイルへのセーブ
sort-tuple		タプルのキーによるソート

図８　ＤＢＳのコマンド

属性名	属性値	子属性	従属属性
arg-infs	list	arg-inf	
arg-inf	list	var	
		io-inf	
		data-types	
var	変数名		
io-inf	char		
data-types	list	data-type	
data-type	text		lang
data-type	char		
lang	char		

図9　入出力関係の属性表

6.4 サービス記述部

上のＬＤＢＳを使って記述部の実装を行なった。記述部の機能として実装されているのは、

・dictionary情報に関する問い合わせ
・directory 情報に関する問い合わせ
・変換サービスに関する問い合わせ

の３つの機能である

dictionary、directory に関しては実際の記述はファイルに存在し、attribute の値としては、そのファイル名がはいる。記述部では、当該ファイルを読んで適当な形になおして処理系に返す。

なお、directory の記述は４章で示した方法によっているが、dictionaryについては自然言語で記述されており、だだ単に表示するとか以外に計算機がその内容を直接扱うことはしていない。

変換サービスとは、データの情報と、サービスの入力データに関する条件が適合しない場合に、実行されるサービスである。例えば、ファイルのフォーマット変換のサービス等がこれに当たる。変換サービスは、あらかじめどのような変換をするサービスがあるかを下の様な形で登録しておく。

```
convert([data-type,dvi,ps],dvi2ps(X,Y))
```

この例は、ｄｖｉ２ｐｓというサービスが、ｄｖｉの形式のファイルをｐｓ（ポストスクリプト）の形式になおすプログラムであることを示している。

また、５章で示した属性間の関係については、各属性に対して、

・属性の値のとる範囲
・親子関係にある属性
・従属関係にある属性及び、その条件

を記述しておく。例えば、入出力データに関する情報については、図９の様に書ける。ここで、ｄａｔａ－ｔｙｐｅという属性に関する記述が２つあることに注意してほしい。１つ目の記述は属性ｄａｔａ－ｔｙｐｅの属性値がｔｅｘｔの時に従属する属性としてｌａｎｇが存在することを示し、２つめの記述は、それ以外の値をとった時には従属する属性は存在しないことを表している。このように従属属性を持つものは、複数の記述で表すことになる。この関係表を利用すると、サービス記述の作成をある程度サポートすることもできる。

6.5 サービスの処理系

処理系は、ｄｐｒｏｌｏｇで書かれたコマンドインタプリタであり、

・サービスの展開
・入出力チェック
　（ファイル転送、変換サービス等の実行）
・サービスの要求
・サービスの実行

を行なう。

7．サービスの実行例

実験システム上で、コンパイラのサービスを記述実行した例を示す。なお、簡単のため、サービス名は各ビューで同じ名前を使っている。プリミティブなサービスとして次の４つのサービスが登録されている。

ａｓ（Ｘ，Ｙ）　…　アセンブラ
ｃｃｏｍ（Ｘ，Ｙ）　…　Ｃコンパイラ
ｐｃ（Ｘ，Ｙ）　…　パスカル・コンパイラ
ｌｄ（Ｘ，Ｙ）　…　ローダ

最初の３つのサービスは入力テキストをコンパイルしてマシンコードを出力するサービスであり、その後でローダを実行すると実行可能モジュールができる。最初の３つのサービスは変換サービスとしても登録しておく。

データとしては次のものがある。

ｔｅｓｔ１．ｐ…パスカル・プログラム（ノード０）
ｔｅｓｔ２．ｃ…Ｃプログラム（ノード１）
ｔｅｓｔ３．ｓ…アセンブラ・プログラム（ノード０）

サービス記述の一部を図10に、サービスの実行例を図11に示す。テキストデータのプログラム言語によって適当なコンパイラが起動されること、データの存在場所によって適当にファイル転送を行なっていることがわかる。

```
/* directry of pc(X,Y) */
[c_directory, [c_name, pc(X,Y)],
              [c_fd, f],
              [arg_infs, [arg_inf, [var, X],
                                   [io_inf, in],
                                   [data_types, [data_type,text],
                                                [lang, pascal]]],
                         [arg_inf, [var, Y],
                                   [io_inf, out],
                                   [data_types, [data_type, object]]]],
              [exec_way, pc(X,'-c','-o',Y)]]).

/* directory of test3.s */
[c_directory, [c_name, 'test3.s'],
              [c_f/d, d],
              [data_types, [data_type, text], [lang, assembler]]]).

/* convert service ccom(X,Y) */
convert([data_types, [data_type, text, object], [lang, c, []]], ccom(X,Y)).
```

図10　サービス記述例

8．結論

ＳＢＳの実験システムを構築し、簡単な例について実行を確認した。システムの効率等については測定していない。

今後の課題としては、実験システム上で、

・サービス記述／管理用ｔｏｏｌの作成

・サービスの実装によるサービスの記述方法の検討

を行なっていく予定である。また、属性に関する情報をメタな情報としてまとめることにより、ＳＢＳの処理系を変更せずに、このメタな情報の変更のみで別の計算機に対応する方法についても検討する予定である。

さらに、現在はユーザが利用しているサービスの意味に関する情報を計算機に扱える形にすることにより、ユーザの要求から推論をして、最適なサービスを選択して実行することも可能になると思われる。

謝辞

実験システムの実装に協力して下さった卒論生の入江君、研究生の石崎氏に感謝いたします。

参考文献

［１］Stalling,W.,“Local Networks”,Computing Surveys,Vol.16,No.1,March,1984

［２］荻野他、「論理型言語を用いたサービスベースシステム」、ＩＮ８６－１３０

［３］荻野他、「サービスベースシステムにおけるサービス記述手法」、第３４回情報処理学会全国大会、１Ｙ－１

［４］石崎他、「サービスベースシステムのためのデータベース機構のの実装」、第３４回情報処理学会全国大会、１Ｙ－２


```
| ?- intp(ld('test1.p', 'p.out'), X, Y).
convert pc(test1.p,temp0)
exec ld(temp0,p.out)

X = [pc(test1.p,temp0)]
Y = ld(temp0,p.out)

yes
| ?- intp(ld('test2.c', 'c.out'), X, Y).
rec file : test2.c from 1 new name temp1
convert ccom(temp1,temp2)
exec ld(temp2,c.out)

X = [ccom(temp1,temp2)]
Y = ld(temp2,c.out)

yes
| ?- intp(ld('test3.s', 's.out'), X, Y).
convert as(test3.s,temp3)
exec ld(temp3,s.out)

X = [as(test3.s,temp3)]
Y = ld(temp3,s.out)

yes
```

図11　サービス実行例