(35)

「LAN／マルチメディアの応用と分散処理」シンポジウム　昭和59年10月

サービスベースシステム

－　分散システムの統合に向けて　－

深沢友雄　田中英彦　元岡達

東京大学　工学部

社団法人 情報処理学会

サービスベースシステム

－　分散システムの統合に向けて　－

深沢友雄　田中英彦　元岡達

東京大学　工学部

1．はじめに

近年の計算機間通信技術の進歩により、一人のユーザが、ＬＡＮや広域網内の複数の計算機を同時に利用することが可能となっている。この時、

・ユーザに対しては、分散性を意識することなく網中の種々のサービス（データやプログラム）を利用できる環境を提供することが望ましい。
・一方、各計算機では、網の形態や、他計算機の管理機構とは独立にサービスの蓄積・管理ができ、かつ網の構成要素として、自由に組入れることもできる網の構成方法が望まれる。

ＬＡＮでは、前者に重点がおかれ、広域網では、主に後者を達成している傾向にある。

将来は、個人用計算機から、汎用大型計算機に至るまで、様々な種類の計算機からなる網が構成されることが考えられ、両者を満足する様な網の構成方法、網上のサービスの管理方法が望まれる。

本研究では、上記の目的を達成するシステムを「サービスベースシステム」と呼び、その構成方法の検討、実験システムの作成を試みてきたので、ここに報告する。

2．サービスベースシステムの概念［1］

2.1　ＳＢＳの対象とするサービス

計算機の提供するサービスは、「データ」に「作用」をほどこすことである。ＳＢＳでは、データとしてファイルやデータベース等を扱い、作用として、ロードモジュールや各計算機の提供するコマンド等を取り扱う。

2.2　ＳＢＳにおける網のモデル

ＳＢＳでは、各計算機は、その計算機が提供するサービスと、その計算機を通して利用可能な全ての計算機が提供するサービスを、ユーザが区別なく利用できる環境を提供する。ここで、或る計算機Ｎに着目した場合、Ｎに要求を出す計算機をＦＮ（Ｆront　Ｎode）と呼び、Ｎが要求を出す先の計算機をＢＮ（Ｂack Ｎode）と呼ぶ。ＳＢＳでは、ＢＮは、ＢＮの提供するサービスと、ＢＮを通して利用可能なすべての計算機の提供するサービスを、計算機Ｎが区別なく利用できる環境を提供する様に構成される。即ち、ＢＮが計算機Ｎに対してＳＢＳを構成しているわけで、これを計算機ＮのＢＳＢＳ（Ｂack Ｓervice Ｂase Ｓystem）と呼んでいる。計算機Ｎは、ＦＮから見るとＦＮのＢＮに見え、計算機Ｎの構成するＳＢＳは、ＦＮからは、ＢＳＢＳに見える。これらの関係は、形式的には、以下の様に書ける。

$BNS(N)=\{J \mid N \rightarrow J\}=\{N1,\cdots,Nm\}$

$BNi(N) \in BNS(N)$

$BSBSi(N)=SBS(BNi(N))$

$SBS(N)=\{N\} \cup \bigcup_{1 \le i \le m} BSBSi(N)$

$FNS(N)=\{K \mid K \rightarrow N\}$

$FNi(N) \in FNS(N)$

但し、Ｉ→Ｊは、計算機Ｉから計算機Ｊにサービス要求が可能であることを示す。ＳＢＳでは、この関係（上記の→の関係）による論理的な網を考えるが、これは、一般的には、物理的な網と独立に構成される。ＳＢＳにおいて「サービス要求が可能である」ことの定義は後で述べる。

ＢＳＢＳi(Ｎ)は、ノードＮのＢＳＢＳの１つを構成するノードの集合を表す。

ＢＮＳ（Ｎ）は、ノードＮの全てのＢＮからなる集合（直接サービス要求を行なう要求先

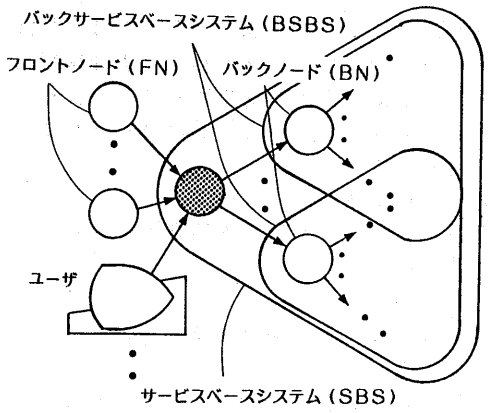


図1　SBSにおける網の論理モデル

の計算機の集合）を表す。
BN_i（N）は、ノードNのBNの1つを表す。
SBS（N）は、ノードNが構成するSBS内に含まれる全てのノードの集合を表す。これらの式から、SBS（N）を構成する為には、各Nは、全てのBN（N）上の情報だけを知れば十分であることが暗黙に示唆される。
FNS（N）は、ノードNの全てのFNからなる集合を表す。
FN_i（N）は、ノードNのFNの1つを表す。

2.3 各ノードの論理構成

SBSの各ノードの主な構成要素は、「サービスの処理系」と「サービスの定義部」である。

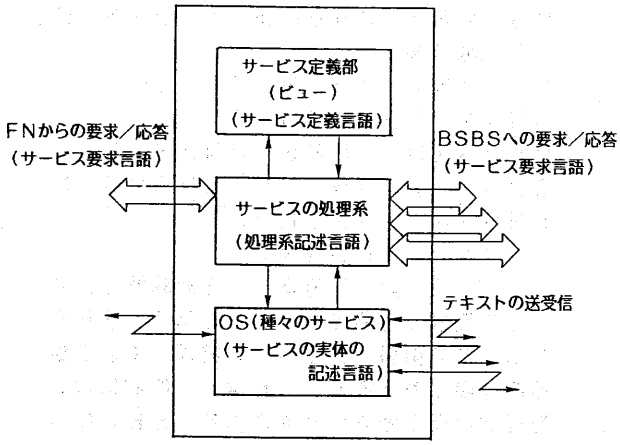


図2 各ノードの論理構成

・処理系の機能

処理系は、定義部の記述に基いて、サービス要求を解釈し、自ノードが提供するサービスは、自ノードで実行し、BSBSに存在するサービスは、BNにサービス要求を行ない応答を解釈する。この様にSBSでは、各計算機が自動的にサービスの存在する計算機にサービス要求／応答を繰り返すことにより、分散して存在するサービスを組合わせたサービスを実行する。

・定義部の構成

SBSではサービスを次の3層にわけて記述する。

i）内部ビュー

各計算機が単独で提供するサービスの記述

ii）概念ビュー

自計算機の提供するサービスと、BSBSの提供するサービスを統合したサービスの記述

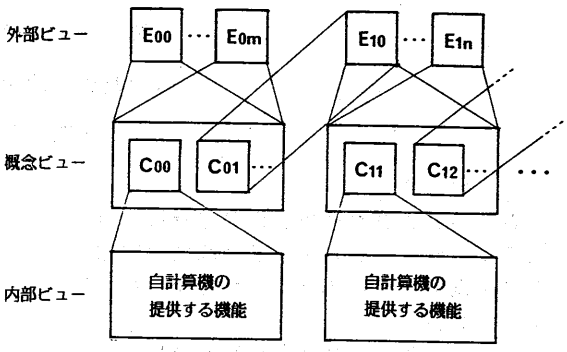


図3 サービスの定義部の構造

iii）外部ビュー

自計算機を通してFNに提供するサービスの記述

i）は、

①各計算機のOSの提供する機能（コマンド等）

②各計算機のOSに管理されている諸サービス（ロード・モジュール）

③処理系が直接実行するサービス（組込みサービス）

からなる。

ii）は、①基本サービスと②合成サービスとに分類できる。①は、自計算機の内部ビュー或いは、BNの外部ビューに記述されたサービスを、概念ビュー上で定義したサービスであり、②は、自計算機の概念ビュー上のサービスを組合わせて、新たに定義するサービスである。

iii）も、①基本サービスと②合成サービスに分類できる。①は、自計算機の概念ビュー上で定義されているサービスの定義であり②は、自計算機の外部ビュー上のサービスを組合わせて定義するサービスである。

以上のように、各計算機の概念ビューは、自計算機の内部ビューとBNの外部ビューから構成される。BNでは、同様に3層にわけてサービスが記述されているので、BNの外部ビュー上には、BSBSが自計算機に提供する全てのサービス記述されていることになる。即ち、各ノードは、自計算機に関する情報と、BN上の情報だけを知っていればよい。

前節で述べた、或るノードが「SBSを構成している」とは、外部ビューを提供することであり、「サービスを要求することができ

る（→）」とは、他計算機の外部ビューを自計算機上の概念ビュー上で定義しているということになる。この様にビューを構成することにより、前節で述べた論理的な網を構成することができる。

2.4 ＳＢＳにおける言語

ＳＢＳを実際に作成する為には、以下の言語が必要である。

ｉ）処理系記述言語
ii）サービスの実体の記述言語
iii）サービス要求言語
iv）サービス定義言語

ｉ）は、
・ＯＳとのインタフェース
・下位通信機構とのインタフェース
・サービスの管理制御
・サービス要求言語の解釈

等を記述し、処理系を作成する為の言語である。

ii）は、各計算機が提供する言語プロセッサに応じて、サービスの実体を作成する為に用いる。

iii）とiv）はＳＢＳ特有の言語である。

iii）は、既存の計算機のコマンド体系に相当するものであり、
・対話性
・記号処理向き

であることが要求される。本研究では、関数型言語（Ｌisp 的な言語）と、論理型言語（Ｐrolog 的な言語）について検討している。

iv）は、各ビューを記述する為の言語であり、各ビュー上の呼び出し名（サービス名）と、その定義体との対応を与える機能が必要となる。定義するサービスが基本サービスの場合は、定義体は、別のビュー上での名前となる。合成サービスを定義する場合の定義体の記述は、サービス要求言語による記述と似ており、プログラミング的な作業が必要となる。一般にiii）とiv）は独立でよいが、本研究では、統一的な言語で記述している。即ち、サービス定義言語で合成サービスを記述する時の定義体の記述と、サービス要求言語は、同じ言語を用いている。

3．関数型言語に基くＳＢＳの構成［5］

本章では、サービス要求言語として、関数型言語を用いた場合のＳＢＳの構成について述べる。主に処理系間の通信と、処理系の動作について、実装の経験を中心に述べる。

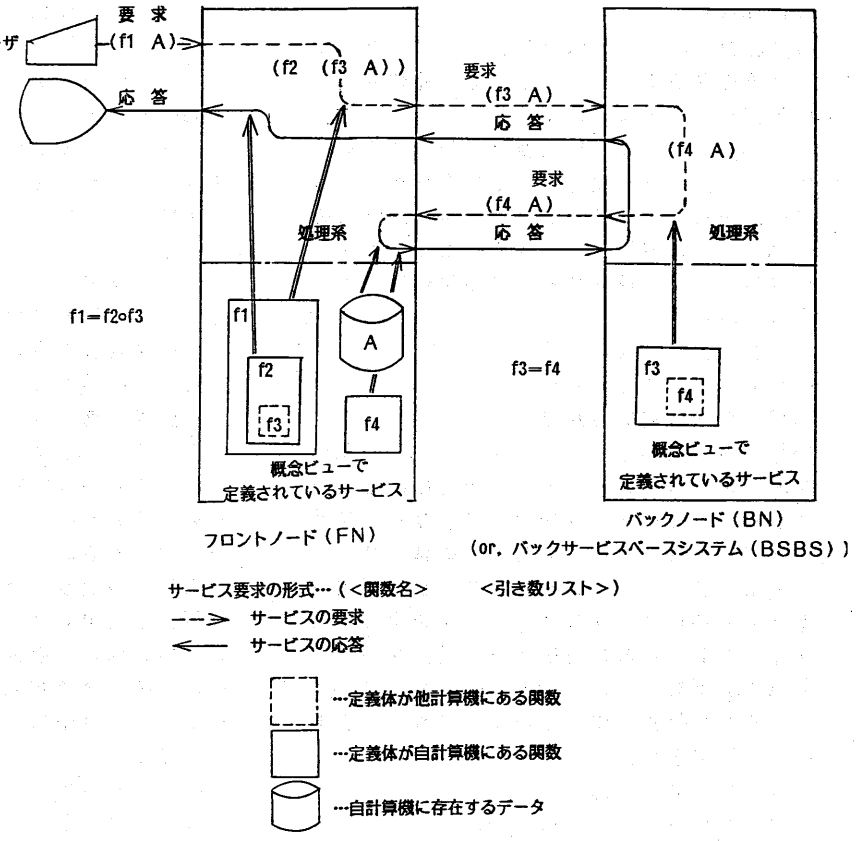


図4　計算機間の論理的な通信（要求と応答）の例
Ｌisp に基くシステムの場合

3.1 関数型言語による通信

サービス要求言語を関数型言語とし、サービスの要求と応答を、関数呼出しと、return-valueに対応させることにより、分散して存在するサービスの実行を自然に行なうことができる。この時の計算機間の通信の様子を図４に示す。

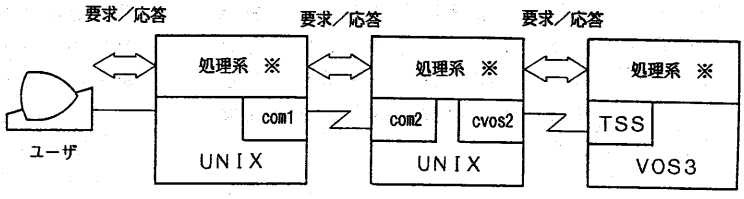


図5 実装システム構成

※処理系記述言語は、Franz Lisp（UNIX）、UTILISP(VOS3)
但し、論理型言語に基くシステム構成では、C－Prolog
OSとのインターフェースは、C言語

3.2 処理系の構成

処理系記述言語には、C言語とLisp を用いて、実験システムを作成した。[4]

C言語は、OSとのインタフェース、通信機能とのインタフェースを記述する。これにより、Lisp に提供する基本機能は、

①system
②send
③receive

の３関数である。①は、OSとのインタフェース用の関数であり、②と③は、通信機能とのインタフェースである。

Lisp では、サービス要求言語を読み込み、ビューの記述を参照して、それを解釈・実行する。即ち、合成サービスの場合は、それをLisp のsemantics に従って基本サービスに分解していく。基本サービスの場合は、それぞれに応じて、

i ）Lisp 関数の起動
ii）他計算機との通信
iii）OSの機能の呼び出し

を行なう。このうち、i ）は、Lisp の機能をそのまま用い、iii ）は、上述の①を用いる。ii）では、②と③を用いて、サービス要求と応答を行なう。この時一般には、サービス要求の結果として、新たなサービス要求が返ってくる場合があり（サービス要求のネスティング）、これに対処する必要がある。この為に、実装では、Lisp のcatch とthrow という制御構造を用いている。これらの実現例を以下に示す。

・サービス要求の形式

```
(<s-name>  . <arg-list>)
```

<s-name>…要求するサービス名
<srg-list>…引き数リスト

・サービスの応答の形式

```
(throw <return-value>)
```

・サービス要求後の処理

```
(catch
  (prog  ()
    loop
    (print (eval (read)))
    (go  loop)))   …(＊)
```

(read)…相手からのメッセージを読む。
(eval)…メッセージの評価（解釈）
(print)…評価した結果の相手への応答
(catch <S－式>)…<S－式>の中で(throw …）という関数が評価されると抜ける。

（＊）を本システムでは「フレーム」と呼んでいる。BNにサービス要求を行なった後、サービス要求を下計算機は、フレームを作って待つ。相手からのメッセージが新たなサービス要求の場合は、このフレーム内で処理される。要求がネスティングしている場合は、それぞれの要求に応じて、別々のフレームを作成するので、任意のレベルのネスティングが可能である。

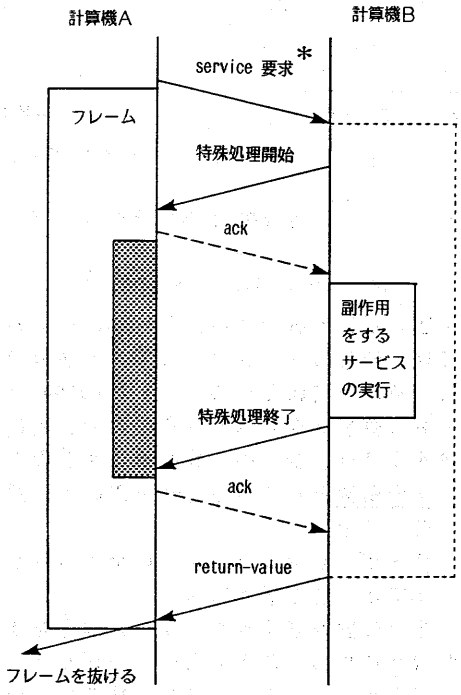
・副作用の取り扱い

関数型言語では、ファイル操作や、外部との入出力は、「副作用」として扱われる。SBSで特に考慮する必要のある副作用は、サービスの実行中に、外部と入出力を行なう場合であり、

①ファイル転送
②サービス要求／応答用の通信路を用いての入出力

等に対して特殊扱いが必要となる。これらの為に、本システムでは、副作用を伴なうサービスを受ける側で、

i）特殊サービス開始
ii）特殊サービス終了
の２種類の特殊なサービスを用意し、ｉ）とii）の間は、計算機間に及ぶサービス要求のネスティングを禁止している。実験システムでは、特殊サービスとして
①file
②transparent
の２つの属性を用意した。①は、通信路への出力を一時ファイルで受けることを示し、②は、ＦＮとＢＮを透過的に接続することを示す。これらの属性は、サービスを提供する側でサービスを定義する時に指定する。①、或いは、②が指定されたサービスを実行する時は、処理系が自動的に、サービス提供者からサービス要求者に対して、副作用のための特殊処理をする様に要求する。この時のプロトコルを図６に示す。



特殊処理開始：　f-open　, t start
〃　終了：　f-close , t-end
＊　Aでは、Bに定義体があるということだけ定義
Bで、副作用があることを定義
（特殊処理に関する指示も定義）

図6　計算機Aが、計算機Bに、副作用を伴うサービスの要求をした場合のプロトコル

3.3　サービスの記述

分散サービスを統合するにあたって、概念ビューの働きが重要なので、本節では、実験システムの概念ビューの記述について述べる。尚、実験システムは、ユーザー人用なので、概念ビューと外部ビューを区別していない。

i）基本サービスの定義
・自計算機の提供するサービスの定義
（defs　＜c-sn＞　［＜i-sn＞］　［＜se-type＞］）
但し、
＜c-sn＞：概念ビュー上の関数の呼び出し名
＜i-sn＞：内部ビュー上の関数の呼び出し名
（省略すると＜c-sn＞と同じ）
＜se-type＞：：＝ transparent｜file
副作用を行なう場合相手に行なってほしい特殊動作の種類
・他計算機の提供するサービスの定義
（defe　＜c-sn＞　＜hostid＞　［＜e-sn＞］
［＜arg-type＞］）
但し、
＜c-sn＞：概念ビュー上の関数の呼び出し名
＜hostid＞：定義体の存在する計算機の識別名（ＢＮの名前）
＜e-sn＞：内部ビュー上の関数の呼び出し名
（省略すると＜c-sn＞と同じ）
＜arg-type＞：：＝e ｜eq
引数の渡し方…
e （eval）：評価して渡す
eq（eval quote）：評価しないで渡す
ii）合成サービスの定義
Ｌisp の関数定義と同じ
（putd　＇＜sn＞　＇＜body＞）
＜sn＞…関数名
＜body＞…λ-expression

3.4　サービスの定義・実行例

図５の構成で、ＶＡＸ－11／780 から、ＵＮＩＸとＶＯＳ３の提供するサービスを組合わせて利用する例を図７に示す。

４．論理型言語に基くＳＢＳの構成[6],[7]

前章では、サービス要求言語を関数型言語とし、関数呼出しとreturn-valueをサービス要求と応答に自然に対応させることにより、処理系を構成する方法を示した。ＳＢＳの目的の１つである「ユーザインタフェースの向上」という面から考えると、サービス要求言

図7　サービスの定義・要求例

サービスの定義
UNIX側
　(defe lists 1)
　(defs grep)　M 280Hのこと
VOS3側
　(defs lists　(file))　下線はユーザの入力

文字列“LOAD”がファイルに含まれているVOS3上のファイル名の表示

```
-> (grep 'LOAD (lists '%))
PO          760          30      LD0021    A2700.CLIB.LOAD
PO          684          17      ARCHIV    A2700.CLIB.Z.LOAD
PO          779         392      ARCHIV    A2700.FORLIB.LOAD
PO           19           6      ARCHIV    A2700.OFILT.LOAD
PO           19          14      ARCHIV    A2700.PASLIB.LOAD
```

lists …ファイル名の出力（VOS3の提供するサービス）
grep…文字列の探索（UNIXの提供するサービス）

語とサービス定義言語をハイレベルな言語に設定し、ユーザインタフェースに優れたSBSを構成することが望ましい。そこで、サービス定義言語として、最近注目を集めている、論理型言語を用い、サービスに関する定義を記述すれば、ユーザは簡単な要求で、望むサービスを得ることができるであろう。ここで、サービス要求言語は、必ずしも論理型言語にする必要はないが、使い易さ、処理効率、処理系の作成しやすさから、サービス要求言語も論理型言語に基くシステムの作成を試みたので、本章で紹介する。

4.1　サービスの分散と論理型言語

論理型言語向きSBSでは、サービスは、述語で定義される。SBSでは、サービスが分散して定義されているので、論理型言語に基くSBSでは、「述語」が分散して存在することになる。ユーザは、分散して存在する述語を自由に組合わせたサービス要求を行なう。この時の計算機間の通信の概念図を図8に示す。関数型言語では、サービス要求と応答が、関数呼び出しと、return-valueに自然に対応したが、論理型言語では、サービスの処理の結果の転送方法が問題となる。即ち、サービス要求は、ゴールの転送と自然に対応がつくが、要求をした側は、処理の結果として、変数のユニファイされた情報と、真偽値の2種類の情報が必要となり、これらの転送が必要となる（図8のS*，Q*等）。ゴールの転送に関しては、ゴール全体の単位で要求する方法と、リテラル単位で要求する方法とが考えられるが、今回は、後者についてのみ検討してみた。SBSにおけるこれらの動作、即ち、サービス要求の解釈・実行と、計算機間の通信は、処理系が行なう。

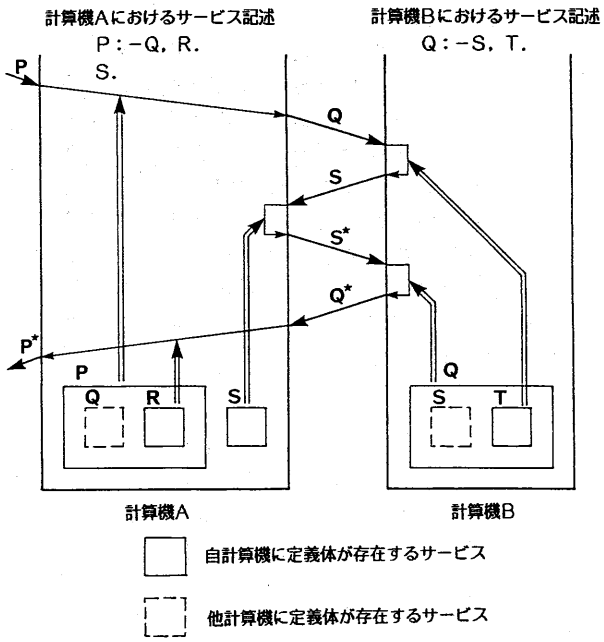


図8　論理型言語に基く処理系間の通信

4.2　処理系間の通信

4.2.1　論理型言語に基く計算機間通信

本システムのサービス定義・要求言語は、エジンバラ大学で開発されたC－Prologをもとにしたので、以下、C－Prologに基く記法を用いて説明する。Prologによる通信では、変数のユニファイに関する情報をどう伝えるかが問題となる。C－Prologでは、変数に直接アクセスできないので、変数を転送する時は、全ての変数を或る特定のファンクタの引数として扱う方法を用いた。更に、解を返す場合、

・サービス要求（ゴール）の全ての解が求まってから解を返す方法
・1つの解が求まったらすぐに解を返す方法

が考えられる。前者の場合、処理がとまらない場合が生ずることがあり、後者の場合は、通信回数が増加することになる。通信の簡便さの点から、今回は、後者の場合について検討した。

4.2.2　処理系の動作と定義部とのリンク

処理系が処理を進めている途中で、他計算機のサービスが必要となった場合、図9の形式のサービス要求・応答が行なわれる。他計

図9　サービス要求・応答の形式

a）サービス要求…setof（varlist（<<goal>中の変数の並び>），<goal>，<解用変数>）．
　但し、<goal>は、他計算機で処理するゴール
b）サービスに対する応答…[varlist（<solution-1>），varlist（<solution-2>），…]．
　但し、<solution-i>は、要求された<goal>の処理の結果ユニファイされた変数の並び

図10　サービスの定義部とのリンク

<s-name>（<arg-list>）：－external（<es-name >（<arg-list>），<hostid>）．
　但し、<s-name>は、他計算機で定義されているサービスの呼び出し名、<arg-list>は、引数リスト、
　　<es-name >は、他計算機上での当サービスの呼び出し名、<hostid>は、他計算機の名前

算機で実行されるサービスも、全て、自計算機のサービス定義部に記述されている必要がある。本システムでは、サービス定義時に、図10の様な定義を行なう。処理系で、externalという述語を用意することにより、サービス定義言語とのリンクを行なう。述語externalは、第１引数を図９-a）の<goal>として、サービス要求を行ない、図９-b）の形式の応答を順次、要求したゴールの変数とユニファイする。相手からの応答が、新たなサービス要求の場合は、サービスを処理して、結果を返す。計算機間にわたるサービス要求のネスティングは、任意のレベル可能である。

4.3　サービスの記述

関数型言語同様、概念ビューの記述について述べる。

i）基本サービスの定義

内部ビュー上では、サービス名をキーとするrelationで表現できるが、現在は、relationのデータを直接扱う事はできないので、

service（サービス名、属性名、属性値）というclauseをassertする事により管理してる。service の定義のために、以下の述語を用意した。

①自計算機の提供するサービスの定義

assert-int（<c-sn>，<i-sn>，<atr-list>）．

但し、

<c-sn>：概念ビュー上のサービス名

<i-sn>：内部ビュー上でのサービス名

<atr-list>：：＝（<atr-name>（<value >），<atr-list>）

<atr-name>：：＝place ｜map ｜out ｜arg-type

<value >：各サービスの<atr-name>の値

②他計算機の提供するサービスの定義

assert-ext（<c-sn>，<e-sn>，<hostid>，<atr-list>）

但し、

<c-sn>，<arg-list>：①と同じ

<e-sn>：ＢＮの外部ビュー上のサービス名

<hostid>：定義体の存在する計算機の名前

・サービスの定義例

自計算機のＯＳが提供する flist（引数１つ、画面に出力がでる）というサービスをfl という概念ビュー上の名前で定義する場合、ユーザは、例えば、

assert-int（fl（Ｘ），flist （Ｘ），[out （stdout）]）．

と定義する。この時、

service （fl（Ｘ），place ，int ）．

service （fl（Ｘ），map ，flist （Ｘ））．

service （fl（Ｘ），out ，stdout）．

という３つのclauseがassertされる。

ii）合成サービスの定義

Ｃ－prologの述語の定義と同じ方法で定義する。

4.4　サービスの定義・実行例

図11に、或るノード（n0）に着目した場合のサービスの要求・応答のトレース結果を示す。

５．応用システム例

図５のＶＡＸ－１１／７８０とＭ２８０ＨからなるＳＢＳ上で、

i）遠隔データベース検索システム

ii）プログラム開発環境支援システム

等を開発している。（ i）は、ＶＯＳ３で提供している文献データベースＥＮＧを利用できる環境を提供する。ＥＮＧとは、東大大型計算機センタ（ＶＯＳ３）で提供している、工学全般に関する文献情報のデータベースである。（ii）は、ＶＯＳ３上で、コンパイル、実行するプログラムを、ＵＮＩＸ上のエディタで編集する機能等を提供する。

6．考察・検討

6.1 効率

サービスの処理にかかる時間は、
①通信にかかる時間
②処理系の処理のオーバヘッド
③サービスの実行時間
の和となる。これらの比は、サービスによって異なるが、実験システムでは、①がもっとも多い。

①は、SBSの下位の通信機能による部分である。しかし、サービスの要求と応答の回数を減らすといった最適化は、SBSのレベルで可能であり、これは、今後の課題である。SBSにすることによるオーバヘッドは②であるが、実験システムでは極めて小さい。

6.2 定義部の管理［2］

定義部の管理方法について詳述しなかったが、これは、SBSの扱う「データ」とみなすことができる。SBSでは、データモデルとして、関係データベースを考えているので、定義部をrelationで表現すれば、SBSのデータ管理機構で管理ができる。現在、データの管理機構と、定義部のrelationによる表現を検討中である。

6.3 サービスの並列処理［3］

・速度の向上
・並列処理によって実現できる機能の提供

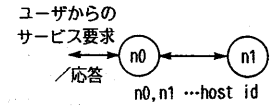
の為に、並列処理をする必要がある。

これを実現するにあたって、
①処理系でのみ並列処理を行なう
②サービス定義・要求言語で並列性を陽に記述する
の2つのレベルが考えられる。現実験システムでは、ファイル転送や、副作用の取り扱い等の最低限必要な機能を提供する為に、処理系内で対処している。しかし、サービスの記述力の向上や、効率などの点から、②の機能を提供することを検討している。

6.4 サービス定義／要求言語

計算機間通信との対応のよさから関数型言語に基くシステムの構成を試み、一方、ハイレベルなユーザインタフェースを目指して、論理型言語に基くアプローチを試みた。実験システムでは、サービス要求言語と定義言語として統一した言語を設定したが、これらをそれぞれ実現に適した別々の言語を設定することも可能である。更にこの時、計算機－計算機間と人間－計算機間の言語を異なるものに設定することも可能であり、今後の検討課題のうちの1つである。また、関数型、論理型言語以外の言語の選択に関しても検討の余地がある。SBSでは、「サービスの要求と応答」という抽象的なインタフェースのみを前提としているので実際の言語の設定に関しては自由であり、特定の計算機－計算機間のみで共通語があればよく、別の計算機－計算機間では異なる言語を使うことも考えられる。

図11 サービスの定義・要求例




```
| ?- d_app(X,Y,[a,b]).
(T:n0->u)
X = []
Y = [a,b] (T:u->n0);
(T:n0->n1)setof1(varlist(_13,_1),d_app(_13,_1,[b]),_25).
(T:n1->n0)setof1(varlist(_25,_4),d_app(_25,_4,[]),_35).
(T:n0->n1)[varlist([],[])].
(T:n1->n0)[varlist([],[b]),varlist([b],[])].
(T:n0->u)
X = [a]
Y = [b] (T:u->n0);
(T:n0->u)
X = [a,b]
Y = [] (T:u->n0);
(T:n0->u)
no
| ?-
```

網の論理構成

____ ユーザの入力
_ _ _ システムの出力

n0,n1 各ノードでのサービスの定義（注）

```
d_app ( [], X, X).
d_app ( [H|X], Y, [H|Z] ) :- ext_app (X, Y, Z).
assert-ext ( ext_app (X, Y, Z), d_app (X, Y, Z), 1, []).
```

各niから見たBSBSの識別名

（注） d_appは、ext_appが d_appだと、appendの定義と同じ。
ext_appは、他の計算機では、d_appだから、結果的に d_appは、appendと同じ。

7．おわりに

本稿では、分散処理システムの構成方法の1つとして、サービスベースシステムの考え方を提案し、その具体的な実現方法について紹介した。サービスベースシステムの構築のためには、残された問題が幾つかあるが、将来、様々な形態の計算機を接続し、高機能なサービスを使い易い形で提供する為には、サービスベースシステム的な網の構成方法が有効であると考える。

＜ 参 考 文 献 ＞

［1］深沢、田中、元岡、「サービスベースシステムの概念と基本構成」、電子通信学会、電子計算機研究会、EC 82-44、1982.10.
［2］矢部、深沢、田中、元岡、「サービスベースシステムにおけるデータ管理についての一考察」、第27回情処全大、1J－2、1983.10.
［3］深沢、田中、元岡、「サービスベースシステムにおける並行処理」、第28回情処全大、4D－8、1984.3.
［4］荻野、深沢、田中、元岡、「サービスベースシステムの実装」、第28回情処全大、4D－10、1984.3.
［5］深沢、荻野、田中、元岡、「関数型言語に基づくサービスベースシステムの構成」、電子通信学会、情報処理ネットワーク研究会、IN 84-37、1984.8.
［6］深沢、荻野、田中、元岡、「論理型言語向きサービスベースシステムの構成」、第29回情処全大、6H－6、1984.9.
［7］荻野、深沢、田中、元岡、「サービスベースシステムにおける論理型言語向きサービス記述」、第29回情処全大、6H－7、1984.9.