関数型言語に基くサービスベースシステムの構成

SERVICE BASE SYSTEM BASED ON FUNCTIONAL LANGUAGE

分散システムの統合に向けて

TOWARDS THE INTEGRATION OF DISTRIBUTED SYSTEM

荻野 正　　深沢友雄　　田中英彦　　元岡 達

Tadashi OGINO, Tomoo FUKAZAWA, Hidehiko TANAKA, Tohru MOTO-OKA.

東京大学 工学部

Faculty of Engeneering, University of Tokyo

The purpose of Service Base System is to integrate distributed services in the network with keeping the autonomy of each system in the network. Each computer communicates with each other, when it is needed in processing the service which is requested by the user in the mixed form of distributed services. To implemnet the SBS, we used lisp as the interface language between computer and computer, and human and computer. In this paper, the organization of service base management system, and the implemention of an experimental system are described.

1. はじめに

将来の計算機システムでは、ユーザは、蓄積されたサービスを組合わせて使用するといった利用形態が増加すると思われる。こういった計算機で、網を構成した場合、ユーザは、サービスの分散性を意識することなく、網中のサービスを組合わせて利用できることが望ましい。一方、拡張性を考えると、各計算機で独立にサービスの蓄積が行なえることが望ましい。そこで、計算機網中に分散して存在する種々のサービス（プログラムやデータ）をユーザに提供する時に、各計算機の独立性を保ちつつ、かつ、ユーザには分散性を意識させないユーザインタフェースを提供することを目的としたシステムをサービスベースシステム（ＳＢＳ）と呼び、研究を進めている。[5]

ＳＢＳでは、計算機－計算機間通信、及び、人間－計算機間通信をすべて、サービスの要求と応答として取り扱っている。このサービスの要求と応答を行なう言語として、関数型言語を用いたＳＢＳの実験システムを構成したので、本稿で紹介する。

2. サービスベースシステムの概念

2.1 サービスのモデル

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。したがってＳＢＳでは、サービスを「データ」と「作用」にわけて取り扱う。また、データと作用を組合わせた「サービス」の単位でも取り扱う。

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

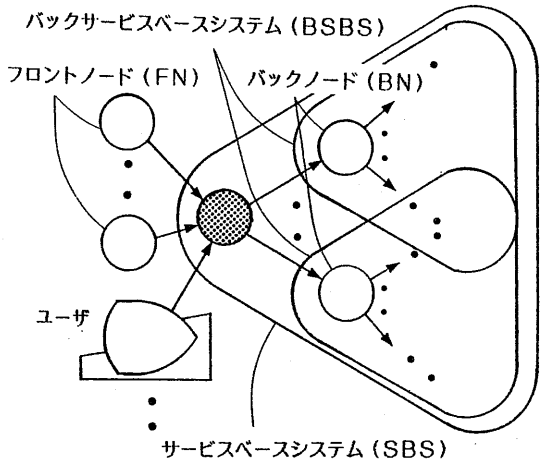


図1 SBSにおける網の論理モデル

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。

$BSBS_i(N) = SBS(BN_i(N))$

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

$BN_i(N) \in BNS(N)$

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

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

$FN_i(N) \in FNS(N)$

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（上述の→の関係）による論理的な網を考えるが、これは、一般的には、物理的な網とは独立に構成される。尚、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の計算機の集合）を表す。

ＢＮi(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。

ＦＮi(Ｎ)は、ノードＮのＦＮの１つを表す。

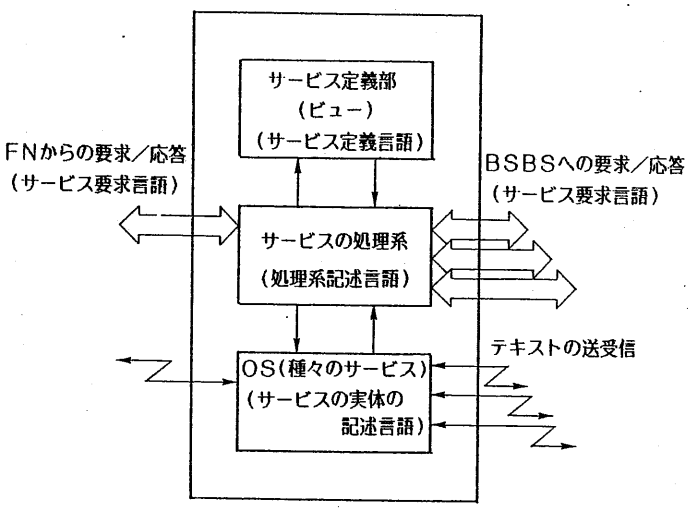


図２　各ノードの論理構成

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。

・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

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

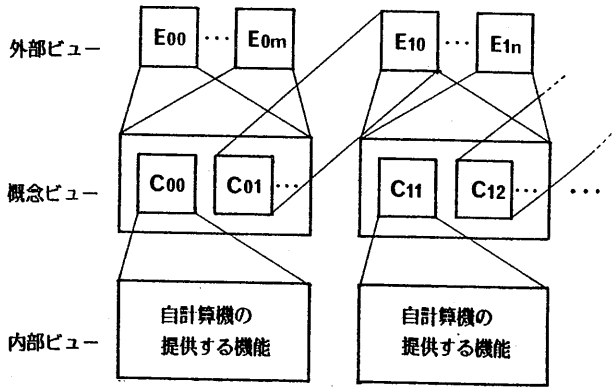


図３　サービスの定義部の構造

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、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。i）とii）のビューは、各計算機内で１つづつ用意するが、iii）は、各計算機内で、その計算機のＦＮ、或いはその計算機を直接使うユーザ毎に１つづつ用意する。

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。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が、一般にiii）とiv）は独立でよい。

本稿では、次章以下、iii）、iv）として、関数型言語を用いた場合について紹介する。

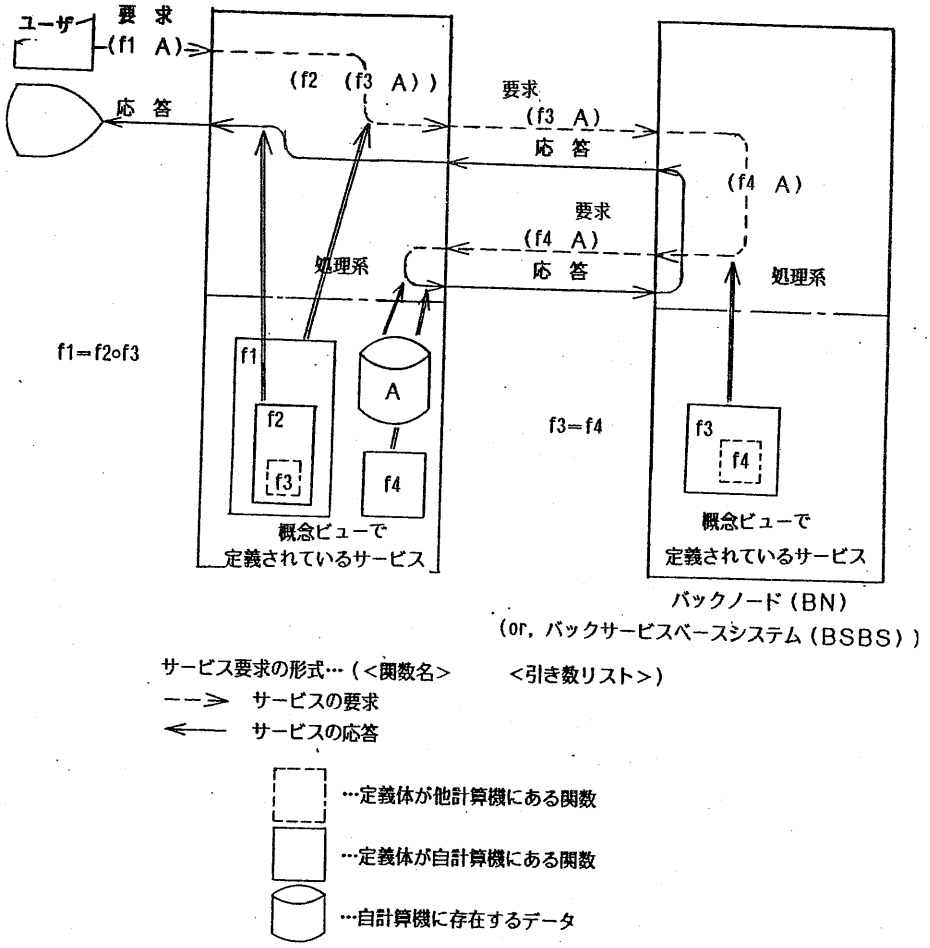


図４　計算機間の論理的な通信（要求と応答）の例

3．関数型言語に基く処理系間の通信

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。

4．処理系の構成とサービスの処理［3］

処理系記述言語として、Ｃ言語とＬisp を用いて、実験システムを作成した。

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、

①system
②send
③receive

の３関数をＬisp から関数として呼べる様にした。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・実行する機能を実現した。即ち、合成サービスの場合は、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
＜arg-list＞…引き数リスト・サービスの応答の形式（結果の場合）
（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、サービス要求をした計算機は、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。新たなサービス要求は、フレーム内の（eval）で処理されるが、処理中に更にＢＮにサービス要求を行なう必要が生じた場合は、サービス要求後、別のフレームを作成する。（eval）はこの様な動作が可能な様に新たに作成した。（Ｌisp の（eval）の拡張）

ＢＮからのメッセージがサービスの応答の場合は、（catch ）を抜けて、次のサービスの処理に移る。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。

5．副作用を伴なうサービスの取り扱い

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、

・ファイル転送
・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、

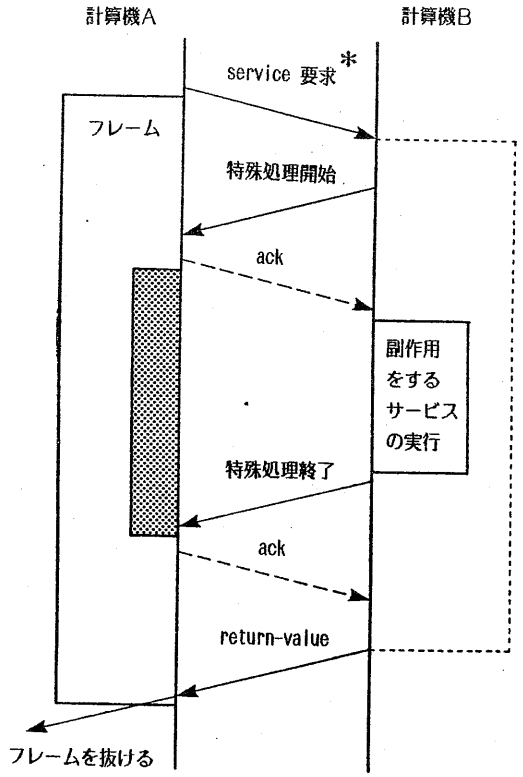
ｉ）特殊サービス開始
ii）特殊サービス終了

を示す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
② transparent

の２つの属性を用意した。①は、サービス提供者の通信路への出力を、サービス要求側で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５に示す。

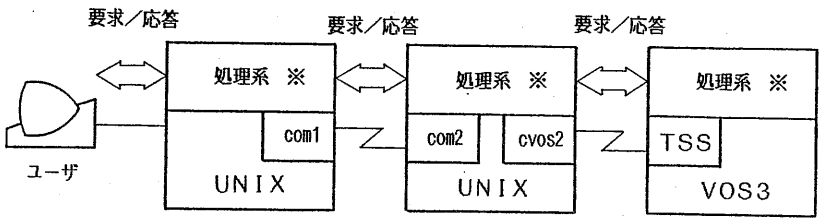
6．サービスの定義方法



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5 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



com1,com2,cvos2 …通信用ソフトウェア

VAX11／730　　VAX11／780　　M280H

図6 実装システム構成

※処理系記述言語は、Franz Lisp（UNIX）、UTILISP（VOS3）
OSとのインターフェースは、C言語

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、本章では、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＞：内部ビュー上での呼び出し名（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＞：ＢＮの外部ビュー上の名前（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

7．サービスの定義・実行例

図６の構成で、両計算機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。

図7 サービスの定義・要求例

文字列“ＬOAD”がファイル名に含まれている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
```

下線…ユーザの入力

サービスの定義

ＵＮＩＸ側 (defe lists 1) ← VAX11／780上からみたM 280Hのhostid

(defs grep)

ＶＯＳ３側 (defs lists (file))

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）

8．検討・評価

8.1 サービス定義・要求言語と関数型言語［4］

サービス要求／応答と、関数呼び出し／return-valueを自然に対応させることにより、処理系間の通信機構を容易に実現できた反面、

i）ファイル転送など、基本的なサービスを「副作用」という形で実現しなくてはならない。

ii）ハイレベルなユーザインタフェースの提供がまだ不十分

といった問題点がある。ii）に関しては、関数型言語の範囲内でも改善できるが、現在、サービス定義・要求言語として「論理型言語」を提供するシステムも検討している。論理型言語で、サービス定義を行なうことにより、サービスに関する情報を知識化することができ、よりハイレベルなユーザインタフェースの提供が可能となるであろう。また、ＳＢＳの概念自体は、言語に依存しないので、計算機－計算機間に応じて、適した言語を用いるといった構成も考えられる。

8.2 並列サービスの実現［2］

ＳＢＳでは、

i）効率の向上

ii）計算機同士の協調動作の制御

等の為に、並列サービスの起動が必要となる。

ii）の実現にあたっては、

①処理系内でのみ、（できる範囲で、或いは、必要な場合にかぎり）並列処理を行なう。

②サービス定義・要求言語として、陽に並列サービスを記述する制御構造を提供する。

の２のレベルが考えられる。現在は、ファイル転送などの、並列処理が必要な場合に限って、①を行なっている。

②の実現に関しては、

・並列サービスの実態（プロセス）の制御・管理

・プロセス間通信のサポート

に関して、サービス定義・要求言語による記述方法と、その実現方法について検討している。

8.3 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

の和である。これらの割合は、サービス毎に異なるが、実装システムの例では、①が最も多く、②は、極めて小さい。①は、ＳＢＳの下位通信機能の性能に依存する部分であり、ＳＢＳの処理系の効率自体は、十分よいといえる。①の負荷を減らす為には、処理系で、あらかじめ、サービスの処理を行なう順序をスケージューリングする必要がある。実験システムの処理系では、サービス要求言語を解釈するとすぐ実行しているが、この様な方式を「インタプリタ式」と呼び、効率化などの為に、スケジューリングや最適化を行なう方式を「コンパイラ式」と呼ぶ。コンパイラ式の処理系の作成も、今後の課題のうちの１つである。

8.4 定義部（ビュー）の管理［1］

現システムでは、ビューの管理は、ファイルに頼っているが、サービスの増大に対応したり、より系統的な管理を行なうために、ビューをデータベース化することを検討している。ＳＢＳでは、データとして、関係データベースをサポートするので、定義部をrelationで表現することにより、ビューをデータと同様に管理することができる。ビューをrelationで表現する時、原則として、サービス名をkey としたタプルの集合として表す。ビューを蓄えるrelationの具体的なスキーマに関しては、現在検討中である。

9．おわりに

本稿では、関数型言語に基くＳＢＳの構成方法とその具体例について述べた。残された問題は幾つかあるが、ＳＢＳによって、一般の網上の分散サービスを利用しやすい形で簡単に提供することがができる。この時、関数型言語でサービス要求／応答が容易に実現できることを示した。

今後、関数型言語によるサービス記述の蓄積と並行して、記述のハイレベル化、並行サービスの記述などに適した、サービス定義・要求言語の検討を行なっていく予定である。更に、種々のサービス要求・定義言語を網内で同時に使用できる様なＳＢＳのモデルとその構成方法についても検討する予定である。

＜　参　考　文　献　＞

［１］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、情処全大、１Ｊ－２、pp.781-782.、1983.10

［２］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」、情処全大、４Ｄ－８、pp.851-852.、1984.3

［３］荻野、深沢、田中、元岡、「サービスベースシステムの実装」、情処全大、４Ｄ－１０、pp.855-856.、1984.3

［４］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」、情処全大、1984.9

［５］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、1982.10