論　文

サービスベースシステムの概念とその構成

正員　荻野　正[†]　正員　深沢　友雄[††]　正員　田中　英彦[†]

Service Base System : Its Concepts and Design

Tadashi OGINO[†], Tomoo FUKAZAWA[††] *and* Hidehiko TANAKA[†], *Members*

あらまし　我々は，LANや広域網をも含めたネットワークのもとでの，計算機資源の有効利用のための枠組みとして，サービスベースシステム（SBS）を開発してきた．SBSは，既存のネットワークやOSを利用して，使いやすい網環境を提供するシステムである．本論文は，SBSの概念的枠組みとその基本構成，およびその可能性を示すことが目標である．本システムの主な特徴として，①計算機の提供する機能をサービスという抽象的なレベルでモデル化することにより，ユーザは，計算機の違いによる使い方の差異をあまり気にすることなく各計算機の機能を利用することができる．②分散したサービスの管理を3層ビューという構成で行い，各ノードでの，サービスの追加，組合せといった機能の拡張を容易に行うことができる，の二つを挙げることができる，本論文では，SBSの基本的概念と，各ノードの構成を示した．システムの有効性を示すために，M680HとVAXを接続した簡単な実験システムを構築した．実験システムにより，分散しているサービスをある程度自由に組み合わせて利用できることは確認できた．資源の記述方法等が今後の課題である．

1. まえがき

計算機ネットワークの分野の研究は，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，ファイル転送，リモートジョブエントリ等，計算機同士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)]．このような分野での発展により，計算機網を利用して複数の計算機を利用することは，ごく自然なこととなっている．しかし，それは物理的な側面だけであり，実際には，同じUNIXマシンを数台接続するだけでも，その上で互いの機能を利用しあうような複雑な処理を行うのは非常に困難である．そして，将来，さまざまな計算機が数十台，数百台と接続されたとき，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われていない．

† 東京大学工学部電気工学科，東京都
Faculty of Engineering, The University of Tokyo, Tokyo, 113 Japan
†† NTT LSI研究所，厚木市
NTT LSI Laboratory, Atsugi-shi, 243-01 Japan

我々は，複数台の計算機をLANや広域網をも含めたネットワークで接続した環境のもとで，

・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ことができる．

・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うことが容易にできる．

等を目標としてサービスベースシステム（SBS）を開発してきた[(2),(3)]．本論文は，このSBSの概念的枠組みとその基本構成，およびその可能性を示すことが目標であり，詳細な要素技術のリファインは今後の課題であると考える．

2. サービスベースシステム

サービスベースシステムが対象としているのは，LANや広域網を統合した計算機網環境でのシステムである．そして，この研究は，新しいNetwork OSを作るのではなく，既に存在しているOSや網環境の上で，それを統合する枠組みを提案することが目標である．従って，計算機のOSはもちろん，ファイル転送，リモートログイン等のプリミティブな通信機能は既に存在しているものと仮定する．

2.1 基本方針

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，大きく次の二つに分けることができる．

（1） 分散性：計算機資源が網中に分散して存在する．

（2） 異種性：各ノードの計算機に相違がある．

分散性を解決するために最低限必要な機能は，ある計算機資源の存在場所を知る機能である．この機能を実現する方法として，二つの極端な例を挙げると，

（1） すべての資源に関する情報を1個所に集める．

（2） すべてのノードは，自分のノードに存在する資源に関する情報のみをもつ．

がある．(1)の方法は網の規模が大きくなれば，問題となるのは明らかであり，(2)の方法はある計算機資源の存在場所を知るための負荷が重すぎる．結局，(1)の方法と(2)の方法の中間的な方法を，応用に応じて考えなければならない．

異種性に関しては，統一のための研究も行われているが，実際には，どの程度までをユーザに意識させずにシステムの内部で処理するか，異種性に関する情報をどうやってシステムに格納するか等が，研究の課題になる．

SBSは，

① 各計算機では，その計算機を介して利用することのできるサービスに関する必要な情報（サービスの存在場所，実行方法等）をあらかじめもっておく．そして，この情報は3層ビューという構成で管理する．

② 計算機-計算機間およびユーザ・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．

こと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）．既存の複数のサービスを組み合わせて一つの新しいサービスとすることもできる．既存のサービスを組み合わせたサービスを合成サービスと呼び，それに対し，それ以上分解できないサービスを基本サービスと呼ぶ．通常我々が使用している計算機環境では，作用はコマンドに，データはファイルに対応している．このモデルのもとでは，作用とデータを指定すること（これをサービスの要求と呼ぶ）で，計算機の特定の機能を利用することができる．計算機の提供するすべての機能を，このサービスというモデルで統一することができれば，ユーザと計算機間および計算機と計算機間でも，サービスの要求という統一されたインタフェースを介してさまざまな計算機の機能を利用することができることになる．計算機ごとによって異なるコマンド体系やシステムによって異なる環境設定といった計算機の使い方や概念の差異の大部分は，この共通のモデルと各計算機の機能とのマッピングに吸収することができるはずである．

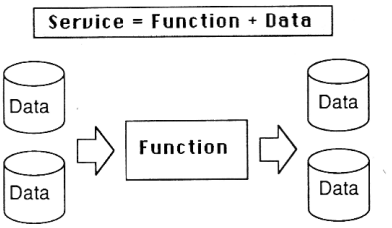


図1 サービスのモデル
Fig. 1 A Model of Service.

SBSでは，データモデルとしては柔軟性のあるRelationを基本として議論を進めていく．また，作用とは，0個以上のデータを入力して，1個以上のデータを出力するものとする．Remote Procedure Call等も作用に含めて考えることができる．また，ファイルの更新は，入力データと出力データが同じである特殊な場合と考えることができる．

2.3 3層ビュー

SBSでは，ユーザからは，計算機の提供する機能はサービス（作用＋データ，あるいはその組合せ）としか見えないが，それぞれのサービスの実行のために必要な情報等は計算機がもつことにする．現在の計算機環境ではユーザが知らなければならない情報を，SBSでは各サービスにサービスの記述としてもたせるわけである．

各計算機では，サービスに関する情報を次の三つのレベルに分けて記述，管理する（図2）．

① 外部ビュー
② 概念ビュー
③ 内部ビュー

内部ビューは，各計算機が独自で提供するサービスに関するビューであり，網内の各計算機ごとに別々に一つずつ存在する．

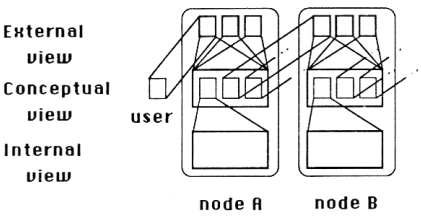


図2　3層ビュー
Fig. 2　Three layered view.

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．各計算機には，概念ビューが一つずつ存在する．

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る．各計算機は，その計算機を利用するユーザあるいは計算機対応に複数の外部ビューをもつ．

このように，3層ビューという構成にすることにより，各計算機は，自分の必要とするサービスに関する情報のみをもつことになり，一部のノードに情報が集中したり，また，不必要な情報までもつようなことはなくなる．

また，3層ビューという構成のため，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うことが容易となる．

以下でサービスの拡張を行う場合の必要な操作を考えてみる．

まず，あるノード(ノードA)が，別のノード(ノードB)に外部ビューを提供しているとき，ノードAをノードBのBN(Back Node)，逆にノードBをノードAのFN(Front Node)と呼ぶことにする(図3)．

例えば，自計算機で新たに新しい機能の追加を行いたければ，自計算機の内部ビューと概念ビューに登録すればよい．ユーザは，そのサービスを自分の外部ビューに追加するだけで，その新しいサービスを利用することができるようになる．

また，BNの概念ビューに既に登録してあるサービスを利用する場合には，BNの自計算機用の外部ビューと自計算機の概念ビューに記述を追加することにより，そのサービスを利用することができる．ユーザは，自分の外部ビューに追加するだけで，その新しいサービスを利用することができる．

2.4　サービスの要求/応答

サービスの要求とは，「作用」と「(対象となる)データの集合」を指定することである．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，統一したインタフェースで利用することができ，細かい使い方の差異は吸収することが可能である．

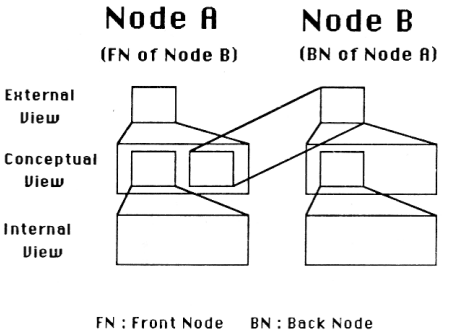


図3　FNとBN
Fig. 3　FN and BN.

各ノードは，分散性に関する情報を3層ビューにもっている．そして，ユーザあるいはFNからのサービス要求があったとき，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，BNに存在するサービスであれば，新たにサービス要求を行う．このように，3層ビューにより分散性を吸収しているので，ユーザはサービスの実際の存在場所を知らなくてもよく，同様に，あるノードがBNにサービスを要求するときにも，サービスの存在場所を知らなくてもよい(図4)．

もちろん，サービスの存在場所などの情報については，ユーザが知って，それを利用することもできる．

2.5　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概念的な構成は図5のようになっている．各計算機には，その計算機固有のOS，およびDBシステムがあらかじめ存在する．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自身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ための通信機構をCMSと呼ぶ．異な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では以上の部分はあらかじめそれぞれの計算機に存在することを前提にする．これらの上にSBSを構築するために，サービスの処理系(Service Processing System，SPS)とサービスの記述管理部(Service Dictionary/Directory Systom，SDDS)を設ける．

SPSは，サービスの要求/応答の処理を行う．SDDS

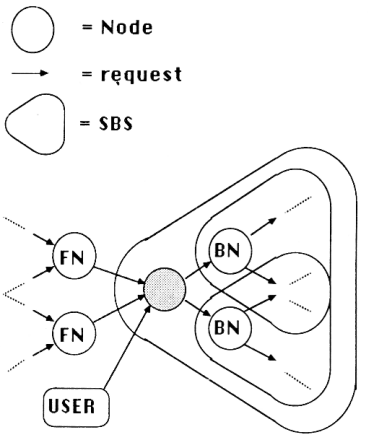
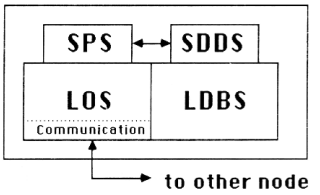


図4　SBSの網の論理構成
Fig. 4　Logical structure of SBS.



SPS : Service Processing System
SDDS : Service Dictionary
Directory System

図5　SPSとSDDS
Fig. 5　SPS and SDDS.

は，3層ビューとその管理システムからなる．これは，LDBSによって実現する．SDDSを実現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った[4]．SBSの最も基本的な概念についてはこのとき確認しているが，システム構成は実験用の非常に単純なものであった．本論文では，「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，SPSも，SDDS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．

SBSの各ノードの構成の詳細については，次の章で説明する．

3. サービスベースシステムのノード内構成

サービスベースシステムを構成する各ノードは，次の部分から構成される[5](図6)．

① 入出力機構(User Interface)

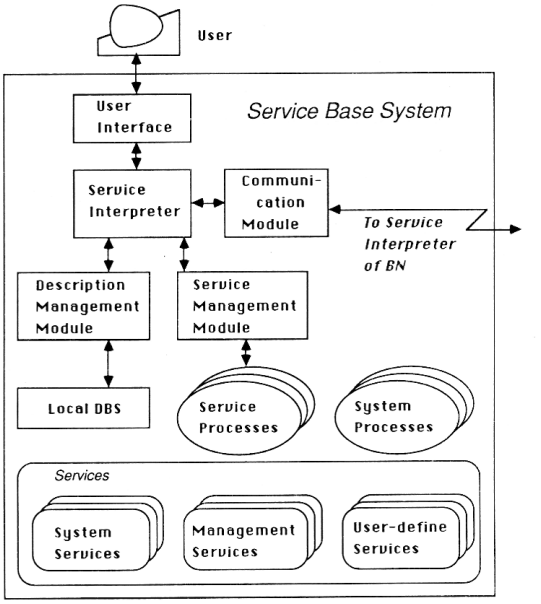


図6　SBSのノード構成
Fig. 6　A node configuration of SBS.

② サービスインタプリタ(Service Interpreter)
③ 記述管理モジュール(Description Management Module)
④ サービス管理モジュール(Service Management Module)
⑤ ローカルデータベースシステム(Local DBMS)
⑥ サービス実行プロセス(Service Processes)
⑦ サービス群(Services)
⑧ システム常駐プロセス(System Processes)

①，②，④がSPS，③がSDDSに相当する．

以下各部の詳細について述べる．

① 入出力機構　入出力機構はユーザとの入出力の制御を行う．グラフィック端末やマウス等を用いてユーザフレンドリなインタフェースを構成することもできるが，現段階では詳細な検討はしない．

② サービスインタプリタ　サービスインタプリタの機能は大きく次の三つに分けられる．

(a) サービスの分析
(b) サービスの実行
(c) サービスの要求

以上の三つの機能の実行の様子はサービス本体の記述言語の仕様による．

(a) サービスの分析　サービスの分析は，次のような順番の処理によって行われる．

1. ユーザ(またはFN)からの入力を受ける．

入力の形式：サービスの組合せ

2. 記述管理モジュールへの問合せをする．

問合せの種類：ビュー情報の問合せ

変換サービスの問合せ

3. 前処理をサービス管理モジュールへ依頼する．

前処理の内容：ファイル転送，変換サービス

変換サービスとは，言語やファイル名なども含めた意味での広い範囲のフォーマット変換をするサービスを指す．例えば，コンパイラは，高級言語からマシンコードへの変換サービスであると考える．

(b) サービスの実行　サービスの分析の結果，自ノードで実行できるサービスであれば，サービス管理モジュールへ依頼して実行する．

(c) サービスの要求　分析の結果，他ノードのサービスであれば，サービスの要求を行う．このときの処理は，まず，記述管理モジュールへ網情報を問い合わせて，必要ならばコネクション設定を行う．その後，その遠隔ノードのサービス管理モジュールへサービスの要求を行う．

サービスの実行/要求の後で必要ならば，後処理を行う．

サービス要求モジュールは，自分がコネクションを張っているBNを覚えておき，既にコネクションがはられているノードへの要求であればそれを利用してサービスの要求を行い，そうでなければ，コネクションの設定から開始する．

③ 記述管理モジュール　記述管理モジュールは，サービスインタプリタからの問合せに答える．問合せの種類には，

・サービスのビュー情報問合せ
・ノード情報問合せ

がある．実際のデータはLDBS内に存在する．

④ サービス管理モジュール　サービス管理モジュールは，サービスの実行時にサービスをプロセスとして起動し管理する．必要な機能としては，プロセスの起動，待合せ，プロセスの終了，プロセス情報問合せ等がある．これらの機能は，LOSの機能を利用する．

⑤ ローカルデータベースシステム　ローカルデータベースシステムは，そのノードに存在するDBSであり，ビュー情報等の実際のデータを格納する．

⑥ サービス実行プロセス　実行されているサービスはプロセスとして存在する．

⑦ サービス群　登録されているサービスは，

(a) 管理サービス群 (Management Services)
システム管理者用のサービス

(b) システムサービス群 (System Services)
システムがあらかじめ提供するユーザ用のサービス

(c) ユーザサービス群 (User-defined Services)
ユーザが登録したサービス

に分類できる．

(a) 管理サービス群　システム管理者用のサービスには次のようなものがある．

・ビュー管理サービス
　サービスの追加，変更，削除
　BNのビューを取り込むサービス
　FNのビューを作るサービス
・ユーザ管理サービス
　ユーザの登録・抹消
・ノード管理サービス
　ノードの登録・抹消
　ノード状態の問合せ，表示
・ビュー構造管理サービス
　ビューの構造を書き換えるサービス
・変換サービス管理サービス
　変換サービスの登録・削除

(b) システムサービス群　システムがあらかじめ提供するユーザ用のサービスには次のようなものがある．

・基本サービス記述言語ツール
　通常のプログラミング言語のコンパイラ，インタプリタ，エディタ，デバッガ等
・合成サービス記述言語ツール
　合成サービス記述言語のエディタ，デバッガ等
・サービス仕様記述言語
　仕様記述言語のエディタ
・ファイル転送サービス
　テキスト転送
　バイナリコード転送等
・変換サービス
　フォーマット変換
　コンパイラ等

(c) ユーザサービス群　システム構築後，システムサービス群を用いてユーザが作成，定義する．

⑧ システム常駐プロセス　ユーザ対応のプロセス以外に，常駐しているプロセスで，以下のようなものがある．

・ノード管理プロセス
　他ノードの状態の監視を行う．

・端末 logger
　ユーザが login したときにプロセスを立ち上げる．
・FN 監視プロセス
　FN からユーザの最初のサービス要求がきたときに，そのユーザ用のサービスインタプリタを起動する．上の端末 logger とほぼ同様の機能をもつ．
・account 用プロセス
・統計用プロセス

4. 実験システム

4.1 システム構成

サービスベースシステムの有効性を示すために簡単な実験システムを構築した．この実験システムは，サービスに関する記述としては，名前と存在場所，画面出力の有無に関する情報のみをもっている．この実験システムは，分散したノードに存在するコマンドや，その簡単な組合せをサービスとして定義することにより，サービスの要求と応答のみで，それらのサービスの実行が行えることを確認するための簡単なシステムである．実験システムは，東京大学大型計算機センターの M680H，VAX8600，および，当研究室の VAX-11/730 を接続して構成した（図 7）．OS は，M680H が VOS3，2 台の VAX が UNIX である．M680H-VAX8600 間の通信用ソフトウェアとして，CVOS を使用し，また VAX8600-VAX-11/730 間の通信用に pcom というソフトウェアを C 言語で開発した．処理系の記述言語としては C-prolog および C 言語を使用した．

4.2 サービスの記述[7]

このシステムでは，簡単のため，外部ビューと内部ビューに関する記述は省略する．シングルユーザの環境のみで実験を行うので，外部ビューは概念ビューと同じと考えることで問題はない．内部ビューは OS の提供するコマンド名，ファイル名のみとして OS の機能でこれを用用する．

概念ビューのサービスに関する記述は，後で述べる述語によって登録するが，内部的には，あるサービスに関するある属性の値が，

　service（サービス名，属性名，属性値）

という prolog の fact の形で記憶されている．属性としては，次のものが実装されている．

name	サービス名
map	他のビューでのサービス名
place	サービスの存在場所
out	画面への出力の有無

サービス名という属性は，サービスの存在を確認するときに使われている．

また，サービスの登録をするための述語として次のものを作成した．

（1） 自計算機のサービスの定義
　　assert_int（サービス名，内部ビューでのサービス名，属性名（属性値）のリスト）
（2） 他計算機のサービスの定義
　　assert_ext（サービス名，外部ビューでのサービス名，サービスの存在場所，属性名（属性値）のリスト）

4.3 サービスの定義・実行例

以下にサービスの定義の例を示す．なお，ここでは簡単のために，VAX8600 と M680H の 2 台の構成にしてある．

　VAX8600 での定義

```
assert-int(grep(X, Y), grep(X, Y),
[out(stdout)].
assert-ext(lists(X), [ ]).
listsf(X, F):-mfile(lists(X), F).
```

　M680H での定義

```
assert-int(lists(X), [out(stdout)]).
```

grep は，UNIX 上のコマンドで，文字列の検索を行う．lists は，VOS3 上のコマンドで，所有しているファイルの情報を画面に出力する．listsf は，M680H で実行した lists の出力をファイルに格納する．

定義したサービスを 3 層ビュー上で示すと図 8 のようになる．定義したコマンドを組み合わせたサービスの実行例を図 9 に示す．

4.4 実験結果

VAX 上にいくつかのコマンド（grep，ls，sort etc）を定義し，M 上にいくつかコマンド（eng，edit etc）を定義し，それらを組み合わせて実行するシステムを作成した．

実験システムは，主にサービスの要求/応答の処理を扱う部分は C-prolog で，通信のための処理やプロセス

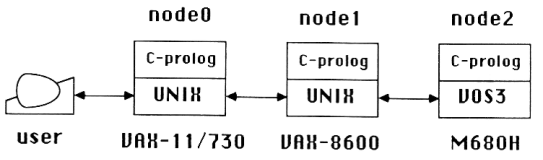


図 7　実験システムの構成
Fig. 7　A configuration of pilot system.

管理の部分はC言語で記述した．プログラムサイズは，ソースレベルで，C-prologの部分は約1,300行，C言語の部分は約2,400行である．

また，異なるノードで同時に処理を実行する，ノード間の並列処理や，相手のノードで処理を行いながら，その処理結果を自分のノードで入力として利用して同時に処理を進めていく，ノード間のパイプラインについても比較的容易に実装することができた．但し，これらの組合せは，システムに精通している人間が組み合わせたものであり，どんな場合でもうまくいくレベルには達していない．そのためには，サービスに関するより多くの情報を記述しておかなければならない．

現在，OSのコマンド，ユーザプログラム等合わせて約40のサービスについて記述してあるが，これらのサービスに関しては自由に組み合わせて利用することができている[8]．

この実験システムは，処理の効率等を考慮してはいないが，参考のためにM680H上の文献検索用のコマンドengを実行したときのVAX 8600上の処理系のCPU時間を表1に示す．この表1には，M上でのコマンド(eng)の処理時間は含まれていない．比較のために，リモートログインをして同じ処理をしたときの通信用プログラムのCPU時間も示す．表では，SBS構成にしたことにより，オーバヘッドは約3.5倍に増えているのがわかる．しかし，実際には，M上でのengの処理時間に対する通信プログラムのオーバヘッドは十分小さいものであり，SBSの構成にしたことによるオーバヘッドはユーザにはほとんど感じられない程度である．また，処理系のプログラムを改良することにより，VAX 8600での処理時間を現在の約40％に減少させることができるというデータも得られているが，オーバヘッドを減らすことが本来の目的ではないので詳細については省略する．いずれにせよ，SBSの構成にしたことによる全体の処理時間に対するオーバヘッドの増加は十分小さなものである．

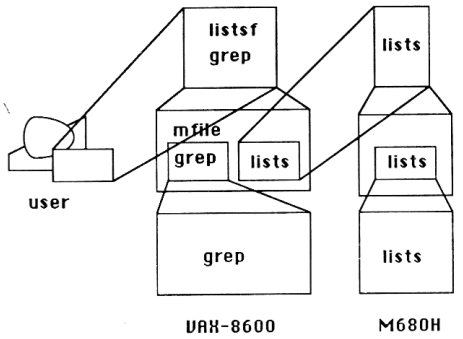


図8　3層ビューの例
Fig. 8　An example of three layered view.

4.5 評　　価

この実験システムによってSBSという構成で分散しているサービスをある程度自由に組み合わせて利用できることは確認でき，その主要目的は達成できた．しかし，この実験システムでは，次に示すようないくつかの不十分な点がある．

・サービスの記述として，その名前と存在場所に関するものしか実装されていない．

・サービス記述が属性とその値という1対1の構造になっており，複雑な情報を記述することが難しい．

サービスベースシステムは，分散して存在しているさまざまなサービスを容易に組み合わせて実行できる環境を一つの目標としている．そのためには，サービスの名前と存在場所だけでなく，その仕様をきちんと記述できなければならない．仕様の記述法については，現在検討中である[9]．

表1　各プロセスのCPU消費時間
（コマンドengを実行した場合）

	C言語で書かれたシステムプロセス	C-prologで書かれたシステムプロセス	通信用CVOSプロセス
SBS構成の場合	2920ms	360	2190
リモートログインの場合	—	—	1560

```
| ?- listsf('%',temp),grep('VDATA',temp).

PO        114         99     ARCHIV  A80595.LISPLIB.COMP.VDATA
PO         95         38     ARCHIV  A80595.LISPLIB.VDATA
PO         95         28     LD0005  A80595.PROLOG.VDATA
PO         95         71     LD0024  A80595.PROLOG.VDATA.OLD
PO        171        113     ARCHIV  A80595.SB.VDATA
PS         19         18     ARCHIV  A80595.SBLIB.VDATA
PO         19         10     ARCHIV  A80595.TEST.VDATA
PO         76          3     LD0023  A80595.UTIL.NEW.VDATA
```

図9　サービスの実行例
Fig. 9　Execution example(lists).

5. むすび

本論文では，計算機網内の計算機資源の有効利用を目指したサービスベースシステムの基本的な概念とその構成方法について述べた．

本システムの主な特徴は，

① 計算機の提供する機能をサービスという抽象的なレベルでモデル化する．従来の計算機環境では必要であった環境の設定とか，フォーマットの変換といった実際の仕事とは離れている枝葉末節の部分をできるだけ計算機にやらせる．その結果として，ユーザは，計算機の違いによる使い方の差異をあまり気にすることなく，網中に分散して存在するサービスを自由に組み合わせて利用することができる．

② 分散したサービスの管理を3層ビューという構成で行う．各ノードには，そのノードを使うユーザに必要なサービスに関する情報のみが存在することになる．各ノードでは，サービスの追加，組合せといった機能の拡張は他のノードとは独立に行うことができる．の二つを挙げることができる．本論文では，簡単な実験システムを構築して，このサービスベースシステムの有効性の一部を示した．

今後検討すべき点としては，①については，現在の計算機の機能とサービスというモデルを結ぶためにどのような情報をどのような方法で記述するかという点，②については，サービスの変更，削除といった，他ノードと関連あるようなサービスの管理の機構などがある．

更に，多くのサービスをこの上に実装してみることによって，この方法の汎用性をより具体的に示すことも残された課題である．

サービスベースシステムの実用のためには，各種ツールの実装も含め，サービスの再配置，最適化に関する検討も必要である．

謝辞 本研究を進めるにあたり，常日ごろ貴重な議論をして頂いた田中(英)研のみなさん，特にSIGDSのメンバーに感謝いたします．

文　献

(1) A. S. Tanenbaum and R. van Renesse : "Distributed operating systems", ACM, Computing Survey (Dec. 1985).

(2) T. Ogino and H. Tanaka : "Service base system : A framework for distributed utilities", Joint Workshop on Computer Communication (June 1987).

(3) 荻野　正，田中英彦："論理型言語を用いたサービスベースシステムの実装"，情処学マルチメディア通信と分散処理研究会(昭60-07)．

(4) 荻野　正，深沢友雄，田中英彦，元岡　達："関数型言語に基づくサービスベースシステムの構成"，信学技報，IN84-37 (1984-08)．

(5) 荻野　正，橘高大造，何　千山，田中英彦："サービスベースシステムのノード構成"，情処学第35全大，3U-6 (昭60-09)．

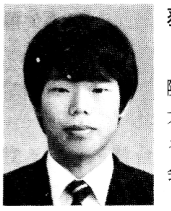
(6) 荻野　正，田中英彦，元岡　達："論理型言語を用いたサービスベースシステムの実装"，情処学第31全大，7Q-7 (1985-09)．

(7) 荻野　正，深沢友雄，田中英彦，元岡　達："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"，情処学第29全大，6H-7 (1984-09)．

(8) 荻野　正，田中英彦："サービスベースシステムにおけるサービスの実装"，情処学第32全大，3D-8 (1986-03)．

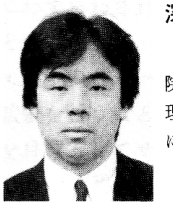
(9) 荻野　正，田中英彦："サービスベースシステムにおけるサービス記述手法"，情処学第34全大，1Y-1 (1987-03)．

(昭和63年1月19日受付，4月13日再受付)



荻野　正

昭58東大・工・電子卒．昭60同大大学院電気工学専門課程修士課程了．現在，同大学院博士課程在学中．分散処理，計算機ネットワークの研究に従事．情報処理学会会員．



深沢　友雄

昭55東大・工・電気卒．昭60同大大学院博士課程了．現在NTTに所属，分散処理，並列処理，DAシステムに関する研究に従事．情報処理学会会員．工博．

田中　英彦

昭40東大・工・電子卒．昭45同大大学院博士課程了．工博．同年東京大学工学部講師，昭46助教授．昭53～54ニューヨーク市立大学客員教授，昭62東京大学工学部教授，現在に至る．計算機アーキテクチャ，並列推論マシン，知識ベースマシン，オブジェクト指向計算システム，分散処理などの研究に従事．著書「計算機アーキテクチャ」，「VLSIコンピュータI，II」(いずれも共著)，「情報処理システム」．IEEE，ACM，人工知能学会各会員．