4D-11

サービスベースシステムにおける会話型サービスの実現

大政浩一　深沢友雄　田中英彦　元岡達

(東京大学　工学部)

1. はじめに

サービスベースシステム(SBS)とは、計算機の提供する機能をすべてサービス(ロードモジュール、データ)の要求と応答としてとらえ、ネットワーク上での分散性をユーザに意識させないようにするシステムである。

また、「サービス=データ+作用」と考えることにし、現システムでは各作用を関数的なものとして取扱っている。ところがこの場合、エディタのような会話型サービスを直接行うことが困難であった。本稿では、このような会話型サービスの取扱い及び、処理系への実装について述べる。

2. SBSで取扱うデータ

前述のようにSBSは関数型モデルのため、データに関数を作用させることでサービスを実行しているが、取扱うデータは以下のように分類できる。

① 処理系が直接解釈するデータ

関数の引数として値を与えることにより、サービスを実行するもの。

② 二次記憶上のデータ

関数の引数としてデータの格納場所の名前を与えて、その副作用の形でデータの実体を転送するもの。そのために、下位レベルの通信機能としてファイル転送機能を用いている。データのタイプとしては、ファイルとリレーションの2種類を考えている。

③ 外部からinteractive に与えられるデータ

外部(ユーザ)と会話的なやりとりを行うようなサービスで扱われる場合のデータのことで、これも関数の副作用として実行される。例としては、エディタを使っている時に入出力するデータが考えられる。

今までのシステムでは①、②の処理については実装済みであるが、③については未解決であった。

3. 会話型サービスの実現方法

3.1 自計算機をtransparent にする

ユーザが他計算機の会話型サービスを利用する場合は、自計算機の端末が直接他計算機につながれているように見せる必要がある。したがって、自計算機をトランスペアレントにし、処理系を通さずに直接他計算機とつなげてやればよい。そこで、自計算機の処理系を通してシステムを利用するモード(SBSモード)の他に、自計算機をトランスペアレントにするモード(透過モード)を考え、モードを切り換える関数を作成した。

以上により会話型サービスの実現には、自計算機を透過モードにする関数と、要求しているサービスを他計算機で起動させる関数を並行して実行すればよいことになる。ただし、現システムでは関数は1つずつ逐次的にしか動作しないので、その点を工夫してやる必要がある。

3.2 会話型サービスの属性を用いた定義

多種の会話型サービスについて別々に以上のようなことをする関数を作成していたのでは一般的でないし、また新たなサービスが生じた時のためにも、サービスが会話型であることを属性を用いて定義するだけでよいようにしたい。これによって、処理系がサービスの要求を受けた時に、サービスの属性をチェックすることにより会話型であることを認知し、自動的に環境を設定するようにできる。

4. システムへの実装

4.1 システム構成

システムは図1のようになっており、処理系はL

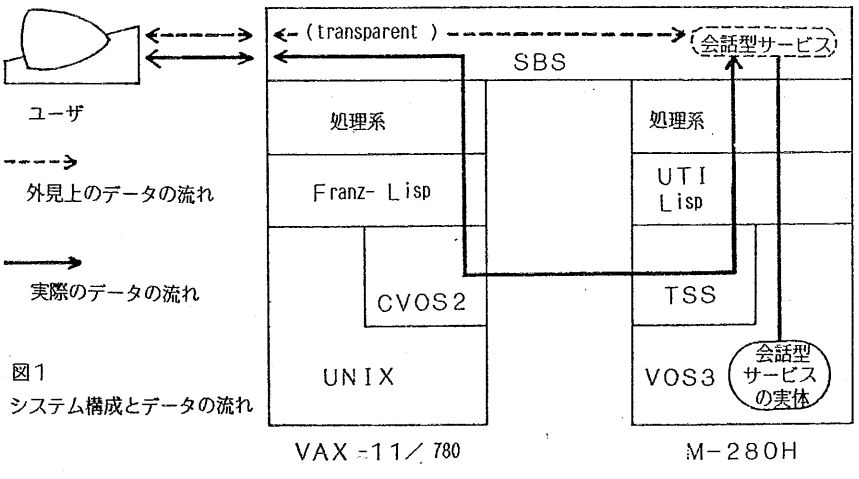

図1
システム構成とデータの流れ

ＩＳＰで書かれている。ここでは、ＶＡＸ－11／７80からＨＩＴＡＣ　Ｍ－280 ＨのＯＳの提供するエディタなどを動かすことを目標とした。［１］

4.2　会話型サービスを起動するシーケンス

3.1で述べたことを実装すると図２のようなシーケンスとなる。ここで少々複雑になったのは、ＳＢＳの処理モデルとして、逐次的に処理が行われる関数型モデルを用いたことによる。即ち、サービス要求をした場合、サービスが終了してその応答が返るまで、次のサービスが実行できない。このために、自計算機でトランスペアレントなサービスが起動されても他計算機に応答が返らないので、他計算機側では同時にサービスを起動させてやることができない。そこで、疑似的な応答を返すことによって他計算機側でのサービスを実行可能にすることにした。

4.3　Plistによるサービスの属性の記述

ＳＢＳでは、概念ビュー上のサービス記述を記憶する内部構造として、各サービス毎にＬＩＳＰのＰlistを用いている。今までのシステムでは、Plistの属性によって、サービスの実体の存在する場所や名前を知っていたのであるが、これを会話型サービスの場合にも利用してみた。つまり、処理系がevaluateする時に、サービスが会話型であるか否かを属性で調べ、もし会話型なら図２のようなシーケンスを実行するようにした。

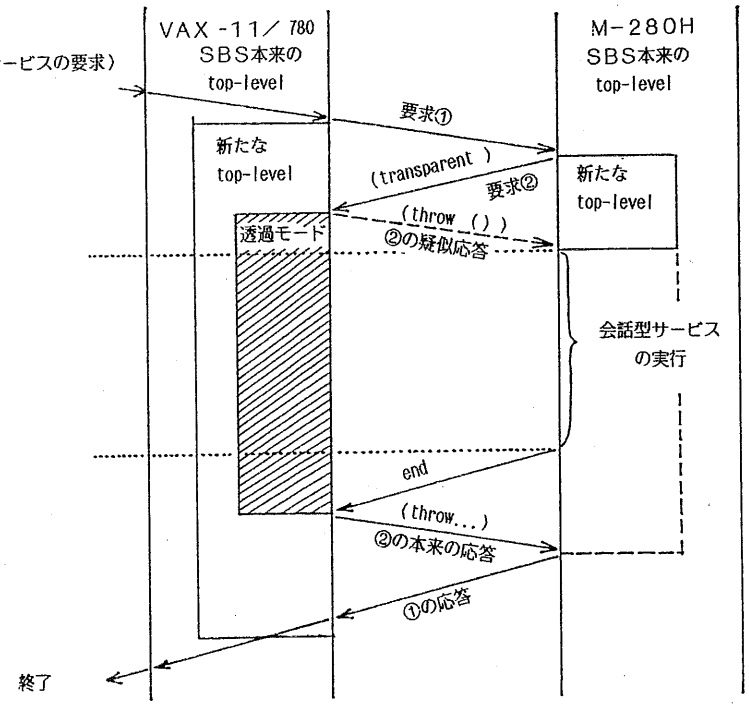

図2　会話型サービスのシーケンス

5．実際の使用例

サービスの定義は次のように行う。

```
(define-service
    ＜ＶＡＸ上でのservice 名＞
    ＜Ｍ上でのservice 名＞　＜属性＞)
```

ここで、＜属性＞のところに会話型であることを記述する。これを用いて会話型サービスとして定義すれば、サービスの実体の場所を意識しなくても以下のように使えるようになる。これは、Ｍ上のＣのファイルをエディットして、ＵＮＩＸのＣコンパイラにかけ実行させる例である。

```
-> (define-service 'editm 'edit 'interactive)
editm
-> (progn (ccm (editm 'test.c)) (a.out))
JET120012A ENTER NEW OR OLD -
new
INPUT
00010 main()
00020 {
00030       printf("service base system\n");
00040 }
00050
E>end s
JTD5355I  SAVED TO NEW DATA SET ('Z0448.TEST.C')
service base system
nil
->
```

6．検討

会話型サービスを起動するシーケンスが図２のように複雑になったのは、関数呼び出しの場合はすべてが終わるまで答が返って来ず、次へ進めないことが問題であった。そこでこれを解決するには、トランスペアレントにする関数と、要求しているサービスを起動する関数が並行して動作すれば良いわけである。そのような並行サービスを用いた会話型サービスを実現することが今後の課題である。（並行サービスについては文献［２］を参照）

＜参考文献＞

［１］　荻野、他、「サービスベースシステムの実装」、本大会予稿集、４Ｄ－１０

［２］　深沢、他、「サービスベースシステムにおける並行処理」、本大会予稿集、４Ｄ－８