大規模ネットワークにおける透過的なサービス管理システム

栗山幸三[1]・山本富士男[2]・荒木智行[2]

[1] 大学院工学研究科情報工学専攻
[2] 情報工学科

Transparent Service Management System in Large Area Network Environment

Kohzoh KURIYAMA[1)], Fujio YAMAMOTO[2)], Tomoyuki ARAKI[2)]

Abstract

A method for constructing transparent service control system applicable to wide area network was studied. We paid attention to maintain flexible reaction against changes in network environment, as well as managing security. In this study, we adopted the Distributed Objects supplied by Apple Inc. as our distributed object environment. Our results proved that it is fairly possible to construct such a service control system in this environment.

Key Words: Distributed Objects, Distributed System

1. まえがき

本研究では，大規模ネットワークにおいて透過的にサービスを提供できるサービス管理システムの検討を行い，それをApple社の分散オブジェクト環境であるDistributed Objects[1)]環境に実装することを試みる．

近年，ネットワーク環境の肥大化により，ネットワーク上のサービスの所在を確認するのが困難になりつつある．そこで，サービスの所在を管理するサービス管理システムが必要となる．既存のサービス管理システムとして，X.500[2)]準拠の様々なシステムが存在しているが，本研究では，Distributed Objectsでの実装を試みる．その理由として，Distributed Objectsは，Apple社の次期MacOSのRhapsodyで導入が予定され，今後の普及が大いに期待されることと，その環境にサービスを適切に管理するシステムが，公式にはまだ発表されていないことがあげられる．また，Distributed Objectsを用いることにより，透過的な分散システムの実装を，通常のシングルプロセスのシステムにかかる工程とほぼ同じ程度で実現することができる．このことから，今後，分散システムが目覚ましく普及すると考えられ，サービス管理システムの配置は，システムの構成上，重要となる．

そこで，本研究では，Distributed Objects環境において，透過的なサービス管理，環境の変化への柔軟な対応，セキュリティの配慮を実現するサービス管理システムの検討・開発を目的とする．

以降，本論文では，第2節で実装に用いる，Distributed ObjectsシステムをふくむOPENSTEPフレームワーク[1)]の概要を記す．次の第3節では，本研究で構築するサービス管理システムの全体像を記し，第4節で，システムの中核となるサービス管理サーバの詳細について記す．そして，第5節では，サービス管理システムの実装について記す．

2. OPENSTEPフレームワークの概要

2.1. OPENSTEPフレームワークの動作環境

OPENSTEPフレームワークは，Apple社の次期MacOSであるRhapsodyに導入されるフレームワークである．Rhapsodyにおいて，OPENSTEPフレームワークはYellow Boxと呼ばれるレイヤに位置付けられる．このレイヤは，OSレイヤの上に置かれ，ウィンドウサーバレイヤの下に置かれる．これを図に表すと図1のようになる．このように，Rhapsodyでは，システムを階層化しているため，OPENSTEPフレームワークの導入に特定のOSおよびウィンドウサーバを必要としない．実際に，Yellow Boxは，Microsoft社のWindosNT,Windows95,Apple社の旧MacOS,Carnegie Mellon大学のMachOS上に置かれ，それぞれの形式で画面表示される．また，RhapsodyではYellow BoxとJavaを相互運用するアプリケーションの開発もできる．

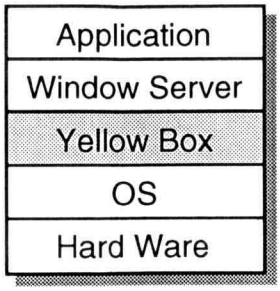

図1 Rhapsodyの階層

その他のOPENSTEPフレームワークを搭載する製品としてSun Microsystems社のOPENSTEP Solarisがある．これは，OSとしてSolarisを置いている．また，GNU ProductsよりGNU Stepが提供されている．これは，OSとしてLinux,Windows95,WindowsNTを置いている．

このように，OPENSTEPフレームワークは，現在，広く用いられている，さまざまなOS上で動作し，それぞれの形式で画面表示される．OPENSTEPフレームワークによるアプリケーションは，利用するOSが異なっても，ソースレベルでは完全な互換性がある．それは，システムコールや，GUI環境のプログラムについてもである．また，OPENSTEPフレームワークに含まれるDistributed Objects環境を用いることにより異なるOSのアプリケーション間で透過的な通信が実現される．このように，OPENSTEPフレームワークを利用することにより，マルチプラットホームに対応したアプリケーションの開発を簡単にすることができる．

2.2. OPENSTEPフレームワークのオブジェクト

異なるOS間でアプリケーションを相互運用するには，まず，データの互換性が問題となる．特に文字列については，多くの問題が存在する．そこで，OPENSTEPフレームワークでは，データをすべてオブジェクトとしてカプセル化する．それは，文字列オブジェクト，整数オブジェクト，日付オブジェクトなどである．それらのオブジェクトには，型変換や，コード変換などのメソッドが実装されている．これにより，異なるプラットホーム間でのデータ交換が簡単に行える．

また，それらのオブジェクトは，OPENSTEPフレームワークで提供されるクラスをアロケートし，そしてイニシャライズすることにより生成される，アロケートやイニシャライズなどのオブジェクトを生成するための基本的なメソッドは，すべてのOPENSTEPクラスが継承するNSObjectクラスで実装される．よって，OPENSTEPフレームワークで独自のクラスを定義するには，NSObjectまたは，それを継承するクラスをサブクラス化する．

このように，OPENSTEPフレームワークでは，オブジェクトをプログラム実行時に大量に生成するため，その解放の問題が浮上する．それは，複数のオブジェクトが1つオブジェクトを共有しているにもかかわらず，その中の1つが共有しているオブジェクトを解放してしまうといった問題である．OPENSTEPフレームワークでは，この問題をAutorelease poolを導入することにより解決している．その機能は，まずオブジェクトが生成されるとAutorelease poolのカウンタが1足される．そして，そのオブジェクトが他のオブジェクトに引き渡され，retainメッセージが送られた場合もカウンタが1足される．逆に，そのオブジェクトを利用しているオブジェクトがそのオブジェクトにreleaseメッセージを送ると，カウンタは，1引かれる．Autorelease poolはカウンタの値をイベントループの後に観察し，もし0であれば，オブジェクトを解放する．これらの一連の動作は分散オブジェクト環境に置いても適応される．

次に，複数のデータをグループ化したい場合，通常，配列や構造体が用いられる．これらについても，OPENSTEPフレームワークではオブジェクトとして実装する．構造体に関しては，辞書オブジェクトが代替となる．これらは，1つのオブジェクトが複数のオブジェクトの位置情報を保持することで実装される．これらをオブジェクトで実装する利点は，要素となるオブジェクトを管理するオブジェクトが，多くの，要素を操作するメソッドを実装しているため，それらを用いるプログラムが簡単になることであろう．

OPENSTEPフレームワークでは，以上のようなデータに関するクラスの他に，GUI・イベント処理に関するクラス，Task・Threadの処理のようなシステムコールに関するクラス，そして，本研究で用いるDistributed Objects環境に関するクラスなどが用意されている．それらを組み合わせることにより，簡単にマルチプラットホームに対応した，分散アプリケーションを実現することができる．

2.3. Distributed Objects環境

Distributed Objectsは，OPENSTEPフレームワークにおいて，分散アプリケーションを製作するのに用いられる分散オブジェクト環境である．Distributed Objectsを利用することにより，透過的な通信を簡単に実現することができる．このような通信をおこなうために，Distributed Objectsでは，システム内にProxy Objectと呼ばれる通信を専門に行う特別なオブジェクトを配置する．メッセージのセンダーは，このオブジェクトを中継して通信を行うことになる．つまり，センダーは，Proxy Objectに対し通常のメッセージを送るだけでリモートメッセージが実現される．これを図に表すと図2のようになる．

また，透過的な通信を実現するために，Distributed Objectsでは，ASCII文字列を用いクライアント・サーバ間の接続を行う．まず，サーバ側では，クライアントに提供するオブジェクトにASCII文字列による名前を付けネットワーク上に公開する．クライアントでは，その文字列を用いサーバとの接続要求を行う．これにより，クライアントはサーバの厳密な位置情報を知らなくとも両者の接続が実現し，透過性が高められる．また，多対一通信に関しても特別な設定を必要としない．

クライアントとサーバの接続が成功したら，クライアントからサーバへのメッセージセンディングが行われる．この際に，クライアントが送ったメッセージに対して，サーバがインターフェースを持っているか，という

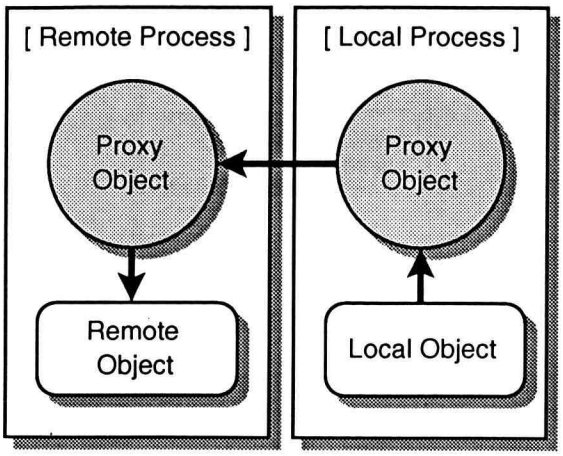

図2　Proxy Objectの代理

問題が出てくる．Distributed Objectsではこの問題に対し，まず，クライアントは，送るメッセージに対しサーバ側でインターフェースを持っているかを確認するメッセージをサーバに送り，インターフェースを持つという応答が返ると，実際にメッセージを送る．しかし，このような手順では，2度クライアント・サーバ間の通信をしなくてはならなくなり効率が悪い．そこで，サーバのインターフェースをあらかじめクライアントに登録しておくことができる．Distributed Objectsでは，これをプロトコルと呼ぶ．これを用いることにより，クライアント・サーバ間のメッセージセンディングが，一回の通信で実現される．

Distributed Objectsには，この他にメッセージのタイムアウト設定や，メッセージの認証，非同期メッセージのサポートなど，様々な機能を持っている．また，Distributed ObjectsとOPENSTEPフレームワークの通知オブジェクトを組み合わせることもできる．このオブジェクトは，自分に登録されているすべてのオブジェクトに同一の通知を行うのに用いられる．これにProxy Objectsを登録することにより，分散環境内での通知が可能となる．このように，Distributed Objectsには，多くのオプションが存在し，これらを組み合わせることにより，簡単であるが強力な通信機能を実現することができる．

3. サービス管理システムの概要

3.1. サービスの定義

この節では，最初にサービスの定義を与え，本研究で検討・開発するサービス管理システムの概要について記す．サービスという用語はそれを扱う分野によって様々なとらえかたができるが，ここでは，ユーザからデータを受け取り，そのデータを処理し，その結果をユーザに返すといった，汎用的なものをサービスであると定義する．その図を図3に記す．このサービスの適応例としては，英和辞書や，通貨変換，文字コード変換などが考えられる．

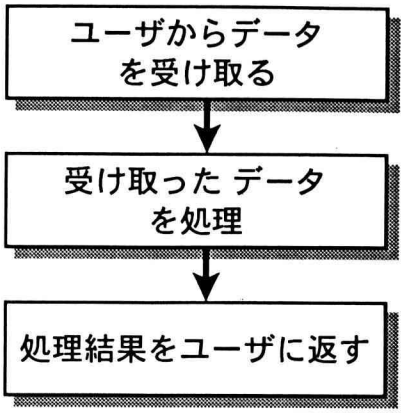

図3　サービスの定義

3.2. サービス管理システムの概要

本研究で検討・開発するサービス管理システムは，実際のサービス，クライアント，サービス管理サーバの三つの部分に分けられる．その相関図を図4に記す．

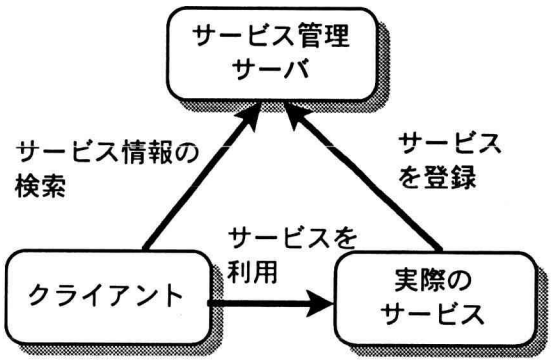

図4　サービス管理システムの相関図

実際のサービスは，ユーザが実際に利用するネットワーク上に置かれたサービスである．実際のサービスは，ネットワーク上の任意の場所に置かれるため，その位置情報を保有する．また，クライアントがサービスにアクセスするためのインターフェース情報を保有する．それらの情報はサービス管理サーバに登録され，そして，その情報が変更される度に，それが登録されているサービス管理サーバに通知される．

クライアントは，ユーザがサービスを利用するのに直接操作する部分である．クライアントはサービスと接続するために，サービス管理サーバからサービスの位置情報とインターフェース情報を引きだす．そして，それらを元に実際のサービスに接続する．

サービス管理サーバは，サービスの位置情報とインターフェース情報を管理する機能を持つ．サービス管理サーバには，他にも，大規模ネットワークに対応するような機能や，ユーザのサービスに対するアクセス権を設定する機能を持たせる．それらについては次節で述べる．

4. サービス管理サーバの詳細

4.1. サービス管理サーバの機能

サービス管理サーバの概要を第3節で述べたが，ここでは，大規模ネットワークに対応させる機能，ユーザのサービスに対するアクセス権を設定する機能についての詳細を述べる．

まず，大規模ネットワークへの対応について，サービス管理サーバを単体で構成するのではなく，複数のサービス管理サーバを連動させる．具体的には，複数のサービス管理サーバを階層化して結合する．この階層化について，下位層に位置付けられているサーバは，上位層に位置付けられているサーバに登録されているサービスを利用できるようにする．このために，各サーバには，0または1個の上位層サーバの位置情報を保持する．なお，上位層から下位層のサービスは参照できないようにする．

次に，サービスのアクセス範囲に関して，各サービス管理サーバに，接続許可をするユーザ情報を保持させる．そして，クライアントは初めに自分のユーザ情報が登録されているサービス管理サーバと接続される．これと，前述のサーバの階層の機能を組み合わせることにより，各ユーザに対してサービスのアクセス範囲を設定することができる．

また，特定のユーザグループでのみ利用可能なサービスを定義するために，サービス管理サーバで管理するサービスを2種類に分類する．その1つをローカルサービスと名づけ，そのサービス管理サーバに登録されているユーザのみ利用可能なサービスとする．もう1つをグローバルサービスと名づけ下位層に登録されているユーザからも参照できるサービスとする．

4.2. サービス管理サーバの具体例

前述したサービス管理サーバは，図5のようになる．

クライアントを利用しているユーザをUser5と想定すると，クライアントは，初めにUser5を管理しているサービス管理サーバ4に接続される．すると，User5は，そのサーバで管理されている，Global Service3，Local Service4とサービス管理サーバ1のGlobal Service1を利用可能となる．なお，サービス管理サーバ2で管理されているサービスとサービス管理サーバ3で管理されているLocal Service3,そして，サービス管理サーバ1で管理されているLocal Service1については利用不可能となる．

また，User4に関しては，サービス管理サーバ3のLocal Service3とサービス管理サーバ1のGlobal Service1を利用可能である．

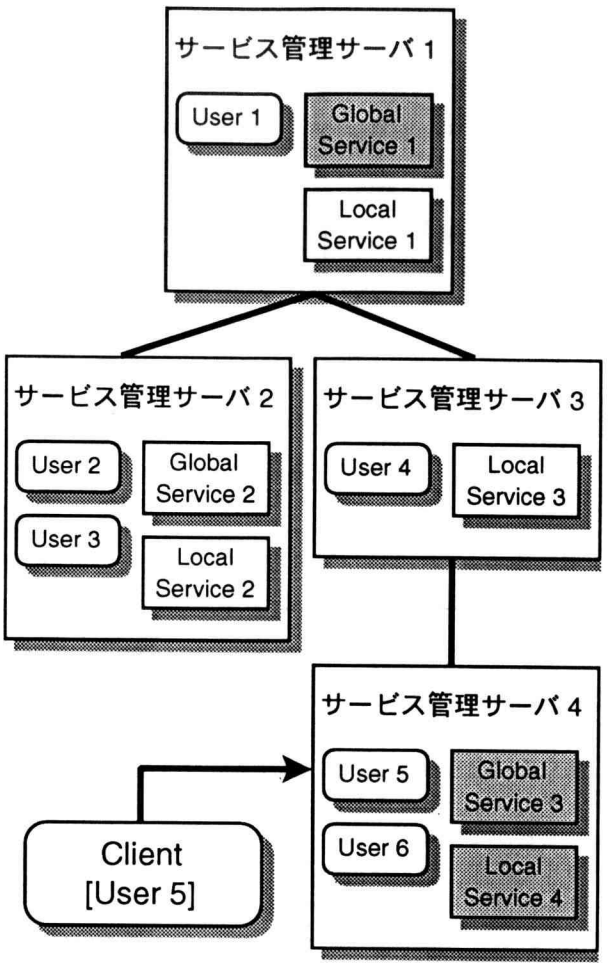

図5 サービス管理サーバの具体例

5. サービス管理システムの実装

5.1. クライアントー実際のサービス間の接続

サービス管理システムを実装するうえで，初めにクライアントと実際のサービス間の接続を図6に記す．

実際のサービスは，セキュリティ強化の理由で定期的に変更される位置情報とインターフェース情報を持つ．ここで，位置情報とはDistributed Objectsシステムでクライアント・サーバ間を接続するのに用いられるASCII文字列とする．そして，それらの情報はService Spoolに蓄えられる．Client Objectsは，Service Spoolに蓄えられた情報を元に実際のサービスと接続する．また，Service SpoolにサービスのInformationを登録するのはClient Controlが行う．その折に，Client Controlはサービスの情報が追加されたことをClient Objectに伝える．

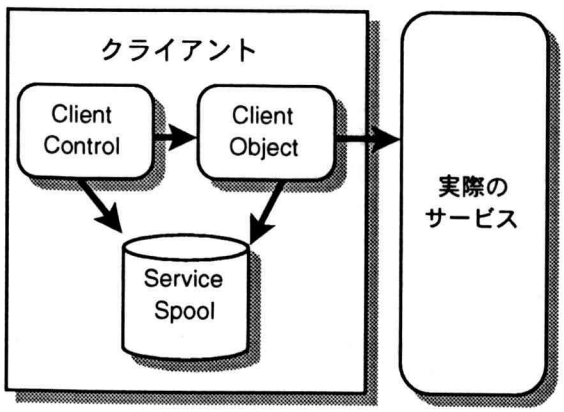

図6 クライアントーサービス間接続

5.2. クライアントーサービス管理サーバ間接続

次に，クライアントとサービス管理サーバ間の接続を図7に記す．

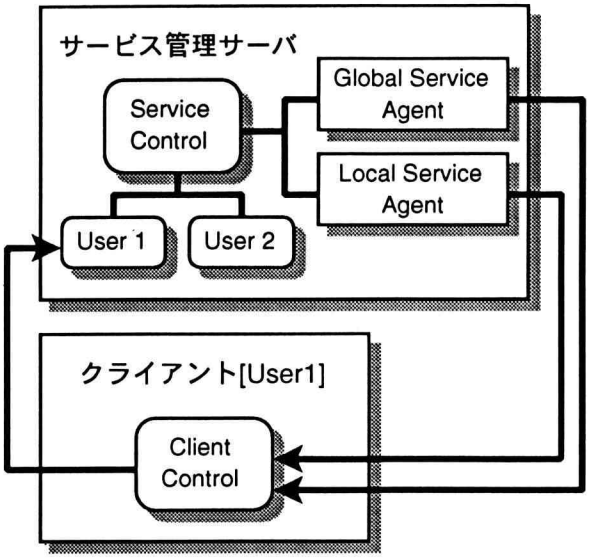

図7　クライアントーサービス管理サーバ間接続

Client Controlは，前述したService Spoolに登録する情報を得るために，サービス管理サーバと接続する．それについて，Distributed Objectsの機能により，接続に要するASCII文字列としてユーザ名を用いて，サービス管理サーバのUser Objectと接続する．図7では，クライアントを利用しているユーザをUser1と想定しているので，User1 Objectと接続される．その接続の折に，User Objectはパスワードでのユーザ認証を行う．接続が確立されると，Client Controlはサービス管理サーバにClient Controlの位置情報を引き渡す．その位置情報は，グローバルサービスの管理を自立的に行うGlobal Service Agentとローカルサービスの管理を自立的に行うLocal Service Agentに登録される．それらのAgentは，初めに自分の管理するすべての情報をClient Controlに引き渡し，以後，自分の管理するサービス情報が変更されたらその情報をAgentに登録されているすべてのClient Controlに送信する．これについては，OPENSTEPフレームワークの通知オブジェクトで実装する．

5.3. サービス管理サーバ間接続

次に，サービス管理サーバどうしの接続を図8に記す．

サービス管理サーバ2は，Client Controlから受け取ったクライアントの位置情報を上位階層のサーバであるサービス管理サーバ1に渡す．その位置情報は，Global Service Agentに登録される．このAgentは，前述したAgentと全く同じ機能を持つ．これらの一連の動作が上位サーバの登録のないサーバまで繰り返される．ここで，上位サーバとの通信には，タイムアウトを設ける．そして，もし上位サーバとの通信がタイムアウトとなったら上位サーバの登録がないものとみなし，それまでに接続されたサーバの範囲でサービス情報の提供が行われる．

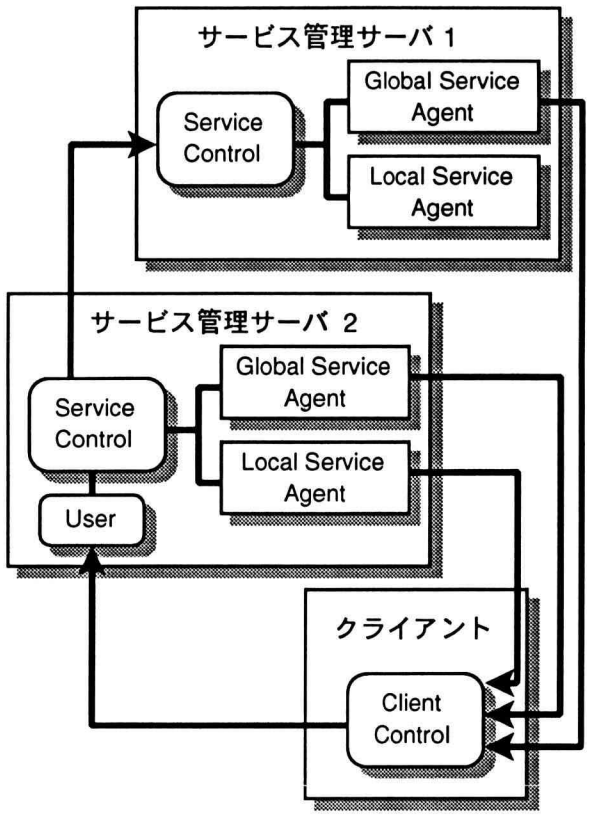

図8　サービス管理サーバ間接続

6. むすび

本論文では，透過的なサービス管理に関して，複数のサーバを連動させ，ユーザ名を使ってクライアントとサービス管理サーバ間の接続を行うことにより実現できることを明らかにした．また，環境の変化への柔軟な対応については，サービス情報の管理を自立的に行うサービス管理エージェントを配置し，また，リモートメッセージにタイムアウトを設定することにより実現できると考えられる．セキュリティの配慮に関しては，まだ不十分であるが，実際のサービスの位置情報を定期的に変更し，それをサービス管理サーバで管理する機構と，クライアントがサービス管理サーバに接続する際に，パスワードによる認証については，実現できると考えられる．今後の課題としては，セカンダリサービス管理サーバの配置や，メッセージの暗号化などを考えている．

参考文献

1) OPENSTEP Reference :NeXT Software,Inc. :1996.
2)U.D.ブラック :情報ネットワークシステムの基礎 :トッパン:1995.