

IN をベースにした呼処理アーキテクチャ

寺島 美昭[†]、清水 桂一[†]、伊藤 修治[†]、水野忠則[‡]

三菱電機（株） 通信システム研究所[†]、情報電子研究所[‡]

247 神奈川県鎌倉市大船 5-1-1

あらまし 本論文ではPBXを基本とするプライベートネットワークを対象に、サービスの高度化、構築性向上を実現する呼処理アーキテクチャを議論している。この目的を達成するには、従来のように呼処理サービス資源に閉じた方法ではなく、他系サービス資源と協調による適用範囲の拡大、及びアプリケーションを考慮したトップダウンアプローチによる呼処理機能構成の検討が必要である。このため我々はサービス資源に共通な分散協調基盤コンセプトに基づき、クライアント/サーバ型のサービス資源の相互利用を効率的に実現する呼処理アーキテクチャを提案している。

和文キーワード インテリジェントネットワーク、分散システム、呼処理、クライアント/サーバモデル

Call Processing Architecture Based on IN

Yoshiaki Terashima[†], Keiichi Shimizu[†], Shuji Ito[†] and Tadanori Mizuno[‡]

Communication Systems Development Laboratory[†],
Computer & Information Systems Laboratory[‡],
Mitsubishi Electric Corporation

5-1-1 Ofuna, Kamakura, Kanagawa 247, Japan

Abstract *This paper focuses private networks and discusses call processing architecture that can achieve high capability and productivity for telecommunication services. For accomplish our purpose, it is necessary to have ability to increase fields of call processing applications by co-operation with other service resources and functional structure considered by a topdown approach from application view. We propose call processing architecture based on a common distributed concept that enables efficiently to cooperate with several service resources.*

英文 key words Intelligent Network, Distributed System, Call Processing, Client/Server

1 はじめに

我々はPBXを基本とするプライベートネットワークを対象に、サービスの高度化、構築性向上を目的としたアーキテクチャの検討を行なっている。このサービス制御を実現する呼処理ソフトウェアは、システムが収容する多彩な通信資源に対して複雑な通信処理を行なうと同時に、高度な多重動作、及び信頼性など厳しい性能条件が求められる。このため現状の呼処理アーキテクチャは、機能間を密結合とするなど条件を満足する事を優先し考えられており、拡張性、保守性を損う原因となっていた。

しかしハードウェア性能向上など技術革新が進み、処理能力に対する障害は次第に除かれつつある。同時に高速/大容量のネットワーク性能の充実/普及が進んだ事により、呼処理に限らず様々なサービス資源において、従来異質であった資源間が相互に利用し合う事により、サービスを高度化しようという期待が高まっている。呼処理機能に対しても、他のサービスとの相互利用によるサービスの拡大、及び体系的な機能構成の実現による、サービス構築性能の向上に対応したアーキテクチャが求められている。

この問題に対して、我々は分散協調アーキテクチャに基づいた呼処理の提案を行なう。これは様々なサービス資源に共通な分散コンセプト上へ、それぞれのサービス資源分散処理インタフェースを実現する事により、クライアント/サーバ型の協調としてサービスを実現するものである。このコンセプト上でINモデルをベースとした呼処理機能アーキテクチャを実現する事により問題点の解決を目指す。

2 呼処理オブジェクト設計方針

プライベート網構成の基本要素となるPBXに収容される呼処理は多様な通信リソースを収容し、これらの間にスイッチングによる通信路を設定する制御機能である。この機能は公衆網と比較すると、回線容量、信頼性などに対する条件が比較的軽減される反面、マンマシンに近い、きめ細かいサービスを短いサイクルで実現する事が求められる。このためサービスの構築性、高度化アーキテクチャが強く求められている。この種の問題は呼処理に限らず、全てのサービス資源において議論されている重要な問題である。現在、有望な解決策として研究されているアプローチは、個々のサービス資源アーキテクチャ内で閉じた実現手段ではなく、それぞれのサービスの特徴をネットワーク上に提供し、これらが相互に利用し合う協調として、効率良く目的を達する事を目指した分散処理によるアプローチである。

このため、分散処理を考慮した呼処理アーキテクチャに関する研究が活発に行なわれている。従来交換に閉じた技術により構築されていた呼処理アーキテクチャに対して、他の分野との統一された思想による検討が必要であり、この結果が将来的な知的な通信の実現につながるという指摘は[1]など各方面で行なわれている。そして具体的な実現のための技術として、オブジェクト指向モデルに基づく呼処理アーキテクチャの研究が数多く発表されている。[2],[3],[4],[5]の提案は、局用交換機を対象としたシステムの階層化による機能構成の明確化、及び機能をビルディングブロックとして分散する事による機能の高度化を目指すものである。また[6]の提案はサービスの部品化に基づくサービス構築の効率化、及び機能の高度化を目指したものである。さらに[7]は、CTRONベースにオブジェクト指向実行環境を実現する提案である。これらはいずれも呼処理機能をオブジェクト指向モデルにより分析し、既存サービスを含めたサービス効率的な構築/保守/拡張を目指すものであり、他のデータベース専用マシンの利用など、従来異質であったサービス資源の効率的な利用を可能としている。

また標準アーキテクチャの立場からの研究が、CCITTや各種ベンダ共同体などで行なわれている。この代表的なものがCCITT

によるインテリジェントネットワーク(IN: Intelligent Network)であり、主として公衆網を対象としたサービスアーキテクチャを、キャリアの立場より研究し、マルチベンダによるサービス開発の可能性を探っている。

しかしこれらの研究の多くは、主に公衆網のような通信機能自体の高度化を目指す視点で検討されているが、プライベート網を対象とした呼処理アーキテクチャには、きめ細かいサービスの構築性などの条件を満たす方法が不可欠である。この実現には呼処理主体に他のサービス資源を利用するという検討のみではなく、対等な関係による相互利用の方法が必要である。しかし従来の研究には利用者の立場による、サービスの実現というトップダウンに考えた分散協調の視点によるものは少ない。CCITTによるINの1992年最新勧告でも、ネットワークの高度化を目指した概念的なモデルの提案と、限定されたケーススタディに関する検討結果が報告されるが、サービス資源の分散協調という視点による検討は次回以降となり、現段階では厳密なモデルは明らかにされていない。さらに類似の研究としてコンピュータ-PBX連係システムがある。これはコンピュータ、PBX呼処理サービス資源の相互利用を目的とし、従来PBX内に閉じた呼処理サービスプロセスの制御に関して、PBX外へのインタフェースを用意したものである。この結果、コンピュータ利用により呼処理としてはサービス範囲を拡大させる事ができたが、実現方法は従来アーキテクチャの一部を拡張サービス制御として外出しにしたに過ぎない。このためコンピュータと呼処理が密結合しており、サービスの再構築に複雑な手順が必要となるなど、サービスの拡大はある程度果たしているものの、サービスの発展性、構築性という観点からは限界がある。

以上の考察により我々は分散協調によるサービスの相互利用という視点からの研究を行なった。つまりこの環境上では、分散協調に対する一つのサービス要素として呼処理アーキテクチャが実現される。この位置付けを図1に示す。

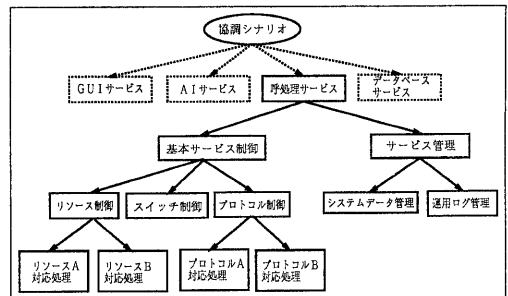


図 1: 分散環境上での呼処理の位置付け

呼処理が協調するサービス資源としては、情報処理系の資源として捉えられるデータベース、GUI、AIなどが想定される。これらの資源に対する分散処理技術の研究は、既に分散コンピューティングなどの枠組で研究され、一部の技術は実用になりつつある。例えばオブジェクト指向に関する標準化団体であるOMGによるOMA(Object Management Architecture)や、その他、ワークステーションベンダを中心とするグループなどから幾つかの提案があり、具体的な製品が発表され始めている。

我々は以上の要求と動向を考慮し、呼処理と情報処理系サービス要素との対等な関係による協調を目的に、各種サービス資源に共通のコンセプトとして分散協調基盤を提案し、この技術に基づ

く呼処理アーキテクチャを実現する。この考えに基づき、呼処理アーキテクチャを次に示す方針により、オブジェクト指向モデルに基づくモデル化を行なった。

- 呼処理にはサービス制御とサービス管理の側面があり、制御には構築性、管理には保守性が主として求められる。
- サービス制御は呼処理資源であるリソース、スイッチをプロトコル制御により操作する機能を基礎とする。これはプリミティブな通信制御機能と捉えられ、基本サービス制御の機能により呼処理資源に閉じたサービスを実現する。この機能からは呼処理が収容する多彩なプロトコルの違いは隠蔽され、仮想プロトコルとして認識される。
- 基本サービス制御を他のサービス資源と協調させる事により、ローカルな呼処理サービスを、ネットワークワイドへ拡大するために協調シナリオを実現する。

この結果、図1に示す呼処理の位置付けと定義し、この機能をサーバとして、そして協調シナリオをクライアントとして実現する。このためクライアント／サーバ型の効率的なサービス資源の利用が可能となり、INの特徴である基本制御と拡張制御の分離した考えに一致したアーキテクチャを実現できる。

3 分散処理基盤

3.1 システム概要

各種サービス資源を協調により実現するための、分散協調アーキテクチャ概要を図2に示す。

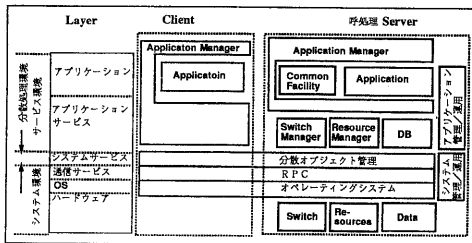


図 2: 分散協調アーキテクチャ概要

このアーキテクチャは、呼処理と他の情報処理資源の協調を共通の基盤により実現する事を重視したものである。このため分散環境を構築する各ノードの特徴を、それぞれが収容する特徴的な資源とこれら进行操作する方法に分類して捉え、この操作方法を分散処理環境へ提供する事が考えられている。このサービス実行イメージを図3に示す。

ここに示すようにコンピュータ上に存在するサービス制御オブジェクトは分散協調を制御するシナリオに位置付ける事ができる。これはクライアントとして動作し、テレコム系端末を制御する端末オブジェクト、及びデータベースオブジェクトをサーバとして利用する事によりサービスが実行される。この環境により呼処理にとっては、従来利用の難しかったリレーショナルデータベース機能などを容易に取り込む事が可能となる。

以上の環境を実現するためのアーキテクチャ機能を、レイヤ構成により整理して説明している。これらのレイヤは、その役割により各ノード独自の環境であるシステム環境、分散協調の共通基盤を実現する分散処理環境、及びアプリケーション構築に関するサービス環境の3環境に分類する事ができる。

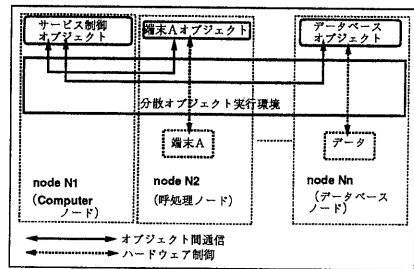


図 3: 分散環境上でのサービス実行イメージ

3.2 システム環境

この環境はハードウェアレイヤから通信サービスレイヤまでにより構成されている。ハードウェアレイヤは、各ノードの特徴となるハードウェア資源を持つレイヤである。例えば呼処理ノードにおいては、呼処理機能の操作対象である各種リソース、スイッチ、システムデータを蓄積しているファイルなどが実装される。OS レイヤは、各ノードの持つ基本的ソフトウェア、ハードウェア資源を、カーネル機能、及びデバイス機能という仮想的な概念で提供する OS 機能である。例えば呼処理ノードにおいては CTRON、コンピュータノードにおいては UNIX, MS-DOS などが想定される。通信サービスレイヤは、分散システムを実現するために各サービス資源間で通信処理を行なうレイヤである。これらの機能は、上位レイヤに対して RPC 機能を提供する。

3.3 分散処理環境

この環境はシステムサービスレイヤにより実現される。このレイヤは各種サービス資源の環境の差を隠蔽し、上位レイヤに対して共通の分散オブジェクト実行環境を提供するレイヤである。このノードの違いの隠蔽方法を図4に示す。

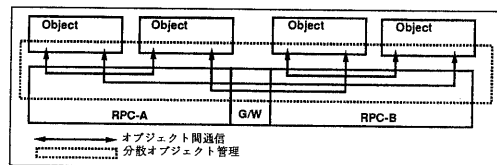


図 4: 通信アーキテクチャの隠蔽

例えばこの図では、RPC-A と RPC-B という RPC 上に分散オブジェクト実行環境を実現する事を示している。この環境によるオブジェクト間インタフェースのバインディング方法は、次の2方法により定義できる。

- 静的インタフェース定義方法：インタフェース定義言語を用意し、コンパイラによりクライアントとサーバのスタブを生成する方式。
- 動的インタフェース定義方法：送信されたメッセージがオブジェクトへ到達してから、動的にオブジェクトの内部処理と結合される方式。

また、RPC-A と RPC-B の関係のように RPC 方式が異なるノード間の場合は、ノード間にゲートウェイを設ける事により、

上記の2方式の利用を可能とする。

分散オブジェクト管理には図5に示す機能が用意される。これらの機能要素について次に説明する。

- オブジェクト実行管理：オブジェクトの実行に関する管理を行なう。例えばオブジェクトの生成／消滅、インタフェース定義などを行なう。各ノードで動作するオブジェクトは、上位レイヤからは同一のものに見えるが、それぞれのサービス資源の動作条件により実現形態は異なる。
- セキュリティ管理：分散環境で動作するセキュリティを管理する。オブジェクト実行レベルのセキュリティを対象としており、個々のサービスに依存するロジック的なセキュリティはサービス環境により実現される。
- ディレクトリ：ディレクトリサービスは分散環境上でオブジェクト実行のための、環境内でユニークなネームを管理している。
- 時刻管理：時刻管理は分散処理を構成するノード間で、共通の時間を持つ事を目的とする。これは運用ログの収集を始めとする同期的なサービスなど、システムの管理に不可欠な要素である。

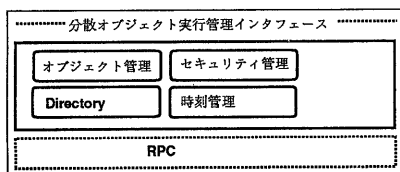


図 5: 分散オブジェクト管理構造

これらの要素技術は各サービス資源に共通のコンセプトとして用意される。各サービス資源は、それぞれの条件を満足する方法で、これらの技術を実装方法を検討しなければならない。この結果実現される分散オブジェクト実行環境は、それぞれのサービス資源からは同一のコンセプトとして参照できるが、それぞれ独自の条件を隠蔽しなければならない。

呼処理動作を厳しい性能条件の下に実現するためには、既に[3]などに提案されているように呼を実現するためのオブジェクト実行モデルが必要である。呼処理の並列実行性などを考えた場合、単一の能力のオブジェクトのみではサービスを表現するには不十分であり、並列／直列オブジェクト能力の考え方が必要となる。本アーキテクチャ上では、オブジェクト実行モデルはオブジェクト実行管理により実現される。さらに分散オブジェクト実行環境として実現するために、オブジェクトIDのディレクトリによるネームとしての実現、オブジェクトに対するセキュリティ機能の実現、時刻管理を利用した分散環境上での同期方法などに関して検討を深める必要がある。

3.4 サービス環境

この環境はアプリケーションサービスレイヤとアプリケーションレイヤより構成される。アプリケーションサービスレイヤは分散オブジェクト環境上で、そのノードが提供するサービス資源をオブジェクトとして提供するレイヤである。呼処理サーバとして提供する資源とは各種リソースとスイッチであり、これらの機能を効率良く制御するインタフェースが実現される。このレイヤは、

サービス資源にアクセスする方法であるコマンド、プログラムライブラリなどを総合的に管理している。この中でプログラムインタフェースでは、コンパイル形式、インタプリタ形式が用意される。例えばコンパイル形式は実運用に対して適用され、インタプリタ形式は試験などサービス導入に関して適用される。

さらにこのレイヤに関しては今後ともサービス生成などソフトウェア開発手法、及びシステム運用中のサービス追加、障害の対処などを含めて研究が必要である。本論文では呼処理の提供するサービス資源を、オブジェクトインタフェースとして提供する方法を次章以降に説明している。

アプリケーションレイヤでは、実際に各サービス資源の提供するアプリケーションサービスレイヤ上で、それぞれの資源に対する操作をサーバとして利用するクライアント／サーバ型のソフトウェアが実現される。例えば呼処理サーバでは、INアーキテクチャに基づくサービス実現の基本要素を以下のように実現する。

- SCF(Service Control Function): サービスシナリオに位置づけられ、呼処理サーバを利用するクライアントとして実現される。
- CCF(Call Control Function)／SSF(Service Switching Function): 基本的な呼処理制御を行う機能であり、SCFと同様にアプリケーションとして実現されるが、呼処理サービス資源のみを制御する。また必要に応じてSCFを起動する。
- SDF(Service Data Function): INサービスデータを管理する機能であり、必要に応じて呼処理ノードのデータベース、あるいはデータベースノードなど適切な位置に存在する。

さらにCommon Facilityはサービス資源を制御する共通機能をオブジェクトライブラリとして用意しているものである。INにおいて規定されるSIB(Service Independent Block)もこの形態で実現する。

このようなクライアント／サーバ型のサービス実現方法は、INの特徴である基本呼制御と拡張制御を分離した形態を効率良くマッピングできる。実際の物理的なオブジェクトの位置を考えた場合、SSF/CCFという基本呼に関わるオブジェクトは呼処理の厳しい性能のために呼処理ノードに存在する。拡張制御の存在位置は限定されないが、多くの場合プログラマブルな性能に優れているコンピュータノード上に実装される。基本呼制御オブジェクトはサーバとして動作し、拡張制御オブジェクトはクライアントにマッピングできるため、呼処理を含めた様々なサービス要素を協調させるシナリオを、INの考えに一致した方法で実現する事が可能である。

4 リソース制御

本章では呼処理サーバが提供するリソースをオブジェクト動作として捉え、その操作インタフェースを分散環境上へ提供するための検討を行なう。

4.1 リソースの分類

リソースの実体は通信／トランスポートリンクなどハードウェアとして実現されるもの、あるいは端末グループなど実体の存在しない仮想的に実現されるものなど様々である。表1に呼処理が収容するこれらのリソースを分類する。またリソースはその位置付けを明らかにするために、図6に示すクラス階層として関係を整理する。

リソース種別	機能	例
ラインリソース	プロトコル制御の対象となるリソース	アナログ電話、デジタル電話、ISDN 回線、デジタル専用線など
共通トランクリソース	共通トランクとして用意されるリソース	会議トランク、モデムプールなど
トーンリソース	トーン関係のリソース	トーン／トークトランク、多重制御トランクなど
ハードウェアリソース	ハードウェア資源となるリソース	カード、ハイウェイ装置、クロックなど
仮想リソース	仮想的に取り扱われるリソース	内線代表グループ、方路、トランクグループなど

表 1: リソースの分類

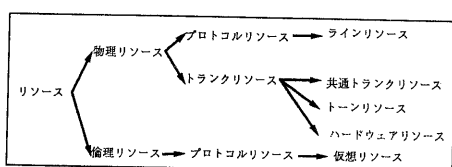


図 6: リソースクラス階層

4.2 リソースのオブジェクト動作

リソースはサービス制御によるアプリケーション構築性を考慮して、それぞれをオブジェクトとして実現する。しかし呼処理に要求される機能は、既に説明したようにサービス管理の立場も同時に存在する。この機能では保守や障害時に、リソースの運用状況を迅速に把握する必要がある。しかしサービス制御の効率性を考慮したリソースオブジェクトの実現では、必要となる個々の通信情報はオブジェクト内に隠蔽されており、収集のためには全てのアクティブなオブジェクトを、オブジェクト実行管理に問い合わせた後、それぞれオブジェクトへ情報を要求しなければならない。これは迅速性を求めるサービス管理にとっては効率が悪い方法である。このためリソース・オブジェクトをリソースマネージャオブジェクトにより集中管理させる方式を採用した。全てのリソース・オブジェクトの実行状況を、常にリソースマネージャ・オブジェクトが把握する事とする。さらに各リソース種別が持つ多様な情報による動作は、単一の管理では複雑になり難いと考え、リソース種別対応のリソース個別マネージャ・オブジェクトにより管理させる。従って個々のリソース・オブジェクトは、それぞれのリソース種別対応のリソース個別マネージャ・オブジェクトと、これらを統括するリソースマネージャ・オブジェクトによる段階的な構成により管理される。

この構成によるリソースの動作を説明する。このリソース管理を図7に示す。リソースマネージャ・オブジェクトはシステム起動時に生成される。このオブジェクトは、リソース種別単位のリソース個別リソース・オブジェクトの生成/消滅を管理する。それぞれの生成は、システムにとってはリソースの収容を意味する。従って特定のリソース属性には依存せず、システムデータベースに蓄積されているリソースの属性に関する情報、及び収容リソース情報などを元に個別リソースマネージャ・オブジェクトを生成する事となる。

またリソース個別マネージャ・オブジェクトは、そのノードに

収容されている単一種別のリソースの状況を管理しており、アプリケーションからアサインの要求がある場合には、リソースを活性として登録し、対応したリソースオブジェクトを生成し管理する。この時、リソースオブジェクトには個別情報がシステムデータベースに登録されており、この情報を属性として保持するオブジェクトが生成される事となる。

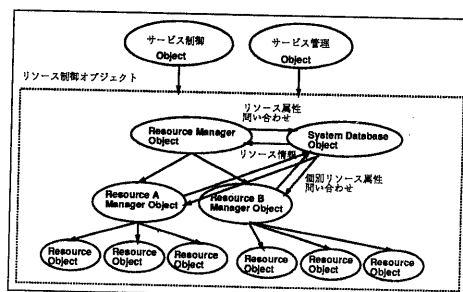


図 7: リソース管理

以上のオブジェクト構成により、リソースを制御するという立場より考えた呼処理に対し、適切な機能を提供する事ができる。

5 スイッチ制御

本章では呼処理サーバが提供するスイッチをオブジェクト動作として捉え、その操作インタフェースを分散環境上へ提供するための検討を行なう。

5.1 スイッチ処理のオブジェクト動作

スイッチに関しても、リソースのオブジェクト指向化と同様にサービス制御とサービス管理の要求があるため、個々のスイッチを管理するスイッチマネージャ・オブジェクトの管理下において、個々の通信対応のスイッチ・オブジェクトを用意する。スイッチ処理は、そのノードの持つスイッチ能力を統括する位置付けのスイッチマネージャ・オブジェクトと、リソース間の通信路の接続手段とする位置付けのスイッチ・オブジェクトにより実現される。図8に、この関係を示す。

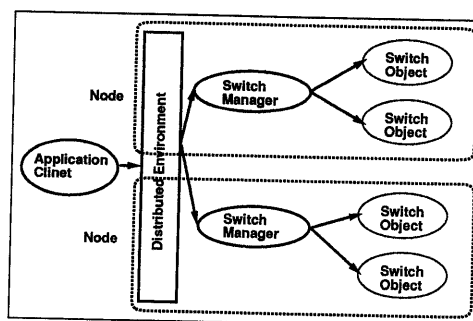


図 8: スイッチにおけるオブジェクト構成

この図に示すように、アプリケーションからは分散環境を介して物理的存在位置は意識せずに、スイッチ操作を行なう事ができる。

個々の通信チャンネルは次の情報を持ち、スイッチ・オブジェクトによって保持される。

- 関係付けの対象となる 2 つのリソース
- 接続方向
- 占有するチャンネル

通信路のアサインは図 9 に示すように、スイッチ・オブジェクトの生成が対応付けられ、同じく解放が消滅に対応される。生成されたスイッチ・オブジェクトが以降チャンネルを管理し、呼処理サービスを実行する。

スイッチ・オブジェクトはスイッチング・ハードウェアの動作を反映するが、このハードウェアは主体的には状態を変更しない。このためスイッチング・オブジェクトはスイッチハードウェアに対して、常に能動的に動作する。

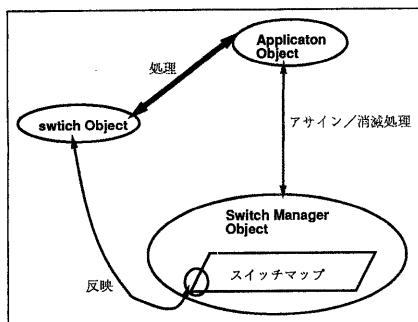


図 9: チャンネルアサインの動作例

5.2 オブジェクトとハードウェアの動作

ここでは呼処理ノードに存在するメインスイッチについて説明したが、同様のスイッチ処理は多重系をもつリソースの管理についても必要となる。これは呼処理ノードにとってはサブスイッチに位置付けられるが、メイン/サブの双方のチャンネル管理において、できるだけ共通に利用できるアーキテクチャを実現する事が必要である。

またスイッチング動作を考える場合、スイッチ・オブジェクトが既に説明したように、スイッチハードウェアと密接に関係して動作する事を考慮しなければならない。このためスイッチオブジェクトは、オブジェクト間インタフェースを持つと同時に、スイッチハードウェアを制御するファームウェアともインタフェースを持たなければならない。あるいは、スイッチハードウェアを隠蔽するオブジェクトを定義する方法なども考えられ、今後検討を深める必要がある。

6 プロトコル制御

リソース、スイッチという呼処理資源を制御する、プロトコル制御について説明する。この機能は対象となるリソース間で、それぞれがサポートするプロトコルによる通信路接続のための情報交換を行ない、その結果としてリソース間でスイッチによる通信路の制御を行なうものである。この機能は呼処理のプリミティブな通信制御機能であり、IN アーキテクチャにおける基本呼制御の一部に対応され、Common Facility などの実現形態で呼処理サーバに用意される。

6.1 プロトコル処理のオブジェクト指向化

サービス制御へプロトコル種別に依存しない仮想プロトコルの機能を実現するために、次の方針で検討を行なった。

- 回線接続型サービスの構築性を考え、呼のモデルとして既に [3] などで提案されている発着分離型の Caller/Callee モデルを採用する。
- 仮想プロトコル処理は、各種プロトコルに共通である通信状態間の遷移を基本単位として処理を行なう。これらの処理の要求を受け、各プロトコル対応オブジェクトは共通機能を独自の処理に変換する。
- 対象とする通信に係わるリソース、スイッチという通信情報の管理を一括するために、コールプロファイル・オブジェクトを定義する。

これらの方針に従って、プロトコルの仮想化制御の方法について、及びコールプロファイルによる呼の実現について説明する。

6.2 プロトコル制御構成

一般に呼処理のプロトコル機能には、多彩なプロトコルの収容が要求される。通信路の設定のために行なうリソース間の情報交換は、同一のプロトコル間、あるいは異なるプロトコル間で行なわれる事になる。しかし多くの呼処理サービスは、例えばデジタル電話、アナログ電話、ISDN 回線などのプロトコルの違いはなく、全ての端末、回線で同一のサービスが提供される事が求められる。このためサービス制御からは個々のプロトコルの違いは隠蔽しなければならない。さらにプロトコル追加があった場合などは、追加プロトコルに対しても既存サービス全ての実装しなければならない。構造が複雑になり保守性が劣化させる原因となっていた。この問題に対照し効率的にプロトコル制御を実現するためには、仮想化した機能としての制御を提供する事が重要である。この要求を満たすため、図 10 に示すように仮想プロトコル制御部、及び各種プロトコル対応部に分けて管理する。

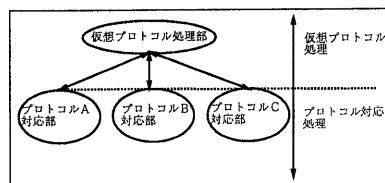


図 10: プロトコル処理機能の概要

ここでは、プロトコル処理を行なうための論理的な要素をオブジェクトとして取り扱う。また仮想プロトコル処理とプロトコル対応処理間のオブジェクト・インタフェースをプロトコルに依存せず統一する事により、プロトコルを仮想化した処理として捉え制御される事が可能である。各種プロトコル対応部オブジェクト群は、対応するリソースオブジェクトと必要に応じて情報交換する事により、プロトコル動作を実現する。この情報はリソース属性に関わるものもあるが、このような情報は必要に応じてリソース・オブジェクトへ問い合わせる。

この結果実現される Caller オブジェクトによる発信処理の例を、図 11 に示す。

この図ではサービス制御が要求する接続要求が、最終的にプロトコルとして実現される過程を説明している。サービス制御オブ

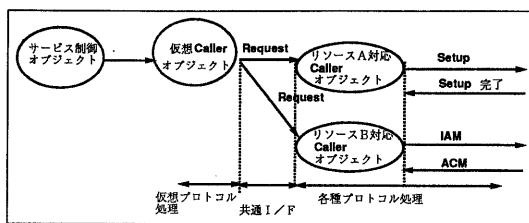


図 11: 各種プロトコル処理

ジェクトからの要求は、最終的にプロトコルに依存した処理としてリソース A の場合は Setup メッセージに、リソース B の場合は IAM メッセージに変換される。この方法による情報交換の結果、必要に応じてリソース対応の Caller オブジェクトが、既に説明した呼処理資源であるリソース、スイッチの操作を、それぞれリソースオブジェクト、スイッチオブジェクトなどを利用する事により実現する。

6.3 仮想的なプロトコル制御

仮想プロトコル制御は安定した状態間の遷移を指示する機能である。この状態はリソースとスイッチの状態の総合状態として表現される。個々のリソースにはさらに詳細な状態が存在している場合があるが、ここでは各リソースに共通の状態のみを取り扱い、リソース固有の状態との組合せについては、個々のリソース対応オブジェクトの取り扱い状態として保持する。この通信状態と、これらの状態間の遷移を次の図 12 と、表 2 に示す。

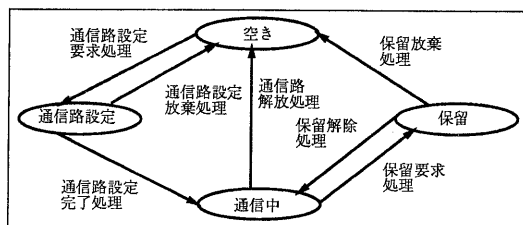


図 12: 基本的な状態遷移

この表に示す状態遷移に係わる処理が、仮想プロトコル制御へサービス制御から指示される機能となる。

6.4 コール・プロファイルによる制御

仮想プロトコル制御の操作対象である通信状態を管理する実体を、コールプロファイル・オブジェクトとして定義する。このオブジェクトは呼として存在し、通信路情報として次のものを保持している。

- 関連リソースオブジェクト情報
- 関連スイッチオブジェクト情報

この関連とは制御対象になっているという意味である。

また通話路情報は、これらリソースとスイッチによる関係として定義される。この通信路のモデルは図 13 に示す。ここでは 2 者間接続の例を (a) に、また 3 者間接続の例を (b) に示している。

処理名	状態遷移
通信路設定要求処理	空き → 通信路設定の遷移を行なう。
通信路設定放棄処理	通信路設定 → 通信中の遷移を行なう。
通信路設定完了処理	通信路設定 → 通信中の遷移を行なう。
通信路解放処理	通信中 → 空きの遷移を行なう。
保留処理要求処理	通信中 → 保留の遷移を行なう。
保留解除処理	保留 → 通信中の遷移を行なう。
保留放棄処理	保留 → 空きの遷移を行なう。

表 2: プロトコル処理

通信路はリソース間のスイッチングにより実現される。言い換えればリソースオブジェクト間の関係を、スイッチオブジェクトとして表現しているという事ができる。また 3 者間接続も実際のハードウェア的な接続イメージに則して、会議トランクオブジェクトを介した接続と定義できる。このように実際のハードウェア構成を反映する事により、仮想プロトコル・オブジェクトからは単純化されて参照できる。

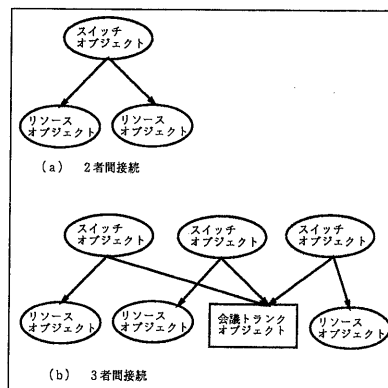


図 13: 通信路設定例

以上の検討により、表 2 に示す処理の中の通信路設定要求処理において、コールプロファイル・オブジェクトの位置付けは図 14 のようになる。

通信の一連のプロセスである呼に対応し、アプリケーションが呼処理機能とアクセスする場合は、始めにコールプロファイルが生成される。またこのオブジェクトを通信サービスを解放するまで保持する。

呼に対する制御の一貫性を保証するために、このオブジェクトはプロトコル対応オブジェクトからの指示により状態遷移を実行するが、基本的に単一のオブジェクトからの指示のみを許す。そしてコールプロファイル・オブジェクトネームをオブジェクト間で受け渡す事により、結果として通信の制御権移行が行なわれる。ただし例外として、呼処理に対する障害処理などの実行性能に対処する処理が存在する。このため、次の処理においては制御権

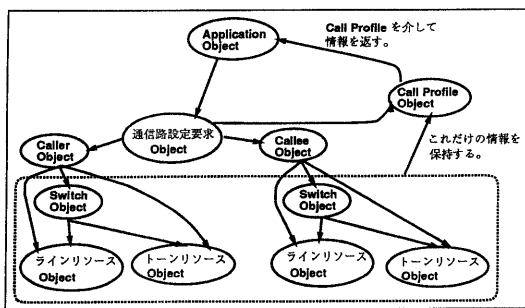


図 14: コールプロファイル・オブジェクトの位置付け (通信路設定処理)

を持つオブジェクト以外からのアクセスも許されなければならない。

- 障害発生時の運用／管理を実現するアプリケーションオブジェクトからの強制切断、障害閉塞、保守閉塞などの処理。
- コールプロファイル・オブジェクトの暴走。(このコールプロファイル・オブジェクト)を操作するオブジェクトが存在しない状態、あるいはオブジェクト通信不可などの障害状態。)

コールプロファイル・オブジェクトを生成したアプリケーションオブジェクトからの操作と、それ以外からの操作に関してはセキュリティ管理が必要であり、この方法についてはさらに検討が必要である。例えばコールプロファイル・オブジェクトに対して生成アプリケーション、あるいはそのグループをオーナーとする操作が許されるが、この機構では上記のような機能に対しては、特権を持つスーパーバイザ的な操作が許可されなければならない。

7 問題点

本論文では提案する呼処理アーキテクチャの基本的な考えについて説明したが、さらに多くの検討が必要と考えている。この検討項目の幾つかについては本文の中で指摘しているが、これらの課題に加えて次のような視点が必要である。

- 呼処理の特徴である高度な多重動作に対応するための、分散環境上におけるオブジェクトの並列的動作モデルの実現。
- オブジェクトのプライオリティの管理方法。
- ハードウェア動作とオブジェクト動作の関係についての検討。例えばリソースオブジェクトの中には、ハードウェア状態を反映するものも存在する。必要に応じて他のオブジェクトに対してオブジェクト間インタフェースを持つ、ファームウェアとして実現するなどの方法をとる事も考えられる。
- サービスが多重に実行される事によるオブジェクト輻輳の問題が予想され、オブジェクトの最適化などの検討が不可欠となる。
- 分散したサービスリソースの協調アルゴリズムの検討が必要となる。

特に呼処理の特徴として、通常ユーザへ提供するサービスのロジカルな実現手法を明らかにすると共に、多重実行性、ハードウェアリソースとの協調、及びメンテナンス性能、障害対処などに対する要求の厳しさを考慮しなければならない。

8 まとめ

以上、情報処理サービス資源との対等な関係による協調を目的とした、分散協調アーキテクチャのコンセプトを提案し、この考えに基づいた呼処理アーキテクチャを検討した。

このアーキテクチャは、アプリケーションマネージャによるリソース、スイッチという資源操作を、オブジェクトインタフェースとして提供するための検討、さらにこれらを制御する基本呼制御として、コールプロファイル・オブジェクトを呼の実体として定義した、プロトコル制御の実現を検討した。この実現では多様なプロセスを共通の仮想プロトコル制御の考えにより、クライアントとしての拡張制御の実現を可能としている。

この結果、共通コンセプトによる呼処理によりサービス拡大が実現できる事はもとより、従来必要とされた呼処理のサービス追加／変更に対して、既存設備に大幅な修正を加える事なく同一のコンセプト内で実現できる。また機能構造が整理され、リソース、スイッチという呼処理資源がサービス制御とは分離してオブジェクトとして独立性が高く実現されており、資源の追加／削除を含めた処理が各リソースマネージャの管理下でされる。これらは信頼性、保守性を要求するサービス管理にも対処する機能であり、労力的、経済的な効率向上を実現する。

今後、問題点の解決に加えて、マルチメディアの利用や、従来個別環境であったソフトウェア開発、ドキュメント作成、電話／FAX 通信など一般オフィス業務の統合、複雑な判断を含むインテリジェントビル管理などプライベート網に対して要求の高いサービスへ適用する事により評価を加える予定である。

参考文献

- [1] 野口 正一、"インテリジェントネットワークと知的コミュニケーション"、電通学会論文 91/11 Vol. J74-B-I, No. 11
- [2] 山田 茂樹、他、"オブジェクト指向交換プログラム向きのケーパビリティプロテクション方式"、SSE92-12
- [3] 丸山 勝巳、他、"通信網用分散処理プラットフォーム PLATINA" SSE91-170
- [4] 田中 聡、他、"分散オブジェクト指向交換プログラム向きの軽量プロセス実現法"、SSE91-100
- [5] 大崎 憲嗣、他、"オブジェクト指向交換プログラムにおけるプログラム更新"、SSE91-78
- [6] 大伏 健二、他、"オブジェクト構造交換ソフトウェアの実行環境"、SSE91-101
- [7] 山崎 準一、他、"C-TRON ベースのオブジェクト指向呼処理プログラムの試作"、SSE91-79