3V−5

INをベースとした呼処理アーキテクチャの検討

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‡

1 はじめに

呼処理サービスの高度化、及び効率的な構築性を実現するために、プライベートネットワークを対象に、INベースの分散呼処理アーキテクチャを提案する。これはネットワーク上におけるサービス資源の最適な配置、及び資源間の協調による効率的なサービスの実現を目指すものである。これにより電話系サービスを含めたマルチメディア利用、電話／FAX通信などを利用する一般オフィス業務の統合、インテリジェントビル管理のなどへの利用が期待できる。

2 分散処理技術の利用

既に呼処理サービス資源の一部をコンピュータなどへ提供する事により、サービスの高度化を目指したアーキテクチャが実用化されている。これらの多くは呼処理と他のサービス資源の関係を固定としたものが多く、相互の関係は密結合である。従ってサービス高度化はある程度達成するが拡張性に乏しく、分散された資源の有効活用という点からは限界がある。この問題を解決するためには汎用的な分散処理基盤によるサービス資源の協調を行なう事によりサービスの実現を目指す事が必要である。例えば図1に示すように、呼処理機能、データベース機能を必要に応じてサービス制御オブジェクトから利用する、クライアント／サーバ形態のサービスを実現する。

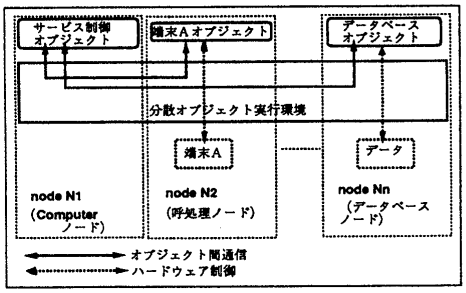

図 1: 分散環境上でのサービス実行イメージ

呼処理機能をサーバとして分散環境上に提供する事により、次に示す利点が得られる。

- 呼処理系サービスを含めた全てのサービスリソースを、共通のコンセプト上に構築する事により協調動作による呼処理サービスの拡大が実現できる。
- ノード間の機能分担を明確にする事ができるため、アプリケーションの構築性、及びメンテナンス性が向上する。これはサービスの構築性ばかりではなく、障害などの原因を特定しやすく対処が容易に実現できる事を示している。

以降、提案する分散協調アーキテクチャの概要を述べ、このコンセプトに基いた呼処理サーバの実現について述べる。

Call Processing Architecuture based on IN
Yoshiaki TERASHIMA, Keiichi SHIMIZU, Shuji ITO, Tadanori Mizuno
Mitsubishi Electric Corp. , Communication Systems Development Laboratory, Computer & Information Systems Laboratory

3 システム概要

図2に分散協調アーキテクチャの概要として、呼処理サーバの利用について記す。これは分散環境上に機能を提供する、全てのサービス資源に共通の考えとして規定する。

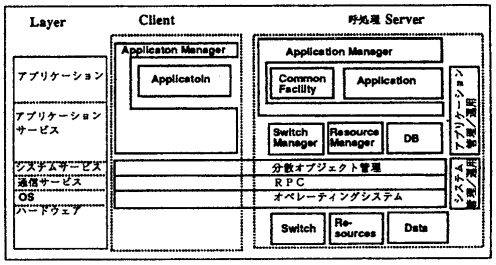

図 2: 分散協調アーキテクチャ

ここでは通信サービス以下のレイヤは各ノードに独自であるが、この差を図3に示すようにシステムサービスレイヤである分散オブジェクト管理が隠ぺいする。このためアプリケーションサービスレイヤへ分散オブジェクト実行環境を実現できる。またアプリケーションサービスレイヤが各種サービス資源を効率的に利用する環境を実現するため、アプリケーションレイヤにおいて、サービスを実現するソフトウェアをアプリケーションとして構築できる。

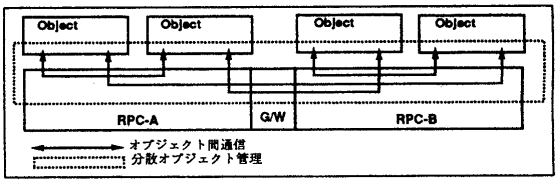

図 3: 通信アーキテクチャの隠蔽

呼処理サーバは回線交換機能を分散協調環境上へ提供するために、以下の機能を持つ。

- アプリケーションサービスレイヤ: 各ノードの持つサービス資源を分散環境上へ提供する。呼処理サーバでは、接続対象であるラインリソース、接続動作を行なうスイッチをオブジェクトとして実現する。
- アプリケーションレイヤ: アプリケーションをクライアント/サーバの形態で実現する。このクライアントからは各オブジェクトの存在位置は意識されないが、例えば呼処理ノードにはラインリソース、スイッチに関係するオブジェクトが存在する。これらを制御するオブジェクトは、一般に物理的ノードに依存しない論理的な機能を持つため、存在位置はリアルタイム性、オブジェクト規模などを考慮して適切に選択する事ができる。

このためINにおけるサービス実現の基本要素は、以下のように実現される。

- 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。

4 呼処理動作

例えば最も単純な呼処理サービスである2者間接続は、2つのラインリソースの間の通信路を設定する処理である。通信路の設定とは、アプリケーションの視点からは、端末、回線などのラインリソース間をスイッチングにより関係づける機能といえる。このため予め想定された手順、つまりプロトコルにより、ラインリソース間でスイッチングのための情報交換を行なう。呼処理サーバは基本的にラインリソース、スイッチという資源を持ち、クライアントからの指示によりこれらを操作する機能である。

4.1 リソースオブジェクト

呼処理ノードの持つリソースは多様なものがあるが、その性格により図4に示すようなクラス階層を定義する。

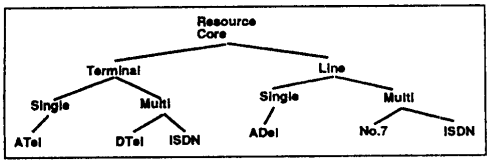

図 4: リソースクラス階層

呼処理の持つリソースは大きくは端末系(Terminal)とライン系(Line)に分かれ、それぞれに多重系(Multi)とそうでないもの(Single)が存在する。スイッチングの対象となるリソースはこの中のラインリソースである。これらのリソースはオブジェクトとして実現されるが、保守のために集中して情報を管理する必要があり、リソースマネージャが生成／消滅を管理する。そして必要な機能をオブジェクトインタフェースとして、アプリケーションへ提供する。

例として図5にリソースのアサインの例を示す。

ここではUser Objectの要求により特定のResouce Objectをアサインする例を説明する。処理は図に示された順番で進む。

1. User ObjectがResource Managerへ指定するリソース種別のアサインを要求する。2. Resource Managerは指定されたリソースをオブジェクトとして生成する。3. Resource Objectは、Resource Managerへ生成が完了した事を通知する。4. Resource managerは、生成したRessource ObjectのIDをUser Objectへ通知する。5. 以降、User Objectは必要に応じて、Resource Objectと通信して処理を実現する。

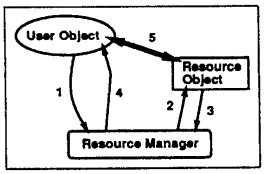

図 5: リソースのアサイン

4.2 スイッチオブジェクト

スイッチはリソース間の交換接続の関係を管理するものであり、スイッチハードウェア上の接続マップを反映するものである。スイッチオブジェクトはINにおける通信路のモデルに相当し、関係付けの対象となる2つのリソース、接続方向、占有するチャネルの3種類の情報を保持する。これらのスイッチオブジェクトはスイッチマネージャが生成/消滅を管理しており、アプリケーションレイヤに対してオブジェクトインタフェースを提供する。

4.3 データ管理

INに関するデータは、クライアントからサービス実行のために必要に応じて参照されるサービス実行のための情報、及びシステムを運用／管理するため参照する情報の2つの立場より参照される。

これらは全て分散データベース機能により同一のレベルで参照されるが、データが実際に存在する位置はデータの性格により決定されなければならない。例えば呼処理のリアルタイム処理を考慮すると、各交換ノードに存在しなければならない情報が存在する。一方、コンピュータに存在する持つデータベースを利用する方が効率が良い情報もある。さらに将来的には、知識ベースなどにより高度な判断機能を参照する事も考えられる。従ってデータの利用方法としては、1. 交換ノードに存在するデータ。2. 交換ノード以外に存在するデータ。3. 複数の関係するノードに存在するデータ。の3種類がある。これらのデータは一貫性の保持が不可欠であり、トランザクション的な処理が必要となる。

以上、これらの資源をオブジェクトとして分散環境へ提供する事により、INに基づいたサービスを構築する事ができる。同時に従来異質であった他のサービス資源との協調が可能となるため、呼処理利用の拡大が可能となる。

5 おわりに

本稿ではINの目的とするサービスの高度化、構築性向上を達成するアーキテクチャとして、分散協調アーキテクチャに基づいた呼処理サーバの実現を説明した。今後、呼処理に要求されるオブジェクトの多重実行、障害、サービス救済など信頼性に対する対処などに関して検討を加える予定である。

参考文献

[1] 丸山、他 : “オブジェクト指向による交換プログラムの構成法”、論文誌 B-I Vol. J74-B-I No.10 pp.757-768