マルチメディア通信と分散処理 42－11
（１９８９．５．１９）

OSI ディレクトリ･システムの実装と評価

小花貞夫　西山 智　鈴木健二
国際電信電話株式会社

筆者らは、OSI(開放型システム間相互接続)のディレクトリ･システムのソフトウェアを実装した。本ソフトウェアは、ア) OSIに完全に準拠し、今後の多様化する応用を実現するための応用層構造(ALS)に柔軟に対応できる汎用的なソフトウェア構成、イ) 汎用のリレーショナル型DBMSパッケージを用い、多様なディレクトリ･スキーマに対応できる柔軟なDIB(ディレクトリ情報ベース)機能、ウ) ディレクトリ操作のための分散処理機能、を効率よく実現している。

本稿では、OSI ディレクトリ･システム実装のための基本的な事項、a) OSIの応用層構造を考慮したソフトウェア構成、b) DIBの実現方法について検討し、ついでこれらの検討結果に基づいて実装したディレクトリ･システムのソフトウェアの詳細について報告する。さらに実証実験等を通して、ソフトウェアの機能、性能等についても評価、考察する。

Implementation of OSI Directory System

Sadao OBANA　Satoshi NISHIYAMA　Kenji SUZUKI
KDD Kamifukuoka R & D Labs. 2-1-15, Ohara, Kamifukuoka, Saitama, 356

Softwares for Directory System for OSI (Open Systems Interconnection) have been implemented. These softwares have an architectural flexibility to adapt OSI Application Layer Structure (ALS) which realizes wide variety of OSI applications, provide flexible and powerful DIB (Directory Information Base) handling facilities using a relational DBMS (Database Management System) software package, and provide distributed processing facilities for distributed operations over several DSAs.

In this paper, design strategies for the softwares are firstly discussed. Then detailed specifications of the softwares are discussed. Furthermore, functionalities and throughput of these softwares are evaluated.

1. はじめに

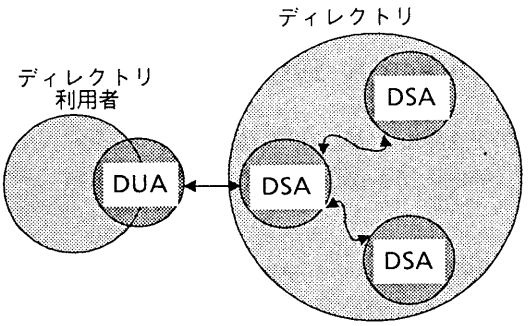
CCITTとISOでは、OSI通信で必要な通信支援機能を提供するOSIディレクトリ・システムの勧告/標準化[1]を、精力的に進めている。筆者らは、これまでにMHS(メッセージ通信処理システム)やFTAM(ファイル転送、アクセスと管理)のプログラムを実装しているが[3]、これらのプログラムを用いて、実際の環境で通信を効率よく行うためには、O/Rアドレス、プレゼンテーション・アドレスやファイル・ディレクトリなど通信相手に関する情報を提供するOSIディレクトリ・システムが必要となる。

そこで筆者らは、VAX8700(およびμVAX)VMS上にOSIディレクトリ・システム・ソフトウェアを実装した[4][5][6]。本稿では、実装のための基本的事項、特に応用層構造を意識した柔軟なソフトウェア構成、DIB(ディレクトリ情報ベース)の実現方法について検討するとともに、それに基づいて実装したソフトウェアの詳細について報告する。さらに実証実験等を通して、ソフトウェアの機能、効率等についても評価、考察する。

2. ディレクトリ・システムの概要

2.1 ディレクトリ・システムのモデル

ディレクトリは、通信の対象となる"物"(オブジェクトと呼ぶ)に関する情報のデータベースで、これらの情報に対する各種のアクセス(読出し、探索、変更など)機能を、利用者(例えば人や計算機プロセスなど)に提供する。ディレクトリが提供する情報をディレクトリ情報ベース(DIB)と呼ぶ。



DUA:ディレクトリ利用者エージェント
DSA:ディレクトリシステムエージェント

図1 ディレクトリシステム

利用者がディレクトリにアクセスする場合、実際にはディレクトリ利用者エージェント(DUA)と呼ばれる応用プロセスが、利用者に代わってアクセスする(図1)。

ディレクトリは、さらにディレクトリシステムエージェント(DSA)と呼ばれる互いに協同動作する応用プロセスの集合から構成され、各DSAは、DIBを部分的に格納、管理する。

2.2 情報の構造

ディレクトリが扱うオブジェクトは名前(識別名)により明確に識別される。

ディレクトリでは、各オブジェクトの名前管理(明確に識別できる名前を与えること)のために、また複数のDSAにまたがってDIBを容易に分散管理できるようにするために、DIBはディレクトリ情報木(DIT)と呼ばれる木構造で論理的に表現される(図2)。DIT木構造における木の節はエントリを、また木の枝はエントリ間の従属関係を表す。

エントリにはオブジェクトエントリと別名エントリがある。オブジェクトエントリはひとつのオブジェクトを表し、それに関する属性情報(例えば、人名、住所、電話番号、プレゼンテーション・アドレス、O/Rアドレスなど)を持つ。オブジェクトによっては、オブジェクトエントリの他にオブジェクトに別名をあたえる別名エントリも存在する。

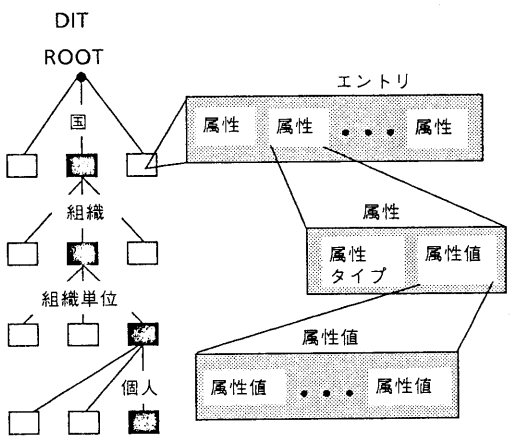


図2 ディレクトリ情報木

また複数のDSAが共同動作して分散処理を行う場合、DIBのどの部分がどのDSAに収容されているかに関する知識(Knowledge Reference)が必要で、DITと同様に木構造(知識木:KT)で表現される。

2.3 ディレクトリ操作

DITに対しては、既知の識別名からオブジェクトエントリの情報を読出す(Read)、特定のオブジェクトの直接下位のオブジェクトの識別名をリストする(List)、特定の条件に合致するエントリを検索し、その属性情報を得る(Search)、エントリの様々な情報を更新する(Modify Entry)など9種類の操作(Operation)が規定されている。

2.4 分散処理

DUAからの操作要求が、ひとつのDSA内で処理が完結しない場合、複数のDSA間で分散処理を行う。分散処理の形態には、ア)他のひとつのDSAに操作を依頼する(チェイン)、イ)複数のDSAに同一の操作を依頼する(マルチキャスト)、ウ)どのDSAが処理可能かのヒントを返す(リフェラル)がある。

複数のエントリが対象となる操作(例えばListやSearch)が単独のDSAで処理が完結しない場合、そのDSAは適切な操作を新たに生成して他のDSAに操作を依頼する。DSAはそれらの結果と自DSAの結果

を合成(結果合成)して、操作の要求元に返送する。

2.5 プロトコル

ディレクトリ・システムでは、DUAがDSAにアクセスするためのディレクトリ・アクセス・プロトコル(DAP)とDSAが分散処理を行うためにDSA間で授受するディレクトリ・システム・プロトコル(DSP)を、応用層のプロトコルとして規定している(図3)。またそれらを実行する応用サービス要素として、それぞれディレクトリ・アクセスの3つのサービス要素(readASE、searchASEおよびmodifyASE：以下DASEと呼ぶ)および分散処理のための3つのサービス要素(chainedReadASE、chainedSearchASEおよびchainedModifyASE：以下DSSEと呼ぶ)を規定している。

DASEとDSSEは、同じ応用層にあるアソシエーション制御サービス要素(ACSE)や遠隔操作サービス要素(ROSE)を用いてディレクトリ・プロトコル(DAPとDSP)を実行する。

DUA(またはDSA)は、操作要求に先立ち、ACSEを用いてアソシエーションを設定し、操作が終了し、不必要になったときに解放する。アソシエーションの設定の際には、必要な操作の種類に対応する通信環境(応用コンテクスト)が設定される。現在DAPには、1種類(DASE、aCSEおよびrOSEの組合せ)の応用コンテクストが、またDSPにも応用コンテクスト1種類(DSSE、aCSEおよびrOSEの組合せ)が定義されている。アソシエーション設定/解放以外のディレクトリ操作はROSEの操作を用いて転送される。

3. ディレクトリ・システム実装のための基本検討

OSIディレクトリ・システムを実装する場合に重要となる以下の項目について検討を行った。

(1) 応用層構造(ALS)を考慮したソフトウェア構成

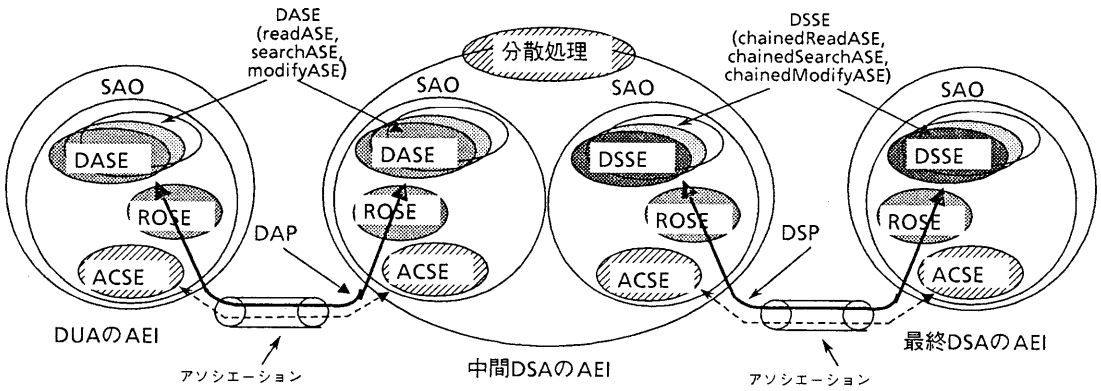
OSI応用層のソフトウェアを実装する場合、以下の理由により、各種のASE(例えば、ACSE、ROSE、RTSE、CCR SEなど共通に使用されるASEや、FTAM、MHS、ディレクトリ、RDA(遠隔データベースアクセス)、VT(仮想端末)などの応用業務特有のASE)を独立のプロセスとしたマルチ・プロセスの構成が有効である[3]

- OSIの応用層構造(ALS)では、多様な応用を実現可能とするため、応用コンテクストにより、ASEの組合せやそれらの使用法などの通信環境を自由に設定できるようにしている。ひとつのシステム内で複数の異なる応用を同時に提供するためには、ASEの自由な組合せを実現できるようにする必要がある。
- 実行時のプログラム・モジュールの共有化を図り、全体のプロセス・サイズをコンパクトにすることにより、オーバレイやスワッピングなどによる効率低下を回避する。

またここでは、プロセス間はキュー・インタフェースで結合し、メッセージとして各ASEで規定されるサービス・プリミティブをフォーマット化したものを用いる。サービス・プリミティブの利用者データは、抽象構文の内部表現として、ASN.1で符号化された転送構文を使用する。

筆者らは、既にACSEおよびプレゼンテーション層以下のプロトコルを実装しており[3]、今回ディレクトリ・システムに必要なROSEおよびディレクトリ・システムのASE(DAPのためのDASEおよびDSPのためのDSSE)のソフトウェアを新たに追加作成する。この際上記の手法に従って、ROSEとディレクトリ・システムのASE(DASEおよびDSSE)を別のプロセスとして構成することとする。

なお今回ALSのSACF(単一アソシエーション制御機能：応用コンテクストに従い、ひとつのアソシエーション上の各ASE機能の調整を行う)の機能は、ASEに対応するソフトウェアの上下関係により実現することとする。



DAP：ディレクトリアクセスプロトコル　　DASE：ディレクトリアクセスサービス要素
DSP：ディレクトリシステムプロトコル　　DSSE：ディレクトリシステムサービス要素
AEI：応用エンティティ・インボケーション　　SAO：単一アソシエーション・オブジェクト

図3　ディレクトリシステムの応用サービス要素(ASE)とプロトコル

(2)DIBの実現方法

DSAは、ディレクトリ情報(DIBの一部やKT)を蓄積、管理する一種のデータベース機能を持つ。DSAにおけるデータベース機能の実現方法として、次の2つの方法が考えられる。

ア)OSIディレクトリ・スキーマに基づく専用のデータベース管理システム(DBMS)を作成する。

イ)汎用のDBMSを使用して、OSIディレクトリ・スキーマをそれにマッピングする。

ア)の場合、DIBの情報構造やそれに対する操作を効率よく実現することが可能となる反面、比較的大規模なDIBを扱うシステムでは、障害回復などデータベースの保守性をも考慮する必要があり、この場合その開発コストが大きい。

イ)の場合、DIBの情報構造や操作と汎用DBMSのそれらとのギャップは多少あるものの、データベースの保守性のよい汎用DBMSを用いることにより、比較的容易に大規模なDIBを構築できる。

今回の実装では、比較的大規模なDIBにも対応することを考慮し、イ)の方法をとることとする。汎用DBMSとしては、リレーショナル(RDB)型とネットワーク(CODASYLなど)型、オブジェクト指向型などが考えられる。このうち、ネットワーク型やオブジェクト指向型のものは、RDB型に比べて、DIBの情報構造などとの対応に有利であると考えられるが、以下の理由によりRDB型のものを採用することにする。

- 普及度が高く、しかもSQLなど標準化(ISO標準)されたユーザインタフェースが提供されているため、作成するディレクトリ・システムのソフトウェアの移植性が高まる。(たとえSQLでなくとも、一般にRDB型のデータベースでは、SQLに対応の付く言語が提供されている。)
- ハードウェア化し易く、処理の高速化が期待できる。既に専用のバックエンド・システムとして商用化されているものも、少なくない。

4. OSIディレクトリ・システムの実装

上記の検討結果に基づいて、筆者らはOSIディレクトリ・システムを実装した。

4.1 実装の前提条件

- VAX8700およびμVAX(OS:VMS)上に実装する。
- 既にに実装済みのプレゼンテーション層以下のプロトコル、およびACSEのプロトコル・プログラム[3]に加え、新たにROSとディレクトリのプロトコル・プログラムを作成し、OSIに完全に準拠した形で実現する。
- DSAにおけるディレクトリ情報の蓄積管理機能を、汎用リレーショナル型DBMS(ORACLE)パッケージを使用し、ディレクトリのDIBやKT(知識情報木)とのデータ構造の対応、およびディレクトリ操作とORACLEのDML(データ操作言語)との対応をとることで実現する。
- 他の計算機への移植を考慮し、プログラムはC言語で記述する。

4.2 実装範囲

- ディレクトリ標準[1]に準拠し、表1に示す機能を実現する。
- DUA(遠隔およびDSAローカル)のユーザインタフェースとして、既に実装したMHSやFTAMなどのOSIプロトコル・プログラムからの利用を考慮した関数呼出しレベルと端末利用者(人間)のためのマンマシンインタフェースをサポートする。
- DSAは、同時に複数のDUAおよび他のDSAからの操作を処理できるようにする。
- 標準では、データの更新にともなう一貫性の保証について明確な規定がないため、今回更新系の操作については、DUAからのアクセス・ポイントとなったDSA内に閉じた範囲とし、また別名エントリとそれが指すオブジェクト・エントリとの一貫性は、更新者の責任とする。

表1 ソフトウェアの主な仕様

分散処理	チェイン/マルチキャスト、リフェラル 操作分割/結果合成
DIB	ディレクトリ・システムで規定されるすべての要素(オブジェクト・タイプ、別名および属性タイプなど)(ただし、追加定義可)
ディレクトリサービス(操作)	・読出 Read、Compare、Abandon ・探索 List、Search ・変更 AddEntry、ModifyEntry RemoveEntry、ModifyRDN
セキュリティ	パスワードによる簡易認証
プロトコル	・DAP(ディレクトリ・アクセス・プロトコル) ・DSP(ディレクトリ・システム・プロトコル) ・ROSプロトコル (プレゼンテーション・マッピング)

4.3.ソフトウェア構成

ソフトウェアは、図4に示すように、ROSEプロセス、利用者プロセス、DIBプロセスおよびディレクトリ・プロセス群から構成される。各プロセス間のデータの授受は、上り/下りのキューを通して行われる。今回、キュー管理には、VMSのメイルボックス機能を用いた。なおアソシエーションの確立/解放を伴う場合以外(例えば、ROSのRO-INVOKEが直接P-DATAにマッピングされるとき)は、図5中②に示すように、ACSEを介さずにROSEとプレゼンテーションがデータ授受できるようにし、不必要なプロセス間インタラクションを行わないようにした。

4.4 ROSプロセス

ROSEプロセスは、ROSの標準で規定される汎用の対話操作機能を提供する。3(1)の検討結果に従い、ディレクトリ以外のROSを使用する応用(88年版MHS、RDAなど)も共存できるように汎用化し、独立のプロセスとした。

ここでは、すべての操作クラスとアソシエーション・クラスをサポートした。(ただし、ディレクトリでは、操作クラス2(非同期、結果/エラー報告

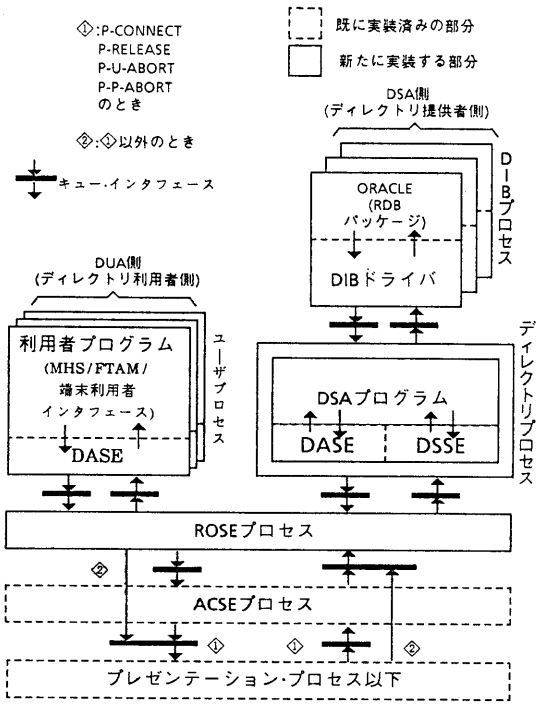


図4 ソフトウェア・プロセス構成

有り)、アソシエーション・クラス1(起動側が操作発行)を使用する)

プレゼンテーション/ROSEプロセス間、ROSE/ディレクトリ・プロセスまたはユーザ・プロセス間のインタフェースでは、それぞれプレゼンテーション、ROSの標準で規定されるサービス・プリミティブをフォーマット化(ASN.1の基本符合化規則に従って符号化)したものを使用した。ただしROSEでは、アソシエーション制御のサービス・プリミティブが規定されていないが、状態遷移はアソシエーションの状態に依存するため、ACSEのプリミティブをROSEを通し、ROSEでアソシエーションの状態を把握できるようにした。

4.5 ユーザ・プロセス

ユーザ・プロセスはDUAの機能を提供し、利用者プログラム(MHS、FTAMあるいは端末利用者対話インタフェースなどの各プログラム)とDASEプログラムから構成される。DASEはDAPのプロトコル処理を行い、利用者プログラムに対し、関数呼出しの形でディレクトリにアクセスするインタフェースを提供する。今回利用者プログラムとして、端末利用者がメニュー方式等により、簡易にディレクトリにアクセスできる対話型の端末利用者インタフェース・プログラムを作成した。

4.6 ディレクトリプロセスとDIBプロセス

ディレクトリプロセスと(複数の)DIBプロセスにより、DSAの機能を遂行する。DSAの機能を複数のプロセスから構成したのは、つぎの理由による。

- DIBを実現するために、ORACLEのRDBパッケージを使用したが、ORACLEでは、ひとつのプロセスから非同期に複数のDML(データ操作言語)を実行要求できない。このため複数のORACLEユーザ・プロセス(ここではDIBプロセス)を起動し、同時に複数ディレクトリ・ユーザからの操作要求を処理できるようにした。

ディレクトリ・プロセスとDIBプロセスは、プロセス間通信の回数を減らしオーバヘッドを極力減らすことを考慮して、以下の機能分担およびプロセス間インタフェースを決定した。

(1) DIBプロセスの機能

ア)名前解読、イ)操作実行などDBMSに依存する処理を行う。

(2) ディレクトリ・プロセスの機能

ア)DASE/DSSE処理、イ)アソシエーション管理、ウ)操作管理、エ)操作実行管理(DIBプロセスの割り付け)、オ)分散処理制御などDBMSに依存しない処理を行う。

(3)DIBドライバ/ディレクトリ・プロセス間インタフェース

ディレクトリ操作(例えばRead、Search操作など)やそれらに対する応答に対応するメッセージ(ASN.1で符号化)を授受する。なおDIBプロセスからの応答は、自DSA内で蓄積されているオブジェクト情報やエラー情報のみでなく、他のDSAへの分散処理情報(チェイン/マルチキャストすべき操作やアクセスポイントなど)などディレクトリ・プロセスが分散処理するために必要な情報を一括して返すようにする。

このような機能分担/インタフェースを設定することにより、ア)DBMSのデータ構造やDML(データ操作言語)に依存せずに、分散処理の制御が行える、イ)各操作に対して、1回の要求/応答で自DSA側のDIB操作が完了するため、複数ユーザ(操作)の実行制御が容易となる。

4.7 下位サービスのプロセス

既に実装済みのACSEおよびプレゼンテーション以下のプロトコル・ソフトウェア[4]を使用した。ただし、今回これらのソフトウェアに以下の機能拡充を行った。

- セションは、不定長ユーザデータをサポートするバージョン2の機能を扱えるようにした。
- プレゼンテーションは、ROSを含むディレクトリ・プロトコルのPDUの転送構文解析のモジュールを追加作成した。ASN.1マクロについては、展開した形で構文を記述した。
- プロセス間のキュー・インタフェースであるVMSメイルボックスでは、1メッセージ長に制限(64Kバイト)がある。今回ListやSearch操作の結果が多量になる場合を想定し、セション以上のプロセスでは、複数のメイルボックス・メッセージでひとつのサービス・プリミティブを構成できるようにした。

5 ディレクトリ·プロセスとDIBプロセスの処理

5.1 ディレクトリ·プロセスの処理(図5)

(a)DASE / DSSE処理

DUAからのアクセス·プロトコル(DAP)および分散処理のチェイン操作のためのプロトコル(DSP)のためのプロトコル処理(受信 / 送信)を行う。

(b)アソシエーション管理

応用アソシエーションを管理するため、図6のアソシエーション管理テーブル(ASMT)を使用する。アソシエーションは、ア)要求時に確立、イ)常時確立(接続頻度の多いDSAの場合)のモードがあり、イ)の場合、アソシエーションの異常終了時に一定時間後に再確立するようにした。

(c)操作管理

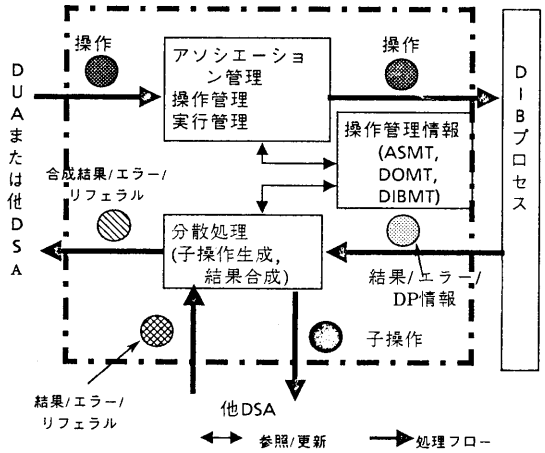
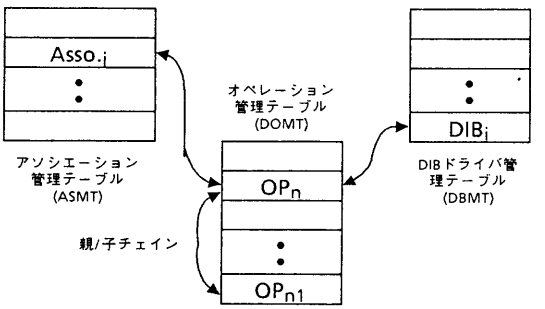


図5 ディレクトリ·プロセスの処理



アソシエーション管理テーブル(ASMT)エントリの項目

AEP番号、DUA(DSA)名、PSAPアドレス、応用コンテクスト、プレゼンテーションコンテクスト識別、状態(未確立、確立待ち、確立、障害中)、モード(常時確立、要求時確立)など

オペレーション管理テーブル(DOMT)エントリの項目

操作種別(Read, Search, Listなど)、AEP番号、インボークID、処理状態(バインド中、正常終了、実行結果待ち、実行待ち、強制終了待ち、強制終了、タイムアウト処理待ち)、アーギュメント、結果、エラー、リファレンス、親子種別(親操作、子操作)、分散処理モード(チェイン、マルチキャスト(結果合成無し)、マルチキャスト(結果合成有り))など

DIBドライバ管理テーブル(DBMT)エントリの項目

操作、状態(空き、応答待ち、強制終了待ち)など

図6 ディレクトリ·プロセスの操作管理情報

同時に複数操作を実行管理するため、図6のディレクトリ操作管理テーブル(DOMT)を使用し、これには操作内容、処理状態等の情報が含まれる。各操作ごとに一つのDOMTを割当てる。操作が発生すると、DOMTを新たに生成し、操作が終了すると除去する。他DSAとの分散処理のために新たに生成する操作にもDOMTを割当て、元の操作(親操作)のDOMTに子操作としてリンクする。

(d)実行管理

ディレクトリ·プロセスは、DOMTにある実行待ちの操作を、(もしあれば)空きのDIBプロセスに割当て、DIBプロセスに操作を発行する。すべてのDIBプロセスが使用中である場合、いづれかのDIBプロセスが空くまで操作を待たせる。DIBプロセスの空き状態やDOMTとの対応は、図6のDIB管理テーブル(DIBMT)を用いて管理する。

DUAや他DSAからAbandon操作を受理すると、ア)元の操作が実行待ちのときDOMTを除去する、イ)DIBプロセスで実行中のときDIBプロセスにAbandon操作を発行する、ウ)他DSAに子操作を発行済みのときそれらのDSAにAbandon操作を発行し、(もしあれば)既に得た結果を廃棄する。

(e)分散処理制御

DIBプロセスからの応答に、分散処理情報(DP情報:自DSAのみでは処理が完結せず、他のDSAとの分散処理が必要であることを示す)が含まれる場合、その情報(処理モード(チェイン、マルチキャスト)、アクセスポイントおよび操作内容など)から新たな操作(子操作)を生成し、指示されたアクセス·ポイントのDSAに対して操作を発行する。ただし、元の操作でチェインが禁止されている場合、また他のDSAにアクセスできない場合には、アクセス·ポイントなどDP情報の一部を、リフェラルとして操作の要求元(DUAまたはDSA)に返送する。

(f)結果合成

DIBドライバからの応答を操作の要求元(DUAまたはDSA)に返送するため、ディレクトリで規定されるデータ構造に組み立てる。

ListやSearch操作で、複数のDSAが分散処理する場合、他DSAからの結果および自DSAからの部分結果をひとつに合成して、要求元に返送する。一部のDSAよりエラーが返送された場合でも、部分結果を優先して、要求元に返送することとした。

(g)認証

DUA / DSAとのアソシエーション確立時に、簡易認証を行うため、相手のパスワードをCompare操作を使用してDIBプロセスに問合せる。

(h)サービス制御

操作の共通アーギュメントに、タイムリミットやサイズリミットなどが指定されている場合、以下の処理を行う。

●タイムリミット

操作を受理するとタイマをセットし、タイムアウト時には、DIBドライバおよび(もしあれば)チェイニングを行ったDSAにAbandon操作を発行し、処理を強制的に終了させ、操作の要求元にエラー("timeLimitExceeded")を返送する。(ただし、List

操作やSearch操作の場合、エラーではなくそれまでに得られた部分結果を返送する。)

●サイズリミット

結果が指定されたサイズを超える場合、指定されたサイズで結果を返送し、サイズリミットを超えたことを通知する。

●優先度

指定された優先度(low, medium, high)に従い、必要に応じて実行待ちのDOMTの順序を入れかえる。

5.2 DIBプロセスの処理(図7)[5][6]

(a)DIB情報 / KT情報

KT情報のために、4つのテーブル(NCP(DITの部分木とDSAとの対応)、SUPR(上位DSA)、REF(自DSA内の部分木の知識および下位リファレンス)、NSSR(不特定下位リファレンス)を使用する。

DIB情報として、エントリ·テーブル等を各オブジェクト·クラス毎に設けた。これらのテーブルは、ORACLEに格納する。(ただし、NCPやSUPRなど動的に変化しないテーブルは、効率向上のためメモリ上に常駐させることとした。)

(b)ディレクトリ·スキーマ情報

新たなオブジェクト·クラスの追加等に柔軟に対応するため、ディレクトリ·スキーマ情報をプログラム構造とは独立に管理した。この情報には、ORACLEにあるDIBのエントリ·テーブルなどとの対応情報(属性が、どのテーブルのどのカラムに対応するかなど)が含まれる。

(c)名前解読処理

ディレクトリ·プロセスから、操作(Abandon操作を除く)を受理すると、KT情報のテーブルを使用して、対象オブジェクトを持つDSAを決定する。自DSAに存在する場合には、(d)の操作実行を行う。他DSAに存在する場合には、DP情報をディレクトリ·プロセスに返す。

またトレース情報アーギュメントを用いてDSAにまたがる操作のループ検出を行う。

(d)操作実行

まずディレクトリ·スキーマ情報を参照して、アクセスするエントリ·テーブルとそのカラムを決定する。それをもとに適切なSQL文を生成し、ORACLEに発行することにより、対象となるオブジェクトの検索 / 更新を行なう。複数のオブジェクトが操作対象となるListやSearch操作の場合には、KT情報のREFテーブルを用い、エントリ間の上 / 下関係を得て、検索を行ない、結果をディレクトリ·プロセスに返す。この際、KT情報よりDIT部分木が他のDSAに存在することが判明した場合、エントリ·テーブルからの部分結果と未処理の部分木に対するDP情報を一括して返す。

(e)強制終了

操作(検索系)の処理中に、ディレクトリ·プロセスからAbandon操作を受けると、処理を終了(ORACLEへアクセス中は、その応答を待って処理を終了する)し、確認をディレクトリ·プロセスに返す。ただし、Abandon操作が5.1 (h)のタイムアウトなどの理由によりディレクトリ·プロセスが生成したものであり、ListおよびSearch操作を実行している場合には、それまでに得た部分結果も返す。

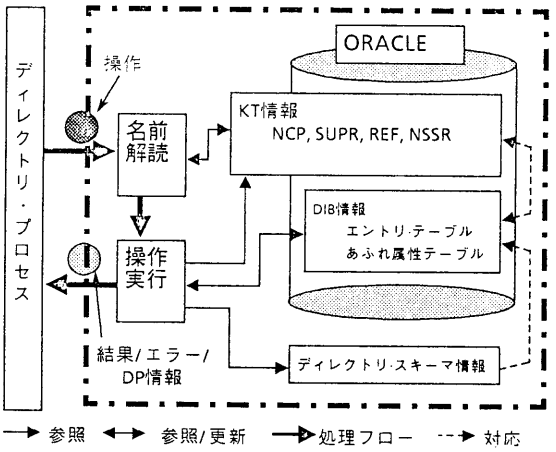


図7 DIBプロセスの処理

6. 評価と考察

6.1 実証実験

実装したOSIディレクトリ·システムのソフトウェアの処理効率を評価するために以下の2つの形態で実験を行った。

(1)ネットワーク層の内部折返し

VAX8700上にトランスポート層以上のプロセス群を1つのDUAと2つのDSA毎に別に用意し、ネットワーク層をVAX8700上で折り返した。この際、図8に示すDIBの構造を使用し、DUAからDSA_1に以下の操作を発行した。

- DSAの分散処理無し(図8中、Read-1, Search-1)
- DSAの分散処理有り(図8中、Read-2, Search-2)

(2)X.25網経由

μVAX(DUA側)およびVAX8700(DSA側)上に、それぞれトランスポート層以上のプロセス群を用意し、ネットワーク層にX.25網(9.6Kbps)を使用した。この際(1)と同じDIBを使用し、DUAからDSA_1に分散処理無し(図8中、Read-1, Search-1)の操作を発行した。

実験結果を表2に示す。操作の応答速度は、転送されるデータ量(オブジェクトの属性情報のデータ量)にかなり依存する。そこで、今回MHS / FTAMなど

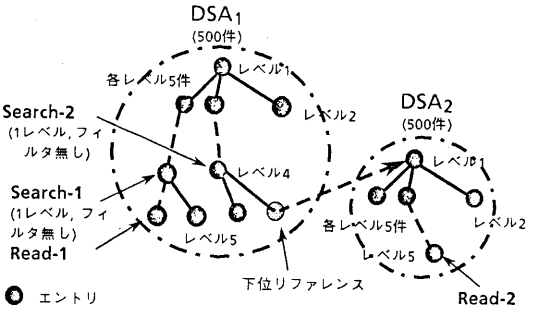


図8 実証実験のデータ構造 / 操作

表2 アクセス時間測定結果

実験形態 / 操作	ローカルDIB プロセスの 処理時間 (単位 秒)	内部折返し (単位 秒) 注1)	X.25網経由 (単位 秒) 注1), 注2)
アソシエーション確立	———	0.7[注3]	4[注3]
Read-1[注5]	0.5	1.0	2.5
Search-1[注6]	1.9	2.5	4.1
Read-2[注5]	———	3.0[注4]	———
Search-2[注6]		4.8[注4]	

注1) プレゼンテーション・カーネル、トランスポート・クラス0、セションBCS(全二重)
注2) X.25 9.6Kbps
注3) 認証のためのCompare操作の時間(0.5秒)を含む
注4) DSA間のアソシエーション確立時間含む
注5) 1属性抽出、注6) 1レベル,フィルタ無し,1属性抽出

のOSI通信システムからのRead要求では、オブジェクトのO/Rアドレスやプレゼンテーション・アドレスなど、たかだか1属性の情報しか転送されないことを考慮し、操作で抽出する属性を1つとして測定を行った。

6.2 考察

(1)効率について

- 実証実験の効率評価結果より、応答時間については、十分実用できる見通しを得た。なおディレクトリでは、DIBが木構造(DIT)で表現されるため、探索操作で探索範囲を大きく指定すると、かなりの応答時間を要する。このため、探索ではあらかじめ探索範囲を十分絞りこむなど、使用上の配慮が必要である。

(2)ソフトウェア構成について

- これまでに提案したOSIの上位層のプロセス構成方法[3]に従ったマルチ・プロセス構成とすることで、効率よくディレクトリ・システムを実現できた。またこれにより、現在ISOなどで検討されている、分散処理機能も考慮したOSIの応用層構造(ALS)に対しても、スムーズに移行できる。

(3)機能について

- RDBにおけるカラム(列)長制限に起因してDIBのオブジェクト属性値に長さ制限があるなど、一部実装上の制約があるが、基本的にOSIディレクトリシステムの全機能を実現できた。
- 汎用RDBパッケージを使用して、効率よくDSAのDIB機能を実現できた。この際ディレクトリ・スキーマ情報(DIT構造上の制約、オブジェクト・クラス、属性タイプ、属性シンタックス)をプログラムとは独立に管理することにより、今後各種の応用で規定されるさまざまなオブジェクトの情報に柔軟に対応できるようにした。
- ローカルなDSA機能(DIBプロセス)と分散処理制御機能(ディレクトリ・プロセス)を分離したことで、DSAを効率よく実現できた。ただし、今回自DSAでの処理が完了してから、他DSAへ操作要求を発行する形態をとっており、今後自DSAの処理をも含めた並行処理について検討が必要である。
- 今回、端末利用者(人)にメニュー方式による対話インタフェースを提供し、利用者の利便を図ったが、さらにDIBの複雑な構造を極力意識させないようにする検索支援機能が必要である[7]。
- アクセス制御や強認証などのセキュリティ機能の実装については、今後の課題であるが、ディレクトリシステムの運用上極めて重要であり、ディレクトリの処理効率を低下させない、効率よい処理メカニズムの検討が必要である。
- 現行の標準/勧告では、他DSAの知識となるクロスリファレンスやデータのコピーを持つキャシュなど、処理の高速化の手法が検討されているが、知識の獲得やコピーの獲得については、実装上の問題としている。頻繁にアクセスの対象となるオブジェクトや部分木については、分散処理結果で得られた知識やデータのコピーを動的に保存することで、容易に実現できると考えられる。

7. おわりに

筆者らは、OSIディレクトリシステムのソフトウェアを実装した。本ソフトウェアは、ア)OSIに完全に準拠し、今後の多様化する応用を実現するための応用層構造(ALS)に柔軟に対応できる汎用的なソフトウェア構成、イ)汎用のリレーショナル型DBMSパッケージを用い、多様なディレクトリ・スキーマに対応できる柔軟なDIB(ディレクトリ情報ベース)機能、ウ)ディレクトリ操作のための分散処理機能、を効率よく実現している。

本稿では、実装のための基本的事項、特に汎用的なOSIのソフトウェア構成方法、DIBの実現方法について検討し、その結果に基づいて実装したソフトウェアの詳細について述べた。さらに実証実験を通して、機能、性能等について実用性を実証した。

今後は、アクセス制御等セキュリティ機能の拡充やMHSおよびFTAMプログラムとの接続試験、さらに他のOSIディレクトリ・システムとの相互接続試験等を通した実証実験を行っていく予定である。最後に日頃御指導頂くKDD上福岡研究所 小野所長、浦野次長に感謝します。

参考文献

[1] ISO 9594-1~8 (ディレクトリ・システム標準)
[2] ISO 8822, 8823, 8824, 8825, 8649, 8650, 9072-1~2, DIS 9545 (プレゼンテーション、ASN.1、ACSE、ROSおよび応用層構造(ALS)標準)
[3] 小花、加藤、鈴木、「OSI FTAM、ACSE、プレゼンテーション・プロトコルの実装」情処学会マルチメディア通信と分散処理研究会資料33-6、1987.5.
[4] 小花、西山「OSIディレクトリシステムの実装(1) - 基本設計-」36回情処全大、1988
[5] 西山、小花「OSIディレクトリシステムの実装(2) - 汎用RDBパッケージを用いたDSA機能の実現」36回情処全大、1988
[6] 西山、小花「リレーショナル型データベースを用いたOSIディレクトリ情報ベース(DIB)の実現と評価」情処マルチメディア通信と分散処理研究会, 42-12、1989.5
[7] 西山、小花「ディレクトリの端末利用者インタフェース」信学全大(春季)、1989