マルチメディア通信と分散処理　34－7
（１９８７・７・２４）

ＯＺ：対象指向開放型分散システムアーキテクチャ

－－－　アーキテクチャとその実現方法　－－－

塚本享治　　棟上昭男
（電子技術総合研究所）

本報告は、対象指向開放型分散システムＯＺ(Object Oriented Open Distributed System Architecture)のアーキテクチャと実現方法について述べている。ＯＺは３レベルから構成される。第１レベルのネットワークは論理リンク制御タイプ３をベースにした確認つきコネクションレス型プロフィルのＬＡＮ、第２レベルの分散カーネルは仮想計算機を実現するためのオブジェクト指向型のドメイン、第３レベルのプログラミング言語はクラスとモニタから成る分散型の対象指向言語である。

OZ: Object Oriented Open Distributed System Architecture

--- Arcitecture and its realization ---

Michiharu TSUKAMOTO　　Akio TOJO
(Electrotechnical Laboratory)

(英術住所)

This paper describes the architecture of the system ＯＺ (Object Oriented Open Distributed System Architecture) and the method of its realization. The ＯＺ consists of three levels. The first level is a local area network whose profile is acknowledged and connectionless. The second level is object oriented domain for communicating virtual computers. And the last level is a distributed object oriented programming language .

１．　まえがき

強力な計算機の低廉化と計算機を接続するネットワークの標準化によって、新たな分散システムの構築が可能になってきた。ここで、分散システムとは、データベースシステムや銀行システムなどの特定業務向けの分散型応用システムを意味するものではなく、計算機の分散を意識させない使いやすいインタフェースを提供するオペレイティングシステム（ＯＳ）を意味する[1)]。

このようなＯＳは、ネットワークＯＳと分散ＯＳに分けられる。ネットワークＯＳは既存のＯＳに通信機能を付加したものであり、資源の管理や実行の制御はもとのＯＳに依存するため、利用者は色々な点でこのことを意識する必要がある。これに対し、分散ＯＳでは資源の管理や実行の制御をシステム全体を単位として行うため、利用者は分散であることを意識しないで共通のインタフェースでシステムが利用できる。ネットワークＯＳは既存のＯＳを逐次改良することで実現できるところから、商用のシステムで広く採用されているが、分散ＯＳは方式から再検討する必要があるため、幾つか試作されてはいるが今後の研究に待つところが大きい。

ところで、世の分散ＯＳでは開放型システム間相互接続（ＯＳＩ）のようなコネクション型の階層構成を持つ汎用ネットワークは効率が悪いという理由から、特殊なプロトコルを実現することが多かった[1),2)]。筆者らは、従来のＯＳＩプロフィル（各層のプロトコルの選択）は分散ＯＳが対象とするＬＡＮに適していないと考え、ＬＡＮの特徴を活かした新たなＯＳＩプロフィルのネットワークを開発して、その上に対象指向型の分散ＯＳを実装するプロジェクトを開始した。このプロジェクトはＯＺ(Object Oriented Open Distributed System Architecture)と名づけられている。以下にそのアーキテクチャと実現方法の概要について述べる。

２．　ＯＺの目標

２．１　ネットワーク

ＯＳＩにはコネクション型とコネクションレス型の種類のモデルが存在する。ＯＳＩを構成する各層のコネクション型プロトコルがほぼ標準化されたというので、それらを組み合わせて実際のネットワークを実現するための実装仕様（プロフィルともいう）の開発が世界の各所で進められている。これまでに開発されたコネクション型プロフィルでは、広域ネットワーク（ＷＡＮ）とローカルエリアネットワーク（ＬＡＮ）は、下位層においてそれぞれの事情に即してコネクションと高品質伝送を実現し、上位層には同一のコネクション型応用サービスを実現するようになっている。ＯＳＩのコネクション型プロフィルはＴＣＰ／ＩＰの流れに沿うものであり、ＬＡＮとＷＡＮに共通なネットワークを実現するという長所を持っているが、高速高品質伝送というＬＡＮの特徴を活かしていない。そこで、新しい分散ＯＳやＦＡシステム（ＭＡＰでも実時間応答に必要な部分は非ＯＳＩである[4)]）などの実現を志す人達は高速性を望むために第２層あるいは第３層の上に直接個別な応用プロトコルを実現することが多い。このような人達の要求に答えるために、ＩＳＯでは現在各層のコネクションレス型プロトコルの標準化を進めている。ＯＺのネットワークでは、ＯＳＩに準拠する高速高機能なコネクションレス型プロフィルのＬＡＮを実現し、標準化に役立てることを目標としている。

なお、このＯＺの実装はＯＳＩネットワークの実現を推進する通産省工業技術院の大型プロジェクト『電子計算機相互運用データベースシステム（通称、インオペ大プロ）[3)]』の一貫として行われているものであるが、メーカ各社のネットワークがデータベース実現のためのコネクション型プロフィルであるのに対し、ＯＺのネットワークは分散ＯＳ実現のためのコネクションレス型プロフィルである。

２．２　分散ＯＳ

一般にネットワークで接続されたシステムを効率良く使えるようにするレベルとしては、ハードウエア、カーネル、応用プログラムの３種が考えられる。ネットワークＯＳはネットワークというハードウエアレベルだけで統合するのに対し、分散ＯＳは資源の管理と実行の制御をカーネルレベルで統合する。そのため、分散ＯＳの多くは応用プログラムからは単一計算機用ＯＳと類似に見える。ところが、分散ＯＳだけではどのように分散処理のプログラミングを行ったらよいのかわからない。そこで、ＯＺでは分散ＯＳ（厳密には分散カーネル）の使い方をプログラミング言語の形で提供することを目差している。もちろん、ネットワークレベルや分散カーネルレベルで使用することも可能である。

なお、ＯＺの分散ＯＳは筆者の一人がロボット用分散ＯＳとして研究してきたもの[13)]を前述のネットワークに合わせて一般化して実装するものであり、ＯＡからＦＡまでの広い分野を狙っている。

３．　システム構成

ネットワークと分散ＯＳを検証するためのＯＺシステムのハードウエアとソフトウエアの構成の概要につ

いて述べる。

３．１　ハードウエア構成

ＯＺのシステム構成は図１のようなものになる予定である。ネットワークは１つの幹線セグメントと２つの支線セグメントで構成され、幹線セグメントと支線セグメントとはブリッジで接続される。幹線セグメントにはインオペ大プロに参加するメーカがコネクション型プロフィルを実証するために開発するIEEE802.4のブロードバンドトークンバス方式のものを使用し、支線セグメントにはIEEE802.4 のキャリアバンドトークンバス方式のものを使用する。

各セグメントにはバス接続のＬＡＮコントローラボードを介してワークステーションが接続され、それぞれにはＯＺシステムが実装される。特に幹線上の１台のワークステーションにはＯＺシステムとＩＮＴＡＰのコネクション型プロフィル[4)]のシステムの両方を実装し、ＯＺシステムとＩＮＴＡＰシステムを接続するゲートウエイを実現する。

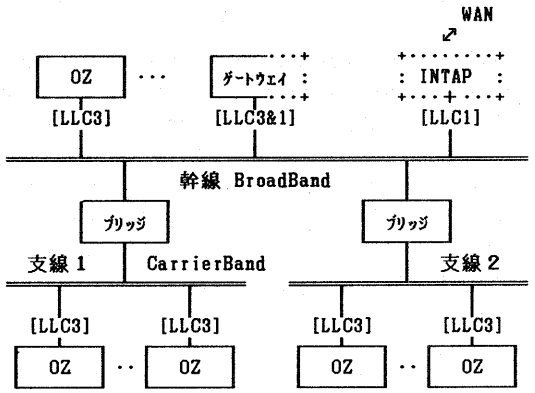

図１　　ＯＺのハードウエア構成

３．２　ソフトウエア構成

ＯＺのソフトウエアはコネクションレス型プロフィルのネットワーク上に２レベルで構成する（図２）。下位のレベルは仮想計算機を格納する実体を実現する分散カーネルであり、上位のレベルは分散カーネルが実現する実体に格納する具体的な仮想計算機を実現する高級言語マシンと各種サーバである。ＯＺでは分散システム上で高度な分散処理の応用ができるように分散型対象指向言語マシンを実現し、この言語をＯＺのユーザインタフェースとする。

分散カーネルは分散型対象指向言語マシンから既存のシステムを利用するためであるとともに、既存のシステムからＯＺシステムが利用できるようにするためのものである。既存のシステムを以後サーバと呼ぶ。ＩＮＴＡＰのシステムとはゲートウエイサーバを作ることによって接続する。

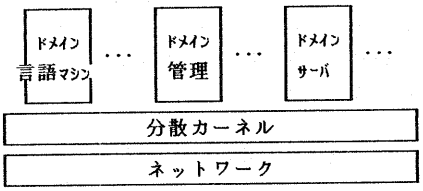

図２　ＯＺのソフトウエア構成

４．　ＬＡＮ

ＯＺシステムに使用するＬＡＮはＩＳＯのコネクションレス型プロトコルを使用するコネクションレス型プロフィルである。

４．１　確認つきコネクションレス型プロフィル

ＯＳＩのＬＡＮ用コネクション型プロフィルはアソシエーションの設定・解放とデータ転送に時間を要する　ため、一般的に分散ＯＳには適さない。ＯＺではユーザごとに使用するワークステーションやサーバにドメインを置く必要があるが、情報交換するドメイン間にアソシエーションを設定するのではアソシエーションの数（階乗）が多くなり、要求のたびにアソシエーションを設定し解放するのでは効率が悪い。

そこで他の分散ＯＳにならって[1)]、ＯＺでもコネクションレス型プロフィルを実現することにした。ＬＡＮの特徴を活かして効率が良く、汎用性があり、しかもＯＺの分散カーネルで使いやすいことに留意して、以下に述べる『確認つきコネクションレス型プロフィル』を実現することにした（表１）。このプロフィル

表１　確認つきコネクションレス型プロフィル

層	基本標準	オプションの選択
MAC	トークンバス(DIS8802/4)	即時／非即時応答使い分け
LLC	タイプ３(NWI SC6N4326)	ack(reply&exchange使用せず)
ネットワーク	ＣＬＮＰ(DIS8473)	インアクティブ
トランスポート	ＣＬＴＰ(IS8072/AD.1,DIS8602)	チェックサムなし
セション	ＣＬＳＰ(PDAD SC21N1972,N1974)	なし
プレゼンテーション	ＣＬＰＰ(WD SC21N1965,N1966)	?
応用	MMS(2ndDP9506/1&2)	アソシエーションレス（変更）
但し、(1) 各層のＰＤＵ長はＭＡＣのフレーム長とする。 (2) 送信局が宛先局と同一セグメント上にあれば、 ＭＡＣの即時応答機能を使用するのを推奨する。		

は、ＭＡＣ副層はトークンバス方式、ＬＬＣ副層は標準化の始まったタイプ３[6)]の確認通知機能、セグメントの接続にブリッジを使用するためネットワーク層はインアクティブ、トランスポート層からプレゼンテー

ション層まではアドレスセレクタだけ、の極めて簡単なものであり、ネットワーク層からプレゼンテーション層までのオーバヘッドはない。

応用層ではＩＳＯで検討中でＭＡＰで採用されることになっているＭＭＳ(Manufacturing Message Specification)[5] を選択し、これをアソシエーションレスで実現することとした。

４．２　論理リンク制御タイプ３[6]

ＬＬＣタイプ３には、確認通知、応答、交換の３機能があるが、応答と交換はＯＳＩのモデルに適合しないるため、本プロフィルでは，LSAPに送信すると受信局から確認通知が戻される確認通知機能だけを採用している。送信エラーや確認通知タイムアウトを検出するとＬＬＣで再送が行われ、再送回数がオーバすれば上位の層にエラーが指示される。ＯＳＩでは送信の確認通知を上位の層に指示する機能はないが、これは実装事項として扱う。

トークンバス方式ではＭＡＣフレームで即時応答か否かが指定できるが、本プロフィルでは宛先局が送信局と同一セグメント上にあればＭＡＣの即時応答機能を使用するが、同一セグメントになければ非即時応答機能を使用する。同一セグメント上では極めて効率が高く、同一セグメント上にないときでもブリッジを介して確認通知が戻されるため信頼性も極めて高い。

４．３　ブリッジ

ＭＡＣ副層の上位にあってセグメントを接続するものがブリッジである。ＩＥＥＥやＭＡＰのブリッジ[7]と異なる次のようなブリッジである。

ＭＡＣフレームの中継にあたっては受信したＭＡＣの優先度を送信の際に保存する。これはＬＬＣタイプ３においては、宛先局では送信局のＭＡＣアドレスと優先度ごとに順序制御を行うためである。

また、ブリッジが宛先局と同じセグメント上にあればＭＡＣの即時応答機能を使用する。これは効率化のためであって、必要な条件ではない。

４．４　MMS

ＭＭＳは遠隔操作（ＲＯＳ）プロトコルのメッセージの形式と意味を定めるものであり、変数、型、セマフォ、ロード、ジャーナル、など多様なメッセージを定めている。ＲＯＳとプロトコルチェックだけを実装すれば良いので効率が良く、しかもメッセージの追加が標準で認められており拡張性もある。

ＯＺではすべての通信にこのＭＭＳを使用する。分散カーネルでは送信量がＰＤＵの最大長に納まらない場合がある。その場合、このプロフィルはフロー制御機能を持たないために、次のようにして大量データを信頼性良く転送する。すなわち、送信要求側はまず相手側に大量データの獲得を通知し、相手側にＲＯＳでデータを逐次獲得させ、最後に送信要求側から転送終了を通知する。

アソシエーションを設定しないため、送信のたびに宛先局を指定しなければならない。ＯＺでは PSAP=SSAP=TSAP （NSAPなし）とする。LSAPの数は６４個と限られており、しかも送信に対する確認通知が戻るまでは同じLSAPを使って同じ局に送信できないという制約がある。そこで、送信のたびに確認通知を待っていない空いたLSAPを選択して使用することとする。

４．５　ＭＡＰとの比較

ＭＡＣ副層以下はＭＡＰと同じものであるが、プロフィルは異っている。ＭＡＰでは主として幹線用にはコネクション型プロフィル（フルＭＡＰ）を使用し、実時間の応答が問題となる支線用にはＭＡＣ副層とＬＬＣ副層はPROWAYに準拠し、ネットワーク層以上を空とするプロフィル（ミニＭＡＰ）を使用している。ミニＭＡＰのＬＬＣ副層では宛先局によってＭＡＣの即時応答と非即時応答を使い分けていないし、ＬＬＣの確認通知と応答をブリッジで中継することもしない。そのため、ミニＭＡＰは単一セグメント上でしか通信できず、ＬＡＮ全体と通信するには両プロフィルを持つＥＰＡをゲートウエイとして使用せざるを得ない[8]。これに対し、確認つきコネクションレス型プロフィルは、ＯＳＩに準拠し、しかもこれだけでＬＡＮ内全域と通信できる。

５．　分散カーネル

ＯＺの分散カーネルは、仮想計算機を格納する実体であるドメインを実現するものである。

５．１　ドメイン

分散システムでは、メモリやプロセッサ、各種サーバなどが様々に分散される。これらの分散の仕方や相互の関連を意識しないで済むように抽象化された実体を導入する。これをドメインと呼ぶ。ドメインはメモリとプロセッサを一体化して抽象化した仮想計算機であり、各種のサーバは用途にあわせて特殊なプログラムを理想的な仮想計算機に入れたものと考える。ドメインは分散の単位であり、その間の情報交換は通信で行われる。

ドメインは以下の構成をとるオブジェクトである。

(1) 識別子:　　識別名（番号）

(2) タイプ: 仮想機械の種類や定義
(3) 特殊性: 機種、移動、保存などの属性
(4) ステータス : 仮想機械の状態
(5) 構成表 : 辞書、表、メモリなどの構成
(6) オブジェクト辞書: 定義／参照するオブジェクト一覧
(7) 転送構文辞書: 扱える転送構文一覧
(8) 通信チャネル: 外部ドメインとの通信
(9) メモリ: 仮想計算機のメモリ

ここで、タイプにはドメインに内蔵する仮想計算機の種類と動作を記述するプログラムを登録する。特殊性とは、タイプはドメインが移動や削除できない理由を登録する。オブジェクト辞書には内部に定義するオブジェクト、他ドメインへの参照、大域オブジェクトの識別子、ＧＣ管理情報などが登録される。転送構文辞書にはドメインが取り扱える転送構文と符号化・復号化ルーチンの一覧を登録する。なお、ＯＺのオブジェクトを内蔵しないサーバではオブジェクト辞書は空である。

５．２　機能と構成

分散カーネルで実現する機能は、通信に関するものと各種サーバへの取り次ぎに関するものがある。

(1) メモリ 管理: ドメインやバッファのためのローカルな メモリ管理
(2) ドメイン管理: ドメインの作成、削除、転送の管理
(3) ドメイン間通信: ドメイン間通信を実現
(4) リソースサービス: システム全体に関連する各種Ｉｄやドメインの割当を行うリソースサーバ への取次ぎ
(5) ダウンロードサービス: ディスクレスワークステーション へシステムロード の要求の発行とダウンロードサーバ への取り次ぎ
(6) ディレクトリサービス: システム全体での一意な名前づけと属性値の登録参照を行うディレクトリサーバ への取り次ぎ
(7) ネットワーク管理サービス: ネットワーク管理の取り次ぎと代行

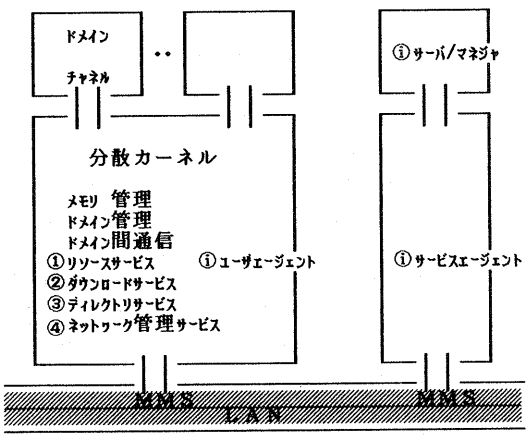

図３　分散カーネルの構成

これらの各種サービスはカーネルは取次／代行をするだけで管理はドメインとして実現されるサーバ／マネジャで行う。これ以外のサービス(ファイル、プリンタ、ゲートウェイ etc)は分散カーネルから直接要求することがないので、ＯＺでは分散カーネルには組み込まない。

５．３　実現方法

分散カーネルの実現にあたっては、ＬＡＮと同様に簡単に実現できることが重要である。ＯＺではＬＡＮ最上位のＭＭＳを活用する。

ドメイン管理およびドメイン間通信のために分散カーネル内にドメイン表を設ける。ドメイン作成時に登録し、ドメイン削除時に削除するほか、ドメイン移動時にフルアドレスの変更、登録または削除を行う。ドメイン識別子から宛先のフルアドレス（意味あるのは、MACアドレス、Lセレクタ、Pセレクタ)を求めたり、発信元フルアドレス（使用するのはPセレクタ)から宛先ドメインを求めるのに使用する。

リソースサービスとダウンロードサービスの実現には、それぞれＭＭＳの変数機能とロード機能を使用する。ディレクトリサービスにはＭＡＰディクショナリを使い、ネットワーク管理サービスにはＭＡＰのそれをＭＭＳで実現する。

ドメインと分散カーネルのインタフェース、およびドメイン間の通信には以下の遠隔操作型サービスを使用する。

(1) d-invoke (dDomId,op,tout,pctx,invId,
par$_1$,･･･,par$_n$) :error
(2) d-reply (dDomId,pctx,invId,rslt,
val$_1$,･･･,val$_m$) :error

d-invoke要求によって宛先ドメインdDomIdの操作opをinvId,par1,･･･,par$_n$ を引数として起動する。toutは分散カーネルが設定するタイムアウト時間であり、invId は d-replyとd-invokeの対応を知るためのものである。d-reply 要求によって処理結果rslt,val$_1$,･･･,val$_m$を戻す。pctxはpar$_1$,･･･,par$_n$,val$_1$,･･･,val$_m$,invId,rsltの転送構文を指示する。rsltはエラーの有無と種類を示すものである。カーネルでエラーを検出すると、それがサービス発行時点ならばerror で指示し、サービス発行後ならばd-invoke起動側にだけd-reply のrsltで指示する。

モデル上、分散カーネルもドメインとし、ドメイン識別子を割り当てる。これにより、ドメイン間、分散カーネル間、ドメインと分散カーネル間に、opが変わるだけで上記のサービスがそのまま使える。

６．　分散型対象指向言語マシン

６．１　分散型対象指向言語のモデル

対象指向型言語[11] ではプログラムの雛形をクラスという単位で記述する。クラスの記述に従って振る舞う実体をインスタンスといい、クラスを伴ったインスタンスをオブジェクトと呼ぶ。メタクラスを導入するとクラスもオブジェクトとなるが、ＯＺでは簡単化のためにメタクラスは採用しない（従って、オブジェクトとインスタンスは同義）。オブジェクトを引数としてオブジェクトのメソッドを呼ぶと、クラスの定義に従って処理され、結果としてオブジェクトが戻される。

従来の実装では引数と結果はポインタであるため、オブジェクトは作成された場所に留まる。そこで、ネットワークで接続されたシステムでは、渡されたオブジェクトにアクセスするために再びネットワークを介した通信が必要となる。分散システム上で対象指向型言語を効率良く稼動させるために、作成された場所にオブジェクトを留めおかないで、オブジェクトのコピーや移動を許すモデルを考える。

オブジェクトは再帰的にオブジェクトで構成されるため、コピーまたは移動の範囲を定めるのはむずかしい。その範囲をオブジェクトの種類で決める。

オブジェクトを共有されない局所的なものと共有される大域的なものとに分ける。それぞれの記述単位をクラス(class) とモニタ(monitor) 、合わせてタイプと呼ぶ。class のインスタンスは局所的かつ受動的であり、それへのアクセスは排他制御しない。これに対し、monitor のインスタンスは大域的かつ能動的であり、呼び出しのたびにプロセスを作り、同一オブジェクト内では同時には１つのプロセスしかアクティブにしない（陽な同期と相互排除にはsignalとwaitを用いる[12] ）。

class の記述は従来の形式を踏襲し、monitor の形式を新たに導入する（図４）。class とmonitor が継承木に混在すると受動的か能動的か判断しにくい。そこで、class の下位には classとmonitor の両方を許すが、 monitorの下位にはmonitor しか許さないことにする。

monitorのmethodは継承木中の呼出しだけに使用し、 classのmethodと同じく引数と結果をポインタで渡す。 monitorのactionを呼出すときは引数と結果は構造を保存してコピーする。大域的オブジェクトはポインタのままとし、局所的オブジェクトはトポロジを保存してコピーする方法である。

オブジェクト間のインタフェースをこのように定義することにより、class のインスタンスはmonitor のインスタンス内だけの局所的なものとなり、monitor のインスタンスだけが大域的となる。class のインスタンスへのアクセスは、それが属するmonitor のインスタンスに限られる。これは厳しい制約だが、同期と相互排除の上では好ましいことである。

```
class   <classId>
 super  <classId>
 var    <varIdList>
 method <methodId> (<parIdList>) <body>
    ........
 end
monitor <monitorId>
 super  <classOrMonitorId>
 var    <varIdList>
 que    <queueIdList>
 method <methodId> (<parIdList>) <body>
    ........
 action <actionId> (<parIdList>) <body>
    ........
 end
```

図４　タイプの記述形式

６．２　オブジェクトとオブジェクト辞書

分散型対象指向言語マシンは前述のモデルを分散されたドメイン上に実現するものである。

オブジェクト辞書はオブジェクトを登録する表であり、オブジェクトは１スロットを占める（図５）。スロットは、スロットをつなぐlink、属性や制御用のflags 、ガーベッジコレクション用のgcG 、オブジェクトを格納するメモリを指すcell、タイプとモニタインスタンス、およびプロセスの場合に識別番号を格納するobjId 、高信頼化処理用のtransId から成る。

スロットは他ドメインのオブジェクトを指すポインタを格納するためにも使用する。このとき、cellの下位２ビットに 11_2 が入り、上位３０ビットにドメイン識別子が入る。

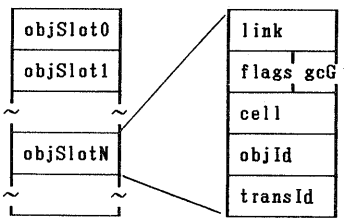

図５　オブジェクト辞書の構成

オブジェクトの主なものを図６に示す。モニタとクラスのタイプコードを格納するモニタタイプ（ＭＴ）とクラスタイプ（ＣＴ）、それらのインスタンスを格納するモニタインスタンス（ＭＩ）とクラスインスタンス（ＣＩ）、およびプロセス（ＰＲ）である。ここで、super は上位タイプ、byteCodeとexecCodeはバイトコードと機械語コードを格納するそのオブジェクト中の場所を示す。entQue , rcvQue , およびrstQueはそれぞれ、モニタの入口における実行待ち、返事待ち、よび実行待ちのプロセスをつなぐキューである。プロセスは特殊であり（実装は別）、linkはプロセスのキ

ュー、procIdは呼び出したプロセス、monObjIdはプロセスが属するモニタインスタンス、transId は高信頼化処理用、curStkIdは現在使用中のスタック領域へのポインタ、xxxPtrは実行制御用である。なお、スタックには実行途中で移動可能なような形式でデータが格納されている。

頻出する整数と実数はメモリ効率を上げるために、有効桁数を落して下位２ビットにタグを入れる（整数＝00_2 、実数＝01_2 、ポインタ＝10_2 、ドメインId＝11_2）。

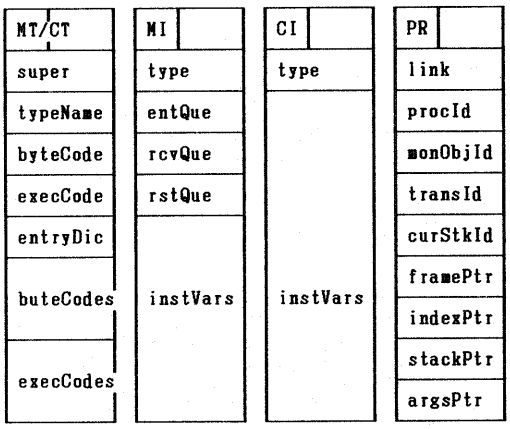

図６　主なオブジェクトの構造

６．３　オブジェクトの転送

ＯＺの分散型対象指向マシンの最大の特徴は、オブジェクトの構造を大域的に保存して転送することである。この方法については、既に発表している[13]のでここでは要点を述べるに留める。

オブジェクトをあるドメインから別のドメインに転送するには、送信側でオブジェクトを分解し符号化してＰＤＵに詰め、受信側でＰＤＵを復号化してオブジェクトに組み立てることが必要である。そのためにはオブジェクト間の関係を含む全ての情報をＰＤＵに入れなければならない。そこで、ＰＤＵにドメインと同じ構造を持たせ（これをＰＤＵドメインという）、送信側では送信元のドメインから構造を保存してＰＤＵドメインにコピーし、受信側ではＰＤＵドメインから受信側のドメインにマージして入れる（構造を保存したコピーとは、マージに過ぎない）。符号化規則＝復号化規則＝マージ規則である。

大域オブジェクトが移動する場合には、それまで登録されていたオブジェクト辞書のスロットのcellフィールドの下２ビットを 11_2 とし上３０ビットには移転先のドメインIdを入れる。自他ドメインを含めて他のオブジェクトが指している可能性があるためである。他のドメインが宛先オブジェクトが移転したのを知らないで、メッセージを送信する場合がある。このときには移転先に先送りする。これを繰り返すと先送りが何段にもなってしまう。ポインタの短縮が必要である。その方法には２通りある。第１の方法は、送信元に応答が戻ったときにポインタを付け変える方法であり、第２の方法は、移転の際に全体に移転先を知らせる方法である。移転先を記憶するスロットはどこからも指されなくなるとガーベッジコレクションで回収される。

オブジェクト転送時に常にタイプを送るのは効率が悪い。２通りの方法がある。第１の方法は、まずインスタンスだけを送り、受信先でマージの際にタイプの不在が解れば必要なタイプの一覧を送信元に戻して、再度タイプだけを送る方法である。第２の方法はタイプを記憶するドメインを設け、ビットベクトルを利用して必要なタイプだけをロードする方法である[13]。後者の方法は無駄なタイプの転送が省ける。いずれにしろ、一度ロードすれば再ロードは不要である。

６．４　ガーベッジコレクション

ガーベッジコレクション（ＧＣ）についても既に発表している[13]ので要点だけを述べる。ＧＣを大域ＧＣと局所ＧＣの２レベルで実施する方法である。局所ＧＣはどのアルゴリズムでも良い。

各オブジェクトには作られた時期あるいはＧＣで伝播された時期を記録する。局所ＧＣの際には、局所的なルートに記憶された時期をオブジェクト間に伝播させる（伝播される時期の方が新しければそれを当らしい時期にして）。ポインタが他のドメインに渡る場合には、宛先に時期を渡す。但し、同一の大域ＧＣの間には２度とは伝播させない。こうして全ドメインで局所ＧＣを行ってもドメイン間で時期の伝播がなくなったと解ると、次の局所ＧＣの際に先の大域ＧＣを開始した時期より古いオブジェクをゴミとして回収する。

６．５　名前づけ

一般的に分散システムでは、ファイル、各種サーバなど資源の数が多く、それらを一意に識別するために名前のつけ方が重要である。これはＯＺにおいても同様であるが、ＯＺでは分散システム用の対象指向型言語に特有な問題がある。

本分散型対象指向言語マシンでは、オブジェクトの転送と探索の効率を上げるために、内部ではオブジェクトは文字列の形をした名前は持たず、識別子で識別する。しかし、人とインタフェースをとる上では文字列の名前が不可欠である。そこで、各ユーザごとに名

前と識別子を一意になるように管理し、相互のマッピングを行う専用のディレクトリドメインを設ける。

分散型対象指向言語の都合で、タイプ、メソッド、アクションに一意な識別子をつける必要がある。これはプログラムをロードする際に行われる。ところが、このプログラムから、同じ分散型対象指向言語で書かれすでにロードされている別の世界のオブジェクトにアクセスしようとすると、識別子のマッピングが必要になる。識別子になったあとではマッピングのコストは非常に高い。オブジェクトId，ドメインId，ユーザIdなどは元来実行時に結合するものであるから問題はないが、タイプ、メソッド、アクションにつては工夫がいる。分散型対象指向言語のプログラム上で所望のディレクトリドメインを指定しておき、ロード時にそのディレクトリドメインから識別子を得ることにする。

7．　サーバとマネジャ

ＯＺシステムでは以下のようなサーバとマネジャを実現する予定である。

(1) ネットワーク管理アプリケーション
通常ＳＭＡＰ(System Management Application Program) と呼ばれるものであり、ネットワーク管理マネジャとオペレータとのインタフェースをとる。

(2) ディレクトリサーバ
ＭＡＰディクショナリをＯＺ用に拡張したサーバ。

(3) リソースサーバ
各種識別子の獲得・解放要求、ドメインのノードへの割り当て要求、などに動的に答えるサーバ。

(4) ファイルサーバ
ＯＺではファイル関係はすべてファイルサーバを使用する。ＩＳＡＭ形式のファイルをサポートする。

(5) ダウンロードサーバ
ディスクレスワークステーションにシステムをダウンロードするためのサーバ。

(6) プリントサーバ
レーザビームプリンタのサーバ。

(7) ゲートウエイサーバ
ＩＮＴＡＰシステムなどとのゲートウエイ。

(8) 仮想端末
当面は単純なもので間に合わせるが、将来的にはマルチウインド化。

以上のように、ＯＺはこれらを新規に作ることはしないで、既存システムをサーバとして活用する。

8．　むすび

分散型対象指向システムＯＺのアーキテクチャと実現方法について述べた。特徴として次のことが挙げられる。

(1) ＯＺのネットワークは、トークンバス方式ＬＡＮの特徴を十分に活かし、ＩＳＯに準拠する確認付きコネクションレス型プロフィルのＬＡＮである。

(2) ＯＺのネットワークは、応用層がＭＭＳであるためにサーバ類の接続が簡単である。

(3) ＯＺの分散カーネルは、ドメインという仮想計算機の格納場所を実現するものである。

(4) ＯＺは分散システムを分散型対象指向言語マシンとして見せ、ドメインやオブジェクトを動的に移動することができる。プログラム開発や動的負荷分散が可能となる。

(5) ＯＺのユーザインタフェースはプログラミング言語であり、分散処理のプログラミングがしやすい。

現在、１年後の完成を目指して実装を行っている。なお、本稿で述べた『確認付きコネクションレス型プロフィル』は、ＩＮＴＡＰ高速高機能ネットワークＷＧの『ＬＡＮ固有プロトコル』案として提案中のものである。

謝辞　　本研究の内容を御討論頂いた当所植村俊亮プログラム研究室長、二木厚吉言語処理研究室長、ＯＺの実装に御協力頂いている松下電器四反田秀樹氏、シャープ舟渡信彦氏、住友電工吉江信夫氏、ならびに３社の関係各位に感謝致します。また、本研究は筆者の一人のロボット用分散ＯＳの研究がベースとなっており、その機会を与えられた白井良明制御部長、高瀬国克システム制御研究室長にも深く感謝致します。

参考文献

1) Tanenbaum,A.S.etal.: Distributed Operating Systems, ACM Computing Serveys, Vol.17, No.4 (1985)
2) Svobodava,L.: Communication Support for Distributed Processing: Design and Implemenation Issues, LNCS Vol.248 (1987)
3)棟上、植村: 分散処理の展望、情報処理、Vol.28, No.4 (1987)
4)塚本: ローカルエリアネットワークの技術、情報処理、Vol.28, No.4 (1987)
5) ISO/2nd DP9506: Manufactuirng Message Service
6) ISO/TC97/6N4326: Proposal for ANetwork Item: Local Area Networks - Logical Link Control, Type 3 Operation - Acknowledged Connectionless Service
7) IEEE802.1 PartD: MAC Bridges, Revison A(1986)
8)塚本: ＯＳＩとＭＡＰ、昭和６２年電気学会産業応用部門全国大会 (1987,8予定)
9) General Motors: MAP3.0 Implementation Release The Mini-MAP Object Dictionary System(1987)
10)General Motors: MAP/TOP3.0 Network Management Requirements Specification (1987,3)
11)Goldberg,A. etal.: Smalltalk-80, The Language and its Implementation,Addison-Wesley(1983)
12)Hoare,C.A.R.: Monitors: an Operating System Structuring Concept, CACM, Vol.17, No.10 (1974)
13) 塚本: 対象指向型分散処理システム－対象指向型分散カーネル－、情報処理学会ＯＳ研究会３３－３(1986,12)