システムソフトウェアと
オペレーティング・システム 73－8
（1996．8．26）

オブジェクト指向分散環境 OZ++のアプリケーションゲートウェイの実装

濱崎 陽一[1] 西岡 利博[2*] 塚本 享治[1]
hamazaki@etl.go.jp nishioka@mri.co.jp tukamoto@etl.go.jp

1: 電子技術総合研究所 2: 三菱総合研究所
〒305 つくば市梅園 1-1-4 〒100 千代田区大手町 2-3-6

*: 情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」研究員

ネットワーク上でクラスとオブジェクトの共有と交換が可能なオブジェクト指向分散環境OZ++で広域網を介した通信を実現するアプリケーションゲートウェイの実装について述べる。

アプリケーションゲートウェイはファイアウォールマシン上で動作し、メソッド起動、アドレス解決、クラス探索およびクラスファイル転送といったOZ++特有の通信の中継を行なうプログラムである。アプリケーションゲートウェイは単に通信を中継するだけではなく、オブジェクトの実行系と共にサイト間通信のセキュリティ機能を実現する。

The Implementation of the Application Gateway for OZ++ : An Object-Oriented Distributed Systems Environment

Yoichi Hamazaki [1] Toshihiro Nishioka[2*] Michiharu Tukamoto [1]

1:Electrotechnical Laboratory 2:Mitsubishi Research Institute, Inc.
1-1-4, Umezono, Tsukuba, Ibaraki, 301 Japan 2-3-6 Otemachi, Chiyoda-ku, Tokyo, 100 Japan

*: Researcher of Research, Development and Evaluation of Open Fundamental Software Technology in Information-Technology Promotion Agency, Japan

This paper presents the implementation of the OZ++ application gateway which provides communications over wide-area networks among sites of OZ++ : an object-oriented distributed systems environment.

The application gateway is a program which runs on a firewall machine and relays communications of OZ++ between inside and outside of the firewall. Communications of OZ++ include method invocation, address resolution, class search and class-file transfer.

The application gateway provides security against the attack from outside by marking packets of method invocations. The executor which executes methods of objects restricts functions for threads which are created according to marked packets.

1 はじめに

OZ++はネットワーク上でクラスとオブジェクトの共有と交換が可能なオブジェクト指向分散環境である。本稿では、OZ++ の広域網を介した通信を実現するアプリケーションゲートウェイの実装について述べる。アプリケーションゲートウェイはファイアウォールマシン上で動作し、メソッド起動、アドレス解決、クラス探索およびクラスファイル転送といったOZ++特有の通信の中継を行なうプログラムである。アプリケーションゲートウェイによって広域網を介した広域分散環境がOZ++によって実現できる。

アプリケーションゲートウェイの導入によって、不特定多数のユーザの遠隔からのアクセスが可能になるため、セキュリティの確保が課題となる。セキュリティの観点から広域網にまたがる全域でユーザ認証が完全にできることが望ましい。しかし規模、性能の両面で、実用的かつ自由に配布できる広域認証システムがまだないことから、OZ++ではサイト間では認証は行なわず、サイトの外はすべて信用できないものとして扱うポリシーをとり、セキュリティレベルを保っている。こうしたセキュリティポリシーを実現するために、アプリケーションゲートウェイによってサイトの内外からのアクセスが区別できるようなっている。

最初にOZ++システムの概要とサイト内の通信について説明し、次にアプリケーションゲートウェイの実装について述べる。最後にワールドワイドプログラミング環境をめざした、今後の課題について述べる。

2 OZ++システムの概要

OZ++はネットワーク接続された計算機群の既存のオペレーティングシステム上にミドルウェアとして実現されている。一つの計算機の上には、オブジェクトを保持し、そのメソッドを実行する仮想計算機であるエグゼキュータが任意個、計算機上のエグゼキュータを管理するニュクリアスが一個、そしてクラスファイル転送に用いるファイル転送用のデーモンプロセスが送受一対存在する (図 1)。

ネットワーク上のブロードキャストが届く範囲内にある計算機群がOZ++の物理的な管理範囲であるサイトを構成する [2]。ニュクリアスは固定のポート番号を持つ通信ポートを持っており、ブロードキャストによって通信しあう事が可能である。

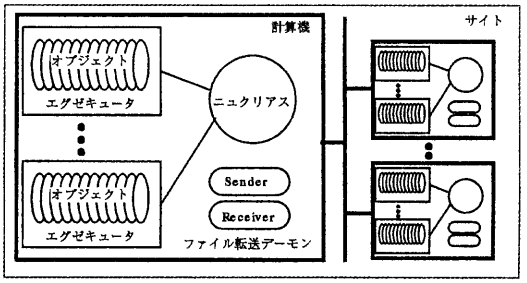

図 1: サイトの構成

2.1 オブジェクトとクラスの識別子

オブジェクト[1]は、全世界でユニークな識別子を持ち、任意の位置からアクセスされうる。また、クラスは全世界で共有できるようにやはりユニークな識別子を持つ。

識別子の生成／管理を簡単にするために、識別子は固定長で階層的な構造を持つ (図 2)。サイト ID は集中管理を行ない、管理組織において要求に応じて順次発行する。サイト ID とそれを管理するアプリケーションゲートウェイの通信アドレスの対応表はサイト情報として管理組織によって公開される。

サイト内では、エグゼキュータの番号を重複がないように発行する。サイト ID とエグゼキュータ番号を合わせてエグゼキュータ ID と呼ぶ。エグゼキュータ ID から、それが属するサイトが特定できる。また、エグゼキュータが起動される計算機はサイト内で起動のたびに変更される可能性がある。

オブジェクトは、それが生成されたエグゼキュータで識別子を与えられ、その識別子 (オブジェクト ID と呼ぶ) から、それが存在するエグゼキュータを特定できる。オブジェクトは生涯同じエグゼキュータ上に留まる。

クラスも同様に、そのクラスがコンパイラにより生成され、クラスを管理するオブジェクトに登録される際に識別子 (クラス ID と呼ぶ) が与えられる。クラスは必要に応じてネットワーク上を配送され、複数のコピーが存在しうるが、同じクラス ID で参照されるクラスは同じものであることはOZ++の管理機構により保証されている。

[1] 任意の位置のオブジェクトから呼び出しできるオブジェクトであるのでグローバルオブジェクトとOZ++では呼ぶ [1]。それ以外にグローバルオブジェクトに内包され、グローバルオブジェクト内でのみ参照可能でグローバルオブジェクト間で call_by_value で渡されるローカルオブジェクトがあり、2 レベルのオブジェクトモデルを採用している。本稿では、特に断らないかぎりオブジェクトという語をOZ++のグローバルオブジェクトの意味で使用する。

サイトID (16bit)	エグゼキュータ番号 (24bit)	番号 (24bit)

図 2: 識別子の構造

2.2 サイト内の通信

オブジェクトIDによって示される任意のオブジェクトのメソッドを起動するためには、オブジェクトIDからそのオブジェクトが存在するエグゼキュータの通信アドレスを求めるアドレス解決を行ない、メソッドを起動する。メソッドの実行に際してクラスがそこに存在しない場合にはクラス探索が必要で、そして遠隔の計算機上にクラスが見つかった場合にはクラスファイルの転送が必要になる(図3)。以下にそれぞれの通信の概略を述べる。

2.2.1 サイト内のアドレス解決とクラス探索

先に述べたように、個々のエグゼキュータが起動される計算機は固定してはおらず、またクラス(およびそのコピー)もどこにあるかは事前に特定できない。こうした位置情報が特定できない対象を探索するためにニュクリアス間のブロードキャストを用いる。

ニュクリアスはエグゼキュータのIDと位置(通信アドレス)の対応表であるエグゼキュータ表を管理しており、エグゼキュータはその計算機のニュクリアスの持つエグゼキュータ表を参照することができる。エグゼキュータはエグゼキュータ表にないエグゼキュータとの通信が必要になった場合に、ニュクリアスにアドレス解決を要求する。アドレス解決要求を受けたニュクリアスはブロードキャストによって他のニュクリアスに問い合わせると、該当するエグゼキュータを管理しているニュクリアスは、その位置を返答する。

クラスはオブジェクトによって管理されているので、クラス探索の要求はエグゼキュータ上のクラス管理オブジェクトにまで達しなければならない。そこで要求はニュクリアスによってブロードキャストされ、それを受けたニュクリアスが自分が管理しているエグゼキュータにマルチキャストする。エグゼキュータはそのエグゼキュータ上のクラス管理オブジェクトにクラス探索の要求を渡す。クラス探索の返答は、オブジェクト間の通信(メソッド起動)によって行なわれる。

2.2.2 サイト内のメソッド起動

メソッド起動はエグゼキュータ間の通信によって行なわれる。通信は、最初に呼びだし側と呼び出される側のオブジェクトID、メソッドのセレクタなどを持つCall_Reqパケットを送り、ひき続き引数を含んだCall_Argパケットを送るという順序で行ない、それに対して、メソッド実行の結果に応じてResult,Exception,Errorのいずれかのパケットが返される。

エグゼキュータ上には複数のオブジェクトが存在し、それらの間のメソッド起動の各々にコネクションを割り当てるのでは資源の無駄使いとなるので、メソッド起動ごとに固有の番号(メッセージID)を与え、そのメソッド起動に関連したパケットに同一のメッセージIDを持たせる事により、通信の多重化を図っている。

2.2.3 サイト内のクラスファイル転送

OZ++ではクラスそれ自身はオブジェクトではなく、その実体はファイルとして管理される。エグゼキュータはクラス管理オブジェクトからファイルへのパスを得ることにより所望のクラス(実行コード)をロードし実行する。

所望のクラスをクラス管理オブジェクトが持っていなかった場合にはクラス探索によって、所望のクラスを管理しているクラス管理オブジェクトのオブジェクトIDと、ファイルのパスを得る。所望のファイルに直接アクセスできない場合には、そうしたクラスファイルを計算機間で転送する必要があり、そのためにファイル転送用デーモンプロセスを用意している。

クラスファイルの転送は、クラス管理オブジェクトがクラス探索の結果得られたオブジェクトIDからファイル転送元の計算機を特定し、受信デーモンを経由してその計算機上の送信デーモンにファイルのパスを送って要求することによりなされる。それぞれのデーモンはファイルのデータ転送用にサブプロセスをフォークして、実際に転送が行なわれる。

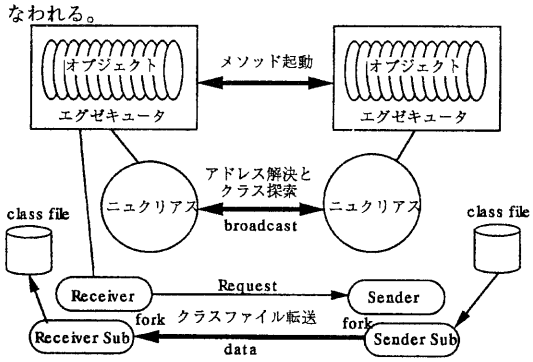

図 3: サイト内の通信

3 OZ++アプリケーションゲートウェイ

セキュリティの観点から通常、組織の内外はファイアウォールで区切られ、インターネットなどの広域網から直接アクセスできるのはファイアウォールマシンのみである。組織内の他の計算機はファイアウォールマシンで中継

しなければインターネットを介した遠隔の計算機と通信できない。

OZ++のサイトは組織内の計算機で構成され、サイト間の通信にはファイアウォールマシン上で動作するOZ++のアプリケーションゲートウェイがその仲介をする(図4)。一つの組織内に複数のサイトが存在することは可能であり、その場合には一つのアプリケーションゲートウェイが複数のサイトを管理することになる。

サイト内では、アドレス解決はニュクリアス間の通信で、メソッド起動はエグゼキュータ間の通信で、クラスファイル転送はファイル転送デーモン間の通信で実現されるが、サイト間にまたがる時にはこれらをアプリケーションゲートウェイが仲介する必要がある。それぞれに対応して、それぞれ別の受信ポートをアプリケーションゲートウェイは持つ。また、サイト内の一つのニュクリアスはアプリケーションゲートウェイと通信する機能を持たせ、これをリレーニュクリアスと呼ぶ。

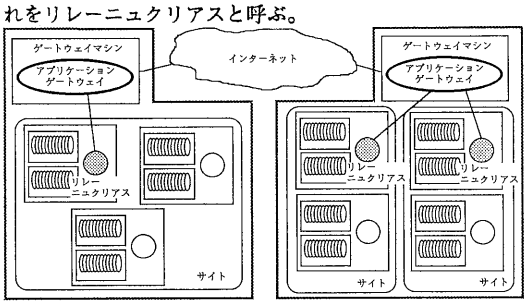

図 4: アプリケーションゲートウェイとサイト

3.1 インターネットセキュリティ

OZ++では実用的で自由に配布できる広域の認証のためのシステムがまだないことと遅延時間の大きな広域網は認証に時間がかかることから広域の認証は行なわず、サイト外からのアクセスに関しては一律に危険があるものとして扱うというポリシーをとった[3]。

OZ++において危険を伴うと考えられるのは、メソッド起動によるサイト外部からのアクセス、メソッド起動の引数あるいは返り値としてサイト外部から転送されるオブジェクト[2]の挙動、サイト外部から転送されるクラスの利用である。

3.1.1 メソッド起動

最初の二つに対応するために、メソッド起動に関連するパケットはアプリケーションゲートウェイで中継の際に外からやって来たというのマーク（外来マーク）をつける事とし、外来マークによってエグゼキュータがセキュリティ上の制限を行なうこととした。これは、アプリケーションゲートウェイのみでは中継をするか否かという大雑把な対処しかできないからである。

サイト外部からのアクセスは外来のマークのついたCall_Reqパケットにより開始されるので、エグゼキュータはそのスレッドに外来のマークをつける(外来スレッド)。また、外来のマークのあるパケットから取り出されたオブジェクトにもエグゼキュータは外来のマークを付ける(外来オブジェクト)。これら外来のマークのあるものから派生したもの(例えば外来オブジェクトのメソッドを実行しているスレッド)にも外来のマークを付けていく。このような外来スレッドに、ファイル操作などセキュリティ上問題になる可能性のある操作を禁止することによりセキュリティを実現している。

3.1.2 クラスファイル転送

他のサイトから転送されたクラスファイルについては、実行コード(ネーティブコード)が不正なものと置き換えられているなどの可能性があるのでそのまま使うのは危険である。そこで、同時に転送されて来るソースコードから再コンパイルし、実行コードのコンパイラインテグリティを確保すると共に、危険なコードを排除する。

クラスファイルがサイト外からもたらされるかどうかは、クラスを管理するオブジェクト間のメソッド起動により返されるオブジェクトが外来か否かで判断できるので、クラスファイル転送では特別な処置は不要である。

3.1.3 組織内の特例

一つの組織内のサイト間、すなわち一つのアプリケーションゲートウェイが管理しているサイト間では、相互に信頼して外来のマーキングをしないようにアプリケーションゲートウェイを設定することが可能である。さらにそれらのサイト間がIP到達可能であれば、アプリケーションゲートウェイはアドレス解決とクラス探索のみの中継を行ない、メソッド起動やクラス転送については直接行なうようにアドレス解決するような設定も可能である。

3.2 サイト間のアドレス解決

リレーニュクリアスは受信したアドレス解決要求が他サイトのエグゼキュータに関するものの場合、それをアプリケーションゲートウェイに転送する。アプリケーションゲートウェイは問い合わせを該当サイトのアプリケーションゲートウェイに転送する。アプリケーションゲートウェイの通信アドレスは、公開されているサイトの情報から得

[2]これはグローバルオブジェクト間で転送されるローカルオブジェクトである。本節中のオブジェクトはすべてローカルオブジェクトである。

られる。他のサイトからの要求を受けたアプリケーションゲートウェイはリレーニュクリアスにブロードキャストを要求し、それに対する返答を転送するように依頼する。

サイト間のアドレス解決の例を図5に示す。要求パケットには要求元のアドレス情報が入っており、それをアプリケーションゲートウェイで順次書き換えて転送していく。例えばAG1では、要求元のEx1のアドレスを自分のメソッド起動中継用アドレスに書き換えて転送する。応答のパケットも同様に順次書き換えられながら転送されていく。アドレス解決が終了した時には図に示したように、両方のエグゼキュータのエグゼキュータ表とそれを仲介したアプリケーションゲートウェイのエグゼキュータ表に中継経路の情報が生成される。

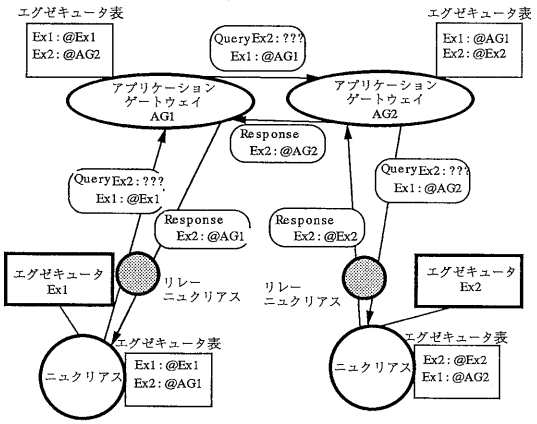

図 5: サイト間のアドレス解決

3.3 サイト間のクラス探索

クラスを探す順序は以下の通りである。

1. 自サイト内をブロードキャストで探す
2. 有用なクラスを蓄積しているサーバオブジェクトに問い合わせる
3. そのクラスが生成されたサイトを探す

リレーニュクリアスは、クラスをサイト外に探す機能を果たすクラスリクエストオブジェクトからの要求でアプリケーションゲートウェイにクラス探索の要求を送り、要求されたサイトのリレーニュクリアスはそのサイトで外部に提供するクラスを管理するクラスサービスオブジェクトに要求を送る。

クラス探索のポリシーは、クラスリクエストオブジェクトとクラスサービスオブジェクトによって決められ、アプリケーションゲートウェイや通信機構はそれに関知しない。

3.4 メソッド起動

サイト間のアドレス解決により、メソッド起動のパケットの転送経路は事前に確定している(図6の例では、図5で示したようにEX1 → AG1 → AG2 → EX2の経路でCall_Req, Call_Argのパケットは転送される)。メソッド起動の最初のパケットCall_Reqには、メッセージIDと共に双方のオブジェクトのIDが含まれる。アプリケーションゲートウェイはこうした情報を読み出してメッセージ表として管理する。Call_Req以外のパケットはメッセージIDをキーにメッセージ表を検索し、さらにエグゼキュータ表を使ってその行き先を決める事ができる。結果のパケットの最後のものを中継したらメッセージ表から該当メッセージIDの情報を削除する。

またセキュリティのために、アプリケーションゲートウェイがメソッド起動に関連したパケットを中継する際に、外来のマークを付けるのは、先に述べたとおりである。

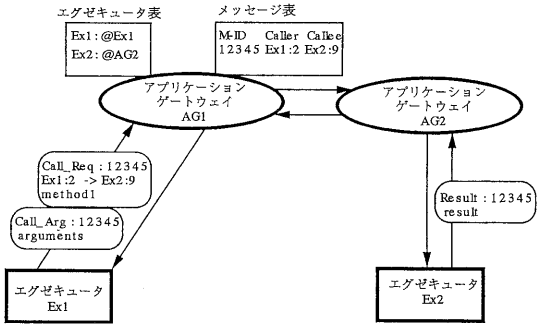

図 6: サイト間のメソッド起動

3.5 サイト間のクラスファイル転送

クラスファイル転送では、サイト内の場合と整合性のよい転送用のサブプロセスをフォークする実装とした。クラスファイル転送に先だって、必ずクラスを管理するオブジェクト間でメソッド起動が行なわれるので、クラスファイル転送を開始する時には必要なアドレス解決が終了している。そのため、アプリケーションゲートウェイではエグゼキュータIDをキーにアドレス解決を行ない、順次要求を転送して行く事ができる。

3.6 性能評価

アプリケーションゲートウェイを介したメソッド起動の性能をethernetで接続された4台のワークステーション(いずれもsparcstation2)を用いて測定した。測定は、同一エグゼキュータ内、計算機間、計算機間でアプリケー

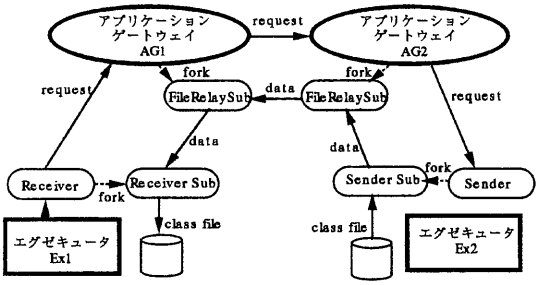

図 7: サイト間のクラスファイル転送

ションゲートウェイを 1 段あるいは 2 段介した場合の 4 つの場合について行なった。

測定したメソッドは引数、返り値ともに無いもの (図中では void と示した)、引数、返り値ともに文字列で、その長さが 100 から 10000 の各場合である。実際に転送される引数のデータ量は配列の構造体 (32B) が加わるので、長さ 100 の文字列で 132B で、結果も同じデータ量である。測定結果を図 8に示す。測定結果は数回の試行の平均値である[3]。但し、マルチユーザ OS でネットワークを利用している状況での計測であるので、厳密な測定ではない

測定結果から、アプリケーションゲートウェイ 1 段あたりのオーバーヘッドは約 3.5ms であり、計算機間のメソッド起動の速度約 9ms(void の場合) に比較して充分な性能が得られている。また、データの転送はアプリケーションゲートウェイのない場合が約 200kB/sec であるのに比べてアプリケーションゲートウェイ 2 段の場合は約 160kB/sec となり、2 割ほど遅い。ただし、これは ethernet 上で測定したためであり、低速の広域網を介した場合には無視できるものである。

また、配列の大きさが 1000 と 2000 での差が大きいが、2000 以上ではメモリの 1 ページ (4kB) にひとつしか領域がとれないために、メモリアロケーションのオーバーヘッドが大きくなるためである。

4 まとめと今後の課題

サイト間でOZ++のメソッド起動、アドレス解決、クラス探索そしてクラスファイル転送を実現するアプリケーションゲートウェイの実装について述べた。アプリケーションゲートウェイは単に通信を中継するのみではなく、セキュリティ機能をもっている。

今後、地球規模でソフトウェア (クラス) やサービス (オブジェクト) を共有し、それらを利用することによって作られたソフトウェアやサービスがネットワーク上に公開され、ソフトウェア資源がネットワーク上を循環し、蓄積されていくようなワールドワイドプログラミング環境のOZ++による構築をめざしている。

[3]OZ++は GC を行なうが測定が GC の影響を受けないように GC の起こらない範囲で測定した。そのため引数が大きなものほど一試行あたりループ回数は少ない

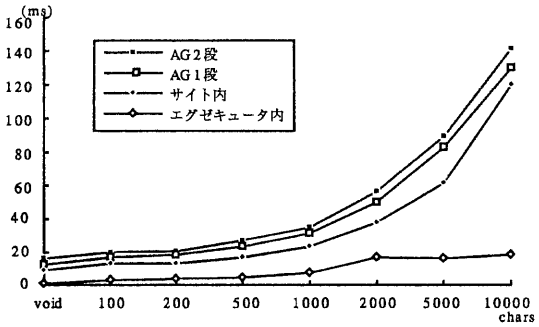

図 8: メソッド起動の性能

まず、以下の課題を解決して、アプリケーションゲートウェイを含むOZ++システムをインターネットに公開する予定である。

1. サイト ID の運用管理とサイト情報の提供方法
2. クラスやオブジェクトをネットワーク上に公開する手法
3. ネットワーク上に公開されたクラスやオブジェクトを検索し、利用する手法

さらに、OZ++による広域分散環境上でワールドワイドプログラミングを試行したいと考えている。

本研究は、情報処理振興事業協会 (IPA) の「開放型基盤ソフトウェア研究開発評価事業」の一環として行われたものである。

参考文献

[1] 塚本、浜崎、音川、西岡：「クラスの共有と配送にもとづくオブジェクト指向分散システムの設計と実現」、情報処理学会論文誌、May.1996

[2] Y.Hamazaki,M.Tsukamoto,M.Onishi,Y.Niibe: "The Object Communication Mechanisms of OZ++ : An Object-Oriented Distributed Programming Environment",Proc. ICON-9,Dec.1994

[3] 大西、濱崎、西岡、塚本：「オブジェクト指向分散環境OZ++におけるインターネットセキュリティ」、マルチメディア通信と分散処理シンポジウム、Oct.1995