分散OS ToMにおけるネットワーク上のRPCの設計

3K−6

岡本 利夫
(株) 東芝 総合研究所

桜川 貴司
京都大学 数理解析研究所

堀切 和典
富士ゼロックス(株) システム技術研究所

山岸 久夫
(株) ビクター・データ・システムズ 技術部

1 はじめに

分散OS ToM[1]ではプロセス間通信の手段として**RPC** (Remote Procedure Call)[1] を採用した。RPCは、C言語のような、サブルーチン・コールの関数型プログラミングスタイルにマッチしている。また、ToMのプログラミングモデルであるスレッドとモジュールの独立という概念にもマッチしている[2]。

ToMでは、分散環境においてホスト間にまたがっても、また、マシンアーキテクチャが異なってデータ表現が異なっても、このRPCのコンセプトを維持できるようにネットワークコア(NC)が処理を行っている[3]。本論文では、ToMのホストをまたがるRPCの実現方法とネットワークプロトコルについて説明する。

2 ToMのRPC

ToMでプロセス間通信の手段としてRPCをプリミティブとした理由は三つある。一つはプロセス間通信の多くが同期型であること。もう一つは、VMTP等のRPC向きの新しい通信プロトコルを用いることにより、非同期型の通信でRPCを実現するのに比べ、高速化やエラー処理などが容易になることである[4]。最後に、RPCは手続き呼び出しの形のためユーザがプログラミングしやすのは言うまでもない。

さらに、ToMにおいては、スレッドとモジュールは独立しているため、他のスレッドに手続きの実行を依頼するのではなく、そのスレッドのカレント・モジュールが切り替わってそのまま実行する。

ToMのRPCは、ケイパビリティ（RPC先のモジュールにアクセスする権利）で、飛び先を指定する。ケイパビリティはミニマルコアの内部で管理され、セキュリティを高めている。ToMでは、すべてのプロセス間の通信は、RPCで統一している。システムコールも、WELL-KNOWNケイパビリティへのRPCの1つであり、ユーザ作成のモジュールへのRPCと同様に扱える。

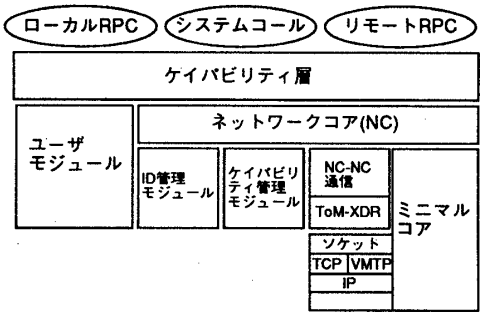
以降、図を参照しながら、OSIのモデルにしたがって、ToMのネットワークモデルを説明する。

3 アプリケーション層

3.1 ケイパビリティ層

ユーザが通常利用する層。RPC先のモジュールの存在位置を隠蔽してあり、RPCはグローバルなケイパビリティへのコールとして取り扱われる。各種のオブジェクトのID（モジュールID、スレッドIDなど）もマシンに依存しないグローバルIDである。＜RPC先の指定、引き数のデータ形式＞は、＜グローバルケイパビリティID、ローカルデータタイプ＞となる。

[1]ToMの研究・開発は、京都大学、慶應義塾大学、京都高度技術研究所が企業(エスアールエー、オムロン、キヤノン、三洋、ソニー、東芝、日本鋼管、日本サン・マイクロシステムズ、日本・データゼネラル、日本ビクター、ビクター・データ・システムズ、富士ゼロックス、横河・ヒューレット・パッカード、リコー)と共同してすすめている。



3.2 ネットワークコア(NC)層

ネットワークコアが存在する層。ネットワークコア自身もユーザアプリと同様であり、RPCによって、他のモジュールと通信する。ここでは、大きく2つの処理を行う。1つは、システムコールの実現。もう一つは、リモートのRPCの実現である。なお、効率のため、ローカルモジュールへのRPCは、NC層を通らず直接そのモジュールへRPCされる。ここでは、上位層からのRPCが処理され、下位層（ミニマルコア、ID管理モジュール、ケイパビリティ管理モジュールなど）へRPCまたはシステムコールされる。また、NC-NC通信を経て、他ホストのネットワークコアと通信する。NC-NC通信へは、＜ホストID、インストラクションID、ローカルデータタイプ＞の形で通信する。

3.2.1 システムコールの処理

システムコールとは、狭義にはミニマルコアへ対してのものだけであるが、ミニマルコアはローカルな処理しか行わない。ホスト境界を意識させないグローバルなシステムコールを実現するため、ユーザから発せられたシステムコールは、一度、ネットワークコアーが横取りして、各種の処理を行い、改めてミニマルコアのシステムコールへ渡す。ここでは主に、グローバルIDとマシンローカルのIDとのID変換を行う。これには、ID管理モジュールを用いる。つまり、ここでは、＜グローバルID＞を＜ホストID+ホストローカルなID＞に変換する。そして、ローカルまたはリモートのホストの、ミニマルコアへのシステムコールの組み合わせに変換し、実行する。

3.2.2 リモートのRPCの処理

他のホストのモジュールへのRPCを実現する。飛び込んできたケイパビリティIDにより、目的のホスト、目的ホスト上のケイパビリティID他をケイパビリティ管理モジュールから得る。ケイパビリティ管理モジュールでは、＜グローバルケイパビリティID＞を＜ホストID+そのホストのネットワークコアが保持しているローカルなケイパビリティID＞に変換する。そして、NC-NC通信を経て相手のホストのNC上に自分のスレッドと同じイメージのスレッドが生成され、目的のモジュールへのRPCが実現する。

The Design of RPC through Network for a Distributed Operating System ToM

Toshio OKAMOTO (TOSHIBA CORPORATION)　Takashi SAKURAGAWA (Kyoto University)
Kazunori HORIKIRI (FUJI XEROX CO., LTD.)　Hisao YAMAGISHI (Victor Data Systems CO., LTD.)

3.3 NC-NC 通信

ホスト間のデータの転送と実行の制御を行う。ここでは、以下の項目の処理を行う。

1. スレッドの移動

   仮想的に、相手ホスト上のネットワークコア上にスレッドを移動させて目的モジュールへRPCする。つまり、相手ホスト上に、目的のケイパビリティの指定して、スレッドを生成し、グローバルなスレッドIDを登録し、各アーギュメントデータの転送し、RPCする。そして、リターン処理する。

2. リモートホスト実行

   指定されたホスト上でシステムコールの実行を依頼する。構成は、リモートRPCと同じである。

3. リモートNC実行

   一つの処理をリモート側のネットワークコア上で行ったほうが効率が良い場合、まとめて依頼する。例えば、同じホストに繰り返しリモートホスト実行する場合、ネットワークトラフィックスを減らすために有効である。

4. その他

   NC自身の状態を調べる。

以上の処理は、＜ホスト名、インストラクションID、ローカルタイプデータ＞の形で通信される。

4 プレゼンテーション層

ToMのRPCのアーギュメントはすべて型付きである。ローカルの場合は、変換しないでそのまま送る。リモートの場合は、マシンアーキテクチャに依存しない中間形式に変換したのち相手に送付する。具体的にはSUNのXDRをベースに作成している[5]。ただし、データタイプのうち、ケイパビリティは特別な処理が必要になる。ここで、データは、＜ローカルタイプデータ＞から＜ToM-XDRタイプデータ＞に変換される。

アーギュメントの中にケイパビリティがある場合には特殊な処理が必要になる。相手側のホストのケイパビリティ管理モジュールに依頼して、インタフェース用のケイパビリティを作成する。そして、そのIDを、相手側で使用する。

なお、現在のインプリメントでは、プレゼンテーション層は、上記のNC-NC通信部に含まれている。

5 セッション層以下

現在のバージョンは、ミニマルコア内にUNIX（TAHOE版）用のコードを移植してある。そのため、セッション層はUNIXのソケット相当のインタフェースを用意し、トランスポート層以下は、TCP/IPを利用してコネクションを張っている。次バージョンからは、RPCに有利なVMTPも利用する。

5.1 VMTP

VMTP[4][6]では、以下の三点の機能面での改良を加え設計されている。

1. RPCに適したプロトコル

   VMTPはリクエストーレスポンス型のコネクションレス・プロトコルを採用し、RPCを行なう際に2パケットのハンドリングで完了するので高速通信が実現できる。また、明確なアクノレッジメントの概念、再送とタイムアウトの機能で高い信頼性も実現している。

2. ネーミング

   ホストアドレスと独立した64bit長のネーミングを可能とし、上位層に対し解放している。これによりinternet-addressに捕らわれないネーミングができ、アプリケーションレベルにおけるcommunication end-to-endでの認識ができることとなる。

3. トランスポート・レベルでの機能の追加

   VMTPは、

   - マルチキャスト機能の追加（IP multicastの上で動作）
   - パケットレベルでの転送機能（forward）のサポート
   - real-time datagramのサポート

   などの、豊富な機能を備えている。

5.2 実装への課題

VMTPの実装に際し、以下の点が検討課題となっている。

1. プロトコルの実装方法

   現在、VMTPをスタンフォード大から提供されているUNIXへの実装用のもの利用する予定でいるが、インタフェースがソケットを前提としたものとなっている。しかし、ソケットは、RPC向きのインタフェースでないため、次バージョンからは、ToMのRPCモデルに最適のVMTPのインタフェースを提供する必要がある。また、コアの軽量化のため、物理層以外をミニマルコアから分離してプロトコル・サーバの形で実装することも検討している。

2. グループの管理

   マルチキャスト機能とも関係するが、グループの認識の仕方を検討する必要がある。例えば、マルチキャストの対象を同一ハウス内にするか、ハウスに捕らわれずにローカル・ネットワーク全体にするか、等のことが検討要因である。

6 おわりに

本論文では、現在開発中の分散OS ToMにおけるRPCの実装の概要についてネットワーク構成と関連させて説明した。現在、誠意インプリメント中であり、評価・考察について触れることができなかったが、本論文を拡大した別の論文で取り扱いたいと思う。最後に、本論文を書くにあたって、ToMのアイデアをまとめるために熱心に議論してくださった、分散OS研究・開発プロジェクトのみなさんに感謝します。また、プロジェクトを強く押し進めていただいている京都大学数理解析研究所の中島玲二教授に深く感謝します。

参考文献

[1] Birrel, A. and Nelson, B.: "Implementing Remote Procedure Calls", ACM Transaction on Computer Systems, Vol. 2, No. 1, pp. 39-59, 1984.

[2] 新井 潤, 桜川 貴司, 立木 秀樹, 萩野 達也, 服部 隆志, 森島 晃年: "分散環境をサポートするOS ToMの構想 — そのプログラミング・モデルとセキュリティ機構 —", 日経エレクトロニクス, 10月16日号, pp. 187-199, 1989.

[3] 岡本 利夫, 桜川 貴司, 堀切 和典, 山岸 久夫: "分散OS ToMにおけるネットワークの概要", 日本ソフトウェア科学会第7回大会論文集, pp. 337-340, 1990.

[4] Cheriton, D.: "VMTP: A Transport Protocol for the Next Generation Communication Systems", Proceedings of the ACM SICOMM'86 Conference, Stowe, Vermont, August 5-7, 1986.

[5] Sun Microsystems, Inc.: "XDR: External Data Representation Standar", RFC-1014, 1987.

[6] Cheriton,D.: "VMTP: VERSATILE MESSAGE TRANSACTION PROTOCOL Protocol Specification", RFC-1045, 1988.