分散システム／インターネット運用技術 18－1
（2000．7．10）

ワールドワイドなインターラクティブシステムのための HTTP コネクション型 RPC の検討

平山秀昭† 本多弘樹† 弓場敏嗣†

WWW(World Wide Web) が開発されて以来、インターネットの利用者数は増大の一途を辿っている。現在、その用途の大半は単純な情報検索であるが、EC(Electronic Commerce) への期待が高まる中、次第に WWW での処理が複雑化してきている。本論文では、ワールドワイドに利用されるグループウェアのようなインターラクティブなアプリケーションモデルを想定し、それを効率よく実行する HTTP コネクション型 RPC という方式を提案する。この RPC は、URL によるサーバアプリケーション指定、サーバからクライアントへのコネクション、同期 / 非同期 RPC が可能であるという特徴を持つ。それによって、WWW や一般的な C/S 型アプリケーションでは困難だったインターラクティブなアプリケーションモデルのためのワールドワイドで柔軟な分散処理環境を提供する。

HTTP Connection Based RPC for World Wide Interactive Systems

HIDEAKI HIRAYAMA,† HIROKI HONDA† and TOSHITSUGU YUBA†

Since WWW (World Wide Web) has been developped, users of the Internet are extremely increasing. Now WWW is mostly used for retrieving information. But more complicated style of distributed processing is increasing according to the needs of EC (Electronic Commerce). In this paper, we propose a new scheme named "HTTP connection based RPC" for world wide interactive application model such as groupware. The RPC has the features such as direction of server applications by URL, connection from server applications to client applications and synchronous/asynchronous RPC. It provides world wide and flexible distributed processing environment for interactive application model.

1. はじめに

WWW(World Wide Web) が開発されて以来、インターネットの利用者数は増大の一途を辿っている。現在、その用途の大半は単純な情報検索であるが、EC(Electronic Commerce) への期待が高まる中、次第に WWW での処理が複雑化してきている。しかし、WWW は元来、ホームページへのアクセスという単純な情報検索を指向したものである。そのため、そこで使用される HTTP(Hyper Text Transfer Protocol) は、相変わらず 1 リクエスト / 1 リプライを基本としていて、複雑な処理には適していない。今後のインターネットの多様なニーズに応えていくためには、HTTP をそのまま用いるのではなく、C/S(Client and Server) 型アプリケーション本来の柔軟性を引き出す仕組みが必要である。

一方、一般的な C/S 型アプリケーションは、WWW の URL のように、ワールドワイドで動的に追加 / 削除されるアプリケーションを簡単に指定する方法を持たない。そのため、例えば LDAP(Lightweight Directory Access Protocol)[1] 等のディレクトリサービスによって、サーバアプリケーションが実行されるノードのアドレスとポート番号を管理しなければならない。これは基本的にはイントラネットでの固定的なサービスを指向したものであり、WWW のようなワールドワイドなサービスには適していない。C/S 型アプリケーションを WWW のようにワールドワイドで用いるには、このアプリケーションの指定方法の問題を解決する仕組みが必要である。

上述したようにワールドワイドで、現在の WWW よりも柔軟なアプリケーションサービスを提供しユーザニーズの多様性に応えるには、現在の WWW でも、一般的な C/S 型アプリケーションでも不十分である。本論文では、ワールドワイドで利用される新しいアプリケー

† 電気通信大学大学院情報システム学研究科，調布市
Graduate School of Information Systems, The University of Electro-Communications, Chofu-shi, Tokyo 182-8585, Japan

ションとして、グループウェア等のインターラクティブなアプリケーションモデルを想定する。そして、そのアプリケーションモデルを実現するための仕組みとして、HTTP コネクション型 RPC(Remote Procedure Call) という方式を提案する。

2. 想定するアプリケーションモデル

図 1を用いて想定するアプリケーションモデルを説明する。それは、ワールドワイドに存在するグループウェアのようなインターラクティブな C/S 型アプリケーションである。このアプリケーションは、WWW のホームページのように動的に追加 / 削除が行われる。すなわち、新しいアプリケーションの登録、古いアプリケーションの抹消が、ワールドワイドで日常的に行われる。

図 1の Community1 および Community2 は、ここで想定する C/S 型アプリケーションのサーバアプリケーションを示す。一方 A から F は、想定する C/S 型アプリケーションのクライアントアプリケーションを示す。状況に応じて、クライアントアプリケーション A から F は、Community1 に接続する場合もあるし、Community2 に接続する場合もある。図 1では、Community1 に A、B、D が接続している。そして、これらが相互に、メイル等の情報を交換する。

例えば A が、メイル等の情報を B や D に送る。その一方で A は、B や D からメイル等の情報を受ける。グループウェア等のインターラクティブシステムでは、ユーザ間で情報がリアルタイムに受け渡される必要があり、A が B にメイルを送った場合、それが B に即時に通知されなければならない。そのため、クライアントアプリケーションがサーバアプリケーションに処理を依頼するだけでなく、サーバアプリケーションがクライアントアプリケーションに処理を依頼できなければならない。

3. 現状システムの問題点

3.1 WWW の問題点

WWW はワールドワイドな情報検索を可能にする仕組みである。アクセス相手を指定する URL は、ワールドワイドで動的に追加 / 削除されるホームページを簡単に指定することができる。WWW は HTML 等で記述されたホームページへのアクセスという単純な操作を基本としている。そのため CGI(Common Gateway Interface) 等により、アクセス先でアプリケーションを起動し、その結果を受け取ることもできるが、一般的な C/S 型アプリケーションのような柔軟な協調動作は効率的に実行できない。

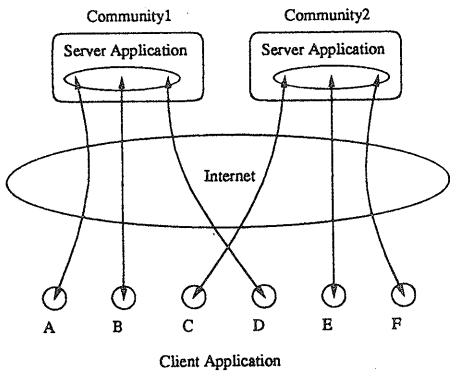


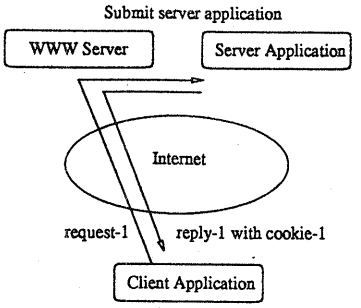
図 1 想定するアプリケーションモデル
Fig. 1 Supposed Application Model

これは WWW で使用されている HTTP が、1 リクエスト /1 リプライを基本としているためである。HTTP は複数のリクエスト / リプライを繰り返すような処理には適さない。現状のシステムでは、このような問題に対応するために、"クッキー"と呼ばれる手法を用いている。

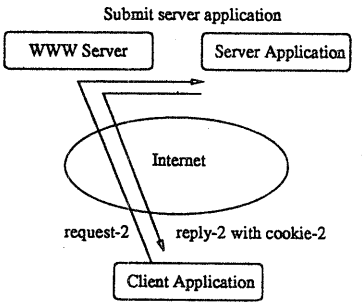
クッキーとは、WWW ブラウザとサーバアプリケーションの間で受け渡されるデータを示し、これを用いることで、いわゆるセッション管理を行う。図 2は、WWW と CGI におけるクッキーを用いたセッション管理を示す図である。(a) では、クライアントアプリケーションがサーバアプリケーションを CGI として起動している。サーバアプリケーションは処理結果をクライアントアプリケーションに戻す。この時、サーバアプリケーションが、この処理の続きを実行するための情報を、cookie-1 によってクライアントアプリケーションに渡す。(b) では、クライアントアプリケーションが、この処理の続きをサーバアプリケーションに依頼している。サーバアプリケーションは CGI として新たにプロセス生成されるが、クライアントアプリケーションから渡された情報 cookie-1 によって、(a) の処理の続きを行える。

このクッキーという手法により、WWW と CGI でも、ある程度のセッション管理が可能となる。しかし、これで従来の C/S 型アプリケーションと同等の柔軟な処理を行えるわけではない。また、CGI ではアプリケーションをリクエストを受け取る毎に生成するので、オーバヘッドも大きい。しかも、クッキーとして渡される情報だけを用いてセッション管理を行うので、プログラム開発が容易でないし、機能的な制約も受ける。

さらに WWW では HTTP を WWW サーバに送る



(a) First request from client application to server application



(b) Second request from client application to server application

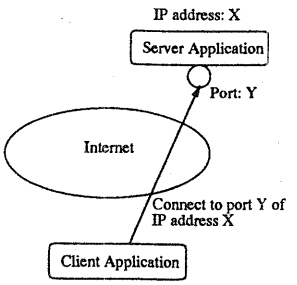
図2 WWW+CGIの動作
Fig. 2 Behaviour of WWW with CGI

時にTCP/IPのコネクションを張る。そのため、図2に示すように、リクエストは常に、WWWサーバを経由して送られる。サーバアプリケーションをWWWサーバとは異なるノードで起動したとしても、リプライは必ずWWWサーバを経由して戻す必要があり、WWWサーバがボトルネックとなり易い。

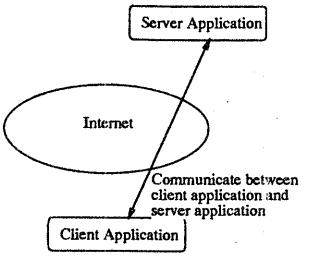
その一方、WWWでは全ての通信にHTTPを用いているので、ファイアウォールを設定し、HTTP以外のプロトコルは通さないようにすることで、サーバ側のセキュリティを確保することができる。

以上より、WWWの問題点をまとめると、以下のようになる。

(1) クライアントアプリケーションとサーバアプリケーションの柔軟な協調動作が効率的に実行できない。
(2) リクエストを受け取る毎にプロセスを生成するのでオーバヘッドが大きい。
(3) 全てのリクエスト/リプライがWWWサーバを経由するためボトルネックを生じ易い。



(a) Connection from client application to server application



(b) Communication between client application and server application

図3 一般的なC/S型アプリケーションの動作
Fig. 3 Behaviour of usual C/S type application

3.2 C/S型アプリケーションの問題点

C/S型アプリケーションは、ユーザが直接アクセスするフロントエンド側と、データ処理を行うバックエンド側を、ネットワークを介して接続する場合の基本的な方法である。そのため、多様な処理形態に柔軟に対応することができる。しかし、その反面、プログラミングは容易でない。また、一般にC/S型アプリケーションは、URLのようにワールドワイドで動的に追加/削除されるアプリケーションを簡単に指定する方法を持たない。そのため、例えばLDAP等のディレクトリサービスによって、アプリケーションが実行されるノードのアドレスとポート番号を管理しなければならない。

図3は、一般的なC/S型アプリケーションの動作を示す図である。(a)では、クライアントアプリケーションが、サーバアプリケーションの実行されるノードのアドレスとポート番号を指定して、コネクションを確立している。(b)では、クライアントアプリケーションとサーバアプリケーションが、(a)で確立されたコネクションを通して、協調動作している。

また、クライアントアプリケーションとサーバアプリケーションの間の通信プロトコルを、WWWのようにHTTPに限定することができないため、ファイア

ウォールでサーバ側のセキュリティを確保するためには、接続先を限定するしか方法がなかった。そのためインターネットで、任意のユーザからアクセスされる場合は、サーバ側のセキュリティを確保する方法がない。

その一方で、サーバアプリケーションが実行される計算機を分散させれば、クライアントアプリケーションとサーバアプリケーションの間の通信路が分散される。そのためWWWのように全ての通信が、WWWサーバを経由し、それがボトルネックになるようなことはない。

また、一旦コネクションを張ったら、全ての処理が完了するまでサーバプロセスは生存し続けるので、リクエストを受ける毎にプロセスを生成する必要はない。

以上、一般的なC/S型アプリケーションの問題点をまとめると、以下のようになる。

(1) 多様な処理形態に柔軟に対応できる反面、プログラミングが容易でない。

(2) ワールドワイドで動的に追加/削除されるアプリケーションを簡単に指定することができない。

(3) インターネットで任意のユーザからアクセスされるのにサーバ側のセキュリティが確保できない。

4. ワールドワイドで柔軟なアプリケーション環境の提供

本論文では、WWWのようにワールドワイドで、しかも多様なインターネットの利用要求に柔軟に対応可能なアプリケーションの分散処理環境について述べる。具体的には、ワールドワイドで使用されるグループウェア等のインターラクティブシステムを想定し、WWWおよび一般的なC/S型アプリケーションの問題点を解決するために、以下の特徴を持ったHTTPコネクション型RPCを提案する。

- URLによるサーバアプリケーション指定
- サーバからクライアントへのコネクション
- 同期/非同期RPC

HTTPコネクション型RPCの基本的な動作は、以下のようになる(図4)。

(1) 初期化関数http_rpc_init()を実行し、サーバアプリケーションを起動する。この時、サーバアプリケーションをURLで指定する(a)。

(2) 起動されたサーバアプリケーションは、クライアントアプリケーションが用意しているポートに対してコネクションを張る(b)。

(3) 一旦、サーバアプリケーションからクライアントアプリケーションに対してコネクションが張られたら、以降の処理は全てこのコネクションを利用して、同期/非同期RPCとして実行する(c)。

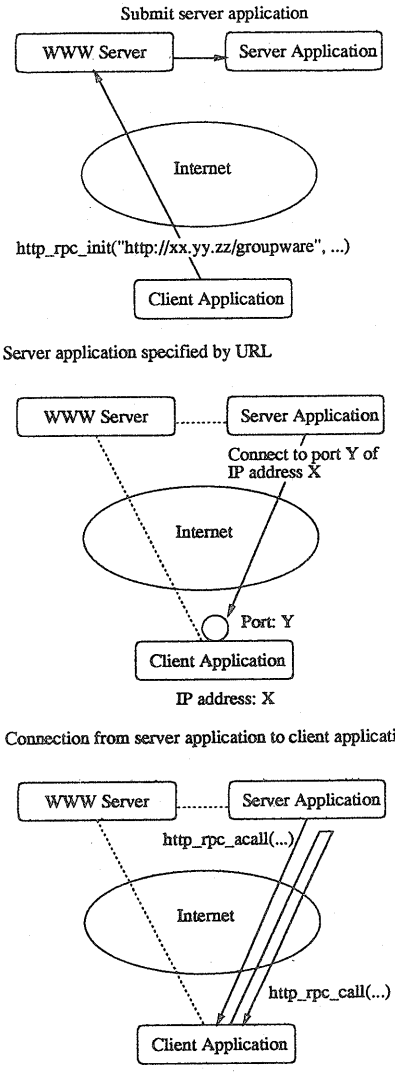


図4 HTTPコネクション型RPCの動作
Fig. 4 Behaviour of HTTP connection based RPC

以降、これらの機能について、詳細に述べる。

4.1 URLによるサーバアプリケーション指定

HTTPコネクション型RPCでは、URLでサーバアプリケーションを指定する。HTTPコネクション型RPCで提供する同期/非同期RPCの実行に先立ち、クライアントアプリケーションは初期化関数http_rpc_init()を実行する。この初期化関数の引数とし

て、サーバアプリケーションを指定するURLを渡す。URLは、アプリケーションを起動するノードのWWWサーバとアプリケーション自身を名前で指定する。アプリケーションがワールドワイドで動的に追加/削除されても、それに柔軟に対応することができる。

図4(a)は、クライアントアプリケーションが、http_rpc_init()でサーバアプリケーションを指定する様子を示す図である。http_rpc_init()の最初の引数であるURLによって、サーバアプリケーションを指定する。正確にはサーバアプリケーションを起動するWWWサーバと、そこで起動すべきアプリケーションの名前である。ただし、一旦サーバアプリケーションが起動されると、それ以降の処理は次節で説明するサーバアプリケーションからクライアントアプリケーションに張られたコネクションを介して行われるので、WWWサーバがボトルネックにはならない。

このURLによるサーバアプリケーション指定機能によって、一般的なC/S型アプリケーションの問題点の(2)が解決される。

4.2 サーバからクライアントへのコネクション

クライアントアプリケーションが、HTTPコネクション型RPCの初期化関数http_rpc_init()を実行することで、クライアントアプリケーションとサーバアプリケーションの間に、TCP/IPコネクションを確立する。HTTPコネクション型RPCでは、最初にサーバアプリケーションを指定するためだけにクライアントアプリケーションとWWWサーバの間でHTTPを利用する。それ以降のRPCの実行は、クライアントアプリケーションとサーバアプリケーションの間で直接接続された経路を用い、WWWサーバを経由しない。

このRPCはHTTPではなく、TCP/IPで実装されているため、ファイアウォールによるサーバ側のセキュリティ確保は、HTTPのみに限定するという方法をとれない。しかし、このRPCのためのコネクションは、サーバアプリケーションからクライアントアプリケーションに対して張るので、サーバ側からクライアント側に対して張ったコネクションのみを通し、逆のコネクションは通さないようにファイアウォールを設定することで、サーバ側のセキュリティ確保が可能となる。

サーバアプリケーションからクライアントアプリケーションにコネクションを張るため、クライアントアプリケーションが実行するRPCの初期化関数では、ソケットを作成し、そのソケットのポート番号をIPアドレスと共にサーバアプリケーションに渡す。サーバアプリケーションは受け取ったIPアドレスの受け取ったポート番号に対してコネクションを張る。

図4(b)は、サーバアプリケーションからクライアントアプリケーションに、コネクションを張る様子を示す図である。http_rpc_init()実行時に、そのHTTPリクエストを受け取ったWWWサーバは、指定されたサーバアプリケーションが起動されていなければ、それを起動する。この後WWWサーバは、サーバアプリケーションに対し、クライアントアプリケーションから渡されたクライアントのIPアドレスとポート番号を伝える。これによりサーバアプリケーションは受け取ったIPアドレスの受け取ったポート番号に対してコネクションを張る。

このサーバからクライアントへのコネクション機能によって、WWWの問題点の(2)と(3)、一般的なC/S型アプリケーションの問題点の(3)が解決される。

4.3 同期/非同期RPC

HTTPコネクション型RPCは、プログラミングの容易さを考慮して、クライアントアプリケーションからサーバアプリケーションへのRPCとして実現している。これを同期RPCと呼ぶ。また、インターラクティブなアプリケーションモデルを想定しているので、同期RPCに加えて、サーバアプリケーションからクライアントアプリケーションに対して非同期に処理を依頼するための非同期RPCを提供する。非同期RPCは、グループウェア等で他のクライアントから、メイル等の情報が転送される場合等に用いる。

図4(c)は、クライアントアプリケーションが実行する同期RPCであるhttp_rpc_call()と、サーバアプリケーションが実行する非同期RPCであるhttp_rpc_acall()の様子を示す図である。

この同期/非同期RPC機能によってで、WWWの問題点の(1)、一般的なC/S型アプリケーションの問題点の(1)が解決され、合わせてWWWおよび一般的なC/S型アプリケーションの全ての問題点が解決される。

5. プログラミング環境

HTTPコネクション型RPCでは、クライアントアプリケーション用に、以下のAPIを提供している。

- http_rpc_regist()
  サーバアプリケーションから非同期実行される関数を登録する。
- http_rpc_init()
  サーバアプリケーションと接続する。
- http_rpc_call()
  サーバアプリケーションの関数を同期実行する。
- http_rpc_acall_invoked()
  サーバアプリケーションからの非同期実行要求を処

理する。

- htp_rpc_close()
  サーバアプリケーションとの接続を解消する。

また、サーバアプリケーション用には、以下のAPIを提供している。

- http_rpc_regist()
  クライアントアプリケーションから同期実行される関数を登録する。
- http_rpc_connect()
  クライアントアプリケーションの接続要求に従い接続する。
- http_rpc_acall()
  クライアントアプリケーションの関数を非同期実行する。
- http_rpc_call_invoked()
  クライアントアプリケーションからの同期実行要求を処理する。
- http_rpc_close()
  クライアントアプリケーションとの接続を解消する。

6. 関連研究

HTTPを用いたWWWサーバとしては、Apache[2]が広く知られている。WWWはHTTPをベースとするため、情報検索以外の複雑な処理には適さない。HTTPコネクション型RPCはWWWのようなグローバルなサービス性と、C/S型アプリケーション本来の柔軟性を兼ね備えた、インターネット上の新しいアプリケーションのサービス形態を提供するものである。

分散オブジェクト技術として、CORBA[3]が広く知られている。CORBAでは独自のネーミングサービスを持っている。しかし、これはWWWのURLのように、ワールドワイドで動的に追加/削除されるアプリケーションを簡単に指定することはできない。

インターネットレベルで、グローバルなアプリケーションサービスを提供するグローバルコンピューティングシステムとして、Ninf[4]、Globus[5]、Legion[6]がある。グローバルコンピューティングシステムは、数値計算ライブラリ等の特定サービスを、ワールドワイドで提供する。これらのシステムは、特定サービスを対象とするため、アプリケーションがワールドワイドで動的に追加/削除されるような場合を想定する必要はない。また、インターラクティブなアプリケーションモデルには適さない。

7. むすび

今後、ワールドワイドで利用される新しいアプリケーションとして、グループウェア等のインターラクティブなアプリケーションモデルが重要になると考え、HTTPコネクション型RPCという方式を提案した。このRPCは、(1)URLによるサーバアプリケーション指定、(2)サーバからクライアントへのコネクション、(3)同期/非同期RPCが可能であるという特徴を持つ。そして、WWWや一般的なC/S型アプリケーションでは困難だったインターラクティブなアプリケーションモデルのためのワールドワイドで柔軟な処理環境を提供する。今後、提案したHTTPコネクション型RPCを評価していく。

参考文献

1) T. Howes, M. Smith 著, 松島栄樹, 岡薫訳, "LDAP インターネット ディレクトリ アプリケーション プログラミング," プレンティスホール出版, 1997年11月.
2) B. Laurie, P. Laurie, "Apache: The Definitive Guide," O'Reilly & Associates, Inc., February 1999.
3) 小野沢博文著, "分散オブジェクト指向技術CORBA," ソフト・リサーチ・センター, April 1996.
4) 小川宏高, 松岡聡, 中田秀基, 佐藤三久, 関口智嗣, "分散メモリ計算機用Ninf APIの実現に向けて," 情報処理学会研究報告 96-HPC-81, 1996年8月.
5) I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastracture Toolkit," International Journal of Supercomputer Applications, 11(2), pp. 115-128, 1997.
6) A. Grimshaw, W. Wolf, "Legion - a view from 50,000 feet," Proceedings of 5th IEEE Symp. on High Performance Computing, pp. 89–99, 1996.