Javaの Socket 通信から HTTP 通信へのソフトウェアの移行技術に関する一提案 *

6Q−6

中 雅史　奥村 友一　小林 要†
金沢工業大学§

1 はじめに

Javaアプリケーションにおいて，Socketクラスを用いたソフトウェアはインターネット上でサービスするとファイアーウォールを通過しない場合がある．Socket クラスを用いたソフトウェアの通信を HTTP に移行することで，ファイアーウォールを通過させ，サービスを拡大させることができる．

本研究は，Java の Socket クラスを用いる通信から HttpServlet[1]クラスによる通信へソフトウェアを移行する技術を提案する．本技術は Socket 通信と HTTP 通信の状態遷移図を対応付ける方法を示し，その対応関係に基づいてソースプログラムを移行する方法である．本方法により，実際の例で実験を行い移行可能であることを確認した．

2 状態遷移図での対応付け

図1は Socket クラスを用いた通信に関する状態遷移図である．図1の状態の受信と送信の定義を以下に示す．

受信　：受信命令から次の送信命令直前までである．
　　送信命令が無い場合はプログラムの最後までが受信となる．

送信　：送信命令から次の受信命令直前までである．
　　受信命令が無い場合はプログラムの最後までが送信となる．

図 1 において，①～③の矢印は，1リクエスト1レスポンス型の HTTP の通信では存在しない矢印である．Socket クラスを用いた通信を HTTP の通信に移行させるには，①～③の矢印に対応する状態の遷移をもつ状態遷移図に移行させる．

①について説明する．HTTP の通信では，クライアントのリクエストに対し，サーバは何らかのレスポンスを返さなくてはいけない．よって，①への対策には，受信した後に送信させるようにする．送信するデータはダミーデータとして扱い，ダミー送信という状態を付け加える．ダミー送信の状態に受信から矢印を引き，ダミー送信からプログラムの開始 or 終了の状態に矢印を引くことによって①の矢印は図2の①'になる．

②について説明する．HTTP の通信ではクライアントのリクエストがないとサーバはレスポンスできない．よって，②への対策には，クライアントにダミーを送信させ，サーバはダミーを受信させるようにする．状態遷移図にはダミー受信という状態を付け加える．引く矢印はプログラム開始 or 終了からダミー受信，ダミー受信から送信である(図2の②')．

③について説明する．HTTP はいったんクライアントにレスポンスを返すとコネクションが切れてしまい，引き続いて，特定のクライアントからのリクエストを受けることができ

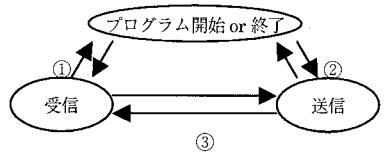

図 1: Socket クラスを用いた通信の状態遷移図

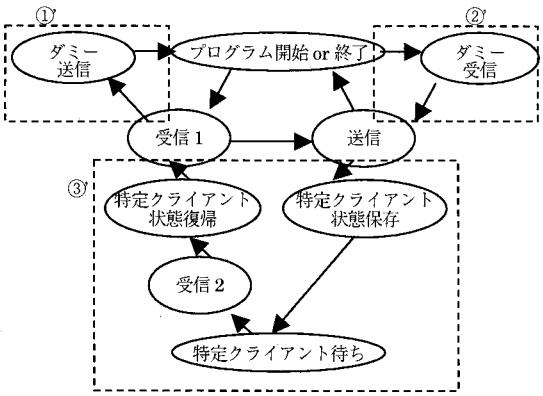

図 2: Socket クラスを用いた通信を，HTTP を用いた通信へ移行させたときの状態遷移図
（例外処理については割愛してある）

---

* A proposal on Software Migration from Socket Class to HttpServlet Class communication in Java

† Masashi Naka, Yuichi Okumura, Kaname Kobayashi

§ Kanazawa Institute of Technology

ない．よって，送信から受信への中間状態を作る．送信から受信への中間状態に入るには，送信の後に特定のクライアントに対しての情報をすべて保存する．

この状態から出るときは，リクエストを出してきたクライアントがどのクライアントであるかを特定する．図2ではクライアントを特定するために受信2状態を付け加える．クライアントを特定した後，特定したクライアントに対しての情報を復帰する．情報を復帰した後で，本来するべき受信を行う（図2の③'）．

以上の方法で，①～③の矢印が HTTP の通信用に変換し，変換したことによってサーバは Socket クラスを用いた通信から HTTP での通信に移行できる．

クライアントについては，サーバがダミー送信するときに対してのダミー受信用のモジュール，サーバがダミー受信するときに対してのダミー送信用のモジュールが必要になってくる．さらに，サーバにクライアントを判別させるための情報を送信するモジュールも必要になる．以上のようにすれば，クライアントの Socket クラスを用いた通信から HTTP での通信への移行が可能となる．

3 ソースコードでの対応付け

Socket クラスを用いた通信から HTTP の通信に移行させるために，Socket クラスを用いた通信を行うソースコードから図2の状態遷移図に対応したソースコードに変換することを考える．

前の項で述べた受信と送信の定義から，特定のクライアントの状態を保存する箇所は送信の直後である．保存する対象は，(1)次に実行する部分はどこからなのかを判別する情報，(2)使用している変数やオブジェクトのすべてである．クライアントを特定することと，特定のクライアントの状態を復帰させる箇所は受信の直前である．サーバの受信でプログラムが終了する場合は，受信の直後にサーバがダミー送信を行い，クライアントが送信の直後にダミー受信を行う．プログラムがサーバの送信で始まる場合は，サーバは送信の直前にダミー受信を行い，クライアントは受信の直前にダミー送信を行う．

4 実験

クライアントがアプレット，サーバがアプリケーションで構成されるシステムを我々が提案する方法を用いて移行実験を行った．

移行するシステムのアプレット・アプリケーション間の通信は，ほとんどが HTTP の 1 リクエスト 1 レスポンス型であった．このため，システムのモジュールの実行部分の多くが移行前のシステムのソースコードを流用することができた．さらに，ダミー送信やダミー受信を必要とする個所がなかったため，ダミー送信またはダミー受信用のモジュールを作成せずに移行が可能であった．移行させたシステムでは図 2 の受信 2 の部分は HttpSession を利用した．HttpSession を利用したことで，移行後のソースコードに新たな受信命令を追加する必要がなく，さらに移行を単純化することができた．

また，システムの通信が 1 リクエスト 1 レスポンスで構成されるならば，それぞれをサーブレット化することができる．状態遷移図からソースコードに対応付ける方法の欠点は，状態を判定するためにサーバ側のソースコードが肥大化することである．移行する際に，1リクエスト1レスポンスごとにサーブレット化する方法をあわせて利用することで，肥大化の欠点を軽減した．実際の移行前と移行後のソースコードの行数を以下に示す．

表 1: 移行によるソースコードの行数の変化

	クライアント	サーバ
移行前	3200 行	1292 行
移行後	3286 行	1186 行

5 おわりに

我々は機械的に Socket クラスを用いた通信を HTTP で通信できるようにする方法を提案した．

今回の実験では我々が提案した方法を実際に適用した例がアプレット・アプリケーション間の通信をしているプログラムであった．もし，アプリケーション・アプリケーション間の通信になっている場合は，サーバの部分はサーバの移行作業を対応させ，クライアントの部分はクライアントの移行作業を対応させることによって移行できると考える．

本稿では，特定クライアントの状態を保存するときに状態のすべてを保存と述べてあるが，実際はもっと効率的に状態を保存できるはずである．効率的に特定クライアントの状態を保存する方法については今後の課題である．

参考文献

[1]: 原田洋子: Java Servlet 最新サーバ・プログラミング，(株)秀和システム，(1999)