Simultaneous TCP Openを用いた二つのエンドポイント間の直接接続に関する研究

松田駿一† 中山泰一†
†電気通信大学情報工学科

1 はじめに

TCPにおける特殊な接続手法として，Simultaneous TCP Openがある．この手法の特徴は，通信する二つのノードが，お互いに向けて **connect()** をすることでSYNパケットを送信し合い，接続を確立できることである．この手法は，通常は **bind()** を用いてのポート割り当てが必要であるが，クライアントに特化したAdobe Flash[4]などの **bind()** を持たない環境もある．

本研究では，エンドポイントの使用ポート番号を予測し，情報をやり取りする機能を持つ仲介サーバを用意し，Simultaneous TCP Openを利用して，**listen()** や **bind()** といった，サーバとして動作するための機能を持たない処理系において，二つのエンドポイント間の直接接続を実現するための手法を提案してきた[3]．

本稿では，今まで本手法の適用が不可能であったOSに対して，ローカルにプロキシサーバを用意することで適用を可能とした．その上で，Adobe FlashやJNEXTといった複数の処理系でクライアントプログラムを作成し，複数のOSやネットワーク環境で検証を行った．

2 関連研究

2.1 Simultaneous TCP Open

TCPにおける接続では，ほとんどの場合は接続する二つのノードがクライアントとサーバに分かれる3-way handshakeが用いられる．一方，特殊な接続手法として，Simultaneous TCP Open[1]が存在する．この手法では二つのノードが，お互いに仲介サーバなどの合図で同時に **connect()** し合うことで接続を確立できる．

2.2 JNEXT

JNEXT(JavaScript Native Extensions)[5]は，主要OSにおけるWebブラウザのプラグインとして動作するJavaScriptの拡張モジュールである．JNEXTを利用すると，JavaScriptを使ってソケットによるTCP通信が可能となる．しかし，JNEXTは **listen()** や **bind()** といった機能を持たず，サーバになることも，任意のポートをソケットへ割り当てることもできない．

本研究では，クライアント機能に特化したAdobe FlashやJNEXTといった環境において，**connect()** のみによる接続の確立を行う．

3 設計

3.1 システムの構成

本システムは，エンドポイントの情報の調査や接続の合図を行う仲介サーバSと，クライアントプログラムが動作し，直接接続を行う二つのピアA，Bからなる（図1(i)）．これに加え，状況に応じてローカルプロキシサーバを各ピアのマシン上へ設置する．

3.2 接続手順

本手法による接続手順は，図1(iii)で示すとおり，まず，ピアAとピアBが仲介サーバSへ接続を繰り返す．これによって，仲介サーバSはそれぞれのピアのポート割り当て規則を調べ，次に使用されると予測されるポート番号や，相手のアドレスといった情報をそれぞれのピアへと伝える．そして各ピアは，仲介サーバSの合図で同時に **connect()** し合い，Simultaneous TCP Openを成立させ，接続を確立する．もし失敗したら再び手順を繰り返す．

3.3 ローカルプロキシサーバ

次に使用されるポート番号の予測が難しいOSでは，本手法の適用が難しい．そこで，そのような場合に備えて，接続の中継をするプロキシサーバを，書くピアノローカルマシン上にオプションとして設置する．

プロキシサーバは，3.2節の手順をそのまま中継する．このプロキシサーバは，外部との通信に使うソケットに対し，**bind()** を試用して明示的にポート番号を割り当てる．これによって，次に使用されるポート番号

A Study of Connection Between Two End-Points Using Simultaneous TCP Open.
Shunichi MATSUDA†, Yasuichi NAKAYAMA†
†Department of Computer Science, The University of Electro-Communications, 182-8585, Tokyo, Japan
matsuda-s@igo.cs.uec.ac.jp, yasu@cs.uec.ac.jp

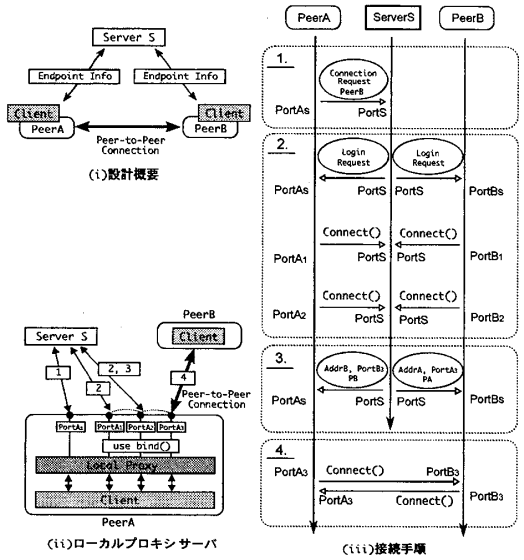

図 1: 本機構の設計

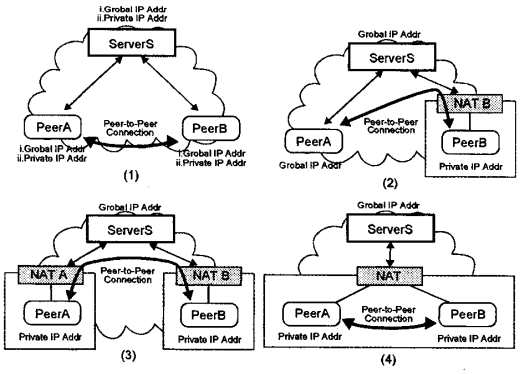

図 2: 評価ネットワーク環境

を外部から予測できるようになる（図 1(ii)）.

4 実装と評価

仲介サーバを C#で，クライアントプログラムを ActionScript3.0 を使用した Flash と，JNEXT を使用した JavaScript で，ローカルプロキシを Python でそれぞれ実装し，評価を行った.

4.1 複数環境における評価

複数の評価環境を用意し，接続確立の評価を行った.

複数の OS 間の検証では，Windows 系の OS が最も接続しやすく，Linux 2.6 ではローカルプロキシを使用することで接続を確立できることがわかった（表 1）.

図 2 に示した，複数ネットワークでの検証では，今回使用した NAT においては，(4) の例以外では NAT を越えて接続を確立することができた．(4) の例ではヘアピン動作 [2] をサポートする NAT ならば，接続の確立が可能であると予想される.

表 1: 各種 OS 間での相性

PeerA	PeerB	Result
Windows	Windows	◎
Windows	Mac OS X	○
Linux 2.6	Any OS	×
Linux 2.6(Local Proxy)	Windows	○
Linux 2.6(Local Proxy)	Mac OS X	△

5 まとめ

本研究では，クライアントに特化した環境において，仲介サーバを用いてポート番号を予測し，二つのノードを TCP で直接接続させる機構の設計および実装を行った．その結果，Windows 系の OS，Mac OS X，Linux2.6 での動作を確認し，またいくつかの環境で NAT トラバーサルが可能であることを確認した.

今後は，より多くの環境で実験を行い，現状では本手法の適用が不可能であるネットワークに対する対策を行っていく予定である.

参考文献

[1] B. Ford, P. Srisuresh, and D. Kegel: Peer-to-peer communication across network address translators, USENIX Annual Technical Conference, Anaheim, CA, April 2005.

[2] G. Guha, Ed., K. Biswas, B.Ford, S. Sivakumar, and P. Srisuresh: NAT behavioral requirements for TCP, RFC 5382, Internet Engineering Task Force, October 2008.

[3] 松田駿一, 中山泰一: Simultaneous TCP Open を用いた二つのエンドポイント間を直接接続させるシステムの実装と評価, 電子情報通信学会論文誌, Vol.J92-D, No.9, pp.1690-1693, 2009.

[4] Adobe - Flash Player 10, http://www.adobe.com/jp/products/flashplayer/

[5] JNEXT - JavaScript Native Extension, http://www.jnext.org/