ユーザレベルUDPを用いたモバイル環境の設計

神里 健司† 川下 啓介† 河野 真治†
†琉球大学理工学研究科情報工学専攻

概要

モバイルコンピューティングにおけるネットワークの利用は、通常のデスクトップ環境にくらべて制限が大きい。ネットワークへのアクセスポイントが頻繁に変更されること、通信媒体が不安定なことが多いことなどである。モバイルコンピューティング用の通信プロトコルはいくつか提案されている。これらのプロトコルを用いれば、通常のアプリケーションをモバイル環境でも利用できる。しかし、モバイル環境を意識しないアプリケーションではモバイルコンピューティングに完全には合致できない。そこで、モバイル用通信ライブラリ、通信プロトコル、モバイル用アプリケーションを含む総合的なモバイルコンピューティング通信環境を提案する。まず、モバイルコンピューティング用ネットワークライブラリとしてUDPに信頼性を付加し、あらたにコネクション管理APIを与えたDatagram Libraryを実装し、その評価を行い、これを利用した簡単な通信プログラムを作成した。

Implementation of Mobile Computing Environment Using User Level UDP

†Takeshi Kamizato †Shinji Kono
†Specialty of Information Engineering, University of the Ryukyus.

Abstract

Differ from at desktop computor, using network resources at mobile computor, we have some problems. One, when mobile computor moved, it may change its point-of-attachment to Internet. Another one, a communication medium may be unstable. MOBILE IP and VIP already exist for communication protocol for mobile computer, but it don't exist that protocol providing communication control API on user level. So, we implement Datagram Library, which based on UDP, adding to it reliablity and user level commuication control API.

1 始めに

モバイルコンピューティングの普及に伴い、場所に依存せずにネットワークリソースにアクセスできるよう、移動計算機用のネットワークの研究もされている。

現在のモバイル環境下における通信プロトコルは、モバイル環境というものをアプリケーションレベル以上ではできるだけ意識しないように透過的に扱おうとしている。ユーザはどこにいても同様の環境でネットワークリソースを利用しようというコンセプトである。しかし、コネクションの

切断 (中断)・再接続時にはユーザはその操作を行わなければならない。結局、ユーザがモバイル環境を意識しないことはない。つまり、モバイル環境での通信をアプリケーションレベル以上で透過的に扱うことは限界があるのである。

逆に、モバイル環境における通信はアプリケーション・ユーザレベルでの積極的な制御を行えばより自然になると考えられる部分もある。PHS の電波の途切れや予期できる通信回線の切断による中断・再接続は結局ユーザもしくはアプリケーションが行うのだから、TCP のタイムアウトに任せず直接制御すればよい。

そこでモバイル環境を意識した通信ライブラリ、アプリケーション、含み、ユーザレベルでのコネクション管理サービスを提供するモバイルシステムを提案する。

そこで、まずアプリケーションプログラミングレベルでの通信の一時中断・再開のコネクション管理 API をもつ通信ライブラリを作成し、その評価を行った。

2 ユーザレベルモバイルコンピューティングシステム

本稿で提案するモバイルシステムは以下のような要素からなる。

- 中断をサポートするモバイル用通信ライブラリ
- モバイル用アプリケーション
  - リモートエディタ
  - マルチポイントマルチサーバニュースリーダ
- delegate[?] をサーバとした Telnet、web サービスの tunneling.
- リモートエディタのサーバやマルチポイントマルチニュースリーダの中間サーバ、delegate を動かす、安定的に Ethernet で安定してネットワークに接続されているサーバマシン

本システムを利用することによって、リモートエディタ、マルチポイントマルチサーバニュースリーダ等のモバイル用アプリケーションと Web ブラウザ、Telnet をユーザレベルでのコネクション管理の上で使用することが可能になる。

本システムの概念図を図??に示す。モバイルコンピュータ上の各クライアントはサーバマシン上のサーバによってサービスを受ける。

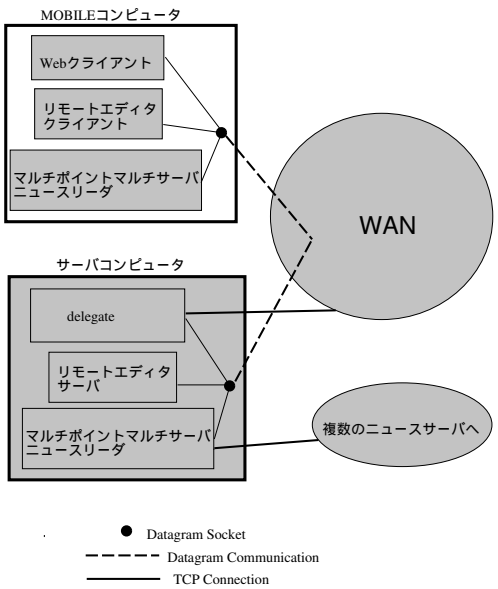

図 1: システムの構成

3 モバイル用通信ライブラリ

通信ライブラリは UDP パケットに信頼性とフロー制御を付加した Datagram Library[4] にコネクション管理 API を加えたものを使用する。
このシステムで Datagram Library を使用すると以下のような利点がある。

- Datagram Library がコネクションレスの通信であり、データパケット毎の細かいコネクション管理が可能。
- 複数の通信相手先に対して 1 つの UDP ソケットで通信を行うため、多くのソケットを必要としない。

4 モバイル用アプリケーション

4.1 リモートエディタ

リモートエディタとは、サーバ・エディタによってファイル管理を行い、クライアント・エディタ

を用いてサーバに存在するファイルに接続し編集を行うエディタである。[3]

4.2 マルチポイント・マルチサーバニュースリーダ

マルチサーバニュースリーダは複数のニュースサーバが提供するニュースを整合性を保ったまま取得することが可能なニュースリーダであり、マルチポイントニュースリーダは特定の計算機にリモートログインすることなくニュースを読み込むことができるニュースリーダである。[5] このニュースリーダはデータベースのみを保持する中間サーバを用いて行われているが、この中間サーバとクライアントニュースリーダ間の通信、ニュースサーバ間とニュースリーダ間の通信を Datagram Library を用いて行うことによってモバイル環境にマッチしたニュースリーダを作成できると考える。

4.3 delegate

delegate は TCP/IP 上の各種プロトコルの汎用のプロトコル中継システムで多機能なプロキシ機能をもつ。これを改造し、モバイルコンピュータ上のアプリケーションと Datagram Library で通信させて WEB ブラウザ等の既存のアプリケーションを使用する。

5 Datagram Library

5.1 特徴

Datagram Library は、並列検証システムにおいて各展開サーバ間の状態空間の交換用に開発された通信ライブラリであり、UDP を用いて実装されている。TCP と比較して UDP は以下のような特徴をもっている。[4]

- 理論的には TCP_STREAM より高速である。
- 通信路に明確な接続・切断状態の区切りがない。
- 信頼性を提供しない。
- プログラミングレベルでの BLOCKING、DEBLOCKING、再送を考慮しなければらない。
- 再送する分通信は遅くなる。
- スループットを優先したいのか、応答速度を優先したいのか識別できない。(TCP は SOCKET オプションで指定可能)

基本的にコネクションの確立、信頼性の付加に余計なパケットを使わない分、TCP Stream よりは高速である。また、通信路に明確な接続・切断状態の区切りがないことは、接続と切断を UDP を用いたプログラミングレベルで定義出来るということになる。TCP を使用したプログラミングにおいては、通信路の CONNECT・LISTEN 等の状態を TCP ソケット自体がもっており、その状態は TCP の中で定義されている。ユーザのプログラミングレベルではソケットの状態の定義はできないため一度切断してしまった TCPstream ソケットは Close され、通信を再開するには再度 OPEN して CONNECT する手順を踏むことになる。

Datagram Library では、UDP に信頼性を提供し、かつソケットの状態に新たに DISCONNECT に加える。

Datagram Library の特徴を以下に示す。

- 通信先毎にパケットバッファ、BLOKING・DEBLOKING QUEUE と、CONNECT/DISCONNECT の状態をもつ。
- 出力の多重化 (いわゆる SLIDING WINDOW) を USER LEVEL で行う。
- 通信先のアドレスを ID とセットで登録する必要がある。
- 再送はタイミングはプログラミングレベルで指定する。

Datagram Library は一つのソケットで複数の通信相手との通信が可能である。ソケットには付属するデータ構造には通信相手先のホスト名、ポート番号に付随して、再送のための送受信するパケットバッファ、BLOCKING/DEBLOCKING QUEUE をもつ。UDP は TCP にくらべて高速であるが、Datagram Library では再送のルーチンを自前でもっており、通信の手間は TCP と同等である。したがって、Datagram Library を用いた通信は TCP に比べて速くはない。むしろカーネルが再送のバッファを持たない分、速度は遅くなる。

Datagram Library で経由されるデータは strem 型のように感じられるが、実際の通信はパケットで行われる。TCP も下位のレイヤではパケットでの通信をしているわけだが、Datagram Library ではパケットの操作を行える点が異なる。

TCP が sliding window によるフロー制御の部分を Kernel で行っているのに対して、Datagram Library は再送のバッファをライブラリレベルで持ち、実際の再送のタイミングはライブラリを使ったプログラム、もしくはユーザ自身がおこなうことになる。再送をプログラミングレベルで行うのは非常にプログラミングを複雑にする。これはライブラリのレベルで行うべきものであるが、簡単ではない。通信先のアドレスを登録する必要があるのはあまり実用的でないように思われる。全ての通信相手先で、アドレスと ID のセットは矛盾なく登録されていなければならない。考えられる解決作としては、まず通信の当初において、ＴＣＰでまずアドレスと ID のセットからなるアドレスデータベースを交換してから、Datagram ソケットを開く方法、アドレス管理のサーバをつくり定期的にアドレスデータベースを更新する方法もある。

5.2 実装

Datagram Library は通信相手先の IP アドレスと ID のセットは全て登録する必要がある。アドレス毎に block、Deblocking の queue を Double linked list として持っている。(2)

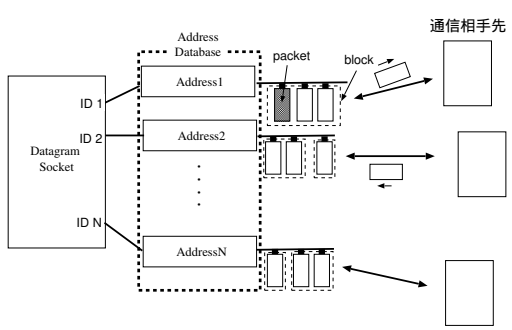

図 2: Datagram 通信ライブラリの構成

5.3 connect/disconnect

datagram library では全ての通信相手先のアドレスは登録されている。connect、disconnect の実装は、アドレス毎に connect/disconnect の状態を持たせ、アドレス毎に状態を遷移させることによって行う。アドレス毎に起こる状態遷移を図 3 に示す。

disconnect 中に発生した write のデータは再送の queue に蓄えられ、connect 時に再送されるためデータのロスはない、read の要求はそのまま block する。

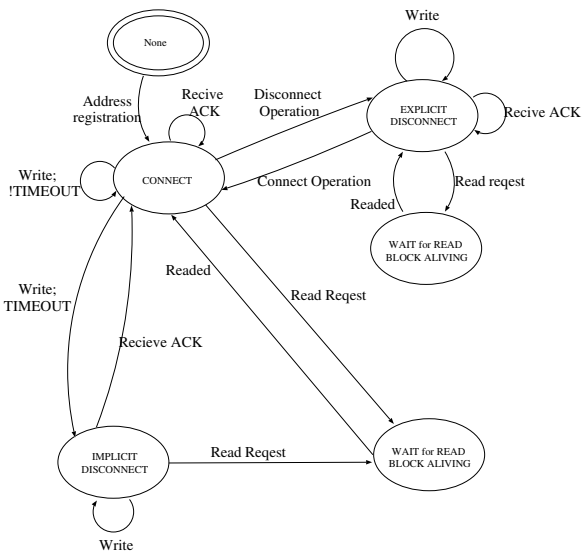

図 3: STATE TRANSITION OF DATAGRAM ADDRESS

Disconnect には 2 通りの場合が考えられる。ユーザもしくはアプリケーションが明示的に Disconnect する場合、もしくは read、write の要求の timeout により暗黙に Disconnect に移行する場合である。ユーザもしくはアプリケーションが明示的に Disconnect する場合、一度 Disconnect オペレーションにより Disconnect 状態に移行したアドレスは Connect されるまで状態遷移しないのが自然である。明示的な、Explicit Disconnect オペレーションの仕組みを図??に示す。

6 Datagram Library の評価

次の表は実装した Datagram Library と TCP/IP、UDP との通信速度を比較した表で

ある。この実験では、約 50byte ごとにそれぞれのソケットに write を行った。1MB の転送では UDP で大量のパケットロスが起こり測定不能となったので No Data となっている。

	10kbyte	100kbyte	1Mbyte
TCP	120.1(msec)	199.3(msec)	1068.7(msec)
UDP	14(msec)	130(msec)	No Data
Datagram	2(sec)	45(sec)	450(sec)

この結果では Datagram Library がかなり遅くなっているが、これはユーザレベルでの bloking・deblocking の際の block サイズが小さすぎたためであると思われる。TCP がカーネルの内部でバッファリングを行い、ある程度の block が溜ったところでパケットにして送信するのに対し、Datagram Library は write の block ごとのパケットで転送する。いわば Datagram Library はプログラミングのレベルで block の大きさを指定できるため、この例では大量のパケットによる通信が起こったため結果的に遅くなっていると考えられる。これを受けて、データ送信のプログラムで block のサイズを適当に大きくして送信を行った。次の表は 10kbyte では約 150byte、100kbyte では約 1250Byte 毎にそれぞれのソケットに write を行った場合の実験結果である。

	10kbyte	100kbyte
TCP	10ms	94ms
UDP	9ms	84ms
Dg Lib	11ms	4421ms

明らかに早くなっている。この結果より、Datagram Library を使用した通信では block のサイズによってかなり通信速度に差がでることがわかる。

Datagram LibraryTCP と同様の bloking / deblocking、sliding window をプログラムレベルで実装している。従って通信の手間は TCP と同等であるから block のサイズが適当なものになれば、TCP と変わらない通信速度が期待できるものと思われる。

7 Telnet の tunneling

Datagram Library を用いた評価用のアプリケーションとしてまず、既存の TELNET を Datagram によって Tunneling させる中間サーバを作成した。この中間サーバを用いたシステムのモデル図を図 ??に示す。通常の TELNET は TCP によって通信を行う。このシステムでは、TELNET クライアントプログラム、TELNET がアクセスする deamon は既存のものを使用し、間に中間サーバをはさむことによって Datagram により tunneling させいる。

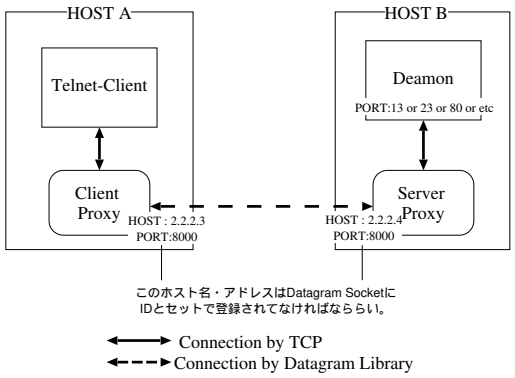

図 4: TELNET

8 まとめ

本稿では、ユーザレベルでのコネクション管理を行えるモバイルコンピューティング環境を設計し、その通信ライブラリの実装を行い、評価を行った。また、その通信ライブラリを用いて、Telnet を tunneling させてみた。

9 これからの課題

9.1 モバイル環境に特化した特殊なアプリケーションへの応用

Datagram Library は Telnet のような既存のアプリケーションに応用しても、コネクション管理 API によるユーザによる通信の中断・再開のサービスをを提供する。しかし、リモートエディタやマルチポイントマルチサーバニュースリーダ等モバイル環境に適したアプリケーションに用いることで効果を発揮すると考える。

9.2 delegate を用いた web サービスへの応用

参考文献

[1] John Bloomer 著, 森島晃年訳 “RPC プログラミング” アスキー 1995.

[2] Yutaka Sato, Electrotechnical Laboratory (ETL), AIST, MITI, Japan: DeleGate Home Page http://www.delegate.org/delegate/

[3] 河野真治: 日本ソフトウェア科学会第 16 回大会論文集, Sep, 1999

[4] 新垣将史 , 河野真治: 日本ソフトウェア科学会第 16 回大会論文集, Sep, 1999

[5] 島袋政辰 , 河野真治: マルチポイント・ニュースリーダの実装と評価, 日本ソフトウェア科学会第 15 回大会論文集, Sep, 1998