5M-06

ファイルシステムによる通信手法の簡略化

○利穂 虹希† 佐野 雅彦†
†徳島大学

1. はじめに

スマートフォンやIoT等のインターネット利用機会の増大により，新しい製品やサービスを提供するための前提知識としての通信プログラミング経験が必要不可欠となっている．しかし，従来のプログラム手法では，ソケットなどの固有の概念が多く，経験の浅い人にとっては理解が難しい．そこで本研究は，一般的なクライアント・サーバ間通信をファイルシステムとして実装することにより，ファイル操作による直感的な理解とファイルシステム以外には特別なライブラリを不要とした環境の実現を目的とする．

分散ファイルシステム[1]など，ファイルの共有を目的としたファイルシステムや，名前付きパイプ及びその拡張などで，端末内プロセスの通信に，ファイルの概念を用いた例[2]やファイルシステムを用いた例[3]は存在するが，クライアント・サーバ間通信をファイルシステムとして実装し，前述の目的のためにファイル操作の関数で操作できるようにした例はない．本稿では，提案するファイルシステムを用いたプロセス間通信プログラムと，従来のソケット通信を用いたプログラムを比較し，プログラムの作成が容易となったか，検討を行った．

2. 提案手法及び仕様

2.1. 提案手法

OSのカーネルコードを修正することなく独自のファイルシステムを実装できるソフトウェアインターフェースであるFUSE(Filesystem in Userspace)[4]を用いて，C言語で実装する．FUSEを用いることにより，FUSEをサポートされる各種Unix系OSにて実装可能となる．本稿ではこの提案するファイルシステムをcomfs (COMmunication File System)と称する．

open()，read()，write()などのファイル操作システムコール関数を用いて，このファイルシステムにアクセスすることで通信を可能とする．また，これらのシステムコールを使用する標準ライブラリ（例えばlibcなど）を用いた関数等においても同様に動作する．よって，通信のための特別なライブラリを不要とする利点がある．

2.2. 通信パラメタ指定

ソケット通信に必要な最低限のパラメタはクライアントとサーバで異なるが，クライアント側を例にすると，通信プロトコル，プロトコルファミリー，宛先ポート番号，宛先IPアドレスである（送信元ポート番号，送信元IPアドレスはOSにより自動設定される）．なお，IPアドレスについては，IPアドレスで与えられる場合と，DNS名で与えられる場合がある．これら必要最低限のパラメタをパス表記し，本提案方式によるファイルシステムにアクセスすることにより，ファイルシステムに対してパラメタを指定する．以下に概要を示す．

（1）プロトコルとプロトコルファミリーの指定

対応する通信プロトコルはTCPとUDPとした．また，プロトコルファミリーについてはIPv4とIPv6を想定する．TCPでIPv4を用いて通信を行うにはTCP4，UDPでIPv6を用いて通信を行うにはUDP6と指定することとした．これらは，提案ファイルシステムのルートディレクトリ直下に配置されるものとした．

（2）IPアドレスおよびポートの指定

IPアドレス(DNS名の場合を含む)とポート番号は"IPアドレス/ポート番号1/ポート番号2"の形式で与えるものとした（ポート番号2はOSにより自動補完のため省略可）．指定するポート番号の意味は，共通となる側のポート番号をポート番号1，通信セッション毎に異なるポート番号をポート番号2としている．

（3）その他の指定

上記以外で標準値でないパラメタを指定する場合には，上記（2）に続けてオプションを追記する方式とした．

以上から表1にパスの記述例を示す．

表1 パスの記述例

	プロトコル	宛先名/アドレス	ポート番号1	ポート番号2	オプション
サーバ	/TCP4	/cliantname	/8888	/(省略可)	/REUSEADD
クライアント	/TCO4	/servername	/8888	/(省略可)	

2.3. パスの有効期限

通信をファイルシステムとして表現するため，通信が終了するとその通信を表現するパスの有効期限は終了し，ファイルシステムから取り除かれる．これはclose()関数（明示的でない場合も含む）がコールされた場合に発生し，このパスは揮発性であり，一般的なファイルシステムと異なり永続化されない．

2.4. サーバとクライアントの識別

通常，サーバとクライアントの通信セットアッププロセスは異なるが，本手法では，ファイルオープン時のパラメタにより識別する方式としている．ここでは，一般的なファイル操作を行う関数であるfopen関数において，そ

Simplification of the communication technique by a file system,
Koki Riho and Masahiko Sano, Tokushima University.

の引数の解釈について述べる.

第一引数では，前節表に示す表記に従い提案ファイルシステム内の場所を指定する.

第二引数は, 書き込みを行う側をクライアント側, 読み込みを行う側をサーバ側とし，'w'を指定した場合はクライアントとしての動作，'r'を指定した場合はサーバとして動作するものと解釈する.

例として, マウントポイント m を comfs でマウントしたディレクトリとして，TCP で IPv4 を用いて，サーバ(servername)のポート番号(8888)に対して通信を始めるプログラム例を図１に示す.

```
//通信の開始
FILE * fp = fopen("m/TCP4/servername/8888", "w");
```

図１　fopen 関数の使用例

2.5.　その他の関数での取扱

fwrite()関数により相手に書き込み（送信），fread()関数により相手から送られてきたデータを読み出す（受信する）．通信を終了させたい場合は fclose()関数を用いる.

図２に，ファイルポインタ fp に対して，書き込み，読み込み，切断を行うプログラム例を示す.

```
//書きこむデータをバッファに保存
fwrite(senddata, sizeof(senddata), 1, fp);
//バッファに保存されたデータを fp に書き込み
fflush(fp) ;
//受信したデータの読み込み
fread(recvdata, sizeof(recvdata), 1, fp);
//通信の切断
fclose(fp);
```

図２　データ送受信例

また，open したファイルディスプリクタを指定することで，read()/write()関数も同様に扱うことができる.

write()関数の場合は，書き込むデータをバッファリングせずにそのまま送信するものとしているため，fflush()関数を使用することなく書き込みを行うことができる.

3.　有効性の調査

3.1.　プログラムの書きやすさ

C 言語による一般的な socket 通信では，sockaddr_in 構造体に，プロトコルファミリー，ポート番号を保存し，socket の生成時に使用する通信プロトコルを指定する. この IPv4 と IPv6 のプロトコルファミリーを示す変数の変数名は AF_INET，及び AF_INET6 となっており，この独自の変数名を用いて指定しなければいけない. また, TCP 通信と UDP 通信を示す変数の変数名は SOCK_STREAM 及び SOCK_DGRAM となっている. いずれも，プログラミング初学者にとっては馴染みのない表現である. また，構造体のどのメンバに設定するべきなのかも分かりづらい. しかし，本手法では TCP4 や UDP6 と書くことで設定することができる. 気をつける点は open 時に指定する path の記述順のみである.

従来のソケット通信では, connect()関数を用いてサーバと接続を行うまでに，C 言語では最低 9 行のプログラムと, 3 つの専用のライブラリをインクルードする必要があった. 本手法であれば，open()関数 1 行と，基本入出力のライブラリ stdio.h のみで通信が行え，コード量を 7 行減らすことができた.

また，Ruby においても，socket ライブラリをインクルードし，TCPSocket.open()関数を用いて接続を行うまで一つのライブラリと一行の特殊な関数を用いる必要があった. しかし，本手法では，File.open()関数一行のみで接続を開始することができる.

4.　おわりに

comfs を用いることで，従来のソケット通信よりも，簡単に通信が行えることがわかった. また，C 言語，Ruby で，open()，write()，read()，release()システムコール関数を利用しているライブラリ関数によって, 通信の確立, 書き込み, 読み込み, 通信を切断する動作を確認することができた. この手法を用いることで, ソケット通信より簡単に通信のプログラムを経験できる.

このシステムは，理解する必要のある独自の概念が少ない為, プログラミング初学者が, 通信を用いたシステムの実装を初めて行う場面などで有効である.

一方，ソケット通信のオプションを複雑に設定するシステムの実装や，通信の速度においては現状では課題がある. また, 本稿執筆時点ではセキュリティや, サーバ側として動作する場合の実装にについても検討の余地がある. 例えば, 複数クライアントを接続する際に, サーバ側ファイルシステム内でどのようなパスにするのかという課題である. 現在は OS が自動的に割り振ったポート番号によってクライアントを識別し，クライアント毎に異なるディレクトリをサーバ側 ファイルシステム内に用意するなどの対応を検討している.

参考文献

[1] 田中 英昭，重兼 史尚，小林 孝史：“P2P 技術を利用した分散ファイルシステムの実装”，2011，電子情報通信学会技術研究報告. NS，ネットワークシステム

[2] Danilo O. TanMohit Arora :“System and method for securing an inter-process communication via a named pipe.” , http://www.freepatentsonline.com/y2018/0278611.html (2019 年 1 月 7 日)

[3] Olivier Dalle : “MPCFS : a virtual FileSystem for MultiPoint Communications”, http://www-sop.inria.fr/members/Olivier.Dalle/mpcfs/ (2018 年 12 月 27 日)

[4] SZEREDI M. : “FUSE : Filesystem in Userspace”, https://github.com/libfuse/libfuse (2019 年 1 月 7 日)