通信インタフェースとオーバヘッドに関する考察

4F－3

根岸　康
日本アイ・ビー・エム(株)　東京基礎研究所

1　はじめに

近年光ファイバ等を使った高速な媒体が利用可能になっている．しかし，このような高速な媒体を利用しても既存の通信機構を使うと通信機構のオーバヘッドにより，媒体の速度を十分活用する事ができない．この原因の大きな部分を通信機構による通信データへのアクセスが占めている ([1], [2])．通信インタフェースを変えずにこのオーバヘッドを削減する方法として，通信データへのアクセスの回数を減らしたシングルコピー方式による実現が使われている ([3])．本稿ではこの方法の問題点とそれをインタフェースの変更により改善する方法について述べる．

2　シングルコピー方式

既存の通信機構の多くは，図1のように実現されている．

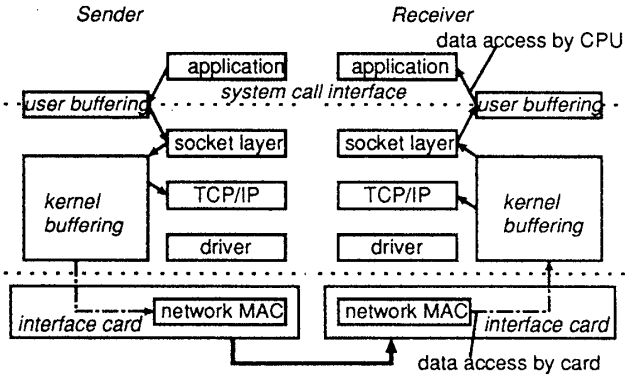

図 1: 既存の通信機構

既存の通信機構の場合，1回の送信・受信それぞれにつき4回のプロセッサによるデータアクセスが起きている．これは既存の通信機構が比較的遅い通信媒体を前提に設計されているためで，シングルコピー方式ではデータアクセスによるオーバヘッドを削減する事を最優先に通信機構を設計する (図2)．

図2のようにシングルコピー方式では通信データはCPUによりカード上のバッファから直接ユーザバッファにコピーされる．しかし，特に受信側が送信側より遅い場合送信側受信側両方のカード上のバッファが長い間占有されるので，大きなバッファが必要になる．また，バッファが足りなくなった場合は，通信データのホスト上の別のバッファへのコピー等の処理も必要になる．

The Relationship between a Communication Interface and its Overhead
Yasushi NEGISHI
IBM Research, Tokyo Research Laboratory

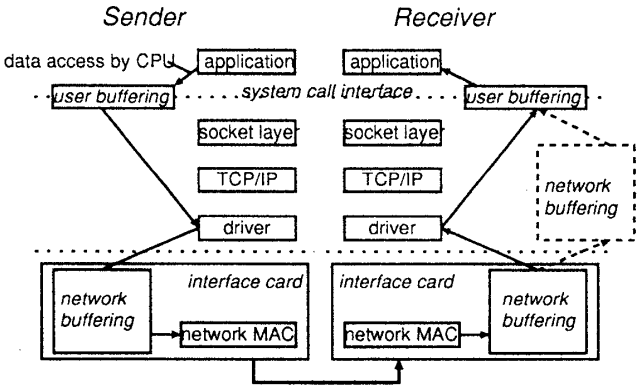

図 2: シングルコピー方式

3　malloc/free 通信インタフェース

一般に通信の際，送信側ではプロトコルのフロー制御により送信できないデータを，受信側では対応する受信システムコールがまだ来ていないデータをバッファリングする事が必要になる．シングルコピー方式ではこのバッファリングを通信カード上のバッファで行っている．

ここでは，このバッファリングをユーザバッファを用いて行う事を可能にする為のシステムコールインタフェースを提案する．

1. `sendandfree(int fd, char *buf, int len, int flags)`
   バッファ内のデータを送信し，そのバッファを解放する．
2. `recvandmalloc(int fd, char **buf, int *len, int flags)`
   適切な長さのバッファを確保し，そのバッファにデータを受信する．

これにより，送信側でシステムコール終了後も通信機構がユーザバッファを利用する事が可能になる，また受信側でも通信機構が受信システムコールの前にユーザバッファを確保してデータを転送する事が可能になる．通信カード上のバッファは，通信機構の動作中およびユーザバッファのページフォールト等の比較的短い処理の間だけ利用されるので，領域も小さく，管理の方法も比較的簡単なもので実現できる (図3)．

このインタフェースは，以下の問題点を持っている．

1. malloc/free の管理をユーザとカーネルの両方で行う必要が生じる．
2. malloc した領域しか送信する事ができずプログラムの変更が必要がある．

1は．malloc/free をシステムコールにする方法等で解決可能だが，オーバヘッドの問題もあり検討が必要である．2は，重大な問題だがRPC等より上位のプロトコルで再び互換性をとる事で改善が可能である．RPC

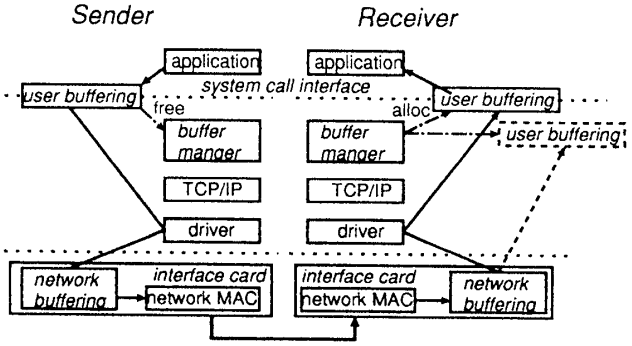

図 3: malloc/free 通信インタフェース

ライブラリがダイナミックリンクされている場合にはバイナリコンパチビリティーも保たれる．以下，このインタフェースを拡張して RPC を実現する方法について考えてみる．

4 malloc/free RPC インタフェース

上の方法により通信機構によるコピーは削減できるが，RPC で必要な複数の引数をまとめ，データ表現を変換 (パケットマーシャリング) するためのユーザによるデータコピーは残されている．これを削減する為にカーネルパケットマーシャリングという方法を利用する ([4])．これは，送受信時，引数とともにマーシャリング関数をカーネルに渡す事により，ユーザデータをマーシャリングしながら通信カード上のバッファにコピーするという方法で，malloc/free 通信インタフェースの考え方と自然に組み合わせる事ができる．このためのシステムコールインタフェースを malloc/free RPC インタフェースと呼び，以下これを使って SUN RPC と DCE RPC を実現する方法について説明する．

malloc/free RPC インタフェースでは，受信したデータを受信システムコールが来ていない時でもアンマーシャリングする事ができるよう，初期化時にマーシャリング・アンマーシャリングの為の関数をあらかじめカーネルに登録する．

SUN/DCE RPC のクライアント側は，ユーザが確保した引数の領域を使うインタフェースになっている．従って malloc/free 通信インタフェースのように送信時に通信機構が勝手にこの領域を解放する事はできない．しかし，RPC のクライアントの場合サーバからの応答があるまでユーザこの領域を利用する事はないので，この間通信機構がそのまま利用する．そのためのインタフェースは次の通り．

3. `sendandreceiveargs(int fd, int funcid, struct iovec *iov, int iovcount)`
   引数をマーシャリングして，送信し，サーバからの応答をアンマーシャリングして引数のバッファに入れる．すべての引数はこのシステムコールが終了するまで，通信機構に管理される．

RPC 呼び出しの戻り値の領域確保の方法は，SUN RPC と DCE RPC で異なり，SUN RPC の場合通信機構が，DCE RPC の場合ユーザが確保した領域を使う事になっている．この違いは，XDR ライブラリと同様に引数へのポインタに "NULL" が入っていた時は通信機構が領域を確保し，それ以外の場合はポインタの先の領域を使う事で吸収する．

一方サーバ側は，SUN RPC も DCE RPC も通信機構が引数の領域を確保するインタフェースになっている．よって，malloc/free 通信インタフェースを複数の引数を扱えるように拡張したインタフェースを用いる．`recvardmallocargs` でクライアントからの要求を受けとり，`sendandfreeargs` でクライアントに応答を返す．

1. `recvandmallocargs(int fd, int *funcid, struct iovec **iov, int *iovcount)`
   引数のバッファを確保し，そのバッファに受信したデータをアンマーシャリングして入れる．"funcid" は，RPC の関数を識別する為のものでファイルディクリプタ毎に一意である必要がある．
2. `sendandfreeargs(int fd, int funcid, struct iovec *iov, int iovcount)`
   バッファ内のデータをマーシャリングして，送信し，そのバッファを解放する．

このインタフェースを採用する事により，引数のマーシャリングを含めて 1 回のコピーで通信を実現する事が可能になる (図 4)．しかし，この方法は，ユーザの関数をカーネル中で呼び出すので安全性に問題がある．あらかじめカーネルが基本データ用の関数を用意し，この組合せをユーザが指定する等，安全性の確保の方法を検討する必要がある．

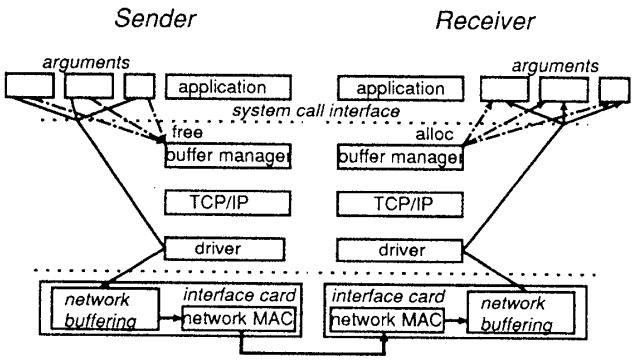

図 4: malloc/free RPC インタフェース

5 まとめ

通信のコピーによるオーバヘッドとインタフェースの関係について考察し，malloc/free 通信インタフェースを提案した．さらにこの方法を RPC に利用するための方法について述べ，効果と問題点について考察した．

文献

[1] David D. Clark, Van Jacobson, John Romkey, Howard Salwen(1989) "An Analysis of TCP Processing Overhead," IEEE Communications Magazine, June 1989, pp. 23 - 29.

[2] Jonathan Kay, Joseph Pasquale(1993) "Measurement, Analysis, and Improvement of UDP/IP Throughput for DECstation 5000," 1993 Winter USENIX - January 25-29, 1993 - San Diego, CA, pp. 249 - 258.

[3] David Banks and Michael Prudence (1993) "A High-Performance Network Architecture for a PA-RISC Workstation," IEEE Journal on selected areas in communications. VOL 11. NO. 2, February 1993, pp. 191 - 202.

[4] Chandramohan A. Thkkath, Henry M. Levy (1993) "Limits to Low-Latency Communication on High-Speed Networks," ACM Transactions on Computer Systems, Vol.11 No.2, May 1993, pp. 179 - 203.