高速ファイル転送プログラムの開発

○手島史綱、長屋貴量、松尾純一、澤昌孝、岩橋建輔、内藤茂樹、水谷文保
自然科学研究機構　分子科学研究所　技術課

概要

計算科学研究センターは、大学共同利用施設として全国の研究者（約 650 人）が利用する計算機システムを運用している。利用者の中には、かなり大きな（1TByte を超えるものも）データを扱うこともあるが、その大きなデータを利用者の PC へ転送しようとするとかなりの時間（最悪 1 日以上）が必要である。そこで、数時間で転送が終了できるような高速ファイル転送プログラムの導入を検討してきた。前回の技術研究会では、市販品やフリーソフトの通信性能評価から、UDP を高速ファイル転送に利用するのが有用であると報告した。その後、センターで検討を重ね OpenSSH を改造した高速ファイル転送プログラムを開発したので報告する。

1　開発までの経緯

自然科学研究機構　岡崎共通研究施設　計算科学研究センター（以下、当センターとする）は、大学共同利用施設として日本全国の分子科学研究分野の研究者を中心に計算環境を提供している。全国の利用者は、インターネット（SINET3）を介して計算機システムを利用しており、計算して出された結果もインターネットを介して利用者自身の PC へ転送される。当センターは、SINET3 に 10Gbps の帯域ラインで接続されており、全国の大学も 1Gbps 以上の帯域ラインで接続されている。また、機関によっては、研究室レベルでも 1Gbps ラインの使用と PC の GbE（Giga Bit Ether）の普及により 1Gbps での通信が可能な環境となってきている。しかしながら、利用者がデータ転送に使用する sftp や scp では、近距離であっても 100Mbps 程度、遠距離になれば数十 Mbps 程度の転送速度しか得ることが出来ず、GbE 環境を活用した通信が出来ていない。利用者の中には、大容量データの転送に十数時間かけているのが現状である。当センターは、2008 年 8 月頃よりこのデータ転送環境を改善すべく調査などを重ねた。一般的な改善方法として、RWIN サイズを変更する方法があり、RWIN を大きくし、一度に受け取れることの出来るデータ量を増やしことによって ACK の回数を減らすのである。しかし、この方法には大きないくつかの欠点がある。それは、OS によっては、レジストリを書き換える必要があるなど利用者側の負担が大きいことと、転送速度が 500Mbps を上回る程度しか出ないことである。このことから、根本的にプロトコルからの検討を始め「TCP マルチコネクション」か「UDP」の 2 つに絞った。「TCP マルチコネクション」の場合、全国のユーザから接続が集中する利用の当センターでは、Firewall にかなりの負荷をかけることになるので「UDP」について調査を開始した。そのなかで最初に見つけたのが Aspera 社の ascp という、UDP を利用した商用アプリケーションである。無料評価ライセンスを得ることが出来たので、九州大、東北大、名古屋大の方にご協力をいただき、通信テストを行った。その結果、１Gbp に迫る（最大 933.67Mbps）転送速度を出した。また、RTT が大きい九州大でも 800Mbps 以上を出しており、sftp、scp などの TCP ソフトウェアのような速度低下は見られなかった。この他、UDP を使用した場合、利用者側機関の Firewall において、外部からの UDP 接続を stateful なコネクションとして認識し接続できるかを簡単な UDP プログラムを作成して通信実験を行い問題なく接続、通信できることが確認できた。これらのことから、大容量データを高速に転送するには UDP が有効であると判断した。また、幸いにして岡崎は、

通信的に日本のへそに位置（北は北見工大から南は琉球大まで RTT が約 40msec で通信できる）していることから、RTT が 40msec の範囲内で大容量データを高速に転送できることもプログラムの導入に向けての前提条件とした。先に評価した ascp は非それらの点で非常に魅力的であったが、導入経費が非常に高いため UDP を利用したファイル転送プログラムが他にないか調査したところ、「Tsunami」、「Sector」、「UDT」[参考文献[4]]があった。それぞれが一長一短であった。この中で UDT が比較的短所が少なくプログラムライブラリ的に利用できることから UDT を選択した。しかし、UDT は、SSL に対応していないので OpenSSH[参考文献[5]]に UDT を組み込み、データ転送のみを UDP で行うソフトウェア hscp（Hybrid scp）を開発することにした。

詳細は、参考文献[1]、[2]、[3]を参照してほしい。

2　hscp のしくみ

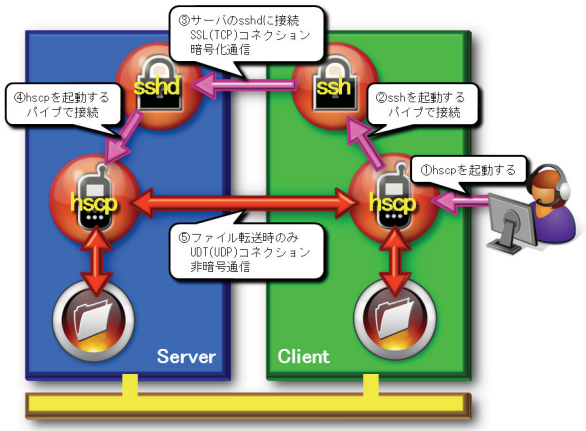


図 1. ssh 処理の流れ

に UDP を非暗号化状態で使用する。このため、データ自身に秘匿性を必要としない転送に向いている。暗号化しないこと、RTT（Round Trip Time）の影響が少ない UDP ベースのプロトコルを使用することで高速性を得ている。クライアントで hscp を利用してデータをサーバに put、get する際の処理流れを図 2 に示す。

UDT は、12 番でファイル内容を UDP で送りながら、パケットロスの数などを見て、「輻輳ウィンドウサイズ」、「パケット送信に要した時間」、「利用可能なバッファーサイズ」などを細かく調整している。hscp には、hscp.conf という設定ファイルが用意されており、この設定ファイルの中で、「MaxWinsize」、「MaxBufSize」や「RecvBufSize」を変更することにで、転送速度の改善が得られる。

hscp のコマンドオプションは scp のものがそのまま使える。今回の開発で追加したオプションとしては、“-I”でがある。これは、転送の様子

OpenSSH の scp をベースに機能追加しており、scp のデータ転送部分を UDP 通信に変えることで高速データ転送を実現している。UDP 通信には、帯域制御（排他的・独占的に使いきることはない）が実装されている UDT を利用している。

基本的な通信は、ssh によって確保された SSL コネクションを使い（ssh による処理の流れは図 1 を参照）、サイト間通信での認証等のセキュリティを確保（ユーザ認証は、ssh の代わりに rsh も利用可能である。この場合は、単なる TCP コネクションになりセキュリティは確保できない。）し、データ転送

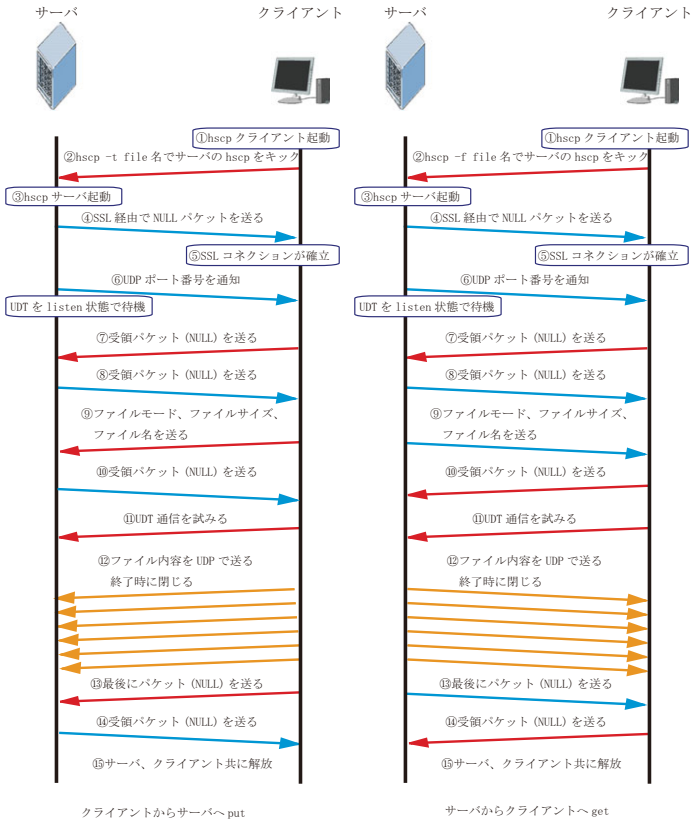


図 2.　hscp 処理の流れ

を指定秒ごとに表示するものであり、このオプションを付けた時には、次の項目が表示される（図 3.は通信例)。

表 1. hscp –I オプション表示項目

RcvR(Mb/s)	データ受け取り速度（Receiving Rate）
RTT(ms)	往復遅延時間（Round Trip Time）
CWnd	輻輳ウィンドウサイズ（Congestion window size）
PktSendPd	パケット送信に要した時間（Packet Sending Period）
SACK	送信 ACK 数（Sent Acknowledgement）
SNAK	送信 NAK 数（Sent Negative Acknowledgement）
AvRcvBfSz	利用可能バッファーサイズ（Available Receiving Buffer Size）
Recv(%)	データ受け取り率（Receive）
Etime	経過時間（Elapsed time）

```
-bash-4.0$ hscp -v
(C)[main] Read config: /usr/local/etc/hscp.conf
(C) HSCP v.0.9.10-1
usage: hscp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
            [-l limit] [-o ssh_option] [-P port] [-S program]
            [-R] [-I display_mode]
            [[user@]host1:]file1 ... [[user@]host2:]file2
-bash-4.0$
-bash-4.0$ hscp -I 1 mizutani@   .   .220.2:/work/mizutani/1G.bin .
mizutani@   .   .220.2's password:
 RcvR(Mb/s)  RTT(ms)  CWnd  PktSendPd  SACK  SNAK  AvRcvBfSz  Recv(%)   Etime
    738.495   20.259    16   900.00ms  1049     0       8191  11.079%   00:01
    923.499   20.421    16   900.00ms  1000     0       8112  21.827%   00:02
    923.862   20.346    16   900.00ms  1000     0       8191  32.586%   00:03
    924.103   20.348    16   900.00ms  1000    33       8117  43.342%   00:04
    925.794   20.197    16   900.00ms  1005    42       8118  54.122%   00:05
    926.179   20.336    16   900.00ms  1000     0       8191  64.911%   00:06
    922.865   20.356    16   900.00ms  1000     0       8110  75.647%   00:07
    918.101   20.203    16   900.00ms  1000     0       8191  86.347%   00:08
    922.856   20.233    16   900.00ms  1006     0       8191  97.089%   00:09
    249.304   19.823    16   900.00ms   276     0       8191 100.000%   00:10
1G.bin                              1073.742MB     115.788MB/s     9.273sec
-bash-4.0$
-bash-4.0$ ping -c 5   .   .220.2
PING   .   .220.2 (   .   .220.2) 56(84) bytes of data.
64 bytes from   .   .220.2: icmp_seq=1 ttl=55 time=19.5 ms
64 bytes from   .   .220.2: icmp_seq=2 ttl=55 time=19.2 ms
64 bytes from   .   .220.2: icmp_seq=3 ttl=55 time=19.2 ms
64 bytes from   .   .220.2: icmp_seq=4 ttl=55 time=19.3 ms
64 bytes from   .   .220.2: icmp_seq=5 ttl=55 time=19.2 ms

--- .   .220.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4025ms
rtt min/avg/max/mdev = 19.212/19.327/19.548/0.169 ms
-bash-4.0$
```

図 3.　hscp の-I オプション通信例

2.1　プラットホーム

現在、以下の環境でビルドまたは動作検証が済んでいる。

AIX5.3	AIX-AIX の場合、大容量転送で不具合発生例あり（要調整）
SunOS5.10	サーバとしては未検証（要調整）

Linux IA32	2GB 超えファイルの通信実績あり
Linux IA64	SGI Altix4700、Fujitsu PRIMEQUEST で検証済み
Linux x86_64	AMD でも検証済み
MacOS X	高速性については未検証（500Mbps 程度であれば実績あり）
Windows コンソール版	PuTTY [参考文献[6]]をベースに開発

（要調整）

hscp.conf で UDPRecvBufSize を標準値から減らさないと（例.768000）実行できなかった。

上記の各プラットホーム用にビルド済みのバイナリーを含むアーカイブを用意している。ダウンロードは、当センターの WEB ページ（http://ccportal.ims.ac.jp/software/hscp）か SourceForge（http://sourceforge.net/projects/hscp/）より出来る。ビルド方法、インストール方法、セットアップ方法、ライセンスの詳細も hscp の WEB ページを見てほしい。

公開以来、次のように毎月 WEB ページにアクセスがありダウンロードされている。ログを見ると、日本国内の大学や研究所、一般企業をはじめ、海外の大学より興味をいただきダウンロードしてもらっている。

表 2. 月毎の延べダウンロード件数

2009 年	6 月	7 月	8 月	9 月	10 月	11 月	12 月
延べダウンロード件数	10	23	40	16	15	24	64

3 データ転送実績

hscp と scp、sftp との通信速度の比較テストを行った時の通信例を次の通りである。

表 3. 通信速度比較（計測日時 2009 年 6 月 8 日 11:00 から 13:00）

ccfep1(AIX) –Tohoku(Linux) RTT23.502msec

プログラム名	転送時間(m:s)	転送速度(MB/s)	比較
hscp	0:23	46.7	-
scp	5:41	3.0	14.8 倍
sftp	8:37	2.0	22.4 倍

ccfep1(AIX) –Nagoya(Linux) 3.334msec

プログラム名	転送時間(m:s)	転送速度(MB/s)	比較
hscp	0:18	59.7	-
scp	1:02	16.5	3.4 倍
sftp	12:43	1.3	42.3 倍

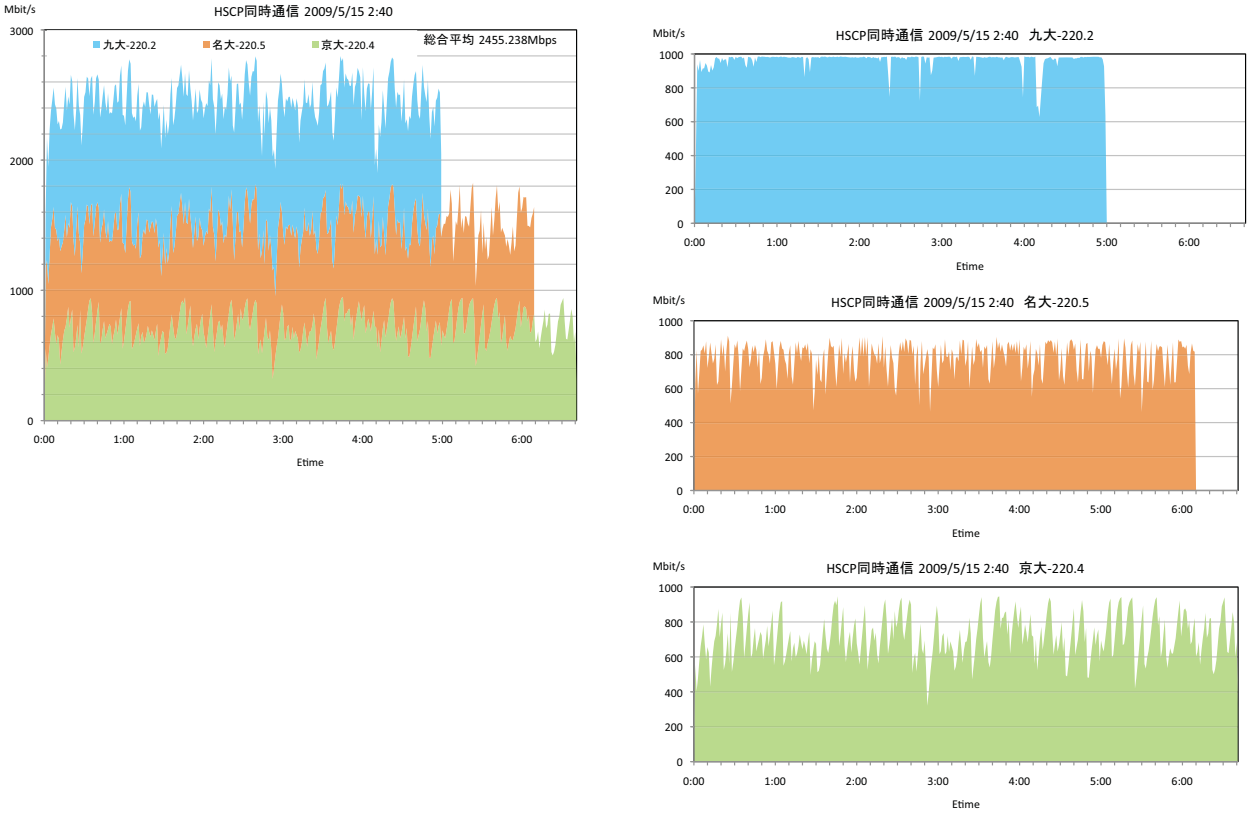
ccfep1(AIX) –Kyoto(SunOS) 9.838msec

プログラム名	転送時間(m:s)	転送速度(MB/s)	比較
hscp	0:28	38.3	-
scp	5:09	3.4	11.0 倍
sftp	5:08	3.4	11.0 倍

ccfep1(AIX) –Kyushu(Linux) 19.200msec

プログラム名	転送時間(m:s)	転送速度(MB/s)	比較
hscp	0:13	82.6	-
scp	1:31	11.3	7.0 倍
sftp	1:37	10.6	7.4 倍

表 4.　３カ所同時通信



これらの比較表から分かることは、一概に RTT によって通信速度が決まってこないことである。

表 4.は３カ所（九大、名大、京大）同時にそれぞれ 16GB のファイルを転送した時の通信速度をグラフにしたものである。この時、九大との間では平均値で 963Mbps でていた。

4　デメリット

しかし、いいことばかりではない。hscp はパケットロスに弱いという欠点がある。ネットワークラインの利用が多くなり、混雑してくると大きく速度が劣化する。当センターでは、通信遅延を発生させる遅延シミュレータシステム（以下、遅延システムとする。NIST Net [参考文献[7]]）を用意し通信遅延における転送速度の評価を行ったのでその方法と結果を記す。

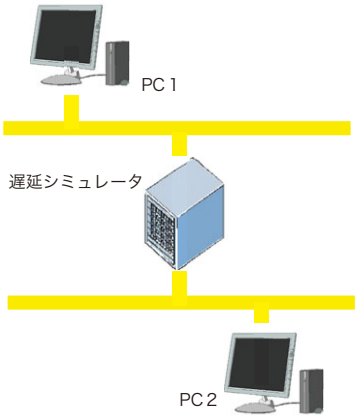


図 4.　構成図

遅延システムは、ネットワーク上で常に設定された遅延時間を発生させる模擬ネットワークシステムであり、図 4.のように PC と PC の間において遅延を発生させ、実際の遠距離間ネットワークを手軽に模擬できる。例えば、遅延システムにおいて 100msec と設定すると、PC1 から PC2 への通信における RTT が 100msec になるようする。このシステムを使用して、sftp、scp、hscp の RTT に対する通信速度をテストした。

その結果を表 5.に記す、この結果から次のことが考察できる。

1. SSL 通信は、暗号化処理で CPU 律速となり速度が出ない。
2. 一般の TCP プロトコルは RTT27msec あたりで RTT 律速となって、RTT 依存の右下がりになる。それ以前は 1Gbps のネットワーク律速である。
3. UDT は夢のプロトコルではなく、やはり RTT 律速になる TCP と同様に 27msec あたりから速度が制限されるが、TCP よりも減少度が少ない。
4. 近距離においては、10GbE を使うことで、1Gbps 以上の速度が期待できるが、RTT 律速の傾きから推測すると、10Gbps の実現性が困難であることが予想され、今後 10Gbps 環境で計測していきたい。

表 5.　RTT に対する転送速度

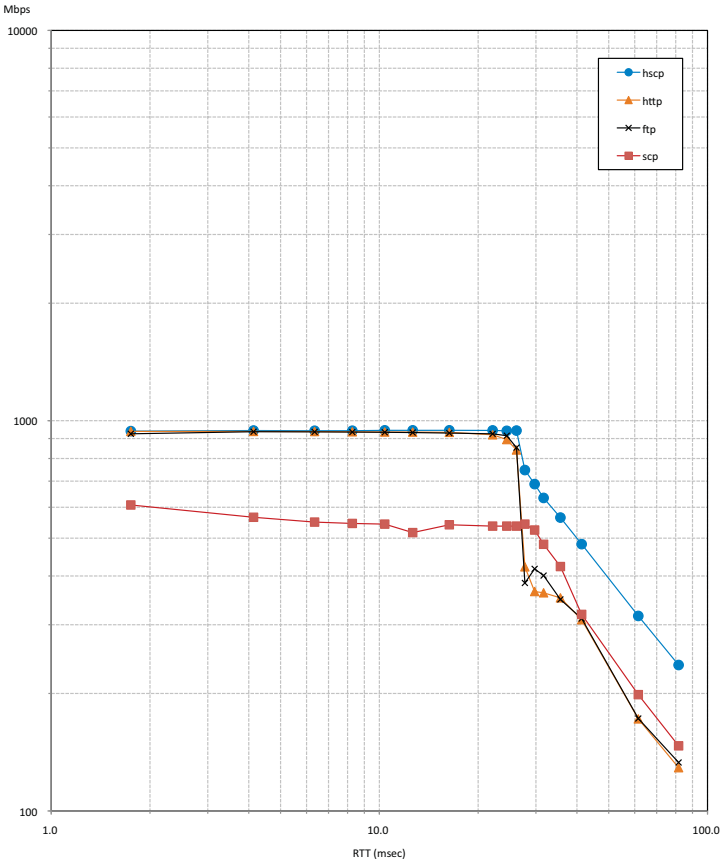


表 6.は、プロトコル毎のパケットロスによる速度劣化の様子である。hscp は、パケットロス 0.002%あたりから急激に速度が落ち、scp をも下回る様子が見える。

表 7.は、九州大学との通信において、速度が良い時と悪いときのものである。この表を見ると、SNAK が多いと速度が著しく低下することが分かる。

表 6. プロトコル毎のパケットロスによる速度劣化

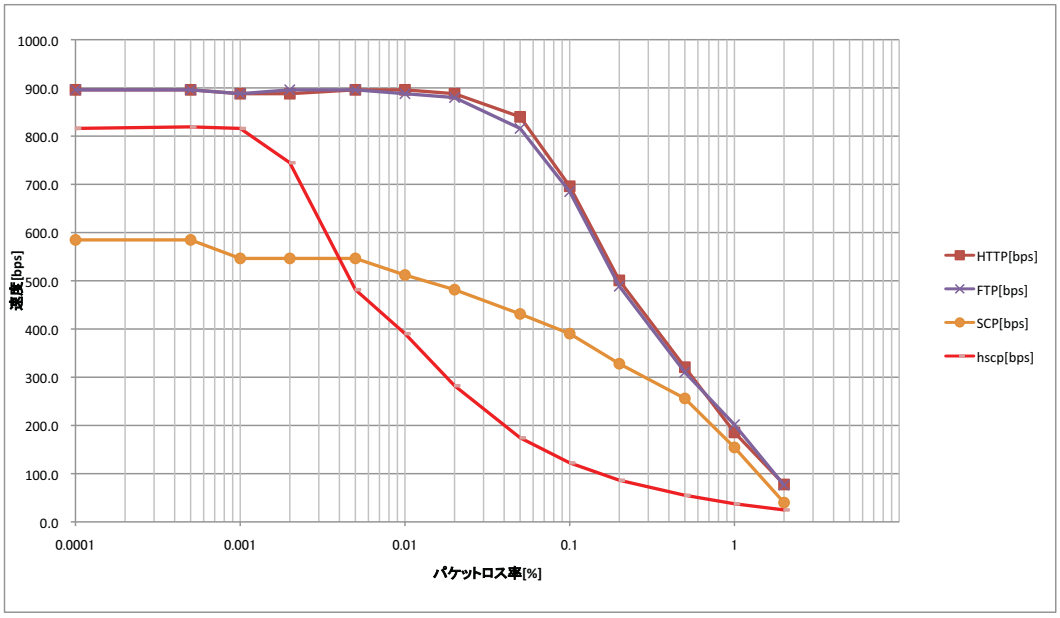
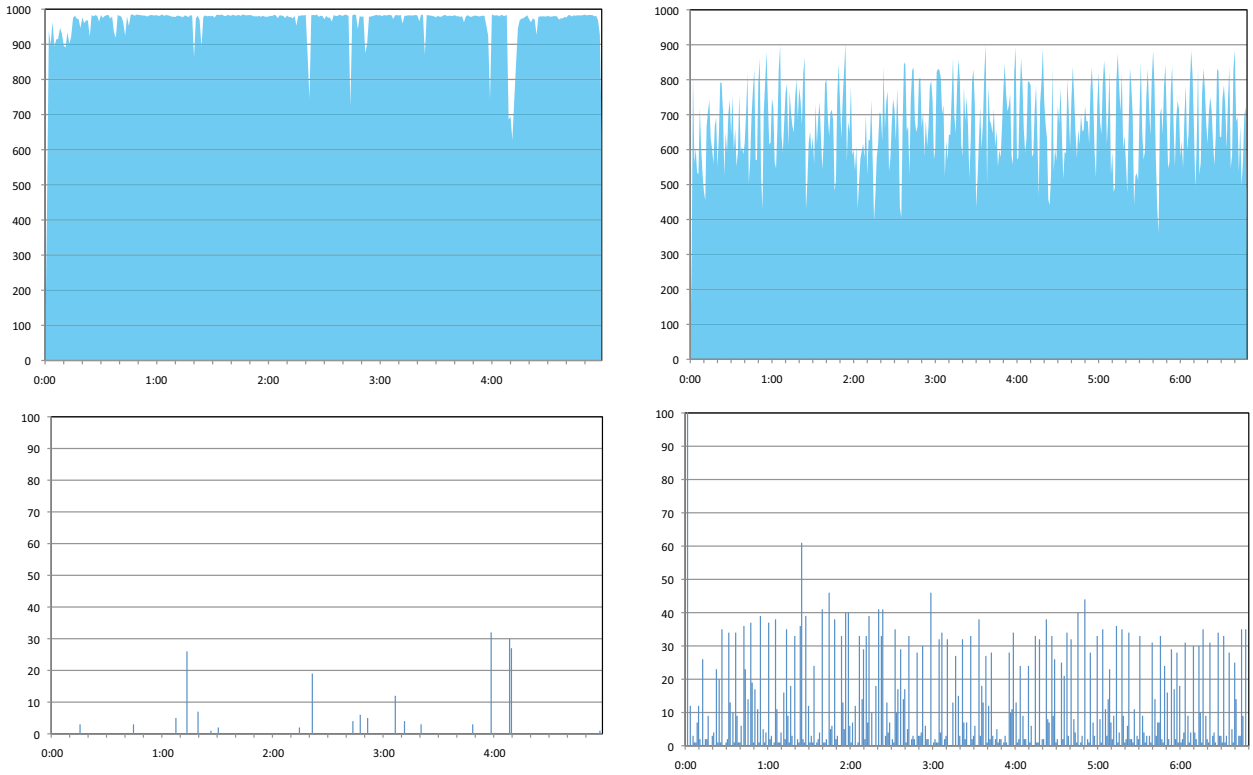


表 7.　SNAK における通信速度の比較



SNAK が少なく速度が良い時　　SNAK が多く速度が悪い時

上側：経過時間に対する転送速度（Mbps）、下側:経過時間に対する SNAK 数

5　今後について

hscp についてはダウンロード件数などを見て、それなりに世間で使用評価されていると思われるが、当センターの共同利用計算機システムにおける利用がないのが実情である。これは、センター利用者が scp 自体あまり使わず sftp を利用していると考えられる。今後は、sftp の UDP 版 hsftp の開発も行って行きたいと考

えている。

謝辞

このプログラムの開発にあたり援助いただいた計算科学研究センターに感謝します。
scp に UDT を使う改良を行った Hironori Kogawa（Hitachi ,Ltd.）に感謝します。
hscp のベースとなっている OpenSSH、PuTTY および UDT の提供者に感謝します。
そして、すべてのオープンソース開発者にも感謝します。
プログラムテストでお世話になりました次の皆様にも感謝します。
Tohoku Univ.　A.Morita, T.Ishiyama,
Tohoku Univ.　IMR, K.Ichinoseki, S.Miura,
Tukuba Univ.　CCS, T.boku, O.Tatebe,
Sophia Univ.　S.Nanbu(kyusyu Univ),
Nagoya Univ.　S.Okazaki, K.Kawaguchi
Nagoya Univ.　T.ishihara,
Nagoya Univ.　ITC, K.Ishi,
Kyoto Univ.　ACCMS,
Kyusyu Univ.　RI2T, M.Aoyagi, T.Takami

参考文献

[1] 長屋　貴量、他、 “広域 WAN を活かした高速ファイル転送プロトコル導入に向けて”、平成 20 年度京都大学総合技術研究会報告集、平成 21 年 3 月、
[2] 内藤　茂樹、”高速ファイル転送-1TByte 転送を見据えて”、第 4 回自然科学研究機構技術研究会、平成 21 年 6 月、P59-P62
[3] 手島　史綱、“高速通信プログラム開発にあたってのよもやま話”、分子研レターズ 60、平成 21 年 9 月、P59
[4] UDT（UDP-based data transfer）, 米イリノイ大学シカゴ校などの共同グループが開発した UDP ベースのプロトコルである, http://udt.sourceforge.net/index.html　（「Poweredby UDT」に記載されている）
[5] OpenSSH, http://www.openssh.com/index.html
[6] PuTTY, http://www.chiark.greenend.org.uk/~sgtatham/putty/
[7] NIST Net,http://snad.ncsl.nist.gov/nistnet/