Test of data transfer over an international network with a large RTT

J. Tanaka*, M. Ishino, T. Mashimo,
H. Matsumoto, H. Sakamoto, I. Ueda,
International Center for Elementary Particle Physics (ICEPP),
the University of Tokyo, Tokyo 113-0033, Japan,

S.Y. Suzuki,
High Energy Accelerator Research Organization (KEK),
Tsukuba, Ibaraki 305-0801, Japan

Abstract

We have measured the performance of data transfer between CERN in Switzerland and ICEPP at the University of Tokyo in Japan. A connection with 1 gigabits per second is available between the two sites. The round trip time (RTT), however, reaches about 270 msec. Moreover the connection is not dedicated to us. Due to the large latency and other traffic on the same network, it is not easy to fully exploit the available bandwidth. We report results of our measurements and investigations.

INTRODUCTION

Motivation

International Center for Elementary Particle Physics, ICEPP, will be one of the regional centers for analysis of the data from the ATLAS experiment [1] which will start data taking in 2007. A few petabytes of data are expected to be generated from ATLAS each year. It is therefore essential to achieve a high throughput of data transfer over the long-distance network connection between CERN and ICEPP.

Network Configuration

Figure 1 shows the configuration of our network connection for this test. A bandwidth of 10 gigabits per second (Gbps) is available in the backbone network [2] between Europe and Japan. However, the bandwidth for our measurements is currently limited to 1 Gbps both at CERN and the University of Tokyo. At each site we set up a special route to bypass the firewall.

TCP

The standard transport layer used on the Internet today is the Transmission Control Protocol (TCP). Most of data transfer applications are based on this protocol. However, the original TCP was not designed for the network with a high bandwidth such as 1 Gbps or more, or the wide area network with a very large round trip time (RTT). As networks of high bandwidth-delay products become popular,

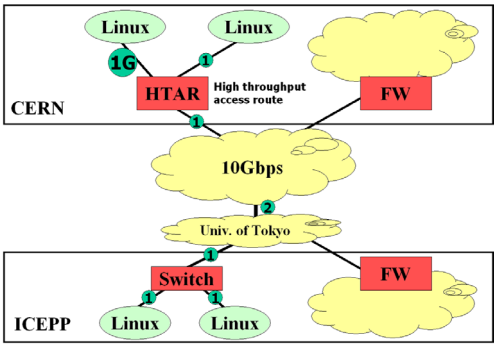

Figure 1: Network configuration: At CERN the so-called High Throughput Access Route (HTAR) is used in order to get around the firewall. The network path at ICEPP also bypasses its firewall. Each PC is equipped with a gigabit Ethernet NIC.

several options have been added to the TCP protocol. In the measurements described in the following sections we always turned on the window scale and the timestamp options to allow a larger window size and to protect against wrapped sequences, respectively. In the standard Linux kernel, the option of selective acknowledgment (SACK) is also turned on for better detection of dropped segments [3]. The SACK option can contain up to 4 blocks of information, but a maximum of 3 blocks can be accommodated in the TCP header when the SACK option is used with the timestamp option.

SETUP

For the measurements, the Linux PCs were installed at CERN and ICEPP sites. Red Hat Linux 9 was used and its kernel was replaced with the following ones:
(1) 2.4.25 with Scalable TCP (version 0.51) [4]
(2) 2.6.6-1.427 with its standard TCP stack.
In the following we will refer to the kernel and TCP stack combination (1) as simply “2.4” and (2) as “2.6”.

We set and varied several parameters of kernels and network interface cards (NIC) as follows.

```
[kernel]
net.ipv4.tcp_sack = 0 or 1
net.ipv4.tcp_rmem/tcp_wmem = 4096 65536 134217728
```

* Junichi.Tanaka@cern.ch

```
net.ipv4.tcp_mem = 134217728 134217728 134217728
net.core.rmem_default/wmem_default = 65535
net.core.rmem_max/wmem_max = 134217728
net.core.netdev_max_backlog = 3000

[NIC]
txqueuelen = 8000
```

On all the PCs the disk access speed exceeds 90 MB/s for both reading and writing.

MEASUREMENT: MEMORY TO MEMORY

We measured the performance of memory-to-memory transfer using iperf [5]. We set the size of the TCP socket buffer to 32 MB for both clients and servers. We transferred data from CERN to ICEPP.

Effect of SACK

First we checked the effect of specifying the SACK option for the 2.6 case. Figure 2 shows a distribution of the measured speed. On average the speed is 240 Mbps for SACK=ON and 270 Mbps for SACK=OFF. There is no improvement in the transfer speed by turning on the SACK option. The results in the following sections were all obtained with the SACK option turned off.

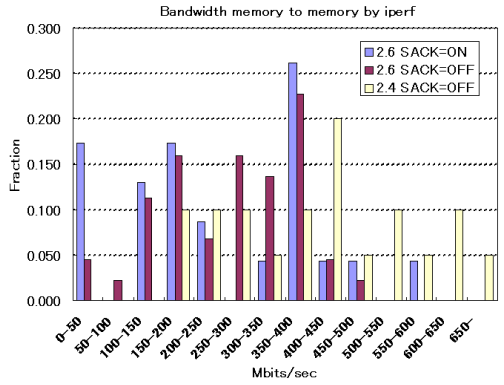

Figure 2: Distribution of the speed measured by iperf. We measured the transfer speed several times. The distribution is normalized to 1.

Effect of TCP Stack

We compared the transfer speed of 2.6 and 2.4. One of the main differences between the standard TCP implementation and the Scalable TCP is the method of the recovery of a TCP window size in the congestion avoidance phase. As shown in Fig. 2, the Scalable TCP performs better than the standard TCP stack (average $\sim$ 410 Mbps).

Comparison between Single and Multiple Streams

We checked the effect of increasing number of simultaneous transfer sessions. The result is shown in Table 1. In the case of 2.6, four parallel streams improves the aggregate throughput. On the other hand, in the case of 2.4, there is no improvement by varying the number of streams. Table 2 shows the ratio of the throughput of the slowest stream to the fastest stream. For both kernels, there is an inequality of throughput among streams. It suggests that a strong interaction exists between the streams.

To observe the variation of the TCP window size during a transfer session, we built a tool to estimate the size of the TCP window from the output of tcpdump [6]. Figure 3 explains the principle of the estimation. Figure 4 shows the size of TCP window as a function of time for the 4 different cases. We clearly see a slow-start phase and a congestion avoidance phase in Figs. 4 (a) and (b). In Figs. 4 (c) and (d), there are streams with small TCP window sizes. It indicates that streams have large effects on each other. The streams with small TCP windows are not contributing to the total throughput effectively.

Table 1: Aggregate throughput measured by iperf

Setting	Average (Mbps)
2.6 / 1 stream	270
2.6 / 4 streams	400
2.4 / 1 stream	410
2.4 / 4 streams	420

Table 2: Ratio of the throughput of the slowest stream to the fastest stream

Setting	Ratio
2.6 / 4 streams	0.25
2.4 / 4 streams	0.26

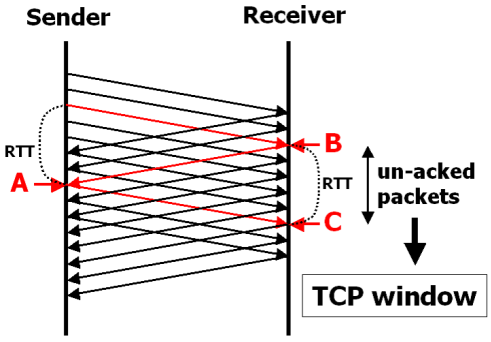

Figure 3: Estimation of a TCP window size: When a packet is received at time C, we know that packets between B and C are sent but their acknowledgments are not received yet by the sender at time A. The total size of packets between B and C corresponds to the TCP window size at time A.

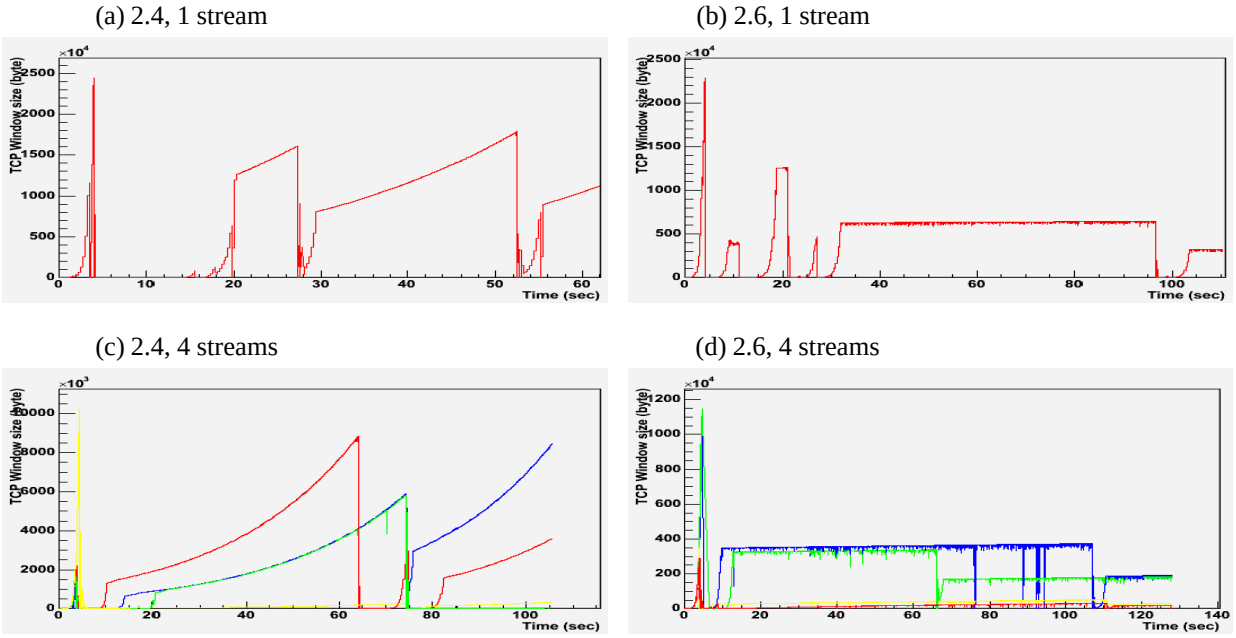

Figure 4: TCP window size (in bytes) as a function of time (sec) for each stream

MEASUREMENT: DISK TO DISK

To evaluate the network performance in realistic cases we measured data transfer speed from disk to disk by using bbftp [7]. We transferred 15 or more files from CERN to ICEPP. The size of each file is about 2 GB. We measured the transfer speed for various numbers of the parallel streams (1, 2 or 4), different kernels (2.4 or 2.6) and different sizes of TCP socket buffer (8, 16 or 32 MB). The results of the averaged transfer speed are shown in Table 3. For the case of 2 or 4 parallel streams the aggregate throughput is shown. In most cases the total throughput is worse than the throughput for the single stream case, in contradiction to a naive expectation. It may be due to the behavior of the TCP window size described in the previous section. In the case of a single stream, the transfer speed generally improves as the TCP socket buffer becomes larger. The Scalable TCP performs better than the standard TCP stack.

Table 3: Average of data transfer speed (MB/s) by bbftp

2.4	**8MB**	**16MB**	**32MB**
1 stream	20.0	26.4	27.8
2 streams	24.4	20.8	-
4 streams	19.0	19.3	-
2.6			
1 stream	9.6	10.5	10.3
2 streams	6.8	6.9	-
4 streams	7.3	7.3	-

We also measured the transfer speed for the two configurations shown in Fig. 5. In one case two clients send files to the same server. In the other case each client sends files to a different server. In the former case ("same destination"), the condition for the server is the same as 2 parallel streams. In the latter case ("different destination"), there are two independent network connections. Table 4 shows the results for these two configurations with 32 MB TCP socket buffer. In the case of 2.4, the method of the "different destination" gives the best result. However the result of 39.2 MB/s is not close to 55.6 MB/s expected from the performance of a single connection. In the case of 2.6, the results are better than the single connection by a factor of $\sim$2. The performance is, however, worse than the 2.4 case in both configurations.

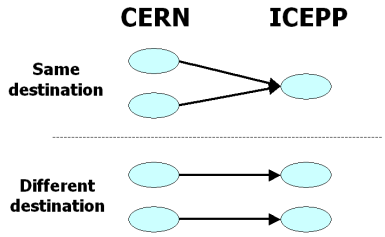

Figure 5: Configuration of servers and clients

Table 4: Average of data transfer speed (MB/s) by bbftp

	2.4	**2.6**
Same destination	33.0	19.2
Different destination	39.2	18.2
1 stream	27.8	10.3

CONCLUSION AND FUTURE PLAN

We have measured performance of the network between CERN and ICEPP. For this non-dedicated network connection (1 Gbps) with a large RTT (270 msec) the results are summarized as follows:

(1) The network performance is not improved by the SACK option.

(2) The Scalable TCP performs better than the standard Linux TCP stack.

(3) With a single PC at each end, use of multiple streams for disk to disk transfer results in poorer performance in the aggregate throughput compared with a single stream.

(4) The total throughput of two independent disk-to-disk transfers using two PCs at each end is better than the two transfers using one common PC at one end.

From these results, we have a plan to develop a file transfer program based on the scheme shown in Fig. 6. The data transfer proceeds as follows:

(1) A file to be sent is divided into multiple portions and they are transferred by multiple PCs.

(2) One PC at one end of the network and another PC at the other end are dedicated to a single connection. Multiple pairs of PCs transfer data in parallel and a copy of the file is assembled at the other end.

(3) When the transfer speed of a connection becomes slower than a given value, the connection is stopped, the information such as slow-start threshold and congestion window is flushed, and a new connection is started to continue the transfer.

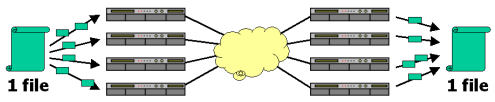

Figure 6: Schematic view of the file transfer method to be studied in the future

ACKNOWLEDGMENTS

We would like to thank P. Moroni (CERN) and Y. Karita (KEK) for their help on establishing the network connection for this measurement by arranging a HTAR route at CERN. We are also very grateful to the network management team of CERN for their help in our bandwidth measurements. We express our thanks to National Institute for Informatics (NII) for providing stable operation of SuperSINET.

REFERENCES

[1] “ATLAS Computing Technical Proposal”, ATLAS Collaboration, CERN/LHCC 96-43, 1996.

[2] National Institute for Informatics (NII) provides international network connections for academic use as well as a 10 Gbps backbone for the Japanese national academic network. There are currently 4 lines of 2.5 Gbps speed between Tokyo and New York, where the lines are directly connected to the European academic network (GEANT).

[3] RFC2018 (http://www.rfc-editor.org/rfc/rfc2018.txt) and RFC2883 (http://www.rfc-editor.org/rfc/rfc2883.txt)

[4] http://www-lce.eng.cam.ac.uk/ ctk21/scalable/.
We extracted the code of the Scalable TCP version 0.51 from the version 2.4.19 of the Linux kernel and then applied it to the 2.4.25 kernel.

[5] http://dast.nlanr.net/Projects/Iperf/

[6] The version 3.8.3 of tcpdump (http://www.tcpdump.org/) and the version 0.8.3 of libpcap are used for building our tool.

[7] http://doc.in2p3.fr/bbftp/