Fusion Engineering and Design 89 (2014) 770–774

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

A TCP/IP-based constant-bit-rate file transfer protocol and its extension to multipoint data delivery

Kenjiro Yamanaka [a,*], Shigeo Urushidani [a], Hideya Nakanishi [b], Takashi Yamamoto [b], Yoshio Nagayama [b]

[a] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
[b] National Institute of Fusion Science, 322-6 Orochi, Toki, Gifu, Japan

ARTICLE INFO

Article history:
Received 25 May 2013
Received in revised form 4 February 2014
Accepted 6 February 2014
Available online 19 March 2014

Keywords:
Network
Data transfer
LFN
Daisy-chain
Clock driven programming

ABSTRACT

We present a new TCP/IP-based file transfer protocol which enables high-speed daisy-chain transfer. By using this protocol, we can send a file to a series of destination hosts simultaneously because intermediate hosts relay received file fragments to the next host. We achieved daisy-chain file transfer from Japan to Europe via USA at about 800 Mbps by using a prototype. The experimental result is also reported. A total link length of a data delivery network can be reduced by daisy chaining, so it enables cost-effective international data sharing.

1. Introduction

Leading-edge scientific projects require high-speed networks and an efficient data transfer method to share measured or calculated data among geographically distant locations. Examples of such projects include experimental analyses and simulations in scientific disciplines such as high-energy physics, climate modeling, earthquake engineering, and astronomy. In these projects, many research groups share data in global collaboration, since each facility tends to be expensive.

The ITER project also requires high-speed networks and the efficient data transfer method. The ITER is a research and engineering project to design, build, and operate an experimental Tokamak Nuclear Fusion Reactor. It is constructing in Cadarache, France with international collaboration of the EU, Japan, Korea, China, India, Russia, and USA. The ITER will generate 100 or 1000 GB data per shot [1] and these data must be delivered in a timely way to participant countries. An efficient and effective way to multipoint data delivery is required.

Since participants are spread around the world, the data delivery method should support long-distance high-speed transfer. Furthermore, it should make efficient use of network bandwidth because long-distance broadband lines are expensive. If ITER data will be delivered to each country by point to point, each participating country should prepare its own broadband network line to Cadarache which results in wasted network bandwidth because the same data will be delivered by each network line. On the other hand, using daisy-chain transfer enables efficient use of network bandwidth. Each country transfers data in sequence until data reaches the final destination from Cadarache. Network bandwidth is shared efficiently by countries in the sequence. However, there is a weak point of this transfer: the transfer rate is restricted by the lowest speed link in sequence. Furthermore, it is well-known that obtaining a high-speed over long-distance line by using TCP/IP is difficult. As a result, if we use a traditional TCP-based data transfer method for a daisy-chaining data delivery, we cannot establish high-speed data transfer because the transfer speed of the longest network line limits overall transfer speed.

We have developed a new TCP-based file transfer protocol, Massively Multi-Channel File Transfer Protocol (MMCFTP). By using this protocol, we can transfer a file at a specified bit rate regardless of the transmission distance. We introduce MMCFTP briefly, and present an extension of MMCFTP for a daisy-chaining multipoint data delivery. As this extension enables high-speed multipoint data delivery, it is useful for international collaboration of leading-edge scientific projects, such as the ITER project.

* Corresponding author. Tel.: +81 342122696; fax: +81 342128430.
E-mail address: yamanaka@nii.ac.jp (K. Yamanaka).

http://dx.doi.org/10.1016/j.fusengdes.2014.02.028
0920-3796/

2. MMCFTP and its extension to multipoint data delivery

The MMCFTP is a new TCP/IP-based file transfer protocol developed in National Institute of Informatics (NII) and has the following features:

(1) The user can specify a transfer rate. [1]
(2) Unless the specified rate exceeds the limit of execution environment,[2] the transfer is performed at this rate, regardless of the transmission distance.
(3) TCP tuning is not required.

The second feature is provided by changing the number of TCP connections (channels) automatically in accordance with the specified rate and TCP transfer rate. This function makes the third feature possible. If TCP tuning is not done, MMCFTP achieves a specified transfer rate using more channels. Several hundreds or thousands of channels are used in the long-distance high-speed transfer. Therefore, we call it a "Massively Multi-Channel" FTP.

2.1. MMCFTP overview

The inputs of the sender program are as follows:

H: Receiver host name. The receiver program should be started before transfer at host *H*.
F: File name to be sent.
T: Timer period. Range: 31.2 ms–1 s.[3]
C: Number of chunks to be sent in a timer cycle. Range: 1–384.
S: Chunk size. Three sizes are selectable: 64 KiB, 256 KiB, 1 MiB.

File *F* is divided into fixed sized chunks, and each chunk is assigned a sequence number. Before sending file *F*, the sender program makes a TCP connection to host *H*, and sends all inputs and the file size of *F* to *H*, then waits for the receiver's response. After receiving the accept response, the sender program prepares more than enough channels to be used for data transmission. By using a periodic timer, the sender program sends out *C* chunks with sequence numbers, for every *T* period.[4]

Therefore, the transmit rate V_s is represented as follows:

$$V_s = \frac{SC}{T} \tag{1}$$

For example,

$$(T, S, C) = (62.4\,\text{ms}, 96, 64\,\text{KiB}) \Rightarrow V_s = 806\,\text{Mbps},$$

$$(T, S, C) = (31.2\,\text{ms}, 384, 1\,\text{MiB}) \Rightarrow V_s = 100\,\text{Gbps}.$$

When sending a chunk, the sender program looks for a channel which is not sending another chunk,[5] then it sends the chunk by using the found channel. The number of TCP channels to be used is automatically balanced out to the TCP transfer rate by this mechanism. This mechanism is the key of the constant bit rate transfer. The receiver program receives chunks and keeps them in a buffer memory. Chunks are then written to disk in accordance with their sequence numbers. In transmission using different channels, reversing the order of the reception can occur. Therefore, random write is used.

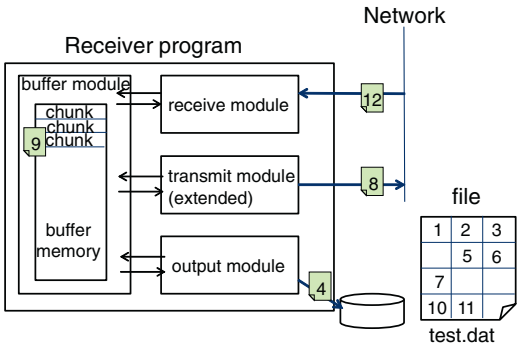

Fig. 1. Block diagram – receiver.

2.2. MMCFTP extension for daisy-chain transfer

To support daisy-chain transfer, the input *H* of the sender program is extended as follows.H_s: Receiver host sequence.

When the receiver program accepts the initial connection, it checks H_s and the next host. If the next host exists, the receiver program removes its name from H_s, makes a TCP connection to the next host, sends received data to the next host, then waits for a response of the next host. After receiving the accept response, the receiver sends the same response to the previous host, then prepares more than enough channels. Before chunks are written to disk, the receiver program sends chunks to the next host by the same manner of the sender program. Fig. 1 shows a block diagram of the extended receiver program.

3. Experimental result

We experimented with a daisy-chain transfer in a real network environment using a prototype program of MMCFTP. Fig. 2 shows this experiment environment. The experiment was performed using NII's campus network that is used on a daily basis. Receiver hosts were rented from a public cloud service, Amazon EC2. As MMCFTP does not require a tuning that depends on machines, rental servers were sufficient. We used the OS default for the TCP configuration. Therefore, the congestion control algorithm of the Tokyo machine and others are NewReno and Compound TCP, respectively. The storage specification of machines is special. Solid-state drives (SSDs) were selected because hard disk drives are too slow to achieve a transfer of 800 Mbps. In the experiment, we specified 806 Mbps as the transfer rate using a 62.4 ms timer period, and used a 11.6 GB file as test data. The estimated transfer time was 1 min 55 s. To compare the difference in transmission characteristics depending on differences in the chunk size, we did transfer experiments by specifying the same transport rate in chunks of three different sizes, 64 KiB, 256 KiB, and 1 MiB.

3.1. Chunk size: 64 KiB

The result is summarized in Table 1, and details are shown in Figs. 3 and 4.

TCP transfer rates are different, but the number of channels is automatically adjusted to keep the specified total rate. This property enables high-speed daisy-chain transfer. As shown in

[1] Rate range is from 525Kbps to 100Gbps, currently.
[2] The limit of execution environment includes network bandwidth, CPU speed, and storage access speed.
[3] Timer period is specified as a multiple number of the timer resolution. In Windows OS, the time resolution is 15.6ms.
[4] All tasks in MMCFTP programs are performed in a timer handler. This programming style is called Clock driven programming (CDP) [2]. The CDP is a software representation of the synchronous circuit design. To keep timer cycle correctly, we do not use blocked I/O. We use asynchronous or non-blocked I/O in CDP programs.
[5] To detect transmission completion correctly, non-buffered sockets [3] are used in the sender program.