An Application of Novel Communications Protocol to High Throughput Satellites

Ken T. Murata[1], Praphan Pavarangkoon[1], Kazunori Yamamoto[1], Yoshiaki Nagaya[1], Norihiko Katayama[1], Kazuya Muranaga[2], Takamichi Mizuhara[3], Ayahiro Takaki[3] and Eizen Kimura[4]
[1]National Institute of Information and Communications Technology, Tokyo, Japan
Email: ken.murata@nict.go.jp
[2]Systems Engineering Consultants Co., Ltd., Tokyo, Japan
[3]CLEALINKTECHNOLOGY Co.,Ltd., Kyoto, Japan
[4]Department of Medical Informatics, Ehime University, Ehime, Japan

***Abstract*—For network communications using modern high throughput satellite (HTS) on geostationary orbits, network throughput of transmission control protocol (TCP), one of the most popular protocols, is limited due to the packet loss on the satellite link. The packet loss is mainly caused by the attenuation of signals in severe weather conditions like heavy rain. It is high time to develop novel network communication techniques on the transport layer in TCP/IP designed for the systems and applications in broadband communications. In this paper, we introduce a high-speed data transfer protocol, named high-performance and flexible protocol (HpFP), to achieve high throughput for the HTS even with packet loss. The HpFP, in comparison with TCP-Hybla and UDP-based data transfer (UDT) protocols, is evaluated on a laboratory experiment simulating a geostationary orbit satellite link of 10 Gbps. It is clarified that the HpFP outperforms both the TCP-Hybla and the UDT showing high throughputs (close to 10 Gbps) when the packet loss ratio (PLR) is 1%, and remains more than 1 Gbps under even 10% PLR condition. Moreover, in case of no packet loss, the HpFP exhibits a quick start-up time (6 sec) at the initial phase to achieve 10 Gbps, while the TCP-Hybla and the UDT take 9 sec and 16 sec to their maximum throughputs, respectively.**

***Keywords*—HpFP, TCP, UDT, high throughput satellite, geostationary orbit, packet loss, long fat network**

I. Introduction

A new concept of high throughput satellites (HTS) have recently proven to represent affordable solutions that provide high data rates to a large number of users. Based on the applications of multi-beam technique of Ka band, the HTS is expected to improve the bandwidth management reaching instantaneous data rates of up to 100 Gbps [1]. Broadband satellite services show significant grows every year, due to the relevance of broadband satellite solutions in areas unserved or undeserved by terrestrial network to promote social inclusion [2]. The first generation broadband satellites such as iPSTAR, WildBlue I or SpaceWay 3 provide total capacities up to 20 Gbps. The second generation of Ka-band satellites, such as Ka-Sat and Viasat-1, achieves an economical scale because of higher frequency re-use factor allowed by multiple narrow satellite antenna beams and higher spectral efficiency modulation. The total capacities of coding schemes eventually reach from 70 to 140 Gbps.

Compared to these development of the HTS satellite communication systems more than 10 Gbps, the services and applications of high-throughput data transfer on the HTS have not yet developed enough so far. The data file transfer requires reliable network communication protocol like TCP (transmission control protocol). However, no challenge to achieve high throughput more than 1 Gbps with TCP protocols on geostationary satellite networks has been successful. One of the major reasons is that the heavy weather conditions and shadowing and multi-path receptions are not negligible. Many of the new HTS satellites utilize Ka-band and Ku-band frequencies for the simple reason that the orbital slot allocation for other bands has long been exhausted. A few literatures discuss packet loss taking place in these frequency bands. Katayama examined the packet loss ratio (PLR) on the Ka-band WINDS (Wideband InterNetworking engineering test and Demonstration Satellite) communication network [3]. Fig. 1 shows dependence of PLR and throughput on transmission attenuator (ATT) in the regenerative mode on WINDS. The packet loss caused by TCP packet drop off in heavy rain exponentially increases as the ATT value increases (when rain gets heavier). The maximum PLR is 14%, and the TCP connection is not maintained at the heavier rainfall moments even if the satellite link is not disconnected. An examination of data transfer on WINDS satellite indicates that the network throughputs of conventional protocols are unable to achieve the maximum bandwidth of the satellite link [4]. It is high time to develop new network protocols on the transport layer in TCP/IP available for applications with broadband communications on the HTS.

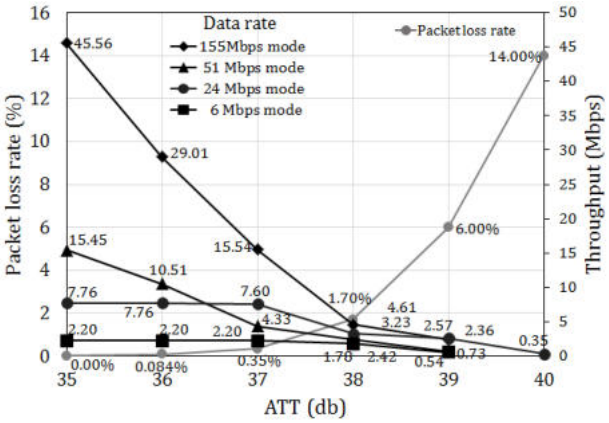

Fig. 1. Dependence of packet loss ratio and throughput on transmission ATT in [3]: window sizes to give maximum throughput are affixed.

For good performance on long-distance space networks such

978-1-5090-0996-1/16/$31.00 ©2016 IEEE

as cislunar orbits or deep space networks, several attempts to develop new congestion controls have been done, but could not get more than 1 Gbps network throughput, e.g., [6]. We introduced a high-speed data transfer protocol, named high-performance and flexible protocol (HpFP), to get high performance on a long fat network (LFN), 140 msec of round trip time (RTT), with packet losses of less than 1% [5]. To achieve high throughput even under the condition of high latency and packet loss on geostationary orbits, the HpFP puts more focus on high latency and packet loss tolerances than fairness and friendliness. In this paper, we describe in more detail of the HpFP and challenge to apply our protocol for HTS networks in severe weather conditions.

The rest of the paper is organized as follows. The HpFP protocol is introduced in Section II. The HpFP is evaluated on a laboratory experiment simulating HTS networks accompanied with packet losses to achieve 10 Gbps throughputs in Section III. Finally, we conclude the paper and discuss future works in Section IV.

II. High-Performance and Flexible Protocol

A. Overview

We introduced a novel data transfer protocol, namely HpFP, to achieve throughput higher than 10 Gbps on LFNs with packet loss [5]. The HpFP is designed on the top of UDP (user datagram protocol), but is a connection-oriented and reliable stream-type protocol like UDT (UDP-based Data Transfer Protocol) [10]. On the UDP segment header (IP stack), parameters are defined at the minimum. To realize the reliability on the UDP, we first design an original packet header format, as shown in Fig. 2. The packet header includes all necessary parameters for high throughput. The operation process of the HpFP using the parameters is discussed in Section II-B.

The HpFP is provided to the programmers as a socket library, which works on several operating systems (OS) such as Linux, Windows, Mac OS X, iOS, and Android. For this purpose, the HpFP is designed to work on user-lands independent of kernel-lands. The basic techniques of the HpFP are given in Section II-C. More details of control algorithms are discussed in Section II-D.

B. Sequence Diagram

In traditional TCP-based data transfer method, the receiver sends an acknowledgment (ACK) packet back to the sender immediately when a data packet arrives. In the LFNs, the data transfer rate decreases significantly as the RTT between sender and receiver increases. The expansion of window size to overcome this issue causes another issue of packet loss tolerance. To overcome this issue, in the HpFP, the ACK packets are sent back from the receiver independently of data packets arrival. The sending interval of ACK is empirically fixed at 200 msec; smaller interval causes CPU overload and larger interval makes time resolution in controls too low. The sender also transmits data packets independently of ACK arrival to avoid deterioration of total data transfer.

The other techniques in the HpFP sequence are as follows: (1) At the beginning of connection, the HpFP determines the maximum packet size without fragmentation using UDP-based path maximum transmission unit (MTU) search, as discussed in Section II-F. For effective measurement, the path MTU search is carried out using real transmission data packet. (2) The HpFP sets an initial target throughput as 50% of user assignment value of target throughput given by *-mt* option in Section II-G. (3) The HpFP intermittently determines pacing intervals based on the maximum packet size and the target throughput. (4) The HpFP intermittently modifies the target throughput based on network parameters carried by the ACK packets via PID (proportional-integral-derivative) control [7] in Section II-D1.

C. Basic Techniques

To achieve high throughput of applications using the HpFP socket library on user-lands, we develop techniques of usage of network I/O (mitigation of overheads), application of multi-core CPU, optimization of both size and number of packets, as well as high-precision algorithms of pace, transmission and flow controls. Moreover, a congestion control mechanism of the HpFP will be developed for use on public communication networks in the near future.

1) Usage of Network I/O: For high efficiency in network I/O, the HpFP supports a jumbo frame UDP packet. Moreover, to decrease the number of I/O call on operating system kernel, we allow the HpFP an alternative choice of IP fragmentation either on user-lands or on kernel-lands. The HpFP works on the user-land when CPU load is not fully occupied, and switches to aggregate on the kernel IP stack in heavy loading period of the CPUs. This technique is a trade-off with accuracy in pace control. However, this technique makes it possible to decrease the overhead by multiple kernel calls.

2) Application of Multi-Core CPU: In general networks, packet losses occur due to several reasons such as buffer overflow on network relay devices, excesses of traffic, short-durational noises, CPU overload in processing on routers or receivers, etc. In order to minimize packet losses during data transfer in LFNs, we perform to control the pacing of UDP packets with high precision on recent high-speed multi-core CPUs so that the maximum traffic is limited lower than wire-rate or available capacity. Note that we carry out the pacing control on user-land, not on kernel-land, because of high portability and versatility to work on any operating systems.

3) Optimization of Packets: We consider both size and number of data packets in design of an original packet header (segment header) of the HpFP, which is on the top of UDP, as shown in Fig. 2. The HpFP packet header is composed of two blocks. One is a common packet header for sequence management and exchange of control information. The other is the additional packet headers such as synchronization (SYN) and ACK. The control information is given by the other parameters including "Flags" in Fig. 2, where PAD, RTT, CRC, PMD, FIN, RST, SYN, and ACK represent data padding in active, RTT in measurement, cyclic redundancy check (CRC) in checking packets, path MTU search (discovery) in active, end of sending, reset of connection, start of connection, and ACK packet, respectively.

The additional headers are selected according to the objectives at each time step. The HpFP adopts a piggyback-like method in which payload data is packed to a packet with additional headers. Not only ACK but also SYN are included in the additional headers for high efficiency in data transfer, as shown in Fig. 2. This enables higher data transfer efficiency

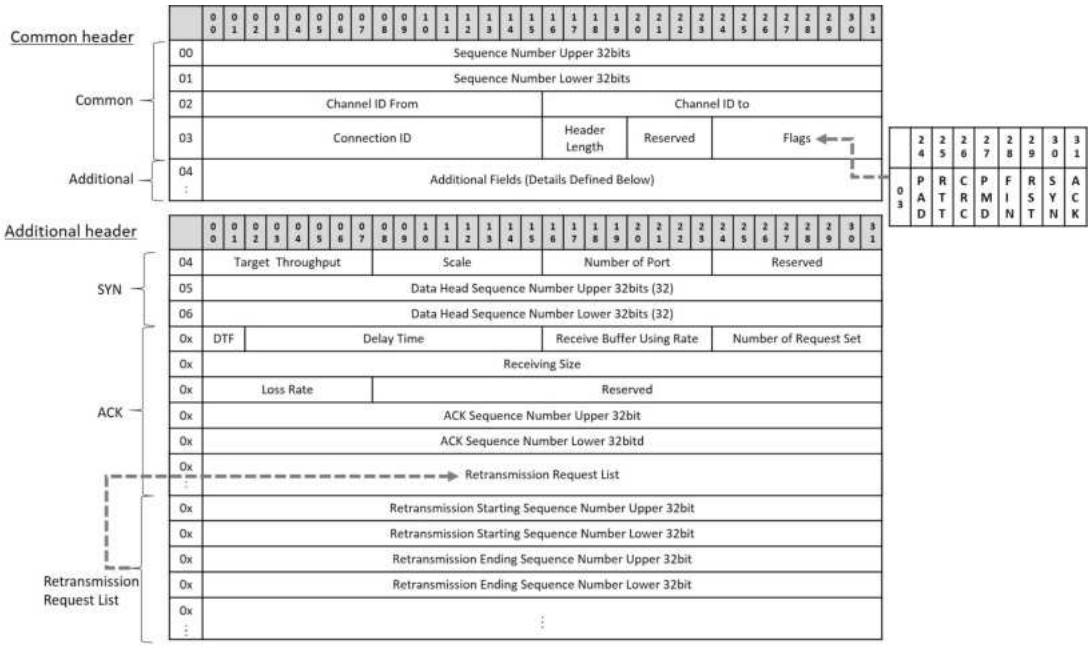

Fig. 2. Packet header format of HpFP

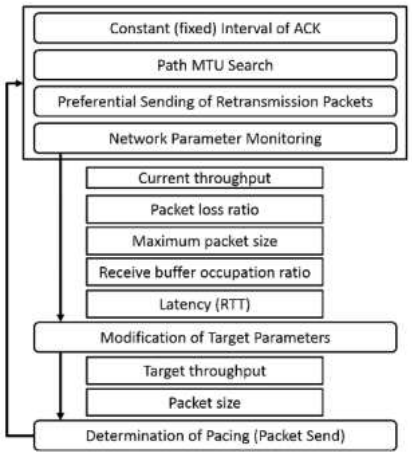

Fig. 3. Flowchart of HpFP pace control

in the HpFP than those in other TCP variants and UDT. Moreover, it also decreases the occupation ratio of header blocks to payload blocks on a data packet. Note that the HpFP payload occupies the residual size of the common and additional headers from the total packet size determined by path MTU search, as discussed in Section II-F.

D. Control Algorithms

As discussed in the section above, one of the most significant techniques in implementation of the HpFP is its control algorithms. The whole algorithm of the HpFP is combination of individual algorithm or functions. In this section, we survey pace controls, flow control, retransmission control, and other controls of the HpFP. The details of these control algorithms are not discussed herein due to the limitation of pages.

We have two pace control techniques. The first is for high throughput, which is higher than 10 Gbps. The second is for intelligent pacing technique for LFNs with high latency and packet loss.

1) Basic Pace Control: In order to achieve high throughput on high bandwidth networks, packet pacing techniques play important role. One of the most attractive schemes in the HpFP is to determine packet sending interval, called pace control. Based on the evaluation of the current throughput, PLR, maximum packet size given by path MTU search, the occupation ratio of receive buffer at a moment, and latency (described in RTT), the HpFP pacing interval for next time step is dynamically determined, as shown in Fig. 3. For this purpose, the HpFP adopts the following algorithms: constant (fixed) interval of ACK packet sending back, path MTU search using UDP packets to determine maximum packet length as discussed in Section II-F, network parameter monitoring such as latency (RTT) and packet loss, determination of pacing ratio based on prediction of actual throughput, and preferential sending control of retransmit packets.

Several ingenuities are operated for pace control of the HpFP throughput. A feedback control based on the PID controller [7] manages its packet sending up to the target throughput value based on the network information carried by the ACK, as shown in Fig. 3. In Fig. 4, the initial throughput is set as a half of the given target throughput by the user. Using two parameters of target throughput and packet size, the pacing interval (dt) is defined by Eq. (1). The details of the PID control is not discussed in this paper.

$$\text{dt (sec)} = \frac{\text{Packet-size (bit)}}{\text{Target-throughput (bps)}} \tag{1}$$

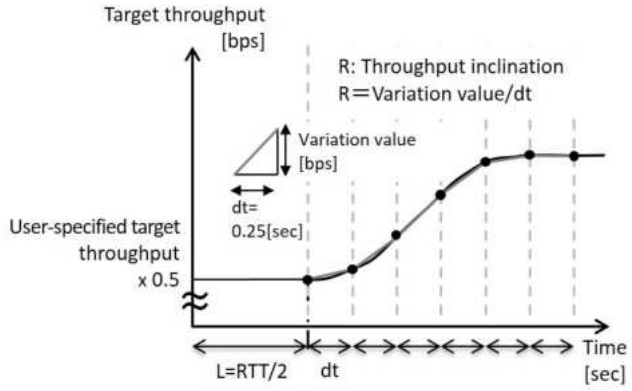

Fig. 4. PID control in HpFP

2) Pace Control on Packet Loss: On the network path of real LFN, there often appears a variety of phenomena that cause the performance degradation of data transfer, such as packet latency, packet loss, jitter (fluctuation of RTT), reversal of packet arrival that requires reorder of packets, temporal fluctuation of network bandwidth. They are the occasional causes of buffer overflow on the receiver. We develop several techniques in retransmission control, flow control, and other controls in order to minimize the effects of such network phenomena and to achieve stable throughput higher than 10 Gbps. The HpFP detects the latency and path MTU at the initial connection to reflect network conditions, as shown in Fig. 3. These parameters and other following parameters measured or estimated during the connection are once stored in a memory and reused to calculate target throughput: stationary measurements of RTT, the PLR constantly informed by the receiver using ACK packets, prediction of buffer overflow time estimated by current buffer utilization rate, and information of retransmission packets.

In case of no other traffic on network, the HpFP attempts to achieve almost wire-rate even on the condition of latency in few seconds with help of retransmission control and flow control. However, depending on the network conditions, e.g., network congestion, delay of packet transmission or packet loss and/or packet error often occur due to the overloading of packet pacing, which cause an overflow of network capacity larger than bandwidth-delay product (BDP). To avoid this congestion, the packet transmission ratio (pacing) from the sender is automatically controlled to adjust to network conditions.

3) Flow Control: In the TCP acknowledgment technique, the sender cannot get any information of the receiver before the arrival of the ACK packet from the receiver. This technique is ineffective since the sender keeps on sending packets up to the arrival time of the ACK packet or the reach of maximum transmission window size to indicate overflow on the receive buffer. To solve this issue, a flow control is incorporated in the HpFP. This flow control constantly monitors the receive buffer and predicts the overflow time from the amount of sending packets and network latency. The HpFP enables to avoid the buffer overflow and consequent packet losses.

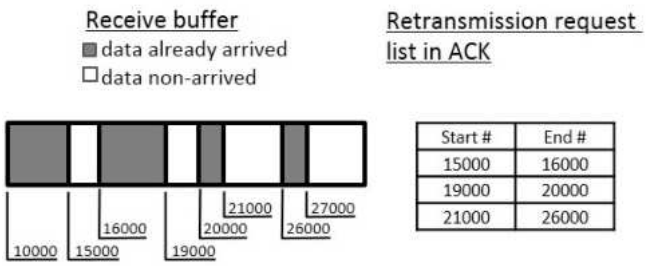

Fig. 5. An example of retransmission request list at packet arrival

E. Retransmission Control

As discussed in Section II-B, the receiver transmits the ACK packets independently of the packet transmission on the sender. When the receiver receives no data packet, the receiver writes the estimated PLR ("Loss Rate" in Fig. 2) with a list of retransmission sequence numbers (Fig. 5) of the ACK packet header. The sender receives these information and retransmits the lost packets according to the list. The receiver creates retransmission request list based on the sequence numbers stored in arrival packets. Moreover, using the number of arrived and non-arrived packets, the PLR is calculated at every 200 msec, the interval of ACK sending (Section II-B). The procedures to manage the retransmission request list are as follows.

1) If a series of sequence numbers received by the receiver is larger than the largest sequence number received in the past and contains a gap in order, the gap of the sequence numbers is considered to be lost packets. Starting and ending number of lost packet(s) are saved as "Retransmit Starting Sequence Number" and "Retransmission Ending Sequence Number", respectively, in the retransmission request list in Fig. 2.
2) If a series of sequence numbers received by the receiver is smaller than the largest sequence number received in the past and already in the retransmission request list, the corresponding sequence numbers are deleted from the list and the retransmission request list is updated.

For real-time data transfer, the balance between data transmission and retransmission is crucial. The HpFP dynamically determines this balance by using the ratio between the number of normal transmission packets and retransmission packets based on the PLR. The sender refers to the PLR in decision of the retransmission ratio between normal data transmission and the retransmission of the lost packets.

F. Other Controls

The HpFP adopts other controls, such as continual RTT estimation given as "Delay Time" in Fig. 2, continual packet loss counting given as "Loss Rate" in Fig. 2, packet loss detection by CRC, and UDP-based path MTU search. In the packet loss detection, the HpFP sets a bit error detection code in the header of sending packet, and then the code is recalculated after receiving the packet. In case of disagreement of both codes, the HpFP decides that the packet is broken. The UDP-based path MTU search technique is important to determine the maximum size of data packet. The sender

intermittently transfers a packet, in which the path MTU search flag is active (Fig. 2), and the HpFP determines the maximum packet size to be transferred without segmentation on the network. The payload size is fixed as the residual of the maximum packet and the header size.

G. hperf

We originally designed a new protocol, named the HpFP, which was built on top of UDP. As mentioned above, one of the original implementations is that it monitors throughput, latency (RTT), and packet loss. Therefore, we implement an easy-use application, named hperf, to measure end-to-end network conditions on LFNs. The hperf is a network quality measurement tool to monitor conditions of LFNs.

The latest version of the hperf is 0.7.39.0 in July 2016 (hereafter simply indicated as hperf) that works on major Linux distributions, and the interface is similar to the iperf [8]. The usage example of hperf on sender and receiver is as follows.

S: ./hperf client <Receiver IP address> -bs <buffer size of sender in unit of byte> -i <interval of log output in unit of msec> -ss <segment size of sender in unit of byte> -mt <target throughput in unit of bps> -sb <sending data size in unit of byte> -cpu <CPU ID used by thread> -crc <interval of CRC check in unit of packet> -o <output logfile name>

R: ./hperf server -bs <buffer size of receiver in unit of byte> -cpu <CPU ID used by thread> -crc <interval of CRC check in unit of packet> -i <interval of log output in unit of msec> -o <output logfile name>

Note that, in this paper, S denotes a sender (client) and R denotes a receiver (server). The target throughput is discussed in Section II-D1. The CRC is measurable on each side independently. This application gives us real-time status of packet loss and latency (RTT). The hperf is a freeware, and downloadable on the web site (http://hpfp.nict.go.jp).

III. Experiments

A. Laboratory Experiments

In this section, we evaluate the performance of the HpFP in a laboratory experiment simulating the HTS on a geostationary orbit in terms of throughput. The throughputs of the HpFP in different weather conditions, that are with different PLR and same latency, are compared with those of the conventional protocols, the TCP-Hybla (hereafter TCP) that shows the best performances among TCP-based protocols on geostationary satellite experiments [9] and the UDT protocol that shows high-throughputs on LFNs [10]. Our laboratory experiments are carried out on two servers (Table I) connected with 10 Gbps optical fiber, as shown in Fig. 6. In between them on the 10G network, we set a network simulator, Anue H Series GE mode, which generates packet loss (from 0% to 100%) and latency (from 0 msec to 250 msec). According to the study on the PLR on WINDS communication network in [3], we survey the throughput dependences on the packet loss from 0% to 10%. The RTT is fixed as 500 msec herein. In all of the experiments in this paper, we set kernel parameters using *sysctl* command to avoid bottlenecks caused by buffer size and window size settings as follows.

```
net.core.somaxconn = 512
net.ipv4.tcp mem = 1073741824 1073741824 1073741824
net.ipv4.tcp wmem = 4096 16777216 1073741824
net.ipv4.tcp rmem = 4096 16777216 1073741824
net.ipv4.udp mem = 1073741824 1073741824 1073741824
net.core.wmem default = 16777216
net.core.rmem default = 16777216
net.core.wmem max = 1073741824
net.core.rmem max = 1073741824
net.core.optmem max = 16777216
```

Herein we set 9000 bytes (jumbo frame) for packet sizes in the HpFP, the TCP and the UDT. The maximum window size of the TCP is fixed in 1 GB, that can provide the maximum throughput of 16 Gbps by the BDP theory. The target throughput parameters in the HpFP and the UDT are fixed at 10 Gbps. The throughputs of the HpFP and the TCP are measured by hperf-0.7.39.0 (Section II-G) and iperf-3.1.3 (TCP), respectively, and that of the UDT (version 4.11) is measured by udt commands.

- HpFP
  S: ./hperf client <receiver IP address> -bs 4g -mt 10g -sb 180g -ss 9000 -cpu 5 -i 1000 -o <output logfile name>
  R: ./hperf server -bs 4g -ss 9000 -cpu 5 -i 1000 -o <output logfile name>
- TCP
  S: ./iperf3 -c <receiver IP address> -t 90 -i 1 -f m
  R: ./iperf3 -s -i 1 -f m
- UDT
  S: ./appclient <receiver IP address> <receiver port number>
  R: ./appserver <listen port number>

We directly edited the methods in the source codes of the UDT (appserver.cpp and appclient.cpp) to change the MTU value, the buffer size, and the target throughput to achieve high performance on the 10 Gbps link as follows:

UDT::setsockopt(client, 0, UDT_MSS, new int(9000), sizeof(int));

UDT::setsockopt(client, 0, UDT_SNDBUF, new int(1000000000), sizeof(int));

UDT::setsockopt(client, 0, UDP_SNDBUF, new int(1000000000), sizeof(int));

UDT::setsockopt(client, 0, UDT_MAXBW, new int64_t(1250000000), sizeof(int));

TABLE I
Specification of servers in Fig. 6

Attribute	Specification
OS	CentOS release 6.8 (x86_64)
CPU	Core i7 980X Extreme (6 Core/3.33GHz/12MB)
Memory	4GB DDR3-1333 x3
NIC	Myricom 10G-PCIE-8B-SE

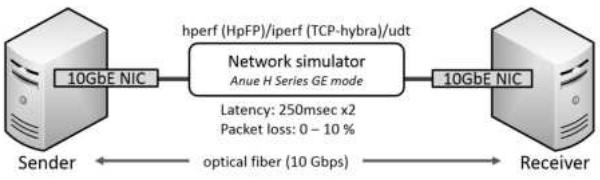

Fig. 6. Model of the laboratory experiments of HpFP, TCP-Hybla, and UDT

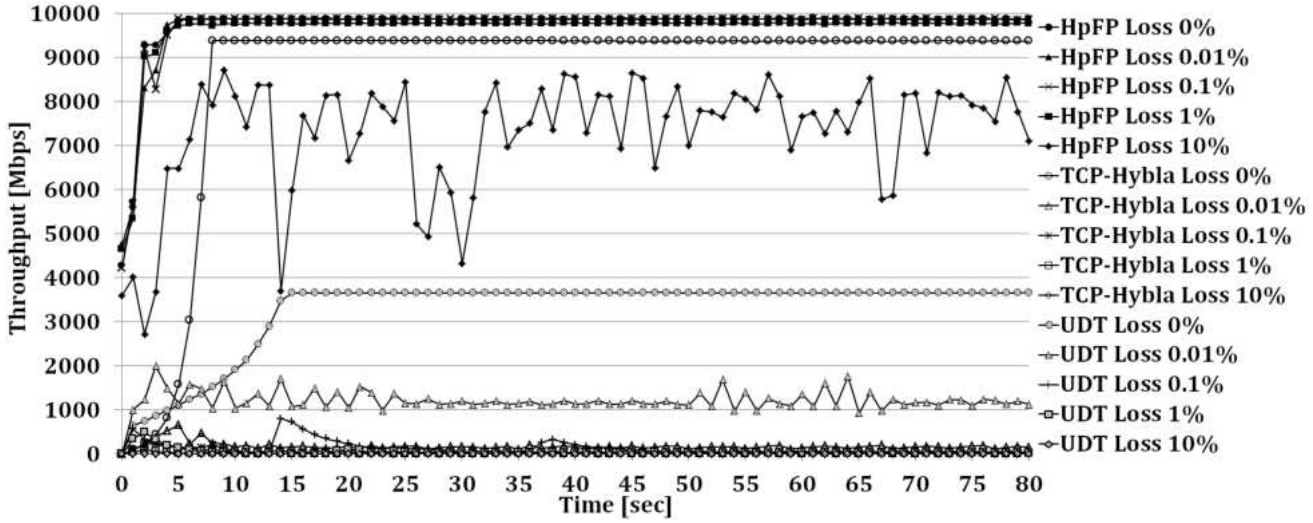

Fig. 7. Time-dependent throughputs of HpFP, TCP-Hybla, and UDT at different packet loss ratios

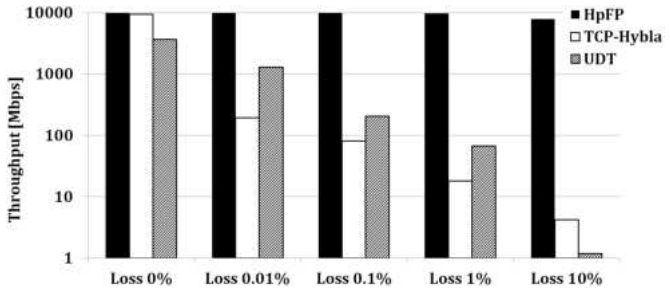

Fig. 8. Comparison of maximum throughputs in Fig. 7 of HpFP, TCP-Hybla, and UDT at different packet loss ratios

Fig. 7 shows the throughputs of the HpFP, the TCP and the UDT on the geostationary network with 0%, 0.01%, 0.1%, 1%, and 10% PLRs during 80 sec. We carried out the measurements three times for each data transfer, and obtained almost same results. Fig. 8 is the summary of Fig. 7. In case of no packet loss, the maximum throughput of the HpFP and the TCP are 9.9 Gbps and 9.38 Gbps, respectively, while that of the UDT is limited to 3.7 Gbps. The TCP throughput drastically decreases to about 200 Mbps even in low PLR (0.01%). The UDT throughput remains more than 1 Gbps in 0.01% PLR, but shows less than 1 Gbps at 0.1% PLR or higher as well. The HpFP throughput shows higher than 9.5 Gbps even in 1% PLR. Note that the HpFP throughput remains more than 1 Gbps even in 10% PLR. In summary, we conclude that both the TCP and the UDT show good throughput on the geostationary satellite link without packet loss. The throughput of both the TCP and the UDT drop down as the PLRs increase, while showing the robustness to packet loss of UDT is higher than that of the TCP [11]. The HpFP overcomes the throughput degradation issue on the geostationary link with packet loss.

Another advantage of the HpFP to the other two protocols is its start-up time. Considering software applications to transfer many big data files over a HTS, the start-up time significantly affects the performance of the data transfer. Table II is the comparison of start-up times between three protocols at no packet loss on the geostationary orbit link. We measure the time reaching to their maximum throughputs (9.9 Gbps in the HpFP, 9.38 Gbps in the TCP, and 3.7 Gbps in the UDT in Fig. 7). The HpFP shows the best performance in start-up time that is 6 sec, while the TCP and the UDT take 9 sec and 16 sec to their maximum throughputs, respectively. The TCP-CUBIC is measured as a reference as well, showing much longer start-up than others.

TABLE II
COMPARISON OF START-UP OF HPFP, TCP-HYBLA, UDT, AND TCP-CUBIC.

protocol	HpFP	TCP-Hybla	UDT	TCP-CUBIC
start-up [sec]	6	9	16	53

B. Discussions

The results of the laboratory experiments in this study indicate that the HpFP outperforms the TCP and the UDT in terms of throughput, tolerance to packet loss, and start-up time at the initial phase over HTS links of 10 Gbps with 500 msec RTT. In this section, we consider the reasons of the outstanding performance of the HpFP.

As shown in Fig. 7, the UDT is weak to packet loss, like the TCP, instead of its high performance on high latency conditions. It is because the pace control of UDT adopts the additive increase/multiplicative decrease (AIMD) algorithm [10], which is a feedback control algorithm best known for its use in TCP congestion avoidance. The AIMD combines linear growth of the congestion window with an exponential reduction when a congestion takes place. Multiple flows using the AIMD congestion control eventually converge to use equal amounts of a contended link. For this reason, the UDT succeeds in keeping fairness and friendliness with TCPs. Both the TCP and the UDT use combination of ACK and packet timer-based retransmission in the hold mechanism and retransmission control of lost packets on receive buffer; selective ACK (SACK)-based retransmission is not effective especially

in large packet loss periods since it simply retransmits a small part of lost sequence. The HpFP makes a list of retransmission packets in every 200 msec, as discussed in Section II-B. The sender intelligently determines the ratio between normal and retransmission packet based on the network condition, as discussed in Section II-E. This intelligent retransmission control enables the HpFP to achieve high performance in packet loss network conditions.

The slow-start and congestion avoidance mechanisms of the TCP cause the long start-up time. The increase of the TCP throughput becomes more gradual on higher latency networks since the return of ACK packet takes more time and the window expansion delays. Moreover, the packet loss on the satellite link is recognized as a congestion, and the window size is controlled to degrade in any enhanced TCP. This trend is almost same in the UDT since it adopts the similar congestion control mechanism to the TCP. On the other hand, the HpFP shows a quicker start-up time. One reason is that the initial target throughput is given as a half of the value given by the user, as discussed in Section II-A. Note that the target throughput is given as 10 Gbps in our examination. The pace control in the HpFP is managed in every 250 msec, which is described as dt in Fig. 4, and it takes only several times of dt to achieve maximum throughput. This is another reason that the PID control shows a quicker start-up time than that of the TCP. In addition, in the presence of packet losses, the difference in retransmission control between the TCP and the HpFP discussed in the previous paragraph becomes more dominant.

Through the discussions of the laboratory experiments in Section III, we proved that the HpFP could be one of the significant candidates of data file transfer protocol replacing the TCP on the HTS network communications. From the viewpoint of applications using the HpFP, however, usability of the protocol is important. The HpFP provides users with a socket library. The target OS of the library is Linux, Windows, Mac OS X, Android, and iOS. The socket library for Linux is ready, while others are in preparation. One of the motivations of the HpFP socket library is that one easily replaces TCP socket library in his/her programs with the HpFP. This enables us to implement applications easily, or to replace the socket library of the TCP with that of the HpFP in any programs.

IV. Conclusion

Recently the geostationary orbit satellites, which provide more than 1 Gbps link, have been developed or under development. The concept of HTS are proposed to provide high data rates to a large number of users. Based on the applications of multi-beam technique, the HTS is expected to improve the bandwidth management reaching instantaneous data rates of up to 100 Gbps. It is high time to develop new network protocols on the TCP (transport) layer in TCP/IP to provide 10 Gbps (and more) throughput and many applications to make flexible communications between users independent of their locations. However, no challenge to achieve more than 1 Gbps with TCP protocols on geostationary satellite networks even in the severe weather conditions has been ever successful.

In communications using geostationary satellites, the throughput of data transfer is limited due to latency on network (500 msec RTT) and packet loss caused by attenuation of signals in severe weather conditions like heavy rain. In this paper, we applied a high-speed data transfer protocol, named HpFP, to achieve throughput even in severe weather conditions higher than the TCP-based protocols on HTS links of geostationary orbit. The HpFP is evaluated on a laboratory experiment simulating a geostationary satellite orbits accompanied with packet losses. It was clarified that the HpFP shows almost 10 Gbps throughputs even in the condition that the PLR is 1%, while the TCP-Hybla and the UDT show less than 1 Gbps throughputs even when the PLR is 0.1%. Moreover, in case of no packet loss condition, the HpFP exhibits a quick start-up time at the initial phase in 6 sec to reach to the maximum throughput, while the TCP-Hybla and the UDT take 9 and 16 sec to their maximum throughputs, respectively.

Acknowledgment

The HpFP is developed on the NICT Science Cloud. Many thanks to Dr. Kenji Suzuki in NICT for information of satellite communication systems.

References

[1] L. D. C. H. R. Gaytan, Z. Pan, J. Liu, and S. Shimamoto, "Dynamic scheduling for high throughput satellites employing priority code scheme," *IEEE Journals and Magazines*, vol. 3, pp. 2044-2054, 2015.

[2] O. Vidal, G. Verelst, J. Lacan, E. Alberty, J. Radzik, and M. Bousquet, "Next generation high throughput satellite system," *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, pp. 1-7, Oct. 2012.

[3] N. Katayama, T. Asai, K. Kawasaki, and T. Takahashi, "A Study on throughput maximization on TCP communication for WINDS regenerative mode," *Trans. JSASS Aerospace Tech. Japan*, vol. 12, no. ists29, pp. Pj1-Pj7, 2014.

[4] K. T. Murata, P. Pavarangkoon, K. Suzuki, K. Yamamoto, T. Asai, T. Kan, N. Katayama, M. Yahata, K. Muranaga, T. Mizuhara, A. Takaki, and E. Kimura, "A high-speed data transfer protocol for geostationary orbit satellites," *2016 International Conference on Advanced Technologies for Communications (ATC)*, Oct. 2016.

[5] K. T. Murata, P. Pavarangkoon, K. Yamamoto, Y. Nagaya, T. Mizuhara, A. Takaki, K. Muranaga, E. Kimura, T. Ikeda, K. Ikeda, and J. Tanaka, "A quality measurement tool for high-speed data transfer in long fat networks," *24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sep. 2016.

[6] M. Sarkar, K. K. Shukla, and K. S. Dasgupta, "Delay resistant transport protocol for deep space communication," *International Journal of Communications, Network and System Sciences*, vol. 4, no. 2, pp. 122-132, 2011.

[7] J. Li-qiang, S. Chuan-xue, and L. Jian-hua, "Controll algorithm of combination with logic gate and PID control for vehicle electronic stability control," *2nd International Conference on Advanced Computer Control (ICACC)*, vol. 2, pp. 345-349, 2010.

[8] O. Olvera-Irigoyen, K. Abdesselem, and T. Laurent, "Available bandwidth probing for path selection in heterogeneous home networks," *IEEE Globecom Workshops (GC Wkshps)*, pp. 492-497, 2012.

[9] H. Yamamoto, G. Komada, K. Nakamura, T. Takahashi, and K. Yamazaki, "Performance comparison of enhanced TCPs over high-speed internetworking satellite (WINDS)," *2011 IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT)*, pp. 274-278, Jul. 2011.

[10] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks," *Computer Networks* vol. 51, no. 7. pp. 1777-1799, May 2007.

[11] C. Song and J. Chang-peng, "Application of UDT in microseism data transmission system," *2009 IEEE ISECS International Colloquium on Computing, Communication, Control, and Management*, vol. 4, pp. 225-228, Aug. 2009.