End-node transmission rate control kind to intermediate routers — towards 10Gbps era

Kei Hiraki, Makoto Nakamura, Junsuke Senbon, Yutaka Sugawara, Tsuyoshi Itoh and Mary Inaba
{hiraki, makoto, bonse, sugawra, tsuyoshi, mary}@is.s.u-tokyo.ac.jp

Abstract

While transferring data using TCP/IP, even though total data transfer rate is same, microscopic behavior of data transmission is quite different by network interfaces of end nodes. Inappropriate data transmission rate causes needless packet losses which result in bad perforamance, and worse, it also causes needless loads for intermediate routers. This problem becomes more serious as bandwidth becomes wider. This paper explains the problem and describes overview of our approaches.

1 Introduction

It is well known that difficulty of TCP/IP data transfer on LFN(Lonf Fat pipe Network) is window size control. Roughly speaking, data transfer rate is $window\ size/RTT$, hence, LFN needs large window size. On the other hand, window size grows by Ack which returns after RTT interval. And, a lot of studies have been done for appropriate window size control.

But, problem of LFN is not only window size control, but also difference of transmission rate and transfer rate. Let X denote transfer rate and Y transmission rate of the network interface. While transferring data, roughly speaking, the packet is sent via interface for only first $\Delta t = X \times RTT/Y$ of every RTT, which we call "busy Δt", then next $RTT - \Delta t$ time, that interface is idle. Hence, network needs buffer of size $Z = (Y - X) \times \Delta t$. This peaky behavior for bysy Δt easily causes needless shortage of buffer memories of intermediate routers which results in needless packet losses. Note that busy Δt is long when RTT is long, and buffer size Z needs to be large when transmission rate Y is large and RTT is long. This problem becomes more serious when 10Gbps interface is available for high-end PCs.

From now, we show our observation, then describe overview of our approaches, i.e., (1)IPG tuning of microsec order, (2)Clustered Packet Spacing of millisec order, and (3)TCP-based 10Gbps ethernet NIC with hardware rate control. All experimental data is taken between Tokyo, Japan and Maryland, U.S., 7200 miles distance with bottle neck OC12/POS. We use `iperf` command on DELL PowerEdge1650(Dual Pentium-III 1.4GHz, 1GB memory, on-board NIC with 82554chip) with Linux 2.4.22 and 2.6.0-test5 with NIC driver e1000 Ver.5.1.13, without using TCP Segmentation Offload. GbE and FE is switched by Summit 5i ethernet switch.

Figure 1. GbE v.s. FE

TxQ	100 packets			25000 packets		
	Min	Max	Middle	Min	Max	Middle
GbE	9.07	120.0	11.6	9.52	78.0	44.6
FE	25.6	25.7	25.6	88.6	88.7	88.7

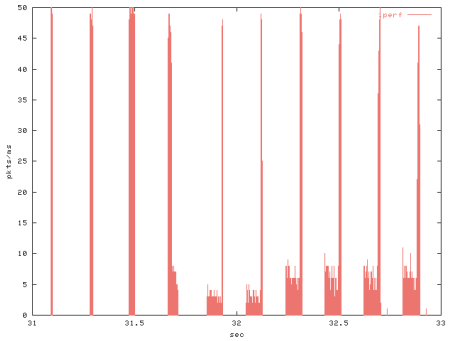

Figure 2. Number of received packets for every millisecond(2sec, GbE)

2 Gigabit Ethernet v.s. Fast Ethernet

Figure 1 shows minimum, maximum and middle throughput of data transfer on Gigabit Ethernet (GbE) and Fast Ethernet(FE) with TxQueue 100 (default) packets and 25000 packets. We observe that performance on FE is more stable and faster than that on GbE in more than half cases. Figure 2 is an enlarged graph of the number of packets received for every millisecond for 2 seconds. With GbE, for first several milliseconds of every RTT, a node send about 80 packets for every millisecond, the other end receives about 50 packets, and there exist a lot of idle time. On the other hand, with FE, about 8 packets are sent and received almost constantly, and there exist no peak. And, with $TxQ = 25000$, busy Δt is RTT. This is the reason why FE is more stable, and shows better performance in many cases.

From now, we show 3 approaches to make data transmission rate same as data transfer rate; ideally, send packets every $RTT/cwnd$, where $cwnd = Window\,Size/Packet\,Size$.

But it's impossible to realize this by software because of precision of kernel timer and heavy load of CPU. Note that one ether packet is sent in about 12 μ sec for GbE and 1.2 μ sec for 10GbE, on the other hand, UNIX kernel timer is about 10msec (Linux 2.4) and 1 msec(Linux 2.6).

3 Inter Packet Gap tuning

Inter Packet Gap(IPG) is a gap which is placed between ethernet packets to make transmitter be idle. Its value should be more than 12 bytes (by IEEE 802.3), but default IPG of ethernet driver e1000 is settled to 8 bytes. We change this IPG value by modifying e1000. Figure 3 shows the maximum and minimum and average throughput when we change IPG from 8 to 1023 bytes.

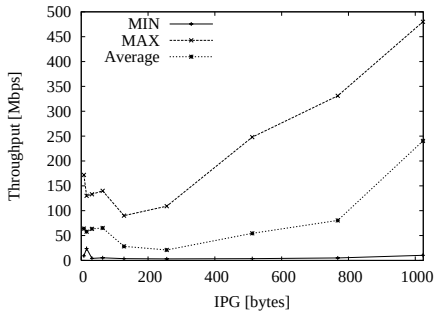

Figure 3. IPG and transfer rate with GbE RTT=198ms

It's true that IPG tuning is effective to make bursty behavior of GbE milder, i.e., busy Δt becomes $(IPG + MTU)/MTU$ times longer, when we assume one packet is about MTU long. But shortcoming of IPG tuning is that it affectes all communication over LFT-tuned interface.

4 Clustered Packet Spacing

To control transmission rate for multiple streams simultaneously, we conduct packet spacing during slow start phase in TCP stack as follows,

(1) While $RTT/cwnd > T$, send packets with $RTT/cwnd$ interval

(2) If $RTT/cwnd < T$, return ordinary TCP routine.

where T is the precision of kernel timer, i.e., 10msec for Linux 2.4 and 1msec for Linux 2.6. (1) is the phase of interval control which settles the seed of cluster, and, in (2), i.e., ordinary TCP procedure, each cluster grows individually by each ACKS. The overhead is almost negligible for interval control lasts at most 10 times since the maximum RTT on the earth is about 500msec of satelite communication.

Figure 4 shows the state just after interval control phase finishes. The effect of Clustered Packet Spacing is, that busy Δt is divided into RTT/T pieces and placed every T interval instead of RTT. Then, the load of intermediate buffer is almost same as the data transfer rate when we consider the interval whose granurality is longer than T precision. But, since there exist

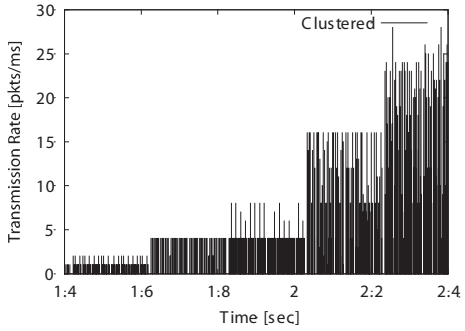

Figure 4. Number of packets for millisecond with Clusterd Packet Spacing, (precision 1ms, Linux 2.6)

RTT/T peaks, it easily suffers the effects of packet loss when window size grows too large.

5 Network Interface Card

Current network interface cards are designed to fit wide-range of applications, but they are not suitable to use for efficient use of LFN because hardware optimization of packet locations on each TCP stream and fine-grain scheduling of parallel TCP streams cannot be attained on them. In order to solve these issues, we are now developing a new 10Gbps network interface card with reconfigurable FPGA based network processor. On this 10Gbps NIC, the network processor is reconfigured to recognize and optimize each TCP stream to each property of connecting networks. A NIC consists of two 10GBASE-LR MAC and optical module, a Xilinx Virtex II pro FPGA with a processing core, a address lookup table (TCAM) and PCI-X interface. This NIC is also designed for local interconnection network inside a cluster computing system without an ethernet switch for reducing communication latency among processing nodes.

Acknowledgement

We thank Prof. Akira Kato of University of Tokyo and Dr. Matsuo and Dr. Masuoka of FLA(Fujitsu Laboratory in America). This study is supported by the Special Coordination Fund for Promoting Science and Technology, Ministry of Education, Culture, Sport, Science and Technology in Japan.

References

[1] M. Nakamura, M. Inaba, K. Hiraki, "Fast Ethernet is sometimes faster than Gigabit Ethernet on LFN – Observation of congestion control of TCP streams", PDCS2003, CA, USA, Nov. 2003. To appear.

[2] K. Hiraki, M. Inaba, J. Tamatsukuri, R. Kurusu, Y. Ikuta, H. Koga, A. Zinzaki, "Data Reservoir: Utilization of Multi-Gigabit Backbone Network for Data-Intensive Research", SC2002, Nov. 2002.