EVALUATION OF COMMUNICATION BANDWIDTH CONTROL MECHANISM BY REGULATING PROGRAM EXECUTION SPEED

Toshihiro TABATA, Yoshinari NOMURA
Faculty of Information Science and Electrical Engineering
Kyushu University
Hakozaki 6-10-1, Higashi-ku
Fukuoka 812-8581, Japan
{tabata, nom}@csce.kyushu-u.ac.jp

Hideo TANIGUCHI
Faculty of Engineering
Okayama University
Tsushimanaka 3-1-1
Okayama 700-8530, Japan
tani@it.okayama-u.ac.jp

ABSTRACT

With the spread of the Internet, services that communicate to other services are increasing. Multimedia applications such as video on demand also ask for network Quality of Service (QoS). Thus, operating systems have to guarantee the allocation of computer resources to services. The computer resources are CPU, disk, network devices and so on. We suppose that the communications have to be controlled well, because services using network are increasing. This paper proposes the communication bandwidth control mechanism by regulating program execution speed. Our proposed mechanism is based on the process schedule method for regulating program execution speed. In the process schedule method, the operating system reserves the amount of CPU time of target processes and guarantees the allocation of CPU time on a sending host. Our proposed mechanism can guarantee a required data rate of target processes by allocating enough CPU time for communications. Because operating systems manage computer resources, they guarantee the allocation of CPU time if the process schedule method is implemented. The allocation of CPU time is not almost affected by non-target processes. This paper introduces the process schedule method and the implementation of it. This paper also shows how to control the communication bandwidth of target processes. Then this paper describes about an evaluation of our proposed mechanism.

KEY WORDS

Process Scheduling, Operating System, Communication, Program Execution

1 Introduction

With the spread of the Internet, services that communicate to other services are increasing. Moreover, the services compete with themselves to get computer resources when the services coexist on a single computer simultaneously. Therefore, operating systems should guarantee the allocation of computer resources. The computer resources are CPU, disk, network devices and so on. We suppose that the communications have to be controlled well, because services using network are increasing.

Multimedia applications such as video on demand require much computer resources. Especially the allocation of CPU time is important for them to provide good services. However it is difficult to guarantee the allocation of CPU time by using conventional time-sharing scheduling mechanisms.

We aim at controlling communications in accordance with the characteristic of service contents. There are three viewpoints for the realization of the communication control.

(1) **Guarantee of communication bandwidth**
A data delivery service with deadline and a video on demand service with real-time request the guarantee of communication bandwidth. The guarantee of the data delivery service is the guarantee of the communication time. This is macrocosmic control. On the other hand, the guarantee of the video on demand service is the guarantee of the packet arrival interval. This demands the control of each packet. This is microscopic control.

(2) **Control of the interference of communication control to the other services**
It is not desirable that the communication control interferes with other uncontrolled services. The interference of the communication control should be controlled. Therefore, it is important to decide the degree of the guarantee of the communication bandwidth. That is the guarantee of other services.

(3) **Best effort with the guarantee of communication**
A best effort network can not guarantee the communication bandwidth of services. It is a signif-

icant problem that the communication looks to have halted, when there are no unused computer resources for a target service. Therefore it is necessary to guarantee the minimum performance for maintaining QoS. Besides, it is also necessary for services to be able to use resources more than the amount of guaranteed resources, if there are unused resources. We call it "best effort with the guarantee of communication".

Communication systems which satisfy above the viewpoints can be classified into three methods.

(1) **Communication protocols**
This method controls the communication flows at the level of communication protocols. An example of this method is resources reservation protocol.

(2) **Operating systems**
Operating systems decide a sending/receiving priority of packets to meet a request of services. An example of this method is a priority scheduling.

(3) **Application programs**
Service application programs control communications. This method cannot control communications well in case some services communicate simultaneously.

This paper proposes the communication bandwidth control mechanism by regulating program execution speed. We implemented the process schedule method that can regulate program execution speed. As the result, the operating system can control communication bandwidth of target processes. This paper describes the method and the implementation of it. This paper also shows how to control the communication bandwidth of target processes. Then this paper describes about an evaluation of our proposed mechanism.

2 Method for Regulating Program Execution Speed

2.1 Basic Mechanism

2.1.1 Time-slot and Time-block

A unit of allocation of CPU time is named "time-slot". CPU time is divided into time-slots. A chunk of time-slots is named "time-block". Figure 1 shows the relation between time-slots and time-blocks. A program is executed for allocated time-slots in a time-block in order to regulate program execution speed. In figure 1, one time-block is divided in to 6 time-slots, two of which are allocated to a process in the time-block. The process is regulated to 33% of bare processor performance. The program execution speed is the ratio of

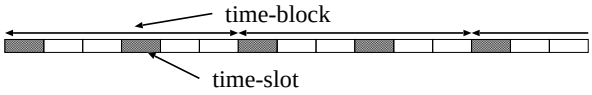

Figure 1. time-slot and time-block.

Table 1. System-call form.

Format
set_power(pid, n)
Function
Pid is process identifier. If pid equals 0, pid is identified with a current running process. When n is from 1 to 100, the process runs with the n% of bare processor performance. When n is from -255 to 0, n is priority.

number of allocated time-slots to the number of time-slots in a time-block, where the processor bare performance is defined as 100%.

2.1.2 Methods for Allocating Time-slots

The degree of regulated program execution speed can be evaluated by the execution time of a program. However the execution time is not enough to evaluate it, because the execution time is calculated by the start time of the execution and the termination time of the execution. If the behavior of the processing is not smooth, the *process* can not provide a good QoS. It is necessary to allocate time-slots to a process cyclically in order to execute a program smoothly.

We proposed the new cyclic like assignment method[1]. In case n time-slots are allocated, *(i-th allocated time-slot position) = (the total number of time-slots in a time-block) * (i–1) / n*. The method cannot always allocate time-slots cyclically, but the allocation of the time-slots is close to the cyclic allocation. Therefore the method can provide the best uniformity of processing in our proposed methods.

2.1.3 Strategy of Allocating Time-slots

Time-slots are allocated when a non-regulated process is set a program execution speed. Time-slots are also allocated when a program execution speed of a regulated process is changed. The format of a system-call for setting program execution speed or priority is shown in table 1. The system-call set_power demands a process identifier. It also demands performance or priority.

2.1.4 Process Dispatch and Coexisted Process Scheduling Mechanisms

Two kinds of processes coexist in our proposed system. One is regulated process. The other is non-regulated process. In order to schedule these processes, two kinds of scheduling mechanisms (the scheduling mechanism for regulating program execution speed, and a priority scheduling mechanism) coexist. When a regulated process is executed, the regulated process is scheduled by allocated time-slot positions in a time-block. When a non-regulated process is executed, the non-regulated process is scheduled by priority. A regulated process has a value of the degree of performance adjustment. The degree of performance adjustment is the ratio (%) of bare processor performance. The priority scheduling mechanism for non-regulated processes is similar to time-sharing scheduling mechanisms.

The scheduler is called by timer interrupts on the boundary of time-slots except for preemption. The scheduling mechanism for regulating program execution speed runs first in two scheduling mechanisms. The scheduler checks whether the next time-slot is allocated to a regulated process. If the next time-slot is allocated to a regulated process, the scheduler checks the state of the regulated process. When the state of regulated process is READY or RUN, the regulated process starts to run or continues to run. If the state of the regulated process is WAIT or the next time-slot is not allocated to any processes, a non-regulated process is selected to run by priority. When the state of a running regulated process becomes WAIT before it uses up the current time-slot, a non-regulated process is selected by priority for using the remains of the current time-slot.

3 Communication Bandwidth Control Mechanism by Regulating Program Execution Speed

3.1 Requirement

We aim at developing the mechanism that can control communication bandwidth of programs freely. We use the process schedule method that is described in the previous chapter.

There are two requirements for communication bandwidth control.

(1) Communication bandwidth can be regulated in accordance with the requested performance.

(2) The arrival intervals of each packet can be regulated.

We did experiments to evaluate whether our proposed mechanism fills the requests. We report the results of them.

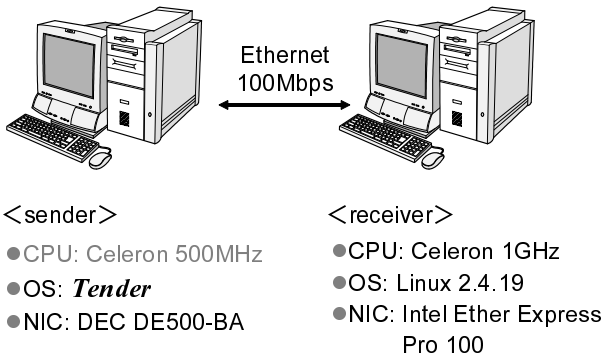

Figure 2. Environment of experiments.

Table 2. Shell command form for setting performance.

Format
exec <arg1> <arg2>
Function
"Exec" is a shell command name. "Arg1" is a processor performance. The performance is an integer from -255 to 100. When the integer is larger than 0, the integer represents the n% of bare processor performance. When the integer is smaller than 1, the integer represents priority. "Arg2" is a command name that is regulated.

3.2 Environment of Experiment

We measure communication time and the arrival intervals of each packet. The measurement was performed on two PC and 100Mbps Ethernet switching hub. A sending program is executed on the computer (Processor: Celeron 500MHz, OS: Tender[2], NIC: DEC DE500-BA). The program sends 1MB data by send system-call. A receiving program is executed on the computer (Processor: Celeron 1GHz, OS: Linux 2.4.19, NIC: Intel Ether Express Pro 100). The program receives 1MB data by recv system-call. A regulated performance (%) and the sending data size of a send system-call were changed on the sending computer. The receiving data size of recv system-call was changed on the receiving computer. The arrival time of each packet was measured on the Ethernet driver of the receiving computer. We implemented a function that can record the receipt time of each packet in the driver. The length of a time-slot is 1 msec and the length of a time-block is 1 sec in the sending computer.

We implemented a shell command that can set the performance of a regulated process. The format of the shell command is shown in table 2. Figure 3

```
On the sending computer:
    % exec 10 send_program 2048 512

On the receiving computer:
    % receive_program 2048 512
```

Figure 3. How to execute a program on the shell.

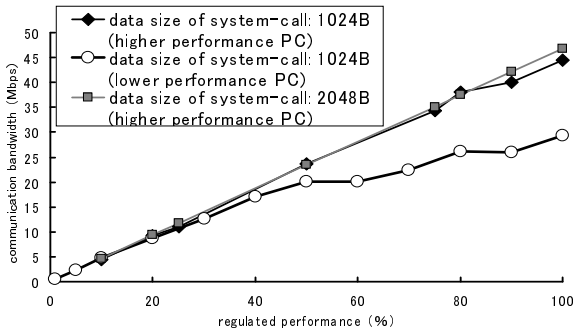

Figure 4. Regulated communication bandwidth.

shows that how to execute a send_program and a receive_program. The sender_program is regulated. In the figure 3, the performance of send_program is 10% of bare processor performance. The receive_program is set a priority. The first argument is the sending/receiving data size of the system-call. The second argument is the number of sending/receiving data.

3.3 Result of Experiment

3.3.1 Communication Bandwidth Control

In addition to the described environment, the experiment is performed on the lower performance computer (Sender: Pentium 133MHz, OS: ***Tender***, NIC: DEC DE500, Receiver: Pentium III 750MHz, OS: BSD/OS 3.1, NIC: DEC DE500). The result of them is shown in Figure 4. Figure 4 shows that our proposed mechanism can regulate the communication bandwidth on the higher performance computer irrespective of the sending and receiving data size. On the other hand, figure 4 shows that our proposed mechanism cannot regulate the communication bandwidth on the lower performance computer well, because the processor performance of the sending computer is too low. We can guess that the computer cannot utilize the performance of the NIC sufficiently.

3.3.2 Packet arrival interval

The frequency distribution of the arrival interval of each packet is shown in Figure 5 to Figure 12. The

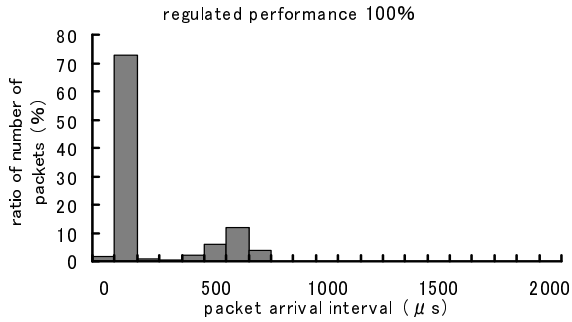

Figure 5. Packet arrival interval (data size of system-call: 1024B).

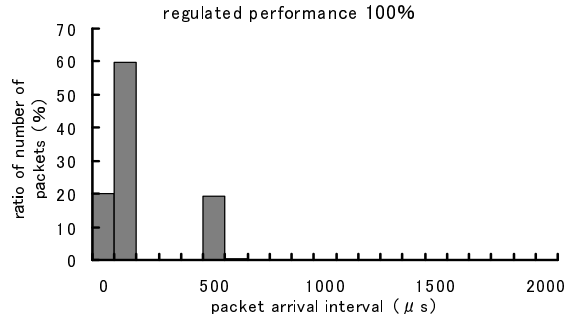

Figure 6. Packet arrival interval (data size of system-call: 2048B).

figures show the things mentioned below.

(1) In case the size of sending and receiving data is 2048B, 20% of the packet arrival interval is less than 100 microseconds. On the other hand, in case the size of sending and receiving data is 1024B, only 2% of the packet arrival interval is less than 100 microseconds.

(2) There are packets those arrival intervals are more than 400 microseconds in both of cases. Besides, the distribution of packet arrival interval of each case is different from the other case.

(3) In case the regulated performance is lower than 100%, the packet arrival interval of some packets becomes long.

The items described above are discussed below. We explain the item (1) by using table 3. The table 3 shows the number of packets in case regulated performance is 100%. The packets are divided in case the size of sending and receiving data is 2048B. As the result, 504 packets were divided to size that is more than 400B and is less than 500B. The arrival interval of these packets is short, because the size is small.

Table 3. Number of packets (regulated performance 100%).

Size of data	Arrival interval	size of a packet (Ethernet frame) (N * 100B)															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1024B	0 — 100μs	0	2	0	0	3	0	2	1	0	0	29	0	0	0	0	0
1024B	100 — 200μs	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1595
2048B	0 — 100μs	0	3	0	0	504	0	3	0	0	0	0	0	0	0	0	0
2048B	100 — 200μs	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	1532

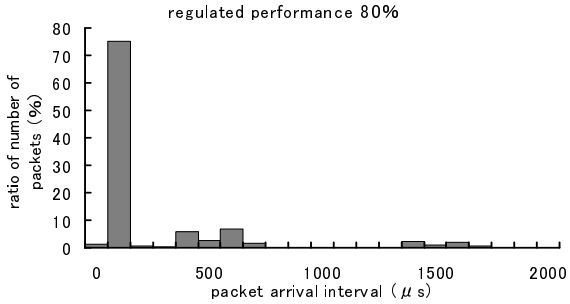

Figure 7. Packet arrival interval (data size of system-call: unit 1024B).

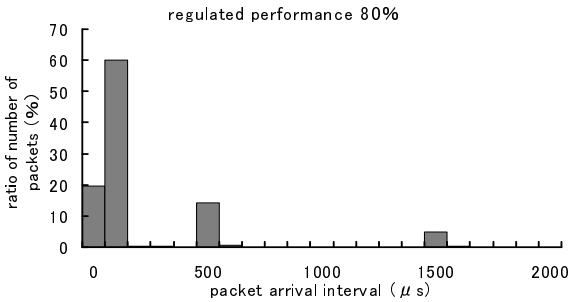

Figure 8. Packet arrival interval (data size of system-call: 2048B).

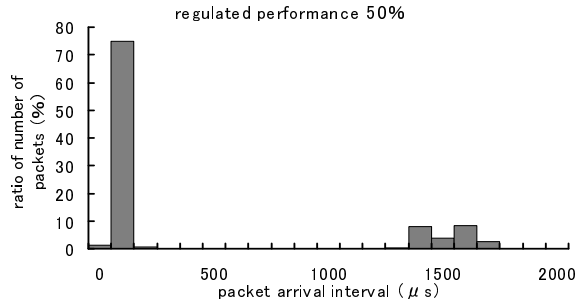

Figure 9. Packet arrival interval (data size of system-call: 1024B).

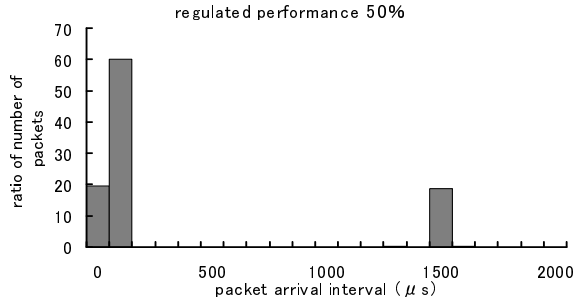

Figure 10. Packet arrival interval (data size of system-call: 2048B).

We explain the item (2). The processing of sending packets on the NIC is the repetition of sending a packet and buffering the next sending data. Several packets are sent at a time. Thus the arrival of packets those are sent after the buffering delays.

We explain the item (3). The cause of item (3) is the existence of non-allocated time-slots. Non-allocated time-slots exists in case regulated performance is below 100%. While non-allocated time-slots, the sending program is suspended. For example, time-slots are allocated on every other time-slot in 50% of regulated performance. The program is suspended on every 1 millisecond. As the result, the packet arrival time of some packet delays 1 millisecond. Two or more sequential non-allocated time-slots exist in less than 50% of regulated performance. In this case, the arrival time of some packets delays n millisecond. The n is the number of sequential non-allocated time-slots.

Figure 7 and figure 8 show that the arrival time of the certain proportion of the packets delayed. The proportion depends on the probability of the allocation of the next time-slot to a regulated process. In the figure 7, the probability of allocation of the next time-slot is 1/4, because a non-allocated time-slot exists allocated every four time-slot. Thus, 3/4 of packets whose arrival time is more than 400 microseconds in case regulated performance is 100% delay 1 millisecond.

It follows from what has been described thus far that the arrival time of the packets of some packets

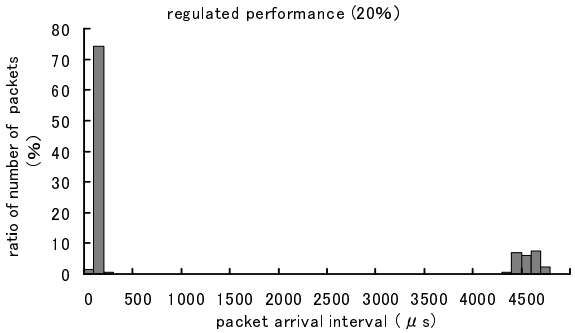

Figure 11. Packet arrival interval (data size of system-call: 1024B).

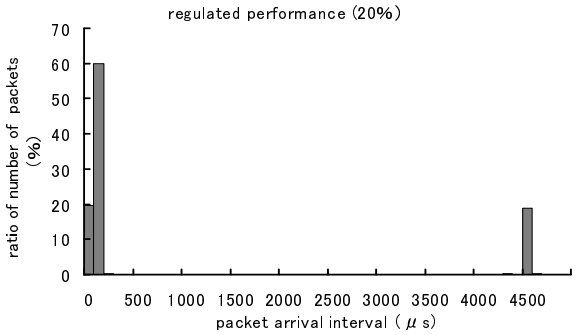

Figure 12. Packet arrival interval (data size of system-call: 2048B).

delays in proportion to the number of sequential non-allocated time-slots. The proportion of delayed packets depends on the probability of allocation of the next time-slot to a regulated process.

4 Related Work

Resource reservation protocol (RSVP) is one of approaches which use a reservation protocol. RSVP can guarantee QoS of data streams by using resource reservation. Jeong et al.[3] proposed an adaptive method of combining error resilient video packetization, FFC, and QoS controlled networks such as IntServ and DiffServ.

Tobe et al.[4] proposed software traffic management architecture for multimedia flows over a real-time microkernel. This architecture defines interface between network driver and user thread. It realizes admission control and the regulation of the rate of traffic flows. Our mechanism manages the CPU time using process scheduler on a sending host.

In order to guarantee QoS and real-time on operating systems, several studies have been made on process scheduling mechanisms. A feedback-driven approach is used for real-time application[5]. The proportional scheduling[6] presents the relative execution rates of computation as degree of CPU allocation. Our proposed mechanism presents the rate to bare processor performance as the degree of the allocation of CPU time.

5 Conclusion

We proposed the communication bandwidth control mechanism by regulating program execution speed. speed. The proposed mechanism guarantees the allocation of requested CPU time to target processes. The proposed mechanism guarantees required communication rates of target processes by allocating requested CPU time. We explained the evaluation of it. The results of it show that the our proposed mechanism can regulate communication bandwidth in accordance with regulated performance. The results also show that the arrival time of the packets is delayed in proportion to the number of sequential non-allocated time-slots. The results show that the proportion of delayed packets depends on the probability of allocation of the next time-slot to a regulated process. As a future work, we will analyze the relation between regulated performance and packet arrival time in detail.

References

[1] T. Tabata, H. Taniguchi, and K. Ushijima. Implementation and evaluation of multiple processes control mechanism for regulating program execution speed. In *Proceedings of International Symposium on Principles of Software Evolution (ISPSE 2000)*, pages 315–319, 2000.

[2] H. Taniguchi, Y. Aoki, M. Goto, D. Murakami, and T. Tabata. ***Tender*** operating system based on mechanism of resource independence. *IPSJ Journal*, 41(12):3363–3374, 2000 (in Japanese).

[3] J. Jeong, J. Shin, and D. Y. Suh. Quality enhancement of video services over qos controlled networks. *IEICE Trans. commun.*, E86-B(2):562–571, 2003.

[4] Y. Tobe, Y. Tamura, and H. Tokuda. Software traffic management architecture for multimedia flows over a real-time microkernel. *IEICE Trans. commun.*, E82-B(12):2116–2125, 1999.

[5] D. C. Steere, A. Goel, J. Gruenberg, and D. McNamee. A feedback-driven proportion allocator for real-rate scheduling. In *OSDI'99: USENIX Association 3rd Symposium*, pages 145–157, 1999.

[6] C. A. Waldspurger and W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *OSDI'94: USENIX Association 1st Symposium*, pages 1–11, 1994.