Adaptive QoS Control Adjusting Receive Buffer Sizes and Parallel TCP Connections on Information Gathering Server

Yasutaka Sakajiri*, Yosuke Tanigawa*, and Hideki Tode*
* Department of Computer Science and Intelligent Systems,
Graduate School of Engineering, Osaka Prefecture University
1-1, Gakuen-cho, Naka-ku, Sakai-shi, 599-8531, JAPAN
Email: { sakajiri@com., tanigawa@, tode@ } cs.osakafu-u.ac.jp

***Abstract*—In this paper, we focus on an information gathering service like human sensing systems and supervisory system for nursing care, where a server collects information from many clients at the same time, under the current IP network following so-called Best-Effort Discipline. In such applications, especially, communication quality should be enhanced in order to attain application-specific target throughput at maximum. To tackle this technical issue, we propose a method which adaptively tunes TCP socket buffer size and the number of TCP connections according to the policy which preferentially ensures target QoS for high priority clients. This is an unprecedented research about many-to-one transport protocol which has functions to coordinate between multiple sessions. In addition, the proposed method is suitable for the system which transports information to a server considering various network access and priority levels. Through extensive simulation experiments, we demonstrate that proposed methods can ensure the required communication quality within physical limitations according to priorities, regardless of TCP versions.**

I. INTRODUCTION

In the current Internet, sufficient communication bandwidth is not always allocated to individual users in high load environment, and as a result, satisfactory communication quality (QoS) is not ensured. In particular, for clients which require urgent network access or have poor communication quality like wireless environment, it is essential to support a certain level of QoS.

Current communication service is classified into the following 3 forms.

(1) one-to-one communication (including P2P)
(2) service for many clients to acquire information from a server
(3) service for many clients to send information to an information gathering server

Although (3) has attracted little attention in the current IP network, in near future, (3) will become important in a service like human sensing systems[1] over the wide area Internet and surveillance systems for nursing in which a server gathers information of many clients (human sensing information) distantly located over the IP network at the same time, and a doctor diagnoses based on the information.

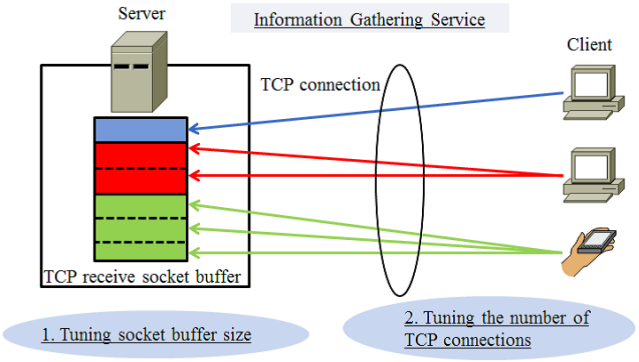

Fig. 1. Schematic diagram of the proposed method

In this paper, we focus on the above information gathering server, and we aim at throughput improvement of specific TCP sessions to an information gathering server from clients who need to ensure minimum communication quality, with coordination control among these multiple sessions. This differs from existing approaches that independently apply FEC (Forward Error Correction) [2], error control[3], congestion control[4], and flow control[5] to each connection, for quality improvement. In the proposed method, that must be applied to the current IP network, we perform the coordination control at the transport layer, in other words, at End-to-End (server-client) level because this approach does not need any improvement of routers and has high compatibility with software implementation in principle. Specifically, we use two adaptive tuning approaches of TCP socket buffer size and the number of TCP connections (Fig. 1).

In this research, as a main target, we assume the application to telediagnosis systems using human sensing information including heartbeats of fetuses and selective monitoring systems using patient pictures for remote care. However, the proposed method not only contributes to above medical care systems, but also enables to realize high quality communication by differentiating multiple sessions to the maximum in various information gathering applications.

978-1-4799-0537-9/13/$31.00 ©2013 IEEE

II. Related Works

As for buffer size tuning, various studies are conducted [6][7][8]. However, these existing studies target information distribution servers such as Web servers that transfer data towards the opposite direction to the information gathering server. In addition, these methods perform only the coordination according to entire throughput maximization, and the priority of each connection and its corresponding target throughput derived from the application's specification are not considered. Moreover, as is case with tuning of the number of TCP connections [9][10], GridFTP-APT[11] utilizes GridFTP for the coordination of the number of TCP connections, but this maximizes total throughput without taking the achievement of the target throughput into account. As described above, existing methods are designed to maximize the use of network bandwidth in each session where data are transferred from a server to many clients. On the other hand, this proposed method achieves target throughput which is set for every client according to its priority.

III. Proposed Method

The proposed method decides TCP socket buffer size for each client on an information gathering server and the number of parallel TCP connections which are assigned to each client. This is based on the policy that ensures individual target throughput according to the priority of each client.

A. Tuning Socket Buffer Size

In TCP, available buffer size on a destination host is advertised to the corresponding source host as the advertised window, and the source transmits packets continuously within minimum value of the available receive buffer size and congestion window value. Hence, if buffer size is small, transmission rate decreases. On the contrary, if buffer size is too large, the number of packets transmitted without acknowledgements increases, and congestion on intermediate routers occurs easily. Thus, it is necessary to allocate optimal size of buffer to each client considering the above factors.

In general, optimal socket buffer size is decided by BDP (bandwidth delay product). Hence, buffer size corresponding to BDP is required at minimum to utilize an available bandwidth effectively.

In the proposed method, to satisfy target throughput, socket buffer size W of each client is decided by the following formula with target throughput T_t and round trip time RTT.

$$W[KB] = T_t[Mbps] * RTT[ms]/8 \quad (1)$$

Buffer size which is calculated by eq. (1) is assigned to each client. If there are remaining buffer resources after the assignment, the surplus buffer space is assigned to every client according to the ratio of target throughput. On the other hand, if the actually assigned buffer size can not satisfy the above size W in all sessions, buffer size for each client is calculated under the condition that target throughput for all clients is set to the minimum target value in all clients. Then, buffer size is allocated again. After the allocation, if buffer volume is still left, the surplus buffer space is allocated in the decreasing order of priority. Otherwise, if the buffer size to satisfy the minimum target throughput cannot be ensured for all clients, total buffer space is divided depending on the ratio of target throughput of each client. This procedure to tune TCP socket buffer size is conducted after measuring RTT of all clients every fixed time (0.5s).

B. Tuning the Number of TCP Connections

By using multiple TCP connections in a session, higher throughput is expected. For example, in congestion avoidance phase, window-flow control is implemented based on AIMD (Additive Increase Multiplicative Decrease). Thus, the increasing rate of congestion window size among aggregated connections is improved and more data can be transmitted efficiently. However, on the contrary, when the number of aggregated TCP connections is too large, throughput decreases. This is because window size per TCP connection becomes small, and time out occurs frequently. Moreover, on a server which gathers information from many clients, the overhead to process TCP protocol stack becomes heavy. Therefore, the optimal number of aggregated TCP connections should be decided depending on network condition.

The proposed procedure to tune the number of TCP connections is described below. First, the number of connections is initialized by using a parameter γ. Throughput is calculated per client every fixed time interval (0.5s), and the proposed tuning is conducted when current throughput is lower than both of target throughput and the estimated throughput which is calculated with eq. (2) multiplied by safety coefficient ξ. Preliminary experiment confirmed that the settings of the above two thresholds stabilized throughput by decreasing the frequency of the tuning. Equation (2) indicates an estimated throughput using buffer size and RTT in case of buffer size becoming bottleneck of data transmission. T_e, W_n, and RTT_n are defined as estimated throughput, current buffer size, and current round trip time respectively.

$$T_e[Mbps] = W_n[KB]/RTT_n[ms]/8 \quad (2)$$

Procedure to tune the number of parallel TCP connections is described as the following steps (I) to (VI). Where, coefficients of additive increase and multiplicative decrease are defined as $\alpha(\geq 0)$ and $\beta(0 < \beta < 1)$, respectively. T_t, T_n, N_n, and N_b are defined as target throughput, current throughput, current number of connections, and the number of connections before the last tuning, respectively.

(I) Calculate throughput. If it satisfies target throughput, algorithm is finished. N_n at that point is kept as "the number of TCP connections".
(II) Initialize the number of connections by multiplying N_n. α and β are initialized by using eqs. (3) (4).
(III) Add α to the number of connections N_n.
(IV) Calculate throughput. If it satisfies target throughput, N_n is determined as "the number of TCP connections", and algorithm is finished. On the other hand, if the target throughput is not satisfied, in case of $\alpha = 1$, throughput

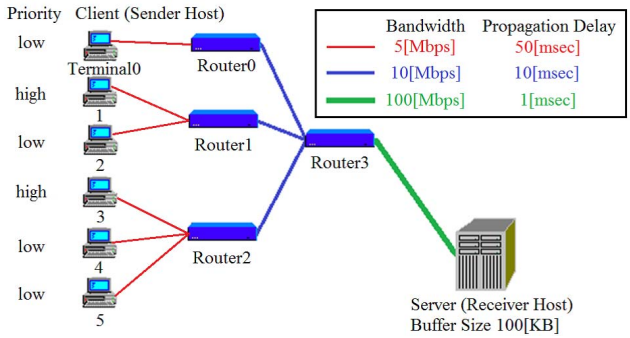

Fig. 2. Simulation model 1 (Model 1)

which was increasing up to now turned to decrease, N_b is determined as "the number of TCP connections", and algorithm is finished. In case that throughput is increasing (closing to optimal number), proceed to (VI), and in case that throughput is decreasing (passed over optimal number), proceed to (V).

(V) Multiply the number of connections N_n by β.

(VI) Renew α and β by eqs. (3) (4). Return to (Ⅲ).

$$\alpha \leftarrow max(1, [T_t/T_n * RTT/RTT_{std}]) \quad (3)$$

$$\beta \leftarrow max(N_b/N_n, T_n/T_t) \quad (4)$$

RTT_{std} in eq. (3) is baseline of RTT for normalizing RTT value, and there is an aim that influence of RTT on α becomes comparable to the one of T_t/T_n (dimensionless). In the following, RTT_{std} is set to 100[ms].

IV. Efficiency Evaluation

A. Simulation Model and Assumption

We evaluate the effectiveness of the proposed method by extensive computer simulation using ns-2.35.

First, we evaluated the proposed method with the simulation model described in Fig. 2. Data are transferred from 6 clients to the server. We denote the parameters that are used for this simulation in TABLE I as default values. Unless otherwise noted, we use these parameters.

TABLE I
Parameter set-up for simulation

TCP congestion control	TCP Reno
Target throughput of high priority client	1.0[Mbps]
Target throughput of low priority client	0.5[Mbps]
Receive socket buffer size on the server	100[KB]
γ (initialization parameter)	0.5
ξ (safety coefficient)	0.95

Values of γ and ξ in TABLE I are recommendation values obtained by preliminary experiment.

B. Comparative Performance Evaluation

Figure 3 evaluates the proposed method (proposal) and the compared method (comparison). In the compared method, maximum throughput of each session is estimated by RTT, the ratio of packet discarding, and time-out value based on [8]. In addition to the throughput estimation, entire buffer size on a server is assigned to every client according to ratio of

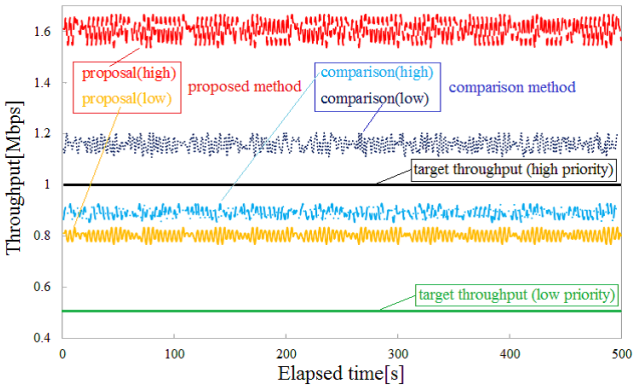

Fig. 3. Temporal change of throughput (Model 1)

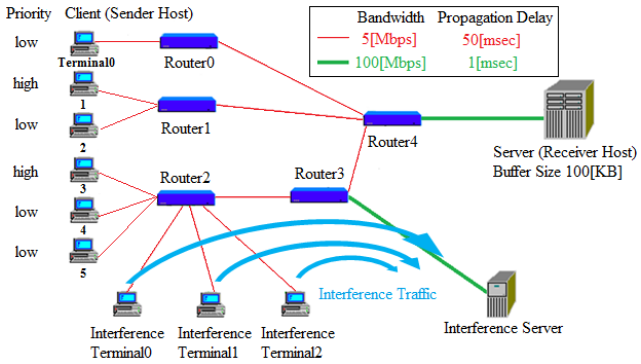

Fig. 4. Simulation model 2 (Model 2)

estimated throughput and the number of TCP connections is set to one. In this evaluation, average throughputs of high and low priorities are compared.

In Fig. 3, the proposed method satisfies target throughput with a margin in both high and low priority clients. On the other hand, in the compared method, the throughput of high priority clients becomes lower than the target value. Though the throughput of low priority clients in the proposed method is lower than that in the compared method, this has no problems because low priority clients require only their target throughput of 0.5[Mbps]. In the compared method, priority is not considered, and large buffer size is assigned to a client which has high available bandwidth even if its priority is low. As a result, like Fig. 3, it may occur that the throughput of low priority client is higher than that of high priority client.

Next, as described in Fig. 4, we generate interference traffic between Router 2 and 3. The interference traffic is transferred from each of interference Terminal 0,1,2 to interference Server and generated during 100-500[s] at simulation time.

Figures 5 and 6 show temporal change of throughput from every terminal in the environment of Fig. 4. In Fig. 5, only the receive socket buffer tuning is applied. In Fig. 6, both the buffer size and the number of TCP connections are tuned. In addition, to simplify the graph, only the results of Terminal 0 and Terminal 4 are described because the results of Terminal 2 and 5 are almost same as those of Terminal 0 and 4, respectively.

From Fig. 5, in case of applying only receive socket buffer

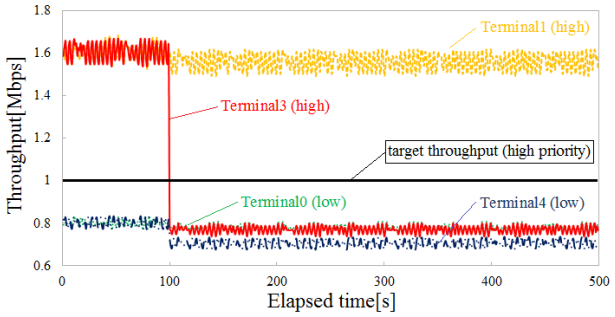

Fig. 5. Temporal change of throughput (Only buffer size tuning) (Model 2)

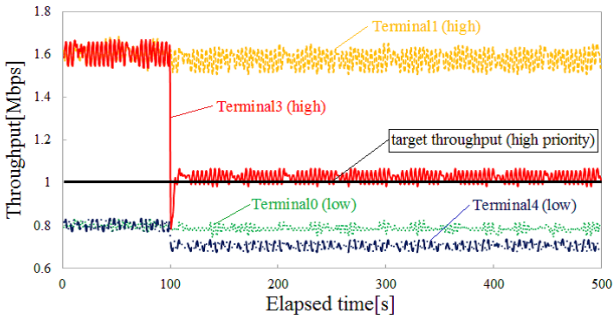

Fig. 6. Temporal change of throughput (Buffer size tuning + The number of TCP connections tuning) (Model 2)

coordination function, it is perceived that communication of Terminal 3 which is high priority client is lower than target throughput. On the other hand, in case of Fig. 6 where the number of TCP connections is also tuned, Terminal 3 satisfies 1.0[Mbps] of the target throughput even after generating interference traffic.

It is considered that target throughput is not satisfied only by coordinating buffer size simply, because available bandwidth of each connection becomes low by many connections sharing the same line when interference traffic is generated. On the other hand, the proposed method maintains target throughput even if network is congested to some extent because of enabling to get higher bandwidth than other communication by coordinating the number of TCP connections.

C. Differences of TCP Versions

We discuss on differences of TCP versions. Specifically, we evaluated the throughput performance of the proposed method under 4 kinds of TCP versions; CUBIC, Compound TCP, Vegas, and Reno in simulation model 1. TABLE II shows the average throughput of each TCP version. Figure 7 shows temporal change of throughput in CUBIC which is highest efficiency, and Reno which is lowest efficiency.

TABLE II
AVERAGE THROUGHPUT OF EACH TCP VERSION.

TCP version	throughput (high priority)	throughput (low priority)
CUBIC	1.67[Mbps]	0.83[Mbps]
Compound TCP	1.66[Mbps]	0.83[Mbps]
Vegas	1.62[Mbps]	0.81[Mbps]
Reno	1.60[Mbps]	0.80[Mbps]

From Fig. 7, target throughput is satisfied in all TCP versions. However, throughput of new TCP versions (CUBIC and Compound TCP) is higher than the one of old TCP versions (Reno and Vegas). It was verified that the proposed method is efficient regardless of TCP versions.

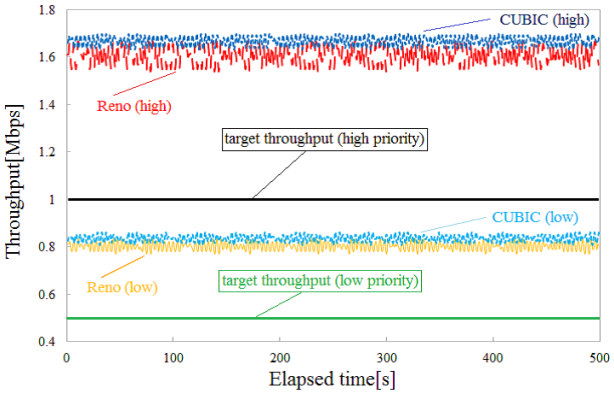

Fig. 7. Temporal change of throughput (CUBIC and Reno) (Model 1)

V. CONCLUSION

In this research, we focus on the information gathering server which collects information from many clients, and proposed two adaptive coordination methods of TCP socket buffer size and the number of TCP connections among multiple sessions. These methods are based on the policy that preferentially ensures target QoS for higher priority clients. The proposed coordinations are performed at End-to-End level, which has high feasibility. Extensive evaluations verified that the proposed method can ensure QoS which applications require according to priority of clients, regardless of TCP versions. As for future task, we will validate the utility of this proposed method on real machine and examine cooperation of this proposed method with FEC.

REFERENCES

[1] J. Jung, et al, "Wireless body area network in a ubiquitous healthcare system for physiological signal monitoring and health consulting," International Journal of Signal Processing and Pattern Recognition, vol. 1, no. 1, pp. 47-54, 2008.
[2] M. Mitzenmacher, "Digital fountains: a survey and look forward," Proc. the IEEE Information Theory Workshop, pp. 271-276, Oct. 2004.
[3] H. O. Burton, et al, "Errors and error control," Proceedings of the IEEE, vol. 60, pp.1293-1301, Nov. 1972.
[4] J. Widmer, et al, "A survey on TCP-friendly congestion control," IEEE Network, vol. 15, pp. 28-37, May 2001.
[5] M. Gerla, et al, "Flow Control: A comparative survey," IEEE Trans. Commun., vol. 28, no. 4, pp. 553-574, April 1980.
[6] B. L. Tierney, et al, "Enabling network-aware applications," Proc. IEEE HPDC 2001, Aug. 2001.
[7] E. Yildirim, et al, "Balancing TCP buffer vs parallel streams in application level throughput optimization," Proc. ACM DADC 2009, pp. 21-30, June 2009.
[8] T. Matsuo, et al, "Scalable automatic buffer tuning to provide high performance and fair service for TCP connections," Proc. IEEE INET 2000, July 2000.
[9] H. Sivakumar, et al, "PSockets: the case for application-level network string for data intensive application using high-speed wide area networks," IEEE Computer Society Press Article, no. 37, 2000.
[10] Z. Chen, "FTS transfer parameter optimisation," CERN openlab Summer Student Report, July 2012.
[11] T. Ito, et al, "GridFTP-APT: automatic parallelism tuning mechanism for data transfer protocol GridFTP," Proc. CCGRID 2006, May 2006.