4-2 QoS Control Method for Large Scale Data Transfer in DataGrid Applications

NORO Masaaki, BABA Ken-ichi, and SHIMOJO Shinji

Many people works about Grid service technology recently. It is difficult to adopt private circuit as wide area link for many users in Grid service environment. It is helpful for scheduler of Grid that data transfer reserves bandwidth in pipeline processing applications. On the other hand, data size of Grid has increased enormously. Large scale data transfer with bandwidth reservation should be bottleneck. Therefore, it is useful to increase numbers of data transfer within same network resource. We propose QoS control method for large data transfer over Diffserv network.

Keywords
Grid, File transfer, Diffserv, QoS control

1 Introduction

In recent years, Grid studies have formed the focus of a range of researchers, particularly those affiliated with various standards bodies[1]. As Grid services can be used with simple web interfaces, we can soon expect to see a rapid increase in the number of new users. On the other hand, service costs remain an important consideration in such an environment; for this reason, it is impractical to use the high-speed dedicated networks adopted for Grid environments in the past, as with OptIPuter[2]. It is therefore essential that we develop technology for application to general-purpose IP networks, technology that will improve the efficiency of use of the computational resources of calculation service providers as well as improve the overall operational efficiency of applications.

Grids currently process enormous amounts of data, with some applications handling several hundred megabytes of data at a time. The performance of these applications depends on data transmission through wide-area networks. For pipeline processing of such large amounts of data, the most effective technique involves performing calculations concurrently with transfer of data being performed in the background. Efficient processing of larger blocks of data in these types of application requires guaranteed file-transfer performance. However, in such cases, bandwidth shortage limits the number of files that can be processed simultaneously. To address this problem, we have proposed a method based on the Diffserv minimum bandwidth guarantee technique[3]. This article discusses the proposed method and the results of evaluations using simulation software[4].

2 Conventional methods and associated problems

When attempting to guarantee performance for file transfer within a Grid, failures in bandwidth reservation often cause re-scheduling of jobs and processing delays. Thus, requests for bandwidth reservation in file transfer must be accepted to the fullest extent possible.

2.1 Fixed bandwidth allocation method

The method of transferring data within a contracted bandwidth using the EF (Expedited Forwarding) service[6] of Diffserv[5] or CBQ (Class-Based Queuing) [7] cannot provide throughput exceeding the contracted bandwidth. Thus, in situations such as that indicated in Fig.1, the bandwidth reservation request of Flow 2 fails (call loss). However, it is common to assign large throughput to data transfer, as in GridFTP[8], so that the probability of such failures is high.

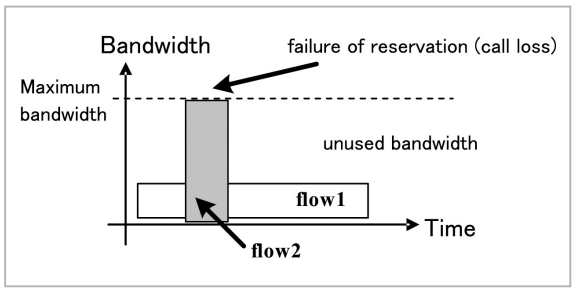

Fig.1 *Call loss due to bursting bandwidth request*

2.2 Method of minimum bandwidth guarantee

Next, we will examine a method that applies a minimum bandwidth guarantee[9] for each file transfer. With this method, each session can secure throughput exceeding the contracted amount when the network is not congested. Thus, contracted bandwidth decreases and call-loss probability is reduced relative to the fixed bandwidth method. However, this alternative method does not completely eliminate the possibility of call loss in cases similar to Flow 2 in Fig.1.

3 QoS control method for large-scale file transfer

First, we will investigate the QoS (Quality of Service) to be provided for file transfer in a distributed computation environment. If the quality assurance request for a newly generated file transfer is denied, there will be undesirable results: for example, the processing time of the job to which the file transfer is assigned (the call) will increase or the job may be reallocated to a location of reserve capacity within the network resources.

The QoS provided by the network may be specified using three indexes: jitter, delay, and packet-loss rate. However, unlike constructing a dedicated network, here the route cannot be specified in an ordinary ISP environment, and thus the QoS contract itself specifies the bandwidth.

Here, when specifying the bandwidth for file transfer, the protocol and the file size are known, thus this specification is nearly equivalent to the specification of the expected (desired) completion time for the file transfer. Further, a file transfer generates non-interactive traffic, so that the file transfer contracted with guaranteed throughput for each flow does not require constant bandwidth during the duration of the flow. The guaranteed QoS need only satisfy the conditions on average during the duration of the flow.

Given the above, the bandwidth contracted with a job in the early stages is considered the target throughput. If the system can accept a greater number of bandwidth reservation requests, using total bandwidth effectively, while at the same time securing the target throughput, the total computational resources will also be used more effectively, including available CPU resources.

3.1 Proposed method

We have proposed a method in which the throughput of each individual flow is guaranteed and the bandwidth contracted at the start of the call is reviewed as the file transfer progresses, using the minimum bandwidth guarantee service (Assured Forwarding service, or AF) of Diffserv. This method is designed to provide the following advantages:

- Minimum bandwidth is guaranteed for each flow to improve efficiency in the use of the routes.
- Required bandwidth is reduced to create empty bandwidth, increasing the possibility that a new call will be accepted.

Figure 2 shows the relationship between the throughput of the TCP (Transmission Control Protocol) flow of data transferred under this method and the contracted bandwidth. This method calculates the bandwidth necessary to achieve the user-specified throughput and then applies the calculated value to execute the contract with the minimum bandwidth guarantee service. Here, the expected completion time is calculated as well as the contracted bandwidth using the characteristics of TCP. Next, after the fixed-length data is transmitted, the bandwidth to be contracted is re-calculated from the amount of remaining data to be transmitted and the expected completion time.

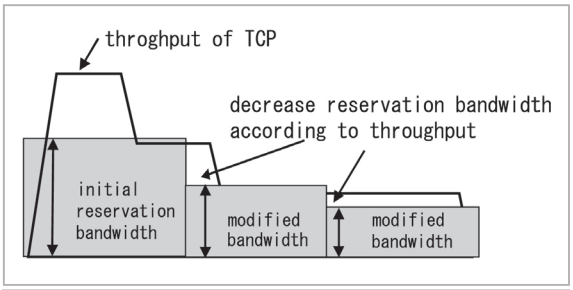

Fig.2 *Data transfer under the proposed method*

If the network is not congested, the throughput of data transfer is higher than the target throughput, so that the bandwidth to be contracted is smaller than the current value. Only when the bandwidth to be contracted is smaller than the current contract is contract the updated to increase empty bandwidth. Here, even when the throughput of data transfer is lower than the target value (in the case of an extremely congested network, for example), the contracted bandwidth is not increased. This is to prevent bandwidth reservation failure in the attempt to increase bandwidth during data transfer.

This method uses the minimum bandwidth guarantee service and modifies the contract only when reducing the bandwidth. As a result of these characteristics, the individual terminals do not need to interrupt the data transfer to signal the bandwidth contract or to synchronize actions with the other terminals. Thus, implementation is easy, and scalability is secured relatively simply.

3.2 Estimation of contracted bandwidth from target throughput

The proposed method requires calculation of both the bandwidth to be contracted and the expected completion time for the data transfer based on the target throughput. Thus, here we will discuss contracted bandwidth in an AF environment with TCP throughput. (See Reference[10] for a more detailed discussion.)

When TCP detects a packet loss, it halves the window size and then increases the window size by 1 packet per RTT (Round Trip Time). Thus, as long as packet losses are not repeated and packets are not lost within the contracted bandwidth, the window size changes as shown in Fig.3. In Diffserv AF, the marking is based on the throughput value averaged over a certain period. Thus, even if the throughput exceeds the contracted bandwidth, this does not immediately signify RED (Random Early Detection). For this reason, the peak of TCP throughput when a packet loss occurs is equivalent to or greater than the contracted bandwidth. Thus the relationship between TCP throughput, S, and contracted bandwidth, B, can be expressed as $S \geq 3B/4$ in an environment in which TCP operates in the congestion-avoidance phase and the contract is maintained within the AF service.

Next, we will consider a case in which this relationship does not hold. This situation occurs when file size is small compared to the target throughput or when TCP frequently switches to the slow-start phase from the congestion-avoidance phase due to repeated packet loss (even though the file may be of sufficient size). However, given that this method targets applications that transfer many files of several hundred megabytes or larger, we expect that such a situation will occur only rarely. Thus, the contracted bandwidth is set at 4/3 of the target throughput in the current algorithm.

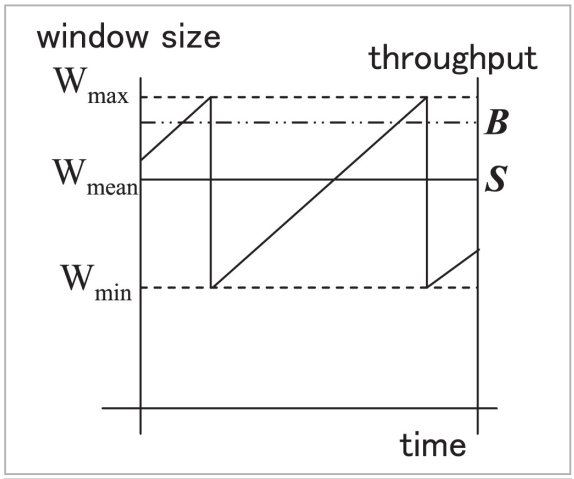

Fig.3 *Relationship between TCP throughput and contracted bandwidth in congestion-avoidance phase*

4 Evaluations

We performed NS2 simulations and evaluated the performance of the proposed method using the following parameters.

(1) Call-loss probability: The probability that the bandwidth reservation request of a newly generated file transfer (session) will fail

(2) Throughput: The amount of data actually transmitted per unit time in the total flow

(3) Expired flow: The probability that the target throughput set at the start of the flow will not be achieved

Here, the following two methods may be compared with the proposed method.

(A) Simple EF: A method that contracts bandwidth for every file transfer and executes each transfer with shaping at the edge so that the file transfer does not exceed the contracted bandwidth

(B) Simple AF: A method that contracts the necessary bandwidth with Diffserv AF for every independent file transfer and maintains the bandwidth contract unchanged until the end of the file transfer

4.1 Evaluation model

Figure 4 shows the evaluation model used in this study. Here, the parameters are as follows.

(1) Probability of file transfer occurrence: Poisson process

(2) File size: Pareto distribution with an average of 100 Mbytes

(3) Initial bandwidth requested by the file transfer: Uniform distribution between 1 Mbps and the maximum throughput

(4) Other items: In each file transfer, bandwidth is reviewed after every 20 Mbytes of transmission.

(5) Background traffic: Two cases are examined: one with no traffic and one with Pareto distribution at a maximum of 1 Gbps.

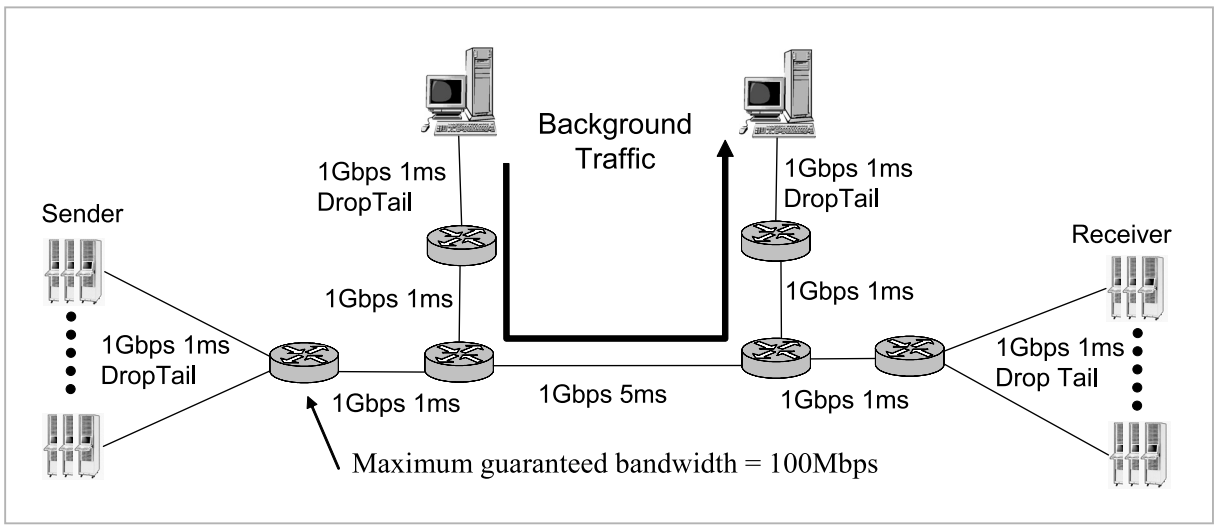

Fig.4 *Simulation model*

4.2 Scenario 1

Here we evaluate the method under a scenario in which TCP Reno is used for the file transfer. The horizontal axis of each graph represents the probability of the file transfer.

(1) Call loss probability

Figures 5 and 6 show the call loss probability with and without the background traffic, respectively. In the proposed method, each flow reduces the bandwidth it uses according to the available throughput even when the load is high, so that the system accepts new flows and the probability of call loss decreases.

(2) Throughput

Figures 7 and 8 show the total throughput with and without background traffic, respectively. The proposed method shows better performance relative to the other methods in both figures. This is because the proposed method features low call-loss probability even when the load is high, so that the system accepts more flows and consequently transfers more data.

(3) Expired flow

Table 1 shows the proportion of throughput requested by the flow that is not provided. All methods show approximately the same value, regardless of the probability of successful file transfer.

Simple EF always fails to fulfill the contract by approximately 1%. This is because file size is small compared to the requested bandwidth. On the other hand, the proposed method and Simple AF show better performance, providing throughput greater than the target value with or without background traffic.

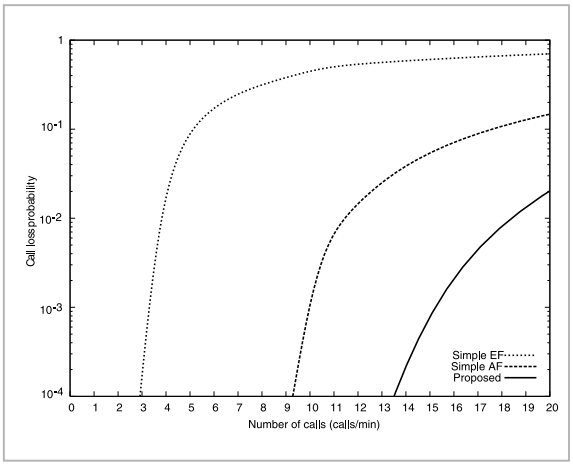

Fig.5 *Call-loss probability (without background traffic)*

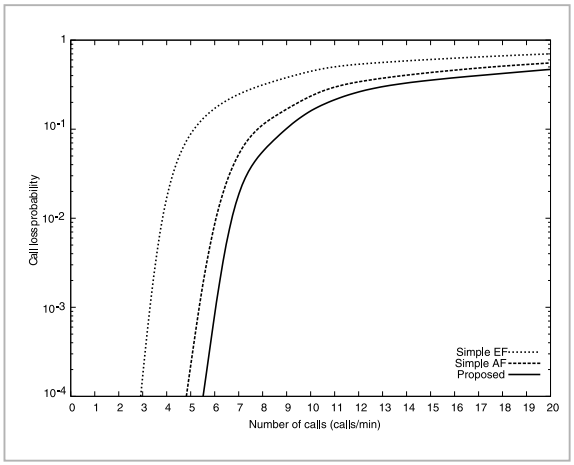

Fig.6 *Call-loss probability (with background traffic in Pareto distribution)*

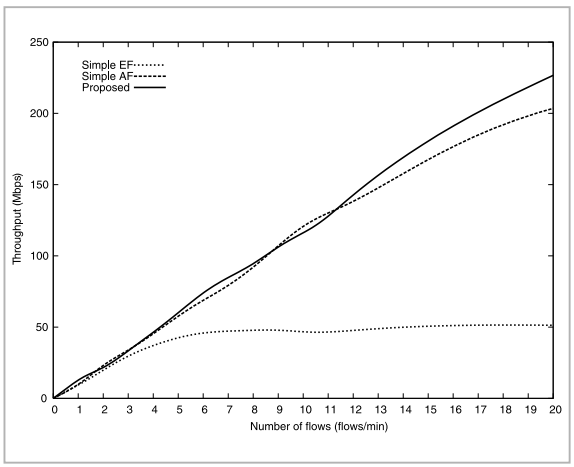

Fig.7 *Total throughput (without background traffic)*

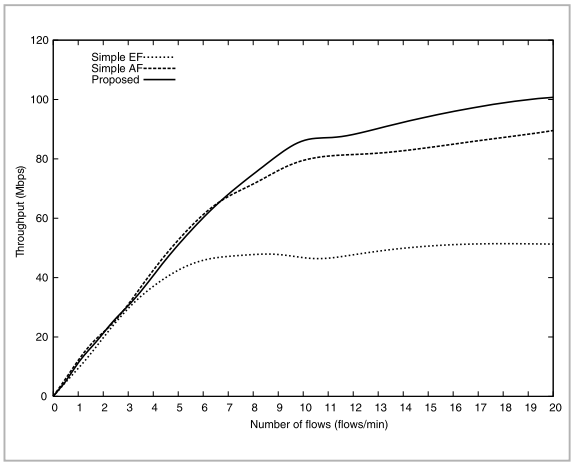

Fig.8 *Total throughput (with background traffic in Pareto distribution)*

Table 1 *Proportion of throughput below requested value*

	Background traffic		
Method	None	Pareto distribution	CBR
Simple EF	Approximately 1%	Approximately 1%	Approximately 1%
Simple AF	0%	Approximately 0.5%	Approximately 0.5%
Proposed method	0%	Approximately 0.5%	Approximately 0.5%

4.3 Scenario 2

This evaluation is based on the use of GridFTP and other software, and automatic adjustment of TCP window size according to a double or triple throughput scenario. The remaining conditions are the same as those of Scenario 1. Figures 9 and 10 show the call-loss probability, and Figures 11 and 12 show the total throughput. For both of these quantities, the proposed method and Simple AF differ only minimally when the throughput is tripled. This is because the number of TCP flows in the system is small, while TCP throughput and reserved bandwidth are large. When a large number of packets are generated with a small number of TCP flows, the probability of loss increases, and throughput decreases as a result. This works against the advantages of the proposed method.

Next, Table 2 shows the probability of the measured throughput falling below the requested throughput. For both doubled and tripled throughput, the proposed method shows performance equivalent to or better than that of the other methods.

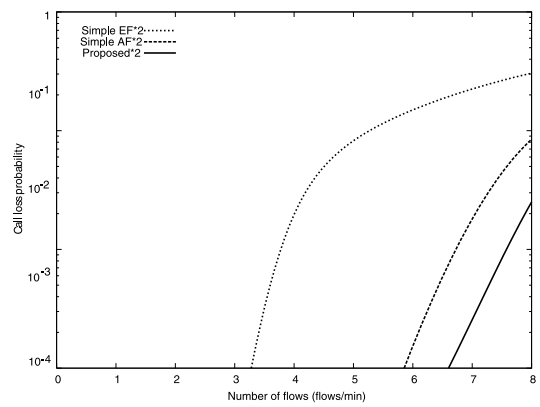

Fig.9 *Call-loss probability (doubled throughput)*

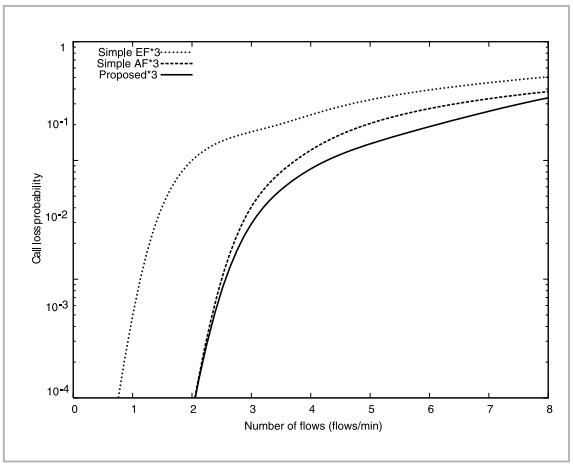

Fig.10 *Call-loss probability (tripled throughput)*

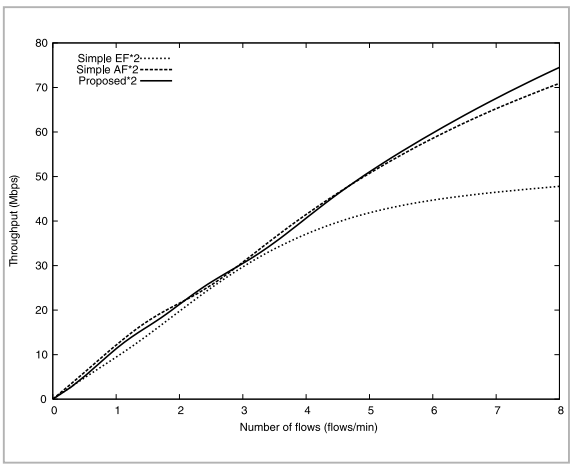

Fig.11 *Total throughput (doubled throughput)*

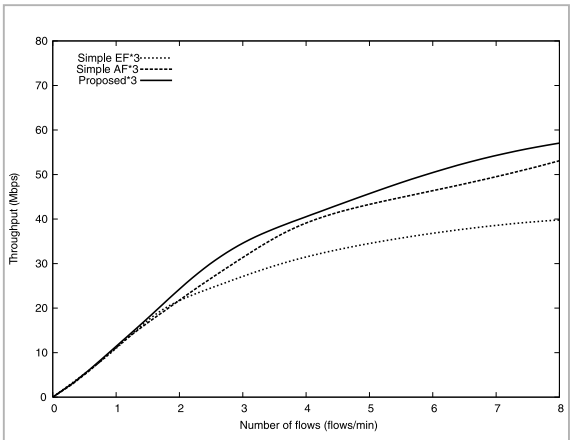

Fig.12 *Total throughput (tripled throughput)*

Table 2 *Proportion of throughput below requested value*

	Throughput per flow	
Method	Doubled	Tripled
Simple EF	Approximately 1%	Approximately 1%
Simple AF	0.5 % or less	Below 0.8 %
Pareto distribution	0.5 % or less	Below approximately 0.8 %

These results point to the necessity of combining methods so as to increase data-transfer throughput when using TCP for data transfer in an environment with few simultaneous TCP flows in the bottleneck link.

6 Conclusions

This study proposed a QoS control method for Grid applications involving large-scale file transfer, and evaluated the proposed method via simulations. When standard TCP was used, the proposed method showed better performance relative to alternative methods, both in terms of the probability of call loss and throughput with or without background traffic. The proportion of cases in which throughput was below the requested value was also equivalent to or better than the corresponding ratio yielded by the remaining methods. On the other hand, evaluation with adjusted window size revealed that the proposed method is somewhat ineffective, due to the weakness of TCP when data is transferred with a single TCP flow per session.

Future problems to address will involve evaluation of performance when the proposed method is combined with data transfer protocols to provide higher throughput in a guaranteed minimum bandwidth service environment, as well as performance evaluation in scenarios that reflect the characteristics of practical applications. These evaluations are scheduled to take place in the short term.

References

1 Global Grid Forum, http://www.gridforum.org/.

2 OptIPuter, http://www.optiputer.net/.

3 M. Noro, I. Hasegawa, K. Baba, and S. Shimojo, “QoS Control Method for Large Scale DataGrid Applications”, IEICE Technical Report, IA2004-22, January, 2005. (in Japanese)

4 VINT project, “ns2”, http://www.isi.edu/nsnam/ns/

5 S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An Architecture for Differentiated Service”, RFC2475, December, 1998.

6 V. Jacobson, K. Nichols, and K. Poduri, “An Expedited Forwarding PHB”, RFC2598, June, 1999.

7 S. Floyd and V. Jacobson, “Link-sharing and Resource Management Models for Packet Networks”, IEEE/ACM Transactions on Networking, Vol.3, No.4, pp. 365-386, August, 1995.

8 The Globus Alliance , “Grid FTP” http://www-fp.globus.org/datagrid/gridftp.html.

9 J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, “Assured Forwarding PHB Group”, RFC2597, June, 1999.

10 M. Tsuru, K. Kumazoe, and Y. Oie, “Towards High Performance TCP for Fast Long-Distance Networks”, IPSJ Magazine, Vol.44, No.9, pp.951-957, September, 2003. (in Japanese)

NORO Masaaki

Expert Researcher, Osaka JGNⅡ Research Center, Collaborative Research Management Department

Internet, Network QoS, and Grid Technologies

***BABA Ken-ichi**, Dr. Eng.*

Expert Researcher, Osaka JGNⅡ Research Center, Collaborative Research Management Departmen (Associate Professor ,Cybermedia Center, Osaka University)

Studies on Broadband Communication Network, Computer Network, and Photonic Network Systems

***SHIMOJO Shinji**, Dr. Eng.*

Expert Researcher, Osaka JGNⅡ Research Center, Collaborative Research Management Department (Professor, Cybermedia Center, Osaka University)

Focusing on a Wide Variety of Multimedia Applications, Peer-to-peer Communication Networks, Ubiquitous Network Systems, and Grid Technologies