A Design and Prototyping of In-Network Processing Platform to Enable Adaptive Network Services

IEICE TRANS. INF. & SYST., VOL.E96–D, NO.2 FEBRUARY 2013

PAPER Special Section on The Internet Architectures, Protocols, and Applications for Diversified Futures

A Design and Prototyping of In-Network Processing Platform to Enable Adaptive Network Services*

Masayoshi SHIMAMURA[†a)], **Takeshi IKENAGA**[††b)], ***and*** **Masato TSURU**[††c)], ***Members***

SUMMARY The explosive growth of the usage along with a greater diversification of communication technologies and applications imposes the Internet to manage further scalability and diversity, requiring more adaptive and flexible sharing schemes of network resources. Especially when a number of large-scale distributed applications concurrently share the resource, efficacy of comprehensive usage of network, computation, and storage resources is needed from the viewpoint of information processing performance. Therefore, a reconsideration of the coordination and partitioning of functions between networks (providers) and applications (users) has become a recent research topic. In this paper, we first address the need and discuss the feasibility of adaptive network services by introducing special processing nodes inside the network. Then, a design and an implementation of an advanced relay node platform are presented, by which we can easily prototype and test a variety of advanced in-network processing on Linux and off-the-shelf PCs. A key feature of the proposed platform is that integration between kernel and userland spaces enables to easily and quickly develop various advanced relay processing. Finally, on the top of the advanced relay node platform, we implement and test an adaptive packet compression scheme that we previously proposed. The experimental results show the feasibility of both the developed platform and the proposed adaptive packet compression.

key words: *adaptive network services, advanced relay node platform, advanced relay processing, provider-managed overlay networks, adaptive packet compression scheme*

1. Introduction

Due to explosive growth of Internet usage along with a greater diversification of communication technologies and applications [2], the Internet is currently facing serious problems [3] and approaching a turning point. In particular, the Internet is required to achieve large-scale broadband communication, diversifying communication, and harmony with human society and activities. However, the current Internet and TCP/IP protocol suites cannot precisely meet these requirements, which lead to a degradation of communication performance and reliability as well as serious security problems and an increase in network management costs. To address these issues, the Internet must meet the following requirements: (1) dynamic resource optimization by distributed controls, especially adaptive resource allocation taking advantage of diversifying environments, (2) appropriate interaction and interfaces between networks and human society, and (3) flexible, efficient, and sustainable architectures and frameworks to adapt for future environments. Therefore, the relationship between networks (providers) and applications (users) has been reconsidered and discussed in various projects [4]–[6].

Regarding diverse environments, TCP/IP had attained success as a global communication infrastructure among a wide variety of applications by hiding diversification. This is known as the Internet hourglass model [7]. However, in situations where the implicit assumption on TCP/IP is not satisfied, cross-layer approaches are used to solve performance degradation issues. To deal with these issues comprehensively, schemes taking advantage of diversification are required in cooperation between networks (routers, middlebox, resource management servers) and users (terminals, applications) such as Refs. [8]–[10]. In general, traffic control inside networks can adapt to the diversity of networks, can react depending on the network situation, and can provide priority and/or fairness controls among users. On the other hand, traffic control by end hosts can adapt to the diversity of terminals and applications, can react to their situations, and can predict and control data transmission. By taking advantage of these diverse environments, distributed control schemes for hierarchical network structures must be achieved.

In this direction, application-layer overlay networks have emerged as a means to add newly required functions to underlay networks, i.e., IP networks, which can provide the transfer of data via other overlay nodes, even if the data can be transferred by underlay networks directly. In overlay networks, applications transfer data via application nodes located at the edge of an underlay network, i.e., Internet service provider (ISP) network, hence they may cause negative effects on underlay networks and applications themselves if the applications do not consider underlay information (routing, link bandwidth, congestion degree). Furthermore, applications are fundamentally greedy and selfish, so the optimization and adjustment of massive numbers of applications are difficult. To solve these issues, several schemes have been studied to control the behavior of applications [8]–[15]. However, to meet the diversifying needs of applications and users, networks themselves must be fundamentally changed and sophisticated.

Manuscript received June 5, 2012.
Manuscript revised October 1, 2012.

[†]The author is with the Global Scientific Information and Computing Center, Tokyo Institute of Technology, Tokyo, 152–8550 Japan.
[††]The authors are with the Network Design Research Center, Kyushu Institute of Technology, Kitakyushu-shi, 804–8550 Japan.

*An earlier version of this work has been presented in NTCAA-2010, November 2010 [1].

a) E-mail: shimamura@netsys.ss.titech.ac.jp
b) E-mail: ike@ecs.kyutech.ac.jp
c) E-mail: tsuru@cse.kyutech.ac.jp

DOI: 10.1587/transinf.E96.D.238

To achieve efficient network-resource utilization inside networks, which requires the dynamic resource optimization addressed at the beginning of this section, we consider that conventional simple packet-relay processing at intermediate nodes must be reconsidered, and advanced relay processing that adapts to the network's status is needed. Future networks will contain various types of heterogeneous networks, including virtual overlay networks using network virtualization technologies [16]–[18], wireless mesh backbone networks using a variety of high-speed wireless technologies [19], [20], and wireless sensor networks [21]. Therefore, networks need to be adaptive, and relay processing needs to adapt to such networks. In general, traffic control inside networks can be more timely, fine-grained, and globally optimal than traffic control on an end-to-end basis, although network-internal control may be costly to implement. If applications can improve their performance by the use of advanced relay processing, they can obtain an incentive to cooperate with such networks.

In this paper, we first discuss the need and feasibility of flexible and adaptive network services by introducing in-network processing nodes inside the network, as a background of our research direction. Then, we design an advanced relay node platform for the purpose to experimentally evaluate the feasibility of diverse adaptive network services. A key feature of the proposed platform is that integration between kernel and userland spaces enables to easily develop various adaptive network services with advanced relay processing at a userland space using related information in a kernel space. As an example of adaptive network services, using the advanced relay node platform, we show the feasibility of the adaptive packet compression scheme proven to be effective in the previous study [22].

The remainder of this paper is organized as follows. Section 2 describes advanced relay and its related work. Section 3 explains provider-managed overlay. Section 4 mentions advanced relay nodes. Section 5 shows our experimental evaluation. Finally, Sect. 6 concludes by briefly summarizing the main points.

2. Advanced Relay

Networks, meaning horizontal (spatial) networks, are responsible for relaying information data (i.e., transferring IP packets in the Internet) in general, but relay functions have a wide variety of objectives and utilities. Basically, relay functions are performed to transmit data between two nodes. However, in some cases, relay functions include data transmission between two nodes that cannot communicate with each other, and multiple address resolution such as route selection by hop-by-hop transmission. On the other hand, to achieve efficient information exchange and sharing, relay functions are applied to traffic engineering, efficient one-to-many or many-to-many data transmission, flow and retransmission control in hop-by-hop transmission on unstable links, coding or transcoding in data transmission, data cache, and anonymity protection. These functions can be assumed to be technologies that use not only network resources but also computational and storage resources aggressively for performance improvement and efficient network-resource utilization. In IP networks, IP routers only provide simple functions of data transmission, so relay functions on applications have been considered for achieving diverse relay.

In RFC3234 [23], middleboxes are presented as relay nodes with additional functions for special purposes. They are defined as *any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host*. Furthermore, they are general frameworks including network address translation (NAT), firewalls, encryption, IP tunneling, transport or application gateways, TCP performance enhanced proxies, detour path selection for load balancing, session initiation protocol servers, transcoders, HTTP proxies, HTTP caches, and application layer multicast. Special nodes that can provide functions in this example are used for special purposes and located in special places. For example, NAT technologies are used for translating a single global IP address into multiple private IP addresses (vice versa) and located in edge of private networks. For more sophisticated approach, an advanced content distribution network has been proposed [24], which provides path optimization, packet loss reduction, transport protocol optimization, and application optimization using overlay networks constructed by their middleboxes. Note that, however, most widely used middleboxes are introduced for limited objectives in limited locations.

For future direction, through reconsideration of the Internet, also known as the future Internet or new generation networks (NwGNs), network virtualization technologies have been emerged [18], [25]–[28]. Reference [17] has mentioned six design goals of network virtualization technologies. The goals includes *programmability*, which is a main focus of the proposed platform. The network virtualization technologies enable network service providers (NSPs), which provide adaptive network services operated by ISPs or other trusted organizations, to construct multiple logical networks on their physical networks flexibly.

For additional functions inside networks, many researchers aim to make intermediate nodes fundamentally sophisticated. They expect that developers can customize networks and install special functions freely. In this context, several approaches aiming to similar goals have been proposed as follows. New paradigms called active/programmable networks have been proposed and studied [29]. Reference [29] defines an active network such that *individuals can inject programs into the network, thereby tailoring the node processing to be user- and application-specific*. Moreover, a flexible and configurable router named Click router has been considered [30], [31], which provides functions in terms of packet scheduling, dropping policy, classifier, and rewriting header information as its basic functions. A flexible flow-based switch named OpenFlow switch has been developed [32]–[34], which enables to determine

route for a flow using a detailed filtering rule. A programmable overlay router has been considered [35], which provides applications (especially peer-to-peer applications) with commonly used functions inside network such as distributed hash table, caching, etc. In all the approaches, developers can customize behaviors of routers and switches although methodologies are different. These routers including switches are important elements to customize networks.

The proposed platform also aims to sophisticate networks in the same direction. The proposed platform can be classified as a kind of middleboxes and active network technologies because the proposed platform helps developers to customize behaviors of routers for in-network processing. Furthermore, the proposed platform can be used in network virtualization environment as well as Click router, OpenFlow switch, and the programmable overlay router. Regarding technical details, we will present a comparison of the proposed platform with Click router, OpenFlow switch, and the programmable overlay router in Sect. 4.3.

3. Provider-Managed Overlay

Through above sections, the need of efficient relay processing and sophisticated networks are discussed. For future networks, efficient network-resource utilization and coordination between networks and applications will be desired. In this section, we describe a network service model that we consider for future networks.

To achieve the efficient utilization of limited network resources inside NSP networks, it can be considered that distributed applications exchange data on NSP platforms, which provide advanced relay processing as high-valued additional services. In such a service model, distributed applications can actively use advanced relay functions on demand, hence they can leave a part of the information processing functions to NSPs. Furthermore, NSPs can control their resources flexibly if they can cooperate with distributed applications, which leads to a potential to improve network efficiency. In this paper, we call such a service model "provider-managed overlay networks." In these networks, NSPs construct overlay networks, and they provide these networks (platforms) to distributed applications and/or general users. If applications can improve their performance by provider-managed overlay networks, they obtain an incentive to cooperate with NSPs. In turn, NSPs can optimize resource consumption on the assumption that distributed applications use their network.

3.1 Network Model

In this section, we describe a network model of provider-managed overlay networks (Fig. 1). In this model, overlay nodes are located as in-network processing nodes that can provide advanced relay processing rather than simple IP-forwarding processing. These nodes are assumed to be partially located on IP networks (underlay networks), and they construct their overlay networks over the underlay networks.

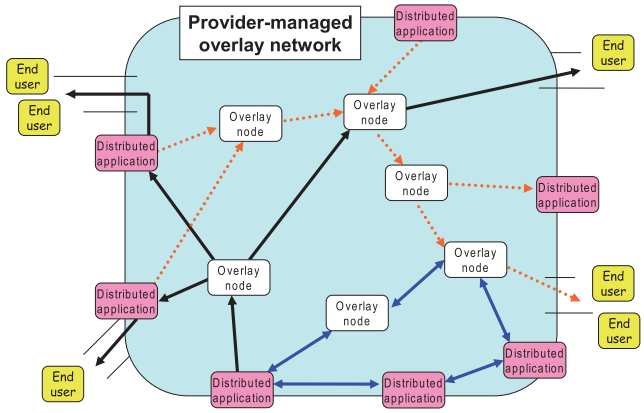


Fig. 1 Provider-managed overlay networks.

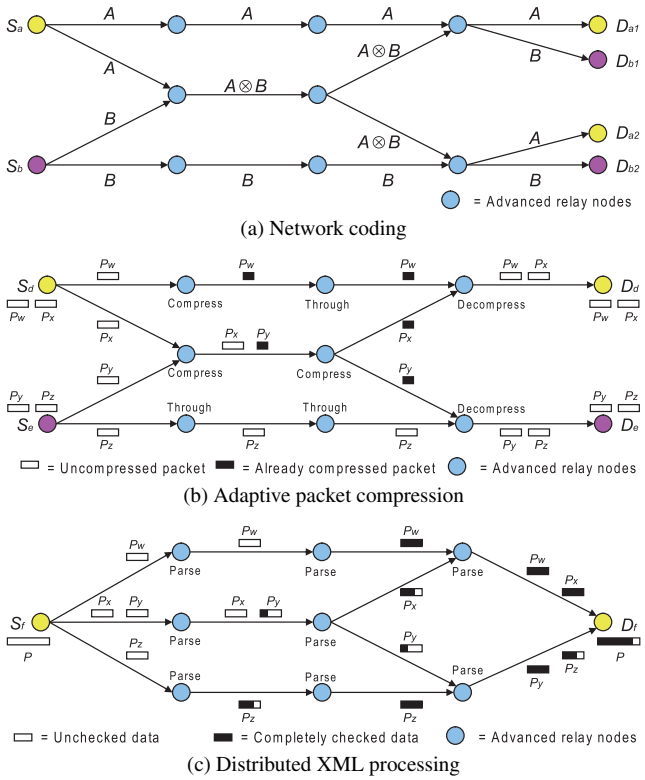
Provider-managed overlay networks can be provided as several types of services: transparent services for users, transparent services for applications, explicit services with certain interfaces for users, and explicit services with certain interfaces for applications. Therefore, provider-managed overlay networks can provide various types of services according to the demands of their users. In particular, the in-network processing nodes inside the networks can provide the processing required for numerous distributed applications. They can be considered as middleboxes because they handle IP packets with advanced relay processing. They also provide various kinds of advanced relay to cooperate with each other.

3.2 Application Examples

On the assumption that routers possess high-performance functionality, traffic engineering to improve network utilization and stability has been studied. In our research group, a flow-based routing scheme [36] and a packet early discard scheme [37] have been considered for delay-sensitive application flows. In general, temporal and spatial resource optimization has been actively studied, e.g., rate and retransmission controls on multi-path routing, network coding, compression, and/or data caching. These methods transparently provide their functions to users, and to applications in many cases. In some cases, the in-network processing nodes located at the edge of a network serve as gateways.

Provider-managed overlay networks can be used in efficient information-exchange services for large-scale, many-to-many communications, while the current Internet cannot efficiently support such services well. These services include multicast, anycast, many-to-one, and many-to-many communications. These technologies have been studied in the research areas of application layer multicast (ALM), grid computing, sensor networks, and network coding (Fig. 2 (a)) among others.

Provider-managed overlay networks can provide high-value additional services. In this model, in-network processing nodes process incoming packets distributively and hierarchically. In our research group, we proposed an adaptive



(a) Network coding

(b) Adaptive packet compression

(c) Distributed XML processing

Fig. 2 Example schemes enabled by in-network processing nodes.

packet compression scheme [22] in which each special node located in a network compresses incoming packets depending on its output queue length (i.e., congestion degree) adaptively, as depicted in Fig. 2 (b). We also proposed an adaptive TCP connection split scheme [38] in which each special node adaptively splits a TCP connection for avoiding heavy congestion. Moreover, we studied a distributed XML processing scheme based on in-network processing [39]–[41]. In this scheme, special nodes distributively parse XML messages, so that the receivers can partly omit parse processing when they receive content data (Fig. 2 (c)).

4. Advanced Relay Node Platform

For evaluating and improving a proposed in-network processing scheme, simulation-based performance analyses are not enough and experimental analyses and evaluation using a prototype implementation are vital. This is because an in-network processing often involves an adaptive inter-work among networks, computational, and storage resources, which prevents an adequate modeling for performance bottleneck estimation. Therefore, to make proof-of-concept testing faster, we have developed an advanced relay node platform running on Linux and off-the-shelf PC, by which a variety of advanced relay processing can easily be proto-typed and tested. The implementation in this study is not limited to certain types of relay processing, and it can provide arbitrary relay processing such as schemes described in Sect. 3.2.

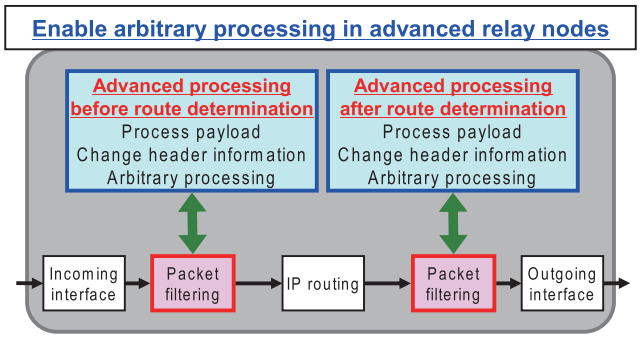


Fig. 3 Conceptual design of advanced relay nodes.

4.1 Conceptual Design

First, we describe a conceptual design of advanced relay nodes. Advanced relay nodes provide advanced relay processing to incoming packets and/or flows before they transmit the packets to the next nodes. To easily and quickly develop various adaptive network services with advanced relay processing, we consider that implementations at a userland space is more suitable than that at a kernel space. In the userland space, complicated processing can be implemented using useful libraries and tools without constraints in the kernel space. Therefore, in this paper, we develop advanced relay nodes on Linux OS and implement advanced relay functions on kernel and userland programs. This approach can easily and flexibly provide various kinds of advanced relay processing. Note that, ideally, they need high-speed processing to avoid performance degradation in terms of packet relay. For implementation in practical use, advanced packet processing can be implemented in kernels of computers, embedded chips, or programmable hardware, such as the field programmable gate array (FPGA) [32].

Figure 3 shows a design overview of advanced relay nodes. They can capture (hook) arbitrary packets at the kernel using flexible filtering rules, and can provide arbitrary processing at userland programs. Finally, they relay the processed packets to other nodes.

4.2 System Design

Advanced relay nodes consist of the following several libraries and application program interfaces (APIs), as depicted in Fig. 4. The nfnetlink library [42] and netfilter_queue library [43] can move hooked packets to userland programs. The rtnetlink library [44] provides the statistics of the network interface and input/output queues to userland programs. The integrated interfaces are used to move packets between the kernel and userland programs.

We develop packet-hook functions at the kernel to integrate the advanced relay processing framework and iptables [45], which provides packet-filtering functions. The packet-filtering policy can be flexibly changed based on the iptables filtering rule. After the filtering rule matches, the

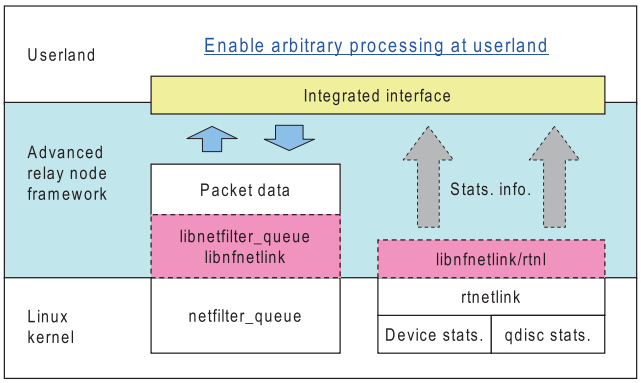


Fig. 4 System design of advanced relay nodes.

```
# iptables -t mangle -A FORWARD \
           -j NFQUEUE --queue-num 0 -i eth0
# iptables -t mangle -A FORWARD \
           -j NFQUEUE --queue-num 1 -i eth1
```

Fig. 5 Configuration example of advanced relay nodes.

incoming packets are hooked, and they are moved to userland programs using NFQUEUE (netfiler_queue).

The advanced relay nodes can hook incoming packets at five certain timings as defined in iptables. In the PREROUTING/POSTROUTING timing, they hook packets before/after the routing processing at the kernel. In the FORWARD timing, they hook packets when they forward these packets to other links. Moreover, in the INPUT/OUTPUT timing, they hook packets when they receive/send packets to or from themselves. For example, the rules of (1) packets entered from the network interface eth0 are moved to a userland program associated with NFQUEUE 0, and (2) packets entered from the network interface eth1 are moved to a userland program associated with NFQUEUE 1, which can be set as described in Fig. 5. Any filtering rules based on the iptables syntax can be set at advanced relay nodes.

Figure 6 shows packet processing in advanced relay nodes. When a packet arrives at advanced relay nodes, it enters a queue called nfq if it matches the iptables filtering rule in phase 1. Then, in phase 2, the filtered packet is hooked and moved to a userland program through the NETLINK socket (nfq interfaces). The hooked packet is copied and stored at the corresponding NFQUEUE associated with the userland program in phase 3. After the packet is processed at the userland program in phase 4, the packet returns to the kernel in phase 5. The packet returned from the userland program is discarded (DROP) or accepted (ACCEPT), based on the indication from the userland program in phase 6. Finally, the accepted packet is forwarded, based on the IP routing in phase 7.

Figure 7 represents the pseudocode of advanced relay processing. In advanced relay nodes, multiple userland programs can be independently launched. Packets filtered at the kernel are moved to the userland programs with the information in NFQUEUE. The advanced relay framework calls

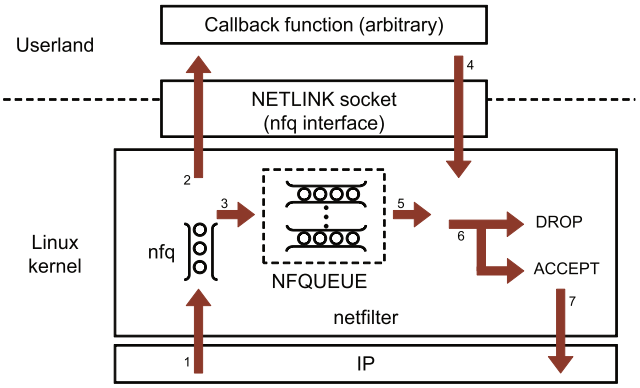


Fig. 6 Packet processing in advanced relay nodes.

```
int main(void){
  // Set callback functions
  // Call these functions when packet arrives
  nfq_setup(dump,    0); // NFQUEUE 0
  nfq_setup(discard, 1); // NFQUEUE 1
  nfq_setup(encap,   2); // NFQUEUE 2

  // Loop (waiting packets)
  while(TRUE){
    // Periodic arbitrary processing
    // E.g., monitoring of I/F statistics
  }
}

// Receive packet information (*data) from kernel
int dump(unsigned char *data){
  // Display the received packet at userland
  packet_dump(data);

  // Send original packet (*data) to kernel
  return THROUGH;
}

// Receive packet information (*data) from kernel
int encap(unsigned char *data){
  // Generate new header
  newhdr = generatehdr();

  // Encapsulate the received packet at userland
  encapsulation(data, newhdr);

  // Send encapsulated packet (*data) to kernel
  return MODIFY;
}

// Receive packet information (*data) from kernel
int discard(unsigned char *data){
  // Discard the received packet at userland
  // Not send packet (*data) to kernel
  return DROP;
}
```

Fig. 7 Pseudocode of arbitrary processing in advanced relay nodes.

the corresponding callback function based on the number of NFQUEUE. The userland program provides advanced relay processing to a hooked packet, and then it returns the packet to the kernel with signals (THROUGH or MODIFY as ACCEPT, and DROP).

4.3 Qualitative Comparison

In this section, we show effectiveness of the advanced relay node platform through qualitative comparison. The aim of the advanced relay node platform is similar to that of other studies described in Sect. 2. We compare the advanced relay node platform with these studies qualitatively using application examples described in Fig. 2.

Click router has many useful function modules for cus-

tomizing behaviors of routers and prepares APIs for developing new modules. Click router performs on kernel space basically, and it can perform on userland space for debugging purpose, while the proposed platform mainly focuses on userland processing for its flexibility. In Click router, if there are no suitable modules and suitable APIs for special in-network processing, developers need to create new modules and APIs with programming on kernel space. In a case that developers attempt to develop application examples in Fig. 2, the adaptive compression scheme is difficult to be developed based on programming manner of Click router because statistics information of output queue is difficult to be obtained.

OpenFlow is capable of flexible route control because a centralized controller determines route for all traffic flows. On the other hand, OpenFlow is not capable of complex in-network processing, so that OpenFlow is difficult to be used for application examples in Fig. 2. However, OpenFlow can be used for route control in the application examples, i.e., it can deliver certain traffic flows to in-network processing nodes.

Programmable overlay router has useful functions for applications. This router is correnspond to explicit services described in Sect. 3.1. Application developers can control this router for their applications through APIs. Examples of functions are distributed hash table, caching, streaming traffic support. This router can be used for explicit services such as Fig. 2 (c) because explicit APIs for cooperation between routers and applications are provided to developers. On the other hand, this router is difficult to be used for transparent services such as Figs. 2 (a) and 2 (b).

Using the advanced relay node platform, developers can develop in-network processing functions on userland space without programming on kernel space. Moreover, the developers can obtain statistics information from kernel space at userland space through the interface described in Fig. 4. Therefore, they can quickly and easily develop novel in-network processing functions from scratch.

5. Experimental Evaluation

In this evaluation, to show the advantages of the advanced relay framework, we investigate the performance of the relay processing speed in the advanced relay nodes and the effect of advanced relay processing in the nodes. For an implementation environment in real computers, we implement the advanced relay framework on Linux OS. The implementation environment is listed below. We use CentOS Linux [46] on a personal computer with CPU of Intel Pentium 4 Processor 3.00 GHz and RAM of 1.0 GB. Information regarding the software and libraries we employ is listed below: CentOS 5.3 with Linux kernel 2.6.27.53 [46], nfnetlink 0.0.41 [42], netfilter_queue 0.0.17 [43], iptables 1.3.5 [45].

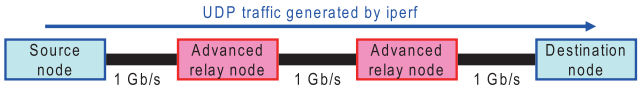


Fig. 8 Preliminary scenario.

5.1 Preliminary Evaluation

In this evaluation, we show the performance of relay processing speed in advanced relay nodes. Because the advanced relay node framework enables the kernel and userland spaces to cooperate with each other, we investigate the overhead of userland processing.

5.1.1 Experimental Procedure

First, we explain the experimental procedure. In this preliminary evaluation, the advanced relay nodes hook the incoming packets using the advanced relay framework, but they perform no advanced processing at a userland space. In other words, they simply move the incoming packets from the kernel space to the userland space and return the packets from the userland space to the kernel space. Therefore, this experimental procedure eliminates extrinsic factors.

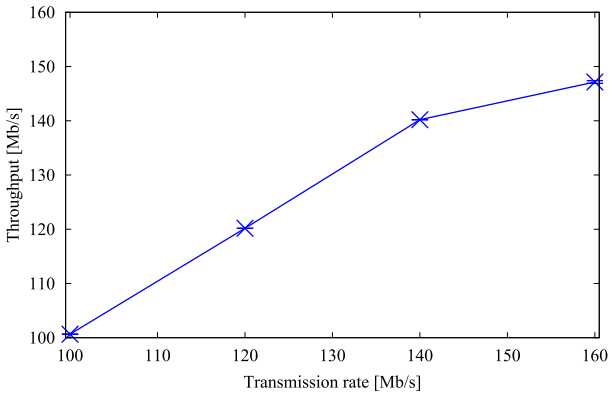
Figure 8 shows a network topology of this experiment. All links are of 1 Gb/s, so that we can easily investigate any bottlenecks. The source node transmits UDP packets using the traffic generator iperf [47] to the destination node over a 10-second period, in order to measure throughput. We vary the transmission rate of iperf. Information on the software and libraries we employ is listed below: iperf 2.0.4 [47].

5.1.2 Preliminary Results

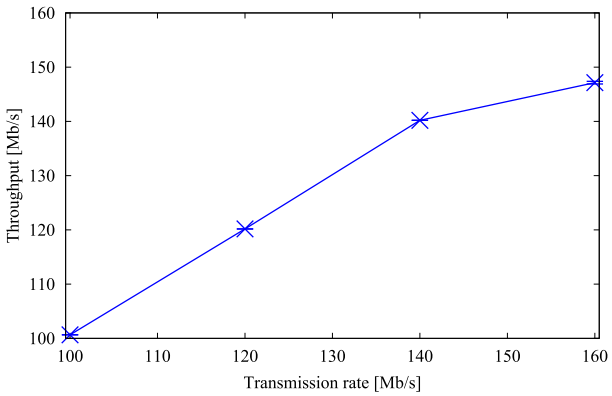
Figure 9 shows the results of the preliminary experiments. These experiments investigate the impact on throughput, and are calculated from the average values and 95% confidence interval (CI) of 20 experiments.

Figure 9 (a) shows the the impact of the transmission rate (bit per second). The x-axis represents the transmission bit rate of iperf. The packet size is set to 1,500 Bytes, including the headers as a fixed value. As the transmission rate increases, the throughput also increases linearly. However, in the range over 140 Mb/s transmission rate, the throughput cannot reach the value of the transmission rate. Note that, in this experiment, the actual transmission rate exceeds the setting transmission rate because of the granularity of the packet-transmission interval time of the traffic generator iperf, e.g., 100 Mb/s transmission rate vs. 100.4 Mb/s throughput.

Figure 9 (b) shows the impact of the transmission rate (packet per second). The x-axis represents the transmission packet rate of iperf. The bit rate is set to 100 Mb/s as a fixed value. Even if the transmission rate increases, the throughput remains 100 Mb/s. However, in the range over approximately 12,000 packet/s transmission rate, the throughput decreases.



(a) Impact of transmission rate (bit per second)



(b) Impact of transmission rate (packet per second)

Fig. 9 Preliminary results.

These results shows that the advanced relay nodes can relay incoming packets with links in the 100 Mb/s environment, although they involve overhead with a high traffic load. To improve the relay performance, high-performance computers should be used. In future work, the implementation will be more sophisticated, because the advanced relay framework has been implemented on normal computers in this study.

5.2 Implementation of Advanced Relay Processing

In the preliminary evaluation, we showed the performance of the advanced relay node platform without any advanced relay processing at a userland space. In this section, as an example of advanced relay processing, we implement the adaptive packet compression scheme [22] described in Sect. 3.2 using the advanced relay node platform. This scheme has already been proven to be effective through simulation evaluations.

First, we explain the adaptive packet-compression scheme that is performed on the advanced relay nodes. This scheme selectively compresses an incoming packet based on its waiting time in the output queue during a period of congestion. Specifically, this scheme compresses a packet only if the following inequality is satisfied:

$$W \geq C - (1 - R) \cdot \frac{S}{B},$$

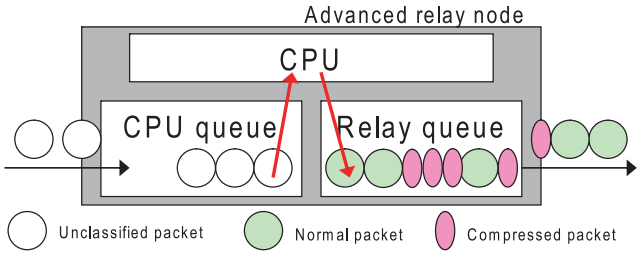


Fig. 10 Design of adaptive packet compression scheme.

where W, C, S, B, and R represent the waiting time in the relay queue, the compression processing time, the packet size, the output-link bandwidth, and the compression ratio, respectively. The values of C and R are parameters and others are given values. If we change parameters of compression, e.g., compression algorithm and compression level, we can adjust values of C and R.

Note that the adaptive packet-compression scheme is corresponding to transparent services for users and applications described in Sect. 3.1. Therefore, this scheme is not aware of applicataion characteristics. To further improve effect of this scheme, we can consider that this scheme cooperates with applications. For example, if this scheme knows application characteristics, we can choose suitable values of C and R for certain applications.

Figure 10 shows an overview of the adaptive packet-compression scheme. This scheme appropriately selects computational or network resources based on the congestion at the output link. This scheme has a likely chance to compress an arrival packet while the packet awaits departure. This compression strategy improves network performance.

For the implementation of the adaptive packet compression scheme, we can use various programming libraries in terms of data compression. Therefore, when the advanced relay nodes hook incoming packets, they can easily compress and decompress packets using these libraries. In this experiments, we use LZO library described later in experimental evaluation.

5.3 Experimental Evaluation

To show the effectiveness of advanced relay processing, we perform experiments using real computers. In this evaluation of advanced relay processing, we use the implementation of the adaptive packet compression scheme described in Sect. 5.2.

5.3.1 Experimental Procedure

Figure 11 depicts the network topology of this experiment. The buffer size of the advanced relay nodes is 1,000 packets (default configuration on CentOS). The source node transmits the UDP packets using the traffic generator iperf [47] and transmits the ICMP packets using the administration utility ping [48] to the destination node over a 10-second period, in order to measure throughput and packet loss ra-

tio, round trip delay time, and jitter, respectively. The packet size of these protocols is set to 1,500 and 84 Bytes, respectively, based on the default value in ping, including the size of the IP headers. The transmitted packets pass through the compression node and are adaptively compressed at this node. After the packets pass through the decompression node and are decompressed (if they are compressed), they arrive at the destination node. Regarding compression, the actual compression ratio of the UDP packets is approximately 0.95 (i.e., 5% data reduction). The values of the parameters C and R in the adaptive packet-compression scheme are set to 20 μs and 0.95. Information on the software and libraries we employ is listed below: LZO 2.03 [49] as the compression library, inetutils 1.8 [48], and iperf 2.0.4 [47].

5.3.2 Experimental Results

Figure 12 shows the experimental results. In these experiments, we examine the throughput, packet loss ratio, and round trip time as performance indices. These results are calculated from the average values and 95% confidence interval (CI) of 20 experiments. The x-axes represent the setting value of the transmission rate. In Fig. 12 (a), the throughput of no-compression is suppressed at approximately 95.6 Mb/s, while that of adaptive compression increases and approaches approximately 100.7 Mb/s (i.e., an increase by 5.3%). In Fig. 12 (b), the packet loss ratio of no-compression increases when the transmission rate exceeds 96 Mb/s, while that of adaptive compression does not occur in the case of under 100 Mb/s transmission rate. In Fig. 12 (c), the adaptive compression can suppress its delay time in comparison with the no-compression. Finally, in Fig. 12 (d), the result is similar to that of round trip time. These results conclude that the advanced relay, i.e., adaptive use of resources, can improve performance.

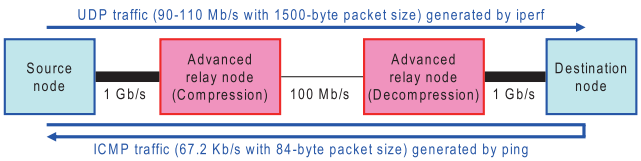


Fig. 11 Experimental scenario.

5.3.3 Experimental Results with Real-Time Applications

Figure 13 depicts the network topology of this experiment. The sender nodes transmit UDP packets using the traffic generator iperf at the transmission rate of 70 Mb/s and the digital video transport system (DVTS) at the transmission rate of 30 Mb/s, respectively. The transmitted packets pass through the compression node and are adaptively com-

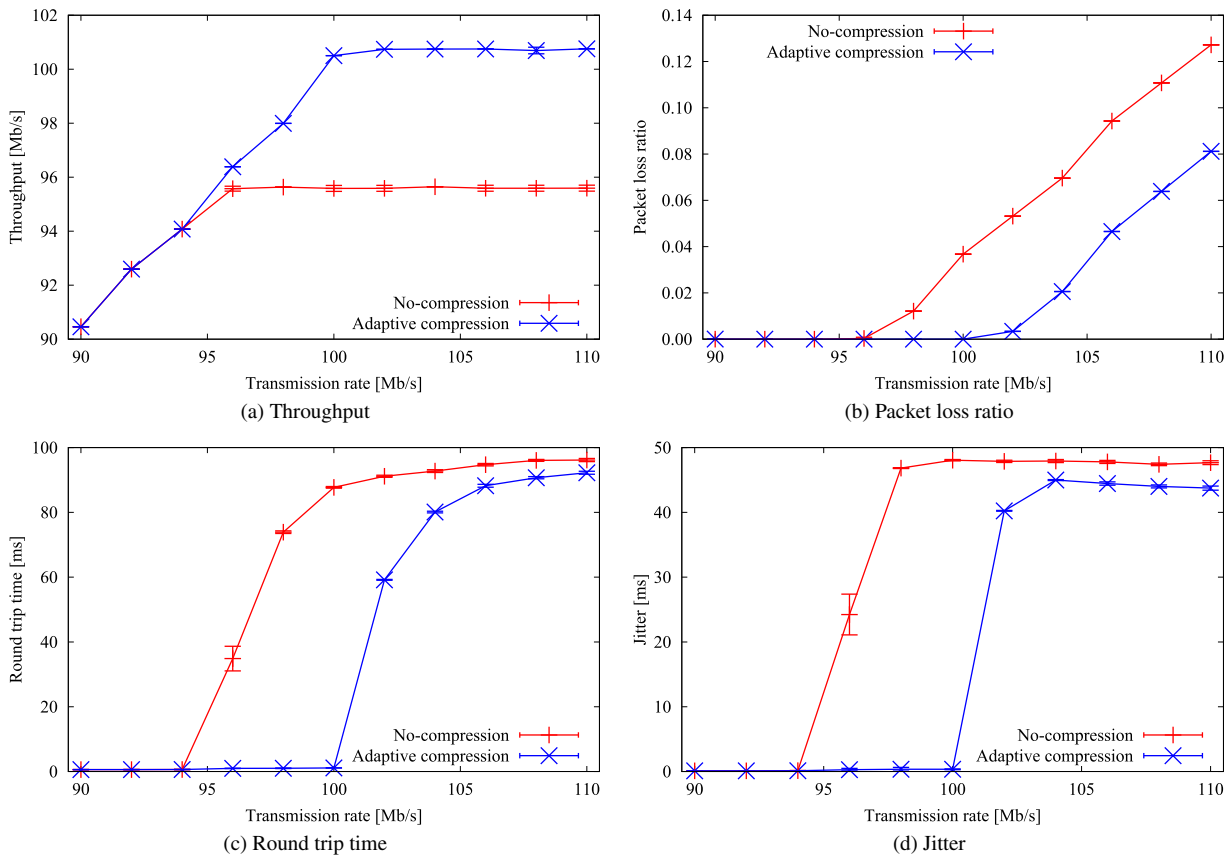


Fig. 12 Experimental results.

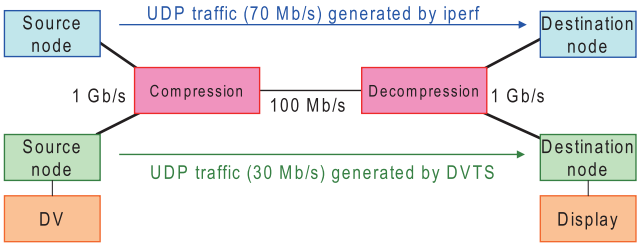
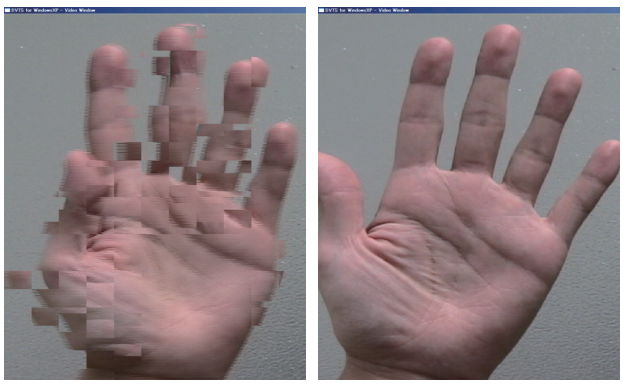


Fig. 13 Demonstration scenario.



(a) Result without adaptive packet compression (b) Result with adaptive packet compression

Fig. 14 Experimental result with real-time applications.

pressed at this node. After the packets pass through the decompression node and are decompressed (if they are compressed), they arrive at the receiver nodes. Information on the software and libraries we employ is listed below: iperf 2.0.4 [47], and DVTS 0.0.2 [50]. Regarding information in terms of DVTS, we use default values as follows: resolution is 720x480 pixel, frame rate is 29.97 frame/s, and packet size is 1400 Bytes. Moreover, DVTS is not capable of any loss recovery mechanisms (e.g., retransmission, error recovery).

Figure 14 shows the captured screens of the result. When the relay nodes provide a simple relay function, i.e., when IP forwarding, congestion and packet loss occur, the video image is distorted by occurrences of block-noise continually during the video image is transmitted, as shown in Fig. 14 (a). In this case, as implied in Fig. 12 (b), the packet loss ratio is approximately 0.04. On the other hand, when the relay nodes provide an advanced relay function, i.e., when the adaptive packet compression, congestion and packet loss are significantly alleviated, the video image is high quality without block-noise, as shown in Fig. 14 (b). In this case, as implied in Fig. 12 (b), the packet loss ratio is zero.

6. Concluding Remarks

The reconsideration of the coordination and partitioning of functions between internal-networks (providers) and end-hosts with applications (users) has emerged and has been actively discussed by many researchers worldwide. In this paper, we have first discussed the need and feasibility of flexible and adaptive network services. In particular, we described the importance of advanced relay processing that attempts to efficiently utilize limited resources by the dynamic resource optimization. We have then design the advanced relay node platform to easily and quickly implement and test a variety of adaptive network services based on not only per-flow but also per-packet processing, which enables to experimentally evaluate the feasibility of new ideas of the services. A key feature of the proposed platform is that integration between kernel and userland spaces enables to easily develop various adaptive network services. On the top of the advanced relay node platform, we have implemented and tested the adaptive packet compression scheme that we previously proposed. Finally, through experimental evaluation using real computers, we have shown the feasibility of both the developed platform and the adaptive packet compression. We will enhance the proposed platform for more high-speed processing.

References

[1] M. Shimamura, T. Ikenaga, and M. Tsuru, "Advanced relay nodes for adaptive network services — concept and prototype experiment," Proc. 2010 International Conference on Broadband, Wireless Computing, Communication and Applications (BWCAA 2010), Second International Workshop on Network Traffic Control, Analysis and Applications (NTCAA-2010), pp.701–707, Nov. 2010.

[2] Cisco Systems, Inc., "Cisco visual networking index: Forecast and methodology, 2008–2013," June 2009. Information available at http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

[3] L. Qiu, Y.R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in Internet-like environments," IEEE/ACM Trans. Netw., vol.14, no.4, pp.725–738, Aug. 2006.

[4] National Institute of Information and Communications Technology (NICT), "AKARI: Architecture design project for new generation network," Oct. 2008. Information available at http://akari-project.nict.go.jp/eng/conceptdesign.htm

[5] National Science Foundation (NSF), "NSF Future Internet Architecture Project." Information available at http://www.nets-fia.net/

[6] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: The FIRE initiative," ACM SIGCOMM Comput. Commun. Review, vol.37, no.3, pp.89–92, July 2007.

[7] NRENAISSANCE Committee and National Research Council, Realizing the Information Future: The Internet and Beyond, National Academies Press, Jan. 1994.

[8] H. Xie, Y. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz, "P4P: Provider portal for applications," ACM SIGCOMM Comput. Commun. Review, vol.38, no.4, pp.351–362, Oct. 2008.

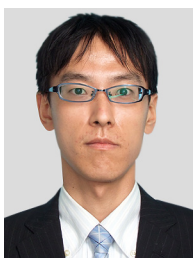
[9] S. Seetharaman and M. Ammar, "Overlay-friendly native network: A contradiction in terms?," Proc. ACM Fourth Workshop on Hot Topics in Networks (HotNets-IV), Nov. 2005.

[10] T. Omizo, K. Masui, and K. Iida, "Design and implementation of inter-ISP virtual backbone infrastructure to meet various QoS requirements," Proc. 11th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT2011), Future Internet Engineering (FIE2011), pp.486–491, July 2011.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A global internet host distance estimation service," IEEE/ACM Trans. Netw., vol.9, no.5, pp.525–540, Oct. 2001.

[12] D. Choffnes and F. Bustamante, "Taming the torrent: A practical

approach to reducing cross-ISP traffic in P2P systems," Proc. ACM SIGCOMM08, Aug. 2008.
[13] A. Nakao, L. Peterson, and A. Bavier, "A routing underlay for overlay networks," Proc. ACM SIGCOMM03, Aug. 2003.
[14] Distributed Computing Industry Association (DCIA). Information available at http://www.dcia.info/
[15] The seventh framework programme (FP7), "Network-aware P2P-TV application over wise networks." Information available at http://napa-wine.eu/
[16] N. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," IEEE Commun. Mag., vol.47, no.7, pp.20–26, July 2009.
[17] N. Chowdhury and R. Boutaba, "A survey of network virtualization," Comput. Networks, vol.54, no.5, pp.862–876, April 2010.
[18] A. Nakao, "Network virtualization as foundation for enabling new network architectures and applications," IEICE Trans. Commun., vol.E93-B, no.3, pp.454–457, March 2010.
[19] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," Comput. Networks, vol.47, no.4, pp.445–487, March 2005.
[20] K. Letaief and W. Zhang, "Cooperative communications for cognitive radio networks," Proc. IEEE, vol.97, no.5, pp.878–893, May 2009.
[21] O. Akan, O. Karli, and O. Ergul, "Cognitive radio sensor networks," IEEE Netw., vol.23, no.4, pp.34–40, July 2009.
[22] M. Shimamura, H. Koga, T. Ikenaga, and M. Tsuru, "Compressing packets adaptively inside networks," IEICE Trans. Commun., vol.E93-B, no.3, pp.501–515, March 2010.
[23] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and issues," RFC 3234 (Informational), Feb. 2002.
[24] E. Nygren, R. Sitaraman, and J. Sun, "The Akamai network: A platform for high-performance Internet applications," ACM SIGOPS Operating Systems Review, vol.44, no.3, pp.2–19, July 2010.
[25] A. Nakao, R. Ozaki, and Y. Nishida, "CoreLab: An emerging network testbed employing hosted virtual machine monitor," Proc. 2008 ACM CoNEXT Conference, 3rd International Workshop on Real Overlays & Distributed Systems (ROADS2008), Dec. 2008.
[26] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," ACM SIGCOMM Comput. Commun. Review, vol.37, no.1, pp.61–64, Jan. 2007.
[27] J. Turner, P. Crowley, J. Dehart, A. Freestone, B. Heller, F. Kuhms, S. Kumar, J. Lockwood, J. Lu, M. Wilson, C. Wiseman, and D. Zar, "Supercharging PlanetLab - high performance, multi-application, overlay network platform," Proc. ACM SIGCOMM07, Aug. 2007.
[28] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," ACM SIGCOMM Comput. Commun. Review, vol.35, no.5, pp.41–54, Oct. 2005.
[29] D. Tennenhouse and D. Wetherall, "Towards an active network architecture," ACM SIGCOMM Comput. Commun. Review, vol.26, no.2, pp.5–17, April 1996.
[30] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The Click modular router," ACM Trans. Comput. Systems, vol.18, no.3, pp.263–297, Aug. 2000.
[31] E. Kohler, The Click modular router, Ph.D. thesis, Massachusetts Institute of Technology, Nov. 2000.
[32] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM Comput. Commun. Review, vol.38, no.2, pp.69–74, April 2008.
[33] OpenFlow, "OpenFlow switch specication version 1.3.0," June 2012. Information available at http://www.opennetworking.org/
[34] Open Networking Foundation (ONF), March 2011. Information available at http://www.opennetworkingfoundation.org/
[35] B. Davie and J. Medved, "A programmable overlay router for service provider innovation," Proc. 2nd ACM SIGCOMM workshop on Programmable Routers for Extensible Services of TOmorrow (PRESTO '09), pp.1–6, Aug. 2009.
[36] Y. Kitatsuji, M. Tsuru, T. Takine, and Y. Oie, "Flow assignment method with traffic characteristics over multiple paths for reducing queuing delay," Telecommunication Systems, vol.37, no.1–3, pp.97–108, March 2008.
[37] K. Kumazoe, M. Tsuru, and Y. Oie, "Adaptive early packet discarding scheme to improve network delay characteristics of real-time flows," IEICE Trans. Commun., vol.E90-B, no.9, pp.2481–2493, Sept. 2007.
[38] M. Shimamura, T. Ikenaga, and M. Tsuru, "Splitting tcp connections adaptively inside networks," IEICE Trans. Inf. & Syst., vol.E95-D, no.2, pp.542–545, Feb. 2012.
[39] D. Cavendish and S. Candan, "Distributed XML processing: Theory and applications," J. Parallel and Distributed Computing, vol.68, no.8, pp.1054–1069, Aug. 2008.
[40] K. Yoshinaga, Y. Uratani, and H. Koide, "Utilizing multi-networks task scheduler for streaming applications," Proc. Fourth Int'l Workshop on Scheduling and Resource Management for Parallel and Distributed Systems, Sept. 2008.
[41] Y. Uratani, H. Koide, D. Cavendish, and Y. Oie, "Distributed XML processing over various topologies: Characterizing XML document processing efficiency," Lecture Notes in Business Information Processing, vol.101, no.2, pp.57–71, Jan. 2012.
[42] netfilter core team, "The netfilter.org "libnfnetlink" project," June 2009. Information available at http://www.netfilter.org/
[43] H. Welte, "The netfilter.org "libnetfilter_queue" project," March 2009. Information available at http://www.netfilter.org/
[44] J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov, "Linux netlink as an IP services protocol." RFC 3549 (Informational), July 2003.
[45] netfilter core team, "The netfilter.org "iptables" project," May 2010. Information available at http://www.netfilter.org/
[46] CentOS Project, "The community enterprise operating system (CentOS)," May 2010. Information available at http://www.centos.org/
[47] Distributed Applications Support Team (DAST), "Iperf," March 2010. Information available at http://sourceforge.net/projects/iperf/
[48] GNU Operating System, "netutils — GNU network utilities," May 2010. Information available at http://www.gnu.org/software/inetutils/
[49] M. Oberhumer, "LZO — a real-time data compression library," April 2008. Information available at http://www.oberhumer.com/opensource/lzo/
[50] A. Ogawa, "DV stream on IEEE1394 encapsulated into IP," April 2006. Information available at http://www.sfc.wide.ad.jp/DVTS/



Masayoshi Shimamura received the B.E. degree from the Faculty of Engineering of Chiba Institute of Technology, Chiba, Japan, in 2003, and the M.E. and Ph.D. degrees from the Graduate School of Information Science of Nara Institute of Science and Technology (NAIST), Nara, Japan, in 2005 and 2009, respectively. From 2005 to 2007, he was an encouragement researcher in the 21st Century COE Program "Ubiquitous Networked Media Computing" in the Graduate School of Information Science, NAIST. From 2008 to 2011, he was a postdoctoral researcher at the Network Design Research Center of Kyushu Institute of Technology (KIT). Since 2011, he has been a postdoctoral researcher and assistant professor at the Global Scientific Information and Computing Center, Tokyo Institute of Technology. His research interests include performance evaluation of networking systems. He is a member of the IEEE.

Takeshi Ikenaga received B.E., M.E. and D.E. degrees in computer science from Kyushu Institute of Technology, Iizuka, Japan in 1992, 1994 and 2003, respectively. From 1994 to 1996, he worked at NEC Corporation. From 1996 to 1999, he was an Assistant Professor in the Information Science Center, Nagasaki University, From 1999 to 2004, he was an Assistant Professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology. He was an Associate Professor from 2004 to 2011 and then has been a Professor since September 2011 in the Department of Electrical, Electronic and Computer Engineering, Faculty of Engineering, Kyushu Institute of Technology. His research interests include performance evaluation of computer networks and QoS routing. He is a member of the IEEE.

Masato Tsuru received B.E. and M.E. degrees from Kyoto University, Japan in 1983 and 1985, respectively, and then received his D.E. degree from Kyushu Institute of Technology, Japan in 2002. He worked at Oki Electric Industry Co., Ltd. (1985–1990), Information Science Center, Nagasaki University (1990–2000), and Japan Telecom Information Service Co., Ltd./Telecommunications Advancement Organization of Japan (2000–2003). In 2003, he moved to the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology as an Associate Professor, and then has been a Professor in the same department since April 2006. His research interests include performance measurement, modeling, and management of computer communication networks. He is a member of the IEEE, ACM, IPSJ, and JSSST.