Quality-of-Service Mechanisms for Flow-Based Routers

Nam-Seok Ko, Sung-Back Hong, Kyung-Ho Lee, Hong-Shik Park, and Nam Kim

In this paper, we propose quality of service mechanisms for flow-based routers which have to handle several million flows at wire speed in high-speed networks. Traffic management mechanisms are proposed for guaranteed traffic and non-guaranteed traffic separately, and then the effective harmonization of the two mechanisms is introduced for real networks in which both traffic types are mixed together. A simple non-work-conserving fair queuing algorithm is proposed for guaranteed traffic, and an adaptive flow-based random early drop algorithm is proposed for non-guaranteed traffic. Based on that basic architecture, we propose a dynamic traffic identification method to dynamically prioritize traffic according to the traffic characteristics of applications. In a high-speed router system, the dynamic traffic identification method could be a good alternative to deep packet inspection, which requires handling of the IP packet header and payload. Through numerical analysis, simulation, and a real system experiment, we demonstrate the performance of the proposed mechanisms.

Keywords: Fow-based router, quality-of-service (QoS), IntServ, DiffServ, packet scheduling, dynamic flow identification.

Manuscript received Aug. 13, 2007; revised Jan. 4, 2008.

This work was partly supported by the IT R&D program of MIC/IITA [2007-S013-01] and the ITRC program of MIC/IITA [C1090-0603-0035], Rep. of Korea.

Nam-Seok Ko (phone: + 82 42 860 5560, email: nsko@etri.re.kr), Sung-Back Hong (email: sbhong@etri.re.kr) and Kyung-Ho Lee (email: kholee@etri.re.kr) are with the Broadcasting & Telecommunications Convergence Research Laboratory, ETRI, Daejeon, Rep. of Korea.

Hong-Shik Park (email: hspark@icu.ac.kr) is with the School of Engineering, Information Communications University, Daejeon, Rep. of Korea.

Nam Kim (email: namkim@chungbuk.ac.kr) is with the School of Electrical & Computer Engineering, Chungbuk University, Cheongju, Rep. of Korea.

I. Introduction

The remarkably rapid development of Internet technology has led to the establishment of a wide variety of businesses that require heavy exchanges of multimedia data. Although some parts of multimedia exchanges do not specify quality-of-service (QoS) requirements, there are several applications that require specific QoS guarantees, such as bandwidth guarantee. To cope with this problem, several QoS mechanisms have been proposed and adopted in networks.

IntServ [1] and DiffServ [2] are examples of such mechanisms. IntServ sets up a session by exchanging explicit signaling messages. If efficient packet scheduling algorithms are used, it can perfectly control each session. However, it has a scalability problem since it requires signaling messages to be exchanged between terminals periodically. Moreover, it is not easy to support per-session QoS in high-speed networks, which require per-session queues for millions of sessions. Therefore, IntServ cannot be deployed in large networks. On the other hand, DiffServ controls only traffic classes rather than each session within a traffic class. The mechanism uses differentiated service code point (DSCP) values in the IP header to deliver the QoS class of the traffic. In the edge nodes, the DSCP field is set to a proper value according to the traffic classification, and the core nodes use the DSCP values, which are already set in the edge nodes. However, DiffServ still lacks controllability, even though performance is far better than when the mechanism is not used. Traffic is classified into predetermined traffic classes, and QoS is addressed in terms of those classes. Weighted fair queuing or other suitable scheduling algorithms can be applied in the system, but there is no way to provide QoS for each flow in the traffic classes with this class-based traffic control architecture [3]-[8].

For these reasons, a simple and efficient flow management architecture is needed with an efficient and scalable packet scheduling algorithms to handle several million flows, which do not use expensive signaling messages. Note that we use the terms *flow* and *session* interchangeably. Most of the scheduling algorithms that have been proposed and deployed in commercial systems are work-conserving algorithms, which require per-flow queues [3]-[11]. They are appropriate for guaranteed services but not for non-guaranteed traffic. If they are applied to best-effort services, the rates for the flows should be dynamically chosen according to the current number of backlogged flows, which creates too high a burden on the system. Therefore, packet scheduling algorithms have generally been applied on a per-class basis.

Recently, there has been active research on flow-based queue management [9]-[11]. For non-guaranteed traffic, drop-probability-based active queue management and hierarchical weighted round robin were proposed separately. A time-slot-based scheduling mechanism was also proposed. These studies mainly focus on how to allocate and manage buffers for flows and classes. However, it is not easy to maintain per-flow buffers and queues in high-speed core networks.

We propose a different QoS architecture for the flow-based router system. The buffer is shared among flows, and the calendar queue is used for the scheduling packets [12]. The packets are stored in the shared memory in the system when they arrive in the system, and the scheduling times for the input packets are decided before sending them to calendar queue. If a packet is not appropriate to be serviced, it is dropped rather than being sent to the calendar queue. Scheduling algorithms are proposed for both guaranteed and non-guaranteed traffic, which are harmonized not to affect each other in a real traffic environment. In addition, we propose a technique to enhance the scheduling algorithm by dynamically identifying traffic on the basis of the traffic characteristics. This improves flexibility of the flow-based router by prioritizing the flows, which are identified dynamically. Dynamically identified flows can be treated differently from normal traffic flows. According to the user's preferences, the traffic can be given higher or lower priority. The performance characteristics of the proposed algorithms are analyzed by numerical analysis, simulation, and real system test.

The remainder of this paper is organized as follows. In section II, we propose a flow-based router framework to support efficient flow-based QoS. Then, traffic management mechanisms to support flow-based QoS are proposed in section III. In section IV, the results of the performance evaluation are given by numerical analysis, simulation, and real system experiment. Finally, conclusions are presented in section V.

II. Framework of the Proposed Flow-Based Router

Our flow-based router is based on the principle of recognizing flows, routing the first packet of a flow, dynamically associating state with it, and then switching the remaining packets in the flow using the state information. The definition of *flow* is not fixed, but it could be defined in various ways according to the requirements of the user or service provider. A flow could be defined as a traffic flow which shares the 5-tuple IP header fields, but for MPLS traffic, it could be defined as traffic which shares the same MPLS label. In this paper, a flow is an IP flow which is defined by 5-tuple IP header fields if it is not specifically mentioned. An IP flow is also called an IP micro-flow since the flow definition is the finest one which is most frequently used in flow-based router networks. The mechanism by which a flow is identified is based on a hashing function. The best known hashing functions, such as XOR and CRC32, were analyzed in [13], but consideration of specific hashing function is beyond the scope of this paper.

If a packet comes into the system, the selected hashing function will generate a hash value. The hash value is used to find the flow state entry for the flow of the packet. If the packet is the first packet of the flow, no flow state entry for the flow exists. Therefore, a new flow state entry must be created for the flow with the appropriate forwarding and QoS information. The information can be gathered by referring to the other tables such as forwarding tables and the QoS classification table. On the other hand, if there is already a flow state entry for the flow, the packet is just processed according to the information in the flow state table.

Since a flow is uniquely identified by its 5-tuple fields, the lookup for the flow state table should be an exact match instead of longest-prefix match as in the IP forwarding table lookup. The hash-based table is best for the flow state table. Even though there are hash collisions in hash-based tables, if the hash function is well designed, the hash collision rate can be quite well minimized [13].

A flow state entry is created and maintained when the first packet enters the system. As shown in Fig. 1, a flow state table is looked up first (❶) and then if there are no matching entries for

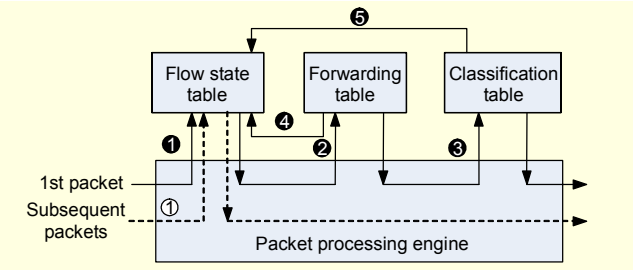

Fig. 1. Conceptual architecture of flow-based router.

the hash value, the forwarding table and classification tables are looked up (❷,❸). Next, the flow state entry is created for the flow to gather the required information for the flow (❹,❺). The subsequent packets of the flow will simply use the already existing information in the flow state entry (①). The created flow block is deleted if a certain time duration passes. This can be done by maintaining the lifetime of flows in the flow state entry [9].

Once flows are identified and maintained in the system, traffic management can be done for each flow. As mentioned in the previous section, our traffic management approach to support QoS is different from those of previous studies [9]-[11]. Previous studies focused on how to allocate and manage buffers for each flow and class. However, it is not easy to maintain per-flow buffers and queues in high-speed core networks. Therefore, we propose a simple but robust QoS architecture for the flow-based router system using a calendar queue and new packet scheduling algorithms in the shared buffer management system.

III. Proposed QoS Architecture of the Flow-Based Router

1. Traffic Management Architecture for Flow-Based Router

A. Scheduler Architecture for the Proposed QoS Mechanisms

The proposed scheduler for the flow-based router does not use per-flow queues or buffers as assumed in existing flow-based router mechanisms [9]-[11]. In the real environment, to handle millions of flows and support per-flow queuing is not easy. Therefore, we propose efficient traffic management mechanisms, which utilize a calendar queue scheduler in shared buffer management systems [7], [14]. The architecture is shown in Fig. 2.

For non-guaranteed traffic, drop-probability-based active queue management is proposed, and we use a virtual clock like a fair queuing algorithm in order to support guaranteed traffic. There are some special features in each traffic management method for guaranteed and non-guaranteed traffic. For non-guaranteed traffic, each flow is dynamically identified and then processed differently than normal traffic. If the traffic is known to be malicious it will be dropped severely. If the traffic is identified to be guaranteed we give it higher priority not to be randomly discarded. In addition, we apply a shaping concept to non-guaranteed traffic according to the congestion status. For guaranteed traffic, our mechanism differs from existing fair queuing algorithms (such as virtual clock and the other WFQ variants) in that the time stamping is not actually tagged to the packet itself, and sorting is not used to select the minimum virtual finishing time. Instead, if a time to send is calculated for the packets, the packets are inserted in the matching calendar

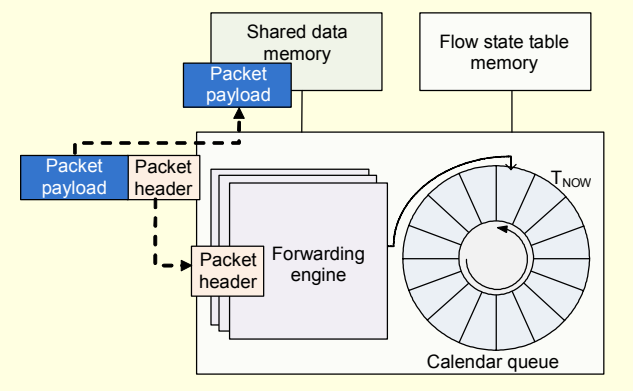

Fig. 2. Proposed forwarding architecture.

queue time slots. Therefore, it is not necessary to maintain a per-flow queue or buffer, which is the main difference from existing flow-based scheduling architectures [9]-[11]. However, the linked list structure should be used to maintain the packet order of a flow as in other shared-memory architectures [10], [11].

In the enqueue process, the time to send is determined according to the algorithm for both non-guaranteed traffic and guaranteed traffic. The time to send of non-guaranteed traffic is determined according to the traffic congestion status of the class with which the flow is associated; thus, non-guaranteed traffic is shaped according to the traffic congestion status. The time to send of guaranteed traffic is purely determined according to its reserved rate as in other fair queuing algorithms. However, instead of tagging time stamps to packets and sorting the packets according to the increasing order of time stamps, the packets are inserted into a time slot of a calendar queue, which is chosen by the calculated time to send. In this calendar-queue-based scheduler, if each packet is allocated to the appropriate time slot in the enqueue process, the dequeue process visits each time slot in sequence and services the packets in it [7], [19]. Those mechanisms are explained in detail in the following subsections.

B. Packet Scheduling Algorithms for Guaranteed Service

Since guaranteed traffic specifies QoS requirements, non-work-conserving algorithms are more appropriate than work-conserving algorithms. In a non-work-conserving algorithm, even when there are backlogged packets in the system, if a packet is not eligible according to the QoS requirements, that packet is not serviced until it becomes eligible. Most work-conserving algorithms are complex because they should adapt their service rates considering the assigned weight and current buffer status. However, if only the requested QoS requirements are to be guaranteed, it is simpler to use non-work-conserving algorithms. In this paper, we focus on the bandwidth requirement in relation to several QoS requirements.

Instead of using virtual time, we use a real-time clock to calculate the finishing time of a packet. In work-conserving algorithms, since the real finishing time is different from the calculated finishing time, the calculated finishing time is called the virtual finishing time. In non-work-conserving algorithms, packets are serviced at the time of the calculated finishing time. We call the scheduling algorithm real-time-clock fair queuing (RCFQ) since the proposed non-work-conserving algorithm is based on the real-time clock. The packet finish time of the k-th packet of flow i (F_i^k) with reserved rate r_i and packet length l_i^k in RCFQ is calculated as

$$F_i^k = \max(F_i^{k-1}, T_{now}) + l_i^k / r_i \ , \quad k=1, 2, \cdots, \tag{1}$$

where $F_i^0 = T_{now}, T_{now}$ is the current time of the real-time clock.

The result of max in (1) is the packet start time of k-th packet. It causes the packet to be delayed until the eligible time when a packet arrives earlier than expected. The algorithm is quite similar to the virtual clock (VC) [8] in that a real-time clock is used instead of virtual time. However, the difference is that we do not have the fairness problem which is intrinsic to the VC. In the VC, if a flow produces a large burst of data, even in a lightly loaded situation, the flow is affected by other newly activated flows. This is because the VC was using real-time clock instead of virtual time. However, since RCFQ is a non-work-conserving algorithm, and it services packets only by each flow's reserved rate, the algorithm has no such problem. In RCFQ, the difference between the finishing time of an HOL packet in each session and the system clock never exceeds a certain amount of time. This will be analyzed in section IV. Another important difference from the VC is that we do not need to maintain a separate sorting structure. The exact time to send is already decided when the starting and finishing times of a packet are calculated.

This RCFQ has a low finishing time computation complexity of $O(1)$ without requiring additional sorting structure by making use of a calendar queue. In the existing fair queuing algorithms, even when they have $O(1)$ virtual time complexity, they have additional $O(logN)$ sorting complexity intrinsically.

C. Packet Scheduling Algorithms for Non-guaranteed Service

Non-guaranteed service is the best-effort service for traffic that does not request any specific QoS requirements. Therefore, if we service this traffic using the packet scheduling method, the overhead is too big compared to the expected QoS level for the traffic. If fair queuing is used in the system for the best-effort traffic, the bandwidth for a flow should be dynamically chosen according to the current number of flows in the system and the congestion status. This is not affordable in high-speed networks; therefore, this kind of traffic may as well be handled by simple active queue management.

In active queue management, RED is a representative mechanism. Some variants have been proposed to address the basic limitations of RED, including BLUE, adaptive RED, flow random early drop (FRED), and dual metrics fair queuing (DMFQ) [3]-[20]. Among them, FRED maintains the average queue for the system as well as a portion of queues for each flow from the total queue length. It calculates each flow's drop probability based on them. By first dropping packets from the flows whose queue lengths are longer than the average flow queue length, FRED provides better protection for unresponsive traffic. The DMFQ mechanism calculates the dropping probability based on the flow arrival rate and flow succession time.

The basic approach for the proposed active queue management mechanism is similar to DMFQ in that both mechanisms use the average flow rate rather than the queue length. However, the proposed architecture is enhanced by allowing the dropping probability to be changed dynamically according to the traffic characteristics. The basic mechanism of the proposed algorithm, which is called adaptive flow random early drop (AFRED), is described as follows for a flow i which is a member of traffic class x.

Constants
- min_th_x: minimum class threshold of class x
- max_th_x: minimum class threshold of class x
- max_p: maximum drop probability

Variables
- Q_x: class x's average queue length
- R_x: class x's average rate
- N_x: average number in a class x
- $\hat{r}_i$: average flow rate of a flow i
- $\tilde{r}_i$: expected fair flow rate of a flow i
- P_x: drop probability of a class x
- p_i: drop probability of a flow i
- n: normalized flow drop probability factor

For each class
- calculate class x's drop probability as follows

$$P_x = (Q_x - min\_th_x)/(max\_th_x - min\_th_x) \tag{2}$$

For each packet of a flow
- calculate expected fair flow rate as follows

$$\tilde{r}_i = R_x / N_x \tag{3}$$

- calculate normalized flow drop probability factor

$$n = (\hat{r}_i / \tilde{r}_i) \times C_1, \tag{4}$$

where C_1 is a constant value
- calculate flow's drop probability as follows

$$p_i = \max(max_p, n \times P_x) \tag{5}$$

- drop packets with the drop probability of p_i

The final drop probability of a flow i (p_i) is calculated based on the class's drop probability (P_x) and the ratio of the current flow's average rate ($\hat{r}_i$) to the expected flow rate ($\tilde{r}_i$), which is calculated with the average number of flows (N_x) and the average class's rate (R_x). Note that a class is a group of flows, which is determined by the classification result. The current flow's average rate ($\hat{r}_i$) and the average class's rate (R_x) are assumed to be known *a priori*. Those values are maintained in ASIC by the exponential moving average calculation. The mechanism will drop fewer packets from the flows with rates lower than the average flow rate, but more packets from the flows with rates higher than the average flow rate. This is quite a reasonable and fair approach.

Although this flow drop probability calculation is effective, it can be enhanced. In previous basic flow drop probability calculations, even when a class is in a heavy state of congestion, the drop probability of a flow that is sending under the average flow rate is not high enough to alleviate congestion quickly. Therefore, we modify the drop probability calculation method such that the drop probability increases slowly when a class is in a light state of congestion, but it increases very quickly when the class is in a heavy state of congestion. The relationship between the class's drop probability and flow's drop probability is visualized in Fig. 3. The enhanced flow drop calculation method is given for each packet of flow as

$$p_i = ((P_x + P_x \times n) < 1.0)? (P_x \times n) : (1.0 - (1.0 - P_x)/n). \quad (6)$$

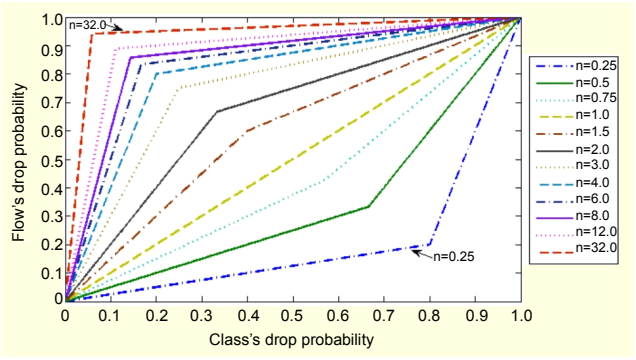

Fig. 3. Drop probability of flow vs. class.

2. Advanced Traffic Management in Flow-Based Router

A. Dynamic Flow Identification (DFI)

In this section, we explore more enhanced mechanisms using more information, which we can derive from the flow state entry. One current Internet problem is that P2P traffic is very prevalent and takes up a large proportion of the network bandwidth. In order to cope with this problem, traffic analysis tools have been developed which identify P2P traffic with deep packet inspection (DPI). Deep packet inspection is based on pattern matching on various IP header and payload fields. This can create a serious burden to the system and makes it inapplicable in high-speed networks.

We propose a useful mechanism to identify flows on the basis of traffic characteristics, such as flow duration, average packet byte, average flow rate, and so on, which are maintained per flow rather than by the packet header or payload. If these traffic characteristics are known for each application, we can identify the type of application of a flow more easily and differentiate P2P traffic from normal traffic. Some studies to extract the traffic characteristic information of applications have been done [21]. We assume that the traffic characteristic information for some applications can be obtained by those methods.

There are two options to differentiate dynamically identified traffic. First, we can adjust the drop probability for the traffic. If traffic of a flow is decided to be an application that should be processed with higher priority, we decrease the drop probability. If the traffic of a flow is decided to be an application with low priority, such as P2P traffic, we increase the drop probability. As a mechanism to achieve this goal, dynamic traffic identification can be used to adjust a flow's drop probability. As a simple extension to the flow drop probability factor presented in the previous section, we can adjust the flow drop probability factor calculation to penalize traffic that exceeds the thresholds of some traffic characteristic parameters, while keeping the previous approach for other normal traffic. We define the adaptive drop priority (ADP) value as a value to adjust the flow drop probability factor as follows:

$$\begin{aligned} ADP = & \max\left(\alpha, \alpha \min\left(\frac{FlowLifeTime}{FlowLifeTime_{threshold}}, \max_{adp}\right)\right) \\ & + \max\left(\beta, \beta \min\left(\frac{AvgFlowRate}{FlowRate_{threshold}}, \max_{adp}\right)\right) \\ & + \max\left(\gamma, \gamma \min\left(\frac{AvgPktSize - PktSize_{threshold}}{MTU - PktSize_{threshold}}, \max_{adp}\right)\right) \\ & + \max\left(\delta, \delta \min\left(\frac{ByteCount}{ByteCount_{threshold}}, \max_{adp}\right)\right), \end{aligned} \quad (7)$$

where each of α, β, γ, and δ is greater than or equal to zero, and $\alpha+\beta+\gamma+\delta = 1$.

The current design of ADP only considers flow lifetime (*FlowLifeTime*), average flow rate (*AvgFlowRate*), average packet size (*AvgPktSize*), and the total byte count of each flow (*ByteCount*), but if we can identify any other distinctive traffic characteristics of a flow, we can add the parameters to the above equation. In (7), the values α, β, γ, and δ, work as weight factors in each traffic parameter. If we do not want to

consider a parameter to calculate ADP, we can just set its weight factor to zero. The value of max_{adp} is used to limit the value of ADP to a certain maximum value. The traffic characteristic of each application is beyond the scope of this paper, but if the characteristics are analyzed clearly, the ADP approach could be used to control specific applications effectively.

Now we can again modify the flow probability calculation. Equation (6) is not changed, but the normalized flow drop probability factor n in (4) is changed:

$$n = (\hat{r}_i / \tilde{r}_i) \times \mathrm{ADP} \times C_2, \quad (8)$$

where C_2 is a constant value.

The threshold-based dynamic flow identification (DFI), its ADP value calculation, and its adaptation to the flow drop probability factor are somewhat limited since they depend only on the threshold values. Therefore, we can also adopt the range concept on the values. This range-based DFI is not directly applied to the flow drop probability calculation; rather, it can be used to give the identified traffic strict priority. The priority could be lower or higher than that of normal traffic. That is, if the traffic is given a higher priority, the traffic will never be dropped until the congestion becomes severe. If the traffic is given a lower priority, the traffic is dropped before dropping the normal traffic.

B. Flow-Based Integrated Packet Scheduling Combining RCFQ and AFRED

In the previous sections, we proposed traffic management mechanisms for guaranteed traffic and non-guaranteed traffic, RCFQ and AFRED, respectively. Both algorithms were proposed for their own specific traffic types: guaranteed and non-guaranteed traffic. However, we did not mention how the two mechanisms should cooperate with each other when they coexist in the system. In this section, we propose a harmonization method for RCFQ and AFRED.

The time to send for packets in RCFQ is clear, and the packets are scheduled to the appropriate time slot in the scheduler. However, the time to send for packets in AFRED is not defined clearly. It is usually assumed to be allocated to the current time as in legacy active queue management schemes. Therefore, if both kinds of traffic are allocated to the same time in the heavy congestion status, there will be cases in which guaranteed traffic is affected by non-guaranteed traffic. To address this issue, we adopted a spacing mechanism for non-guaranteed traffic. Non-guaranteed traffic is spaced properly so that it has a minimal effect on guaranteed traffic. We achieve this goal by adopting a packet finishing time concept for non-guaranteed traffic in a class. However, we use a class's average bandwidth instead of using the each flow's rate. The packet finish time of the k-th packet of class $x(F_x^k)$ is calculated as follows, while still being randomly dropped according to the congestion status by AFRED:

$$F_x^k = \max(F_x^{k-1}, T_{now} + \min(Q_x/R_x, Q_{\max}/R_x)), \quad (9)$$

where Q_{max} is the maximum queue for the class.

With (9), the time-to-send of non-guaranteed traffic is decided clearly to minimize the effect on the guaranteed traffic. We can also use multiple priority queues for guaranteed traffic and non-guaranteed traffic. However, we need to space the non-guaranteed traffic to minimize the probability that one time-slot is congested.

3. Implementation of Flow-Based Traffic Management

From the point of view of implementation complexity, it is clear that the complexity of a flow-based router is much higher than that of a legacy class-based traffic management scheme since the number of flows that should be handled by the flow-based router can be up to several million, compared to 64 in a DiffServ router. However, complexity can be overcome as the hardware technology advances, and its speed increases as time goes on. We can also reduce the implementation complexity by using a calendar queue in the shared memory architecture, while applying efficient algorithms for guaranteed and non-guaranteed traffic. In section IV, we will demonstrate that packets can be scheduled effectively when the proposed scheduling algorithms are used.

As the algorithms for the enqueue process were discussed in section III for both guaranteed and non-guaranteed traffic, the time to send for non-guaranteed traffic is clearly chosen according to the class's congestion status. More specifically, the time to send of a packet is chosen according to the average rate of the class to which the flow of the packet is allocated, while packets are dropped according to the congestion status of the class. Therefore, there is no need to use per-flow queuing; rather, a calendar queue using the proposed algorithms is perfectly suitable for implementation of the proposed mechanisms.

IV. Performance Analysis

In this section, we analyze the traffic management mechanisms that have been proposed in the previous sections. First, we analyze RCFQ numerically. We did not build any concrete numerical methodologies to analyze the active queue management schemes; therefore, we provide simulation results using an event driven network simulation tool, NS-2 [22], and test results from the real system that we have implemented to

demonstrate the performance of the proposed algorithms.

1. Numerical Analysis

A. Delay Analysis of RCFQ

The delay limitation for the RCFQ is clear since the algorithm is non-work-conserving, and it services each sessions with only the reserved rates. However, we show the delay performance more clearly with the numerical analysis in this section. As in other scheduling algorithm analyses, we use the fluid flow model for RCFQ (f-RCFQ). The fluid flow model assumes that a packet is serviced in an infinitesimal unit [5]-[7]. In a normal packet-by-packet system, no other packets are serviced until the service of a packet finishes. However, packets can be serviced at the same time as other packets in a fluid flow model scheduler. Before performing the analysis, we define two concepts.

- A session busy period is a maximal interval of time (τ, t) which satisfies $A_i(\tau, t) \geq r_i(t-\tau)$.
- A system busy period is a maximal interval of time during which there is at least one busy session.

Note that the above definitions are defined with respect to a non-work-conserving system. The system busy period in a work-conserving system is different from the above definition. In a work-conserving system, it is defined as a maximal interval of time during which the server is never idle. However, in the non-work-conserving system, there could be some time intervals during which the server is idle in the system busy period.

Lemma 1. In the fluid model of RCFQ (f-RCFQ), the order of service completion of packets is the same regardless of the arrival patterns of other packets.

Proof. Since the f-RCFQ algorithm is non-work-conserving, the rate of service for each session does not change with future packet arrivals. □

Lemma 2. In the packet-by-packet model of RCFQ, we have the following equality:

$$t_p^k \leq t_f^k + \frac{L_{\max}}{r}, \tag{10}$$

where t_p^k and t_f^k are the departure times of k-th packet in RCFQ and f-RCFQ, respectively.

Proof. The proof of the lemma is very intuitive. The f-RCFQ algorithm is a zero latency server. That is, if the first packet of a session comes into the system, the packet begins to be serviced without delay. However, in an RCFQ system, even though the packet should be serviced right away, if a packet from another session is already being serviced, the packet should wait until the last bit of the packet has departed. Therefore, the packet should wait throughout the time during which a packet with maximum size is serviced with the system capacity in the worst case. □

Theorem 1. The latency of RCFQ is

$$\frac{L_{\max}}{r} + \frac{L_i}{r_i}, \tag{11}$$

where L_{max} is the maximum packet size in the system, and L_i is the maximum packet size of session i.

Proof. Let $W_i^{RCFQ}(\tau,t)$ and $W_i^{f-RCFQ}(\tau,t)$ be the amount of session i traffic served by RCFQ and f-RCFQ in the time interval $[\tau, t]$. We should note that a packet is considered to be serviced when the last bit of the packet has left the system. Therefore, the difference between $W_i^{RCFQ}(\tau,t)$ and $W_i^{f-RCFQ}(\tau,t)$ reaches maximum just before the last bit of a packet is serviced in RCFQ. Let the time instant be t_p. At t_p, the service offered to session i in the RCFQ will be equal to the following, which is the service offered to session i by the f-RCFQ minus the size of the packet being serviced:

$$\begin{aligned} W_i^{RCFQ}(\tau,t_p) &= W_i^{f-RCFQ}(\tau, t_p) - L_i \\ &\geq W_i^{f-RCFQ}(\tau, t_f) - L_i. \end{aligned}$$

From the lemma 2, we have

$$\begin{aligned} W_i^{RCFQ}(\tau,t_p) &\geq W_i^{f-RCFQ}(\tau, t_p - \frac{L_{\max}}{r}) - L_i \\ &\geq \max(0, r_i(t-\tau-\frac{L_{\max}}{r}) - L_i) \\ &\geq \max(0, r_i(t-\tau-\frac{L_{\max}}{r}-\frac{L_i}{r_i})). \end{aligned}$$ □

B. Fairness Analysis of RCFQ

In terms of fairness, it is clear that this is quite a fair algorithm since it is non-work-conserving and serves the sessions based only on their reserved rate. No flow sends more or less than its reserved rate.

2. Test Results of Traffic Management Mechanisms

In this section, we present performance results of the flow-based router both in NS-2 simulation environment [22] and a system prototype. First, we show the simulation results and then provide test result for the prototype. The network topology for the simulation is shown in Fig. 4. The node r1 runs the proposed algorithms. The links between nodes are set to have a link speed of 1 Gbps and 5 ms delay. In this test, we focus on the performance results for unresponsive traffic. Other test results for diverse traffic types will be studied in a future work. Background traffic of 100 flows is generated by s1 and sent to

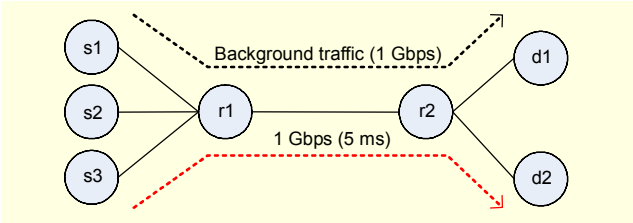

Fig. 4. Test network diagram for simulation.

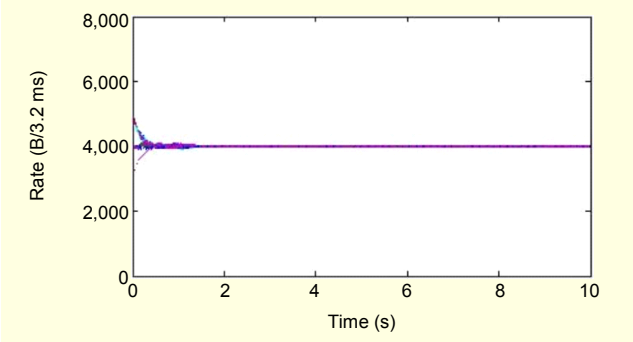

Fig. 5. Throughput changes in FRED.

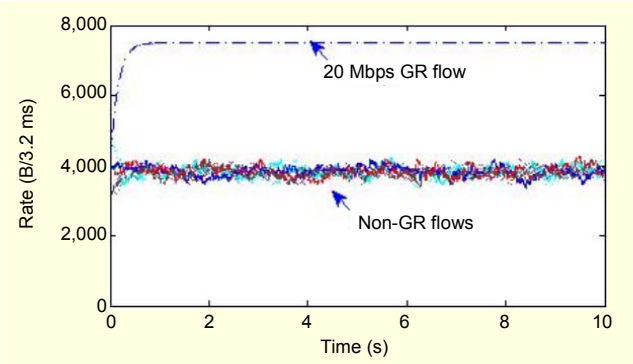

Fig. 6. Throughput changes in AFRED and RCFQ.

d1, and s2 and s3 each generate 50 flows of 20 Mbps to send to s4. Therefore, a total of 2 Gbps traffic is trying to be sent to d2. A CBR traffic generator was used to send the FTP application. The packet size was set to 1,000 bytes. The throughput is calculated as the total number of bytes which has been transmitted per 3.2 ms.

Figure 5 shows node r1's throughput when FRED is applied as the traffic management mechanism. As we can expect, the rates of the flows are fairly distributed. In total, 100 flows share 1 Gbps fairly; therefore, each flow receives about 10 Mbps (4,000 bytes per 3.2 ms). However, if we need to guarantee the bandwidth of a certain flow to 20 Mbps, it is not possible in that architecture.

In Fig. 6, our mechanism can guarantee 20 Mbps for a certain flow, and the rest of the bandwidth can be fairly shared among the other flows. One of the 100 flows is set to be

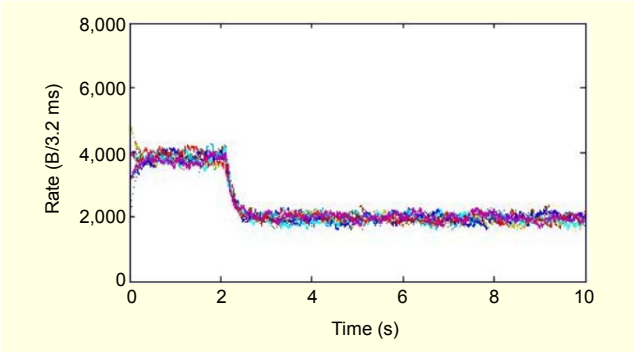

Fig. 7. Throughput changes for dynamically identified flows. Action: class change.

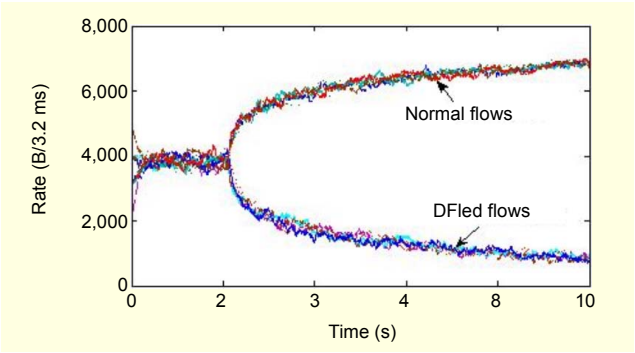

Fig. 8. Throughput changes for dynamically identified flows. Action: ADP.

guaranteed to 20 Mbps. The exact bandwidth is guaranteed by (1) for the guaranteed traffic while the rest of the bandwidth is shared by the other flows.

In Figs. 7 and 8, we show how DFI works, which is one of the unique characteristics of the proposed flow-based traffic management scheme. We can set several different traffic characteristics for this feature, such as flow lifetime, average bytes, packet count, and average rate. In this test, we set the flow's lifetime threshold to two seconds as one of the DFI thresholds for all the flows; therefore, flows taking more than two seconds to be sent would be dynamically identified. Then, we can apply a different class for the dynamically identified flows or increase the drop probabilities of the flows to be higher than those of the normal flows. Figure 7 shows how the bandwidth is limited after two seconds when we limit the total bandwidth of flows to which DFI has been applied to the 5 Gbps class. Initially, each flow receives about 20 Mbps. However, since the flows are dynamically identified after two seconds, and the classes of the flows are changed to the 5 Gbps class, each of 100 flows starts to receive 5 Mbps (2,000 bytes per 3.2 ms) from then on. If the dynamically identified flows are known to be misbehaving flows, we can effectively limit the whole misbehaving traffic to a certain amount of bandwidth. Figure 8 shows how the bandwidth of dynamically

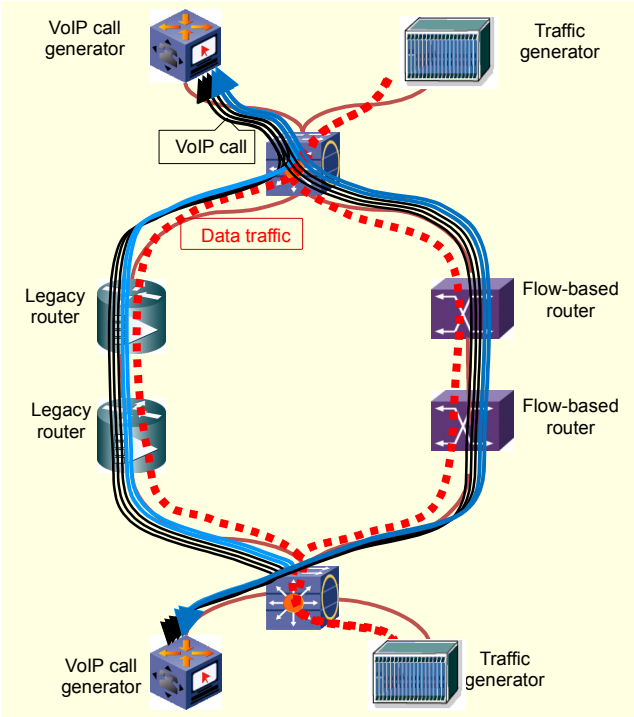

Fig. 9. Test network diagram for real system.

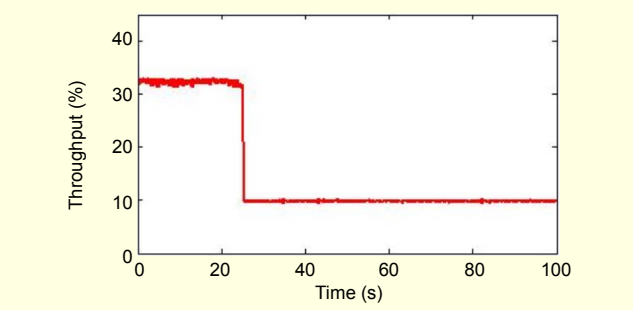

Fig. 10. Throughput changes for dynamically identified flows. Action: class change.

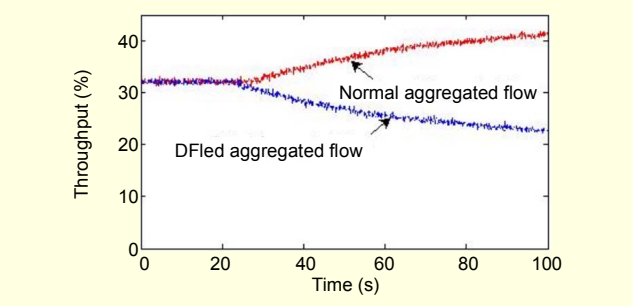

Fig. 11. Throughput changes for dynamically identified flows. Action: ADP.

identified flows starts to decrease as the dropping probability increases according to (6) to (8) after they meet one of the DFI thresholds. The bandwidth of the dynamically identified flows decreases more as they get further from the DFI threshold. The remainder of the bandwidth is given to the normal flows.

Here, we present some performance results of flow-based routers and legacy class-based routers in a real test environment. We set up the test network as shown in Fig. 9. Two flow-based routers and two legacy class-based routers were used. A Spirent AX/4000 traffic generator was used to send the traffic between the systems being tested [23]. Also, a Charriot client and server were used to test VoIP performance [24]. Two traffic classes were set up in the system with maximums of 640 Mbps and 100 Mbps, respectively. The traffic generator generated 64 kbps for each of the 10,000 flows that were included in 640 Mbps class. The AX4000 allowed us to use only aggregate flow mode to generate more than 5,000 flows. Thus, we used two flow aggregates of 5,000 flows each. As a result, each flow aggregate sent 320 Mbps. All the source addresses of the flows in one flow aggregate were set to 10.0.0.0, and those for the other flow aggregate were set to 20.0.0.0. We set a DFI threshold for the traffic with source address 20.0.0.0 to the flow duration of 25 s. Therefore, if a flow with source address 20.0.0.0 took more than 25 s to be sent, it would be dynamically identified.

Figure 10 shows how throughput of a flow aggregate changes when we apply a different class for the dynamically identified flows. For the dynamically identified flows, we changed their class to the smaller class of 100 Mbps. The *x* axis represents testing time and the *y* axis represents the percentage of port bandwidth. Initially, each of the two aggregated flow groups sends about 320 Mbps. However, the throughput of one of the flow aggregates (flows with source address 10.0.0.0) drops to 100 Mbps at about 25 s. This is because the flows in the flow aggregate are dynamically identified, and then the class of the flows is changed to the 100 Mbps class. Figure 11 shows how throughput of a flow changes when we apply ADP for the dynamically identified flows. At the beginning of the test, each of the two flow aggregate groups sent about 400 Mbps. However, since the total bandwidth was limited to the 640 Mbps class, the throughput of each flow aggregate group was 320 Mbps. The throughput of one flow aggregate group dropped gradually after 25 s when the DFI threshold for flow duration was met. The drop probability increased according to (6) to (8) for the dynamically identified flows. On the other hand, the normal flows received more throughput as time went on since the flows could use the extra bandwidth that dynamically identified flows in that class could not use. As shown in Figs. 6 to 8, the results of both tests are almost identical to the simulation results.

Our tests also show that the proposed method guarantees for each flow better packet loss, MOS, delay, and jitter characteristics than those of the legacy class-based router

system under network congestion. A Chariot client was attached to one of the systems under test (either the flow-based router or the legacy class-based router) and a Chariot server was attached to the other system under test. Background traffic flowed at 1 Gbps between the two systems being tested. We set the VoIP traffic to the highest priority in the class-based router. However, the maximum queue percentage to allocate to the priority queue was 30%. Therefore, if the highest input traffic exceeded the highest priority queue, the other traffic used a lower priority queue. In the flow-based router, each VoIP flow was reserved to be serviced at 87 kbps, which is the exact generating rate of a VoIP call in Chariot.

Table 1 summarizes the performance results for the class-based router and the proposed flow-based router. When we sent 100 VoIP calls with 1 Gbps background traffic, the performance of the two systems was almost the same (case 1). However, when we sent 20 more VoIP calls, it made the new VoIP calls exceed the priority queue. Therefore, some VoIP calls had to contend with the background traffic in the legacy router, which led to some performance degradation. On the other hand, bandwidth was allocated and reserved for the guaranteed traffic in the proposed flow-based router, and the performance results were the same as those of case 1.

Table 1. Performance comparison of legacy router and proposed flow-based router.

		Legacy router	Flow-based router
Case 1	Packet loss	0 %	0 %
	MOS	4.38	4.38
	Delay	5 to 9 ms	1 ms
	Jitter	0 to 1 ms	0 to 1 ms
Case 2	Packet loss	0 to 54%	0 %
	MOS	0.99 to 4.4	4.3 to 4.38
	Delay	13 to 16 ms	1 ms
	Jitter	0 to 6 ms	0 to 1 ms

V. Conclusion

We proposed traffic management mechanisms to support QoS for flow-based routers. Separate traffic management mechanisms (RCFQ and AFRED) were proposed for guaranteed traffic and non-guaranteed traffic, respectively. Then, AFRED was extended so that guaranteed traffic could be isolated from non-guaranteed traffic in real networks in which both traffic types are mixed together. The extension was done by incorporating the fair queuing concept into the active queue management mechanism. Non-guaranteed traffic was shaped to minimize the effect on the guaranteed traffic in the mechanism. In addition, we proposed a dynamic traffic identification method to dynamically prioritize traffic according to the traffic characteristics of applications. The drop probability can be changed or a different traffic class can be allocated to dynamically identified traffic. For high-speed router systems, the dynamic traffic identification method is a good alternative to deep packet inspection, which requires significant processing power to handle the IP packet header and payload. We demonstrated the performance of the proposed mechanisms by the numerical analysis, simulation, and real system tests. We believe that the architecture is suitable for flow-based high-speed routers, which handle millions of flows at wire speed.

References

[1] R. Braden et al., *Integrated Services in the Internet Architecture: An Overview*, IETF RFC 1633, June 1994.

[2] S. Blake et al., *An Architecture for Differentiated Services*, IETF RFC 2475, Dec. 1998.

[3] K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," *Proc. IEEE INFOCOM*, vol. 2, May 1992, pp. 915-924.

[4] B.H. Choi and H.S. Park, "Rate Proportional SCFQ (RP-SCFQ) Algorithm for High-Speed Packet-Switched Networks," *ETRI Journal*, vol. 22, no. 3, Sept. 2000, pp. 1-9.

[5] D. Stiliadis and A. Varma, "Efficient Fair Queuing Algorithms for Packet-Switched Networks," *IEEE/ACM Trans. Networking,* vol. 6, no. 2, Apr. 1998, pp. 175-185.

[6] D.Y. Kwak, N.S. Ko, B. Kim, and H.S. Park, "A New Starting Potential Fair Queuing Algorithm with O(1) Virtual Time Computation Complexity," *ETRI Journal*, vol. 25, no. 6, Dec. 2003, pp. 475-488.

[7] F.M. Chiussi, A. Francini, and J.G. Kneuer, "Implementing Fair Queuing in ATM Switches, Part 2: The Logarithmic Calendar Queue," *Proc. IEEE INFOCOM*, vol. 1, Nov. 1997, pp. 519-525.

[8] L. Zhang, "Virtual Clock: A New Traffic Control Algorithm for Packet Switching," *ACM Trans. Computer Systems*, vol. 9, no. 2, May 1991, pp. 101-124.

[9] N. Yamagaki, H. Tode, and K. Murakami, "DMFQ: Hardware Design of Flow-Based Queue Management Scheme for Improving the Fairness," *IEICE Trans. Comm.*, vol. E88-B, no. 4, Apr. 2005, pp. 1413-1423.

[10] D. Yamamoto, H. Tode, T. Masaki, and K. Murakami, "Design and Empirical Evaluation of Control Scheme for End-to-End Delay Stabilization and Packet Loss Improvement in Broadband IP Network," *IEEE ICCCN*, TP9, Hawaii, USA, Aug. 2007.

[11] R. Fujita, H. Shimabara, H. Tode, T. Masaki, and K. Murakami

"QoS Control Scheme Guaranteeing the Delay, Jitter and Throughput in the IP Router," *IEEE LCN*, Tampa, USA, Nov. 2004, pp. 413-414.

[12] R. Brown, "Calendar Queues: A Fast 0(1) Priority Queue Implementation for the Simulation Event Set Problem," *Comm. of the ACM*, vol. 31, no. 10, Oct. 1998, pp. 1220-1227.

[13] Z. Cao and Z. Wang, "Flow Identification for Supporting Per-Flow Queuing," *Computer Comm. and Networks*, Oct. 2000, pp. 88-93.

[14] W. Feng, K.G. Shin, D. Kandlur, and D. Saha, "The BLUE Active Queue Management Algorithms," *IEEE/ACM Trans. Networking*, vol. 10, no. 4, Aug. 2002, pp. 513-528.

[15] D. Lin and R. Morris, "Dynamics of Random Early Detection," *IEEE/ACM Trans. Networking*, Aug. 1993.

[16] F.A.L. Raddady and M. Woodward, "A New Adaptive Congestion Control Mechanism for the Internet Based on RED," *AINA Workshops*, vol. 2, May 2007, pp. 934-939.

[17] J. Hong, C. Joo and S. Bahk, "Active Queue Management Algorithm Considering Queue and Load States," *Computer Comm.*, vol. 30, no. 4, Feb. 2007, pp. 886-891.

[18] M. Shin, S. Chang, and I. Rhee, "Dual-Resource TCP/AQM for Processing-Constrained Networks," *INFOCOM*, Apr. 2006.

[19] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," *http://www.icir.org/floyd/ papers/adaptiveRed.pdf*, Aug. 2001.

[20] S. Floyd and V. Jacobson, "Random Early Detection for Congestion Avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, Aug. 1993, pp. 397-413.

[21] J. Erman, A. Mahanti, and M. Arlitt, "Internet Traffic Identification Using Machine Learning," *Proc. IEEE Globecom*, Nov. 1996, pp. 1-6.

[22] The official NS-2 webpage, http://nsnam.isi.edu/nsnam/index.php/User_Information.

[23] AX/4000, http://www.spirentcom.com.

[24] NetIQ Chariot, http://www.netiq.com.

Nam-Seok Ko received the BS degree in computer engineering from Chonbuk National University, Korea, in 1998, and the MS degree in engineering from the Information Communications University, Korea, in 2000. In 2001, he joined ETRI and has participated in several R&D projects including ATM switching systems and high-speed router systems. Currently he is working towards the PhD degree with the School of Engineering of ICU. His research interests include traffic engineering mechanisms for QoS and fixed mobile convergence technologies including mobility management protocols.

Sung-Back Hong received the BS degree in electronics and telecommunication engineering from Kwangwoon University, Korea, in 1982; the MS degree in electronics engineering from Yonsei University, Korea, in 1990; and the PhD degree in information and communication engineering from Chungbuk University in 2008. Since joining ETRI in 1982, he has been involved with many government-sponsored projects. He is currently the leader of the FMC Technology Team of ETRI. His research interests include fixed-mobile convergence networks and ubiquitous sensor networks and applications.

Kyung-Ho Lee received the BS and MS degree in electronics and telecommunication engineering from Kwangwoon University, Korea, in 1980 and 1982, respectively. Since joining ETRI in 1982, he has participated in many R&D projects including TDX digital switching systems and ATM switching systems. He is currently with the FMC Technology Team of ETRI. His research interests include fixed-mobile convergence networks, QoS technologies in BcN networks, and ubiquitous sensor networks and applications.

Hong-Shik Park received the BS degree in electrical engineering from Seoul National University, Seoul, Korea, in 1977. He received the MS and PhD degrees in electrical engineering from KAIST, Daejeon, Korea, in 1986 and 1995, respectively. In 1977, he joined ETRI, where he worked on the development of the TDX digital switching system family including TDX-1, TDX-1A, TDX-1B, TDX-10, and ATM switching systems. In 1998, he moved to ICU, Daejeon, Korea, where he is currently a professor. His research interests are network architecture, network protocols, and performance analysis of telecommunication system. He is a member of the IEEK and KICS, Korea.

Nam Kim received the BS, MS, and Ph.D. degrees from Yonsei University, Korea in 1981, 1983, and 1988, respectively. He is currently working as a professor in Chungbuk National University in Cheongju, Korea. His research interests include optical networks and mobile communication.