논문 02-27-4C-8 한국통신학회논문지 '02-4 Vol.27 No.4C

DiffServ 방식에서 Assured Service의 QoS 보장을 위한 RIO 및 RIO-DC 방식의 성능 비교

정회원 허 경*, 신 동 범**, 이 상 우**, 엄 두 섭*, 차 균 현*

Performance comparisons of RIO and RIO-DC for QoS guarantee of the Assured Service in Differentiated Services

Kyeong Hur*, Dongbeom Shin**, Sangwoo Lee**, Doo-Seop Eom*,
Kyun Hyon Tchah* *Regular Members*

요 약

본 논문은 IETF에서 DiffServ의 Assured Service를 위하여 제안된 AF PHB의 버퍼 관리 방식들 중 RIO 및 RIO-DC 방식의 성능을 비교 평가하였다. 이를 위하여 Assured Service의 서브 클래스별로 할당하는 대역폭의 비율을 차별화하여 서브 클래스별로 보장하는 최대 지연시간을 상대적으로 차등화 하였다. 또한, 네트워크 토폴로지와 Assured Service의 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 RIO 및 RIO-DC 방식의 변수 값을 설정하였다. 이러한 환경에서, In-profile 트래픽에 대한 수율, 링크 이용률 및 플로간 공평성을 성능의 척도로 하여 제안된 두 방식의 성능을 비교하였다. 시뮬레이션 결과는 제안하는 변수 설정 방안을 적용한 RIO 및 RIO-DC 방식의 Assured Service에 대한 In-profile 트래픽의 수율과 링크 이용률 성능은 동등하나, RIO-DC 방식은 RIO 방식 보다 향상된 플로간 공평성을 보장할 수 있음을 보인다.

ABSTRACT

In this paper, we compare the performances of RIO and RIO-DC buffer management schemes for DiffServ AF PHB standardized in IETF. For the comparison, we relatively differentiate maximum delay for each Assured Service subclass in Differentiated Services by allocating bandwidth to each subclass differently. In addition, we set the values of RIO and RIO-DC parameters considering the buffer size determined by the network topology and the ratio of bandwidth allocated to each subclass. In this simulation environment, the performances of RIO and RIO-DC schemes are analyzed focusing on the throughput of the In-profile traffic, the link utilization and the fairness. Simulation results show that the performance of RIO-DC scheme is comparable to that of RIO scheme with regard to the throughput of the In-profile traffic and the link utilization. However, under the simulation condition RIO-DC scheme improves the fairness between flows much better than RIO scheme.

I. 서 론

사용자가 요구하는 QoS를 보장할 수 있는 차세대 인터넷에 대한 구조로서 DiffServ (Differentiated Services) 방식은 DiffServ Code Point(DSCP)를 이용하여 IP 패킷에 대한 PHB(Per Hop Behavoir)를 규정한다[1]. DiffServ 도메인에 도착한 사용자 플로의 패킷들에 대해 DSCP가 정해지면, 같은 DSCP 코드를 가진 모든 패킷들은 동일한 방식으로 처리된다. 이와 같이 다수의 서로 다른 플로들로 구성된 트래픽은 소수의 클래스들로 분류된다. 이러한 집합(Aggregate) 개념의 메커니즘은 대규모의 플로들을 포워딩하는 내부 네트워크(Core Network)에 적합한

* 고려대학교 전자공학과 (hkyeong@korea.ac.kr)
** 한국전자통신연구원 네트워크 연구소
논문번호 : 010283-1015, 접수일자: 2001년 10월 15일

확장성을 갖고 기존의 IntServ(Integrated Services) 방식의 문제점을 해결할 수 있다[2]. 제안된 DiffServ의 PHB 방식에는 PS(Premium Service)에 해당하는 EF(Expedited Forwarding) PHB와 AS(Assured Service)에 해당하는 AF(Assured Forwarding) PHB가 있다[3][4]. PS는 ATM(Asynchronous Transfer Mode) 네트워크에서 제공되는 CBR(Constant Bit Rate) 특성의 가상 전용선(Virtual Leased Line : VLL)과 유사한 수준의 End-to-End QoS를 제공하고 AS는 PS에 비하여 상대적으로 낮은 수준의 End-to-End QoS 보장성을 갖지만 버스트한 트래픽 특성을 허용한다[5][6].

네트워크에서 사용자에게 일정 수준의 QoS를 보장하기 위해서는 접속 제어(Admission Control) 및 혼잡 제어(Congestion Control)가 필요하다. 기존의 ATM 네트워크에서는 VC(Virtual Circuit) 단위로 플로들의 정보를 바탕으로 한 접속제어 및 Rate-based Congestion Control을 실시하였고, IP 망의 경우는 플로별 트래픽 관리를 위한 시그날링 프로토콜인 RSVP(Resource ReSerVation Protocol)를 이용하여 접속제어를 수행하고 네트워크 혼잡의 발생을 방지할 수 있었다[7]. DiffServ 방식에서는 플로 단위의 정보관리를 실시하지 않는 특성에 적합한 접속제어 방안이 연구되고 있으며, AS 트래픽에 대한 Congestion Control 방안으로 그림1과 같이 RED (Random Early Detection)를 확장한 MRED (Multi-level RED) 방식들이 연구되고 있다 [8]. RED 버퍼 관리 방식은 버퍼가 가득 차기 전에 큐에 도착하는 패킷들을 확률적으로 폐기하여 평균 큐 길이를 제어할 수 있고 따라서 패킷들의 평균 지연시간을 제어할 수 있다[9]. 또한 RED 방식은 Drop-Tail 방식에서 많은 수의 플로들에서 패킷들이 동시에 폐기되어 네트워크의 링크 대역폭에 대한 이용효율을 감소시키는 Global Synchronization의 문제점을 보완하여 수율(Throughput) 및 링크 이용률을 향상시킬 수 있다. 이러한 RED 방식에 대해서는 병목 구간에서 효율적인 혼잡 제어를 위한 RED 변수 값을 설정하는 방안에 관한 연구가 있었다[10].

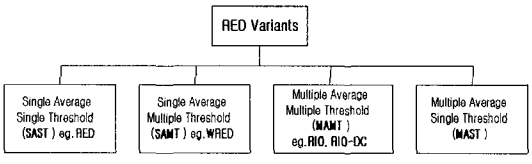


그림 1. AF-PHB를 위한 Active Queue Management 방식

그림 2의 RIO(RED with In and Out) 방식은 사용자와 네트워크 간에 약속한 Traffic profile을 준수하는 In-profile 패킷과 그렇지 못한 Out-of-profile 패킷들에 대해 서로 다른 패킷 폐기 기준을 설정하여 네트워크 혼잡 시 In-profile 패킷을 우선적으로 보호하고 혼잡이 없는 경우는 Out-of-profile 패킷들을 이용하여 링크 이용률을 향상시키기 위한 목적으로 제안된 것이다[11]. 즉, RIO 방식은 큐 내에 저장된 In-profile 패킷들의 수에 따른 평균 큐 내 In-profile 패킷들의 수, $avgIn$에 따라 도착하는 In-profile 패킷의 패킷 폐기를 결정하고, 평균 큐 내 전체 패킷들의 수, $avg-total$에 따라 도착하는 Out-of-profile 패킷의 패킷 폐기를 결정한다. 그리고 RIO-DC(RED with In/Out and De-Coupled Queues) 방식은 In-profile 패킷에 대한 패킷 폐기 기준은 동일하나, 도착하는 Out-of-profile 패킷에 대해 큐 내에 저장된 Out-of-profile 패킷들의 수에 따른 평균 큐 내 Out-of-profile패킷들의 수, $avg-out$값에 따라 Out-of-profile 패킷의 패킷 폐기를 결정한다. 현재 IETF(Internet Engineering Task Force)에서는 RFC 2597에서 제안된 AF PHB를 위한 그림 1의 Active Queue Management 방식들 중 WRED(Weighted RED) 방식과 RIO 방식에 대해 병목 구간의 라우터에서 낮은 패킷 폐기 순위를 갖은 트래픽을 우선적으로 보호할 수 있는 방안이 연구되고 있으며 RIO-DC 방식에 대해서는 연구결과가 제시되지 않고 있다[8][12]. 한편, DiffServ 방식에서 AS를 사용하는 TCP 플로에게 Minimum Rate의 QoS를 보장하기 위해서는 접속 제어가 필요하고 사용자가 계약한 In-profile 패킷 트래픽에 대한 보호 및 수율 보장이 요구된다[13]. 그러나 접속 제어가 수행된 상황 하에서 In-profile 트래픽에 대한 수율과 링크 이용률을 극대화 시킬 수 있는 버퍼 관리 방식에 대해서는 아직 개념 정립 상태에 머물러 있다.

본 논문에서는 IETF에서 DiffServ의 Assured Service를 위하여 제안된 AF PHB의 버퍼 관리 방식들 중 RIO 및 RIO-DC 방식의 성능을 비교 평가

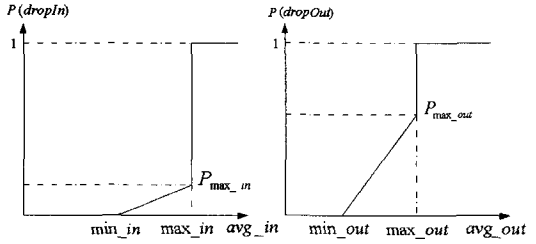


그림 2. RIO-DC 버퍼 관리 방식

하였다. 이를 위하여 Assured Service의 서브 클래스별로 할당하는 대역폭의 비율을 차별화하여 서브 클래스별로 보장하는 최대 지연시간을 상대적으로 차등화 하였다. 또한, 네트워크 토폴로지와 Assured Service의 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 RIO 및 RIO-DC 방식의 변수 값을 설정하였다. 이러한 환경에서, In-profile 트래픽에 대한 수율, 링크 이용률 및 플로간 공평성을 성능의 척도로 하여 제안된 두 방식의 성능을 비교하였다. 본 논문의 구성은 다음과 같다. 제 2 절에서는 AS의 자원 할당에 따른 RIO 및 RIO-DC 방식의 변수 설정 방안을 제안한다. 제 3 절의 시뮬레이션 모델 및 결과는 제안하는 변수 설정 방안을 적용한 RIO 및 RIO-DC 방식의 Assured Service에 대한 In-profile 트래픽의 수율과 링크 이용률 성능은 동등하나, RIO-DC 방식은 RIO 방식 보다 향상된 플로간 공평성을 보장할 수 있음을 보인다. 끝으로 제 4 절에서 결론을 맺는다.

Ⅱ. Assured Service에 대한 RIO-DC 방식의 변수 설정

DiffServ 방식에서 규정된 AS In-profile 패킷의 트래픽 특성은 사용자가 신고한 평균 전송률 r_i 및 최대 전송률 p_i와 사용자와 연결된 DiffServ 도메인의 입구 라우터(Leaf Router)에 있는 트래픽 성형기(Traffic Conditioner) 내 토큰 버킷(Token bucket)의 크기 t_s와 관련된 버스트 길이 l_i(msec)의 세가지 요소로 규정된다[14]. 토큰 버킷의 크기와 AS 트래픽 요소들 간의 관계는 식(1)과 같다. 이러한 AS In-profile 패킷들의 트래픽 특성에 따라 AS의 j 번째 서브 클래스에서 할당하는 대역폭의 양을 Over-provisioning factor μ_j값을 사용하여 $\mu_j r_i$로 결정하면 서브 클래스별로 제공하는 최대 지연시간을 상대적으로 차등화할 수 있다. 즉, 식(1)과 같이 사용자가 요구하는 평균 전송률 r_i에 대해 서브 클래스별로 $\mu_j r_i$의 대역폭을 할당하면 서브 클래스별로 보장하는 최대 지연시간, $d_{\max,j}$는 플로별 트래픽 정보에 따라 상대적으로 차등화가 이루어진다.

$$t_s = (p_i - r_i)l_i = r_i(b_i - 1)l_i, \quad b_i = p_i / r_i$$

$$d_{\max,j} = \frac{(r_i b_i - \mu_j r_i)l_i}{\mu_j r_i} \quad 1 \le \mu_j \le b_i \quad if\ \mu_j = b_i, then\ PS$$

$$if\ \mu_i < \mu_j \quad d_{\max,i} > d_{\max,j} \qquad (1)$$

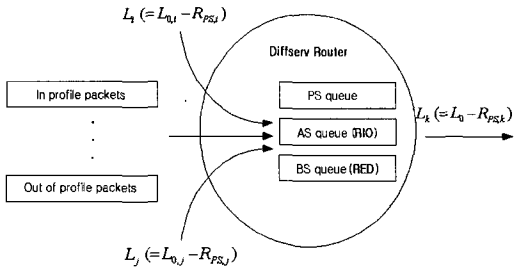


그림 3. 네트워크 토폴로지를 고려한 DiffServ 라우터에서의 AS 트래픽

본 논문에서는 설명의 편의상 그림 3과 같이 임의의 μ_j를 갖는 하나의 AS 서브 클래스를 고려하여 RIO 또는 RIO-DC 방식을 사용하는 하나의 AS 큐를 가정하고 μ_j는 1인 경우만을 고려하기로 한다. 즉, AS에 대해 평균 전송률 r_i의 대역폭을 할당하여 Over-provisioning이 없는 경우를 고려한 것이다. AS 큐에는 In-profile 패킷들과 더불어 Out-of-profile 패킷들이 함께 도착한다. 식(1)로부터 제안하는 시스템에서 AS 사용자에게 보장하는 QoS는 In-profile 패킷들에 대해 $(b_i - 1)l_i$의 최대 지연시간이 된다. 그림 3은 DiffServ 내부 라우터(Core Router)의 네트워크 토폴로지를 고려한 것으로 PS용으로 예약된 자원량이 없는 경우, AS가 사용 가능한 DiffServ 내부 라우터의 출력링크 대역폭을 L_o라고 정의하였다. 또한 시간 u_k에서 PS에 속한 플로들의 BA (Behavior Aggregate)에 대해 예약된 자원량을 $R_{ps,k}$라 할 때 AS 클래스가 사용 가능한 출력 대역폭을 식(2)에서와 같이 L_k로 정의하였다. 동일한 방식으로 주목하는 DiffServ 라우터와 연결된 n개의 입력 링크들 중 j번째 입력 링크를 출력 링크로 하는 이전 DiffServ 라우터에서, AS 클래스가 사용 가능한 대역폭 L_j는 식(2)로 정의된다. 이로부터 식(3)에서 λ_{Tob}은 네트워크 토폴로지에 따른 라우터의 입력대역폭과 출력대역폭 간의 비율를 나타내고 각 $R_{ps,j}$의 합은 $R_{ps,k}$와 같다.

$$L_j = L_{0,j} - R_{PS,j}, \quad L_k = L_0 - R_{PS,k}$$

$L_{0,j}$: j번째 입력링크에서 PS용으로 예약된 자원량이 없는 경우 AS가 사용 가능한 대역폭

$R_{ps,j}$: j번째 입력링크에서 PS용으로 예약된 자원량 (2)

$$\lambda_{top}=\frac{\sum_{j=1}^{n}L_j}{L_k}=\frac{\sum_{j=1}^{n}L_{0,j}-\sum_{j=1}^{n}R_{PS,j}}{L_0-R_{PS,k}}=\frac{\sum_{j=1}^{n}L_{0,j}-R_{PS,k}}{L_0-R_{PS,k}},\ R_{PS,k}=\sum_{j=1}^{n}R_{PS,j} \quad (3)$$

AS 트래픽의 Out-of-profile 패킷은 사용자의 TCP 트래픽이 발생시키는 패킷 양이 사용자와 연결된 DiffServ 도메인의 입구 라우터의 Traffic Conditioner 내에 남아 있는 토큰의 양보다 클 경우에 발생하고 In-profile 패킷들은 남아 있는 토큰의 양만큼 발생한다. 라우터의 출력링크 대역폭을 In-profile 패킷들이 대부분 이용할 수 있도록 해야 하기 때문에, 본 논문에서는 DiffServ 라우터의 AS 큐에서 In-profile 패킷과 Out-of-profile 패킷들이 저장되는 버퍼 공간 크기의 평균적인 비율이 $1:\beta(\beta<<1)$가 되도록 RIO 및 RIO-DC 변수 값을 설정한다. 만약 DiffServ 라우터의 AS 큐에서 In-profile 패킷과 Out-of-profile 패킷들이 저장되는 버퍼 공간 크기의 평균적인 비율이 $1:\beta$가 되면, 출력링크에 대해서도 In-profile 패킷과 Out-of-profile 패킷들이 $1:\beta$의 비율로 출력링크 대역폭 L_k를 이용하게 될 것이다. 즉, In-profile 패킷들은 $L_k/(1+\beta)$의 대역폭을 이용하게 되고 Out-of-profile 패킷들은 $L_k\beta/(1+\beta)$의 대역폭을 이용하게 된다. 또한, AS 큐에서 In-profile 패킷과 Out-of-profile 패킷들이 차지하는 버퍼 공간 크기의 평균적인 비율이 $1:\beta$가 되도록 하기 위해서는 식(4)와 같이 μ_j를 고려한 접속제어를통해 라우터를 경유하는 플로들에서 발생되는 In-profile패킷의 평균전송률의 합이 $L_k/(\mu_j(1+\beta))$의 대역폭을 초과하지 않도록 해야 한다.

$$\mu_j\sum_{i=1}^{n}r_i \le \frac{L_k}{1+\beta}$$

$$\mu_j\sum_{i=1}^{n}r_i b_i l_i \le \frac{L_k\cdot b_{i,\max}\cdot l_{i,\max}}{(1+\beta)} \quad (4)$$

DiffServ 방식은 플로별 정보 관리를 실시하지 않으므로 임의의 시간 구간 τ의 길이를 신고한 플로들의 버스트 길이 l_i들 중 최대값 $l_{i,\max}$으로 설정하면, 그림 3과 같이 μ_j가 1인 하나의 AS 큐의 경우 현재 라우터를 경유하는 n개 플로들에 의해 이전 라우터들로부터 τ동안 도착 가능한 In-profile 패킷들의 최대량은 접속 제어를 통해 $L_k/(1+\beta)$의 대역폭을 모두 예약한 경우로 식(4)와 같이 $L_k b_{i,\max} l_{i,\max}/((1+\beta)packetsize)$가 되고, Out-of-profile 패킷들의 최대 도착량은 $\beta L_k b_{i,\max} l_{i,\max}/((1+\beta)packetsize)$가 된다. 여기서 $b_{i,\max}$, $l_{i,\max}$ 값은 신고한 트래픽 정보들 b_i, l_i들 중 값이 가장 큰 것을 나타낸다. 그리고 각각 이용하는 출력링크 대역폭의 양 $L_k/(1+\beta)$, $L_k\beta/(1+\beta)$에 의해 $\tau(=l_{i,\max})$동안 AS큐에 최대로 남게 되는 패킷양은 In-profile 패킷들이 $L_k(b_{i,\max}-1)l_{i,\max}/((1+\beta)packetsize)$가 되고 Out-of-profile 패킷들은 $\beta L_k(b_{i,\max}-1)l_{i,\max}/((1+\beta)packetsize)$가 된다. 그러나 일반적으로 $\lambda_{\tau op}$값은 1보다 크거나 같고 $b_{i,\max}$보다는 작다. 따라서 $\lambda_{\tau op}$이 $b_{i,\max}$보다 작은 경우에는 τ동안 도착 가능한 In-profile 패킷들의 최대량은 $L_k\lambda_{\tau op}l_{i,\max}/((1+\beta)packetsize)$가 되고, Out-of-profile 패킷들의 최대량은 $\beta L_k\lambda_{\tau op}l_{i,\max}/((1+\beta)packetsize)$가 된다. 그리고 τ 동안 AS큐에 최대로 남게 되는 양은 In-profile 패킷들이 $L_k(\lambda_{\tau op}-1)l_{i,\max}/((1+\beta)packetsize)$가 되고 Out-of-profile 패킷들은 $\beta L_k(\lambda_{\tau op}-1)l_{i,\max}/((1+\beta)packetsize)$가 된다. 이 최대 저장량들은 RIO 및 RIO-DC 버퍼 관리 방식에서 보호되어야 하는 패킷양이다. 한편 RED 방식으로부터 RIO 방식에서 $\max_in$과 $\max_out$은 In-profile 패킷과 전체 패킷들이 각각 최대 이용 가능한 버퍼 크기로 설정하고, RIO-DC방식에서 $\max_in$과 $\max_out$은 은 In-profile 패킷과 Out-of-profile 패킷들이 각각 최대 이용 가능한 버퍼 크기로 설정한다[9]. 따라서 RIO방식에서 $\max_in$과 $\max_out$은 $\lambda_{\tau op}$이 $b_{i,\max}$보다 큰 경우에 각각 $L_k(b_{i,\max}-1)l_{i,\max}/((1+\beta)packetsize)$와 $L_k(b_{i,\max}-1)l_{i,\max}/packetsize$로 설정하고, $\lambda_{\tau op}$이 $b_{i,\max}$보다 작은 경우에는 각각 $L_k(\lambda_{\tau op}-1)l_{i,\max}/((1+\beta)packetsize)$와 $L_k(\lambda_{\tau op}-1)l_{i,\max}/packetsize$로 설정한다. 그리고 RIO-DC 방식의 . 은 RIO 방식과 같고, $\max_out$은 $\lambda_{\tau op}$이 $b_{i,\max}$보다 큰 경우에 $\beta L_k(b_{i,\max}-1)l_{i,\max}/((1+\beta)packetsize)$로 설정하고 $\lambda_{\tau op}$이 $b_{i,\max}$보다 작은 경우에는 $\beta L_k(\lambda_{\tau op}-1)l_{i,\max}/((1+\beta)packetsize)$로 설정한다.

RIO 및 RIO-DC방식의 변수 . 과 값의 설정에 있어서 Out-of-profile 패킷들에 대해

패킷 폐기가 적절하게 이루어져야 In-profile 패킷에 대한 수율이 보장될 수 있고 평균 큐 길이를 적절하게 유지할 수 있다. 또한, Out-of-profile 패킷은 링크 이용률 측면에서 매우 중요한 요소이므로 β값은 링크 이용률을 고려하여 산출되어야 한다. 즉, In-profile 패킷들이 하나도 도착하지 않는 상황을 가정하였을 때 Out-of-profile 패킷들이 출력링크를 모두 이용할 수 있어야 한다. 식(5)는 이러한 경우를 고려하여 산출되는 β값을 나타낸 것이다. 그리고 RIO및 RIO-DC 방식에서 식(5)에서 구한 β값과 PS 예약 자원량에 따라 결정되는 max_*out*보다 크게 max_*out*을 설정한다면 증가한 Out-of-profile 패킷들로 인해 In-profile 패킷이 지연을 겪거나 폐기될 수 있고, 작게 설정한다면 링크 이용률이 감소할 수 있다. 또한 RIO 방식은 평균 전체 패킷 수, $avg-total$에 의해 Out-of-profile 패킷의 폐기를 결정함에 따라 Out-of-profile 패킷들이 버퍼 공간을 계산된 β값의 비율 보다 크게 차지할 경우 In-profile 트래픽 수율이 보장되지 않을 수 있으나, RIO-DC 방식은 $avg-total$보다 작은 평균 Out-of-profile 패킷 수, $avg-out$에 따라 직접적으로 Out-of-profile 패킷의 폐기를 결정하므로 In-profile 트래픽에 대한 보호 능력이 RIO 방식보다 우수하다고 할 수 있다.

$$b_{i,\max} < \lambda_{top}, \quad \frac{L_k \cdot \tau}{packet\_size} = \frac{\beta \cdot L_k \cdot b_{i,\max} \cdot \tau}{\mu_j(1+\beta) \cdot packet\_size} \quad \therefore \beta = \frac{\mu_j}{b_{i,\max} - \mu_j}$$

$$b_{i,\max} \geq \lambda_{top}, \quad \frac{L_k \cdot \tau}{packet\_size} = \frac{\beta \cdot L_k \cdot \lambda_{top} \cdot \tau}{\mu_j(1+\beta) \cdot packet\_size} \quad \therefore \beta = \frac{\mu_j}{\lambda_{top} - \mu_j} \tag{5}$$

결과적으로 DiffServ 방식에서 평균 전송률 r_i를 요구하는 플로에게 Over-provisioning factor μ_j를 갖는 AS 서브클래스에서 $\mu_j r_i$의 대역폭를 할당하여 $(b_i-\mu_j)l_i/\mu_j$의 최대 지연시간을 보장하기 위해서는 DiffServ 라우터가 네트워크 토폴로지와 PS 예약 자원량에 의해 계산된 β값에 따라 접속 제어를 수행해야 한다. 그리고 RIO 및 RIO-DC버퍼관리 방식에서는 In-profile 패킷들과 Out-of-profile 패킷들에게 필요한 버퍼 크기에 따라 max_*in*과 max_*out* 그리고 min_*in*및 min_*out*들을 설정해야 한다. 이때 max_*in*의 값은 In-profile 트래픽의 최대 버퍼 크기에 따른 지연시간을 고려하여 설정하였고 max_*out*의 값은 링크 이용효율을 고려한 β값으로 설정하였다. min_*in*및 min_*out*의 값은 평균 큐 길이를 고려하여 설정하는데 일반적으로 max_*in*과 max_*out*의 1/2 값으로 설정한다 [9]. RIO 및 RIO-DC 방식의 변수 값 설정 방안을 임의의 μ_j를 갖는 AS 서브클래스에 대해 정리하면 표1과 같다.

표 1. 네트워크 토폴로지를 고려한 RIO 및 RIO-DC 변수 설정값

변수	설정값 ($b_{i,\max} < \lambda_{Top}$)	설정값 ($b_{i,\max} \geq \lambda_{Top}$)
max_*in*	$\frac{L_k(b_{i,\max}-\mu_j)\cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size\epsilon}$ $\beta = \frac{\mu_j}{b_{i,\max}-\mu_j}$	$\frac{L_k(\lambda_{top}-\mu_j)\cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size\epsilon}$ $\beta = \frac{\mu_j}{\lambda_{top}-\mu_j}$
max_*out* (RIO)	$\frac{L_k(b_{i,\max}-\mu_j)\cdot l_{i,\max}}{\mu_j \cdot packet\_size}$	$\frac{L_k(\lambda_{top}-\mu_j)\cdot l_{i,\max}}{\mu_j \cdot packet\_size}$
max_*out* (RIO-DC)	$\frac{\beta \cdot L_k(b_{i,\max}-\mu_j)\cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size}$	$\frac{\beta \cdot L_k(\lambda_{top}-\mu_j)\cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size\epsilon}$
min_*in* min_*out*	min_*in* = max_*in*/2 min*out*=max*out*2	min_*in* = max_*in*/2 min*out*=max*out*2

Ⅲ. 시뮬레이션 모델 및 성능 평가

본 논문에서는 DiffServ 방식의 AS에 대한 서비스 제공 방안으로서 서브 클래스별로 큐를 따로 사용하고 할당하는 대역폭의 양을 차별화하여 서브 클래스별로 보장하는 최대 지연시간을 상대적으로 차등화 하였다. 그리고 In-profile 트래픽에 대한 수율과 링크 이용률을 극대화 시킬 수 있도록 In-profile 트래픽에 할당되는 출력링크 대역폭의 크기가 $L_k/(\mu_j(1+\beta))$가 되도록 접속 제어가 수행된 상황 하에서 네트워크 토폴로지 및 AS의 서브 클래스별로 할당되는 대역폭에 의해 결정되는 버퍼 공간의 크기에 따라 표1과 같이 RIO 및 RIO-DC 변수 값을 설정하는 방안을 제시하였다. 본 논문에서는 μ_j가 1인 하나의 AS 서브 클래스만을 고려하여 시뮬레이션을 실시하였다. 한편 TCP 전송 프로토콜을 사용하는 네트워크에서 TCP플로에게 Minimum Rate의 QoS를 보장하기 위해서는 Traffic Conditioner에서의 Token Loss 문제를 해결해야 한다[13]. Token Loss는 Traffic Conditioner내에 남아 있는 토큰이 있음에도 불구하고 전송한 패킷에 대한 Ack가 도착하지 않아 토큰 버킷 내의 토큰이 손실되는 현상으로 이로 인해 송신 호스트로부터

사용자가 계약한 평균 전송률만큼 In-profile 트래픽이 발생하지 못하게 된다. 따라서 TCP 플로에게 Minimum Rate의 QoS를 보장하기 위해서는 Token Loss를 가능한 한 억제하여 In-profile 패킷이 사용자가 계약한 평균 전송률에 근접한 수준으로 발생하도록 하고, Out-of-profile 패킷도 최소한 Token Loss로 인한 평균 전송률의 손실을 보상할 수 있는 만큼은 발생하도록 해야 한다. 이를 전제로 네트워크 라우터에서는 경유하는 플로들의 평균 전송률과 최대 전송률 등의 정보를 바탕으로 버퍼 관리 기준을 설정하고 접속 제어를 수행하는 것이다.

각 송신 호스트의 RTT(Round Trip Time)가 서로 다르고 라우팅된 경로에 따라 패킷 폐기의 가능성이 있는 Hop수 또한 다른 상황에서는 경유하는 Hop 수가 큰 호스트는 RIO 및 RIO-DC의 패킷 기준에 의해 폐기될 가능성이 크고, RTT가 큰 호스트는 패킷 폐기의 영향으로 Token Loss가 발생할 가능성이 더 크다. 즉, 플로들의 트래픽 발생량이 불공평하게 된다. 이러한 환경에서 송신 호스트로부터 Token Loss 없이 사용자가 계약한 평균 전송률 이상으로 트래픽이 발생하게 하는 것은 궁극적으로 동일한 평균 전송률을 신고한 여러 호스트들이 있을 경우 Hop 수와 RTT가 큰 호스트의 In-profile 트래픽 발생과 RTT와 Hop 수가 작은 호스트의 In-profile 트래픽 발생량을 동일하게 하는 것으로 이에 관한 연구가 진행 중에 있다[13]. 본 논문에서는 이러한 Token Loss 문제에 대한 해결책을 적용하지 못하였으나 그림4에서와 같이 각 송신 호스트의 RTT 및 경유하는 Hop수와 계약한 평균 전송률 및 최대 전송률의 Traffic Profile을 동일하게 하여, 모든 호스트의 균등한 In-profile 트래픽의 발생 형태 및 플로간 공평성을 유지하고 Token Loss의 발생 가능성을 최소화하였다. 그림4는 네트워크 토폴로지를 고려하여 접속 제어가 수행된 상황 하에서의 RIO 및 RIO-DC 변수 설정 방안에 대한 성능을 분석하기 위한 시뮬레이션 모델을 나타낸다.

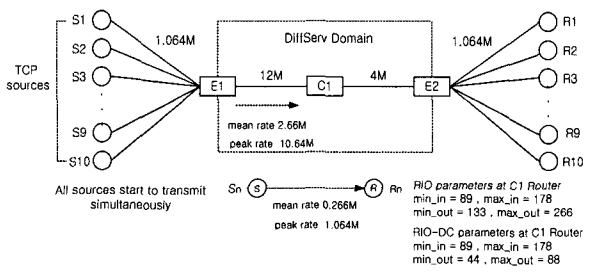


그림 4. RIO-DC 변수 설정 방안에 대한 시뮬레이션 모델

그림4는 표1에서 $b_{i,\max}$값이 $\lambda_{\tau op}$보다 큰 일반적인 경우에 대해 RIO 및 RIO-DC 변수 설정 방안의 성능을 평가하기 위한 시뮬레이션 모델로서 $\lambda_{\tau op}=3$인 경우이다. 그림4에서 E1과 E2 라우터는 DiffServ 입구 라우터를 나타내고 C1은 DiffServ 내부 라우터를 나타낸다. 각 송신 호스트 S_n은 같은 수로 표시된 수신 호스트 R_n에게 평균 0.266Mbps, 최대 1.064Mbps의 데이터를 전송하고 $l_{i,\max}$는 33ms으로 가정하여 식(1)로부터 토큰 버킷의 크기 t_s는 27packets이고 전송되는 *packetsize*는 125 bytes로 설정하였다. 결과적으로 10개의 호스트들로부터 평균 2.66Mbps, 최대 10.64Mbps의 트래픽이 12Mbps 링크를 통과하게 된다. 병목 링크 대역폭은 C1라우터의 출력링크 L_k이고 4Mbps로 설정하였고 C1라우터의 실제 버퍼 크기는 350 packets로 설정하였다. 따라서 그림4는 4Mbps 대역폭의 병목 링크에 2.66Mbps의 평균 전송률의 트래픽이 통과하므로 본 논문의 표1로부터 1/2로 구해진 β값에 따라 고려하는 AS에 대해 접속 제어가 수행된 상황을 나타낸다. 그림4의 네트워크 환경에서 C1 내부 라우터에 RIO 및 RIO-DC 변수 설정 방안을 적용하면 $b_{i,\max}$는 송신 호스트의 최대 전송률과 평균 전송률로부터 4로 산출되어 보장하는 최대 지연시간은 식(1)로부터 0.1sec가 되고 요구되는 전체 보호 버퍼 공간 크기는 표1로부터 266 packets가 된다. 그리고 1/2의 β값에 따라 In-profile 패킷들은 178개의 버퍼 공간과 2.66Mbps의 대역폭을, Out-of-profile 패킷은 88개의 버퍼공간과 1.34Mbps의 대역폭을 이용하도록 RIO의 $\max_in$과 $\max_out$을 각각 178과 266으로 설정하였고, RIO-DC의 max_out은 88로 설정하였다[12][15]. 한편 E1과 E2라우터에서도 RIO 방식을 사용하나 입력 링크 대역폭이 출력 링크 대역폭보다 작아 버퍼에 패킷이 남아 있지 않으므로 설정된 RIO 변수 값에 따른 영향은 고려하지 않아도 된다.

각 송신 호스트는 TCP Reno를 사용하고 RTT는 16ms로 설정하였다. RIO 및 RIO-DC 방식에서 사용되는 w_q의 값은 In-profile 트래픽과 Out-of-profile 트래픽에 대해 공통적으로 0.002를 사용하였고 그림2의 $P_{\max,in}$과 $P_{\max,out}$의 값은 각각 0.02와 0.05를 사용하였다[8][9]. 표2는 그림4에서 시뮬레이션 시 설정된 각 변수 값을 나타낸 것이다.

표 2. 그림4에서의 시뮬레이션 변수 설정 값

변수	$l_{i,max}$	buffer size	max_*in*	max_*out*	t_s	β
RIO	33ms	350packets	100~266	190~350	27packets	1/2
RIO-DC	33ms	350packets	100~266	12~172	27packets	1/2

그림 5는 그림4의 C1라우터에서 RIO 및 RIO-DC 변수 max_*in*과 max_*out*의 각 설정 값에 따라서 10개의 송신 호스트들로부터 100초간 도착한 전체 In-profile 패킷의 수와 통과된 패킷의 수가 어떻게 변화하는 지를 나타낸 것이다. 그림5의 결과를 보면 제안한 방식으로 설정한 RIO 변수 값 max_*in*=178과 max_*out*=266에서 도착하는 전체 In-profile 트래픽의 양이 약 2.55Mbps로서 도착한 In-profile 트래픽 양이 최대값에 매우 근접함을 확인할 수 있다. 한편 도착한 In-profile 트래픽의 양이 통과된 트래픽의 양과 차이가 있으므로 In-profile 패킷에 대한 폐기가 발생하고 있음을 알 수 있고, 제안한 RIO변수 값에서 각 호스트별로 평균 0.255Mbps의 In-profile 트래픽이 발생하므로 평균 0.011Mbps(4.3%)의 최소 Token Loss가 발생하는 것을 알 수 있다. RIO-DC 방식에 대해서는 제안한 RIO-DC 변수 값 max_*in*=178과 max_*out*=88에서 도착하는 전체 In-profile 트래픽의 양이 제안한 RIO 변수 값에서 RIO 방식의 도착량과 동등함을 알 수 있다. 그림6은 C1라우터에서 각 RIO 및 RIO-DC 변수 설정 값에 따라 통과된 전체 패킷 수 즉, 링크 이용률을 나타낸다. 그림6으로부터 제안한 방식으로 설정한 RIO 변수 값에서 약 3.995Mbps의 트래픽이 통과하여 99.875%의 링크 이용률을 나타내며, 최대값에 매우 근접함을 확인할 수 있다. 제안한 RIO-DC 변수 값도 링크 이용률이 가장 높은 변수 값들의 범위에 속하고 약 3.968 Mbps의 트래픽이 통과하여 99.2%의 링크 이용률로서 제안한 변수값을 적용한 RIO 방식의 성능과 거의 동등함을 알 수 있다. 그림5와 그림6으로부

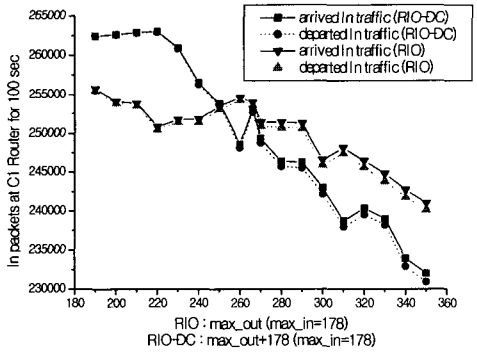


그림 5-(a)

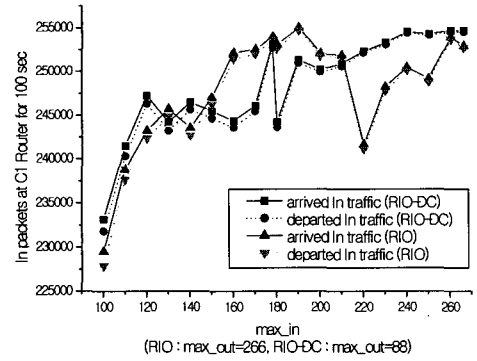


그림 5-(b)

그림 5. C1라우터에서 도착한 In패킷수 및 통과된 In패킷 수

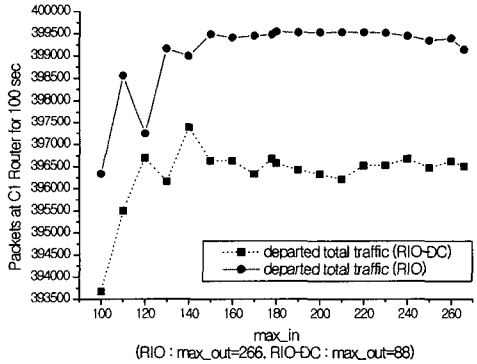


그림 6-(a)

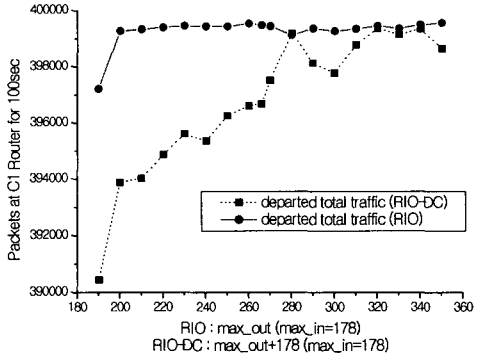


그림 6-(b)

그림 6. C1라우터에서 통과된 전체 패킷 수

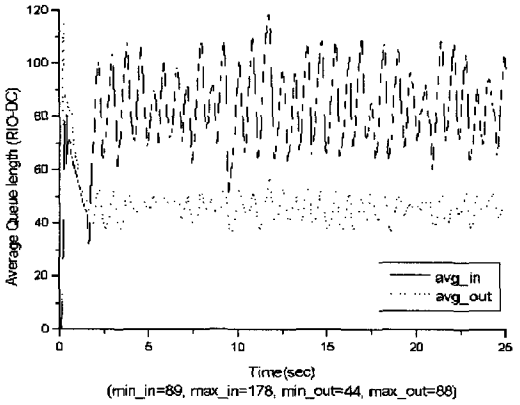


그림 7. C1라우터에서 RIO-DC 방식의 평균 큐 길이 변화

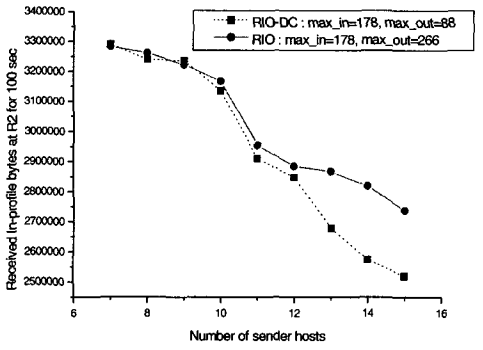


그림 8. 송신 호스트 수에 따른 R2호스트의 수신 In-byte 수

터 접속제어가 수행된 상황 하에서 제안한 RIO 및 RIO-DC 변수 값 설정 방안을 적용하여 두 방식이 동등하게 호스트들로부터 도착하는 In-profile 트래픽의 총양과 전체 링크 이용률을 극대화 시킬 수 있고, 따라서 플로가 수신하는 In-profile 트래픽 수율도 극대화 시킬 수 있음이 예측된다.

그림7는 그림4의 C1라우터에서 제안한 RIO 및 RIO-DC 변수 설정 값을 적용한 경우, 두 방식에 대해 10개의 송신 호스트들로부터 도착하는 패킷들로 인한 평균 큐길이의 변화를 나타낸다. 그림7로부터 TCP의 버스트한 전송 특성과 모든 호스트의 트래픽 발생이 균일하여 평균 큐 길이의 변화가 균일함을 알 수 있다. 그리고 RIO 방식과 RIO-DC 방식의 *avgIn*의 변화가 거의 동일하므로 In-profile 트래픽 수율에 대한 성능에 있어서 동등함을 알 수 있고, 두 방식의 *avgIn*과 *avg−total* *avg−out*값들이 min_*in* 및 min_*out*의 값 주위로 유지되어 링크 이용률 성능도 동등함을 알 수 있으며, In-profile 패킷과 Out-of-profile 패킷에 대해 min_*in* 및 min_*out*에 의한 확률적인 Early Random Drop이 발생함을 알 수 있다. 그림8은 제안한 방식으로 설정한 RIO 및 RIO-DC 변수 값을 적용한 두 방식에서 접속을 허용한 각 호스트 수에 따라 R2호스트가 100초간 수신한 In-profile byte 수를 나타낸다. 최대 지연 시간은 0.1초로서 그림4의 0.266Mbps를 전송하는 호스트에 대해 In-profile 트래픽에 대한 최대 수신 byte 수는 3325000bytes(0.266Mbps)이다. 그림8의 결과로부터 두 방식 모두 송신 호스트 수가 적더라도 계약한 평균 전송률만큼의 수신 byte 수를 나타내지 못하므로 Token Loss가 발생하고 있음을 알 수 있고, 그 원인은 그림7에서 보인 것과 같이 *avgIn*과 *avg−total* *avg−out*값의 증가에 따른 확률적인 Early Random Drop에 의한 TCP의 전송률 감소일 것으로 예측된다. 그림8의 결과를 보면 제안된 변수 값을 적용한 두 방식의 성능이 10개 호스트 수 이하에서 동등함을 알 수 있고, 목표 수용 호스트 수 10개까지 0.254Mbps이상의 In-profile 트래픽 수율(0.011Mbps(4.3%)의 Token Loss)이 유지됨을 알 수 있다. 또한, 두 방식 모두 수용 용량으로 고려한 호스트 개수인 10개를 초과하면 In-profile 트래픽의 수율의 감소가 크고 수용 용량을 초과한 경우에는 RIO 방식이 RIO-DC 방식보다 In-profile 트래픽의 수율이 높다는 것을 알 수 있다. 따라서 두 방식 모두 계산된 β값에 따라 앞에서 제시한 평균 전송률 0.266 Mbps를 전송하는 10개의 호스트의 경우가 최대 수용 용량임을 알 수 있다.

그림 9는 *avg−total*에 의한 Out-of-profile 패킷의 제어의 오류로 발생하는 RIO 방식의 문제점을 나타내는 시뮬레이션 모델로서 그림4와의 차이점은 각 송신 호스트가 10초 간격으로 데이터 전송을 시작하는 것이다. RIO 방식은 *avg−total*에 의해 Out-of-profile 패킷의 폐기를 결정하므로 이전에 전송을 시작한 송신 호스트로부터의 Out-of-profile 패킷들이 내부 라우터 C1의 버퍼를 계산된 β값의 비율 보다 크게 차지할 경우, 뒤에 전송을 시작한 호스트의 In-profile 패킷들이 지연시간을 겪거나 극단적인 경우 폐기될 수 있는 문제점을 가지고 있어 QoS의 저하 및 플로 간에 불공평성이 발생하게 된다. 그러나 RIO-DC 방식은 RIO 방식의 *avg−total* 보다 작은 평균 Out-of-profile 패킷 수, *avg−out*에 따라 직접적으로 Out-of-profile 패킷의 폐기를 결정하므로 위의 RIO 방식의 문제점을 해결할 수 있다.

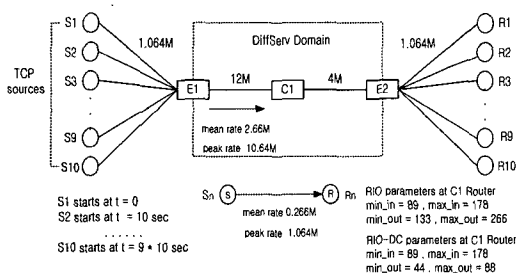


그림 9. AS 플로간의 공평성 보장에 대한 시뮬레이션 모델

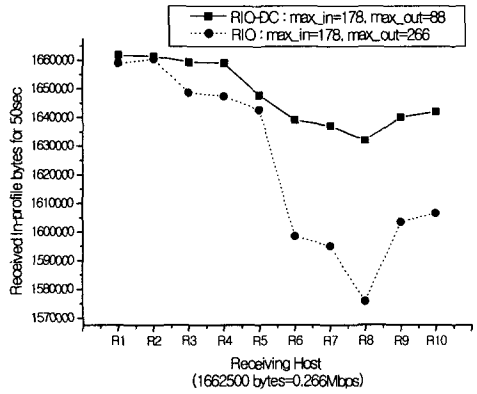


그림 10. 그림 9에서 각 호스트의 50초간 수신 In-byte 수

그림10은 RIO 방식과 RIO-DC 방식을 적용한 경우 각 호스트 R1에서 R10이 전송을 시작한 이후로 50초간 수신한 In-profile byte의 수를 나타낸 결과이다. 그림10의 결과로부터 RIO 방식은 그림9의 환경에서 전송 시작 시간의 차이에 따른 플로간 In-profile 트래픽 수율의 공평성을 유지하지 못하나, RIO-DC 방식은 RIO 방식보다 향상된 공평성을 유지할 수 있어 In-profile 트래픽에 대한 보호 능력이 RIO 방식보다 우수하다고 할 수 있다.

Ⅳ. 결 론

본 논문은 IETF에서 DiffServ의 AS를 위하여 제안된 AF PHB의 버퍼 관리 방식들 중 RIO 및 RIO-DC 방식의 성능을 비교 평가하였다. 이를 위하여 AS의 서브 클래스별로 할당하는 대역폭의 비율을 차별화하여 서브 클래스별로 보장하는 최대 지연시간을 상대적으로 차등화 하였다. 또한, 네트워크 토폴로지와 AS의 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 RIO 및 RIO-DC 방식의 변수 값을 설정하였다. 이러한 환경에서, In-profile 트래픽에 대한 수율, 링크 이용률 및 플로간 공평성을 성능의 척도로 하여 제안된 두 방식의 성능을 비교하였다. 시뮬레이션 결과는 제안하는 변수 설정 방안을 적용한 RIO 및 RIO-DC 방식의 AS에 대한 In-profile 트래픽의 수율과 링크 이용률 성능은 동등하나, RIO-DC 방식은 RIO 방식 보다 향상된 플로간 공평성을 보장할 수 있음을 예시하였다. 결과적으로 AS에 대한 접속 제어가 수행된 상황 하에서 제안한 변수 설정 방안을 적용한 RIO-DC 방식이 RIO 방식보다 In-profile 트래픽에 대한 보호 능력이 우수하다고 할 수 있다.

참 고 문 헌

[1] M.Carlson, et. al., An Architecture for Differentiated Services, RFC 2475, Dec. 1998.

[2] Xipeng Xiao and Lionel M. Ni, Internet QoS: A Big Picture, *IEEE Network*, pp. 1234-1250, March/April 1999.

[3] V. Jacobson, K. Nichols, and K. Poduri, An Expedited Forwarding PHB, RFC2598, June, 1999.

[4] J. Heinanen, F. Baker, and et. al., Assured Forwarding PHB Group, RFC 2597, June, 1999.

[5] Dovrolis, Parameswaran Ramanathan, A Case for Relative Differentiated Services and the Proportional Differentiation Model, *IEEE Network*, September/October 1999.

[6] S. Shenker, C. Partridge, and R. Guerin, Specification of Guaranteed Quality of Service, RFC 2212, September 1996.

[7] R. Braden and et. al., Resource ReSerVation Protocol (RSVP): Version 1: Functional Specification, IETF RFC 2205, September 1997.

[8] Makkar, R and et. al., Empirical study of buffer management scheme for DiffServ assured forwarding PHB, Proceedings of *Ninth International Conference on Computer Communications and Networks 2000*, pp. 632 637, 2000.

[9] S. Floyd, V. Jacobson, Random Early Detection gateways for Congestion Avoidance,

IEEE/ACM Transactions on Networking, vol. 1, no.4, pp. 397-413, August 1993.

[10] W. Feng, D. D. Kandlur, D. Saha, K. G. Shin, A Self-Configuring RED Gateway, Proceedings of *IEEE INFOCOM*, March 1999.

[11] D. Clark, W. Fang, Explicit Allocation of Best Effort Packet Delivery Service, *IEEE/ACM Transactions on Networking*, vol. 6, no.4, pp. 362-373, August 1998.

[12] John B. Pippas and et. al., A modified RIO algorithm that alleviates the bandwidth skew problem in Internet Differentiated Service, Proceedings of *ICC2000*, pp.1599-1603, June 2000.

[13] Wu-chang Feng and et. al., Understanding and Improving TCP Performance Over Networks with Minimum Rate Guarantee, *IEEE/ACM Transactions on Networking*, vol. 7, no.2, pp. 173-187, April 1999.

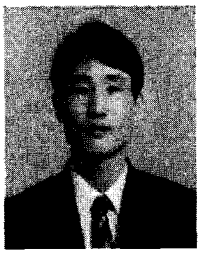
[14] Ilias Andrikopoulos and et. al., A Fair Traffic Conditioner for the Assured Service in a Differentiated Service Internet, Proceedings of *ICC 2000*, pp. 806-810, June 2000.

[15] Benjamin Teitelbaum, Susan Hares and et. al., Internet 2 Qbone: Building a Testbed for Differentiated Services, *IEEE Network*, pp. 645-660, September/October 1999.

허 경(Kyeong Hur) 정회원

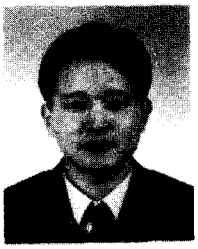
1998년 2월: 고려대학교 전자공학과 학사
2000년 2월: 고려대학교 전자공학과 석사
2000년 3월~현재: 고려대학교 전자공학과 박사과정 재학 중
<주관심 분야> 통신네트워크 설계 및 성능분석, IP 네트워크, 이동 멀티미디어 시스템

신 동 범(Dongbeom Shin) 정회원



1991년 2월: 충남대학교 전자공학교육과 학사
1993년 2월: 충남대학교 전자공학과 석사
1993년 3월~2000년 5월: 국방과학연구소 연구원
2000년 5월 ~ 현재: 한국전자통신연구원 선임연구원
<주관심 분야> IP 네트워크, 통신망 프로토콜

이 상 우(Sangwoo Lee) 정회원



1994년 2월: 광운대학교 전자통신공학과 학사
1996년 2월: 광운대학교 전자통신공학과 석사
1996년 1월~2000년 5월: 대우전자 주임연구원
2000년 5월~현재: 한국전자통신연구원 연구원
<주관심 분야> IP네트워크, metro optical network, ASIC

엄 두 섭(Doo-Seop Eom) 정회원

1987년 2월: 고려대학교 전자공학과 학사
1989년 2월: 고려대학교 전자공학과 석사
1999년 3월: 일본오사카대학 정보통신공학과 박사
1989년 2월~1999년 8월: 한국전자통신연구소 연구원
1999년 9월~2000년 8월: 원광대학교 전임강사
2000년 9월~현재: 고려대학교 전기전자전파공학부

조교수

<주관심 분야> 통신네트워크 설계 및 성능분석,
무선 ATM, IP 네트워크

차 균 현(Kyun Hyon Tchah) 정회원

1965년 2월: 서울대학교 전기공학과 학사
1967년 6월: 미국 일리노이 공과대학 석사
1976년 6월 : 서울대학교 전자공학과 박사
1977년 3월~현재: 고려대학교 전자공학과 교수
1998년 1월~1998년 12월: 한국통신학회 회장
1998년 4월~현재: 한국전자통신연구원 부이사장
<주관심 분야> 통신 이론, 이동 통신, 위성 통신,
이동 멀티미디어 시스템