DiffServ 방식의 Assured Service에서 QoS 보장을 위한 Adaptive RIO 방식의 제안

정회원 허 경*, 김 문 규*, 이 승 현*, 조 성 대**, 엄 두 섭*, 차 균 현*

An Adaptive RIO buffer management scheme for QoS guarantee of Assured Service in Differentiated Services

Kyeong Hur*, Moon Kyu Kim*, Seung Hyun Lee*, Seong-Dae Cho**, Doo-Seop Eom*, Kyun Hyon Tchah* ***Regular Members***

요 약

본 논문은 DiffServ 방식의 Assured Service를 이용하는 플로에게 QoS를 보장하기 위해 Admission Control이 수행된 상황 하에서 발생하는 RIO 방식의 문제점을 제시하고 이에 대한 해결 방안으로 Adaptive RIO 방식을 제안한다. 제안하는 Adaptive RIO 방식은 네트워크 토폴로지와 Assured Service 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 일정 시간 구간마다 도착 가능한 In-profile 패킷과 전체 패킷의 최대량을 고려하여 RIO의 변수 값들을 재설정함으로써 허용된 In-profile 패킷에 대한 확률적인 폐기 (Early Random Drop)를 방지한다. 시뮬레이션 결과는 Assured Service에 대해 Admission Control이 수행된 상황과 Congestion 상황에서 제안하는 Adaptive RIO 방식이 RIO 방식보다 In-profile 트래픽에 대한 보호와 수율 성능이 우수함을 보인다.

ABSTRACT

In this paper, we proposed an Adaptive RIO scheme to solve the problem of RIO scheme that occurs when admission control is performed for QoS guarantee of Assured Service in Differentiated Services. To prevent an early random drop of the admitted In-profile packet, proposed Adaptive RIO scheme updates parameters of RIO scheme every time interval according to the estimated numbers of maximum packet arrivals of In-profile traffic and total traffic during the next time interval. The numbers of maximum packet arrivals during the next time interval are estimated based on the buffer size determined by the network topology and the ratio of bandwidth allocated to each subclass. We found from simulation results that, compared with RIO scheme, proposed Adaptive RIO scheme can improve performance of the throughput for In-profile traffic when admission control is performed or congestion occurs.

I. 서 론

사용자가 요구하는 QoS를 보장할 수 있는 차세대 인터넷에 대한 구조로서 DiffServ 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 네트워크에 적합한 확장성을 갖고 기존의 IntServ (Integrated Services) 방식의 문제점을 해결할 수

* 고려대학교 전자공학과 (hkyeong@korea.ac.kr) ** 한국통신 통신망연구소
논문번호 : 010323-1109, 접수일자: 2001년 11월 9일

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 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. IntServ 방식은 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, RED (Random Early Detection)를 확장한 RIO(RED with In and Out) 방식을 이용하여 AS 트래픽에 대한 혼잡 제어를 실시한다[8]-[10]. RIO 방식은 사용자와 네트워크 간에 약속된 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. 현재 IETF에서는 RFC 2597에서 제안된 AF PHB를 위한 Active Queue Management 방식들을 대상으로 병목 구간의 라우터에서 낮은 패킷 폐기 순위를 갖은 트래픽을 우선적으로 보호할 수 있는 방안이 연구되고 있다[11][12].

한편 DiffServ 방식에서 AS를 사용하는 TCP 플로에게 Minimum Rate의 QoS를 보장하기 위해서는 접속 제어가 필요하고 사용자가 계약한 In-profile 트래픽에 대한 보호 및 수율(Throughput) 보장이 요구된다[13]. 따라서 접속 제어가 수행된 상황 하에서 도착하는 In-profile 패킷에 대한 폐기가 발생하지 않아야 한다. 그러나 RIO 방식은 AS의 QoS 보장에 있어서 평균 큐 내 In-profile 패킷의 수를 나타내는 avg_in을 이용한 In-profile 패킷의 폐기로 인해 발생하는 문제점을 갖고 있다. 즉, RIO 방식의 avg_in은 In-profile 패킷의 도착에 의해 증가하게 되는 값이고, TCP 플로들의 In-profile 패킷들이 버스트하게 도착할 경우 In-profile 패킷이 폐기될 때 까지 avg_in은 지속적으로 증가하게 된다. 이때 임의의 시간 구간 $[t_1, t_1+\tau]$에서 접속 제어를 통해 허용된 플로들로부터 τ 동안 도착하는 In- profile 패킷 도착량에 의해 avg_in이 설정된 min_in을 초과할 경우 In-profile 패킷에 대한 폐기가 발생하게 되어 AS를 사용하는 TCP 플로에게 In-profile 트래픽의 수율을 보장하지 못하게 된다. 이에 대해 본 논문은 DiffServ 방식에서 AS의 QoS를 보장하기 위해 접속 제어가 수행된 상황 하에서 발생하는 RIO 방식의 문제점을 제시하고 이에 대한 해결 방안으로 Adaptive RIO 방식을 제안한다. 제안하는 Adaptive RIO 방식은 네트워크 토폴로지와 AS 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 일정 시간 구간마다 도착 가능한 In-profile 패킷과 전체 패킷의 최대량을 고려하여 RIO의 변수 값들을 재설정함으로써 허용된 In-profile 패킷에 대한 확률적인 폐기를 방지한다. 본 논문의 구성은 다음과 같다. 제2절에서는 Adaptive RIO 방식을 제안한다. 제3절의 시뮬레이션 모델 및 결과는 AS에 대해 접속제어가 수행된 상황과 Congestion 상황에서 제안하는 Adaptive RIO 방식이 RIO 방식보다 In-profile 트래픽에 대한 보호와 수율 성능이 우수함을 보인다. 끝으로 제4절에서 결론을 맺는다.

Ⅱ. AS QoS보장을 위한 Adaptive RIO방식

2.1 RIO 방식

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]. 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 식(1)과 같이 서브 클래스별로 보장하는 최대 지연시간 $d_{\max,j}$를 상대적으로 차등화할 수 있다.

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

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

그림1은 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 출력링크로 하는 이전 DiffServ 라우터에서, AS 클래스가 사용 가능한 대역폭 L_j는 식(2)로 정의된다. 이로부터 식(3)에서 λ_{op}은 네트워크 토폴로지에 따른 라우터의 입력대역폭과 출력대역폭 간의 비율를 나타낸다.

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

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

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

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

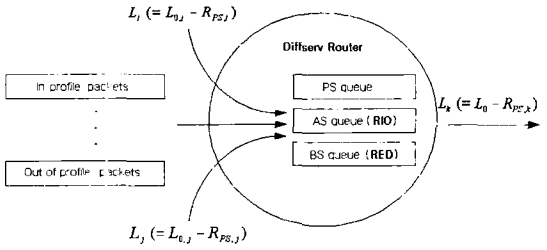


그림 1. 네트워크 토폴로지를 고려한 DiffServ 라우터에서의 AS 트래픽

사용자와 네트워크 간에 약속된 Traffic Profile을 준수하는 In-profile 패킷과 그렇지 못한 Out-of-profile 패킷들을 고려할 때, 라우터의 출력링크 대역폭을 In-profile 패킷들이 대부분 이용할 수 있도록 해야 하기 때문에, DiffServ 라우터의 AS 큐에서 In-profile 패킷과 Out-of-profile 패킷들이 저장되는 버퍼 공간 크기의 평균적인 비율은 $1:\beta(\beta \le 1)$가 되도록 RIO 변수 값들을 설정한다. 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 접속제어를 통해 라우터를 경유하는 플로들로부터 발생되는 In-profile 트래픽의 평균전송률의 합이 $L_k/(\mu_j(1+\beta))$의 대역폭을 초과하지 않도록 해야 한다.

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

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 1과 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. 이때 $b_{i,\max}$값은 신고한 트래픽 정보들 b_i들 중 값이 가장 큰 것을 나타내며 $packetsize$는 플로들이 전송하는 패킷의 평균 길이를 나타낸다. 또한 각 트래픽이 이용하는 출력링크 대역폭의 양 $L_k/(1+\beta)$, $L_k\beta/(1+\beta)$에 따라 τ 동안 AS큐에 최대로 저장되는 패킷양은 In-profile 패킷들에 대해서 $L_k(b_{i,\max}-1)l_{i,\max}/((1+\beta)packetsize)$가 되고 Out-of-profile 패킷들에 대해서는 $\beta L_k(b_{i,\max}-1)l_{i,\max}/((1+\beta)packetsize)$가 된다. 그러나 일반적으로 λ_{op}값은 1보다 크거나 같고 $b_{i,\max}$보다는 작다. 따라서 λ_{op}이 $b_{i,\max}$보다 작은 경우에는 τ동안 도착 가능한 In-profile 패킷들의 최대량은 $L_k\lambda_{op}l_{i,\max}/((1+\beta)packetsize)$가

되고, Out-of-profile 패킷들의 최대량은 $\beta L_k \lambda_{Top} l_{i,\max} / ((1+\beta) packetsize)$가 된다. 그리고 τ 동안 AS큐에 최대로 남게 되는 양은 In-profile 패킷들이 $L_k(\lambda_{Top}-1) l_{i,\max} / ((1+\beta) packetsize)$가 되고 Out-of-profile 패킷들은 $\beta L_k(\lambda_{Top}-1) l_{i,\max} / ((1+\beta) packetsize)$가 된다. 한편 RIO 방식에서 max_*in*과 max_*out*은 In-profile 패킷과 전체 패킷들이 각각 최대 이용 가능한 버퍼 크기로 설정한다[9]. 따라서 max_*in*과 max_*out*은 λ_{Top}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, λ_{Top}이 $b_{i,\max}$보다 작은 경우에는 각각 $L_k(\lambda_{Top}-1) l_{i,\max} / ((1+\beta) packetsize)$와 $L_k(\lambda_{Top}-1) l_{i,\max} / packetsize$로 설정한다.

표 1. 네트워크 토폴로지를 고려한 RIO 변수 설정값

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

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

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

RIO 변수 max_*in*과 max_*out*값의 설정에 있어서 β 값은 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 조건을 고려하여 임의의 μ_j를 갖는 AS 서브클래스에 대해 산출되는 β값을 나타낸 것이다. 그리고 RIO에서 식(5)에서 구한 β와 PS용으로 예약된 자원량에 따라 결정되는 전체 버퍼 크기보다 큰 버퍼 공간을 보호한다면 증가한 Out-of-profile 패킷들로 인해 In-profile 트래픽 수율이 보장되지 않을 수 있다. 결과적으로 DiffServ 방식에서 평균 전송률 r_i를 요구하는 플로에게 Over-provisioning factor μ_j를 갖는 AS 서브클래스에서 $\mu_j r_i$의 대역폭을 할당하여 $(b_i-\mu_j) l_i / \mu_j$의 최대 지연시간을 보장하기 위해서는 DiffServ 라우터가 네트워크 토폴로지와 서브클래스별 할당 대역폭의 비율에 따라 계산된 β에 따라 접속 제어를 수행해야 한다. 그리고 In-profile 패킷들과 Out-of-profile 패킷들에게 필요한 버퍼 크기에 따라 max_*in*과 max_*out*의 값을 설정해야 한다. 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 변수 값 설정 방안을 임의의 μ_j를 갖는 AS 서브클래스에 대해 정리하면 표1과 같다.

2.2 제안하는 Adaptive RIO 방식

AS에 대한 QoS를 보장하기 위해서는 접속 제어가 수행된 상황 하에서 도착하는 In-profile 패킷에 대한 폐기가 발생하지 않아야 한다. 그러나 RIO 방식은 AS의 QoS 보장에 있어서 평균 큐 내 In-profile 패킷의 수를 나타내는 *avg_in*을 이용한 In-profile 패킷의 폐기로 인해 발생하는 문제점을 갖고 있다. RIO 방식의 *avg_in*은 In-profile 패킷의 도착에 의해 증가하게 되는 값이고, TCP 플로들의 In-profile 패킷들이 버스트하게 도착할 경우 In-profile 패킷이 폐기될 때 까지 *avg_in*은 지속적으로 증가하게 된다. 이때 임의의 시간 구간 $[t_1, t_1+\tau]$에서 접속 제어를 통해 허용된 플로들로부터 τ 동안 도착하는 In-profile 패킷 도착량에 의해 *avg_in*이 설정된 min_*in*을 초과할 경우 In-profile 패킷에 대한 폐기가 발생하게 되어 AS를 사용하는 TCP 플로에게 In-profile 트래픽의 수율을 보장하지 못하게 된다. 이에 대해 제안하는 Adaptive RIO 방식은 표1과 같이 네트워크 토폴로지와 AS 서브 클래스별로 할당되는 대역폭의 비율이 따라 결정되는 버퍼 크기를 기준으로 시간 구간 τ 마다 τ 동안 라우터의 AS 큐에 도착 가능한 In-profile 패킷과 전체 패킷의 최대량이 도착했을 때 τ 시간후 변화한 *avg_in*과 *avg_total*의 최대 및 최소값을 고려하여 구간 τ 마다 max_*in*, max_*out*과 min_*in*, min_*out*을 재설정한다.

Adaptive RIO 방식에서 사용되는 *avg_in*과 *avg_total*의 변수는 RIO 방식에서와 같이 평균 큐 길이를 나타내지 않으므로 본 논문에서는 각각

ind_in과 ind_total로 정의한다. 즉, ind_in과 ind_total 변수를 이용하여 τ 동안 RIO 방식으로 In-profile 패킷과 Out-of-profile 패킷에 대해 혼잡 제어를 실시한다. 그리고 Adaptive RIO 방식에서 구간 τ 동안 도착 가능한 In-profile 패킷과 전체 패킷의 최대량 $\lambda_{\max,in}$, $\lambda_{\max,total}$은 표1과 구간 τ 시작 시점에서 큐 내에 저장된 In-profile 패킷과 전체 패킷의 수 $init\_q_{in}$, $init_q$로부터 구해진다. 즉 DiffServ 입구 라우터에서의 토큰 버킷의 특성에 따라 구간 τ 시작 시점에서 큐에 $init_q_{in}$만큼 In-profile 패킷들이 존재한다면 구간 τ 동안 도착 가능한 In-profile 패킷의 최대량은 식(6)와 같이 표1에서 $init_q_{in}$만큼 뺀 값이 된다. 마찬가지로 구간 τ 동안 도착 가능한 전체 패킷의 최대량은 표1에서 $init_q$만큼 뺀 값이 된다.

$$
\begin{aligned}
&for \quad b_{i,\max} < \lambda_{top,k} \\
&\quad \lambda_{\max,in} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j (1+\beta) \cdot packet\_size} - init\_q_{in} \\
&\quad \lambda_{\max,total} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} - init\_q \quad , \\
&for \quad b_{i,\max} \geq \lambda_{top,k} \\
&\quad if \ \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j (1+\beta) \cdot packet\_size} \geq \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j (1+\beta) \cdot packet\_size} - init\_q_{in} \\
&\quad\quad \lambda_{\max,in} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j (1+\beta) \cdot packet\_size} - init\_q_{in} \\
&\quad else \\
&\quad\quad \lambda_{\max,in} = \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j (1+\beta) \cdot packet\_size} \\
&\quad if \ \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} \geq \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} - init\_q \\
&\quad\quad \lambda_{\max,total} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} - init\_q \\
&\quad else \\
&\quad\quad \lambda_{\max,total} = \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j \cdot packet\_size}
\end{aligned}
\qquad (6)
$$

또한, 식(6)과 같이 τ 동안 라우터의 AS 큐에 도착 가능한 In-profile 패킷과 전체 패킷의 최대량이 도착했을 경우를 가정하여 식(7), (8), (9)과 같이 초기화된 ind_in과 ind_total의 τ 시간 후 최대 및 최소 변화값 $\max ind_in$과 $\min ind_in$및 $\max ind_total$과 $\min ind_total$을 구할 수 있다. 식(7), (8), (9)는 RIO 방식에 기초하여 τ 동안 도착 가능한 In-profile 패킷의 최대량 $\lambda_{\max,in}$이 도착했을 때 M_k를 기준으로 한 $init_q_{in}$의 각 경우에서 $\max ind_in$과 $\min ind_in$을 구하는 알고리즘을 나타낸 것이다. 그리고 τ 동안 출력링크로 전송되는 전체 패킷들의 수 $L_k \cdot \tau / packet_size$를 기준으로 한 $init_q$의 각 경우에서 동일한 방식으로 τ 동안 도착 가능한 전체 패킷의 최대량 $\lambda_{\max,total}$이 도착했을 때 $\max ind_total$과 $\min ind_total$을 구할 수 있다. 식(7), (8), (9)에서 τ 동안 라우터의 출력링크로 전송되는 In-profile 패킷들의 수는 $M_k (= L_k \cdot \tau / (1+\beta) \cdot packet_size)$로 나타내었다. 그리고 τ 구간 중간에 $\lambda_{\max,in}$의 In-profile 패킷들이 모두 도착한 때에 AS 큐 내에 남아있는 In-profile 패킷의 수를 $q_{in,\max}$로 나타내었으며, 큐 내 In-profile 패킷들의 수는 q_{in}으로 나타내었다. 또한 $\lambda_{\max,in}$의 도착량에 대해 $\lambda_{top,k}$의 배수로 도착한 회수를 n_{top}로 나타내었고 $\lambda_{top,k}$보다 같거나 적게 도착한 패킷수를 $beta$로 표시하였다. 따라서 n_{top}는 입력링크 대역폭과 출력링크 대역폭 간의 비율 $\lambda_{top,k}$에 따라 $\lambda_{\max,in}$의 도착량에 대해 라우터의 출력링크로 전송된 In-profile패킷수를 나타낸다.

$$
\begin{aligned}
&Case\,1 \quad init\_q_{in} = 0 \\
&\quad when\ j \cdot \lambda_{top,k}\ packets\ arrive \\
&\quad\quad q_{in} = q_{in} + 1 \\
&\quad when\ packet\ arrives \\
&\quad\quad \max ind\_in = \max ind\_in \cdot (1 - w_q) + w_q \cdot q_{in} \\
&\quad\quad q_{in} = q_{in} + 1 \\
&\quad if\ \lambda_{\max,in} \leq M_k \\
&\quad\quad \min ind\_in = init\_ind\_in \cdot (1 - w_q)^{\lambda_{\max,in}} \\
&\quad if\ \lambda_{\max,in} > M_k \\
&\quad\quad \lambda_{\max,in} = (M_k - n_{top}) + beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad for\ i = 1\ to\ beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad\quad q_{in} = q_{in} + 1 \\
&\quad\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_{in} \\
&\quad\quad\quad\quad q_{in} = q_{in} + 1
\end{aligned}
\qquad (7)
$$

$$
\begin{aligned}
&Case\,2 \quad 0 < init\_q_{in} < M_k \\
&\quad when\ j \cdot \lambda_{top,k}\ packets\ arrive \\
&\quad\quad q_{in} = q_{in} + 1 \\
&\quad when\ packet\ arrives \\
&\quad\quad \max ind\_in = \max ind\_in \cdot (1 - w_q) + w_q \cdot q_{in} \\
&\quad\quad q_{in} = q_{in} + 1 \\
&\quad if\ \lambda_{\max,in} \leq M_k - init\_q_{in} \\
&\quad\quad \min ind\_in = init\_ind\_in \cdot (1 - w_q)^{\lambda_{\max,in}} \\
&\quad if\ \lambda_{\max,in} > M_k - init\_q_{in}\ \&\ \lambda_{\max,in} \leq (M_k - init\_q_{in}) \cdot \lambda_{top,k} \\
&\quad\quad \lambda_{\max,in} = (M_k - init\_q_{in} - n_{top}) + beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad for\ i = 1\ to\ beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_{in} \\
&\quad\quad\quad\quad q_{in} = q_{in} + 1
\end{aligned}
$$

$$
\begin{aligned}
&if\ \lambda_{max\_in} > (M_k - init\_q_{in}) \cdot \lambda_{top,k} \\
&\quad \lambda_{max\_in} == (M_k - init\_q_{in}) \cdot \lambda_{top,k} + beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad q_{in} = n_{top} \\
&\quad for\ i = 1\ to\ \lambda_{max\_in} \\
&\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_{in} \\
&\quad\quad\quad q_{in} = q_{in} + 1
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
&case\ 3 \quad init\_q_{in} \geq M_k \\
&\quad when\ j \cdot \lambda_{top,k}\ packets\ arrive \\
&\quad\quad q_{in} = q_{in} - 1 \\
&\quad when\ packet\ arrives \\
&\quad\quad \max ind\_in = \max ind\_in \cdot (1 - w_q) + w_q \cdot q_{in} \\
&\quad\quad q_{in} = q_{in} + 1 \\
&if\ \lambda_{max\_in} == beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad q_{in} = init\_q_{in} - (M_k - n_{top}) \\
&\quad for\ i = 1\ to\ \lambda_{max\_in} \\
&\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_{in} \\
&\quad\quad\quad q_{in} = q_{in} + 1
\end{aligned} \tag{9}
$$

식(7), (8), (9)로부터 구간 τ 마다 설정해야 하는 ind_in과 ind_total의 초기값 $init\_ind\_in$, $init_ind_total$들은 0보다 크게 설정되어야 하고 가중치 w_q보다는 작게 설정되어야 ind_in과 ind_total의 증감을 명확하게 나타낼 수 있음을 알 수 있다. 그리고 제안하는 Adaptive RIO 방식에서 구간 τ 마다 재설정하는 $\max\_in$, $\max_out$과 $\min\_in$, $\min_out$의 값은 구간 τ 마다 계산되는 $\max ind_in$과 $\min ind_in$및 $\max ind_total$과 $\min ind_total$들로부터 식(10)과 같이 도출된다. 표2는 2.1절의 RIO 방식과 제안하는 Adaptive RIO 방식을 정리하여 비교한 것이다.

$$
\begin{aligned}
&\max\_in = \rho_{max\_in} \cdot \max ind\_in + (1 - \rho_{max\_in}) \cdot \min ind\_in \\
&\min\_in = \rho_{min\_in} \cdot \max ind\_in + (1 - \rho_{min\_in}) \cdot \min ind\_in \\
&\max\_out = \rho_{max\_out} \cdot \max ind\_total + (1 - \rho_{max\_out}) \cdot \min ind\_total \\
&\min\_out = \rho_{min\_out} \cdot \max ind\_total + (1 - \rho_{min\_out}) \cdot \min ind\_total
\end{aligned} \tag{10}
$$

표 2. 제안하는 Adaptive RIO 방식과 RIO 방식의 비교

RIO방식 ($b_{i,\max} \geq \lambda_{\tau op}$)	Adaptive RIO 방식
avg_in	ind_in
avg_total	ind_total
$\min\_in = \max\_in / 2$	$\rho_{min\_in} \cdot \max ind\_in + (1 - \rho_{min\_in}) \cdot \min ind\_in$
$\min\_out = \max\_out / 2$	$\rho_{max\_out} \cdot \max ind\_total + (1 - \rho_{max\_out}) \cdot \min ind\_total$
$\max\_in = \frac{L_k(\lambda_{top} - \mu_j) \cdot l_{i,\max}}{\mu_j(1+\beta) \cdot packet\_size}$	$\rho_{max\_in} \cdot \max ind\_in + (1 - \rho_{max\_in}) \cdot \min ind\_in$
$\max\_out = \frac{L_k(\lambda_{top} - \mu_j) \cdot l_{i,\max}}{\mu_j \cdot packet\_size}$	$\rho_{min\_out} \cdot \max ind\_total + (1 - \rho_{min\_out}) \cdot \min ind\_total$

Ⅲ. 시뮬레이션 모델 및 성능 평가

본 논문에서는 DiffServ 방식의 AS에 대한 QoS를 보장하기 위해 접속 제어가 수행된 상황 하에서 In-profile 패킷에 대한 폐기를 방지할 수 있는 Adaptive RIO 방식을 제안하였다. 그리고 네트워크 토폴로지와 AS 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 일정 시간 구간마다 도착 가능한 In-profile 패킷과 전체 패킷의 최대량을 고려하여 RIO의 변수 값들을 재설정하는 Adaptive RIO 방식을 표1과 표2에 제시하였다. 본 논문에서는 μ_j가 1인 하나의 AS 서브 클래스만을 고려하여 RIO 방식과 제안하는 Adaptive RIO 방식에 대한 성능을 비교 분석하였다. 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. 특히, 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 RIO의 avg_in과 avg_total에 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 그림2에서와 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 발

생 가능성을 최소화하였다. 그림2는 네트워크 토폴로지를 고려하여 접속 제어가 수행된 상황 하에서 RIO 방식과 Adaptive RIO 방식의 성능을 분석하기 위한 시뮬레이션 모델을 나타낸다.

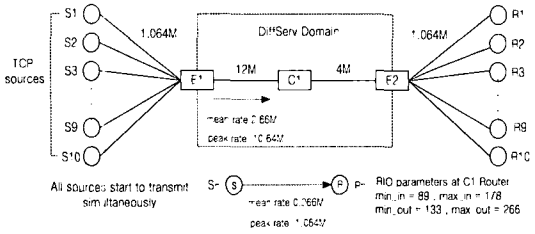


그림 2. Adaptive RIO 방식에 대한 시뮬레이션 모델

그림2는 표1에서 $b_{i,\max}$값이 λ_{Top}보다 큰 일반적인 경우에 대한 시뮬레이션 모델로서 $\lambda_{Top}=3$인 경우이다. 그림2에서 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 링크는 C1라우터의 출력링크 L_k이고 대역폭은 4Mbps로 설정하였고 C1라우터의 실제 버퍼 크기는 350 packets로 설정하였다. 따라서 그림2는 4Mbps 대역폭의 병목 링크에 2.66Mbps의 평균 전송률의 트래픽이 통과하므로 표1로부터 1/2로 구해진 β값에 따라 고려하는 AS에 대해 접속 제어가 수행된 상황을 나타낸다. 그림2의 시뮬레이션 모델에서 RIO 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 버퍼공간 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 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. 또한 RIO 방식에서 사용되는 w_q의 값은 In-profile 트래픽과 Out-of-profile 트래픽에 대해 공통적으로 0.002로 설정하였고 $P_{\max,in}$과 $P_{\max,out}$의 값은 각각 0.02와 0.05를 사용하였다[8][9]. 표3은 그림2에서 설정된 각 변수 값을 나타낸 것이다.

표 3. 그림2에서의 시뮬레이션 변수 설정 값

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

그림3은 그림2의 C1라우터에서 RIO 변수 max_*in*과 max_*out*의 각 설정 값에 따라서 10개의 송신 호스트들로부터 100초간 도착한 In-profile 패킷의 수와 통과된 패킷의 수가 어떻게 변화하는 지를 나타낸 것이다. 그림3의 결과를 보면 제안한 방식으로 설정한 변수 값 max_*in* = 178 과 max_*out* = 266 에서 도착하는 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 변수 값에서 각 호스트별로 평균 0.255Mbps의 In-profile 트래픽이 발생하므로 평균 0.011Mbps(4.3%)의 최소 Token Loss가 발생하는 것을 알 수 있다. 그림4는 C1라우터에서 각 RIO 변수 설정 값에 따라 통과된 전체 패킷 수 즉, 링크 이용률을 나타낸다. 그림4로부터 제안한 방식으로 설정한 RIO 변수 값에서 약 3.995Mbps의 트래픽이 통과하여 99.875%의 링크 이용률을 나타내며, 최대 값에 매우 근접함을 확인할 수 있다.

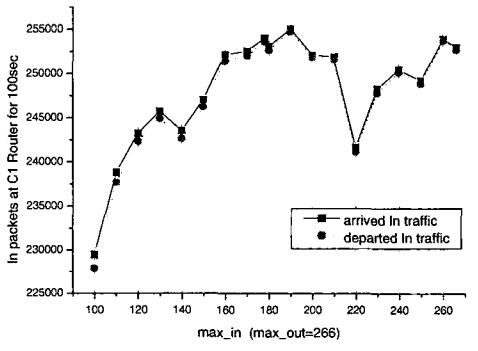


그림 3-(a)

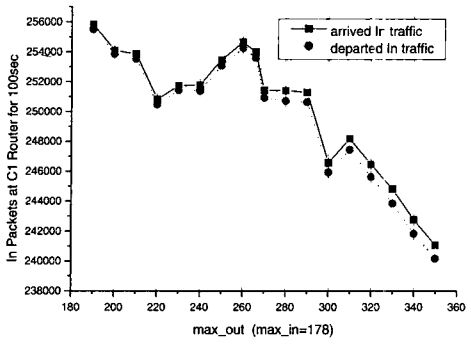


그림 3-(b)

그림 3. C1라우터에서 도착한 In패킷수 및 통과된 In패킷 수

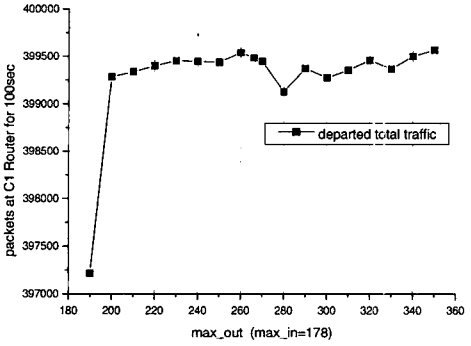


그림 4-(a)

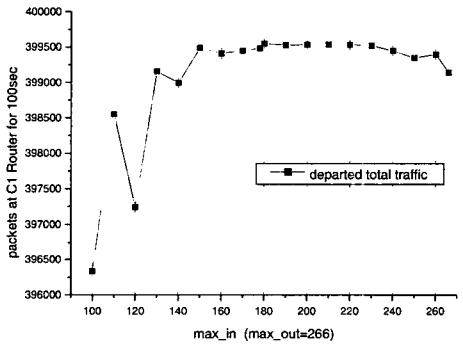


그림 4-(b)

그림 4. C1라우터에서 통과된 전체 패킷 수

그림5는 그림2의 C1라우터에서 RIO 변수 설정 값을 적용한 경우, 10개의 송신 호스트들로부터 도착하는 패킷들로 인한 평균 큐길이의 변화를 나타낸다. 그림5로부터 모든 호스트의 트래픽 발생이 균일하여 평균 큐 길이의 변화가 균일함을 알 수 있고, TCP의 버스트한 전송 특성으로 avg_in은 급격하게 증가하나 패킷 폐기가 발생하더라도 크게 감

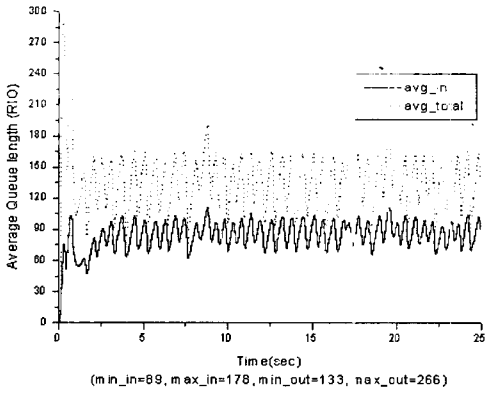


그림 5. C1 라우터에서 RIO 평균 큐 길이의 변화

소하지 않음을 알 수 있다. 따라서 avg_in과 avg_total 값은 $\min_in$과 $\min\_out$ 값 주위로 유지되어 $\min_in$과 $\min_out$에 의해 확률적인 In-profile 패킷과 Out-of-profile 패킷의 Early Random Drop이 지속적으로 발생함을 알 수 있다. 그림6은 그림2의 C1 내부 라우터에 Adaptive RIO 방식을 적용할 경우 매 구간 τ에서의 각 $init_q_{in}$과 $init\_q$ 값에 따라 식(6)에서 허용된 In-profile 트래픽과 전체 트래픽의 최대 도착량을 고려하여 식(7), (8), (9)로 계산되는 $\max ind_in$과 $\min ind\_in$ 및 $\max ind_total$과 $\min ind\_total$ 값을 나타낸 것이다. 그리고 그림7은 C1 라우터에 Adaptive RIO 방식을 적용했을 때 시간에 따라 구간 τ 마다 설정된 $\min\_in$, $\min\_out$ 그리고 ind_in과 ind_total의 변화를 나타낸 것으로 매 구간마다 설정되는 $init\_ind\_in$, $init_ind_total$은 0.0015로 설정하였다. 그리고 ρ_{max_in}은 0.3, ρ_{min_in}은 0.15로 설정하였고, ρ_{max_out}은 0.9, ρ_{min_out}은 0.5로 설정하였다. 그림7로부터 제안한 Adaptive RIO 방식을 통해 $\min_in$은 매 구간마다 도착하는 In-profile 패킷양에 따라 적응적으로 설정되어 ind_in이 설정된 $\min_in$을 초과하지 않아 In-profile 패킷에 대한 확률적인 Early Random Drop이 발생하지 않음을 알 수 있다. 또한 $\min_out$은 In-profile 트래픽과 Out-of-profile 트래픽이 저장되는 버퍼공간크기의 평균적인 비율, $1:\beta$를 고려하여 매구간 τ 마다 도착하는 전체 트래픽의 최대도착량에 따라 설정되어 구간τ 마다 도착하는 전체 트래픽량의 증가로 ind_total이 지속적으로 큰 변화를 나타낼 경우 $\min_out$이 감소하여 Out-of-profile패킷에 대한 확률적인 Early Random Drop이 발생함을 알수 있다.

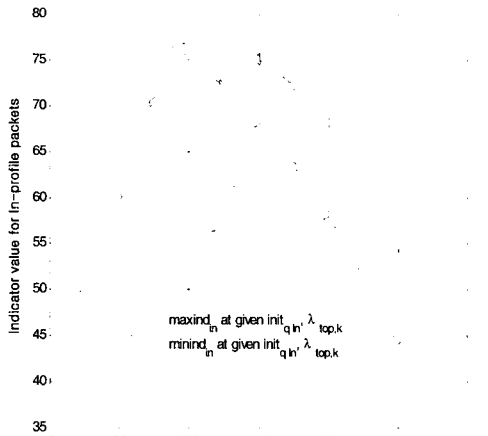


그림 6-(a)

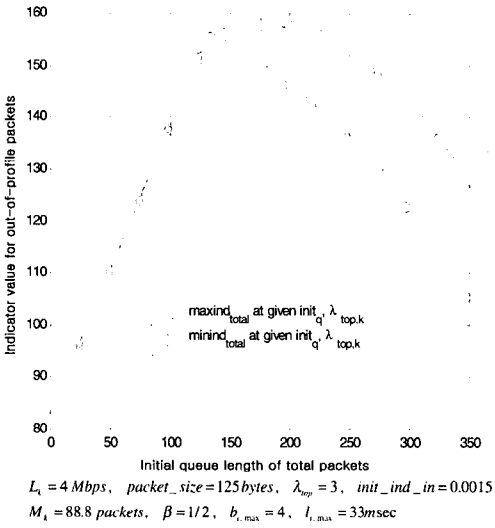


그림 6-(b)

그림 6. Adaptive RIO 방식에서 각 $init_q_{in}$과 $init_q$에 따른 $\max ind_in$과 $\min ind\_in$ 및 $\max ind_total$과 $\min ind\_total$

표4는 그림2의 시뮬레이션 모델에서 제안하는 Adaptive RIO 방식과 RIO 방식을 적용했을 때 접속을 허용한 송신 호스트의 수에 따라 C1라우터에서 100초간 도착한 In-profile 패킷들의 수와 통과된 In-profile 패킷들의 수를 나타낸 것이다. 제안하는 Adaptive RIO 방식은 그림7로부터 In-profile 패킷에 대한 Early Random Drop이 발생하지 않으므로 C1라우터에서 폐기된 In-profile 패킷들은 모두

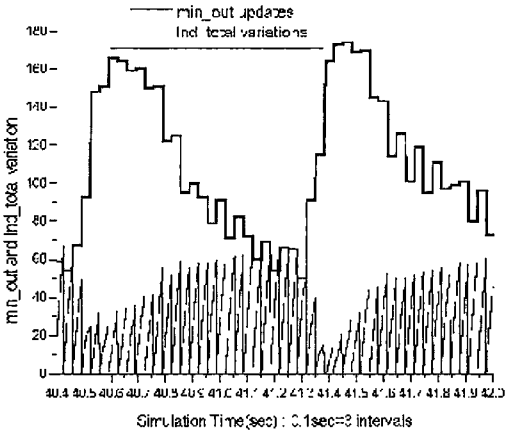


그림 7-(a)

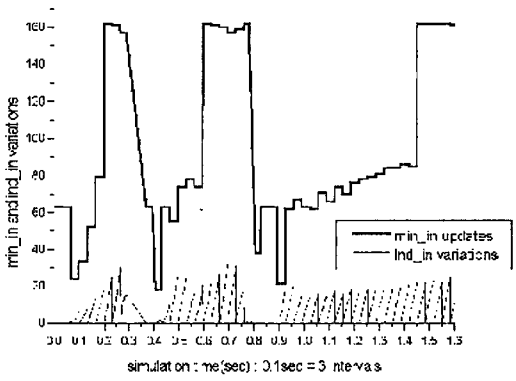


그림 7-(b)

그림 7. min_*in*, min_*out* 및 *ind*_*in*과 *ind*_*total*의 변화

Buffer Overflow로 인한 것이다. 표4로부터 10개 이하의 송신 호스트 수, 즉 접속제어가 수행된 상황과 11개 이상의 송신 호스트 수, 즉 Congestion 상황에 있어서 모두 제안하는 Adaptive RIO 방식이 RIO 방식보다 도착한 In-profile 패킷들의 수와 통과된 In-profile 패킷들의 수가 크므로 제안하는 Adaptive RIO 방식은 RIO 방식보다 In-profile 트래픽에 대한 보호 및 수율성능이 우수하다는 것을 알 수 있다.

그림8은 그림2의 시뮬레이션 모델에서 제안하는 Adaptive RIO 방식과 RIO 방식을 적용했을 때 송신 호스트의 수에 따라 C1라우터에서 100초간 통과된 전체 패킷들의 수와 폐기된 In-profile 패킷들의 수를 나타낸 것으로 제안하는 Adaptive RIO 방식과 RIO 방식의 링크 이용률 성능은 동등함을 알 수 있다. 그러나 RIO 방식은 송신 호스트 수가 증가함에 따라 폐기된 In-profile 패킷 수가 지수적으로 증가하고 Adaptive RIO 방식은 일정함을 알 수

있다. 표4와 그림8에서 Adaptive RIO 방식이 Congestion 상황에서 RIO 방식보다 In-profile 트래픽에 대한 보호 및 수율 성능이 우수한 원인은 그림7에서 보인 것과 같이 구간마다 설정되는 min_out을 통해 In-profile 트래픽과 Out-of-profile 트래픽이 저장되는 버퍼 공간 크기의 평균적인 비율, $1:\beta$를 RIO 방식보다 효율적으로 유지하여 증가한 Out-of-profile 패킷들로 인한 In-profile 패킷의 Buffer Overflow Drop을 감소시켰기 때문이다.

표 4. 각 방식에서 도착한 In패킷수 및 통과된 In패킷 수

Senders	arrived In (RIO)	departed In(RIO)	arrived In (ARIO)	departed In(ARIO)
7	182644	182513	183397	183207
8	207122	206954	207450	207276
9	231707	231542	231852	231771
10	254002	253601	254719	254589
11	269527	268746	275738	275526
12	286085	285042	296884	296619
13	303136	301706	313861	313637
14	318243	316800	329527	329384
15	326779	324947	345572	345393

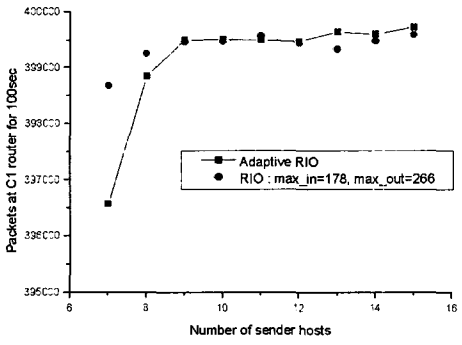


그림 8-(a)

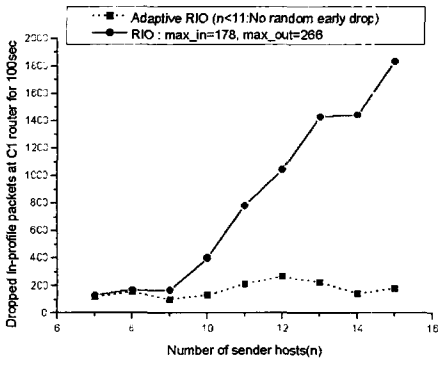


그림 8-(b)

그림 8. 각방식에서 통과된 전체패킷수 및 폐기된 In패킷 수

그림9는 그림2에서 RIO 방식과 Adaptive RIO 방식을 적용했을 때, 접속을 허용한 송신 호스트 수에 따라 R2호스트가 100초간 수신한 In-profile byte 수를 나타낸다. 최대 지연 시간은 0.1초로서 평균 0.266Mbps를 전송하는 호스트에 대해 최대 수신 byte 수는 3325000 bytes (0.266Mbps)이다. 그림9로부터 제안하는 Adaptive RIO 방식은 RIO 방식보다 플로 당 In-profile 트래픽 수율 성능이 우수하다는 것을 알 수 있고, 두 방식 모두 목표 수용 송신 호스트 수 10개까지 0.255Mbps이상의 In-profile 트래픽 수율(0.011Mbps(4.3%)의 Token Loss)이 유지되나, 10개의 송신 호스트 수를 초과하면 In-profile 트래픽의 수율의 감소가 크다는 것을 알 수 있다. 따라서 두 방식에서 모두 계산된 β값으로부터 설정한 접속제어 기준 대역폭의 크기가 유효함을 알 수 있다.

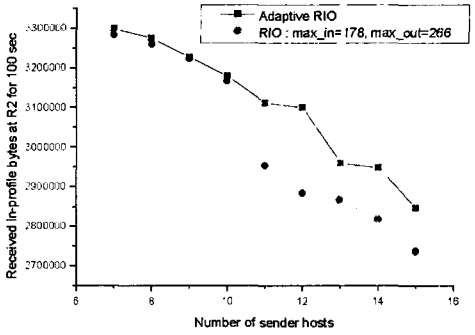


그림 9. 각 방식에서 R2호스트의 수신 In-byte 수

그림10은 그림2에서 각 송신 호스트의 RTT가 다른 경우로서 송신 호스트마다 4ms씩 RTT를 증가시켜 S1 호스트가 가장 작은 RTT를 갖고, S10 호스트가 가장 큰 RTT를 갖는 시뮬레이션 모델이다. 즉, 그림10은 Token Loss에 대한 해결책이 적용되지 못한 경우로 송신 호스트들로부터 불균일한 트래픽이 발생하게 된다. 그리고 그림11은 RIO방식과 제안하는 Adaptive RIO 방식을 적용했을 때 그림10의 각 수신 호스트가 100초간 수신한 In-profile byte 수를 나타낸 것이다. 그림11의 결과에서 제안한 Adaptive RIO 방식은 그림10의 환경에서도 C1 내부 라우터에서 In-profile 패킷에 대한 Early Random Drop이 발생하지 않았으며 폐기된 In-profile 패킷 수가 RIO 방식보다 크게 감소하여 RIO 방식보다 큰 수신 In-profile byte 수와 99.8%의 링크이용률을 나타내었다. RIO 방식은 그림9에서 10개의 호스트 경우에 수신한 In-profile byte 수

보다 적은 양이 수신되었으며 링크이용률은 88.4%를 나타내었다. 표5는 그림10에서 각 송신 호스트 수에 따라 C1라우터에서 100초간 도착한 In-profile 패킷 수와 통과된 In-profile 패킷 수를 나타낸 것으로 Congestion이 발생한 경우에도 제안한 Adaptive RIO 방식이 RIO 방식보다 큰 값을 나타내고, 폐기된 In-profile 패킷 수도 RIO 방식보다 Adaptive RIO방식에서 크게 감소함을 알 수 있다. 따라서 Token Loss에 대한 해결책이 없는 경우에도 제안한 Adaptive RIO 방식을 적용하여 In-profile 트래픽에 대한 수율 및 링크 이용률을 극대화 시킬 수 있음을 알 수 있다.

표 5. 그림10에서 도착한 In패킷 수 및 통과된 In패킷 수

Senders	arrived In (RIO)	departed In(RIO)	arrived In (ARIO)	departed In(ARIO)
7	182086	182001	181471	181395
8	207530	207365	205884	205795
9	228014	227835	229653	229548
10	208212	205535	252134	251945
11	265557	264905	272486	272259
12	281512	280536	291806	291647
13	295776	294546	309168	308887
14	310246	308758	325362	325104
15	318270	316672	341148	340929

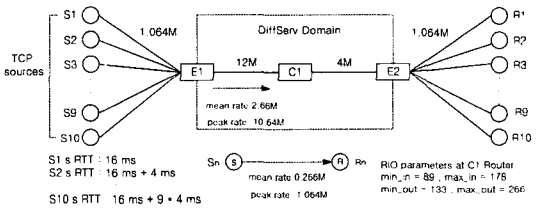


그림 10. 송신호스트의 RTT가 다른 경우의 실험

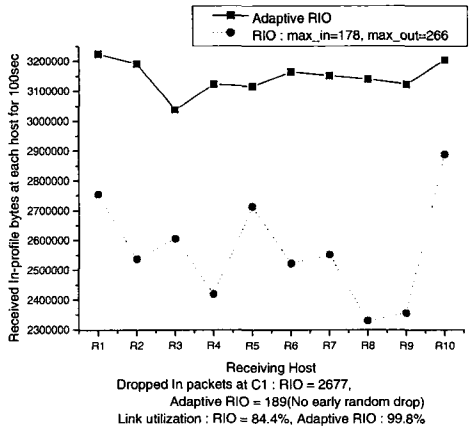


그림 11. 그림10에서 R2호스트의 수신 In-byte 수

Ⅳ. 결 론

본 논문에서는 DiffServ 방식의 AS에 대한 QoS를 보장하기 위해 접속제어가 수행된 상황 하에서 발생하는 RIO 방식의 문제점을 제시하고 이에 대한 해결 방안으로 Adaptive RIO 방식을 제안하였다. 제안하는 Adaptive RIO 방식은 네트워크 토폴로지와 Assured Service 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 일정 시간 구간마다 도착 가능한 In-profile 패킷과 전체 패킷의 최대량을 고려하여 RIO의 변수 값들을 재설정함으로써 허용된 In-profile 패킷에 대한 폐기를 방지한다. 시뮬레이션 결과로부터 Token Loss 발생을 최소화한 경우 AS에 대해 접속제어가 수행된 상황과 Congestion 상황에서 제안하는 Adaptive RIO 방식이 RIO 방식보다 In-profile 트래픽에 대한 보호 및 수율 성능은 우수하고 링크 이용률 성능은 동등함을 알 수 있었다. 또한 Token Loss 발생이 큰 경우에는 제안하는 Adaptive RIO 방식이 RIO 방식보다 더욱 향상된 In-profile 트래픽에 대한 보호 성능 및 링크 이용률 성능을 나타냄을 알 수 있었다.

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 Commu nications 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.

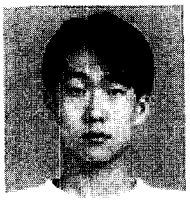
[15] Benjamin Teitelbaum, Susan Hares and et. al., Internet 2 Qbone: Building a Testbed for Differentiated Services, *IEEE Network*, pp. 645-660, September/October 1999.

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

김 문 규(Moon Kyu Kim) 정회원



2001년 2월 : 고려대학교 전자공학과 학사
2001년 3월 ~ 현재 : 고려대학교 전자공학과 석사과정 재학 중

<주관심 분야> 통신네트워크 설계 및 성능분석

이 승 현(Seung Hyun Lee) 정회원

2001년 2월 : 고려대학교 전자공학과 학사
2001년 3월 ~ 현재 : 고려대학교 전자공학과 석사과정 재학 중

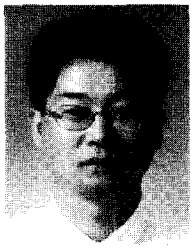
<주관심 분야> 통신네트워크 설계 및 성능분석

조 성 대(Seong-Dae Cho) 정회원

1987년 2월 : 고려대학교 전자공학과 학사
1989년 2월 : KAIST 전기및전자공학과 석사
1999년 2월 : KAIST 전기및전자공학과 박사
1989년 3월 ~ 현재 : 한국통신(KT) 통신망연구소 선임 연구원

<주관심 분야> 광통신시스템, 이더넷 데이터망

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