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

DiffServ 방식의 Assured Service에서 QoS 보장을 위한 효율적인 자원 할당 방안

정회원 허 경*, 조 성 대**, 엄 두 섭*, 차 균 현*

A Novel Resource Allocation Scheme for QoS guarantee of Assured Service in Differentiated Services

Kyeong Hur*, Seong-Dae Cho**, Doo-Seop Eom*, Kyun Hyon Tchah* ***Regular Members***

요 약

본 논문은 DiffServ 방식의 Assured Service에 대한 QoS를 보장하고 서비스 수용 용량을 최대화 시킬 수 있는 자원 할당 방안을 제안하고, RIO 및 Adaptive RIO 버퍼 관리 방식을 적용하여 성능을 분석한다. 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 RIO 방식의 변수 값들을 재설정함으로써 접속제어가 수행된 상황 하에서 허용된 In-profile 패킷에 대한 Early Random Drop을 방지할 수 있다. 시뮬레이션 결과는 제안하는 자원 할당 방안 및 Adaptive RIO 방식을 적용하여 Assured Service에 대한 QoS를 보장할 수 있고 서비스 수용 용량을 최대화 시킬 수 있음을 보인다.

ABSTRACT

In this paper, we propose a novel resource allocation scheme which can maximize capacity for QoS guarantee of Assured Service in Differentiated Services. Performance of the proposed resource allocation scheme is analyzed with each buffer management scheme such as RIO and Adaptive RIO. To prevent an early random drop of the admitted In-profile packet, 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 Out-of-profile 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 can find from simulation results that proposed resource allocation scheme with Adaptive RIO can guarantee QoS and can maximize capacity for Assured Service.

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 있다[2]. 제안된 DiffServ의 PHB 방식에는 PS(Premium Service)에 해당하는 EF(Expedited

* 고려대학교 전자공학과 (hkyeong@korea.ac.kr) ** 한국통신 통신망연구소
논문번호 : 010356-1123, 접수일자: 2001년 11월 23일

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. 따라서 접속 제어가 수행된 상황 하에서 도착하는 In-profile 패킷에 대한 폐기가 발생하지 않아야 한다[13]. 그러나 RIO 방식은 접속제어가 수행된 상황의 사용자가 계약한 In-profile 트래픽에 대한 보호에 있어서 평균 큐 내 In-profile 패킷의 수를 나타내는 avg_in을 이용한 In-profile 패킷의 폐기로 인해 발생하는 문제점을 갖고 있다. 즉, RIO 방식의 . 은 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 $\min_in$을 초과할 경우 In-profile 패킷에 대한 폐기가 발생하게 되어 AS를 사용하는 TCP 플로에게 In-profile 트래픽의 수율을 보장하지 못하게 된다. 이러한 RIO 방식의 문제점과 더불어 AS에 대한 접속 제어 시 고려되어야 할 것은 서비스 수용 용량이다. 즉, 사용자가 수신한 In-profile 트래픽의 수율을 QoS로 정의하면 사용자는 추가적으로 수신된 Out-of-profile 패킷의 양으로 인해 계약한 평균 전송률 이상의 수율을 보장 받게 되나, 라우터 링크의 부분를 Out-of-profile 트래픽이 차지하게 되어 서비스 수용 용량은 감소하게 된다.

이에 대해 본 논문은 DiffServ 방식의 AS에 대한 QoS를 보장하고 서비스 수용 용량을 최대화 시킬 수 있는 자원 할당 방안을 제안하고, RIO 및 Adaptive RIO 버퍼 관리 방식을 적용하여 성능을 분석한다. 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 RIO 방식의 변수 값들을 재설정함으로써 접속제어가 수행된 상황 하에서 허용된 In-profile 패킷에 대한 Early Random Drop을 방지할 수 있다. 즉, 본 논문에서는 AS에 대한 Minimum rate QoS 보장 방안으로 Adaptive RIO 운용 방식을 제시하였고, Adaptive RIO 방식을 사용한 실험 결과로부터 AS의 수용 용량을 최대화할 수 있는 자원 할당 방안을 제안하였다. 한편 DiffServ 방식에서 AS를 사용하는 TCP 플로에게 Minimum Rate의 QoS를 보장하기 위한 기존 연구에서는, DiffServ 방식에서 모델링된 각 사용자 플로의 TCP Behavior에 기초한 TCP 수정 알고리즘 및 개별 플로들의 정보를 관리할 수 있는 DiffServ Domain boundary router에서의 Dropper, Marker, Meter들에 대한 새로운 알고리즘들이 제안되었고 플로 간의 공평성(Fairness)을 향상하기 위해 사용자 플로별 정보관리를 이용하여 집합 트래픽(Aggregate traffic)을 대상으로 한 Marker에서의 새로운 알고리즘들이 제안되었다[13],[16-19]. 즉, DiffServ 방식에서 AS를 사용하는 TCP 플로에게 Minimum Rate의 QoS를 보장하기 위한 기존 연구에서는 개별 사용자 플로들에 대한 정보 관리가 요구되었다. 그러나

제시한 Minimum rate QoS 보장 방안인 Adaptive RIO 방식은 각 사용자가 계약한 트래픽의 양을 In-profile 트래픽으로 marking하는 플로와 연결된 DiffServ Domain 입구 라우터에서의 기본적인 Marker만을 갖고 접속제어가 수행된 상황 하에서 DiffServ 내부 라우터(Core Router)에서의 플로별 관리 없이 QoS를 보장할 수 있는 방식이다. 본 논문의 구성은 다음과 같다. 제 2 절에서는 제안하는 자원 할당 방안을 제시한다. 제 3 절의 시뮬레이션 모델 및 결과는 제안하는 자원 할당 방안 및 Adaptive RIO 방식을 적용하여 Assured Service에 대한 QoS를 보장할 수 있고 서비스 수용 용량을 최대화 시킬 수 있음을 보인다. 끝으로 제 4 절에서 결론을 맺는다.

Ⅱ. AS의 QoS보장을 위한 효율적인 자원 할당 방안

2.1 제안하는 자원 할당 방안

DiffServ 방식에서 규정된 AS In-profile 패킷의 트래픽 특성은 임의의 AS 사용자 i가 신고한 평균 전송률 r_i(token rate) 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_i와 관련된 버스트 길이 l_i(msec)의 요소들로 식(1)과 같이 규정된다[14]. 이러한 AS In-profile 트래픽의 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 차등화 할 수 있다.

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

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

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 접속제어가 수행된 임의의 시간 u_k에서 PS를 사용하는 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 시간 u_k에서 주목하는 DiffServ 라우터와 연결된 n개의 입력 링크들 중 m번째 입력 링크를 출력링크로 하는 이전 DiffServ 라우터에서 AS 클래스가 사용 가능한 대역폭 $L_{m,k}$는 식(2)로 정의된다. 이로부터 식(3)에서 $\lambda_{Top,k}$은 AS에 대해 임의의 시간 u_k에서 네트워크 토폴로지에 따른 라우터의 입력대역폭과 출력대역폭 간의 비율을 나타낸다.

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

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

$R_{ps,m,k}$: 시간 u_k일때 m번째 입력링크에서 PS용으로 예약된 자원량 (2)

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

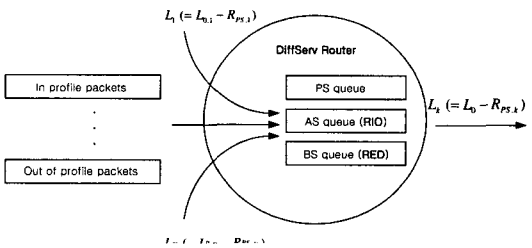


그림 1. 네트워크 토폴로지를 고려한 DiffServ 라우터에서의 AS 트래픽

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.

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

이때 경로상의 병목 구간 라우터에서 식(4)와 같이 접속 제어를 수행하고 허용된 In-profile 패킷들에 대한 폐기가 없을 경우에는 사용자가 계약한 평균 전송률 r_i의 수율이 수신한 In-profile 트래픽의 수율로 보장되게 되고 추가적으로 수신된 Out-of-profile 패킷의 양으로 인해 계약한 평균 전송률 이상의 수율을 보장 받게 된다. 즉, Out-of-profile 트래픽이 이용하는 $L_k\beta/(1+\beta)$의 대역폭의 크기에 대해 식(4)로부터 각 플로가 다중화되면 각 사용자는 계약한 평균 전송률 r_i와 μ_j에 따라 평균적으로 $\beta\cdot\mu_j r_i$의 대역폭만큼 추가적으로 데이터를 수신하게 되어, 사용자는 r_i의 In-profile 트래픽 평균 수율과 전체적으로 $\mu_j r_i(1+\beta)$의 평균적인 데이터 수율을 보장 받게 된다. 그리고 식(4)와 같이 $L_k/(1+\beta)$의 대역폭 크기가 접속 제어 시 기준 대역폭이 되어 AS가 이용 가능한 출력링크 대역폭 L_k만큼 사용자를 수용하지 못하게 된다. 이러한 문제점에 대해서 제안하는 자원 할당 방안은 사용자가 계약한 평균 전송률 r_i와 경로상 L_k가 가장 작은 병목 라우터에서의 β를 고려하여, 각 사용자가 계약한 트래픽의 양을 In-profile 트래픽으로 marking하는 사용자 플로별 Marker의 Token rate를 $r'_i(=r_i/(1+\beta))$로 설정하여 In-profile 트래픽을 발생시킴으로써, μ_j가 1인 경우 사용자에게 계약한 r_i의 평균 데이터 수율을 보장하고 식(5)와 같이 접속 제어 시 기준 대역폭이 L_k가 되어 서비스 수용 용량을 최대화 시키는 방식이다. 예를 들면 사용자가 0.4Mbps(r_i)를 평균 수율로서 계약하였으나 병목 라우터에서의 β가 1/2인 경우, 제안한 자원 할당 방안에서는 사용자 플로별 Marker에서의 In-profile 트래픽 marking rate를 0.4Mbps(r_i)가 아닌 0.266Mbps ($r'_i=r_i/(1+\beta)$)로 설정한다. 그리고 사용자가 계약한 0.4Mbps의 평균 전송률의 Minimum rate QoS를 2.3절의 Adaptive RIO 방식을 적용하여 0.266Mbps의 In-profile 트래픽 수율과 최소 0.133Mbps ($\beta\cdot r_i/(1+\beta)$)의 Out-of-profile 트래픽 수율로서 보장한다. 즉, 4Mbps의 병목 링크에 대해 0.4Mbps의 수율을 계약한 사용자 10명을 수용할 수 있다. 따라서 식(4)와 같이 DiffServ 방식에서 평균 전송률 r_i를 계약한 사용자에게 수신한 In-profile 트래픽의 수율을 평균 전송률의 QoS로 정의하면 임의의 μ_j를 갖는 AS 서브클래스에서는 $\mu_j r_i$의 대역폭를 할당하여 사용자에게 $(b_i-\mu_j)\cdot l_i/\mu_j$의 최대 지연시간을 보장할 수 있다. 그러나 서비스 수용 용량의 감소가 발생한다. 이에 대해 제안하는 자원 할당 방안에서는 Out-of-profile 트래픽을 포함하여 사용자가 수신한 데이터 수율로서 계약한 평균 전송률 r_i의 QoS로 정의함에 따라 모든 AS 서브 클래스의 μ_j는 1로 서브클래스별 대역폭 할당 비율의 차이는 없다. 본 논문에서는 식(4)와 (5)의 방식에 따라 접속 제어가 이루어진 상황 하에서 RIO 방식과 Adaptive RIO 버퍼 관리 방식을 적용하여 성능을 분석한다.

$$\mu_j\sum_{i=1}^{n} r_i' \le \frac{L_k}{(1+\beta)},\ r_i'=\frac{r_i}{(1+\beta)} \qquad \therefore \mu_j\sum_{i=1}^{n} r_i \le L_k$$

$$\mu_j\sum_{i=1}^{n} r_i' b_i l_i \le \frac{L_k\cdot b_{i,\max}\cdot l_{i,\max}}{(1+\beta)} \qquad (5)$$

r_i : 임의의 i사용자가 네트워크와 계약한 평균 전송률

r'_i : r_i에 대해 DiffServ 입구 라우터에서 설정된 Token rate

2.2 RIO 변수 설정

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 연결된 이전 라우터들로부터 τ동안 도착 가능한 In-profile 패킷들의 최대량은 접속 제어를 통해 $L_k/(1+\beta)$의 대역폭을 모두 예약한 경우로 식(4) 및 식(5)와 같이 $L_k b_{i,\max} l_{i,\max}/((1+\beta)packetsize)$가 되고, $1:\beta$의 비율로부터 τ동안 도착 가능한 Out-of-profile 패킷들의 최대량은 평균적으로 $\beta L_k b_{i,\max} l_{i,\max}/((1+\beta)packetsize)$가 된다. 이때

$b_{i,\max}$는 전송 허용폭. 플로들이 신고한 트래픽 정보들 b_i들 중 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 τ동안 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 $\lambda_{\tau op,k}$값은 1보다 크거나 같고 $b_{i,\max}$보다는 작다. 따라서 $\lambda_{\tau op,k}$이 $b_{i,\max}$보다 작은 경우에는 τ동안 도착 가능한 In-profile 패킷들의 최대량은 $L_k\lambda_{\tau op,k}\,l_{i,\max}/((1+\beta)\,packetsize)$가 되고, Out-of-profile 패킷들의 최대량은 $\beta L_k\lambda_{\tau op,k}\,l_{i,\max}/((1+\beta)\,packetsize)$가 된다. 그리고 τ동안 AS큐에 최대로 남게 되는 양은 In-profile 패킷들이 $L_k(\lambda_{\tau op,k}-1)\,l_{i,\max}/((1+\beta)\,packetsize)$가 되고 Out-of-profile 패킷들은 $\beta L_k(\lambda_{\tau op,k}-1)\,l_{i,\max}/((1+\beta)\,packetsize)$가 된다. 한편 RIO 방식에서 max_*in*과 max_*out*은 In-profile 패킷과 전체 패킷들이 각각 최대 이용 가능한 버퍼 크기로 설정한다[8][9][11]. 따라서 RIO 방식의 max_*in*과 max_*out*은 $\lambda_{\tau op,k}$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, $\lambda_{\tau op,k}$이 $b_{i,\max}$보다 작은 경우에는 각각 $L_k(\lambda_{\tau op,k}-1)\,l_{i,\max}/((1+\beta)\,packetsize)$와 $L_k(\lambda_{\tau op,k}-1)\,l_{i,\max}/packetsize$로 설정한다.

표 1. 네트워크 토폴로지를 고려한 RIO 변수 설정값

RIO변수	설정값 ($b_{i,\max}<\lambda_{\tau op}$)	설정값 ($b_{i,\max}\geq\lambda_{\tau op}$)
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

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

$$b_{i,max}\geq\lambda_{top,k},\quad \frac{L_k\cdot\tau}{packet\_size}=\frac{\beta\cdot L_k\cdot\lambda_{top,k}\cdot\tau}{\mu_j(1+\beta)\cdot packet\_size}\quad \therefore\beta=\frac{\mu_j}{\lambda_{top,k}-\mu_j}\qquad(6)$$

max_*in*과 max_*out*값의 설정에 있어서 β값은 링크 이용률을 고려하여 산출되어야 한다. 즉, Out-of-profile 패킷들이 차지하는 β비율의 최대버퍼 공간, max_*out*은 In-profile 패킷들이 하나도 도착하지 않는 상황을 가정하였을 때 Out-of-profile 패킷들이 출력링크를 모두 이용할 수 있도록 설정되어야 한다. 식(6)은 이러한 조건을 고려하여 임의의 μ_j를 갖는 AS 서브클래스에 대해 산출되는 β를 나타낸 것이다. 이에 따라 RIO에서 식(6)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. 한편, 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.

2.3 Adaptive RIO 방식

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 TCP에 의해 데이터 전송이 발생하지 않음으로써 토큰 버킷 내의 토큰이 손실되는 현상이다. 이로 인해 송신 호스트로부터 사용자가 계약한 평균 전송률만큼 In-profile 트래픽이 발생하지 못하게 되고, 결과적으로 송신 호스트로부터 계약한 평균 전송률 만큼 데이터 전송이 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_in, 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 Token Loss 의 발생을 최소화하고 AS에 대한 QoS를 보장하기 위해서는 접속 제어가 수행된 상황 하에서 도착하는 In-profile 패킷에 대한 폐기가 발생하지 않아야 하고 Out-of-profile 패킷에 대한 적절한 폐기가 이루어져야 한다. 그러나 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 $avg\ in$은 지속적으로 증가하게 된다. 그리고 임의의 시간 구간 τ 동안 접속 제어를 통해 허용된 플로들로부터 도착하는 In-profile 패킷 도착량에 의해 avg_in이 설정된 $\min_in$을 초과할 경우 In-profile 패킷에 대한 폐기가 발생하게 되어 AS를 사용하는 TCP 플로에게 In-profile 트래픽의 수율을 보장하지 못하게 되고 Token Loss의 발생 가능성이 증가하게 된다. 이에 대해 Adaptive RIO 방식은 표1과 같이 네트워크 토폴로지와 AS 서브 클래스별로 할당되는 대역폭의 비율에 따라 결정되는 버퍼 크기를 기준으로 시간 구간 τ 마다 τ 동안 도착 가능한 In-profile 패킷과 전체 패킷의 최대량을 고려하여 RIO의 변수 값들을 재설정함으로써 허용된 In-profile 패킷에 대한 폐기를 방지한다. 그리고 In-profile 트래픽과 Out-of-profile 트래픽이 저장되는 버퍼 공간 크기의 평균적인 비율, $1:\beta$를 RIO 방식보다 정확하게 유지하여 버퍼에 저장된 Out-of-profile 패킷들로 인해 발생하는 In-profile 트래픽의 수율 감소(Buffer Overflow Drop)를 완화할 수 있다. Adaptive RIO은 매 시간 구간 τ 마다 avg_in과 avg_total을 0에 가까운 값으로 초기화하고, τ 동안 라우터의 AS 큐에 도착 가능한 In- profile 패킷과 전체 패킷의 최대량이 도착했을 때 도착했을 때 τ 시간후 변화한 avg_in과 avg_total의 최대 및 최소값을 고려하여 구간 τ 마다 $\max\_in$, $\max\ out$과 $\min\_in$, $\min_out$을 재설정한다.

Adaptive RIO 방식에서 사용되는 avg_in과 avg_total의 변수는 RIO 방식에서와 같이 평균 큐 길이를 나타내지 않으므로 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 혼잡제어를 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(7)과 같이 표1에서 $init_q_{in}$만큼 뺀 값이 된다. 그리고 구간 τ 동안 도착 가능한 전체 패킷의 최대량은 In-profile 트래픽과 Out-of-profile 트래픽이 저장되는 버퍼공간 크기의 평균적인 비율, $1:\beta$를 유지하도록 표1에서 $init_q$만큼 뺀 값이 된다.

그리고, 식(7)과 같이 τ 동안 라우터의 AS 큐에 도착 가능한 In-profile 패킷과 전체 패킷의 최대량이 도착했을 경우를 가정하여 식(8), (9), (10)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(8), (9), (10)은 RIO 방식에 기초하여 τ 동안 라우터의 출력링크로 전송되는 In-profile 패킷들의 수, $M_k(=L_k\cdot\tau/(1+\beta)\cdot packetsize)$를 기준으로 한 $init_q_{in}$의 각 경우에서 τ 동안 도착 가능한 In-profile 패킷의 최대량 $\lambda_{\max_in}$이 도착했을 때 $\max ind_in$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(8), (9), (10)에서 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-profile 패킷 수를 나타낸다.

$$for\quad b_{i,\max} < \lambda_{top,k}$$

$$\lambda_{\max\_in} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size} - init\_q_{in}$$

$$\lambda_{\max\_total} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} - init\_q \quad ,$$

$$for\quad b_{i,\max} \geq \lambda_{top,k}$$

$$
\begin{aligned}
&if\ \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size} \geq \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size} - init\_q_{in} \\
&\quad \lambda_{\max\_in} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size} - init\_q_{in} \\
&else \\
&\quad \lambda_{\max\_in} = \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j(1+\beta)\cdot packet\_size} \\
&if\ \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} \geq \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} - init\_q \\
&\quad \lambda_{\max\_total} = \frac{L_k \cdot b_{i,\max} \cdot l_{i,\max}}{\mu_j \cdot packet\_size} - init\_q \\
&else \\
&\quad \lambda_{\max\_total} = \frac{L_k \cdot \lambda_{top,k} \cdot l_{i,\max}}{\mu_j \cdot packet\_size}
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
&Case\,1 \quad init\_q_{in} == 0 \\
&\quad when\ j \cdot \lambda_{top,k}\ packets\ arrive \\
&\quad\quad q_{in} = q_{in} - 1 \\
&\quad when\ packet\ arrives \\
&\quad\quad \max ind\_in = \max ind\_in \cdot (1 - w_q) + w_q \cdot q_{in} \\
&\quad\quad q_{in} = q_{in} + 1 \\
&\quad if\ \lambda_{\max\_in} \leq M_k \\
&\quad\quad \min ind\_in = init\_ind\_in \cdot (1 - w_q)^{M_k - \lambda_{\max\_in}} \\
&\quad if\ \lambda_{\max\_in} > M_k \\
&\quad\quad \lambda_{\max\_in} == (M_k - n_{top}) + beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad for\ i = 1\ to\ beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad\quad q_m = q_{in} - 1 \\
&\quad\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad\quad q_m = q_m - 1 \\
&\quad\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_m \\
&\quad\quad\quad\quad q_{in} = q_{in} + 1
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
&Case\,2 \quad 0 < init\_q_{in} < M_k \\
&\quad when\ j \cdot \lambda_{top,k}\ packets\ arrive \\
&\quad\quad q_{in} = q_{in} - 1 \\
&\quad when\ packet\ arrives \\
&\quad\quad \max ind\_in = \max ind\_in \cdot (1 - w_q) + w_q \cdot q_{in} \\
&\quad\quad q_{in} = q_{in} + 1 \\
&\quad if\ \lambda_{\max\_in} \leq M_k - init\_q_{in} \\
&\quad\quad \min ind\_in = init\_ind\_in \cdot (1 - w_q)^{M_k - init\_q_{in} - \lambda_{\max\_in}} \\
&\quad if\ \lambda_{\max\_in} > M_k - init\_q_{in}\ \&\ \lambda_{\max\_in} \leq (M_k - init\_q_{in}) \cdot \lambda_{top,k} \\
&\quad\quad \lambda_{\max\_in} == (M_k - init\_q_{in} - n_{top}) + beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad for\ i = 1\ to\ beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad\quad q_m = q_m - 1 \\
&\quad\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad\quad q_m = q_m - 1 \\
&\quad\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_m \\
&\quad\quad\quad\quad q_{in} = q_{in} + 1 \\
&\quad if\ \lambda_{\max\_in} > (M_k - init\_q_{in}) \cdot \lambda_{top,k} \\
&\quad\quad \lambda_{\max\_in} == (M_k - init\_q_{in}) \cdot \lambda_{top,k} + beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad q_{in} = n_{top} \\
&\quad\quad for\ i = 1\ to\ \lambda_{\max\_in} \\
&\quad\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad\quad q_m = q_m - 1 \\
&\quad\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_{in} \\
&\quad\quad\quad\quad q_m = q_m + 1
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
&case\,3 \quad init\_q_{in} \geq M_k \\
&\quad when\ j \cdot \lambda_{top,k}\ packets\ arrive \\
&\quad\quad q_{in} = q_{in} - 1 \\
&\quad when\ packet\ arrives \\
&\quad\quad \max ind\_in = \max ind\_in \cdot (1 - w_q) + w_q \cdot q_{in} \\
&\quad\quad q_{in} = q_{in} + 1 \\
&\quad if\ \lambda_{\max\_in} == beta + (n_{top} - 1) \cdot \lambda_{top,k} \\
&\quad\quad q_m = init\_q_{in} - (M_k - n_{top}) \\
&\quad for\ i = 1\ to\ \lambda_{\max\_in} \\
&\quad\quad if\ i = beta + 1 \\
&\quad\quad\quad q_{in} = q_{in} - 1 \\
&\quad\quad if\ i = beta + j \cdot \lambda_{top,k} + 1 \\
&\quad\quad\quad q_m = q_m - 1 \\
&\quad\quad\quad \min ind\_in = (1 - w_q) \cdot \min ind\_in + w_q \cdot q_m \\
&\quad\quad\quad q_{in} = q_{in} + 1
\end{aligned} \tag{10}
$$

식(8), (9), (10)으로부터 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 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(11)과 같이 도출된다. 표2는 2.1절의 RIO 방식과 Adaptive RIO 방식을 정리하여 비교한 것이다.

$$
\begin{aligned}
&\max\_in = \rho_{\max\_in} \cdot \max ind\_in + (1 - \rho_{\max\_in}) \cdot \min ind\_in \\
&\min\_in = \rho_{\min\_in} \cdot \max ind\_in + (1 - \rho_{\min\_in}) \cdot \min ind\_in \\
&\max\_out = \rho_{\max\_out} \cdot \max ind\_total + (1 - \rho_{\max\_out}) \cdot \min ind\_total \\
&\min\_out = \rho_{\min\_out} \cdot \max ind\_total + (1 - \rho_{\min\_out}) \cdot \min ind\_total
\end{aligned} \tag{11}
$$

표 2. 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 보장하고 서비스 수용 용량을 최대화시킬 수 있

는 자원 할당 방안을 제안하였다. 본 논문에서는 사용자가 계약한 평균 전송률 r_i에 대해 Over-provisioning이 없도록 μ_i가 1인 하나의 AS 서브 클래스만을 고려하여 식(4)의 In-profile 트래픽 수율 기준 방식과 식(5)의 전체 트래픽 수율 기준 방식에 따라 접속 제어가 이루어진 상황 하에서 표1과 표2로 제시한 RIO 방식과 Adaptive RIO 버퍼 관리 방식을 적용하여 성능을 분석한다.

그림2의 시뮬레이션 모델은 네트워크 토폴로지를 고려하여 식(4)와 식(5)의 방식에 따라 접속 제어가 수행된 상황을 나타낸다. 그림2에서 각 송신 호스트의 RTT는 서로 다르고 S1 송신 호스트부터 4ms씩 RTT를 증가 시켜 S1 호스트가 가장 작은 10ms의 RTT를 갖고, S10 호스트가 가장 큰 RTT를 갖는 일반적인 경우의 시뮬레이션 모델이다. 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 속도의 In-profile 트래픽과 최대 1.064Mbps속도의 데이터를 전송한다. 사용자 플로별 Marker 내 토큰 버킷의 크기 t_i는 $l_{i,\max}$를 33ms로 가정하여 식(1)로부터 27packets로 설정하였고 전송되는 *packetsize*는 125bytes로 설정하였다. DiffServ 방식에서 각 플로별 l_i는 식(1)에서와 같이 사업자가 플로별로 설치한 Marker 내 토큰 버킷의 크기에 따라 결정된다. 결과적으로 10개의 호스트들로부터 평균 2.66Mbps 속도의 In-profile 트래픽과 최대 10.64Mbps 속도의 트래픽이 12Mbps 링크를 통과하게 된다. 병목링크는 대역폭의 크기가 4Mbps ($=L_k$)인 C1라우터의 출력링크이고 C1라우터의 실제 버퍼크기는 350packets로 설정하였다. 따라서 그림2는 C1 병목 라우터에서 $b_{i,\max}$가 $\lambda_{\tau op,k}$보다 큰 일반적인 경우에 대한 시뮬레이션 모델로서 $\lambda_{\tau op,k}=3$인 경우이다. 그리고 $\lambda_{\tau op,k}$와 표1로부터 1/2로 구해진 β에 따라 4Mbps 대역폭의 병목 링크에 평균 전송률 2.66Mbps의 In-profile 트래픽이 허용된 경우를 나타낸다. 결과적으로 그림2는 AS에 대해 각 송신 호스트가 계약한 평균 전송률 r_i가 0.266Mbps일 때 식(4)의 In-profile 트래픽 수율 기준 방식에 따라 접속제어가 수행된 상황과 각 송신 호스트가 계약한 평균 전송률 r_i가 0.4Mbps일 때 식(5)의 전체 트래픽 수율 기준 방식에 따라 접속제어가 수행되어 경로상 병목 라우터인 C1라우터에서의 β로부터 각 송신 호스트의 Token rate가 0.266Mbps ($r_i/(1+\beta)$)로 설정된 상황을 나타낸다.

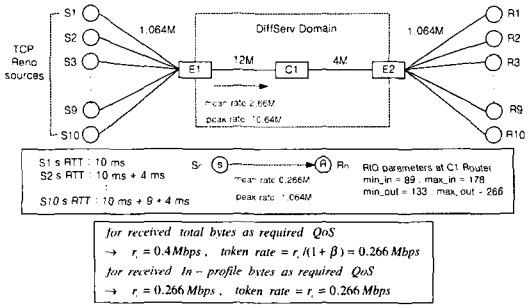


그림 2. 제안한 자원할당 방안에 대한 시뮬레이션 모델

그림2의 시뮬레이션 모델에서 $b_{i,\max}$는 송신 호스트의 최대 전송률(1.064Mbps)과 평균 전송률(0.266Mbps)로부터 4로 산출되어 Token Loss가 없을 경우 보장하는 최대 지연시간은 식(1)로부터 0.1sec가 되고, 요구되는 전체 보호 버퍼공간 크기는 표1로부터 266packets가 된다. 그리고 1/2의 β 값에 따라 In-profile 패킷들은 178개의 버퍼 공간과 2.66Mbps의 대역폭을, Out-of-profile 패킷은 88개의 버퍼공간과 1.34Mbps의 대역폭을 이용하도록 RIO 변수 $\max_in$과 $\max_out$을 178과 266으로 설정하였다. 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 사용하였고 RIO 방식에서 사용되는 w_q값은 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][12][15].

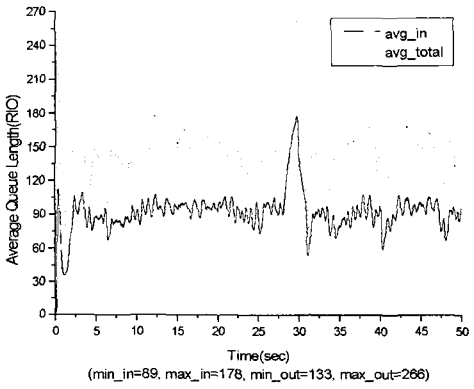


그림 3. C1 라우터에서 RIO 평균 큐 길이의 변화

그림3은 그림2의 C1라우터에 RIO 변수 설정 값을 적용한 경우, 10개의 송신 호스트들로부터 도착하는 패킷들로 인한 평균 큐길이의 변화를 나타낸다. 그림3으로부터 TCP의 버스트한 전송 특성으로 avg_in은 급격하게 증가하나 패킷 폐기가 발생하더라도 크게 감소하지 않음을 알 수 있다. 그리고 avg_in과 avg_total 값은 $\min_in$과 $\min_out$값 주위로 유지되어 $\min_in$과 $\min_out$에 의해 In-profile 패킷과 Out-of-profile 패킷의 확률적인 Early Random Drop이 지속적으로 발생함을 알 수 있다. 그림4는 그림2의 C1 내부 라우터에 Adaptive RIO 방식을 적용할 경우 매 구간 τ에서의 각 $init\_q_{in}$, $init\_q$ 값에 따라 식(7)에서 허용된 In-profile 트래픽과 전체 트래픽의 최대 도착량을 고려하여 식(8), (9), (10)으로 계산되는 $\max ind_in$과 $\min ind\_in$ 및 $\max ind_total$과 $\min ind_total$의 변화를 나타낸다. 그림5는 C1라우터에 Adaptive RIO 방식을 적용했을 때 시간 구간 τ마다 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 $\rho_{\max_in}$은 0.3, $\rho_{\min_in}$은 0.15로 설정하였고, $\rho_{\max_out}$은 0.9, $\rho_{\min_out}$은 0.35로 설정하였다. 그림5로부터 Adaptive RIO 방식을 통해 $\min_in$은 매 구간마다 도착가능한 최대 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 유지하도록 매구간 마다 허용된 전체 트래픽의 최대 도착 패킷량에 따라 설정되어 도착하는 전체 트래픽량의 증가로 ind_total이 지속적으로 큰 변화를 나타낼 경우 $\min_out$이 감소하여 Out-of-profile 패킷에 대한 확률적인 Early Random Drop이 발생함을 알 수 있다.

그림6은 그림2에서 RIO 방식과 Adaptive RIO 방식을 적용했을 때 각 수신 호스트가 100초간 수신한 In-profile byte 수를 나타내는 것으로 Adaptive RIO 방식을 적용하여 RIO 방식보다 플로간 공평성이 향상되었고 증가한 플로 당 평균 수신 In-profile byte 수를 얻을 수 있음을 알 수 있다. 즉, Adaptive RIO 방식은 Token Loss가 발생할 수 있는 그림2의 환경에서 C1 내부 라우터에서 In-profile 패킷에 대한 Early Random Drop이 발생하

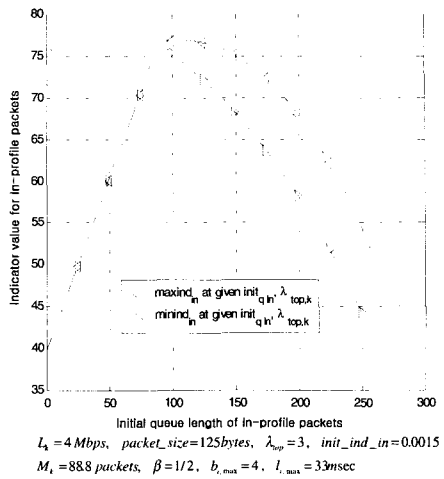


그림 4-(a)

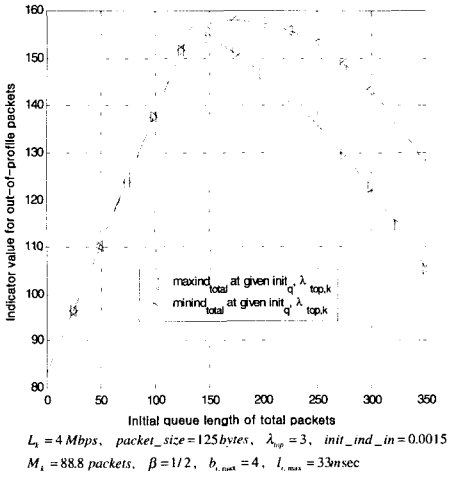


그림 4-(b)

그림 4. Adaptive RIO 방식에서 각 $init_q_{in}$과 $init_q$에 따른 $\max ind_in$과 $\min ind\_in$ 및 $\max ind_total$과 $\min ind\_total$

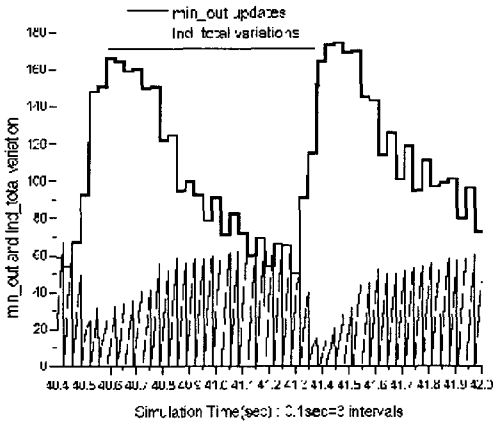


그림 5-(a)

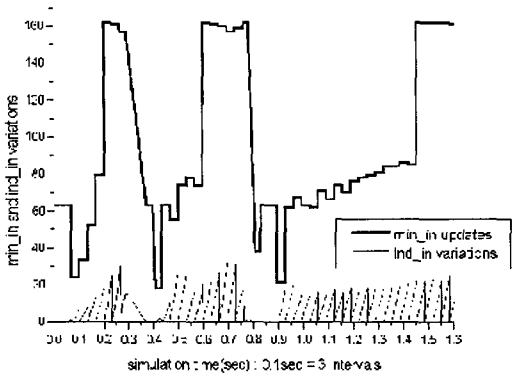


그림 5-(b)

그림 5. min_*in*, min_*out* 및 *ind*_*in*과 *ind*_*total*의 변화

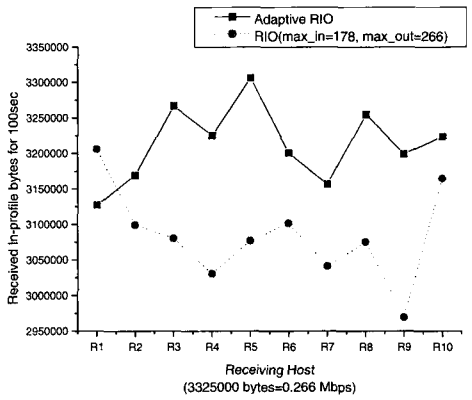


그림 6. 각 방식에서 각 호스트의 100초간 수신 In-byte 수

지 않아 폐기된 In-profile 패킷 수가 RIO 방식보다 감소하여 RIO 방식보다 큰 수신 In-profile byte 수를 나타냄을 알 수 있다. 결과적으로 그림2의 환경에서 AS에 대해 각 송신 호스트가 계약한 평균 전송률 r_i가 0.266Mbps일 때 식(4)의 In-profile 트래픽 수율 기준 방식에 따라 접속제어가 수행된 경우 Adaptive RIO 방식을 적용함으로써 Buffer Overflow Drop으로 인한 In-profile 트래픽 수율 감소분를 제외한 In-profile 트래픽 수율을 보장할 수 있음을 알 수 있다. 그림7은 그림2에서 RIO 방식과 Adaptive RIO 방식을 적용했을 때 각 수신 호스트가 100초간 수신한 전체 byte 수를 나타낸다. 그림7의 결과에서 RIO 방식에서는 RTT가 서로 다른 수신 호스트 간에 불균등한 수신 byte 수를 나타내나, Adaptive RIO 방식에서는 수신 호스트 간에 균등한 수신 byte 수와 RIO 방식보다 증가한 평균 수신 byte 수를 나타냄을 알 수 있다. 그림7의 결과로부터 Adaptive RIO 방식은 RTT가 작은 송신 호스트로부터 RTT가 큰 호스트보다 상대적으로 많이 발생하는 Out-of-profile 패킷들로 인해 RTT가 큰 호스트로부터 발생하는 In-profile 패킷과 Out-of-profile 패킷들이 겪는 큐 내에서의 지연 시간 증가 및 Buffer Overflow Drop의 불공평성을 최소화할 수 있음을 알 수 있다. 즉, 그림5와 같이 Adaptive RIO 방식은 In-profile 트래픽과 Out-of-profile 트래픽이 저장되는 버퍼 공간 크기의 평균적인 비율, 1:β를 RIO 방식보다 정확하게 유지하여 최대 필요 비율, β를 초과하여 버퍼에 저장된 Out-of-profile 패킷들로 인해 이후에 도착하는 In-profile 트래픽 및 Out-of-profile 트래픽이 겪는 성능 열화를 완화할 수 있다. 또한 그림7의 결과로부터 그림2의 네트워크 환경에서 AS에 대해 각 송신 호스트가 계약한 평균 전송률 r_i가 0.4Mbps일 때 (r_i =0.266Mbps) 식(5)의 전체 트래픽 수율 기준 방식에 따라 접속제어가 수행된 경우 Adaptive RIO 방식을 적용함으로써 플로별로 RTT가 다른 경우에도 플로별, 수신 호스트 간 공평한 평균 전송률의 데이터 수율 QoS를 보장할 수 있음을 알 수 있다. 따라서 제안한 자원 할당 방안 및 Adaptive RIO 방식을 적용하여 AS에 대한 QoS를 보장할 수 있고 서비스 수용 용량을 최대화할 수 있음을 알 수 있다.

그림8은 그림6과 그림7에서 보인 바와 같이 RIO 방식보다 향상된 공평성을 보장할 수 있는 Adaptive RIO 방식을 그림2의 시뮬레이션 모델에 적용하여 접속을 허용한 송신 호스트 수에 따라 R2 수신 호스트가 100초간 수신한 In-profile byte 수를 나타낸다. 평균전송률 0.266Mbps의 In-profile 트래픽을 전송하는 플로에 대해 100초간 최대 수신 In-profile byte 수는 3325000 bytes이다. 그림8로부터 Adaptive RIO 방식은 목표 수용 송신 호스트 수 10개까지 계약한 0.266Mbps에 가까운 최대 In-profile 트래픽 수율이 유지되나, 10개의 송신 호스트 수를 초과하면 In-profile 트래픽의 수율의 감소가 크다는 것을 알 수 있다. 따라서 식(4)의 In-profile 트래픽 수율 기준 방식에 따라 접속 제어 수행 시, 계산된 β로부터 설정한 접속 제어 기준 대역폭의 크기가 유효함을 알 수 있다. 또한 Adaptive RIO 방식을 그림2의 시뮬레이션 모델에 적용하여 접속을 허용한 송신 호스트 수에 따라 R2 수신 호스트가 100초간 수신한 전체 byte 수를 나타낸 그림9로부터 평균 전송률 0.4Mbps를 계약한 10개 호스트 수가 최대 AS 서비스 수용 용량임을 알 수 있다.

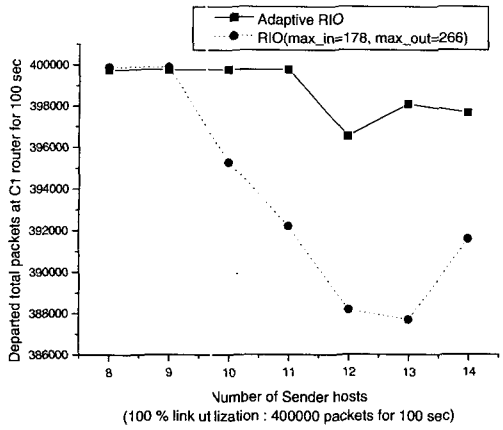


그림 7. 각 방식에서 각 호스트의 100초간 수신전체 byte수

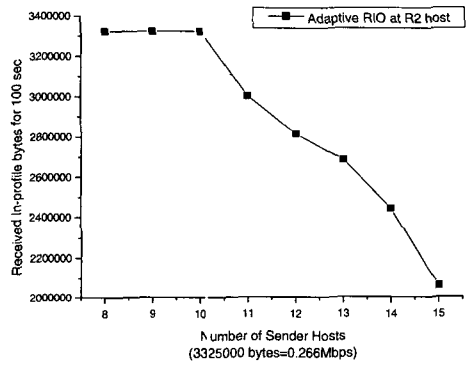


그림 8. Adaptive RIO 방식에서 R2호스트의 수신 In-byte 수

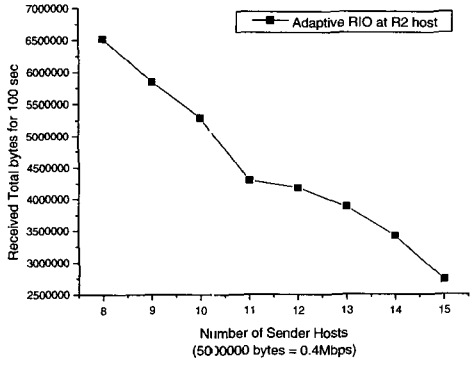


그림 9. Adaptive RIO방식에서 R2호스트의 수신 전체byte수

표3은 그림2의 시뮬레이션 모델에서 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. 표3으로부터 10개 이하의 송신 호스트 수, 접속제어가 수행된 상황과 11개 이상의 송신 호스트 수, Congestion 상황에 있어서 RIO 방식보다 향상된 플로간 공평성을 보장할 수 있는 Adaptive RIO 방식이 RIO 방식보다 도착한 In-profile 패킷들의 수와 통과된 In- profile 패킷들의 수가 크므로 Adaptive RIO 방식이 RIO 방식보다 In-profile 트래픽에 대한 보호 및 수율 성능이 우수하다는 것을 알 수 있다. 그림10과 11은 그림2에서 Adaptive RIO 방식과 RIO 방식을 적용했을 때 송신 호스트의 수에 따라 C1라우터에서 100초간 통과된 전체 패킷들의 수와 폐기된 In- profile 패킷들의 수를 나타낸 것이다. 그림10의 결과로부터 RIO 방식보다 향상된 플로간 공평성을 보장할 수 있는 Adaptive RIO 방식이 RIO 방식보다 플로들로부터 더욱 균일한 트래픽을 발생시킬 수 있으므로 RIO 방식보다 링크 이용률이 높고 100%에 가까운 링크이용률을 나타내는 것을 알 수 있다. 또한 그림11로부터 RIO 방식은 송신 호스트 수가 증가함에 따라 폐기된 In-profile 패킷 수가 지수적으로 증가하나 Adaptive RIO 방식은 일정함을 알 수 있고, Adaptive RIO 방식은 접속제어가 수행된 상황 하에서 In-profile 패킷에 대한 Early Random Drop이 발생하지 않으므로 Adaptive RIO 방식을 적용했을 때 폐기된 In-profile 패킷들은 10개 이하의 송신 호스트 수에서 모두 Buffer Overflow Drop으로 인한 것이다. 위의 결과들에서 보인 Adaptive RIO 방식과 RIO 방식의 성능 차이는 경로상의 병목 구간의 수가 증가할수록 더욱 커질 것으로 사료된다.

Ⅳ. 결 론

본 논문에서는 DiffServ 방식의 AS에 대한 QoS를 보장하고 서비스 수용 용량을 최대화 시킬 수

표 3. C1라우터에 도착한 In패킷수 및 통과된 In패킷 수

Senders	arrived In (RIO)	departed In(RIO)	arrived In (ARIO)	departed In(ARIO)
8	212086	211912	212800	212696
9	233841	233504	239400	239264
10	247357	246768	265993	265827
11	264101	263412	275693	275538
12	270991	270119	277487	277294
13	280648	279575	285664	285499
14	286212	284521	290821	290690

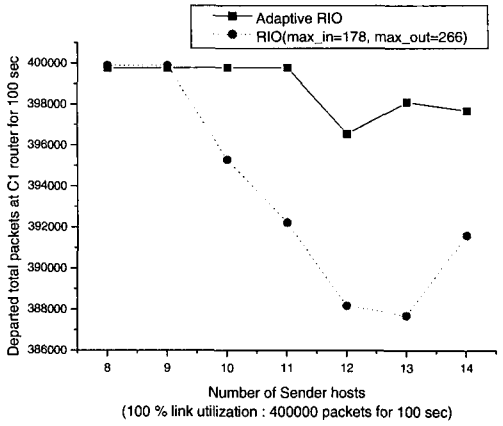


그림 10. C1라우터에서 100초간 통과된 전체 패킷 수

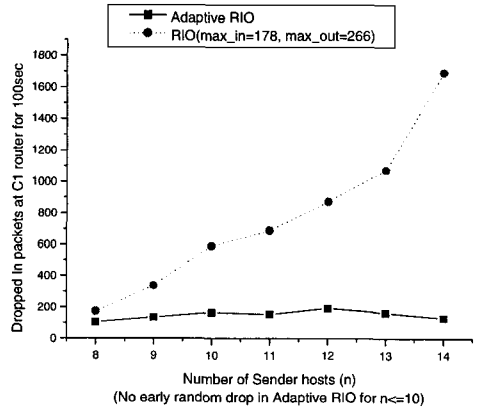


그림 11. C1라우터에서 100초간 폐기된 In패킷 수

있는 자원 할당 방안을 제안하였다. 그리고 In-profile 트래픽 수율을 계약한 평균 전송률의 QoS로 정의한 방식과 전체 트래픽 수율을 계약한 평균 전송률의 QoS로 정의한 제안한 자원 할당 방식에 따라 접속 제어가 이루어진 상황 하에서 RIO 방식과 Adaptive RIO 버퍼 관리 방식을 적용하여 성능을 분석하였다. 시뮬레이션 결과는 플로별로 RTT가 다른 네트워크 시뮬레이션 환경에서 두가지 자원 할당 방식에 대해 Adaptive RIO 방식을 적용하여 RIO 방식보다 향상된 QoS와 플로간 공평성을 보장할 수 있고, 제안한 자원 할당 방식을 적용하여 서비스 수용 용량을 최대화할 수 있음을 나타내었다.

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," RFC 2598, Jun.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] 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.

[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] 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.

[12] Won Hyong Park, Saewoong Bahk, Hyogon Kim, "A modified RIO algorithm that alleviates the bandwidth skew problem in Internet Differentiated Service," Proceedings of *ICC 2000*, 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.

[16] Ikjun Yeom and A. L. Narasimha Reddy, "Marking for QoS Improvement," *Computer Communications*, vol. 1, no. 14, pp. 35 ~ 50, Jan. 2001.

[17] Ikjun Yeom and A. L. Narasimha Reddy, "Modeling TCP Behavior in a Differentiated Services Network," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 31 ~ 46, Feb. 2001.

[18] Ikjun Yeom and A. L. Narasimha Reddy, "Impact of Marking Strategy on Aggregated Flows in a Differentiated Services Network," *IEEE International Workshop on QoS*, pp. 156-158, May. 1999.

[19] Ikjun Yeom and A. L. Narasimha Reddy, "Realizing Throughput Guarantees in a Differentiated Services Network," *IEEE International Conference on Multimedia Computing and Systems*, pp. 372-376, May. 1999.

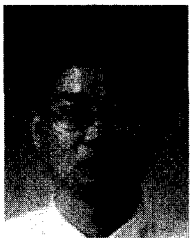
허 경(Kyeong Hur) 정회원

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

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