Next Generation Networks에서의 단대단 성능 보장형 인입제어

정회원 정 진 우*, 준회원 최 정 민*

An Admission Control for End-to-end Performance Guarantee in Next Generation Networks

Jinoo Joung* *Regular Member*, Jeongmin Choi* *Associate Member*

요 약

Next Generation Networks(NGN)는 IP 기반 멀티서비스, multi-access 네트워크로 정의할 수 있다. 여러 종류의 서비스와 access기술이 공존함에 따라 access 네트워크나 코어 네트워크에서의 다양한 전송기술 채택이 NGN의 자연스러운 진화의 방향으로 자리 잡았다. Differentiated Services (DiffServ)와 Multi-protocol Label Switching (MPLS), 혹은 이들이 혼합된 형태의 전송기술들이 복합된 전송 네트워크를 통과하는 플로우들은 통합과 분리를 반복해서 경험하게 된다. 본 연구에서는 이런 환경 하에서 플로우의 단대단 (End-to-end) 지연시간 최대치를 구하는 방법을 제시한다. 이 방법은 네트워크 간 통합플로우에 관한 정보, 특히 각 네트워크에서의 통합플로우별 최대 인입 burst size 값을 교환하는 것을 가정한다. 이를 기반으로, 요청된 단대단 성능 요구를 만족하는지를 판별하고 admission 여부를 정하는 기준을 제시한다. 더 나아가 몇 가지의 실제상황에 가까운 시나리오를 가지고 simulation 하여 이 기준을 평가해 본다.

Key Words : Admission Control, Delay Bound, Flow Aggregation, LR Server, QoS

ABSTRACT

Next Generation Networks (NGN) is defined as IP-based networks with multi-services and with multi-access networks. A variety of services and access technologies are co-existed within NGN. Therefore there are numerous transport technologies such as Differentiated Services (DiffServ), Multi-protocol Label Switching (MPLS), and the combined transport technologies. In such an environment, flows are aggregated and de-aggregated multiple times in their end-to-end paths. In this research, a method for calculating end-to-end delay bound for such a flow, provided that the information exchanged among networks regarding flow aggregates, especially the maximum burst size of a flow aggregate entering a network. We suggest an admission control mechanism that can decide whether the requested performance for a flow can be met. We further verify the suggested calculation and admission algorithm with a few realistic scenarios.

I. 서 론

NGN에 존재하는 DiffServ, MPLS 등 다양한 QoS architecture들은 기본적으로 단일 네트워크 내 (하나의 physical network, 혹은 동일 operator가 운영하는 인접한 네트워크 집합)에서의 동작을 전제로

※ 본 연구는 상명대학교 소프트웨어 · 미디어연구소의 지원으로 수행되었습니다.
* 상명대학교 컴퓨터과학부(jjoung@smu.ac.kr)
논문번호 : KICS2010-01-013, 접수일자 : 2010년 1월 11일, 최종논문접수일자 : 2010년 8월 3일

한다. 이를테면 부분적으로 적용되는 Integrated Services (IntServ)는 주로 access network에서, DiffServ는 코어네트워크에서, 그 밖의 중간 단계 솔루션도 모두 단일 네트워크 내 동작이 전제 된다. MPLS나 Flow-based Traffic Engineering (TE) 네트워크의 제어단위의 granularity가 플로우와 class의 중간이라는 점에서 중간 단계라 할 수 있다. 여기서 그림 1과 같이, IntServ, DiffServ, 혹은 그 밖의 다양한 네트워크를 포함한 인터넷 구성을 생각해 보자. 사실 이러한 네트워크를 ITU-T에서는 Next Generation Networks (NGN)의 기본 형태로 가정하고 있으며 이는 국내에서 정의한 Broadband Convergence Networks (BcN)에서도 마찬가지이다.

이러한 환경의 NGN에서 폭발적으로 증가할 것으로 예상되는 실시간성 multimedia service에 대해서 어떠한 보장을 해줄 수 있는가? 먼저 ITU NGN-GSI에서 정의한 네트워크-레벨 QoS class의 종류와 그 요구사항(requirements)들에 대해서 살펴보면[1], 가장 높은 수준의 서비스를 요구하는 Class 0의 경우, 평균 delay(IPTD)가 100ms 이하, delay variance가 50ms 이하, Loss probability가 10^{-3}이하일 것 등을 규정하고 있다. 과연 우리는 delay, loss rate등의 service parameter와; bandwidth, buffer size등의 resource parameter간의 관계를 알고 있는가? 이러한 문제를 해결하기 위해서, ITU-T Recommendations Y.1541[1]와 Y.1542[2]는 각 네트워크들의 delay, loss rate, jitter 등이 주어져 있을 때 단대단 delay, loss rate, jitter를 구하는 절차를 기술하고 있다. 특정 traffic에 대한 단일 네트워크에서의 성능을 알아내는 데에는 측정(measurement)과 추정(estimation) 두 가지 방법이 있다. 측정은 상대적으로 static한 네트워크에서 잘 맞지만 몇 가지 단점이 있다. 단 방향 성능 측정은 일반적으로 overhead를 내포하며, dynamic한 환경에서 정확하지 못하며, 통계적으로 의미 있는 값을 얻는대 까지 시간이 많이 걸린다. 설령 가능하다고 해도 상당히 복잡한 과정이 필요하며, 특히 loss probability 등의 경우 확률적으로 의미 있는(statistically valid) 값을 얻기 까지 상당히 오랜 시간이 필요하다는 것이다.

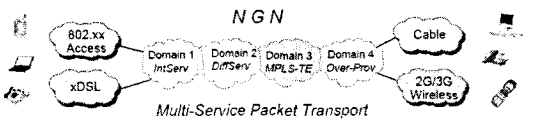


그림 1. NGN의 기본 형태

많은 경우 실시간성 트래픽의 경우라도 플로우의 지속시간(lifetime)이 상당히 짧을 수 있다. 더욱 중요한 것은 이러한 측정기반 접근(measurement-based approach)방식으로는 해당 플로우의 setup 이전에는 품질보장에 관한 어떠한 대답도 할 수 없다는 것이다. 두 번째 방법인 추정은 flow에 대한 specification이 제공될 때에 가능하다. 이 방법으로 특정한 경우에 성능에 대한 최대치를 제공할 수 있다. 제한된 경우이지만 일부 QoS architecture의 경우 플로우와 네트워크의 특성을 설명하는 parameter 값을 알면 delay bound 등의 품질관련 값들을 계산할 수 있다. IntServ에서의 delay bound가 가장 잘 알려져 있는 경우이며[4], DiffServ도 특정조건이 만족되면 해당 값을 계산할 수 있다[5],[6]. 하지만 계산에 필요한 parameter 중 중요한 몇몇(예를 들면, 네트워크 인입 시 플로우의 maximum burst size)은 플로우와 해당 네트워크의 특성만으로 알아낼 수 없다. 이들은 인접한 이전 네트워크에서 제공한 정보를 바탕으로 계산(혹은 추정)이 가능하다. ITU-T Recommendation Y.2122는 통합플로우에 대한 네트워크 운영 parameter들을 교환하여 단대단 성능을 추정할 수 있게 한 표준이다[3]. 이렇게 추정된 성능은 측정된 값에 의해서 좀 더 보완될 수 있다. 이러한 계산에 의한 delay bound 들은 너무 보수적인 경향이 있어서 실제 의미가 퇴색되는 경우가 많다는 문제점도 가지고 있다. 더구나 이렇게 계산된 delay bound들을 더한 값은 네트워크 간의 correlation을 고려하지 않아 더욱 보수적인 값을 가지게 된다. 따라서 worst case의 deterministic한 값보다 statistical 특성을 고려한 계산방법들도 고안되었으나, 이러한 방식은 인입 플로우의 stochastic modeling 자체가 부적절한 문제가 있으며, 일부 IntServ 방식이 strict하게 적용된 경우를 제외한 대부분의 네트워크에서의 계산은 너무 복잡하거나(일반적으로 $O(H^3)$의 complexity를 가짐 - H는 hop count), 현실적으로 의미가 적다[7].

한편 이렇게 계산된 delay bound는 admission control에 사용될 수 있다. Admission control은 크게 Parameter based admission control (PBAC)과 Measurement based admission control (MBAC)으로 나눌 수 있다. 최근 각광받고 있는 MBAC은 측정된 traffic 특성으로 네트워크의 상태를 유추해서 인입될 flow의 성능을 예측하는 알고리즘을 기반으로 한다. 이러한 특징 때문에 MBAC는 비교적 작은 complexity로 인입제어가 가능하다는 장점이 있

으나 성능보장은 불가능하다는 단점이 있다. 본 논문에서 제안하는 admission control 알고리즘은 통합플로우 기반의 정보교환으로 MBAC 정도의 complexity를 가지고서도 플로우의 성능보장이 가능하다.

Ⅱ. 단대단 Delay Bound

2.1 단일 네트워크에서의 지연시간 최대치 추정

지난 20년 동안 다양한 QoS architecture를 가진 네트워크에서의 delay bound를 구하려는 시도가 있었다. IntServ 네트워크의 경우, 전체의 delay bound는 각 노드(서버)의 latency의 합의 함수로 다음과 같이 표현된다[4]. 플로우 i가 겪게 되는 서버 S_i에서의 latency를 θ_i^S 라 할 때 네트워크에서 플로우 i가 겪는 delay:

$$D_i \le \frac{\sigma_i}{\rho_i} + \sum_{i=0}^{N} \Theta_i^S \ .$$

한편 또 다른 축을 이루는 DiffServ에 관해서는 highest priority 트래픽 혹은 premium service 트래픽에 대해서 다음과 같은 delay bound가 존재한다. $D \le \frac{H}{1-(H-1)\alpha}(\triangle + \tau)$. 여기서 H는 네트워크의 최대 hop count, $\triangle = \max_S(L/r^S)$, r^S는 서버 S의 용량, L은 최대 packet 크기, τ는 link 당 최대 burst 크기, α는 네트워크 utilization 이다. 다만 이러한 bound가 존재하기 위해서, $\alpha < 1/(H-1)$의 조건이 만족되어야 한다. 즉, 대략 지름이 10개의 노드를 포함하는 네트워크의 경우 utilization이 10% 이하여야 한다는 것이다[5]. 다른 연구에서는 이 delay bound를 다른 방법으로, 해당 utilization 관련 조건 없이 다음과 같이 구하였다[6]: $D_i \le \frac{\sigma_i}{\rho_i} + \tau\frac{(1+\alpha)^H - 1}{\alpha}$. 이것은 [5]에서 구한 bound 보다 모든 utilization level에서 tight하다. 다만 이 경우 네트워크 topology가 tree 형태로, loop가 없다는 조건이 필요하다.

다음으로는 네트워크 내에서, 혹은 네트워크의 경계에서 플로우의 통합, 분리, 재통합이 있는 경우에 대해서 고려해 보자. 이러한 경우가 중요한 이유는 차세대 네트워크의 다양한 QoS architecture 네트워크를 통과하는 플로우의 입장에서는 필연적으로 다른 플로우와의 통합(MPLS FEC, DiffServ BA 등)을 겪고 다시 분리되는 과정을 거치게 되기 때문이다. 플로우 통합은 DiffServ와 IntServ의 중간단계의 해결책으로 생각될 수 있다. 말단 사용자는 IntServ에서 그랬듯이 필요한 resource를 요청하지만, 네트워크에서는 여러 플로우를 하나의 통합 플로우로 처리하여 복잡성 문제를 해결하는 것이다. 하지만 이러한 플로우 통합이 개별 플로우에 미치는 성능에 대한 영향은 완전히 이해되지 못하고 있다. 일부 연구에서는 평균 지연시간에 나쁜 영향을 미치지 않는다는 결론을 내리며[8], 또 한편으로는 통합 구역(aggregation region) 안에서는 지연시간 최대치(delay bound)가 줄어든다는 연구도 있다[9]. 하지만 플로우의 통합과 분리를 반복하는 것은 확실히 단대단 지연시간에 나쁜 영향을 줄 수 있으며, 이 사실은 DiffServ의 경우를 생각해 보아도 분명하다. DiffServ는 매 노드에서 플로우를 통합, 분리하는 형태의 구조로 생각할 수 있다.

2.2 여러 개의 네트워크를 지나는 플로우의 지연시간 최대치 추정

여러 개의 통합 구역 (Aggregation Region, AR)을 지나는 플로우의 성능을 고찰한다. 각 AR에서 플로우는 네트워크 정책에 따라 통합플로우(Flow aggregate, FA)로 통합된다. 여기서 용어에 대한 정리를 하도록 하자. Flow는 같은 IPv4의 경우에 같은 IP 5-tuple을 가지는, IPv6의 경우에 같은 flow label을 가지는 패킷들의 집합이다. 통합 에지 노드(Aggregation Edge Node, AEN)는 flow 별로 정의되며, 플로우가 속한 통합플로우의 membership이 바뀌는 노드이다. 이를테면, 속해 있는 FA에 새로운 플로우가 추가되는 노드는 AEN이며 여기서부터 새로운 AR이 시작된다. 특정 플로우의 AEN이 다른 플로우에게는 AEN이 아닐 수 있다. AR도 마찬가지로 플로우별로 정의되는데, AEN으로부터 시작되어 다음 AEN 직전까지의 노드들의 집합으로 정의된다. 네트워크의 경계에서 두 개의 AEN이 연속적으로 존재할 수 있다. AR의 정의는 다르게도 내릴 수 있다. 이를테면 AEN이 AR의 마지막에 올 수도 있다. 네트워크 내부에도 AEN이 존재할 수 있다. 단일 네트워크 내에 여러 개의 AR이 존재하는 예로 DiffServ를 들 수 있다. DiffServ 네트워크에서는 일반적으로 하나의 노드가 AR을 형성한다. MPLS LSP aggregation 네트워크[10]도 이러한 예로 들 수 있다. 아래 그림 2는 AEN의 개념도이다. I'''

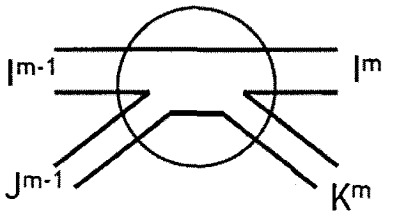


그림 2. AEN의 개념도

은 플로우 i의 m번째 FA이다. 그림 2에서 플로우 i는 I^{m-1}로부터 분리되어 I^m으로 통합된다. J^{m-1}은 I^m에 플로우를 공급하는 FA들을 통합한 FA이다.

여기서 FA I^m에 플로우를 제공한 FA I^{m-1}를 parent FA라고 정의하자. 마찬가지로 I^m을 I^{m-1}의 child FA로 정의한다. FA I^m은 I^{m-1}과 J^{m-1}로부터의 플로우들을 포함한다. 아래 그림 3은 AR의 개념도이다. AR^m은 플로우 i의 m번째 AR이다.

플로우 I의 m번째 AR인 AR^m에서, I^m에 플로우를 공급하는 AR들이 여러 개 있을 수 있는데, 이 중 i이 m-1번째로 지나온 AR인 AR^{m-1}을 제외한 나머지 AR들을 하나로 묶어 AR'^{m-1}로 정의한다. 여기서 특이한 경우에, AR^{m-1}과 AR'^{m-1}의 하나가 동일할 수 있다. 이 경우 AR^{m-1}을 통과하였지만 I^{m-1}의 멤버가 아닌 플로우가 AR^m에서 i와 같은 FA에 속하게 된다.

"Burst-in"이라 정의되는, AR의 인입 시점에서의 최대 burst 크기를 알 수 있다면, 이러한 AR에서의 delay bound는 계산할 수 있다. 하지만 문제는 burst-in이 traffic descriptor에 명시한 값과는 다르다는 데에 있다. ITU-T Recommendation Y.2122에서 이 문제를 해결할 방법을 제안하였다[3]. 이 제안대로 혹은 비슷한 방법으로 네트워크 간에 운영 parameter들이 교환되었다고 가정하면 아래와 같이 Core 네트워크에서의 성능 추정이 가능하다. 먼저 몇 가지 parameter를 정의한다. σ_i^m를 플로우 i가 가지는 AR^m으로부터의 burst-out으로 정의한다. $\sigma_i^0 = \sigma_i$이다. 즉, UNI에서 명시된 maximum burst size값을 σ_i^0로 나타낼 수 있다. 또한, AR에 속한 여러 개의 LR server들은 하나의 equivalent한 LR server로 표현될 수 있다. N개의 LR 서버로 구성된 AR이 FA I를 ρ_I의 rate으로 서비스하는 경우를 고려해 보자. 이 경우 N개의 LR server를 아래와 같은 latency값을 가지는 하나의 서버로 대체해서 생각할 수 있다.

$$\Theta_I^{\mathrm{AR}} = \sum_{S \in \mathrm{AR}} \Theta_I^S.$$

이를 AR latency로 정의한다. 다음과 같이 정리할 수 있다.

Lemma 1. S는 LR server이다. i, j와 k는 parameter (ρ_i, σ_i) 등의 leaky-bucket parameter로 S 앞에서 constrained된 플로우라고 가정한다. S로 들어갈 때, i와 j는 이미 통합되어 있으며, 또한 해당 flow aggregate의 유일한 member라 가정한다. 더 나아가 S 내에서 i, j 와 k는 FIFO queue로 aggregate 된다. Flow i busy period 동안, S 는 flow i에게 다음과 같은 service를 제공 한다.

$$W_i^S(T_0,t) \geq \max\left(0, \rho_i\left(t - T_0 - \frac{\sigma_j + \sigma_k}{\rho_i + \rho_j + \rho_k} - \Theta_{i+j+k}^S\right)\right). \quad (1)$$

T_0는 flow i busy 기간의 시작 시간이다.

[증명] 일반성의 손실 없이 $S1$을 i와 j가 통합되는 S 직전의 서버라고 가정하자. 여기서, flow i, j와 k가 처음으로 통합되는 가상 서버 S'가 있다고 가정하자. 그러면

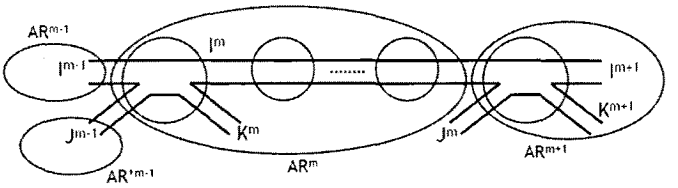


그림 3. AR의 개념도

$$A_i^{S'}(T_0,t) \geq W_i^{S1}(T_0,t) = A_i^{S}(T_0,t). \quad (2)$$

Lemma 1의 서버 S는 i, j와 k에 대해서 가상 서버 S'와 동등하다. S'에서 플로우 j 와 k는 leaky bucket parameter $(\rho_j + \rho_k, \sigma_j + \sigma_k)$로 constrain된 가상 flow j'로 간주될 수 있다. 이 상황은 [11]의 Lemma 1에서 증명된 내용의 특정 경우로 볼 수 있다. □

위 Lemma 1에 따라 다음과 같이 주장할 수 있다.

Theorem 1. Flow aggregate의 모든 flow가 LR 서버와 함께 m번째 aggregation region AR^m 앞에서 leaky bucket parameter로 constrain 되어있고, aggregated data rate가 link capacity 보다 작다는 조건 하에, aggregation region AR^m은 각각의 flow에 대해서 다음과 같이 주어진 latency값을 가지는 LR 서버이다.

$$\Theta_i^{\mathrm{AR}^m} = \frac{\sigma_{I^m \ominus i}^{m-1}}{\rho_{I^m}} + \Theta_{I^m}^{\mathrm{AR}^m},$$

여기서 i는 AR^m 에서 flow aggregate I^m에 속해 있으며,

$$\sigma_{I^m \ominus i}^{m-1} = \sum_{k \in I^m \mathrm{and} k \neq i} \sigma_k^{m-1}.$$

여기서 $\sigma_{I^m \ominus i}^{m-1}$을 Other Burst In (OBI)로 정의하기로 한다. OBI는 플로우 i를 제외한, FA I^m에 속한 모든 플로우들의 burst-in값을 더한 것이다. 정확한 OBI값을 알기 위해서는 각각 플로우들의 burst-out값을 알아야 한다. 하지만 이러한 정보를 모두 저장해 놓는 것은 불가능하다. 대신 우리는 $\sigma_{I^{m-1}}^{m-1}$와 $\sigma_{J^{m-1}}^{m-1}$의 정보를 바탕으로 다음과 같이 추정한다.

$$\sigma_{I^m \ominus i}^{m-1} \approx \frac{n(I^m \ominus I^{m-1})}{n(J^{m-1})} \sigma_{J^{m-1}}^{m-1} + \frac{n(I^{m-1})-1}{n(I^{m-1})} \sigma_{I^{m-1}}^{m-1}, \quad (3)$$

여기서 $n(I)$은 통합플로우 I에 속한 플로우들의 개수이다. 위의 식 (3)에서, $n(I^{m-1})$과 $n(J^{m-1})$은 FA 정보 교환을 통해 쉽게 알 수 있다. $n(I^m \ominus I^{m-1})$ 값을 알기 위해서, AEN이 FA별로 parent FA의 정보를 저장한 table을 유지하고 있어야 한다. 즉, I^m은 모든 parent FA I^{m-1}에 대해서 $n(I^m \cap I^{m-1})$, 즉 동시에 I^m과 I^{m-1}의 member인 플로우들의 개수를 저장한 table을 유지하여야 한다. 만약 I^m이 k개의 parent FA를 가지고 있다면, k개의 정수를 저장하는 것으로 충분하다. Parent FA와 현재 FA 모두의 member인 flow의 개수를 세는 것은 플로우의 admission 때 한번 시행하면 되므로 그렇게 복잡한 작업은 아니다. 다음의 예를 생각해 보자. 어떤 FA I^m이 I_1^{m-1}부터 I_K^{m-1}까지 K개의 parent FA를 가지고 있다. 그렇다면 I^m을 생성한 AEN은 $n(I^m \cap I_1^{m-1})$부터 $n(I^m \cap I_K^{m-1})$까지 K개의 정수를 저장하고 있어야 한다.

위 식 (3)의 계산을 위해 플로우 i에 대해서 다음과 같이 정리할 수 있다.

$$n(I^m \ominus I_1^{m-1}) = \sum_{k=2}^{K} n(I^m \cap I_k^{m-1}).$$

이렇게 OBI $\sigma_{I^m \ominus i}^{m-1}$의 값을 추정하여 최종적으로 아래와 같이 단대단 최대 delay값을 계산할 수 있다.

$$D_i^{\mathrm{E2E}} \leq \frac{\sigma_i}{\rho_i} + \sum_{m=1}^{M} \Theta_i^{\mathrm{AR}^m}.$$

2.3 제안된 계산 기법에 기반을 둔 admission control 알고리즘

Traffic specification과 delay 요구사항을 동반한 플로우의 admission 요청이 들어오면, 해당 path상의 aggregation region들은 각 AR에서의 플로우의 latency를 추정하고, 이 수치를 다음 AR에게 전달한다. 이 latency 값들이 누적되어 계속 전달되고 최종 AR에서는 이 값을 기반으로 단대단 지연시간 최대치를 추정한다. 플로우가 명시해야 하는 traffic specification은 sustainable data rate과 sustainable maximum burst size를 포함하여야 한다. 이 외의 QoS 요구사항, 예를 들어 loss rate 등은, 전체 bandwidth가 특정 threshold 값 이하이면 만족된다고 가정할 수 있다. Latency 값의 추정은 인접한 AR 간의 information 교환을 기반으로 한다. 이러

표 1. 통합플로우별로 저장이 필요한 정보

FA I^m을 위한 Information element	수학적 표현
Flow들의 개수	$n(I^m)$
Aggregate된 최대 burst size의 합	σ_{I^m}
Aggregate된 sustainable rate의 합	ρ_{I^m}
I^m의 parent FA인 모든 FA의 flow들의 개수	$n(I^{m-1})$ 과 $n(J^{m-1})$
Aggregate된 I^m의 parent FA인 모든 FA의 최대 burst size들의 합	$\sigma_{I^{m-1}}$ 과 $\sigma_{J^{m-1}}$
I^m의 parent FA인 모든 FA를 위한 I^m과 I^{m-1}들 다의 member들인 flow들의 개수	$n(I^m \cap I^{m-1})$

한 admission control 과정은 통합플로우의 state만을 유지하면서 가능하다는데 그 매력이 있다. 플로우의 state는 admission 결정 과정에서만 일시적으로 필요하다. AEN들은 이러한 admission control과 information exchange 과정이 구현되는 핵심 노드들이다. 다음 사항들이 upstream AEN으로부터 받은 정보를 통합플로우 별로 저장해 놓아야 하는 대상이다: FA내 플로우의 개수, maximum burst size의 총합, sustainable rate의 총합. 아래 Table은 AEN에 저장해 놓아야 하는 정보를 정리한 것이다. J^{m-1}은 I^m으로 플로우를 공급하는 FA들의 집합을 의미한다.

FA 내의 플로우의 개수의 변화 등, 미리 정해진 trigger event가 발생하면, 정보 전달 절차가 시작된다. FA 정보 교환은 request 혹은 report의 방식으로 진행된다. 이런 방식은 ITU-T Y.2122에서 명시한 방법이기도 하다. Request는 downstream AR에 의해서 시작된다. Request 동작은 실시간성의 dynamics를 필요로 하지는 않는다. Request에 의한 정보 교환 시간은 수초가 걸릴 수도 있다. 이 방식의 정보 수집은 주기적으로 혹은 요구에 의해 시작된다. 주기적 정보 요구 방식에서 polling 간격은 네트워크의 dynamics에 따라 변할 수 있다. 일반적인 경우에는 수분에 한 번씩 요청을 할 수 있다.

알고리즘 1. 통합플로우 정보 Report 알고리즘

Require: Flow acceptance 나 Flow termination 같은 정보를 갱신하기 위해 AR^m에서 미리 정의된 triggering event에 따라

AR^m 은 $n(I^m)$와 $\rho_{I^m}^m$의 정보를 다음의 $AR(AR^{m+1})$로 전달한다.

3: AR^{m+1} 은 I^m이 parent FA인 모든 I^{m+1}마다 $n(I^m)$와 $n(I^{m+1} \cap I^m)$를 갱신한다.

AR^{m+1}은 $\sigma_{I^m}^m$를 갱신한다.

Report 동작에서는, upstream AR이 downstream AR에게 report를 보낸다. Upstream AR에서의 FA에 대한 정보의 변동이 있을 때 report를 보내는 것을 권장한다. 이 방식의 정보 수집 방법도 주기적으로 혹은 FA 구성이 바뀐 경우 이루어 질 수 있다. 아래 기술된 Algorithm 1은 Report에 의한 정보 교환 절차를 상세히 설명한 것이다.

플로우 admission은 상기한 바와 같이 ITU-T Y.1223 등에서 정의된 TSPEC과 함께 user(sender)의 요청으로 시작된다. 이 요청은 단대단 path 상의 AEN들에 의해 검토된다. 각 AEN에서의 자세한 algorithm이 Algorithm 2에 명시되어 있다.

$\Theta_{I^m}^{AR^m}$의 계산을 위한 자세한 과정은 AR에서 사용되는 스케줄러에 따라 달라진다. PGPS, Frame-based Fair queueing, Starting Potential-based Fair queueing등의 Fair-queueing 기반의 많은 scheduler

알고리즘 2. Admission decision 알고리즘

Require: 전송자가 ρ_i와 σ_i의 정보와 함께 flow i의 admission을 요청함에 따라

Ensure: M은 i가 단대단으로 통과할 AR들의 개수이다.

Ensure: $\sigma_{I^0}^0 = \sigma_i$ 과 $\rho_{I^0}^0 = \rho_i$.

Ensure: 표 1의 정보는 단대단 경로에서의 모든 AR을 보여주고 있다.

For $m = 1$ to M do

3: 표 1에 명시되어 있는 정보에 기반을 두어, 다음과 같이 AR^m은 플로우에 관한 AR내의 latency 값을 추정한다.

$$\Theta_i^{AR^m} = \frac{\sigma_{I^{m-1},i}^{m-1}}{\rho_{I^m}} + \Theta_{I^m}^{AR^m},$$

where

$$\sigma_{I^{m-1},i}^{m-1} \approx \frac{n(I^m \ominus I^{m-1})}{n(J^{m-1})}\sigma_{J^{m-1}}^{m-1} + \frac{n(I^{m-1})-1}{n(I^{m-1})}\sigma_{I^{m-1}}^{m-1},$$

and

$$n(I^m \ominus I_1^{m-1}) = \sum_{k=2}^{K} n(I^m \cap I_k^{m-1}).$$

플로우의 E2E latency를 갱신한다.

$$\Theta_i^{E2E} = \Theta_i^{E2E} + \Theta_i^{AR^m}.$$

AR내 FA의 burst-out을 추정한다.

$$\sigma_{I^m}^m = \sigma_{I^m}^{m-1} + \rho_{I^m}\Theta_{I^m}^{AR^m}.$$

6: Passes the updated latency (Θ_i^{E2E}) to the next $AR(AR^{m+1})$.

End for

M번째(마지막) AR은 upstream AR들의 latencies에 기반 하여 E2E delay를 계산한다. admission은 egress node에서 결정된다. egress는 intermediate AEN들에 admission을 통지한다.

들은 $L_i/\rho_i + L_{\max}/r$의 latency를 갖는다. 이 경우 전체 AR의 latency는 $\Theta_I^{AR} = N\left(\frac{L_I}{\rho_I} + \frac{L_{\max}}{r}\right)$, 가 된다. 이 때 N은 AR내의 server (노드)의 개수이며, L_I는 FA I 내 packet size의 최대값, $L_{\max}$는 server들 안에서의 packet size 최대값이며, r은 link의 capacity이다. 이 경우, link capacity와 AR 내에서의 홉 수는 local parameter이며, packet size 최대값은 application 혹은 사용자가 명시한 TSPEC으로부터 알 수 있으므로 새로운 플로우가 들어올 때의 AR latency는 각각 증가된 parameter 값으로부터 쉽게 얻어질 수 있다.

Ⅲ. Bound의 유효성 시뮬레이션

본 연구에서는 현실적인 시나리오를 통해 단대단 delay bound의 정확성을 검증하였다. "ns-allinone-2.31" NS2 package를 사용해 시뮬레이션을 하였다. 시뮬레이션에서 사용한 네트워크 토폴로지는 그림 4과 같다.

토폴로지 상에는 8개의 flow 그룹이 있고 각각 A부터 H까지 표시되어 있다. A, C, E, G 각각의 그룹은 더 나아가 예를 들어 A1과 A2와 같이 2개의 하위 그룹으로 나뉜다. A1은 관찰 대상 flow인 i를 포함한다. 위의 그림 4에서 보는 것과 같이 i는 AR_1^1에서 FA I_1^1의 멤버이고 AR_1^2에서 FA I_2^1 의 멤버이며 AR^3에서 FA I^3 의 멤버이다. FA I_1^1 는 flow 그룹 A와 B를 포함한다. 유사하게 FA I_2^1는 flow 그룹 A와 C를 포함하고, FA I^3 은 flow 그룹 A1, C1, E1과 G1을 포함한다. 이와 같이 flow i는 목적지에 도달 할 때 까지 3번의 aggregation과 2번의 de-aggregation을 겪게 된다. 시뮬레이션을 위해

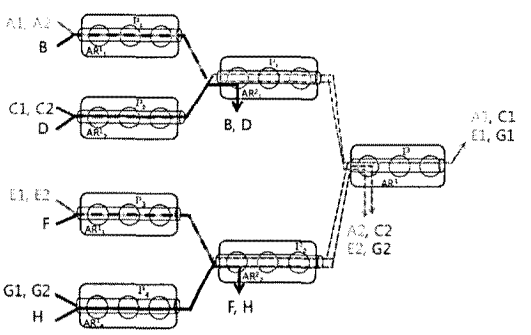


그림 4. Simulation Topology

다음의 parameter를 사용하였다. link bandwidth는 10Mbps이다. 8개의 그룹으로부터 packet의 최대 크기는 400Bytes로 설정되었다. 모든 flow의 기본 설정 data rate는 128Kbps이다. 모든 flow의 기본 설정 최대 burst size는 1600Bytes이다. 모든 AR에서 모든 노드를 교차하는 cross traffic이 존재한다. 이 시뮬레이션에서 cross traffic의 총합은 항상 모든 link의 utilization이 99%가 되도록 설정하였다. 시뮬레이션이 혼잡한 환경으로 설정 되어 있으며, 이 경우가 end-user들의 성능에 중요한 환경이다. 모든 cross traffic은 하나의 노드만을 지나간다. 모든 flow는 Variable Bit Rate(VBR)으로 설정되었다. 이 VBR은 최대, 최소 packet size 및 최대, 최소 data rate를 설정할 수 있고, 최대, 최소값 사이에서 uniformly distribute 한 값을 선택하여 사용한다. 최대 data rate의 경우 평균 data rate의 1.5배를 사용하였다. 이와 같이 최소 data rate 역시 평균 data rate의 절반 값이 사용된다. 모든 flow는 400Bytes의 Maximum Transmission Unit(MTU)을 가지는 User Datagram Protocol(UDP)을 사용한다. 따라서 최대 Packet 길이는 400Bytes이다. 모든 link는 10Mbps의 용량을 가지며, propagation delay는 없다. scheduler는 1Kbps당 0.01의 Quantum size를 할당하는 deficit round robin scheduler이다. 예를 들어 flow가 128Kbps의 data rate를 가지는 경우 quantum size은 1.28이 할당된다. 시뮬레이션의 측정은 네트워크에 packet이 들어오는 시간과 네트워크에서 packet이 나가는 시간의 차이를 packet 단대단 delay로 측정했다. 각각의 시뮬레이션은 250초 동안 수행했다. 시뮬레이션 시작 후 50초에서 250초 사이에 발생하는 packet들을 측정하였다.

각각의 측정 포인트 마다 20회의 시뮬레이션을 하고 평균값을 계산했으며 결과는 아래의 그래프와 같다.

그림 5는 flow 그룹 A를 제외한 모든 다른 그룹으로부터의 flow의 최대 burst size를 변화시키면서 시뮬레이션 한 결과이다. 여기서 flow 그룹 B부터 H까지의 최대 burst size를 기본 설정 최대 burst size의 0.25배에서 8배까지인 400Bytes에서 12800Bytes까지 변화시켰다. 맨 위에서부터, 계산된 단대단 bound, 관찰된 최대 bound, 상위 0.5%를 제외하고 관찰된 최대 delay, 상위 5%를 제외하고 관찰된 최대 delay, 관찰된 평균 delay가 그려져 있다. 계산된 delay bound가 확실히 측정된 최대 delay bound보다 큰 값을 가지고 있음을 알 수 있다. 계산된

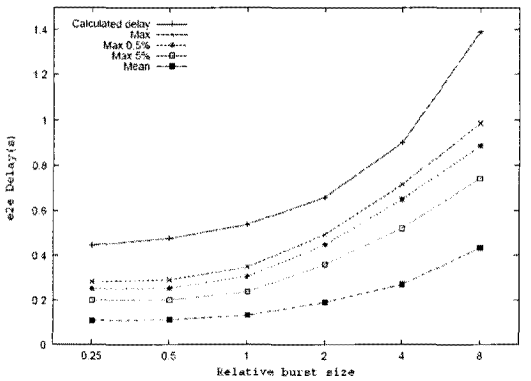


그림 5. Simulation result with the burst size variation

bound는 burst size가 증가함에 따라 delay가 증가한다는 것을 성공적으로 보인다는 의미에서 실제로 효과적이라는 것을 알 수 있다. 또한 최대 burst size와 같은 traffic parameter 에 의하여, 실제의 delay값이 상당한 폭으로 달라진다는 것을 주목해야 한다.

그림 6은 비슷하게 flow 그룹 A를 제외한 나머지 그룹으로부터의 flow의 data rate를 변화 시키면서 시뮬레이션 한 결과이다. 여기서 그룹 B부터 H까지로 부터의 flow의 data rate를 기본 설정 data rate에서 0.01배부터 8배까지인 1.28Kbps에서 1.024 Mbps까지 변화시켰다. 계산된 단대단 delay bound는 네트워크 parameter가 변함에 따라 크게 변하며, 이것은 또한 계산된 bound가 실제의 delay 경향을 표시하는 데에 효과적이라는 것을 보여준다. 또한 data rate와 같은 traffic parameter 에 의하여, 실제의 delay값은 상당한 폭으로 달라진다는 것을 주목해야 한다. 시뮬레이션의 최대 delay는 파라미터의 변화에 따라 0.45초에서 0.2초까지 변한다.

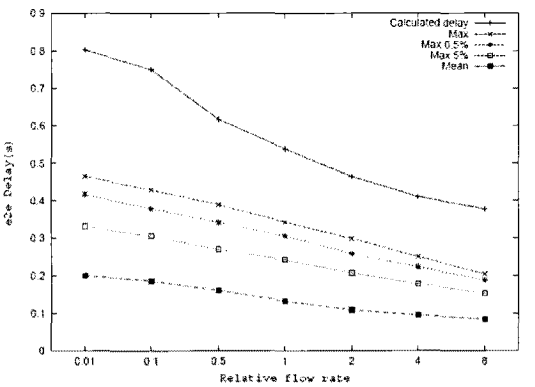


그림 6. Simulation result with the sustainable rate variation

Ⅳ. 결 론

본 연구를 통해 통합과 분리를 반복하는 플로우의 단대단 성능을 추정하는 방법을 제시하였으며, 특정 파라미터 하에서 플로우의 통합이 단대단 성능에 나쁜 영향을 줄 수 있다는 점에서 기존의 대역폭 할당의 admission control에 문제가 있다는 것을 밝혔다. 또한 시뮬레이션을 통해서 다음과 같은 중요한 관찰이 이루어졌다. 첫째, 계산된 delay 최대치가 관찰한 delay 최대치보다 항상 크지만 그 차이가 두 배 이상 벌어지지는 않았다. 둘째, 측정된 최대와 평균 delay 모두 주요 파라미터가 변함에 따라 크게 변화하였다. 마지막으로, 이러한 변화를 계산된 최대 delay가 효과적으로 예측할 수 있다. 이는 실제 네트워크 운영에 있어서, 이렇게 계산된 최대 delay가 유용하게 사용될 수 있음을 의미한다. 더 나아가서 단대단 성능의 측정값을 admission 이후에 얻게 되면 계산된 추정 값을 최적화시키는 데에 사용할 수 있을 것이다. 제안된 admission control은 요구 성능을 명시한 플로우의 인입 요청 시에 단대단 성능을 보장할 수 있게 한다. 따라서 제안된 admission control로 기존 방식인 대역폭 할당보다 훨씬 정교한 플로우 제어가 가능하다. 본 연구와, measurement-based admission control과의 향 후 접점을 찾는 작업이 수행되어야 할 것이다.

참 고 문 헌

[1] ITU-T Recommendation Y.1541, Network performance objectives for IP-based services, Feb., 2006.

[2] ITU-T Recommendation Y.1542, Framework for achieving end-to-end IP performance objectives, July, 2006.

[3] ITU-T Recommendation Y.2122, Flow Aggregate Information Exchange Functions in NGN, 2009.

[4] D. Stiliadis and A. Varma, Latency-Rate servers: A general model for analysis of traffic scheduling algorithms, IEEE/ACM Trans. Networking, Vol.6, No.5 Oct., 1998.

[5] A. Charny and J.-Y. Le Boudec, Delay bounds

in a network with aggregate scheduling, In Proc. First International Workshop of Quality of Future Internet Services (QOFIS), 2000.

[6] J. Joung, Feasibility of Supporting Real-Time Traffic in DiffServ Architecture, in Proc. of WWIC 2007, also in LNCS Vol.4517, pp.189 -200, May, 2007.

[7] F. Ciucu, A. Burchard, and J. Liebeherr, A Network Service Curve Approach for the Stochastic Analysis of Networks, in Proc. of SIGMETRICS, June, 2005.

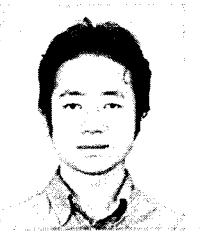
[8] D. Clark, S. Shenker, and L. Zhang, Supporting Real- Time Applications in an Integrated Services Packet Network: Architecture and Mechanism, in Proc. SIGCOMM '92, Sep., 1992.

[9] W. Sun and K. G. Shin, End-to-End Delay Bounds for Traffic Aggregates Under Guaranteed-Rate Scheduling Algorithms, IEEE/ ACM Transactions on Networking, Vol.13, No.5, Oct., 2005.

[10] K. Kompella and Y. Rekhter, Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE), IETF RFC 4206, 2005.

[11] J. Joung, J. Song, and S.S. Lee, Flow Aggregation Criteria in Networks with Rate-Guaranteeing Servers, 한국통신학회논문지 33권 12호, 2008년 12월.

정 진 우 (Jinoo Joung) 정회원



1992년 KAIST 전기전자공학과 공학사

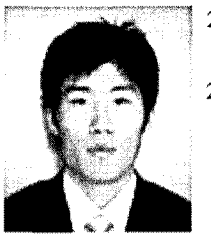
1997년 Polytechnic Univ., NY, USA, 공학박사(Ph.D.)

1997년~2001년 삼성전자 중앙연구소

2001년~2005년 삼성종합기술원

2005년~현재 상명대학교 컴퓨터과학부

최 정 민 (Jeongmin Choi) 준회원



2009년 상명대학교 컴퓨터과학부 학사

2009년~현재 상명대학교 대학원 석사과정