논문 19-44-01-12 The Journal of Korean Institute of Communications and Information Sciences '19-01 Vol.44 No.01
https://doi.org/10.7840/kics.2019.44.1.72

입력포트 기반 계층적 구조의 서버를 이용한 네트워크 지연시간 보장

정 진 우◆

Network Delay Guarantee with Input Port Based Hierarchical Server

Jinoo Joung◆

요 약

다양한 응용분야에서 단대단 네트워크 지연시간 (End-to-end network delay)에 대하여 수 밀리 초에서 수초까지의 엄격한 제한(bound)을 요구하고 있다. 기존 Integrated Services (IntServ) 프레임워크에서의 플로우 기반 스케줄러는 플로우의 수가 N일 때 O(N)혹은 O(log N)의 복잡도를 가진다. 이러한 복잡도가 구현의 제한요소가 되므로 현재 대부분 네트워크는 차선책인 클래스(class)기반 스케줄러를 채택하고 있으나, 해당 스케줄러는 사이클(Cycle)이 형성된 네트워크에서는 사이클을 따라서 최대 버스트(burst)가 계속 증가하기 때문에 지연시간을 제한할 수 없다. 이러한 최대 버스트를 강제로 제한하는 트래픽 레귤레이터(regulator)를 클래스 기반 스케줄러 앞뒤에 배치하여 버스트의 크기를 줄이면서 동시에 스케줄러의 복잡도를 낮추려는 시도가 등장하고 있다. 관련 국제 표준인 IEEE 802.1 Time Sensitive Network (TSN)과 IETF Deterministic Network (DetNet)이 이러한 기술을 표준으로 채택하고 있다. 하지만 레귤레이터가 플로우의 상태정보를 필요로 하기 때문에 다시 복잡도가 증가하는 모순이 존재한다. 본 연구에서는 고순위(high priority) 트래픽에 대해서 인입포트별 큐를 서비스하는 비작업보존형 스케줄러와 클래스별 스케줄러를 계층적으로 사용하는 방안을 제시하였다. 이렇게 함으로써 공정한 스케줄링과 트래픽 레귤레이션이 동시에 구현된다. 복잡성을 크게 낮춘 계층적 구조의 서버를 통해서도 수 밀리 초의 단대단 지연시간을 보장하는 현실적인 해결책이 가능함을 수치 분석을 통해 증명하였다.

Key Words : End-to-end delay guarantee, TSN, DetNet, scheduler, hierarchical server

ABSTRACT

Numerous applications require strict bound on the end-to-end network delay, which is ranged from a few msec to a few seconds. Flow-based schedulers in traditional Integrated Services (IntServ) framework are O(N) or O(log N), when N is the number of flows. Due to such complexity class-based schedulers are adopted in the actual deployments. The class-based systems, however, cannot provides a bounded delay in networks with cycle, since the maximum burst grows infinitely along the cycled path. Regulators adjacent to a scheduler, which limit the maximum burst are adopted as a viable solution. International standards, such as IEEE 802.1 Time Sensitive Network (TSN) and IETF Deterministic Network (DetNet) adopted this approach as a standard. The regulator, however, requires flow state information, therefore contradicts to the purpose of the class-based schedulers. This paper proposes per input port non-work conserving scheduler both for fair scheduling and for traffic regulation.

※ 이 성과는 2018년 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No. 2016R1D1A1B03932066).
◆ First Author : (ORCID:0000-0003-3053-9691)Sangmyung University Department of Computer Science, jjoung@smu.ac.kr, 정회원
논문번호 : 201811-363-B-RN, Received November 13, 2018; Revised December 7, 2018; Accepted December 11, 2018

The low priority traffic share the output port with the high priority traffic that have just passed the said non-work conserving scheduler. The result is a hierarchical server with considerably lower complexity. Despite the lower complexity, we shows the hierarchical server can bound the end-to-end delay in realistic network scenarios, within a few milliseconds.

Ⅰ. 서 론

스마트 팩토리, 차량 간 통신, 차량 내 통신, 전문 음향 네트워크, 대규모 전력 제어 망 등 다양한 응용분야에서 단대단 네트워크 지연시간 (End-to-end network delay)에 대하여 수 밀리 초에서 수초까지의 엄격한 제한(bound)을 요구하고 있다. 관련 국제 표준들도 속속 등장하고 있는데, IEEE 802.1 Time Sensitive Network (TSN)[1]과 IETF Deterministic Network (DetNet)이 대표적이다. TSN은 이더넷을 기반으로 지연시간 보장 및 무손실의 확정적 서비스를 제공하는 기술이다. TSN은 오디오/비디오 전송을 목적으로 하는 Residential Ethernet에서 출발해서 좀 더 전문적인 응용의 AVB(Audio Video Bridging)로 확장되었다가 스마트 팩토리 등 산업용 네트워크까지 추가하여 다양한 실시간성 응용분야를 이더넷 기반의 범용 인터페이스로 통일하고자 시도하고 있다. TSN의 전신인 Residential Ethernet과 이의 후속 기술인 AVB는 A/V 응용을 고려하여 비교적 높은 대역폭을 가지는 소수의 플로우(flow; 같은 발신지/목적지, 같은 application을 가지는, 시간적으로 근접한 패킷들의 집합)가 기술 적용대상인데 반해서[2], TSN은 다수의 저 대역폭 산업용 sensor-actuator 네트워크에서의 플로우까지 고려하고 있다. 이를 위해 크게 포워딩 기술, 시간 동기화 기술, 경로 설정 및 자원 예약 기술, 신뢰성 확보 기술 등을 표준화하고 있다. 본 논문에서는 이 중 포워딩 기술에 중점을 둔다.

단대단 지연시간을 보장하는 잘 알려진 해결책으로 Integrated Services(IntServ)프레임워크를 들 수 있다[3]. IntServ 프레임워크에서의 플로우 기반 스케줄러는 플로우의 수가 N일 때 O(N)혹은 O(logN)의 복잡도를 가진다. 이러한 복잡도가 구현의 제한요소가 되므로 현재의 네트워크는 트래픽의 우선순위(priority)로 구분된 클래스(class)를 바탕으로 한 클래스기반 스케줄러를 채택하고 있으나, 해당 스케줄러는 사이클(Cycle)이 형성된 네트워크에서는 사이클을 따라서 최대 버스트가 계속 증가하기 때문에 지연시간을 제한할 수 없다. 이러한 최대 버스트를 강제로 제한하는 트래픽 레귤레이터를 클래스 기반 스케줄러의 앞뒤에 배치하여 버스트의 크기를 줄이면서 동시에 스케줄러의 복잡도를 낮추려는 시도가 있으며, TSN과 DetNet은 이러한 기술을 표준으로 채택하고 있다. 하지만 레귤레이터가 플로우의 상태정보를 필요로 하기 때문에 다시 복잡도가 증가하는 모순이 존재한다. 본 논문에서는 현재 TSN에서 제시되는 단대단 지연시간 보장에 관한 기술들의 대안으로 다음과 같은 두 단계의 서버를 제안하고 성능을 분석한다. 전체 트래픽은 고순위와 저순위로 나눈다. 고순위 트래픽은 먼저 입력 포트 별로 구분되어 라운드 로빈 기반의 스케줄러를 통과한다. 이 후 strict priority 기반의 스케줄러를 통과하며 저순위 트래픽과 섞여서 출력 포트를 나간다. 결과적으로 간단한 입력 포트 기반 라운드 로빈 스케줄러와 strict priority의 계층적 서버가 하나의 출력 포트 당 하나씩 존재하게 된다. 이를 본 논문에서는 계층적 서버라 칭한다. 본 논문은 다음과 같이 구성되었다. 다음 장에서 관련 연구 및 표준화 동향을 살펴보고, 3장에서 상기한 계층적 서버의 구조를 제시하고 분석한다. 4장에서는 수치 분석을 통해 본 논문에서 제시하는 서버의 성능을 평가하고 기존 방안과 비교한다. 마지막으로 5장에서 본 논문에 대한 결론과 향 후 방향을 제시한다.

Ⅱ. 관련 연구

90년대부터 활발히 연구되어 온 IntServ의 핵심모듈인 스케줄러는 각각의 플로우에게 공정하게 일정 속도 이상의 서비스를 제공해야 한다. 이렇게 각각의 플로우에게 일정 수준 이상의 서비스속도(service rate)를 정해진 시간 (이를 latency라고 하며 delay와는 다름) 이전에 제공하는 다양한 스케줄러들을 Latency-Rate(LR) 서버라고 한다[4]. 하나의 플로우 i가 네트워크를 지나면서 LR 서버들만을 통과 한다면, 이 플로우 i의 패킷들이 겪는 단대단 지연시간의 최대치는 다음과 같은 식으로 표현된다.

$$D_i \leq \frac{\sigma_i - L_i}{\rho_i} + \sum_{j=1}^{k} \Theta_i^{S_j} \quad (1)$$

아래에서 자주 사용하는 수학 기호에 대해서 표로

표 1. 수학 기호와 그 의미
Table 1. Notations and their meaning

Notation	Meaning
L_i	Max packet length of flow i
r	Link capacity
σ_i	Max burst size of flow i
ρ_i	Input data rate of flow i
ϕ_i	Quantum value assigned for flow i
$\Theta_i^{S_j}$	Latency of flow i at sever S_j
D_i	Delay experienced by packets of flow i

정리하였다.

(1)의 의미는 다음과 같다. 경로 상에서 여러 개의 LR 서버를 지나더라도 플로우 지연시간의 최대치는 각각의 latency의 합에 최초 한 번의 최대 버스트에 의한 지연시간만을 더한 것이다. 이러한 LR 서버의 특성을 "Pay burst only once"라고 한다. 또한, LR 서버에 인입되는 플로우가 인입커브 (ρ_i, σ_i)를 따른다면, LR 서버를 통과한 후에는 인입커브 $(\rho_i, \sigma_i + \Theta_i^S \rho_i)$를 따른다. 즉 최대 버스트 크기가 $\rho_i \Theta_i^S$ 만큼 증가하게 된다[4].

가장 이상적인 스케줄러인 GPS(Generalized Processor Sharing)의 패킷들의 서비스 완료 시점인 "Virtual finish time"을 계산해서 이것이 작은 패킷부터 순서대로 서비스 해주는 Packetized-GPS, Self-clocked fair queuing, Virtual Clock 등과, 간단한 라운드 로빈 기반의 Deficit round robin(DRR)[5]과 Weighted round robin이 LR 서버에 포함된다. 이러한 LR 서버 중 virtual finish time 기반의 서버들은 일반적으로 N이 플로우의 수일 때 O(N) 혹은 O(logN)의 구현 복잡도를 보이기 때문에 코어 네트워크에서 구현되기 어렵다. 퀀텀(quantum) 크기가 패킷의 최대길이보다 작은, 일반적인 DRR의 latency는 다음과 같이 주어진다[6].

$$\Theta_i^{DRR} = \frac{1}{r}\left[(F-\phi_i)(1+\frac{L_i}{\phi_i}) + \sum_{n=1}^{N} L_n\right]. \qquad (2)$$

여기서 F는 모든 플로우 ϕ_i들의 합이다. F를 프레임의 크기라고도 한다. DRR의 동작 에 대한 설명은 [5]를 참조하라.

이런 라운드로빈 스케줄러도 플로우의 수가 많아지면 고속으로 동작하도록 구현하기 힘들다. 따라서 다양한 형태의 클래스기반 스케줄러들이 제시되어 왔으며 잘 알려진 DiffServ 프레임워크가 그 대표적인 예이다[7]. DiffServ에서는 네트워크의 상대적으로 플로우의 수가 적은 가장자리 (boundary) 노드들에서 레귤레이션을 하고 내부 (interior) 노드에서는 클래스에 따른 큐 할당과 이에 따른 스케줄링을 수행하는 것으로 지연시간 등의 성능을 어느 정도 제공하는 것을 목표로 한다. 하지만 1장에서 언급한 바와 같이 사이클이 네트워크 내부에 존재하면 지연시간 최대치를 보장하지 못한다[8]. 이에 따라 트래픽 레귤레이터를 모든 노드에 스케줄러와 같이 구현하는 방안이 제안되었다[8,9]. 여기서 제안된 레귤레이터는 그림 1과 같이 플로우별로 동작한다. 해당 시스템은 스케줄러의 복잡도를 낮추고, 병렬로 배치된 레귤레이터의 복잡도를 높이는 방법으로 구현 가능성을 높였으나 역시 실현되지는 않았다.

TSN TG에서 최근 제시된 Asynchronous Traffic Shaping(ATS)은 상기한 시스템의 개선안이라 할 수 있다. AST 기술을 채용한 방식을 TSN 비동기식 방안이라고 부른다[10]. ATS는 Interleaved regulator라고도 하는데, 입력포트별, 클래스별 트래픽 레귤레이션을 출력 포트의 시작점에서 구현하는 것이다. 다만 입력포트

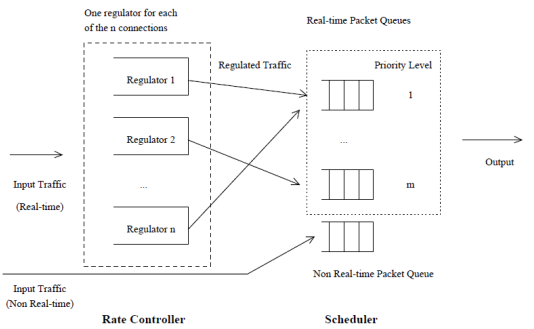

그림 1. [8]에서 제안된 레귤레이터-스케줄러 시스템 구조
Fig. 1. Architecture for regulator-scheduler system in [8]

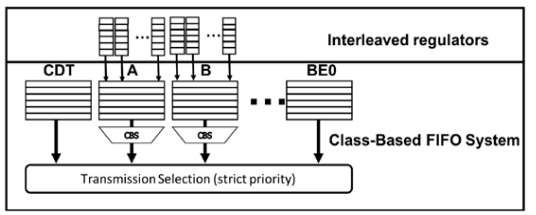

그림 2. TSN의 비동기식 방안: ATS와 기존 Class 기반 FIFO 시스템[10]
Fig. 2. TSN Asynchronous approach: ATS and Class based FIFO system[10]

별 레귤레이션이라 하더라도 모든 플로우의 상태정보를 기억하고 있다가 큐의 가장 앞에 있는 패킷이 속한 플로우를 파악하고 해당 플로우의 상태정보에 따라 레귤레이션 여부를 결정해야 하는 복잡성이 내포되어 있다. 그림 2는 이 비동기식 방안을 도시한 것이다. Interleaved regulator와 클래스별 스케줄러가 구현되어 있다.

Ⅲ. 제안하는 계층적 서버의 구조와 특성

위에서 본 바와 같이 기존 기술들이 플로우 기반의 레귤레이터를 사용하는데 대비하여 본 논문은 입력포트 기반의 레귤레이터를 사용하여 복잡도를 크게 낮춘다. 이러한 입력포트 기반 레귤레이터는 플로우간 공정성을 보장해 주기 위해서 상기한 LR서버 중 하나인 DRR를 바탕으로 비작업보존형으로 만들어진다. 본 논문에서는 이를 SDRR(Smoothing DRR)이라고 부른다. 이러한 SDRR을 출력 포트의 시작점에서 입력 포트별로 통합된 고순위 플로우들에 적용하는 것이 본 논문의 핵심 제안이다. 입력 포트별로 통합된 고순위 플로우들의 집합을 통합플로우(flow aggregate)라고 한다. SDRR과 클래스 기반 스케줄러를 앞뒤로 배치한 계층적 서버를 출력포트에 구현하는 것이 본 논문의 제안이다. 그림 3은 해당 계층적 서버의 구조를 도시한 것이다.

이렇게 계층적 서버에서 입력 포트 별 통합플로우 I의 패킷이 겪는 지연시간은 다음과 같다:

$$D_I \le \frac{\sigma_I - L_I}{\rho_I} + \Theta_I^{S_{SDRR}} + D_I^{S_{SP}}. \qquad (3)$$

여기서 $\Theta_I^{S_{SDRR}}$는 SDRR 서버에서 통합플로우 I의 latency이다. $D_I^{S_{SP}}$는 SP 서버에서 통합플로우 I의 지연시간이다. $\Theta_I^{S_{SDRR}}$와 $D_I^{S_{SP}}$의 수식을 구하자. 먼저

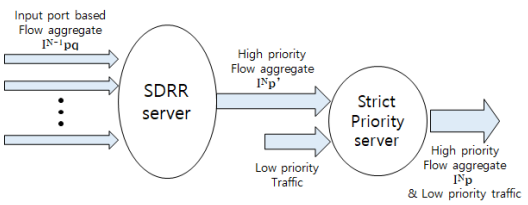

그림 3. 제안하는 계층적 서버의 구조
Fig. 3. Proposed Hierarchical server architecture

$\Theta_I^{S_{SDRR}}$에 대해서 고려하도록 한다. 먼저 아래의 Theorem 1과 2에서 DRR이 제공하는 서비스의 총량이 일정 수준으로 제한된다는 것을 확인한다.

Theorem 1. 플로우 i가 시간구간 (a,b] 동안 계속 backlog되어 있다고 가정하자. (a,b] 동안 DRR turn이 플로우 i에게 k번 서비스를 제공한다고 하자. 이 기간 동안 플로우 i가 받은 서비스의 총량 $W_i(a,b)$는 다음과 같이 제한된다.

$$k\phi_i - \delta_i^k \le W_i(a,b) \le k\phi_i + \delta_i^0$$

여기서 δ_i^k는 i의 (a,b]기간의 첫 번째 라운드부터 세어 k번째 라운드의 끝 시점에서의 deficit값이다.

증명. [5]의 Theorem 4.2의 증명과 동일함. ■

Theorem 2. Backlog 기간 중의 임의의 기간 (a,b] 동안 DRR 서버가 플로우 i에 제공하는 서비스의 총량은 다음과 같이 제한된다.

$$W_i(a,b) \le \rho_i \frac{F_{\max}}{f}(b-a) + \phi_i + L_i.$$

여기서 f는 이 기간 동안 계속 backlog된 플로우들의 퀀텀 값들의 총합이며 $F_{\max}$는 서버의 대역폭과 플로우들의 인입속도가 같은 가상의 상황에서의 퀀텀 값들의 총합이다.

증명. a에서 시작해서 k번째 라운드가 끝나는 시점을 t_k라고 하자. $t_0 = a$이다. 하나의 라운드 기간, $(t_{k-1}, t_k]$의 길이 $t_k - t_{k-1} = \frac{1}{r}\sum_{j\in B_k}(\phi_j + \delta_j^{k-1} - \delta_j^k)$이다. 여기서 B_k는 이 기간 $(t_{k-1}, t_k]$ 동안 backlog되어 있어 서비스를 받는 플로우들의 집합이다. $(t_{k-1}, t_k]$동안 B_k의 원소는 바뀌지 않는다. 여기서 B를 $(t_0, t_k]$동안 계속 backlog되어 있는 플로우들의 집합으로 정의하자. 모든 k에 대해서 $B \subset B_k$이다. 여기서 다음의 부등식이 성립한다. $t_k - t_{k-1} \ge \frac{1}{r}\sum_{j\in B}(\phi_j + \delta_j^{k-1} - \delta_j^k)$. 이 부등식을 k에 대해서 모두 합하면 $t_k - t_0 \ge k\frac{f}{r} + \frac{1}{r}\sum_{j\in B}(\delta_j^0 - \delta_j^k) \ge \frac{f}{r}(k-1)$이다. 왜냐하면 $\Sigma_{j\in B}\delta_j^0 \ge 0$이고 $\Sigma_{j\in B}\delta_j^k \le f$이기 때문이다. 따

라서 $k \le \frac{r}{f}(t_k - t_0) + 1$이다. Theorem 1로부터 $(t_0, t_k]$ 기간 동안

$$W_i(t_0,t_k) \le k\phi_i + \delta_i^0 \le \frac{r}{f}(t_k - t_0)\phi_i + \phi_i + \delta_i^0 \le \rho_i \frac{F_{\max}}{f}(t_k - t_0) + \phi_i + L_i$$

이다. $L_i \le \delta_i^0$이며 $\rho_i / r = \phi_i / F_{\max}$이기 때문이다. 즉, t_0를 포함하여 언제나 deficit 값은 최대 패킷 길이 L_i보다 클 수 없다. 플로우의 인입속도와 퀀텀 값은 비례하며, $F_{\max}$는 서버의 대역폭과 플로우들의 인입속도가 같은 경우의 frame 크기이므로 r에 비례한다. 따라서 t_k와 t_{k+1} 사이의 임의의 시간 b에 대해서

$$W_i(a,b) = W_i(a,t_k) \le \rho_i \frac{F_{\max}}{f}(t_k - a) + \phi_i + L_i \le \rho_i \frac{F_{\max}}{f}(b - a) + \phi_i + L_i$$

이며 Theorem이 성립한다. ■

다음은 기존 DRR을 변형하여 비작업보존형 스케줄러로 동작시키면 결과적으로 레귤레이터로 작동한다는 것을 확인한다. 다음과 같은 algorithm으로 동작하는 비작업보존형 DRR서버를 생각해 보자.

1. 가상의 플로우(v)를 상정한다. 해당 플로우는 실제 고순위 플로우들의 인입 속도의 총합과 서버의 대역폭과의 차이만큼의 인입 속도를 가진 것처럼 취급된다 ($\rho_v = r - \sum_i \rho_i$).

2. 가상 플로우를 포함하여 모든 플로우의 큐가 항상 backlog 되도록, 큐의 서비스 차례 직전에 플로우의 큐가 비어 있으면 deficit 값을 0으로 만든 후 가상의 패킷을 생성한다. 생성되는 가상 패킷의 크기는 해당 플로우의 퀀텀 크기로 한다. 즉, 한 번의 라운드에 서비스 될 수 있는 최대 크기의 패킷을 생성한다.

3. 가상의 패킷을 서비스하는 동안 해당 큐에 실제 패킷이 도착하면 그 즉시 가상 패킷의 서비스를 멈추고, deficit값을 0으로 만든 후 다음 차례의 큐를 서비스한다.

4. 가상 패킷의 서비스가 완료되어도 실제로 링크로 전송하지는 않는다.

Theorem 2에 따라, (a,b]에서 backlog된 플로우의 인입속도의 총합이 서버의 대역폭과 같은 경우, 플로우 i에 대한 서비스는 다음과 같이 제한된다.

$$W_i(a,b) \le \rho_i(b-a) + \phi_i + L_i. \quad (4)$$

따라서 SDRR은 시간당 서비스 총량이 제한되어 bucket 크기가 $\phi_i + L_i$ 인 레귤레이터를 통과한 효과를 가진다. SDRR 스케줄러를 통과한 통합플로우 i의 최대 버스트는 $\phi_i + L_i$이며, ϕ_i는 해당 통합플로우에 할당된 퀀텀의 크기이다. 이렇게 입력 포트별로 큐를 할당하여 SDRR을 적용하면 플로우별로 적용하는 것보다 complexity가 크게 줄어 결과적으로 퀀텀의 크기를 줄일 수 있다. 실제의 구현에서 적용 가능한 퀀텀의 크기는 수십 바이트 정도일 것으로 예상한다.

한편 이렇게 레귤레이트된 트래픽은 Strict priority(SP) 스케줄러에서 저순위 트래픽과 섞인다. SP 스케줄러에서의 고순위 클래스에 속한 개별 플로우의 지연시간 D_I^{Sp}에 대해서 알아보자. 먼저 다음 lemma를 증명하자.

Lemma 1. Strict priority(SP) 서버는 고순위 트래픽이 $A_H(t_0,t) \le \rho_H(t-t_0) + L_H$의 인입커브를 만족하고 인입속도가 서버의 대역폭보다 작을 때 ($\rho_H < r$), 고순위 패킷들에 대해서 다음과 같이 최대 큐 길이와 최대 지연시간을 보장한다.

$$Q_H \le L_H + L_L,\ D_H \le \frac{L_H + L_L}{r}.$$

여기서 D_H와 L_H는 고순위 트래픽의 지연시간과 최대 패킷 길이를 의미한다.

증명. 고순위 트래픽의 인입속도가 서버의 대역폭보다 작으므로 큐 길이가 0인 시각이 존재한다. 즉, 어떤 시각 t_0의 직전에 $Q_H(t_0^-) = 0$ 인 t_0이 존재한다. 나아가 t_0를 고순위 트래픽의 backlog 기간(여기서의 backlog 기간이란 앞에서 정의한 "플로우 busy period" 와는 다르며, 실제로 고순위 트래픽의 패킷이 계속 큐에 있는 기간을 의미한다.)의 시작점이라고 하자. t_0에서 시작한 고순위 트래픽의 backlog 기간 중의 t에 대해서 다음의 부등식이 성립한다.

$$W_H(t_0,t) \ge \max[0, r(t-t_0) - L_L].$$

즉 backlog 기간 동안 고순위 트래픽에 해준 일의 양은 해당 기간 직전에 서비스를 받기 시작한 저순위 패킷에 대한 서비스를 제외하고 계속 고순위 트래픽을 서비스한 것과 일치한다. 한편 고순위 트래픽은 순차적으로 들어오며 인입속도가 r보다 작다는 것과 t_0에서 들어온 패킷의 최대길이를 고려하면 해당 기간 동안의 누적 인입 $A_H(t_0,t) \le r(t-t_0)+L_H$이다. 따라서 t에서의 큐 길이 $Q_H(t) = A_H(t_0,t) - W(t_0,t) \le L_H+L_L$ 이고, $D_H \le \max_t Q_H(t)/r \le (L_H+L_L)/r$ 이다. ■

한편 SP서버는 그 자체로 LR서버이며 따라서 고유의 latency를 가진다. 아래의 Theorem 3과 Corollary 1에서 이것을 증명한다.

Theorem 3. Strict priority (SP) 서버는 고순위 트래픽이 하나의 입력 포트에서 순차적으로 들어오고 최대 인입속도가 서버의 대역폭보다 작을 때 고순위 트래픽에 속하는 개별 플로우 i에 대해서 LR 서버이며 플로우 i의 Latency Θ_i는 다음과 같이 주어진다.

$\Theta_i \le (L_H+L_L)/\rho_i$.

증명. t_0를 플로우 i의 busy period의 시작 시점이라고 하자. busy period 내의 임의의 시각 t에 대해 다음 부등식이 성립한다.

$$W_i(t_0,t) = Q_i(t_0)+A_i(t_0,t)-Q_i(t) \ge A_i(t_0,t)-Q_i(t).$$

$W_i(t_0,t)$는 t_0부터 t까지 i가 받은 서비스의 양이다. 또한 플로우 busy period의 정의에 따라 $A_i(t_0,t) \ge \rho_i(t-t_0)$ 이며 Lemma 1에 따라 $Q_i(t) \le L_H+L_L$ 이다. $W_i(t_0,t)$는 정의에 따라 0보다 작은 값을 가질 수 없으므로 $W_i(t_0,t) \ge \max[0, \rho_i(t-t_0-(L_H+L_L)/\rho_i)]$이 다. LR 서버의 정의에 따라 Theorem이 성립한다. ■

Corollary 1. Strict priority (SP) 서버는 고순위 트래픽이 순차적으로 들어오고 최대 인입속도가 서버의 대역폭보다 작을 때 고순위 트래픽 H에 대해서 LR 서버이며 H의 Latency Θ_H는 다음과 같이 주어진다.

$\Theta_H \le (L_H+L_L)/\rho_H$.

증명. Lemma 1과 Theorem 3의 증명에서 플로우 i를 전체 트래픽 H로 치환하면 같은 방법으로 Corollary를 증명할 수 있다. ■

계층적 서버에서 입력 포트 별 통합플로우 I의 패킷이 겪는 지연시간을 다시 한 번 기술하자.

$$D_I \le \frac{\sigma_I - L_I}{\rho_I} + \Theta_I^{S_{SDRR}} + D_I^{S_{SP}}. \quad (5)$$

$\Theta_I^{S_{SDRR}}$은 Theorem 2와 (1)에 의해, $D_I^{S_{SP}}$는 Lemma 1에 의해 아래와 같은 식으로 구한다.

$$\Theta_i^{SDRR} = \frac{1}{r}\left[(F_{\max}-\phi_i)(1+\frac{L_i}{\phi_i}) + \sum_{n=1}^{N} L_n\right]. \quad (6)$$

$$D_i^{SP} \le (L_H+L_L)/r. \quad (7)$$

위 시스템에서 SDRR의 출력에서 I의 최대 버스트는 Theorem 2과 LR서버의 특징에 의해서 $\phi_I + L_I$로 제한되며, SP의 출력에서 I의 최대 버스트는 Theorem 3에 의해 SDRR서버에서의 latency와 SP서버의 latency를 합한 것에 ρ_I를 곱한 값, $\sigma_I^{SP_{out}} = \sigma_I^{SDRR_{out}} + \rho_I \Theta_I^{SP} = \phi_I + L_I + L_H + L_L$로 제한된다. 일반적인 $L_I = L_H$인 경우라면 $\phi_I + 2L_H + L_L$로 제한된다.

한편 Corollary 1에 의해서 SP스케줄러의 출력 포트에서 출력되는 전체 고순위 트래픽 H (그림 3에서의 I_p^N)의 최대 버스트는 $\sigma_H = \sum_I (\phi_I + 2L_H + L_L)$이다. H전체가 전달된 다음 노드의 입력 포트의 출력인 통합플로우 I의 최대 버스트 σ_I는 통합플로우가 통과한 이전 노드의 출력 포트에서 방출되는 통합플로우의 최대 버스트 보다 같거나 작다. 왜냐하면

$$A_H(t_0,t) = \sum_I A_I(t_0,t) = \sum_I (\sigma_I + \rho_I(t-t_0)) = \sum_I \sigma_I + \sum_I \rho_I(t-t_0) = \rho_H(t-t_0) + \sum_I \sigma_I = \sigma_H + \rho_H(t-t_0)$$

이며 따라서 H에 속한 모든 I에 대해서 $\sigma_I \le \sigma_H$이 성립한다.

결과적으로 본 제안에서는, 단일 플로우에 대해서 스케줄러를 적용하지 않고 인입포트 별로 적용하는 데

에 따라 매 노드마다, 이전 노드에서의 최대 버스트가 매 노드의 지연시간에 영향을 끼친다. 다만 이는 복잡도를 고려하면 필연적인 선택이며, 이러한 영향은 SDRR의 레귤레이션으로 최소화 된다. 특히, 최대 버스트 크기가 매 노드에서 일정하다.

Ⅳ. 수치 분석

본 장에서는 제안하는 계층적 서버의 성능을 분석하고 더불어 TSN에서 제시된 방안과의 비교를 위해 [10]에서 제시된 그림 4와 같은 토폴로지를 그대로 가져와서 사용한다. 관찰대상인 고순위 플로우 f_1은 1, 2, 3, 4번 스위치를 거쳐 목적지에 도착한다. f_2부터 f_5는 관찰 대상과 동일한 spec의 플로우이다. 저순위 트래픽도 존재한다. 따라서 고순위 트래픽은 전체 링크 대역폭의 일부만 사용한다.

아래는 분석에 사용한 parameter 값들이다.

노드에서의 지연시간은 기본적으로 (5)로 구한다. 첫 번째 노드에서는 2개의 플로우가 같은 입력포트와 같은 출력포트를 공유한다. 이를 감안하면 첫 번째 노드의 SDRR 스케줄러에서는 f_1과 f_2가 하나의 큐에서 처리되며 최초 인입시의 최대 버스트는 최대 패킷 길이와 일치한다. 따라서 $D_i^1 \le \Theta_i^{SDRR1} + D_i^{S_{SP}}$이다. 이 때 $\Theta_i^{SDRR1} = \frac{1}{r}\left[(\frac{r}{\rho_i}\phi_i - 2\phi_i)(1+\frac{L}{2\phi_i}) + 2L\right]$ 이며 $D_i^{SP} \le 2L/r$이다.

두 번째 노드부터 마지막 노드까지에서의 지연시간은 $D_i \le \frac{\sigma_i - L}{\rho_i} + \Theta_i^{S_{SDRR}} + D_i^{S_{SP}}$이다. 여기서의 플로우 인입시 최대 버스트 $\sigma_i = 2(\phi_i + 3L)$이다. $\Theta_i^{SDRR} = \frac{1}{r}\left[(\frac{r}{\rho_i}\phi_i - \phi_i)(1+\frac{L}{\phi_i}) + 3L\right]$이며 $D_i^{SP} \le 2L/r$이다. SP 서버의 출력 지점에서의 최대

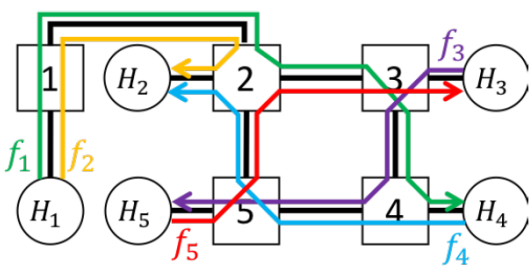

그림 4. 수치 분석을 위한 서비스 시나리오
Fig. 4. Service scenario for Numerical analysis

표 2. 분석에 사용한 파라미터 값
Table 2. Parameter values used in the analysis.

Parameter	Value
L (Max packet length, both high & low priority traffic)	100~1500B
r (Link capacity)	100Mbps
σ_i (Max burst size)	100~1500B
ρ_i (Input data rate)	10~40Mbps
ϕ_i (Quantum size)	10~50B

버스트 $\sigma_i^{SP_{out}} = \sigma_i^{SDRR_{out}} + \rho_i\Theta_i^{SP} = \phi_i + 2L_H + L_L$ 이지만 이 최대 버스트가 다음 노드의 SDRR 서버를 지난 후 다시 $\phi_i + L_i$로 줄어들게 된다. 따라서 해당 플로우가 지나는 모든 계층적 서버의 출력 지점에서의 최대 버스트는 $\phi_i + 2L_H + L_L$로 고정된다. 본 논문에서 고려하는 시나리오에서는 두 개의 플로우가 계층적 서버를 공유하므로 $\sigma_H^{SP_{out}} = 2(\phi_i + 2L_H + L_L)$이며 본 시나리오에서는 $L_L = L_H$이므로 $2(\phi_i + 3L)$이다. 이 값이 다음 노드의 SDRR 서버로의 관찰 대상인 플로우의 입력 최대 버스트와 일치한다 ($\sigma_I^{SDRR_{n+1}} = \sigma_H^{SP_n}$).

그림 5는 퀀텀 값을 80bit로 고정하고 최대 패킷 길이와 플로우 인입속도를 변화시키면서 계산한 단대단 최대 지연시간이다.

아래 표 3에서 그림 5의 특정 부분의 파라미터들과 해당 최대 단대단 지연시간을 표시하였다.

표 3에서와 같이 퀀텀 값을 80bit로 고정하면 10Mbps의 플로우에 대해서 최대 패킷 길이가 1000bit

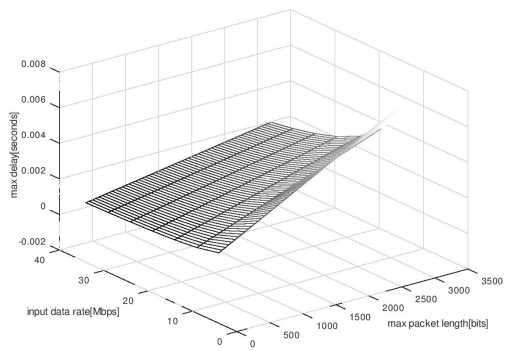

그림 5. 퀀텀값이 80bit일 때 최대 패킷 길이와 플로우 인입속도 변화에 따른 최대 단대단 지연시간
Fig. 5. Max end-to-end delay with variable max packet size, flow arrival rate and fixed quantum value 80bit

표 3. 그림 4의 모서리 값
Table 3. Corner values of Fig. 5

L\ρ_i	10Mbps	40Mbps
400bit	0.876msec	0.261msec
1000bit	2.076msec	0.628msec
3200bit	6.476msec	1.976msec

까지 커져도 약 2msec의 최대 지연시간을 보인다.

그림 6에서와 같이 퀀텀 값은 최대 지연시간에 큰 영향을 끼치지 않는다. DRR의 구현 복잡도는 퀀텀 값에 비례하므로 적당히 큰 퀀텀 값을 선택하여 구현을 좀 더 간단하게 할 수 있겠다.

한 편 [10]의 ATS와의 성능비교를 해보자. [10]은 최대 패킷 길이 1000bit, 플로우 인입속도 20Mbps의 단일 파라미터 집합에 대해서만 값을 제공하고 있는데, 이 경우 0.7msec의 단대단 최대 지연시간을 보인다. 같은 파라미터일 때 퀀텀 값을 80bit로 하면 본 논문이 제안하는 서버의 최대 지연시간은 1.111msec이다. 본 논문에서 제안한 계층적 서버의 성능이 다소 떨어지나, 플로우별 상태 정보를 필요로 하지 않아서 복잡도가 훨씬 낮다는 장점을 감안하면 고무적인 결과라 할 수 있겠다.

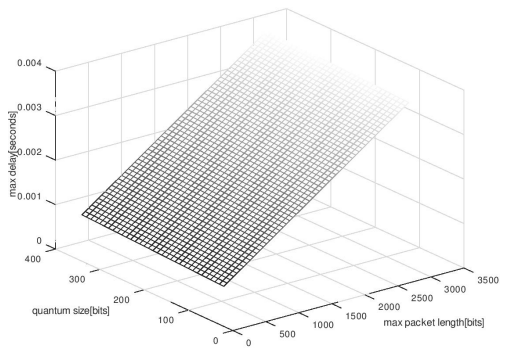

그림 6. 플로우 인입속도가 20Mbps일 때 최대 패킷 길이와 퀀텀값 변화에 따른 최대 단대단 지연시간
Fig. 6. Max end-to-end delay with variable max packet size, quantum value and fixed flow arrival rate 20Mbps

V. 결 론

본 연구에서는 플로우별 큐를 서비스하는 작업보장형 스케줄러 대신 입력 포트별 큐를 서비스하는 비작업보장형 스케줄러를 사용하는 방안을 제시하였다. 추가적으로 저순위 트래픽과는 strict priority 스케줄러를 통해 하나의 출력 포트를 공유하도록 하였다. 이러한 계층적 서버가 구현된 노드들로 이루어진 네트워크의 지연시간 최대치를 구하였다. 플로우 상태 정보가 전혀 필요 없는, 복잡도를 크게 낮춘 계층적 구조의 서버로 대규모 네트워크에서의 단대단 지연시간을 보장하는 현실적인 해결책이 가능함을 증명하였다. 이러한 간단한 구조에도 불구하고 최대 패킷 길이를 1000bit로 제한하면 4 hop에서 2msec 정도의 최대 지연시간을 얻을 수 있었다.

향 후 연구에서는 DRR보다 좀 더 복잡한 형태의, virtual finish time에 기반을 둔 LR서버를 비작업보존형으로 수정하여 계층적 구조 서버에 적용하는 방안을 연구할 예정이다. 좀 더 작은 최대 지연시간 값을 얻을 것으로 기대한다. 더 나아가 일반적인 비작업보존형 LR서버와 Strict priority 스케줄러를 연속적으로 채택한 계층적 구조 서버의 이론적인 연구를 계속해 나아갈 예정이다.

References

[1] *IEEE 802.1 Time-Sensitive Networking Task Group Home Page*, Retrieved Dec. 14, 2018 from http://www.ieee802.org/1/pages/tsn.html

[2] *Residential Ethernet Tutorial*, Retrieved Dec. 14, 2018 from www.ieee802.org/80 2_tutorials/05-March/tutorial_1_0305.pdf

[3] R. Braden, D. Clark, and S. Shenker, *Integrated Services in the Internet Architecture: an Overview RFC 1633*, 1994.

[4] D. Stiliadis and A. Varma, “Latency-rate servers: A general model for analysis of traffic scheduling algorithms,” *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, Oct. 1998.

[5] M. Shreedhar and G. Varghese, “Efficient fair queueing using deficit round-robin,” *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375-385, Jun. 1996.

[6] L. Lenzini, E. Mingozzi, and G. Stea, “Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers,” *IEEE/ACM Trans. Netw.*, vol. 12, no. 4, Aug. 2004.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “*An architecture for differentiated service*,” RFC 2475, 1998

[8] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *J. High Speed Networks*, vol. 3, no. 4, pp. 389-412, 1994.

[9] L. Georgiadis, R. Guerin, V. Peris, and K. N. Sivarajan, "The effect of traffic shaping in efficiently providing end-to-end performance guarantees," *Telecommun. Syst.*, vol. 5, no. 1, pp. 71-83, Mar. 1996.

[10] E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, "Latency and backlog bounds in time-sensitive networking with credit based shapers and asynchronous traffic shaping," *ITC 30*, 2018.

정 진 우 (Jinoo Joung)

1992년 2월 : KAIST 전자공학과 졸업

1994년 8월 : NYU 전기전자공학과 Master

1997년 8월 : NYU 전기전자공학과 Ph.D.

1997년 10월~2005년 2월 : 삼성전자 종합기술원

2005년 3월~현재 : 상명대학교 컴퓨터과학과 교수

<관심분야> 유무선통신, 네트워크, 임베디드 시스템