https://doi.org/10.7840/kics.2024.49.6.829

패킷 메타 데이터를 활용한 링크 총 점유율 추론 방법

정 진 우◆, 권 주 혁°

Total Link Service Rate Estimation with Packet Metadata

Jinoo Joung◆, Juhyeok Kwon°

요 약

네트워크의 단대단 지연 시간 상한을 보장하는 방안에 관해서 많은 연구가 진행되고 있다. 단대단 지연시간 보장을 위한 필요조건은 각 링크에서의 플로우들에 의한 총 점유율(service rate)을 해당 링크의 최대 용량(capacity)보다 낮게 유지하는 것이다. 이를 보장하기 위해 플로우 별로 데이터 송신 전에 수락 제어 (admission control) 과정을 통해 플로우가 지나가는 각 링크의 용량보다 총 점유율이 낮게 유지되는지를 확인한다. 이 과정에서 프로토콜 신호 누락, 노드 상태 변동, 플로우 종료 등의 상황 변화에 대응하기 위해, 개별 플로우들의 점유율 등 정보를 모든 노드들이 유지 관리하여야 한다. 이러한 플로우 별 정보를 "플로우 상태, flow state"라고 통칭한다. 하지만 이러한 플로우가 많아짐에 따라 상태 정보를 일일이 관리하는 것의 복잡도가 증가하여 문제가 된다. 모든 노드들에 기록된 플로우 상태 정보가 동일하도록 유지하는 일은 더욱 어려운 일이다. 따라서 플로우의 상태 정보 관리 없이 인입 제어를 할 수 있도록 링크의 총 점유율을 추론할 수 있는 방안이 절실하다. 본 연구는 코어 노드에서의 플로우 별 상태 유지 관리 없이 수락 제어를 할 수 있도록, 패킷의 메타 데이터(metadata)만으로 링크의 총 점유율을 정확히 추론하는 방안을 제시한다. 점유율을 추론하는 시각을 명시함으로써 정확한 시점에 정확한 점유율을 추론할 수 있다. 이의 효과성을 시뮬레이션을 통해 증명하였다.

Key Words : end-to-end latency, deterministic networking, flow states, TSN, finish time

ABSTRACT

Guaranteeing end-to-end network latency bounds is required in many applications. A necessary condition for ensuring end-to-end latency is to keep the total service rate for flows on each link lower than the maximum link capacity. To ensure this, before data transmission for each flow, through an admission control process, it is necessary to check whether the total service rate remains within the capacity in each link through which the flow passes. In this process, all nodes must maintain information such as the service rate of individual flows in order to respond to missing protocol signals or node failures. This flow-specific information is collectively referred to as "flow state." However, as the number of flows increases, the complexity of managing state information increases. Therefore, it is needed a method to infer the total service rate of the link without flow state maintenance. This study proposes a method to accurately infer the total service rate of a link using only packet metadata so that admission control can be performed without state maintenance for each flow at the core node. By specifying the time of inference, the exact service rate can be inferred at the correct time. Its effectiveness is proven through simulation.

※ 본 연구는 상명대학교 교내 연구과제의 지원을 받아 수행된 연구임.
◆ First Author : Sangmyung university, Department of Human-centered AI, jjoung@smu.ac.kr, 정회원
° Corresponding Author : Sangmyung University Department of Artificial Intelligence and Informatics, juhuk98@gmail.com, 종신회원
논문번호 : 202401-007-C-RN, Received January 3, 2024; Revised March 13, 2024; Accepted March 22, 2024

Ⅰ. 서 론

네트워크에서 각 플로우의 요구성능을 만족시키기 위해서 각 링크에서의 플로우들에 의한 총 점유율(service rate)을 해당 링크의 최대 용량(capacity)보다 낮게 유지해야 한다는 것이 잘 알려진 필요조건이다. 이를 보장하기 위해 플로우 별로 데이터 송신 전에 수락 제어 (admission control) 과정을 통해 1) 해당 플로우의 점유율을 알려주고, 2) 플로우가 지나가는 각 링크에서 용량보다 총 점유율이 낮게 유지되는지를 확인한 후, 3) 해당 플로우의 인입을 수락한다. 이러한 수락 제어 프로토콜의 대표적인 예가 RSVP(Resource reSerVation Protocol)이다[1]. 이 과정에서 프로토콜 신호 누락, 노드 오류 등 상태 변동, 플로우 이상 종료 등의 상황 변화에 대응하기 위해, 개별 플로우들의 점유율 등 정보를 모든 노드들이 유지 관리하여야만 총 점유율을 정확하게 알아낼 수 있다. 이러한 플로우 별 정보를 "플로우 상태, flow state"라고 통칭한다. RSVP는 플로우별 soft state를 유지한다.

하지만 이러한 플로우가 많아짐에 따라 상태정보를 일일이 관리하는 것의 복잡도가 증가하여 문제가 된다. 모든 노드들에 기록된 플로우 상태정보가 동일하도록 유지하는 일은 더욱 어려운 일이다. 따라서 플로우의 상태 정보 관리 없이 인입 제어를 할 수 있도록 링크의 총 점유율을 추론할 수 있는 방안이 절실하다.

Ⅱ. 관련 연구

이러한 링크의 총 점유율을 추정하는 기존 기술로 [2]를 들 수 있다. [2]는 $b(p)=\{A(p)-A(p-1)\}r$ 값을 이용해서 점유율을 추정한다. 여기서 p는 패킷, r은 p가 속한 플로우의 점유율 (서비스율, service rate 이라고도 함), p-1은 해당 플로우의 바로 이전 패킷, A(p)는 p가 최초 네트워크 인입 노드에 도착한 시간이다. b(p) 값을 패킷 메타 데이터(metadata)로 기재하여 이후 노드에서 b(p) 값을 이용하여 총 점유율을 추정한다. 일정 시간구간 (T) 동안 인입한 패킷들의 b(p)의 총합을 T로 나누어 이 값을 총 점유율로 추정한다.

하지만, 이 방법은 최초 노드에서는 정확한 값을 추론할 수 있지만, 이후 노드에서는 패킷들의 실제 도착 시간 간격이 최초 노드의 도착 시간 차 A(p)-A(p-1) 값과 다르기 때문에, 패킷 지연 시간의 최대 변화량만큼 오류가 발생한다. 또한, 기간 T가 길어지면 추정이 좀 더 정확해지나, T 동안 새롭게 허락되어 신규로 패킷을 생성하는 플로우들에 대한 보상이 이루어져야 하는 점도 복잡도를 키우게 되며, 전체 추정의 정확도에도 다시 영향을 준다. 본 논문에서는 이러한 오류를 배제하면서 간단하게 정확히 추론하는 방안을 제시한다.

표 1. 논문에서 사용하는 수학 심볼들의 정의
Table 1. Definitions of symbols used

Symbol	정의
p	p-th packet of the flow under observation.
h	h-th node in the path of the flow under observation
$F_h(p)$	finish time (완료시간) of packet p at node h
$A_h(p)$	arrival time of packet p at node h
$L(p)$	length of packet p
$r,\ r(p)$	service rate (점유율) of the flow under observation
$d_h(p)$	delay factor function of packet p at node h

먼저 본 논문에서 사용하는 수학 심볼들을 표 1과 같이 정의한다.

한편, 발신지와 목적지가 동일한, 같은 응용에 속하는 패킷들의 집합인 플로우를 유체(fluid)로 보고 이를 바탕으로 중계 노드에서의 적절한 스케줄링으로 모든 플로우들에게 정확하게 요구한 만큼의 서비스를 제공해주는 기술이 1990년대부터 제시됐다. Generalized processor sharing (GPS)은 전송할 데이터를 유체로 보고 이상적인 서비스 순서를 결정하는 패러다임을 제시했고, 이를 패킷 환경에서 구현한 PGPS(혹은 Weighted fair queuing)는 이러한 유형의 패킷 기반 스케줄러들의 선구적 역할을 하였다[3]. PGPS는 아래와 같은 수식으로 도출된 완료시간(Finish time)의 오름 차순으로 패킷의 서비스 순서를 결정한다.

$$F(p)=\max\{F(p-1),\ V(A(p))\}+L(p)/r. \quad (1)$$

여기서 p는 플로우의 p번째 패킷, A(p)는 패킷 p가 노드에 도착한 시간, L(p)는 p의 길이, r은 p가 속한 플로우에 할당된 서비스 rate을 의미한다. V(t)는 virtual time function이라고 부르며 시간 t에 서비스받는 플로우들의 r의 총합과 링크 용량의 비율을 곱해준 값이다. 완료시간 F(p)는 이상적인 플루이드 환경을 가정했을 때 공정하게 산출한 p의 완료시간을 나타내며 이 값이 작은 순서대로 노드의 서비스를 받는다.

(1)의 핵심은, 최악의 경우 즉 모든 플로우들이 active 하여 링크가 full로 사용되는 경우, 플로우에 속한 이전 패킷 대비 L(p)/r만큼의 간격으로 서비스를 받게 한다는 것이다. 그러면서 동시에 작업보존방식 스케줄러를 사용함으로써 사용되지 않고 낭비되는 링크자

원이 없도록 방지하고 있다.

(1)을 구하기 위해 플로우의 F(p-1), 즉 이전 패킷의 완료시간을 기억하고 있어야 한다. 패킷이 인입되면 어떤 플로우에 속하는지 찾아내어 해당 플로우의 최근 패킷의 완료시간을 알아내어야 한다. 이 최근 패킷의 완료시간 F(p-1)이 소위 말하는 '플로우 상태'를 대표하는 값이다. 이러한 상태 정보를 기억하고 있다가 읽어야 하는 점이 수백만 개의 플로우를 관리하는 코어 노드 입장에서 상당한 복잡도를 의미하여, 인터넷에서 이러한 PGPS 계열의 스케줄러가 실제로는 사용되지 않는 가장 주요한 이유가 되었다.

V(t)를 계산하는 일도, 현재 서비스받는 플로우들을 실시간으로 추적해서 이들의 r의 총합을 계산해야 하는 복잡성을 내포하고 있다. 플로우들의 시작이 임의의 짧은 시간 동안 무수히 많을 수 있다는 것을 고려하면 상당이 어려운 계산임이 틀림없다. 따라서 V(t)를 정확하게 계산하는 대신 이를 간단한 방법으로 추정하는 방안들이 제시됐다[4-6]. 이 중 Virtual clock[4]은 V(t) 대신 현재시간 t를 사용해서 완료시간을 결정한다.

최근 제안되어 2023년 11월 현재 IETF DetNet WG[7]에서 활발하게 논의되고 있는, 작업보존형 stateless fair queuing (C-SCORE)[8,9,10] 방식에서는 Virtual clock을 활용하여 코어 노드에서 state 관리 없이, 패킷의 메타 데이터를 이용하여 각 노드에서의 이상적인 "완료시간"을 구해서 이에 따라 패킷의 서비스 순서를 결정한다.

C-SCORE에서 노드 h에서 패킷 p의 완료시간을 $F_h(p)$라고 하며 아래와 같은 방법으로 결정된다. 먼저, 최초 인입 노드를 0이라고 하면, 노드 0에서의 완료시간 $F_0(p)$는 아래와 같이 구한다.

$$F_0(p) = \max\{F_0(p-1), A_0(p)\} + L(p)/r. \quad (2)$$

$F_0(1)=0$이다. 최초 노드 이후의 노드 h에서의 완료시간 $F_h(p)$는 아래와 같이 구한다.

$$F_h(p) = F_{h-1}(p) + d_{h-1}(p). \quad (3)$$

여기서 $d_h(p)$를 delay factor라고 하며, 다양한 함수로 제시할 수 있다. (3)을 이용해 $F_h(p)$를 구하기 위해서는 $F_{h-1}(p)$와 $d_{h-1}(p)$가 필요한데, 이들을 packet p의 메타 데이터로 노드 h-1에서 기재하여 다음 노드 h에서 사용할 수 있어야 한다.

C-SCORE[8-10]는 상당히 높은 수준의 플로우 분리 역량을 가지고 있어서 주목받고 있으나 수락 제어의 구체적인 방법은 제시하고 있지 않다. 본 논문에서는 C-SCORE가 DetNet WG에서 채택되어 인터넷에 적용된다는 가정하에, 간단한 방법으로 해당 기술이 사용하는 메타 데이터를 최대한 활용하고 플로우 상태 정보 관리 없이 총 링크 점유율을 정확하게 추정하는 방안을 제시한다.

Ⅲ. 총 링크 점유율 추론 방안

(2)를 변형하여 아래와 같이 전개할 수 있다.

$$F_0(p) = F_0(p-1) + L'(p)/r. \quad (4)$$

여기서 $L'(p)$는 다음과 같이 정의된다.

$$L'(p)/r = \begin{cases} L(p)/r, & \text{if } A_0(p) < F_0(p-1), \\ A_0(p) - F_0(p-1) + \dfrac{L(p)}{r}, & otherwise. \end{cases}$$

더 나아가, 위 (3)에서 $d_h(p)$, delay factor를 노드와 플로우만의 함수로 규정한다. 즉, 특정 플로우에 속한 패킷 간에는 $d_h(p)$의 변화가 없다. 패킷의 최초 노드에서 도출한 "완료시간" F(p)와 이전 패킷의 완료시간 F(p-1)의 차이가, 이후의 모든 노드에서 동일하다. 이러한 경우 모든 노드에서 아래의 관계가 성립한다.

$$F(p) = F(p-1) + L'(p)/r. \quad (5)$$

(5)를 바탕으로, 전체 시간이 (0, F(1)], (F(2)-L'(2)/r, F(2)], ...의 구간들로 분할(partition)된다. 즉, 임의의 시각 t가 (F(p)-L'(p)/r, F(p)]에 속하면, 다른 구간에 속하지 않는다. 따라서, 임의의 시각 t에서 해당 플로우의 점유율을 알고 싶으면, t가 속한 구간 (F(p)-L'(p)/r, F(p)]을 파악하고, 즉, $F(p)-L'(p)/r < t \leq F(p)$임을 파악하고, 해당 패킷 p의 메타 데이터로 기재된 r을 읽고 이를 t에서의 플로우의 점유율로 추론하면 정확하다. 이러한 추론을 위해 메타 데이터로 F(p), L'(p), r 값이 필요하다. A(p)와 F(p), L(p)와 L'(p)의 관계가 그림 1에 도시되어 있다. 전체 시간은 L'(p)/r의 간격으로 분할된다. 따라서 임의의 시간 t는 하나의 (F(p)-L'(p)/r, F(p)]에 반드시 속한다.

t를 정하면 이를 "감싸는" 구간이 정해지며, 이 구간에 해당하는 패킷 p가 결정된다. p에 기재된 메타 데이터를 통해 r(p) = r을 알아낼 수 있다. 플로우가 하나만 있을 때는 이러한 접근 방식이 필요 이상으로 복잡하다

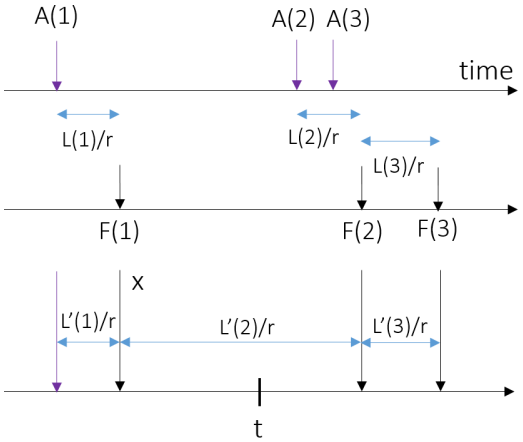

그림 1. 임의의 시간 t가 하나의 플로우가 분할한 시간 구간 (F(2)-L'(2)/r, F(2)]에 속한다.
Fig. 1. A random time t belongs to a interval (F(2)-L'(2)/r, F(2)] in the partitioned time by a single flow.

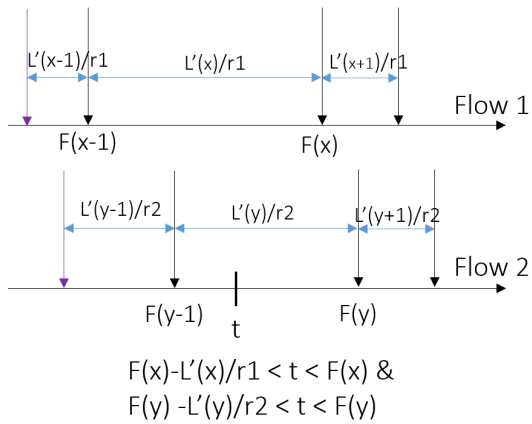

그림 2. 임의의 시간 t가 두 개의 플로우가 각각 분할한 시간 구간 (F(x)-L'(x)/r, F(x)]과 (F(y)-L'(y)/r, F(y)]에 속한다.
Fig. 2. A random time t belongs to intervals (F(x)-L'(x)/r, F(x)] and (F(y)-L'(y)/r, F(y)] in the partitioned time by two flows, respectively.

고 생각될 것이다. 하지만 플로우가 두 개 이상이면 이것이 강력한 추론 방식임을 알게 된다.

특정 시점에서 특정 링크를 공유하고 있는 플로우들의 총 점유율도 비슷한 방법으로 추론할 수 있다. 다만 이 경우, 특정 패킷 p가 어떤 플로우에 속하는지 알 수 없다는 점을 고려하여야 한다. 예를 들면 그림 2에서 두 개의 플로우가 공존하는 경우에 총 점유율 TR (Total rate)을 추론해 보자.

먼저, 추정기준시각 t를 결정하고, 패킷의 완료시간 F(p)와 F(p)-L'(p)/r(p)를 t와 비교한다. r(p)는 p가 속한 플로우의 점유율이다. 아래 부등식 (6)을 만족하는 패킷 p의 r(p)를 추출해서 모두 더한 값이 TR이다.

$$F(p) - L'(p)/r < t \le F(p) \qquad (6)$$

(6)을 만족하는 패킷 p가 추정기준시각 t를 "감싼다"고 하자. 본 논문은 추정기준시각을 감싸는 완료시간을 가지는 패킷이 플로우에 하나만 존재한다는 점에 착안하였다. 이러한 추론을 위해 메타 데이터로 F(p), L'(p), r(p) 값이 필요하다.

추정기준시각 t를 감싸는 패킷 p들을 찾기 시작하는 시각을 추정시작시각이라 하자. 마찬가지로 감싸는 패킷 p들을 찾는 것을 멈추는 시각을 추정종료시각이라고 하자. 시작시각과 종료시각을 어떻게 정해야 t를 감싸는 모든 p를 찾을 수 있을까?

먼저, 추정기준시각은 추정시작시각보다 충분히 먼 이후의 시각으로 결정한다. 예를 들어, 추정시작시각을 t*라고 하면, 추정기준시각 t는 (t* + Ts)로 결정한다. 이때, 추정기준시각을 감싸는 패킷들 모두를 검사해서 추출할 수 있는 Ts가 존재한다. 즉, $A_h(p) < t^*$인 임의의 패킷 p는 $F_h(p) < t^*+Ts = t$ 이어서 (6)을 만족하지 않는 Ts가 존재한다. 아래와 같이 증명될 수 있다.

Theorem 1. 추정기준시각을 t라 하고 추정시작시각을 t*라 하자. t* = t-Ts,

$$Ts = \max_p\{Ba(p)/r(p)\} + DF$$

로 설정하면 t*보다 이전에 인입한 패킷들이 (6)을 만족하지 않는다. 여기서 DF는 임의의 패킷 p와 노드 h에 대해서 Delay factor $d_h(p)$의 0부터 h-1까지의 합의 상한이다. 즉 $\sum_{k=0}^{h-1} d_k(p) \le DF$이다.

Proof: 이 경우 $F_h(p) \le F_0(p) + DF$ 이다. [8]의 Lemma 2에 의해 $F_0(p) = Ba(p)/r(p) + A_0(a)$이다. 여기서 a는 p가 속한 active period의 activation packet으로 정의된다. Ba(p)는 a에서부터 p까지 패킷 길이의 합이다. a는 p에 항상 선행하거나 혹은 둘은 같은 패킷이다 ($a \le p$). 따라서 $F_h(p) \le Ba(p)/r(p) + A_0(a) + DF \le Ba(p)/r(p) + A0(p) + DF \le Ba(p)/r(p) + A_h(p) + DF$이다. 임의의 p가 $A_h(p) < t^*$를 만족하면, $F_h(p) \le t^* + Ba(p)/r(p) + DF$를 만족한다. 따라서

$$Ts = \max_p\{Ba(p)/r(p)\} + DF$$

로 설정함으로써 t*보다 이전에 인입한 패킷들이 (6)을 만족하지 않는다는 것을 보장할 수 있다. ■

일반적으로 Ba(p)의 최대치는 p가 속한 플로우의 최대 버스트 크기이다.

한편, 패킷 추정 검사의 종료시각도 추정기준시각보다 충분히 먼 이후의 시각으로 결정한다. 예를 들어 추정종료시각을 (t + Te)로 결정하면, 종료시각인 t+Te 이후에 인입하는 패킷들은 (6)을 만족하지 않는다. 이는 다음과 같이 증명된다.

Theorem 2. 추정기준시각을 t라 하고 추정종료시각을 t*라 하자. t* = t+Te,

$$Te = \max_p\{L'(p)/r(p)\}$$

로 설정하면 t*보다 이후에 인입한 패킷들은 (6)을 만족하지 않는다.

Proof: [8]의 Theorem 2에 의해 $C_h(p)<F_h(p)$이다. $C_h(p)$는 노드 h에서의 패킷 p의 서비스가 실제로 완료되는 시점이다. 따라서 $A_h(p) < C_h(p) < F_h(p)$이다. 노드 h로 (t+Te)에 인입하는 패킷을 p*라고 하면 $A_h(p^*) < F_h(p^*)$이다. 따라서 t+Te = $A_h(p^*) < F_h(p^*)$이며 $Te = \max_p\{L'(p)/r(p)\}$로 설정하면, p*를 포함하여 이후 인입하는 모든 패킷의 $F_h(p)$값은 t + L'(p)/r(p)보다 크다. 이들은 (6)을 만족하지 않는다. ■

본 논문에서 제안한 방안을 구현을 위해 크게 세 가지의 구성 요소가 필요하다. 첫번째는 패킷에 메타 데이터 정보를 기록하고 update하며 이를 바탕으로 패킷을 서비스하는 기능 블록, 두번째는 이를 바탕으로 링크의 총 점유율을 추론하는 기능 블록이며, 마지막은 추정기간과 추정기준시각을 결정하는 블록이다. 이들 블록의 동작 흐름은 그림 3과 같다.

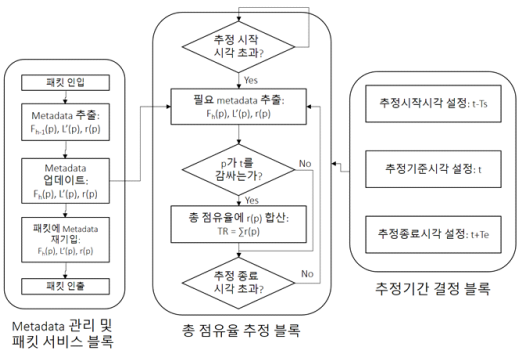

그림 3. 총 점유율 (TR) 상한값 추론 방법 흐름도
Fig. 3. Flowchart for estimation of total link service rate (TR)

Ⅳ. 유효성 검증

본 절에서는 시뮬레이션을 통해 Ⅲ장에서 제안한 방안이 실제 의도한 대로 동작하는지 검증한다.

4.1 알고리즘

Ts와 Te는 플로우들 중 최고값을 사용한다. 이는 추정기준시각과 추정종료시각을 통일하기 위함이지만 실제 구현에서 추정기준시각은 통일하기 어렵다. Ts가 추정 기간 도중 변하면 원래 서비스되던 플로우의 추정기준시각이 변하기 때문이다. 이로 인해 동일한 플로우가 중복해서 링크 총 점유율에 반영될 수 있고 이렇게 추정 링크 총 점유율이 실제보다 큰 경우를 overestimation이라고 한다. 이와 반대로 추정한 링크 총 점유율이 실제보다 작은 경우를 underestimation이라 한다. 이러한 overestimation 문제를 고려해 추정 기간에 새로 들어온 플로우는 다른 추정기준시각을 갖는다. 추정종료시각은 추정 기간 중에 변하여도 상관없으므로 추정종료시각은 모든 플로우가 동일한 값을 가진다.

링크 총 점유율의 추정값은 추정종료시각에 얻을 수 있는 값이기에 해당 추정 기간에는 이전에 얻은 링크 총 점유율 추정값을 사용하여 인입제어를 수행한다. 때문에 추정 기간에 새로운 플로우가 들어오거나 플로우가 서비스를 종료하면 underestimation 문제나 overestimation 문제가 발생할 수 있다. 이러한 문제를 완전히 해결할 수는 없으나 영향을 줄이기 위해 추정기간을 감소시키도록 일정 간격마다 (7)에 따라 Te를 갱신한다.

$$Te_{new} = 0.9 \times Te_{old} + 0.1 \times max\{Te_{observed}\} \quad (7)$$

Ts도 동일한 방법으로 감소시킨다.

플로우별 최초 패킷의 경우 이전 패킷이 없기에 L'(p)/r(p) 값이 없다. 따라서 최초 패킷에 추정기준시각을 감쌀 수 있는 충분한 임의의 L'(p)/r(p)를 할당한다. 다만, 이렇게 할당된 L'(p)/r(p)로 인해 추정 기간이 길어질 수 있으므로 최초 패킷의 L'(p)/r(p)는 Te에 반영하지 않도록 한다.

모든 플로우는 그림 4처럼 마르코프 on-off 모델을 따라서 on 상태에서 버스트 트래픽을 생성한다. 마르코

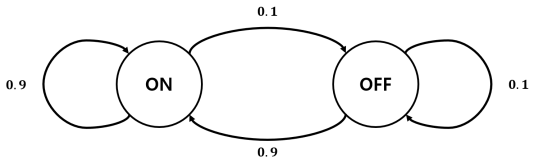

그림 4. 패킷 생성에 사용한 마르코프 on-off 모델[8]
Fig. 4. Markov on-off model used to generate packets[8]

프 on-off 모델에서 상태 전이는 지수 분포를 따라 정해진 시간마다 발생한다. 생성된 패킷은 토큰 버킷을 따라 서비스된다. 이는 순간적으로 플로우의 Tspec을 넘기는 경우를 막기 위함이다. 토큰 버킷은 패킷 생성에 포함된 단계로 이로 인한 지연 시간은 고려하지 않는다.

플로우의 버스트 트래픽 생성이 지수 분포를 따르므로 패킷 생성 간격은 무작위로 정해진다. 다시 말해, L'(p)/r(p)가 무작위로 결정된다. L'(p)/r(p)가 무작위로 결정되면 추정기준시각을 감싸는 패킷이 서비스된다는 보장이 없어 underestimation 문제가 발생할 수 있다. 이 문제를 해결하기 위해 2가지 접근 방법을 제시한다. 먼저 충분히 긴 임의의 시간을 Te의 최소값으로 정하여 추정종료시각을 키우는 방법이다. 이 방법은 underestimation 문제를 완전히 해결하지 못하고 service rate가 낮은 플로우에 대해서는 효과가 없지만 추가 구현 없이 underestimation 문제를 완화할 수 있다.

다른 방법으로 플로우가 더미 패킷을 서비스하여 플로우별 Te의 최대값을 고정하는 방법이 있다. 플로우별 Te의 최대값은 max{Ba(p)/r(p)}로 고정한다. max{Ba(p)/r(p)}는 토큰 버킷에서 토큰이 최대로 쌓이는데 걸리는 시간과 동일하다. 플로우가 시간이 max{Ba(p)/r(p)}만큼 지나도 패킷을 서비스하지 않았으면 더미 패킷을 서비스한다. 해당 방법은 추정기준시각을 감싸는 패킷이 반드시 서비스되기에 L'(p)/r(p)의 무작위성으로 인한 underestimation을 완벽히 해결할 수 있다. 다만, 더미 패킷을 사용하기 위해서는 더미 패킷이 완료시간 계산에 영향을 주지 않아야 한다. 더미 패킷이 완료시간 계산에 영향을 주게 되면 완료시간이 원래 계산 결과보다 증가하게 되어 추정기준시각을 감싸는 패킷이 없을 수 있다. 더미 패킷이 완료시간에 주는 영향을 없애기 위해 더미 패킷의 완료시간 계산 방법이 달라져야 한다. 더미 패킷의 최초 노드에서 완료시간 계산 식은 (8)과 같다.

$$F_0(p) = \max\{F_0(p-1), A_0(p)\} \quad (8)$$

이후 노드에서는 기존과 동일한 방식을 사용한다. 이 방법은 플로우가 더미 패킷을 생성해야 하고, 더미 패킷을 구분해 완료시간 계산을 다르게 해야 하기에 구현이 상대적으로 복잡하다.

알고리즘 1은 시뮬레이션에 사용한 링크 총 점유율 추정 알고리즘이다. 링크 총 점유율은 total_rate, 추정 시각은 t_start, t, t_end로 표시하였다. Ts_next는 Ts의 갱신이 바로 반영되지 않고 다음 추정 기간에 반영될

알고리즘 1. 링크 총 점유율 추정 알고리즘

```
Input: p: packet
       Te_upper_bound: 플로우별 최초 패킷을 구분하기 위해 사용하는 큰 수
       Te_lower_bound: Te의 하한값으로 임의의 값을 사용, 더미 패킷을 사용하면 값은 0
Ts←0, Ts_next←0, Ts_observed←0
Te←0, Te_observed←0
t_start←0, t←0, t_end←0, t_update←0
total_rate←0, R←0
while true do
    if packet p is received then
        b←max{Ba(p)/r(p)}+DF
        Ts_next←max(Ts_next, b)
        Ts_observed←max(Ts_observed, b)
        if L'(p)/r(p)<Te_upper_bound then
            Te←max(Te, L'(p)/r(p))
            Te_observed←max(Te_observed, L'(p)/r(p))
        end
        if t_end==0 then
            t_start←current time
            Ts←Ts_next
        else if t_end<current time then
            total_rate←R
            R←0
            t_start←t_end
            Ts←Ts_next
        end
        if t_update<current time then
            Te←max(0.9×Te+0.1×Te_observed, Te_lower_bound)
            Ts_next←0.9×Ts_next+0.1×Ts_observed
            Te_observed←0
            Ts_observed←0
            t_update←t_update+1second
        end
        t←t_start+max(Ts, b)
        t_end_←t+max(Te, Ts_next)
        t_end←max(t_end, t_end_)
        if F(p)-L'(p)/r(p)<t<F(p) then
            R←R+r(p)
        end
    end
end
```

수 있도록 추가한 변수이다. Ts가 바로 갱신되면 하나의 플로우가 여러 추정기준시각을 갖게 되어 overestimation 문제가 발생할 수 있기 때문이다. t_update는 Ts와 Te를 감소하는 최소 간격이다. 줄 1-4는 변수들의 초기화 단계이다. 줄 6에서 알 수 있듯 링크 총 점유율 추정은 패킷이 들어오는 순간에 수행한다. 줄 7-13은 각 변수들을 갱신한다. 줄 14-16은 최초로 패킷이 들어왔을 때 추정시작시각과 Ts를 초기화한다. 줄 17-22는 추정이 종료되었을 때 링크 총 점유율과 추정시작시각, Ts를 갱신한다. 줄 23-29는 Te와 Ts_next를 감소시킨다. Ts가 아니라 Ts_next를 감소시켜야 동일한 플로우의 추정기준시각을 일정히 유지할 수 있다. 감소 주기는 최소 1초로 정하였다. 줄 30-32는 추정기준시각과 추정종료시각을 계산한다. Te의 최대값은 클수록 미래에 들어올 패킷을 고려할 수 있으므로 Te와 Ts중 큰 값으로 정하였다. 추정종료시각은 최대값을 사용한다. 줄 33-35는 패킷이 추정기준시각을 감싸는지 확인하고 감싼다면 변수 R에 패킷이 속한 플로우의 service rate를 더한다.

4.2 시뮬레이션

시뮬레이션은 OMNeT++[11]로 구현하였다. 시뮬레이션에 사용한 토폴로지는 그림 5와 같이 dumbbell 토폴로지를 사용하였다. 모든 링크의 용량은 100Mbps이다. 가장 많은 플로우가 지나는 switch0에서 추정 링크 총 점유율을 관찰하였다. 스케줄러로 host에서는 FIFO를 사용하고 스위치에서는 C-SCORE를 사용하였다. 시뮬레이션에서 사용한 플로우의 특성은 표 2와 같다. 모든 플로우의 패킷 길이는 1000 bytes, 최대 버스트 크기는 10000 bytes이다. 시뮬레이션에서 service rate가 높은 플로우부터 서비스를 시작해서 점점 service rate가 낮은 플로우를 추가하였다. L'(p)/r(p)를 점점 커지게 하여 L'(p)/r(p)의 무작위성으로 인한

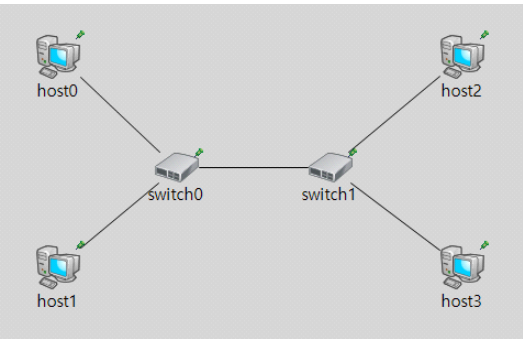

그림 5. 시뮬레이션 토폴로지
Fig. 5. Simulation topology

표 2. 시뮬레이션에서 사용한 플로우들의 특성
Table 2. Characteristics of flows used in simulation

Flow	Source, Destination	service rate	Service period
Flow 1	host0, host2	4Mbps	0-60 sec
Flow 2	host0, host2	2Mbps	10-40 sec
Flow 3	host1, host3	1Mbps	20-50 sec
Flow 4	host1, host3	50Kbps	25-35 sec

underestimation 문제와 추정 기간의 증가로 인한 overestimation과 underestimation 문제를 관찰하기 위함이다. 시뮬레이션은 100번 반복하였다.

그림 6은 더미 패킷을 사용하지 않고 Te의 최소값도 설정하지 않았을 때의 링크 총 점유율 추정 결과이다. 0초에서 25초까지 L'(p)/r(p)의 무작위성으로 인한 underestimation 문제가 관찰된다. 25초가 되면 이러한 underestimation 문제가 사라지는데 Flow 4의 service rate가 낮아 추정 기간이 길어졌고 이보다 긴 L'(p)/r(p)가 관측되지 않았기 때문으로 보인다. Flow 4의 서비스가 종료된 이후에도 Flow 4로 인해 추정 기간이 길어졌으므로 underestimation 문제가 발생하지 않았다. 긴 추정 기간으로 인한 underestimation 문제와 overestimation 문제는 25초부터 관찰된다. Flow 4의 경우 25초에 서비스를 시작했으나 28초 즈음에 링크 총 점유율에 반영이 되었다. Flow 4의 service rate는 50Kbps로 Ts는 1.6+DF이다. Te의 최대값은 Ts를 사용해 구할 수도 있으므로 총 추정 기간은 적어도 3.2초가 된다. 해당 기간에는 Flow 4가 서비스되고 있음에도

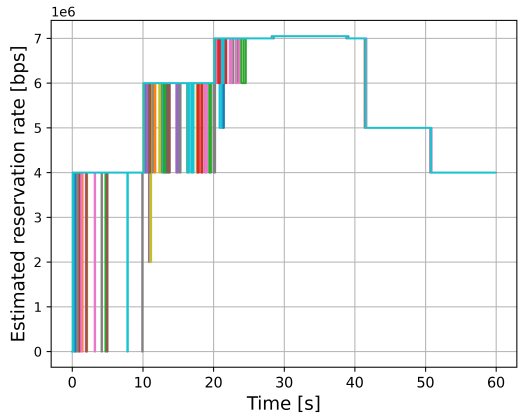

그림 6. 더미 패킷을 사용하지 않고 Te의 최소값이 0인 링크 총 점유율 추정 알고리즘 시뮬레이션 결과
Fig. 6. Simulation results of the link total reservation rate estimation algorithm without using dummy packets and with the minimum value of Te being 0

추정이 종료되지 않아 이전에 구한 링크 총 점유율을 사용하였기에 underestimation 문제가 발생한다. 마찬가지로 Flow 4는 35초에 서비스를 종료했으나 추정 링크 총 점유율에는 39초에 반영되어 긴 추정 기간으로 인한 overestimation이 발생한다. 다만 해당 overestimation 구간은 아직 서비스되지 않은 패킷이 있을 수 있기에 해당 구간 전체에서 overestimation 문제가 발생하지는 않는다. Flow 4로 인해 길어진 추정 기간은 이후 링크 총 점유율 추정에서 overestimation 문제를 일으킨다. 40초에서 Flow 2가 서비스 종료되었으나 추정 링크 총 점유율에는 2초 정도 늦게 반영되었고 50초에서 Flow 3이 서비스 종료되었으나 추정 링크 총 점유율에는 1초 정도 늦게 반영되었다. Ts와 Te 감소 알고리즘을 통해 시간이 지날수록 overestimation 구간이 감소하는 모습을 볼 수 있다.

그림 7은 더미 패킷을 사용하지 않고 Te의 최소값을 100ms로 정했을 때의 링크 총 점유율 추정 결과이다. 10초부터 25초까지 L'(p)/r(p)의 무작위성으로 인한 underestimation 문제가 관찰된다. 이는 100ms가 Flow 1의 추정 기간을 감싸기에는 충분했으나 Flow 2와 Flow 3의 추정 기간을 감싸기에는 크기가 충분하지 않았음을 의미한다. 그림 6과 비교했을 때 0초에서 10초 구간은 underestimation 문제가 발생하지 않았고 10초에서 25초 구간은 underestimation 문제가 발생하는 건수가 감소했음을 알 수 있다.

그림 8은 더미 패킷을 사용하는 방법으로 얻은 링크 총 점유율 추정 결과이다. 시뮬레이션 간 결과 차이가 거의 없어 하나의 그래프처럼 보인다. 결과 차이가 보이지 않는 이유는 시뮬레이션 간에 플로우의 패킷 생성 패턴이 다르더라도 더미 패킷으로 인해 비슷한 시각에 동일한 링크 총 점유율을 추정하기 때문이다. L'(p)/r(p)의 무작위성으로 인한 underestimation 문제는 관찰되지 않으나 긴 추정 기간으로 인한 underestimation 문제와 overestimation 문제는 관찰된다.

그림 9는 더미 패킷 방식과 [2]에서 제안한 추정 방법과의 비교 결과이다. [2]에서도 더미 패킷을 사용하기 때문에 더미 패킷 방식과 결과를 비교하였다. [2]의 방식은 set-up 메시지로 수락 제어를 하는 단계, 메타 데이터 b(p)를 모으는 단계, 일정 기간 T마다 추정 reservation rate를 계산하고 출력하는 단계로 이루어진다. [2]는 set-up 메시지 전송단계가 있기 때문에 더미

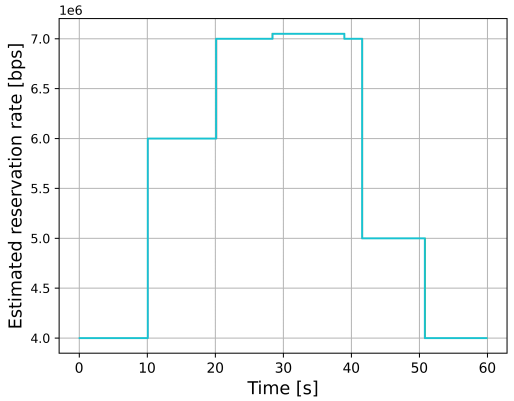

그림 8. 더미 패킷을 사용한 링크 총 점유율 추정 알고리즘 시뮬레이션 결과
Fig. 8. Simulation results of link total reservation rate estimation algorithm using dummy packets

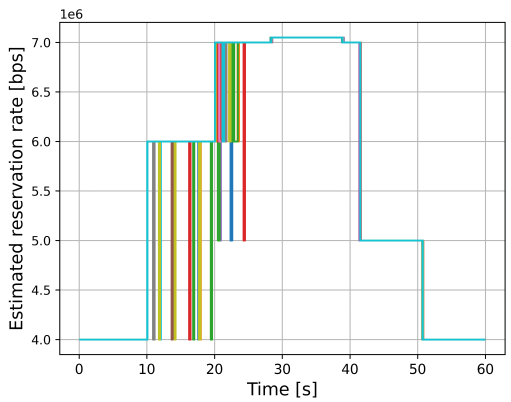

그림 7. Te의 최소값을 100ms로 정한 링크 총 점유율 추정 알고리즘 시뮬레이션 결과
Fig. 7. Simulation results of the link total reservation rate estimation algorithm with the minimum value of Te set of 100ms

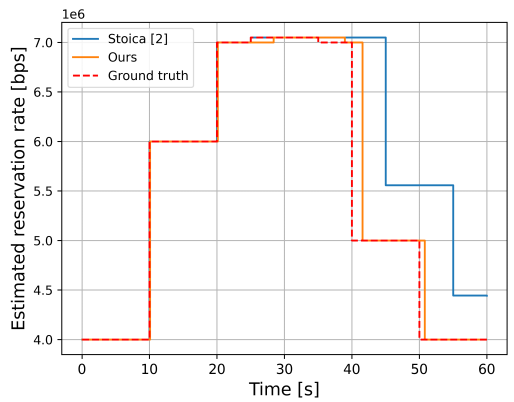

그림 9. 제안하는 더미 패킷을 사용하는 방식과 [2]의 링크 총 점유율 추정 방식 결과 비교
Fig. 9. Comparison of the results between the proposed method using dummy packets and the method of estimating the link total reservation rate in [2]

패킷 방식과 달리 underestimation 문제가 없다. 하지만 일정 시간마다 reservation rate를 추정하고, underestimation 문제를 방지하기 위해 추정값을 실제 값보다 크게 계산하기에 overestimation 문제가 더미 패킷 방식보다 크게 나타난다. 또한 [2]의 방법은, 일정 시간 구간 T, 지터의 최대값, 최대 패킷 전송 간격을 미리 정하거나 알고 있어야 사용이 가능하다는 한계가 있다. 더미 패킷 방식은 underestimation 문제가 있으나, set-up 메시지 전송 없이 총 점유율 추정이 가능하며 overestimation 문제에 상대적으로 강인하다.

V. 결 론

본 논문에서는, 최근 제안되어 IETF 표준 채택을 시도하고 있는 C-SCORE[8-10] 기술을 기반으로 하여, 플로우의 상태 정보 관리 없이 인입 제어를 할 수 있도록 링크의 총 점유율을 추론하는 방안을 제시하였다. 기존 기술 [2]는 최초 노드에서는 정확한 값을 추론할 수 있지만, 이후 노드에서는 패킷들의 도착시간 간격이 최초 노드에서의 도착시간 간격과 다르기 때문에, 패킷 지연 시간의 최대 변화량에 비례하여 오류가 발생한다.

C-SCORE에서는 패킷의 최초 노드에서 도출한 "완료시간" F(p)과 이전 패킷의 완료시간 F(p-1)의 차이가, 이후의 모든 노드에서 동일하다. 따라서, 추론하고자 하는 시각을 감싸는 완료시간을 가지는 패킷이 플로우에 하나만 존재한다. 이를 바탕으로 점유율을 정확하게 추정할 수 있으며 이러한 방안을 본 논문에서 제시하였다. 또한, 추정 기간 대신 추정시각으로 간단히 추정하기 때문에 복잡도도 낮으며, 추정 기간동안 새롭게 허락되어 신규로 패킷을 생성하는 플로우들에 대한 보상이 필요 없다. 이에 더하여, 특정 시각에 활성화되어 있는 플로우들의 수와 각각의 점유율을 알 수 있다.

본 논문에서 제시한 방안의 유효성을 시뮬레이션을 통해 검증하였다. 시뮬레이션은 서로 다른 service rate를 갖는 플로우 4개가 각기 다른 시간에 추가, 제거되는 시나리오를 dumbbell 토폴로지에서 구현하였다. 플로우는 무작위로 버스트 트래픽을 생성하므로 underestimation 문제가 발생할 수 있다. 이에 Te의 최소값을 정하여 추정종료시각을 키우는 방법과 플로우가 더미 패킷을 생성하여 플로우별 L'(p)/r(p)의 상한을 제한하는 방법을 제시하였다. 시뮬레이션 결과 Te의 최소값을 정하는 방법은 아무것도 적용하지 않았을 때와 비교하여 L'(p)/r(p)의 무작위성으로 인한 underestimation 문제가 완화되었다. 더미 패킷을 사용한 방법은 L'(p)/r(p)의 무작위성으로 인한 underestimation 문제가 발생하지 않았다. 두 방법 모두 긴 추정 기간으로 인한 underestimation 문제와 overestimation 문제는 발생하였다. 더미 패킷을 사용한 방법은 [2]의 방법과 비교하여 overestimation 문제에 강인하였고 set-up 메시지 없이도 총 점유율 추정이 가능함을 보였다.

해당 방안은 C-SCORE의 최신 버전[10]에 삽입되어 IETF 118에서 발표되었다.

References

[1] Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification, IETF RFC 2205, Sep. 1997, Retrieved Nov., 22, 2023, from https://www.rfc-editor.org/info/rfc2205

[2] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 81-94, 1999. (https://doi.org/10.1145/316194.316208)

[3] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344-357, 1993. (https://doi.org/10.1109/90.234856)

[4] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," in *Proc. ACM Symp. Commun. Architectures & Protocols*, pp. 19-29, Aug. 1990. (https://doi.org/10.1145/99517.99525)

[5] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. INFOCOM'94 Conf. Comput. Commun.*, pp. 636-646, 1994. (https://doi.org/10.1109/INFCOM.1994.337677)

[6] D. Stiliadis and A. Varma, "Efficient fair queueing algorithms for packet-switched networks," *IEEE/ACM Trans. Netw.*, vol. 6, no. 2, pp. 175-185, 1998. (https://doi.org/10.1109/90.664266)

[7] *IETF Deterministic Networking (detnet) Working Group Home page*, Retrieved Nov, 22, 2023,

from https://datatracker.ietf.org/wg/detnet/about/

[8] J. Joung, J. Kwon, J.-D. Ryoo, and T. Cheung, "Scalable flow isolation with work conserving stateless core fair queuing for deterministic networking" *IEEE Access*, vol. 11, Sep. 2023. (https://doi.org/10.1109/ACCESS.2023.3318479)

[9] J. Joung, "Work conserving scheduler with global finish time for network latency guarantee," *J. KICS*, vol. 48, no. 1, pp. 55-64, 2023. (https://doi.org/10.7840/kics.2023.48.1.55)

[10] J. Joung, J-D. Ryoo, T-S. Cheung, Y. Li, and P. Liu, "Latency guarantee with stateless fair queuing," *IETF DetNet WG*, Internet draft, draft-joung-detnet-stateless-fair-queuing-01, Nov. 2023.

[11] *OMNeT++*, Retrieved Apr., 8, 2024, from https://omnetpp.org/

정 진 우 (Jinoo Joung)

1992년 2월 : KAIST 전자공학과 학사
1994 8월 : NYU 전기전자공학과 Master
1997년 8월 : NYU 전기전자공학과 Ph.D.
1997년 10월~2005년 2월 : 삼성전자 종합기술원
2005년 3월~현재 : 상명대학교 휴먼지능정보공학과 교수

<관심분야> 유무선통신, 네트워크, 임베디드시스템

[ORCID:0000-0003-3053-9691]

권 주 혁 (Juhyeok Kwon)

2020년 8월 : 상명대학교 휴먼지능정보공학과 학사
2020년 9월~2022년 8월 : 상명대학교 지능정보공학과 석사
2022년 9월~현재 : 상명대학교 지능정보공학과 박사과정

<관심분야> 유무선통신, 네트워크, 임베디드 시스템