논문 21-46-02-13 The Journal of Korean Institute of Communications and Information Sciences '21-02 Vol.46 No.02
https://doi.org/10.7840/kics.2021.46.2.293

지연시간의 통계적 분석을 통한 FAIR 프레임워크의 타당성 연구

권 주 혁*, 정 진 우°

Study on Feasibility of the FAIR Framework Based on Delay Statistics Analysis

Juhyeok Kwon*, Jinoo Joung°

요 약

인터넷의 실질적인 품질보장 솔루션인 Differentiated Services (DiffServ)에서 채택한 class 기반 strict priority 스케줄링은 cycle이 존재하는 토폴로지의 네트워크에서 지연시간을 보장하지 못한다. 이에 대한 해결책으로 interleaved regulator (IR) 기반의 asynchronous traffic shaping (ATS) 기술이 제시되었다. IR이 지연시간 최대치를 증가시키지 않는다는 ATS의 핵심 이론을 확장한 flow aggregate-interleaved regulator (FAIR) 프레임워크도 최근 제시되었다. FAIR 프레임워크는 ATS보다 복잡도와 지연시간 최대치 측면에서 우수한 것이 수학적으로 증명되었지만 ATS와 FAIR 프레임워크에서 사용되는 레귤레이터의 비작업보존 특성으로 인해 평균 지연시간이 커질 것이라는 우려가 제기되어 왔다. 본 연구에서는 ATS와 FAIR 프레임워크의 통계적 성능 특성을 시뮬레이션을 통해 분석하고 레귤레이터가 없는 FIFO 스케줄링 프레임워크와 비교하였다. Cycle이 없는 네트워크에서의 평균 지연시간은 FIFO, FAIR, ATS 프레임워크 순서로 좋으며 FIFO와 FAIR 프레임워크의 차이는 크지 않았다. Cycle이 있는 소규모 네트워크에서의 평균 지연시간은 FAIR, FIFO, ATS 프레임워크 순서로 좋았다. FAIR 프레임워크는 이로써 지연시간 최대치와 평균 지연시간의 측면에서 모두 ATS보다 월등하다는 것이 증명되었다.

Key Words : End-to-end delay guarantee, TSN, scheduler, flow aggregate, interleaved regulator

ABSTRACT

The class-based strict priority scheduling in Differentiated Services (DiffServ), which is adopted in the Internet, does not guarantee delay in a network with a topology where cycles exist. Asynchronous traffic shaping (ATS) technology, with the key theory that interleaved regulator (IR) does not increase the delay upper bound, was proposed as the solution to this problem. Based on the extension of the key theory of ATS, the flow aggregate-interleaved regulator (FAIR) framework is also proposed. It has been proven that FAIR framework is superior to the ATS in terms of complexity and delay bound. In this study, statistical performance characteristics of the ATS and FAIR frameworks are analyzed with simulations and compared with a FIFO scheduling framework without regulators. In the simulation without cycle, statistical delay performance was in the order of FIFO, FAIR, and ATS. In the simulation with cycle, statistical delay performance was in the order of FAIR, FIFO, and ATS. This proved that the FAIR framework is superior to the ATS in terms of both delay bound and average delay.

※ 본 연구는 상명대학교 교내연구비를 지원받아 수행하였음.
◆ First Author : 상명대학교 컴퓨터과학과, Sangmyung University, Department of Computer Science, 학생회원
° Corresponding Author : 상명대학교 휴먼지능정보공학과, Sangmyung University, Department of Human-centered AI, jjoung@smu.ac.kr, 정회원
논문번호 : 202010-248-B-RN, Received October 5, 2020; Revised December 5, 2020; Accepted December 7, 2020

Ⅰ. 서 론

스마트 팩토리, 차량 간 통신, Tactile Internet, 대규모 전력 제어망 등 다양한 응용 분야에서 단대단 네트워크 지연시간 (end-to-end network delay)에 대하여 수 msec에서 수초까지의 엄격한 제한을 요구하고 있다. 하지만 인터넷의 실질적인 품질보장 솔루션인 Differentiated Services (DiffServ)에서 채택한 class 기반 strict priority 스케줄링은 cycle이 존재하는 토폴로지의 네트워크에서 지연시간을 보장하지 못한다. IEEE 802.1 time sensitive network (TSN)[1] TG에서 제시된 asynchronous traffic shaping (ATS)[2] 프레임워크는 interleaved regulator (IR)를 채택하여 이러한 DiffServ의 약점을 보완하였다. ATS 프레임워크는 입력포트별, 클래스별 트래픽 레귤레이터를 출력 포트의 시작점에 구현한다. 이러한 regulation 기능을 적절히 활용하면 FIFO 스케줄러를 통과하면서 플로우가 통합 분리를 반복함에 따라 발생하는 burst 축적 현상을 방지할 수 있다[3]. 한편, IR을 단위 네트워크의 edge에 배치하고 단위 네트워크 안에서는 port별 통합 플로우 (flow aggregate, FA)기반 스케줄링을 함으로써 대규모 네트워크에서 낮은 복잡도로 지연시간의 최대치를 보장하는 방안도 제시되었으며[4] 이를 FAIR (flow aggregate - interleaved regulator) 프레임워크라 한다. 해당 FAIR 프레임워크에서는 ATS에서 모든 노드의 출력포트에 IR이 구현되어야 하는 것에 비해 네트워크의 edge에만 IR이 필요하다. 이에 따라 FAIR 프레임워크가 대규모 네트워크에서 낮은 복잡도로 ATS와 기존 integrated services (IntServ) 프레임워크보다 더 작은 지연시간의 최대치를 보장해 준다는 것이 증명되었으며[4], 이를 바탕으로 ITU-T SG13에서 FAIR 프레임워크가 5G와 6G 네트워크의 핵심 기술 표준으로로 채택되었다[5]. 하지만 ATS와 FAIR에서의 레귤레이터의 비작업보존 특성으로 인해 평균 지연시간이 커질 것이라는 우려가 꾸준히 제기되어 왔으며 이러한 점이 레귤레이터 기반의 프레임워크가 시장에서 받아들여지는데 결정적인 난관으로 작용하고 있다.

본 연구에서는 ATS와 FAIR 프레임워크의 통계적 성능 특성을 시뮬레이션을 통해 분석하고 레귤레이터가 없는 FIFO 스케줄링 시스템과 비교하였다. 이를 통해 FAIR 프레임워크의 통계적 성능이 ATS보다 우수함을 보였다. FAIR 프레임워크는 이로써 지연시간 최대치와 평균 지연시간의 측면에서 모두 IEEE 표준 기술인 ATS보다 월등하다는 것을 증명하였다. 더욱이 FAIR 프레임워크의 통계적 성능이 현재의 인터넷과 비슷하다는 것을 보임으로써 복잡성, 최대 지연시간, 및 평균 지연시간의 모든 측면에서 우수하며, 실제 미래 네트워크에서 적용될 수 있는 프레임워크임을 증명하였다.

Ⅱ. 관련 연구

네트워크에서 단대단 (end-to-end) 지연시간의 최대치(upper bound)를 보장하기 위해서 다음과 같은 세 가지 조건이 만족되어야 한다.

조건 1) 인입하는 모든 플로우들의 평균 인입 속도 (average arrival rate)와 최대버스트 크기 (max burst size) 제한.

조건 2) 모든 노드의 출력포트에서 출력되는 플로우들의 인입 속도 총합이 해당 포트의 용량 (capacity) 이하.

조건 3) 모든 노드의 출력포트에서 모든 플로우들에게 해당 플로우의 평균 인입 속도 이상 서비스 제공 보장.

여기서 플로우는 발신지와 목적지가 동일하고, 같은 어플리케이션이 생성하였으며, 생성시간이 인접한 패킷들의 집합이다. 트래픽 제어 대상의 최소단위라고 할 수 있다. 조건 1과 2를 만족시키기 위해서 인입제어와 자원예약 기능이 필요하며, 조건 3을 만족하기 위해서 플로우 기반의 스케줄러가 필요하다. 특히 조건 3을 만족하는 스케줄러들을 LR(Latency-Rate) 서버라고도 한다[6]. 이러한 세 가지 조건과 이를 만족시키는 기능을 포괄적으로 규정하는 프레임워크가, 잘 알려진 Integrated Services(IntServ)이다. 하지만 IntServ의 최대 약점은 조건 3을 만족시키기 위한 스케줄러의 복잡성이다. Core 네트워크에서 플로우의 수는 수백만 개 수준이라고 알려져 있다. 이들을 독립된 큐에 저장하고 서비스 순서를 실시간으로 정하는 일은 현재의 ASIC 기술로도 불가능한 일이다. 따라서 다양한 application들에 대해서 상대적인 성능 차별을 제공하면서도 간단한 구조로 구현 가능한 Differentiated Services (DiffServ) 프레임워크가 각광을 받아 현재의 인터넷에 채택되었다. DiffServ의 핵심은 모든 패킷을 8개(혹은 32개)의 class로 구별하고, class 별 큐만 유지하면서 상위 우선순위의 큐를 우선적으로 서비스하는 스케줄러이다. 이러한 스케줄러를 strict priority (SP) 스케줄러라고 하는데, 위의 조건 1, 2가 만족되면 SP 스케줄러도 조건 3에 부합하는 LR 서버로 동작한다. 따라서 SP 스케줄러와 상위 우선순

위 트래픽에 대해서 조건 1, 2를 만족해주는 인입 제어기능이 더해지면 상위 우선순위 트래픽에 대해서 지연시간 최대치가 보장될 수도 있다. 하지만 SP 스케줄러의 지연시간이 같은 큐에 속한 플로우들의 최대 버스트 크기에 비례한다는 치명적인 단점이 존재한다. 플로우의 최대버스트가 SP 스케줄러를 통과하면서 같은 큐를 사용하는 다른 플로우들의 최대버스트 크기에 영향을 받아서 노드를 지날수록 기하급수적으로 커진다. 이에 따라 mesh 구조처럼 네트워크에 cycle이 존재하면 플로우의 최대버스트 크기가 feed-forward되어 무한으로 커진다. 결국, DiffServ 기반 현재의 인터넷에서는 지연시간 최대치 보장이 불가능하다.

DiffServ의 구조를 유지하면서 플로우의 최대버스트 크기를 강제로 제한하여 임의의 토폴로지에서 지연시간 최대치를 보장하려는 시도가 있다. IEEE 802.1 TSN 표준에서 제안된 ATS 기술은 interleaved regulator(IR)를 클래스별 SP 스케줄러와 나란히 구현한다. IR은 모든 플로우를 하나의 큐에 저장하고, 큐의 가장 앞(head of queue, HoQ)에 있는 패킷이 소속 플로우의 인입속도와 최대버스트 규정을 동시에 만족하는 시점(eligible time)에 내보낸다. IR은 플로우 별 state 정보를 유지해야 하지만 큐가 하나뿐이어서 플로우 기반 스케줄러와 비교하면 훨씬 낮은 수준의 복잡도를 가진다. 얼핏 HoQ에 있는 패킷 때문에 뒤에 기다리는 다른 플로우의 패킷들이 크게 손해를 입을 듯하다. 하지만 다음의 조건이 만족되면 IR이 SP 스케줄러의 지연시간 최대치를 증가시키지 않는다는 핵심 이론을 제시하고 있다[7].

조건 4) SP 스케줄러는 최고순위 class의 모든 packet들의 FIFO 특성 유지.

조건 5) IR은 모든 플로우에 대해서 SP 스케줄러로의 인입 특성 재현.

조건 6) (Minimal IR) IR은 HoQ의 packet이 eligible 해지는 순간 즉시 전송.

조건 7) IR은 eligible packet들에 대해서 zero 지연시간을 제공 (전송지연 포함, cut-through 필요).

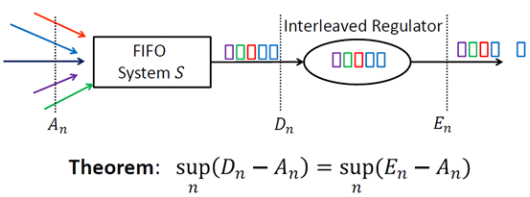

그림 1. [7]의 핵심 이론의 도식화
Fig. 1. Depiction of the key contribution of [7]

이러한 ATS의 최대 단점은 모든 노드의 입출력 포트 쌍마다 존재하는 IR이 평균 지연시간에 막대한 타격을 줄 것이라는 예측이다. 이에 대한 뚜렷한 증거는 없지만 이로 인해 TSN에서 ATS기술의 표준 채택이 미루어지고 있는 실정이다.

FAIR 프레임워크[4]에서는 [7]에서 제시된 이론을 확장하여, 그림 2에서와같이 단위 네트워크의 입출력 포트를 공유하는 모든 플로우를 하나의 통합플로우(flow aggregate, FA)로 묶고, 단위 네트워크 내의 노드에서는 FA단위로 스케줄하여 FIFO 특성을 유지하며, 단위 네트워크의 edge에서 FA별 IR을 통과하게 하여 최초 인입 시의 플로우 특성을 강제한다. 결과적으로 이런 단위 네트워크들이 연결된 대규모 인터네트워크에서도 지연시간이 보장된다.

FAIR 프레임워크는 IntServ와 ATS의 hybrid로 볼 수 있다. 단위 네트워크가 전체 네트워크가 되면 입력 port에서 하나의 플로우가 인입하는 IntServ와 같다. 단위 네트워크가 단일 노드로까지 작아지면 ATS와 유사하다. 이 경우 ATS가 FIFO 스케줄러를 사용하는 데 반해 FAIR는 port 쌍별로 분리된 큐를 LR서버로 처리한다는 것이 다르다. 수치 분석을 통해 homogenous 한 특성을 가진 네트워크에서 단위 네트워크의 크기와 상관없이 FAIR의 지연시간 최대치가 IntServ나 ATS에 비해서 낮은 값을 가진다는 것이 증명되었다. 이것은 FAIR가 IntServ보다 낮은 복잡도를 가지며, ATS보다 IR의 수가 적다는 것을 감안하면 고무적인 결과이다. 이러한 장점을 바탕으로 ITU-T SG13에서 FAIR 프레임워크가 표준으로 채택되었다.[5].

본 연구에서는 ATS와 FAIR 프레임워크의 통계적 성능 특성을 시뮬레이션을 통해 분석하고 레귤레이터가 없는 FIFO 스케줄링 시스템과 비교하였다.

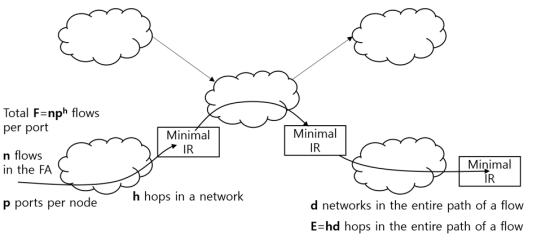

그림 2. FAIR 프레임워크[4]
Fig. 2. FAIR Framework[4]

Ⅲ. 시뮬레이션

본연구의 시뮬레이션 환경은 OMNeT++를 사용해 구축하였다. OMNeT++는 C++ 기반의 네트워크 시뮬레이션 프레임워크로 discrete event network simulation을 구축하는 데 유용하다[8].

3.1 Cycle이 없는 네트워크 토폴로지

Cycle이 없이 symmetric한 네트워크 토폴로지에서의 시뮬레이션이다. 그림 3과 같은 구조를 갖는 네트워크 토폴로지를 바탕으로 FAIR 프레임워크, ATS, FIFO 시스템을 구축하였다. 그림 3에서 원은 2x2 스위치 노드이다.

FAIR 프레임워크에서는, 모든 노드에 통합플로우별 큐와 이를 처리하는 DRR 스케줄러가 있고, 4번째 노드에는 추가로 port마다 통합플로우 별 IR이 있다. 결과적으로 FAIR 프레임워크에서 그림 3의 토폴로지 상에 두 개의 단위 네트워크가 존재하며, 단위 네트워크의 사이에 FA별 IR들이 구현되었다. ATS 프레임워크에서는, 모든 노드에 FIFO 스케줄러가 있고 port마다 input port 별 IR이 있다. FIFO 프레임워크에서는, 모든 노드에 FIFO 스케줄러만 존재한다. FAIR 프레임워크는 단위 네트워크 2개를 통과하도록 구현하였다. 단위 네트워크의 hop 수가 2 이상인 경우부터 ATS와 FAIR 프레임워크 간 구현 측면의 차이가 발생한다. hop 수가 4 이상이면 시뮬레이션 시간이 너무 길어져 단위 네트워크의 hop 수를 3으로 정하였다. 시뮬레이션 네트워크 토폴로지의 파라메터는 표 1과 같이 정리된다.

데이터를 생성하는 source는 그림 4와 같이 마르코프 on-off 모델을 따른다. On 구간에서 source는 nominal input rate으로 데이터를 생성한다. On 구간에서 off 구간으로의 transition rate은 α, off 구간에서 on 구간으로의 transition rate은 β로 표현한다. Source가 on인 확률은 $\beta/(\alpha+\beta)$이다. 실제로 소스

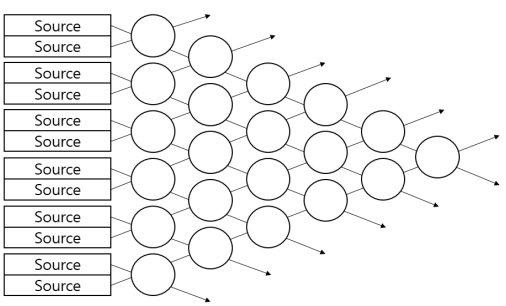

그림 3. 네트워크 토폴로지
Fig. 3. Network topology

표 1. 시뮬레이션 네트워크 파라메터
Table 1. Simulation network parameter

Simulation parameter	value
Number of ports in a node	2
Link capacity	1 Gbps
Link capacity (token bucket, IR)	10 Gbps
Number of flow per source	64
Number of flows in flow aggregate in FAIR framework	8

알고리즘 1 FAIR 프레임워크에서 사용한 IR 알고리즘이다. Schedule() 함수는 정해진 시간에 자신을 다시 호출하는 함수이다.

```
F: flow count per port
FA: flow aggregate count per port
TC[1..F]: Token count
ET[1..F]: eligible time
procedure FAIR(event)
  for i in 1..F do
    TC[i] += token arrival rate * ( simTime - last event time )
    if TC[i] > bucket size then
      TC[i] = bucket size
    end if, end for
  if packet arrival event then
    queue[packet.FA()].enqueue(packet)
  end if
  if link clear then
    for i in 1..FA do
      if not queue[i].empty() then
        while( not queue[i].empty() and TC[queue[i].head().flow()] >= queue[i].head().size()) then
          TC[queue[i].head().flow()] -= queue[i].head().size()
          send(queue[i].pop())
        end while
        if not queue[i].empty() then
          ET[i] = (queue[i].head()'s length - TC[queue[i].head().flow()])/token arrival rate
        else
          ET[i] = large number
        end if, end if, end for
      if not min(ET) == large number then
        Schedule(simTime + min(ET))
      end if, end if
  last event time = simTime
  return
end procedure
```

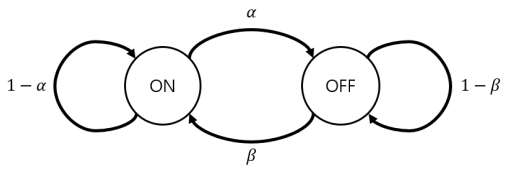

그림 4. 마르코프 체인 모형
Fig. 4. Markov chain model

가 데이터를 생성하는 rate을 'actual input rate'으로 표현하며, nominal input rate * $\beta/(\alpha+\beta)$이다. 표 2에서 실험에 사용한 on-off 모델의 parameter set을 나타내었다. Nominal input rate와 actual input rate를 구분한 이유는 token 생성 속도와 데이터 인입 속도가 동일한 경우 최소 지연시간이 발산하는 현상을 실질적인 데이터 인입 속도를 낮추는 방식으로 막기 위해서이다. 데이터 인입 속도를 이렇게 결정한 이유는 일반적인 인터넷의 인입 속도보다 높은 인입 속도를 갖게 해서 토폴로지 간의 성능 차이를 더 확연히 드러나게 하고자 함이다.

Source가 생성하는 데이터의 특성은 표 3와 같다. 하나의 source는 F개의 플로우를 포함하며 플로우 별로 정해진 속도로 burst를 생성한다. 하나의 burst는 가변 길이의 패킷 1~5개로 이루어진다. 패킷 길이는 시뮬레이션의 단위 시간을 us로 맞추기 위해 link capacity인 1000 bit/us의 배수로 정했다. Unif(a,b)는 a와 b를 포함한 구간에서 uniformly random하게 선택한 정수를 의미한다.

Source의 on 구간 동안 하나의 플로우는 1 us마다 수식 1과 같은 확률로 데이터를 생성한다.

표 2. 데이터 인입 속도, 마르코프 체인 모형 변수
Table 2. Data input rate, Markov chain model parameters

Nominal input rate	Actual input rate	α	β
70%	49%	0.09	0.21
80%	64%	0.06	0.24
90%	81%	0.03	0.27

표 3. 생성하는 데이터의 특징
Table 3. Characteristic of the generated data

Characteristic of the data	value
Packet length	Unif(2,10)×1000 bit
Number of packets in a burst data	Unif(1,5)
Max burst size	50000 bit

$$\frac{nominal\ \ input\ \ rate(bit/\mu s)}{F\ \times\ \ Average\ \ burst\ \ size(bit)} \quad (1)$$

스케줄러와 레귤레이터에서 사용되는 파라메터들은 표 4와 같다. DRR 스케줄러의 Quantum은 simulation 속도를 고려하여 최대 packet 길이인 10000으로 설정하였다. Token의 단위는 1bit로 설정해 simulation의 정밀성을 높였다. Token arrival rate는 데이터 인입 속도에 비례하여 설정된다. 본 시뮬레이션에서는 패킷 손실에 의한 지연시간 증가는 고려하지 않는다. 따라서 스케줄러와 레귤레이터에서 사용한 큐의 크기는 무한하다고 가정한다.

본 시뮬레이션은 특정 플로우의 패킷이 hop 수가 6인 네트워크를 지나는 동안 걸리는 지연시간을 측정한다. FAIR, ATS, FIFO 프레임워크 별로 데이터 인입 속도를 바꿔가며 특정 플로우의 패킷 지연시간 비교를 진행하였다. 시뮬레이션은 토폴로지와 데이터 인입 속도별로 100번씩 반복했다. 시뮬레이션마다 50,000개의 패킷 지연시간을 얻었고, 전체 500만 개의 패킷 지연시간을 사용해 결과를 구했다. 그림 5는 데이터 인입 속도에 따른 각 네트워크 토폴로지의 패킷 지연시간의 상자 수염 그림이다.

최소 지연시간은 데이터 인입 속도나 토폴로지에 상관없이 큰 차이를 보이지 않는다. FAIR와 ATS의 레귤레이터인 IR에서 eligible packet이 cut-through하는 경우가 존재한다. 이런 경우 packet이 겪는 delay는 레귤레이터가 없는 환경과 동일하다. 더 나아가 DRR은 그 자체로 work-conserving 특성을 가지고 있으므로 토폴로지에 상관없이 FIFO와 최소 지연시간이 동일하다. 인입 속도에 상관없이 최소 지연시간이 비슷한 이유는 packet 생성이 Markov process를 따라 생성되기에 packet이 생성되지 않는 구간이 있기 때문이다. 생성되지 않는 동안 token이 쌓이고 스케줄러가

표 4. 스케줄러와 레귤레이터에서 사용한 시뮬레이션 파라메터
Table 4. Simulation parameter used by scheduler and regulator

Simulation parameter	value
DRR Quantum	10000
Token unit size	1 bit
Token Bucket size	50000 bit
Token arrival rate	(10.9375, 12.5, 14.0625) bit/us
Number of queues in DRR	8
Queue size	Infinite

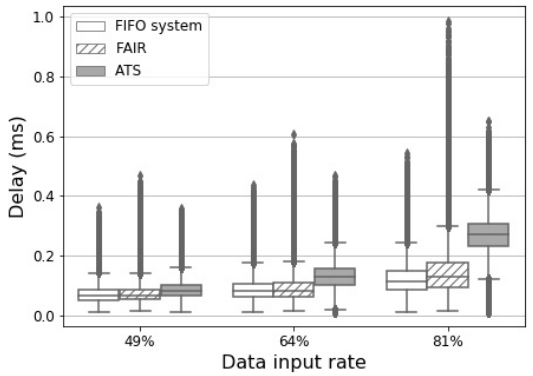

그림 5. 데이터 인입 속도에 따른 각 네트워크 토폴로지의 지연시간의 상자 수염 그림
Fig. 5. Box plot of delay of each network topology according to data input rate

한가해져 이 기간 직후에 생성되는 packet은 최소 지연시간을 갖게 된다.

한편, 이론적인 수치와 달리 FAIR 시스템에서 관찰된 최대 지연시간이 가장 크다. 이는 첫째, 이론적인 최대 지연시간은 낮은 확률의 이벤트가 연속적으로 발생하는 상황을 가정한 것이므로 관찰하기가 극히 어렵다는 측면에 기인한 것으로 보이며, 둘째, FAIR 시스템 고유의 DRR 스케줄러가 동일한 큐에 연속된 긴 패킷이 인입했을 때 해당 큐에 대해서 FIFO 스케줄러에 비해서 상대적으로 긴 지연시간을 제공하기 때문인 것으로 추론된다.

평균 지연시간은 모든 데이터 인입 속도에서 FIFO가 가장 좋고 FAIR, ATS의 순서로 좋다. 측정된 최대 지연시간은 데이터 인입 속도 64%, 81%에서는 FIFO 시스템이 가장 좋고 ATS, FAIR 프레임워크 순서로 좋으며 데이터 인입 속도 49%에서는 ATS가 제일 좋다. 그림 6은 데이터 인입 속도에 따른 FAIR 프레임워크와 ATS, FIFO 시스템의 평균 지연시간만을 다룬 그래프이다.

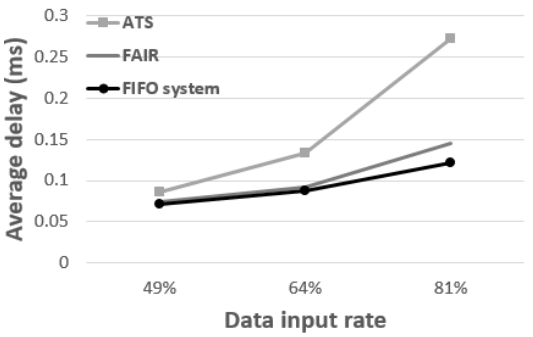

그림 6. 데이터 인입 속도에 따른 FAIR 프레임워크, ATS, FIFO 시스템의 평균 지연시간
Fig. 6. Average delay of FAIR, ATS, FIFO system according to data input rate

모든 경우에서 FAIR 프레임워크가 ATS보다 평균 지연시간이 낮다. 이는 레귤레이터로 인한 queueing delay의 영향이다. 레귤레이터가 많은 순서인 ATS, FAIR, FIFO 순서로 평균 지연시간이 안 좋게 나오는 것으로 추측된다. 데이터 인입 속도가 증가할수록 FAIR 프레임워크와 ATS와의 평균 지연시간의 차이가 증가하는 것을 확인할 수 있다. ATS는 패킷이 통과하는 레귤레이터가 많아 평균 지연시간이 큰데 데이터 인입 속도가 증가할수록 레귤레이터 내의 큐에서 대기하는 시간이 길어져 FAIR 프레임워크나 FIFO 시스템보다 지연시간 증가 폭이 크다. 또한, 데이터 인입 속도가 49%, 64%인 경우에는 FAIR 프레임워크와 FIFO 시스템의 평균 지연시간이 차이가 거의 없고 81%에 경우에도 ATS보다 확연히 작은 차이를 보인다. 이를 통해 평균 지연시간 측면에서 FAIR 프레임워크가 ATS에 비해 좋고 FIFO 시스템과 차이가 크지 않음을 알 수 있다.

3.2 Cycle이 있는 네트워크 토폴로지

본 섹션에서는 cycle이 존재하는 토폴로지의 네트워크를 시뮬레이션하고 이를 분석한다. 현재의 DiffServ기반 인터넷이 지연시간 최대치를 보장하지 못하는 환경을 설정하였다. 네트워크 토폴로지는 그림 7과 같다. 그림 7에서 원은 2x2 스위치 노드이다. 그림 7에서 데이터를 생성하는 source는 cycle이 없는 시뮬레이션과 동일하게 그림 4와 같이 마르코프 on-off 모델을 따른다.

FAIR 프레임워크에서는, 모든 노드에 통합플로우별 큐와 이를 처리하는 DRR 스케줄러가 있고, 3번과 4번 노드 각각에 두 노드를 연결하는 링크의 port에 통합플로우 별 IR이 있다. ATS에서는, 모든 노드에 FIFO 스케줄러가 있고 port마다 input port 별 IR이 있다. FIFO 시스템에서는, 모든 노드에 FIFO 스케줄

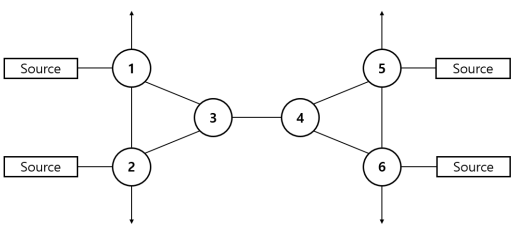

그림 7. Cycle이 있는 네트워크 토폴로지
Fig. 7. Network topology with cycle

러만 존재한다. FAIR 프레임워크에서 단위 네트워크의 hop 수는 Cycle이 없는 시뮬레이션과 동일하게 비교하기 위해 3으로 정했다. 시뮬레이션 네트워크 토폴로지의 파라메터는 표 5와 같이 정리된다.

표 6에서 실험에 사용한 on-off 모델의 parameter set을 나타내었다. Cycle이 없는 시뮬레이션과 마찬가지로 인터넷의 평균 인입 속도에 비해 큰 인입 속도를 갖는 환경을 구성해 레귤레이터에 주는 영향을 키워 토폴로지간 지연시간 차이를 더 잘 보고자 하였다.

Cycle이 있는 네트워크 토폴로지에서 플로우는 Cycle이 만들어지도록 정해진 경로를 따라 이동한다. 플로우의 경로는 표 7과 같다. 표 7에 route의 숫자는 그림 7에서 노드의 번호이다. 이 중 3번 플로우나 4번 플로우는 6 hop 거리를 지나므로 이 두 플로우의 지연시간을 측정한다.

플로우가 생성하는 데이터의 특성은 표 8과 같다. 패킷 길이는 시뮬레이션이 us단위를 따르도록 link capacity인 200 bit/us의 배수로 정했다.

표 5. 시뮬레이션 네트워크 파라메터
Table 5. Simulation network parameter

Simulation parameter	value
Number of ports in a node	2
Link capacity	200 Mbps
Number of flow per source	1
Number of flows in flow aggregate in FAIR framework	1

표 6. 데이터 인입 속도, 마르코프 체인 모형 변수
Table 6. Data input rate, Markov chain model parameters

Nominal input rate	Actual input rate	α	β
70%	49%	0.09	0.21
80%	64%	0.06	0.24
90%	81%	0.03	0.27

표 7. 플로우 별 경로
Table 7. Route by flow

Number of flow	Route
1	1 → 3 → 2
2	6 → 4 → 5
3	2 → 1 → 3 → 4 → 5 → 6
4	5 → 6 → 4 → 3 → 2 → 1

표 8. 생성하는 데이터의 특징
Table 8. Characteristic of the generated data

Characteristic of the data	value
Packet length	Unif(1,5)×200 bit
Number of packets in a burst data	Unif(1,5)
Max burst size	5000 bit

스케줄러와 레귤레이터에서 사용되는 파라메터들은 표 9와 같다. Cycle이 없는 시뮬레이션과 마찬가지로 패킷 손실에 의한 지연시간은 고려하지 않는다. 이를 위해 레귤레이터와 스케줄러의 큐의 크기는 무한하다고 가정한다.

본 시뮬레이션은 특정 플로우의 패킷이 cycle이 있는 단위 네트워크 2개를 지나는 동안 걸리는 지연시간을 측정한다. 관찰하는 플로우는 6 hop 거리를 이동한다. FAIR 프레임워크, ATS, FIFO 시스템 별로 데이터 인입 속도를 바꿔가며 특정 플로우의 패킷 지연시간 비교를 진행하였다. 시뮬레이션은 토폴로지와 데이터 인입 속도별로 100번씩 반복했다. 시뮬레이션마다 50,000개의 패킷 지연시간을 얻었고, 전체 500만개의 패킷 지연시간을 사용해 결과를 구했다. 그림 8은 데이터 인입 속도에 따른 각 네트워크 토폴로지의 패킷 지연시간의 상자 수염 그림이다.

최소 지연시간은 데이터 인입 속도나 토폴로지에 상관없이 큰 차이를 보이지 않는다. 이는 토폴로지간 이론상 최소 지연시간이 동일하고, 패킷이 Markov process를 따라 생성되어 패킷이 생성되지 않는 기간이 있기 때문이다.

평균 지연시간은 모든 데이터 인입 속도에서 FAIR 프레임워크가 가장 좋고 FIFO 시스템, ATS의 순서로 좋다. 본 시뮬레이션 환경에서는 단위 네트워크를 연결하는 링크에서 지나가는 플로우 수가 다른 링크에 비해 적다. 이로 인해 FAIR 프레임워크에서 IR이 지

표 9. 스케줄러와 레귤레이터에서 사용한 시뮬레이션 파라메터
Table 9. Simulation parameter used by scheduler and regulator

Simulation parameter	value
DRR Quantum	1000
Token unit size	1 bit
Token Bucket size	5000 bit
Token arrival rate	(70, 80, 90) bit/us
Number of queues in DRR	2
Queue size	Infinite

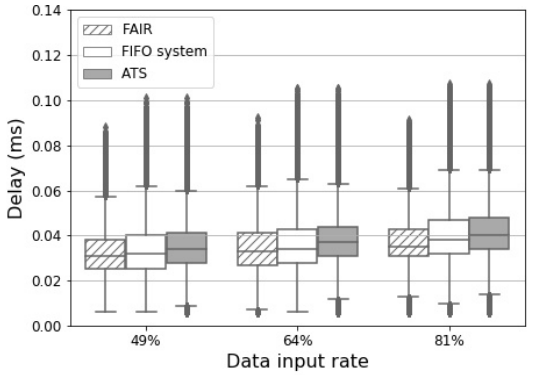

그림 8. 데이터 인입 속도에 따른 각 네트워크 토폴로지의 지연시간의 상자 수염 그림
Fig. 8. Box plot of delay of each network topology according to data input rate

연시간에 주는 영향이 작아진다. 지연시간은 스케줄러에 의해 상당부분 결정되는데 DRR이 FIFO보다 평균 지연시간 측면에서 우수하기에 FAIR가 FIFO보다 좋은 결과를 보인다. ATS는 FIFO에 레귤레이터에서의 지연시간이 추가되었으니 가장 안 좋은 결과를 보인다.

측정된 최대 지연시간은 모든 경우에서 FAIR 프레임워크가 가장 좋고 FIFO 시스템과 ATS는 동일한 지연시간을 갖는다. ATS에서 IR이 평균 지연시간에만 영향을 주고 최대 지연시간에는 영향을 끼치지 않았다. FIFO에서 충분히 지연된 패킷은 IR에서 지연 없이 전송된 것으로 보인다. FAIR 프레임워크의 최대 지연시간이 가장 낮은 지연시간을 갖는다. Cycle이 없는 환경과 달리 본 시뮬레이션 환경에서 FAIR의 최대 지연시간이 작은 값을 보이는 이유는 스케줄러의 큐가 적어서 연속된 긴 패킷으로 인한 delay 증가 효과가 크지 않기 때문으로 추론된다.

그림 9는 데이터 인입 속도에 따른 FAIR 프레임워크와 ATS, FIFO 시스템의 평균 지연시간만을 다룬 그래프이다.

모든 경우에서 FAIR 프레임워크가 FIFO 시스템과 ATS보다 평균 지연시간이 낮은 것을 확인할 수 있다. Cycle이 없는 네트워크 환경과 달리 토폴로지와 인입 속도에 따른 평균 지연시간 차이가 적은 이유는 스케줄러나 레귤레이터가 처리하는 플로우의 수가 작아서 스케줄러 방식이나 레귤레이터 존재 여부가 크게 작용하고 있지 않기 때문으로 추론된다

본 시뮬레이션 결과를 통해 Cycle이 있는 네트워크 토폴로지에서 FAIR 프레임워크가 평균 지연시간 측면에서 ATS와 FIFO 시스템에 비해 성능이 우수함을 알 수 있다.

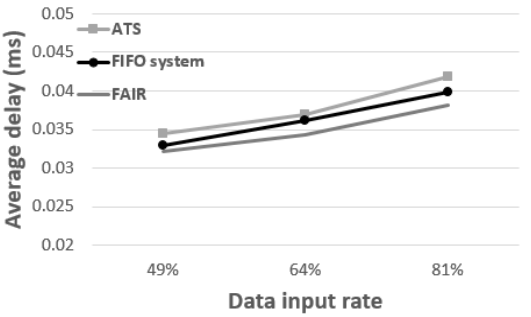

그림 9. 데이터 인입 속도에 따른 FAIR 프레임워크, ATS, FIFO 시스템의 평균 지연시간
Fig. 9. Average delay of FAIR, ATS, FIFO system according to data input rate

Ⅳ. 결 론

본 논문에서는 지연시간 최대치를 보장하는 네트워크 프레임워크들에 대해 알아보고 그중 ATS와 FAIR 프레임워크의 통계적 성능 특성을 시뮬레이션을 통해 분석하여 레귤레이터가 없는 FIFO 시스템과 비교하였다. 첫 번째 시뮬레이션은 2x2 스위치 노드로 구성된 3 hop 길이의 단위 네트워크 2개로 이루어진 cycle이 없는 네트워크 토폴로지 상에서 FAIR 프레임워크, ATS, FIFO 시스템의 데이터 인입 속도에 따른 패킷 지연시간을 비교하였다. 두 번째 시뮬레이션은 2x2 스위치 노드로 구성된 cycle이 있는 단위 네트워크 2개로 이루어진 6 hop 길이의 네트워크 토폴로지 상에서 FAIR 프레임워크, ATS, FIFO 시스템의 데이터 인입 속도에 따른 패킷 지연시간을 비교하였다. 두 시뮬레이션은 각 토폴로지와 데이터 인입 속도마다 100번씩 수행하였고 최소 지연시간, 평균 지연시간, 최대 지연시간을 모두 비교하였다. Cycle이 없는 시뮬레이션 결과 모든 데이터 인입 속도에서 FAIR 프레임워크는 평균 지연시간 측면에서 ATS보다 우수하며 FIFO 시스템과는 유사한 성능을 보였다. Cycle이 있는 경우에는 평균 지연시간 측면이 FAIR 프레임워크가 ATS와 FIFO 시스템보다 우수하였다. FAIR 프레임워크는 이로써 지연시간 최대치와 평균 지연시간의 측면에서 모두 ATS보다 월등하다는 것이 증명되었다. 또한, FAIR 프레임워크는 평균 지연시간을 증가시킨다는 우려로 인해 사용되지 않고 있으나 시뮬레이션 결과 평균 지연시간은 FIFO 시스템과 유사함이 증명되었다. 따라서 FAIR 프레임워크는 우려와 달리 평균 지연시간을 크게 증가시키지 않으면서도 Cycle에 상관없이 ATS보다 낮은 지연시간 최대치를 보장하는 기술임을 증명하였다.

본 논문에서는 동일한 특성의 플로우들이 트리 구조의 네트워크에서 동일한 hop 수의 노드를 지나가는 symmetric 환경과 간단한 Cycle이 있는 환경을 상정하였다. 실제로 낮은 priority 트래픽이 존재하는 DiffServ 프레임워크의 환경과는 차이가 있다. 향후에는 다양한 토폴로지, 다양한 특성의 플로우를 포함한 네트워크에서의 확률적 성능 특성을 연구할 예정이다.

References

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

[2] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched Ethernet networks," in *Proc. 28th ECRTS*, pp. 75-85, Jul. 2016.

[3] J. Joung, "End-to-end delay guarantee in IEEE 802.1 TSN with Non-work conserving scheduler," *J. IIBC*, vol. 18, no. 6, Dec. 2018. DOI: 10.7236/JIIBC.2018.18.6.121.

[4] J. Joung, "Framework for delay guarantee in multi-domain networks based on interleaved regulators," *Electronics*, vol. 9, no. 3, Mar. 2020.

[5] Recommendation ITU-T Y.3113 (2021) "*Requirements and framework for latency guarantee in large scale networks including IMT-2020 network.*"

[6] 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.

[7] J.-Y. Le Boudec, "A theory of traffic regulators for deterministic networks with application to interleaved regulators," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, Dec. 2018.

[8] *OMNeT++*, https://omnetpp.org/

권 주 혁 (Juhyeok Kwon)

2020년 8월 : 상명대학교 휴먼지능정보공학과 학사

2020년 9월~현재 : 상명대학교 지능정보공학과 석사

<관심분야> 유무선통신, 네트워크, 임베디드 시스템

정 진 우 (Jinoo Joung)

1992년 2월 : KAIST 전자공학과 학사

1994년 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, 임베디드 시스템

[ORCID:0000-0003-3053-9691]