논문 21-46-11-25 The Journal of Korean Institute of Communications and Information Sciences '21-11 Vol.46 No.11
https://doi.org/10.7840/kics.2021.46.11.1982

TSN지원 소프트웨어 기반 브릿지 성능 분석

변 일 무◆°, 고 경 준*,
김 성 민**, 정 아 름***

Performance Analysis of Software-Based Bridges Supporting Time Sensitive Networks

Ilmu Byun◆°, Kyeong-jun Ko*,
Sung min Kim**, Arm Jeong***

요 약

본 논문에서는 TSN(Time Sensitive Network)을 지원하는 브릿지를 범용 프로세서에서 소프트웨어로 구현하는 방법을 제시하고 이를 바탕으로 구현된 브릿지의 성능을 분석한다. 본 논문에서는 Intel DPDK로 TSN의 CBS(Credit Based Shaper)와 TAS(Time Aware Shaper)를 구현하였으며, TSN지원 브릿지를 이용한 2 홉 통신의 단방향 지연시간은 99%달성률 기준으로 CBS와 TAS가 각각 718μs와 560μs를 달성하였다.

Key Words : Time sensitive network, low latency, deterministic latency, software defined network

ABSTRACT

In this paper, we present a method for implementing bridges that support time-sensitive networks (TSNs) as software in general-purpose processors and analyze the performance of bridges implemented based on them. In this paper, CBS(Credit Based Shaper) and TAS(Time Aware Shaper) of TSN are implemented with Intel DPDK, and the 99 percentile one-way latency of 2 hop communication through the TSN-supported bridge is 718μs and 560μs, respectively.

Ⅰ. 서 론

TSN(Time Sensitive Network)은 IEEE 802.1 working group에서 개발하고 있는 표준으로 확정적 지연(deterministic latency)을 보장하는 것이 특징이다. TSN은 실시간 네트워크를 제공하므로 스마트 팩토리, 차량 및 철도 네트워크, 5G URLLC (Ultra Reliable Low Latency Communication)와의 코어망 등에서 활발히 연구가 진행되고 있다.

프라운호퍼에서는 TSN 브리지를 다양한 플랫폼에서 구현하고 그 성능을 비교하였다[1]. 비교를 통해 범용 CPU에 실시간 리눅스 커널을 적용하면 최소 125μs를 달성함을 보였고, FPGA 개발 시에는 최소 1μs 지연시간 달성이 가능함을 보였다. 국내에서는 XMOS 하드웨어를 기반으로 TAS (Time Aware Shaping)적용 스위치를 개발하여 큐(Queue) 매니지먼트 기술을 검증하였다[2].

Intel DPDK를 이용한 소프트웨어 기반 네트워크 장치 개발에서도 많은 연구가 진행되었다. Intel DPDK를 이용하여 저지연 가상네트워크 구현하였고[3], Intel DPDK가 200Gbps를 제공하는 라우터 구현에 적용될 수 있음을 보였다[4]. 또한, Intel DPDK를 기반으로 TAS를 구현하여 4 홉 통신 시 97%의 확률로 73ms의 지연시간을 달성할 수 있음을 보였다[5].

본 논문에서는 범용 프로세서에서 Intel DPDK기반으로 소프트웨어 기반 TSN지원 브릿지를 개발한다. 개발한 소프트웨어 기반 브릿지는 큐 수신 및 송신 전용 프로세서를 할당함으로써, 리눅스 커널 기반 TSN 브리지와는 다르게 다른 응용프로그램으로 인한 성능 저하를 예방할 수 있는 장점이 있다. 본 논문에서는 TSN의 CBS(Credit Based Shaper)와 TAS(Time Aware Shaper)를 지원하는 브릿지를 구현하였으며, 해당 브릿지의 2홉 단방향 지연시간은 99백분위수를 기준으로 CBS는 718μs를 TAS는 560μs를 달성하였다.

※ 본 연구는 한국철도기술연구원 기본사업의 연구비 지원으로 수행되었습니다.
◆° First & Corresponding Author : Korea Railroad Research Institute, ilmubyun@krri.re.kr, 선임연구원, 정회원
* Korea Railroad Research Institute, kkj8000@krri.re.kr, 선임연구원, 정회원
** Department of Artificial Intelligence, Sungkyunkwan University, semih@skku.edu, 학생(박사과정)
*** TSN lab, arm@tsnlab.com, 연구원
논문번호 : 202109-224-D-LU, Received September 2, 2021; Revised October 1, 2021; Accepted October 5, 2021

Ⅱ. 소프트웨어 기반 브릿지 설계

본 논문에서 개발한 브릿지의 구조는 그림 1과 같다. 본 논문에서는 브릿지의 데이터 중계 지연시간을 감소시키고 스트림 증가에 따른 지연시간 변화를 최소화하기 위해서 각 프로세서 별로 서로 다른 역할을 가지도록 하였다. 브릿지에는 총 3개의 프로세서가 사용되었으며, 이 중 하나는 Rx 프로세서로 DPDK 큐에 저장된 데이터에 큐잉 알고리즘을 적용해 각 이더넷 포트별 Tx 큐에 저장하는 역할을 수행한다. Rx 프로세서는 이더넷 프레임의 VLAN ID와 PCP(Priority Code Point)를 이용하여 해당 프레임이 저장될 큐를 선정하고 해당 큐에 프레임을 저장하는 역할을 수행한다. Tx 프로세서는 이더넷 프레임의 트래픽 클래스에 따라 802.1Q표준의 CBS와 TAS를 적용하여 Tx 큐의 이더넷 프레임을 Tx 버퍼로 내보내는 역할을 수행한다. OS프로세서는 각 프로세서의 파라미터 세팅 및 브릿지 관리를 위한 역할을 수행한다.

Rx 프로세싱은 표 1과 같이 각 Rx 포트의 Rx 버퍼에 저장된 패킷을 라운드 로빈(round robin) 방식으로 읽는다. 이는 Rx 버퍼에 저장된 패킷을 읽기 전에는 패킷의 우선순위를 파악할 수 없기 때문이다. 만약 패킷 중에 시간 동기화를 위한 IEEE 1588 PTP(Precison Time Protocol) 패킷이 존재하면 우선순위(priority)를 가장 높게 설정하고 전용 큐에 저장한다. 이는 TSN에서는 시간 동기화가 되어야만 TAS 및 CBS 동작을 정상적으로 수행할 수 있기 때문이다. 다른 패킷들은 우선순위를 파악한 후 이에 따라 별도의 큐에 저장한다.

Tx 프로세싱은 표 2와 같이 각 포트 별 큐의 우선순위에 따라 수행한다. 만약, 우선순위가 동일한 서로 다른 포트의 큐에 패킷이 있다면 라운드로빈 방식으로 동작한다. Tx 프로세싱에 사용되는 NIC(Network Interface Card)는 TSN을 이해하지 못하므로 Tx 버퍼에 패킷이 도착한 순서대로 패킷을 송부한다. 그러므

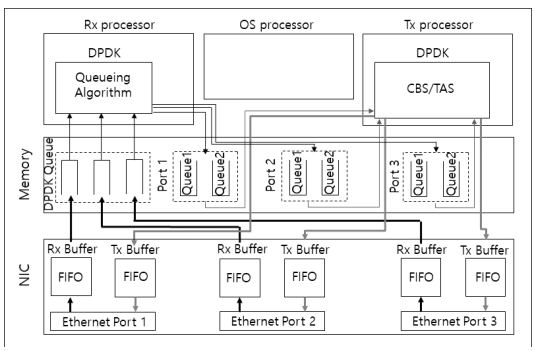

그림 1. 소프트웨어 기반 브릿지 구조도
Fig. 1. Structure of software-based bridge

표 1. Rx 프로세싱의 수도 코드
Table 1. Pseudo code of Rx processing

```
Rx processing
while
 for each port
   packet = read_packet_from_rx_buffer();
   if packet is PTP packet
      parse_ptp_packet();
      priority = HIGHEST;
   else
      priority = packets.vlan.priority;
   end
   routing_table = routing_table_update(packets);
   pusch_packets_into_queues(packets,priority,
   routing_table);
 endfor
endwhile
```

표 2. Tx 프로세싱의 수도 코드
Table 2. Pseudo code of Tx processing

```
Tx processing
while
 if PTP queue is not empty
   //return index.port and index.queue for PTP
    index = queue_selection_for_PTP();
 else if TAS mode is on
   //return index.port and index.queue if any TAS
   queue has packets and TAS slot is available
   index = queue_selection_with_TAS();
 else if CBS mode is on
   //update credits for queues
   credits = credit_updates_for_queues();

   //return index.port and index.queue of the queue
   having the maximum credit
   index = queue_selection_with_CBS(credits);
 endif

 if index == NULL
    index = normal_queue_selection();
 endif

 send_first_packet_from_queue_to_buffer(index);
endwhile
```

로 Tx 프로세싱을 수행할 때 TAS와 CBS가 적용될 수 있도록 패킷을 필터링해서 Tx 버퍼에 패킷을 저장하는 것이 필요하다. TAS를 적용할 때는 TAS 패킷은 특정 타임 슬랏(slot)에서 반드시 전송되어야 하므로, 이를 고려하여 TAS 패킷을 큐에서 Tx 버퍼로 이동시키는 것이 필요하다. 또한, CBS 적용 시에는 크레딧(credit)이 0 이상인 CBS 큐 중 크레딧이 가장 큰 큐의 첫 번째 패킷을 큐에서 Tx 버퍼로 이동시킨다.

Ⅲ. 브릿지 구현 및 성능분석

3.1 브릿지 구현

2장의 브릿지 설계를 기반으로 표 1의 하드웨어를 이용하여 그림 2의 브릿지를 구현하였다. 브릿지에 4코어를 갖는 CPU를 사용하여 코어별로 Rx프로세싱, Tx프로세싱, OS관리를 담당하도록 하였다. 메모리는 4GB DDR3를 이용하였고, 네트워크 인터페이스 카드는 1Gbps를 지원하는 인텔 i210를 이용하였다. 그림 2는 CBS와 TAS를 적용한 브릿지 개발제품이고 표 3은 개발제품의 하드웨어 스펙이다.

표 3. 브릿지 구성 HW
Table 3. bridge HW

구성	스펙
CPU	Intel(R) Celeron(R) CPU J3160 @ 1.60GHz 4core
메모리	4GB DDR3
NIC	1Gbps Intel i210 rev3 x 4 port

그림 2. 소프트웨어 기반 브릿지 개발제품
Fig. 2. Development product of software-based bridge

3.2 브릿지 성능 분석

브릿지의 성능 검증을 위한 시뮬레이션 환경은 그림 3과 같이 단말(end point) 3개와 브릿지 1개로 구성하였다. 단말 A는 단말 B에게 TSN 트래픽을 전송하고 있으며, 단말 C는 단말 B에게 BE(Best Effort) 트래픽을 전송하고 있다.

단말 A와 C가 단말 B에게 전송하는 패킷 사이즈는 1460바이트이다. 단말 A는 주기적으로 전송하는 반면, 단말 C는 본 시험에서 사용된 이더넷의 최대 전송용량인 1Gbps를 모두 소모하도록 패킷을 단말 B에게 연속적으로 전송하였다. 그림 4는 이러한 환경에서 단말 A가 패킷을 총 100,000회 전송하였을 때의 단말 A와 B간의 단방향 지연시간 cdf이다.

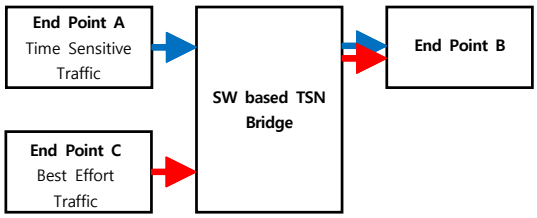

그림 3. 시뮬레이션 환경
Fig. 3. Simulation Environment

그림 4에서 'TAS'는 단말 A가 TAS를 이용하여 TSN 트래픽을 전송하는 경우이고, 'CBS'는 단말A가 CBS를 이용하여 TSN 트래픽을 전송하는 경우이며, 'BE'는 단말 A가 기존의 Best effort 방식대로 TSN 트래픽을 전송하는 경우이다. 또한, 'TAS, CBS, BE'는 단말 C가 트래픽을 전송하지 않아 네트워크에 부하가 없는 상황이고, 'TASw, CBSw, BEw'는 단말 C가 트래픽을 전송해서 네트워크에 부하가 발생한 상황을 의미한다. 마지막으로 'Direct'는 단말 A와 B가 직접 연결된 경우의 단방향 지연시간이다.

'Direct'와 'TAS/CBS'를 비교하면 SW기반 브릿지의 평균 지연시간을 알 수 있다. 'Direct'의 평균 지연은 244μs이고 'TAS'의 평균 지연은 288μs이며 'CBS'의 평균 지연은 285μs이므로 브릿지가 추가됨으로 인해 각각 44μs와 41μs의 지연시간이 증가했음을 알 수 있다.

그림 4를 통해 단말 C가 네트워크에 부하를 발생시

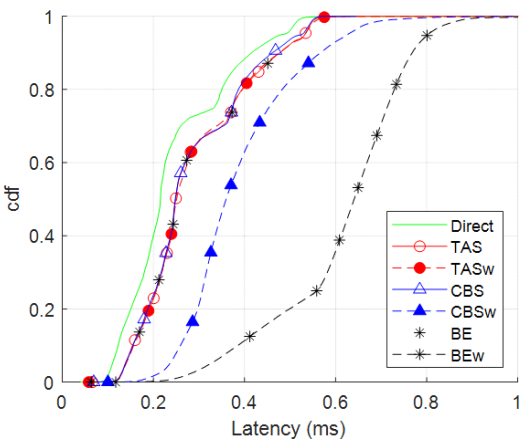

그림 4. CBS와 TAS 적용 시 단말 A와 B간 2홉 통신의 단방향 지연 시간
Fig. 4. One-way latency of two-hop communication between end-points A and B when CBS and TAS are applied

키지 않는 이상적인 상황에서는 'TAS', 'CBS', 'BE'의 지연시간이 거의 동일함을 알 수 있다. 이를 통해 TAS와 CBS 구현을 위해 브릿지에 기능이 추가되었으나, 이로 인한 프로세싱 시간은 무시할 수준임을 알 수 있다.

단말 C가 네트워크에 부하를 발생시키는 경우에는 지연시간의 변화가 발생하였다. TAS가 적용된 경우에는 TSN 트래픽 지연시간의 변화가 거의 없었으나, CBS의 경우는 평균 지연시간이 285μs에서 389μs로 증가하였으며 BE의 경우는 평균 지연시간이 285μs에서 615 μs로 증가하였다.

네트워크에 부하가 걸린 상황에서 CBS와 TAS의 90% 지연시간은 각각 567μs와 476μs로 TAS가 91μs 더 짧았으며, 99% 지연시간은 각각 718μs와 560μs로 TAS와 CBS간 격차가 158μs로 증가하였다. 99.9%지연시간은 각각 898μs와 582μs로 격차가 316μs로 증가하였다. 이는 TAS 기법은 TSN트래픽 전송을 위해 타임슬롯을 주기적으로 할당하는 반면 CBS는 크레딧이 0 이상인 경우에도, BE 트래픽이 전송 중인 경우 대기 시간이 발생하기 때문이다.

3.3 브릿지 성능 향상 방안

본 논문에서 개발한 브릿지는 범용 PC를 이용하였으므로, CPU의 성능을 증가시키면 Rx 및 Tx 프로세싱의 통신 Latency를 감소시킬 수 있다. 또한, CPU 수를 증가시켜서 Latency를 감소시킬 수 있다. 현재 논문에서는 3개의 포트를 하나의 Rx프로세서가 관장하였으나, 포트별로 별도의 Rx프로세서를 할당한다면 Latency를 감소시키는 효과가 있다.

Ⅳ. 결 론

본 논문에서는 Intel DPDK기반의 TSN지원 브릿지를 개발하였고, 99백분위에서 2-홉 단방향 전송 지연이 560μs이고 최대 지연시간이 1ms를 넘지 않음을 보였다. 이 값은 현재 열차제어용 네트워크의 요구사항[6]인 10ms 보다 작은 값으로 열차제어 용으로 사용할 수 있고, 스마트 팩토리의 프로세스 자동화 요구사항인 50ms의 단대단 지연 요구보다 짧으므로[7], 해당 어플리케이션에도 적용이 가능하다. 또한, 소프트웨어 기반 브릿지를 이용하면 접근성이 높고 낮은 비용으로 빠르게 TSN을 도입하는 장점이 있을 것으로 판단된다.

References

[1] Fraunhofer IPMS(Institute for Photoni MicroSystems), *Latency Optimized TSN networks*, Fraunhofer, White paper, pp. 1-5, May 2021.

[2] M. Son, C. Yoon, J.-U. Kim, and P. Park, "Performance of controlled delay active queue management scheme for time sensitive networking with gated scheduling and its implementation," *J. IEIE*, vol. 56, no. 4, pp. 20-33, Apr. 2019.

[3] Z. Xiang, et al., "Reducing latency in virtual machines: Enabling tactile internet for human-machine co-working," in *IEEE J. on Sel. Areas in Commun.*, vol. 37, no. 5, pp. 1098-1116, May 2019.

[4] H. Redžović, et al., "Implementation and performance comparison of high-capacity software routers," *Comput. Netw.*, vol. 183, Oct. 2020.

[5] W. Mandarawi, H. Chahed, and H. de Meer, "A framework for virtualizing time-aware shaper using high performance NFV," *2020 25th IEEE Int. Conf. ETFA*, 2020, pp. 1621-1628, doi: 10.1109/ETFA46521.2020.92 11960.

[6] IEC 61375-3-4, *Electronic railway equipment - Train communication network (TCN) - Part 3-4: Ethernet Consist Network*, 1st Ed., Mar. 2014.

[7] 3GPP TS 22.261, *Service requirements for next generation new services and markets (release 16)*, v16.6.0, Dec. 2018.