논문 22-47-07-06 The Journal of Korean Institute of Communications and Information Sciences '22-07 Vol.47 No.07
https://doi.org/10.7840/kics.2022.47.7.944

DDQN을 활용한 강화학습 기반 타임슬롯 스케줄링

류 지 혜•, 권 주 혁*, 정 진 우°

Timeslot Scheduling with Reinforcement Learning Using a Double Deep Q-Network

Jihye Ryu•, Juhyeok Kwon*, Jinoo Joung°

요 약

트래픽 특성과 요구사항의 다양한 변화에 네트워크 자율적으로 유연하게 적응하고 대처하는 방안으로, 강화학습을 적용한 네트워크 스케줄러가 최근 주목받고 있다. 본 연구에서는 딥러닝을 적용한 강화학습 모델인 double deep q-network(DDQN)을 사용해 우선순위 기반의 타임슬롯 스케줄링을 구현한다. DDQN 에이전트의 행동에 대한 가치를 평가하기 위해 reward는 timeslot에서 전송된 패킷의 추정 delay와 deadline의 차이, 그리고 패킷의 우선순위에 기반해 지급하는 function으로 정의하였다. 시뮬레이션 결과, 학습된 스케줄러가 strict priority (SP) 혹은 weighted round robin(WRR)과 같은 기존 알고리즘으로 스케줄링했을 때 우려되는 문제점을 극복한 것을 확인할 수 있었다. 또한, 기존 스케줄러보다 높은 누적 보상의 합인 score를 기록하며, deadline 내에 더 많은 packet이 도착함을 확인하였다. 본 연구를 통해서 대규모 유선 네트워크에서 자율 네트워크 스케줄링 기능 실현의 가능성을 타진하였다. 특히 제안하는 DDQN 기반 강화학습 에이전트를 사용하면 자율성과 성능을 모두 개선할 수 있을 것으로 기대한다.

Key Words : Reinforcement learning, Network, Deep learning, Autonomous networking, Q-learning, Double Deep Q-network, Deep Q-network

ABSTRACT

To adopt reinforcement learning in the network scheduling area is getting more attention than ever, because of its flexibility to adapt to the dynamic changes in network traffic specifications and their requirements. In this study, a timeslot scheduling algorithm based on priority is designed using Double deep q-network (DDQN), a reinforcement learning algorithm. To evaluate the behavior of the DDQN agent, a reward function is defined based on the difference between the estimated delay and the deadline of packets transmitted at timeslot; and on the priority of packets. The simulation showed that the designed scheduling algorithm performs better than the existing algorithms such as the strict priority (SP) or weighted round robin (WRR) scheduler, in the sense that more packets have arrived within the deadline. By using the proposed DDQN-based scheduler, it is expected that autonomous network scheduling can be realized in the upcoming network framework.

※ 본 연구는 과학기술정보통신부 '정보통신방송연구개발사업'의 지원을 받아 수행되었음. (과제고유번호: 2018-0-00846)
• First Author : Sangmyung University, Department of Artifical Intelligence & Informatics, jh_r_1004@naver.com, 학생회원
° Corresponding Author : Sangmyung University, Department of Human-centered AI, jjoung@smu.ac.kr, 정회원
* Sangmyung University, Department of Artifical Intelligence & Informatics, 학생회원
논문번호 : 201911-039-B-RN, Received March 18, 2022; Revised May 4, 2022; Accepted May 8, 2022

Ⅰ. 서 론

IoT, 스마트 팩토리, 센서 네트워크, 5G등 다양한 네트워크 디바이스의 환경에서 QoS 보장 문제는 이전보다 더 엄격한 요구사항을 갖기 때문에 효율적으로 한정된 자원을 분배하는 것이 무엇보다 중요하다. 네트워크와 딥러닝 분야의 연구는 오랜 시간 동안 독립적으로 발전되어 왔다. 현재 QoS에서 flow들의 우선순위에 따라 패킷을 스케줄링하기 위해 높은 우선순위를 먼저 전송하는 SP 방식과 같은 알고리즘을 사용하고 있다. 그러나 5G 이후로 점점 네트워크의 크기가 방대해져 많은 데이터들과 디바이스들이 연결되는 시대로 접어들면서 네트워크 운영자가 없이도 네트워크가 스스로 운용할 수 있는 자율네트워킹이 등장하였다[1]. 자율 컴퓨팅에서 착안하여 네트워크의 지능화, 자동화를 이루기 위해 자율네트워킹을 최초로 제안한 논문에서는 self-management를 위해 네 가지 영역의 목표를 디자인하였다. self-configuration을 통해 네트워크가 관리자나 관리 시스템의 개입 없이 스스로 설정하는 영역이다. 다음으로 self-healing은 자동적으로 문제를 해결하거나 변화한 환경에 적응하는 기능이다. self-optimizing은 네트워크의 요구사항 목표를 달성하기 위해 스스로 최적의 방식을 찾는 것을 뜻한다. 마지막으로 self-protection은 잠재적으로 공격에 대응하기 위해 자동으로 대비하는 역할을 뜻한다. 이러한 추세에 맞춰 현재 기술의 한계를 극복하기 위해서 이미 다양한 분야에서 성능이 입증된 딥러닝이 솔루션이 될 수 있다는 관점에서 다양한 연구가 이루어지고 있다[2-4,10]. [2]의 연구에서는 모바일 네트워킹 분야에서의 딥러닝 연구의 필요성과 관련 분야에서의 연구가 어떻게 이루어지고 있는지 전체적으로 조사하였다. 해당 논문에 따르면 네트워크 컨트롤 분야에서 스케줄링에 딥러닝을 수행한 기존 연구들이 있으며 모두 강화학습을 사용하여 진행한 바 있다. 그중 [3]의 연구는 미래 모바일 네트워크에 등장하게 될 IoT 디바이스들의 새로운 어플리케이션과 트래픽의 클래스들을 위해 DQN을 활용하여 스케줄링하는 방법을 고안하였다. 매우 동적인 트래픽의 변화에 적응하기 위해 강화학습을 적용한 스케줄러를 구현해 최적의 IoT 트래픽 스케줄을 달성하였다. [4]에서는 제한된 리소스를 IoT 에지 컴퓨팅 시스템에 효율적으로 분배하는 에이전트를 DQN을 통해 학습하였다. 이외에도 DQN은 교통신호 체계 연구에도 활용된 사례가 있다. [5]에서는 실제 교통 데이터를 활용해 지능형 신호 체계 모델을 제안한다. [6]에서는 교통상황에서 적절한 state와 reward의 설정에 대해 분석하였다. 교통신호의 최적화가 곧 차량 주행시간의 최소화임을 증명해 다양한 state와 reward의 조합으로 학습을 진행한 결과, 가장 단순한 state와 reward로도 최적의 성능을 보였다.

본 연구에서는 궁극적으로 self-optimization이 가능한 자율네트워킹의 실현을 목적으로 딥러닝 기반의 강화학습을 적용한 스케줄러를 구현하였다. 타임슬롯에 따라 변화하는 다양한 네트워크 시뮬레이션을 통해 학습하여 dynamic한 스케줄링을 구현할 수 있었다. 강화학습은 Markov Decision Process (MDP)로 모델링 될 수 있는 네트워크 제어에 뛰어난 효과를 보인다. 네트워크에 딥러닝이 적용된 여러 연구 사례에서 네트워크 스케줄링과 같이 시간에 따라 상태가 달라지고, 달라진 상태에 맞는 행동을 해야 하는 문제에서 강화학습을 이용하였을 때 좋은 결과를 얻었음을 확인하였다. 본 연구에서는 네트워크 제어요소 중 우선순위 기반의 타임슬롯의 스케줄링알고리즘을 대체할 수 있는 DDQN 강화학습 에이전트를 적용할 것을 제안한다.

Ⅱ. 관련 연구

2.1 Background

본 연구는 IEEE TSN에서 제안한 소규모 네트워크를 위한 지터 및 지연시간 최소화 기술 표준[7]의 환경 등을 참고하여 flow의 우선순위(priority)에 따라 동기화된 타임슬롯 기반 스케줄링 환경을 가정한다. 우선순위는 기본적으로 플로우의 class에 따라 결정된다. 또한 군용 통신망 등에서 사용되는 precedence의 개념을 도입하여, 같은 class의 flow에서도 사용자의 권한에 따라 다른 우선순위를 가질 수 있다고 가정한다[11]. 즉, flow들은 같은 class이고 비슷한 수준의 요구사항을 가질 수 있으면서 다른 precedence를 가짐으로써 서로 다른 우선순위 큐에 인입되어 다르게 스케줄될 수 있다.

2.2 강화학습

강화학습은 환경 속에 놓인 학습 주체 agent가 무작위의 행동을 선택하는 시행착오를 겪어나가며, 최대의 보상을 받을 수 있는 행동을 학습하면서 최적의 정책에 수렴하는 과정이다. 이러한 과정을 위해서는 여러 횟수의 시뮬레이션이 필요하다. 시뮬레이션 시작부터 종료까지를 episode라 한다. 에이전트와 환경은 timestep 마다 정보를 주고받으며 학습을 진행하고, 여러 timestep이 지나 시뮬레이션이 종료되면 다음 episode로 넘어가 처음부터 시뮬레이션을 진행하게 된

다. agent가 현재 상황에서 얻을 수 있는 정보인 state를 관측한 후 action을 선택하면 환경은 다음 timestep에서 agent에게 reward, action으로 인해 변화한 새로운 state를 전달한다. 이 reward는 agent가 action을 선택함으로써 환경에서 계산한 뒤 주어지는 보상 값이기 때문에 즉각적인 보상이라고도 한다. 강화학습의 대표적인 알고리즘 Q-learning은 agent의 state와 action에 따른 누적 가치를 수치로 나타낸 Q-value를 사용해 학습하는 알고리즘이다. Q-value는 즉각적인 보상뿐만 아니라 timestep t이후 받을 모든 보상들을 고려해 측정한다. t시점에서 state에 따른 action 선택 후, 보상이 주어지는 시점은 $t+1$ 이다. 그러므로 $R(t+1)$은 t시점에 받은 reward가 된다. 미래에 받을 보상의 가치는 현재 시점에서 환산하였을 때 그 가치가 감소하므로, t시점에서의 state-action 쌍 (s,a)의 가치를 환산하기 위해서 0에서 1사이의 γ값인 할인율(discount factor)을 사용해 수식 (1)과 같이 현재의 가치로 변환하게 된다.

$$Q(s,a)=R(t+1)+\gamma R(t+2)+\cdots+\gamma^{n-1}R(t+n) \quad (1)$$

$$\max_a Q(s',a)=R(t+2)+\cdots+\gamma^{n-1}R(t+n) \quad (2)$$

$t+1$이후 가장 큰 Q-value를 갖는 행동을 선택한다고 하면 (2)와 같이 표현할 수 있다. (1)에서 (2)에 해당하는 부분을 치환하면 최종적으로 Q-value 수식을 아래의 (3)과 같이 유도할 수 있다.

$$Y_t = R(t+1)+ \gamma\max_a Q(s',a) \quad (3)$$

수식 (3)에서 $\max Q(s',a)$는 다음 state인 s'에서 가장 큰 보상을 받는 a를 선택할 때 주어지므로 수식 (4)로 표현할 수 있다.

$$\max_a Q(s',a) = Q(s', \arg\max_a Q(s',a)) \quad (4)$$

Q-value를 계산하기 위해서는 t 이후 받는 미래의 보상들을 전부 알아야 하므로, Q-learning에서는 s를 행, a를 열로 하는 2차원 행렬 Q-table이 도입되었다. 행렬의 각 요소들인 $Q(s,a)$를 업데이트하기 위해 여러 번의 시뮬레이션이 필요하다. 위에서 언급한 바와 같이 (4)로 (3)을 계산해 Q-table을 업데이트하므로 state나 action의 조합이 다양해지면 Q-table의 크기가 커지고 연산량이 방대해지게 된다. 이러한 한계를 해결하기 위해 DQN은 Q-value를 추정하는 딥러닝 신경망 Q-network으로 강화학습의 새로운 문을 열었다[8]. 일반적인 딥러닝은 input을 신경망의 파라미터들로 연산하고 activation function을 통과하는 단계들을 거쳐 output을 얻어낸다. 이때 output은 신경망이 예측한 값이기 때문에 맞춰야 하는 target이 존재한다. target과 예측값의 오차를 역전파하여 output을 target에 가깝도록 신경망을 update하는 것을 딥러닝 학습이라고 한다. Q-network의 input은 state s, output은 선택할 수 있는 모든 a에 대한 Q-value가 된다. 수식(3)에서 Y는 t에서의 즉각적인 보상 $R(t+1)$과 $\max Q(s',a)$로 계산할 수 있으며 이는 target이 된다. Y와 신경망이 예측한 $Q^{pred}(s,a)$과의 오차를 통해 학습을 진행할 수 있다. 이때, Q-network를 통해 target의 $\max Q(s',a)$를 예측하면 target은 학습이 진행될 때마다 변하게 된다. 일반적으로 지도학습의 경우 고정된 target이 존재해 이러한 문제가 발생하지 않지만, DQN은 target에 해당하는 값이 변하기 때문에 학습이 제대로 이루어지지 않을 가능성이 있다. 이는 Q-network와 동일한 구조의 target network로 $\max Q(s',a)$를 계산해 주기적으로 Q-network의 학습된 신경망의 가중치들로 동기화하는 soft update 방법을 사용해 해결한다. 이를 Double DQN (DDQN)이라고 하며 그 학습 과정은 그림 1과 같다.

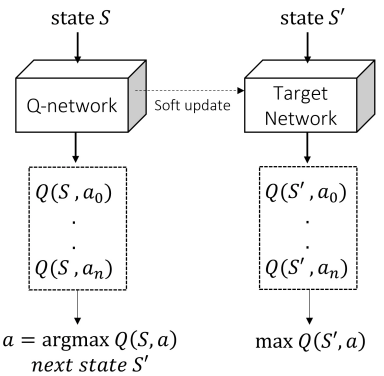

그림 1. DDQN 학습 과정
Fig. 1. Training process of DDQN

Ⅲ. 시스템 모델링

3.1 타임슬롯 스케줄링

본 연구에서 사용하는 네트워크 node의 구조는 그림 2와 같다. DDQN agent는 node에 존재하는 두 개의 우선순위 큐들의 타임슬롯 스케줄링을 담당한다.

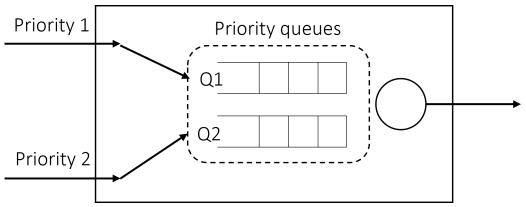

그림 2. 네트워크 노드의 구조
Fig. 2. Node architecture in Network

output port는 한 개이다. 여러 node들이 존재하는 network topology 상의 node임을 감안하여, 인입되는 패킷들이 출발부터 거쳐온 hop 수를 다양하게 설정하였다. 패킷이 source에서 출발해 여러 번의 hop을 거치면서 겪은 delay를 current delay라 표현하고, destination까지 남은 hop을 remaining hop count로 표현하였다. current delay와 remaining hop count는 임의의 값으로 설정하였고 이는 아래의 표3에서 확인할 수 있다. 모든 패킷은 고정된 크기를 가지며, timeslot의 크기는 패킷 1개가 전송될 수 있는 시간으로 설정하였다.

두 우선순위의 패킷 모두 deadline이 존재하며, deadline을 만족하여야 한다. 다만 그 중요도에 차이가 있다. SP는 우선순위가 높은 패킷이 존재하는 경우 항상 해당 queue를 서비스하기 때문에 테스트 결과 utilization이 높은 상황에서는 priority2 패킷이 deadline 내 전송되지 못하는 상황이 자주 발생하였다. 아래의 표1을 보면 priority1 패킷보다 priority2 패킷들의 도착 확률이 낮다는 것을 알 수 있다. 따라서 모든 우선순위 packet들이 그 중요도에 따라 적절히 deadline 내 도착하는 것을 목표로 강화학습을 적용하였다. 강화학습 에이전트가 최적의 스케줄링을 구현하기 위해서 높은 우선순위의 패킷을 조금 지연시키되 요구사항은 만족시켜야 하고, 낮은 우선순위의 패킷을 더 보내주되 높은 우선순위의 패킷들을 먼저 만족시킬 수 있는 수준까지만 보내야 한다. 이는 매우 섬세하게 동작하는 알고리즘이 필요하며 환경이 바뀜에 따라 알고리즘 또한 바뀌어야 한다는 점에서 자율적으로 적응할 수 있는 강화학습 기반 스케줄링의 필요성이 강조되는 부분이다.

표 1. SP 환경에서 priority 별 deadline 내 패킷이 도착할 확률
Table 1. Probability of meeting deadline of each priority with SP scheduler

	priority 1	priority 2
도착 확률	100%	80%
평균 추정 E2E 딜레이	2ms	42ms

3.2 DDQN 구현과 적용

DQN에서는 Q-network를 통해 현재 state에서 가장 좋은 action을 선택한다. $Q^{pred}(s,a)$는 Q-network의 output이며 Q-network의 파라미터들은 θ^{Q}으로 표현한다. 마찬가지로 $Q(s',a)$ 은 target network에서 파라미터 θ^{target}를 사용한 output이다. target은 수식(5)로 표현할 수 있다. target과 $Q^{pred}(s,a)$의 제곱오차를 계산해 아래 수식(6)과 같이 딥러닝 학습에 사용되는 Loss function을 얻을 수 있다.

$$Y_t = R(t+1) + \gamma \max Q(s',a;\theta^{target}) \quad (5)$$

$$Error = \left(Y_t - \ Q^{pred}(s,a;\theta^{Q})\right)^2 \quad (6)$$

학습 효과를 높이기 위해 Epsilon-greedy 알고리즘을 사용하였다. 이 방법은 1부터 0.01까지 점차 감소하는 확률값 ε을 사용하여, agent가 ε에 따라 action을 random으로 선택한다. 에피소드를 거듭할수록 ε는 감소하기 때문에 점점 최대 Q-value를 만들어내는 action을 선택하게 된다. 이러한 방법을 통해 agent가 다양한 데이터베이스를 구축하고 학습할 수 있다. timestep마다 하나의 샘플 데이터$\langle s,\ a,\ r,\ s' \rangle$ 를 관측해 메모리에 저장하고 모든 데이터를 mini-batch로 나눠 신경망 학습을 진행한다. DDQN은 DQN, Q-learning에서 Q-value가 과대평가 되는 문제를 해결하기 위해 제안되었다[9]. 더 정확한 Q-value를 예측함으로써 더 높은 Reward에 안정적으로 도달하는 결과를 보였다. DQN과의 차이점은 target을 계산하는 데에 있다. DQN의 경우 수식 (5)와 같이 θ^{target}만을 가지고 다음 state의 action을 선택한다. 반면 DDQN은 θ^{Q}를 가지고 next state에서의 action을 선택하고, θ^{target}에서 선택한 action의 Q-value에 해당하는 output을 대입한다. 수식으로 나타내면 (7)과 같다.

$$Y_t = R(t+1) + \gamma Q(s', \arg\max_a Q(s',a;\theta^{Q});\theta^{target}) \quad (7)$$

이는 과대평가된 Q-value로 propagation이 일어나는 것을 방지할 수 있다고 알려져 있다. state와 action, reward는 아래와 같이 정의하였으며 해당 요소들의 정의에 사용된 기호는 표 2에 설명하였다.

표 2. DQN 수식에 사용한 기호
Table 2. Symbols used in DQN formulation

symbol	description
h	목적지까지 남은 hop 수
c	패킷이 현재까지 겪은 current delay
qd	패킷의 queueing delay
p	우선순위 $p \in \{1,2\}$
e_p	우선순위 p인 큐에서 전송한 패킷의 추정 End-to-End delay
E_p	우선순위 p인 큐에서 대기하는 패킷들의 추정 End-to-End delay 집합
L_p	우선순위 p의 큐 길이
d_p	우선순위 p인 큐에서 전송한 패킷 deadline
ω_p	패킷 전송 여부 $\omega_p \in \{0,1\}$

3.2.1 State

state는 최대한 간결하면서도 강화학습 환경이 알아야 할 노드의 상황을 모두 담을 수 있는 정보를 고려하였다. 타임슬롯 단위의 End-to-End(E2E) delay를 패킷의 current delay와 remaining hops로 계산하여 노드에서 전송되었을 때 패킷의 deadline 보장 여부를 추정할 수 있다. remaining hops의 경우, 타임슬롯 기반 전송 환경에서 패킷 1개가 전송되는 데 1슬롯의 시간이 소요됨을 가정한다. 따라서 E는 타임슬롯 단위를 가진다.

$$e_p = h + c + qd$$
$$e_p \in E_p$$
$$s_t = [L_1, L_2, \max(E_1), \max(E_2)]$$

3.2.2 Action

하나의 timeslot 당 1개의 패킷이 나갈 수 있는 상황을 가정했으므로, agent가 선택할 수 있는 action은 priority1 패킷을 보낼지, priority2 패킷을 보낼지 결정하는 것이다. agent는 timeslot 마다 0 또는 1의 action을 선택해 0은 priority1 패킷 전송, 1은 priority2 패킷 전송을 의미한다.

$$a_t \in \{0,1\}$$

3.2.3 Reward

Reward는 강화학습 agent에 기대하는 행동을 유도하도록 설정해야 한다. 최대한 많은 패킷을 deadline 안에 보내는 것이 목표이기 때문에, $R1$은 e_p가 d_p미만일 경우에 보상을 지급한다. $R1$만 사용했을 때 agent는 패킷을 보냈음에도 불구하고 패킷을 보내지 않은 행동과 같이 0의 보상을 받는 상황을 방지하기 위해 α보다 훨씬 작고 0보다는 큰 가중치 β을 사용하여 패킷을 전송했을 때 무조건 받을 수 있는 $R2$를 정의하였다. 최종 reward는 R_t이다.

$$R1_t^p = 1\{d_p > e_p\}$$
$$R2_t^p = 1\{\omega_p\}$$
$$R_t = \alpha \sum_{i=1}^{p} R1_t^i + \beta R2_t^p$$

Ⅳ. 시뮬레이션 결과

네트워크 시뮬레이션은 강화학습 시뮬레이션 플랫폼인 Simpy를 사용하였다. Simpy에서 정의된 시뮬레이션 환경 외부에서 DDQN학습이 진행되므로 네트워크 환경 내의 패킷 지연시간이나 시뮬레이션의 소요시간에는 영향을 주지 않는다. 다시 말해 DDQN 스케줄러를 학습하는데 딥러닝 모델의 추론 및 학습 연산으로 인한 delay가 발생하나 타임슬롯 기반 가상환경에서의 간섭은 존재하지 않는다. 그러나 실제 네트워크 환경에서 ms 단위의 타임슬롯마다 스케줄링이 이루어져야 하는 환경에서는 딥러닝 추론 연산 시간에 의한 문제가 발생할 수 있다. 연구에서는 이러한 상황을 대비하여 네트워크 내에 존재하는 모든 스위치 node들의 상황을 알지 못해도 node가 자기 자신의 상태만 파악할 수 있으면 행동을 선택할 수 있도록 설계하였다. 학습된 DDQN의 역할은 이러한 상태를 입력으로 하여 최적의 action을 출력해주는 역할을 하므로, node에서 발생하는 상태에 따른 최적 action을 lookup-table로 구현하여 CNC(Central Network Controller)와의 통신 없이 자체적으로 간단하게 스케줄링할 수 있다. 1 episode는 모든 패킷의 전송이 완료되어 더는 전송할 패킷이 없을 때까지 진행된다. 아래에 서술할 DDQN 학습 결과를 보이기 위해 설정한 파라미터는 표3의 네트워크 시뮬레이션 파라미터와 표 4의 DDQN 학습 파라미터로 정리하여 나타냈다. DDQN의 모델 구조는 그림 3에서 확인할 수 있다. 최종 출력단의 activation function은 linear, optimizer는 Adam을 사용하였다. 결과 비교로 사용한 알고리즘은 SP와 Weighted round robin(WRR)이 있다. SP는 가장 높은 우선순위 큐에 있는 패킷을 무조건 먼저 내보내는 알고리즘이다. Weighted round robin은 모든 각 queue에 할당된 weight에 비례해서 순차적으로 queue를 서비스하는

표 3. 학습에 사용한 네트워크 파라미터
Table 3. Network parameters for training model

parameter	value
1episode에 전송되는 priority1 패킷의 수	40
1episode에 전송되는 priority2 패킷의 수	100
priority1의 deadline	5ms
priority2의 deadline	50ms
패킷 크기	1500byte
timeslot size	0.6ms
bandwidth	20Mbps
random h의 범위	0~4
priority1의 random c의 범위	0~2 slots
priority2의 random c의 범위	30~45 slots
priority1 패킷 생성 주기	1 slot
priority2 패킷 생성 주기	1 slot

알고리즘이다. 연구에서는 priority1, 2 각각에 3:1의 weight를 주었다. DDQN을 포함한 모든 시뮬레이션에서 1개의 우선순위 큐에만 패킷이 존재하는 경우에는 스케줄링의 결과와 상관없이 그 패킷을 내보내는 work conserving을 적용하였다.

표 4. 학습에 사용한 DDQN 파라미터
Table 4. DDQN parameters for training model

parameter	value
총 episode 횟수	20000
최대 timeslots	330 slots
update 빈도	500 episode
reward1 weight (α)	[0.6, 0.1]
reward2 weight (β)	0.01
discount factor (γ)	0.99
learning rate	1e-4

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                320
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130
=================================================================
Total params: 4,610
Trainable params: 4,610
Non-trainable params: 0
_________________________________________________________________
```

그림 3. 딥러닝 네트워크 구조
Fig. 3. Neural network model summary

4.1 학습 결과

하이퍼파라미터 튜닝을 위해 여러 번의 학습을 진행해본 결과, α는 priority1에 할당되는 값이 클수록 학습 효과가 개선되는 것을 확인할 수 있었다. β는 α보다 영향력이 작아야 하고, 0보다는 커야 하므로 0.01의 값을 할당하였다. 그림 4는 20000번의 episode로 DDQN학습을 진행한 결과를 10000 episode의 DQN 학습 결과와 SP결과와 비교한 학습 곡선이다. 결과 지표는 episode 동안의 각 step에서 받은 reward의 누적합인 score를 사용하였다. score는 window size 1000으로 이동평균을 구해 나타냈으며, 이동표준편차를 투명도 30%의 범위로 나타냈다. 각 score는 episode에서 최대로 받을 수 있는 score값과 0으로 정규화하여 범위를 0~1사이로 조정하였다. ε가 감소함에 따라 DDQN의 score가 상승하며 제대로 학습이 진행되고 있음을 알 수 있다. 그림 5를 보면 약 15000회의 episode 동안 학습한 이후에는 기존 알고리즘인 SP의 score보다 높은 값의 score를 기록한 것을 볼 수 있다. 손실함수 또한 0에 가깝게 점점 수렴함을 확인하였다.

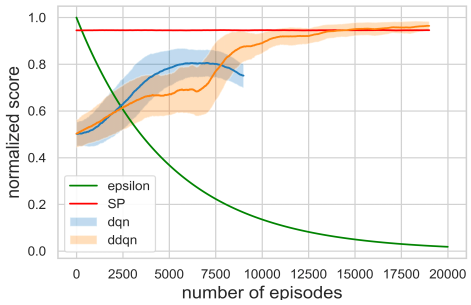

그림 4. 20000회의 에피소드 동안의 학습 곡선
Fig. 4. Learning curves during 20000 episodes

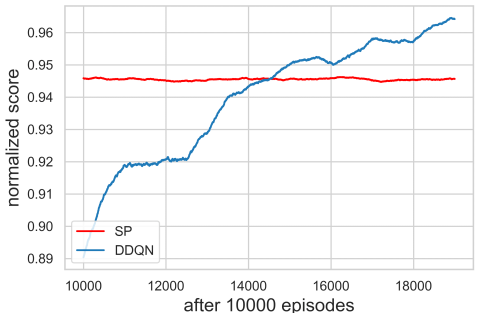

그림 5. 10000 에피소드 이후 학습 곡선
Fig. 5. Learning curves after 10000 episode

4.2 single node 테스트 결과

학습된 딥러닝 모델과 기존 알고리즘의 비교를 그림 6으로 나타냈다. 모든 알고리즘은 10회씩 테스트하여 데이터들의 분포를 볼 수 있도록 하였다. 또한 priority1, 2 패킷이 1 slot 주기로 생성되기 때문에 200%의 utilization을 갖는 학습 환경과 달리, 생성 주기를 모두 2 slot으로 설정해 테스트하였다. deadline은 모두 7 slots으로 고정하고 패킷의 개수도 각각 40개로 설정하였다. remaining hops가 클수록 적은 current delay를 갖는다. 그림 6에서 확인할 수 있듯 파라미터를 바꾸었음에도 학습 후 DDQN은 normalized score가 100%에 육박하는 성능을 보여 여러 환경에 잘 적응함을 증명하였다. 반면 기존 알고리즘들은 90% 초반 정도의 성능에 머무르는 것을 확인할 수 있다. 이는 DDQN이 다른 알고리즘들과 다르게 우선순위가 존재하는 타임슬롯 스케줄링에서 더 많은 패킷의 deadline 요구사항을 보장할 수 있음을 보여준다.

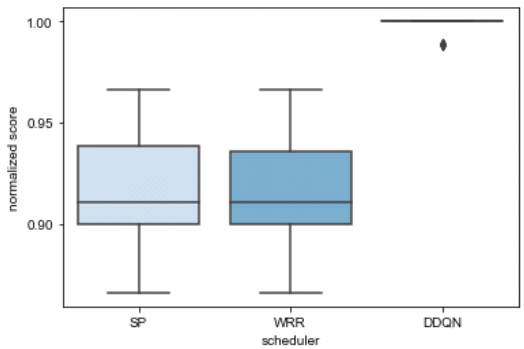

그림 6. SP, WRR, DDQN스케줄러의 score 비교 박스플롯
Fig. 6. Boxplot for comparing scores of schedulers

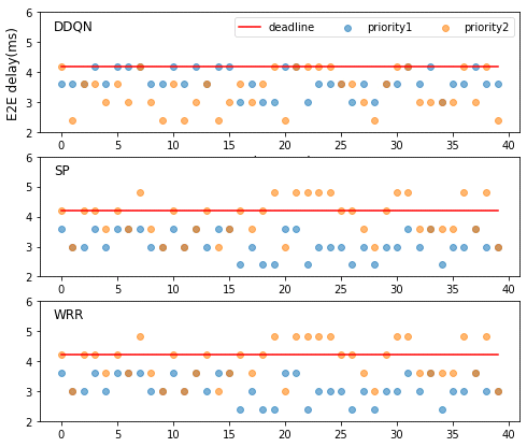

그림 7. DDQN, SP, WRR 스케줄러의 패킷별 E2E delay
Fig. 7. E2E delay of DDQN, SP, WRR

Deadline의 보장 여부뿐 아니라 스케줄링의 목적인 낮은 delay의 보장을 확인하기 위해 그림 7에서 각 알고리즘별, 우선순위 별 delay를 확인하였다. 빨간 선은 deadline을 의미한다. 연구의 목적과 의도에 맞게 priority1의 delay는 소폭 증가했으나 deadline을 맞출 수 있었고, priority2의 delay는 감소하여 다른 알고리즘들과 달리 모든 packet이 deadline 이내에 도착하는 것을 확인하였다.

4.3 네트워크 확장 테스트 결과

4.2에서는 single node 환경을 전제로 다양한 파라미터에도 의도에 맞게 동작하는 것을 확인하였다. 이번에는 DDQN 스케줄러가 네트워크 규모를 확장하였을 때에도 잘 작동하는지 확인하기 위해 그림8과 같은 9개 노드가 있는 mesh형 네트워크 토폴로지를 가정하고, 표 5에 나타낸 바와 같이 2개의 우선순위를 갖는 8개의 flow를 설정한 후 같은 모델을 적용해 다양한

표 5 . 확장 토폴로지 내 8개의 flow
Table 5. The Flows in network topology

Flow	Priority	Route([nodes])
F1	1	[1, 2, 5, 6, 9]
F2	2	[3, 2, 5, 4, 7]
F3	2	[4, 1, 2]
F4	2	[4, 7, 8]
F5	2	[6, 3, 2]
F6	2	[6, 9, 8]
F7	2	[7, 8, 5, 6, 3]
F8	1	[9, 8, 5, 4, 1]

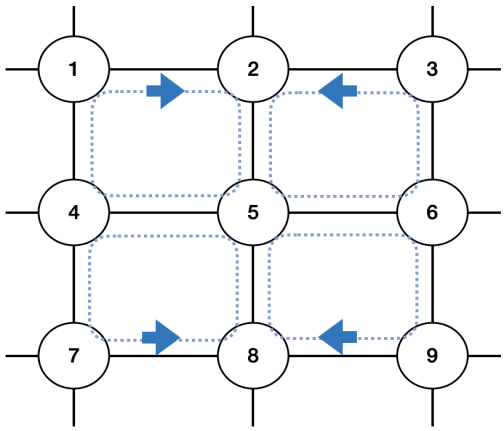

그림 8. 확장한 네트워크 토폴로지
Fig. 8. Expanding network topology

시나리오 하에서 스케줄러 간 성능을 비교하고 마찬가지로 delay를 분석하였다.

시나리오는 아래 표 6과 같이 총 6개의 시나리오로 구성하였다. deadline, period는 각각 priority 1,2에 대하여 튜플로 나타내었으며, T는 timeslot을 의미한다. 시나리오를 바탕으로 DDQN, SP, WRR 스케줄러를 테스트한 결과는 그림 9, 10, 11과 같다. delay 추정치인 ET를 사용해 single node에서 학습한 모델로도 100%의 deadline 만족도를 보이는 시나리오가 있으며, 전반적으로 DDQN의 성능이 더 높은 것을 확인할 수 있다. SP의 경우 priority1에 있어서는 100% deadline을 만족하나 priority2의 deadline을 제대로 보장하지 못한다. WRR의 경우 work conserving으로 인해 패킷

표 6. 시나리오별 파라미터
Table 6. Parameters of each scenario

Scenario	Period	Deadline	Flows
S1	(2T, 2T)	(8T, 7T)	All flows
S2	(2T, 2T)	(7T, 7T)	All flows
S3	(2T, 2T)	(7T, 8T)	All flows
S4	(2T, 2T)	(7T, 8T)	Without F3~F6
S5	(1T, 5T)	(8T, 9T)	Without F3~F6
S6	(2T, 4T)	(6T, 7T)	Without F3~F6

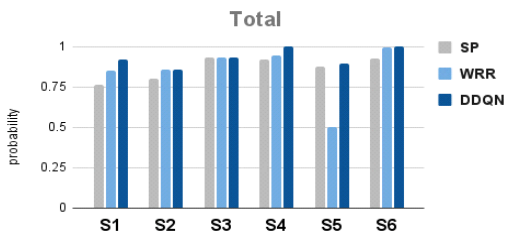

그림 9. 시나리오별 deadline 내 패킷 도착 확률
Fig. 9. Probability of packet arrival within deadline for each scenario

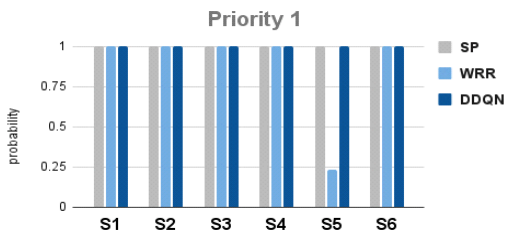

그림 10. 시나리오별 deadline 내 priority1 패킷 도착 확률
Fig. 10. Probability of priority1 packet arrival within deadline for each scenario

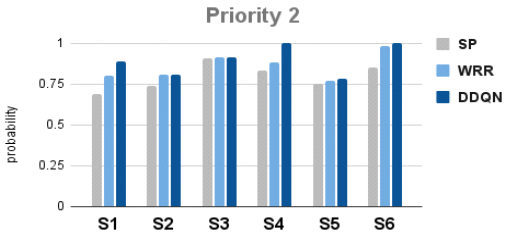

그림 11. 시나리오별 deadline 내 priority2 패킷 도착 확률
Fig. 11. Probability of priority2 packet arrival within deadline for each scenario

의 생성 주기가 2 slot 이상으로 긴 경우 SP와 비슷한 스케줄링을 하게 된다. 앞서 언급한 바와 같이 3:1의 weight를 갖고 있으므로 생성 주기가 크게 달라진 S5 같은 환경에서 매우 낮은 deadline 만족도를 보인다. 이는 WRR을 사용할 경우 변화하는 환경에 따라 weight도 변경해주어야 함을 뜻한다. 제안하는 DDQN 스케줄러의 경우, SP와 WRR의 단점을 성공적으로 극복한 결과를 보인다.

V. 결 론

하위 priority의 트래픽에도 deadline이 요구되는 상황에서, 상위 priority가 적절한 시기에 하위 priority에게 양보하면 더 많은 packet을 deadline 안에 전송할 수 있을 것이라는 가설을 세우고, packet의 추정 E2E delay가 deadline보다 작을 때 reward를 주도록 DDQN 학습을 진행하였다. 약 15000번의 episode를 지난 후 DDQN의 score가 SP를 능가하는 학습 곡선을 보였다. 학습 환경보다 작은 utilization의 환경하에서 테스트한 결과 SP나 WRR은 90% 수준, DDQN은 100% 수준의 score를 보였다. 이는 DDQN이 기존 알고리즘보다 더 많은 packet의 deadline을 보장할 수 있음을 의미한다. 네트워크 구조를 확장해 테스트한 결과에서도 DDQN은 기존 알고리즘들의 단점을 보완하면서 항상 같거나 더 높은 성능을 보였다. DDQN 스케줄링을 적용한 패킷들의 E2E delay는 deadline 안에 들어올 수 있도록 조정된 것을 확인하였다. 강화학습을 사용하면 관리자의 개입 없이도 방대한 데이터가 전송되는 네트워크의 요구사항을 지키기 위해 최적의 스케줄링을 제공하고, 네트워크 환경 변화에 즉각적으로 반응하는 자율네트워킹의 목표에 도달할 수 있을 것이다. 본 연구에서는 미래의 지능형 네트워크에 딥러닝 도입 가능성을 보였다. 추후 연구에서는 DDQN보다 더욱 성능이 좋다고 알려진 Dueling DDQN,

Rainbow 등과 같은 여러 DQN 기반 최신 알고리즘을 활용하여 좀 더 다양한 우선순위에 적용할 수 있도록 일반화할 예정이다. 그와 더불어 IEEE TSN의 timeslot 기반 스케줄링 환경 등 다양한 표준에 적용할 수 있는 딥러닝 기반 네트워크 스케줄링에 관한 연구를 진행할 계획이다.

References

[1] M. Behringer, et al., "*Autonomic Networking: Definitions and Design Goals*," RFC7575, pp. 1-16, 2015.

[2] C. Zhang, et al., "Deep learning in mobile and wireless networking: A survey," in *IEEE Commun. Surv. & Tuts.*, vol. 21, no. 3, pp. 2224-2287, thirdquarter 2019.

[3] Sandeep P. Chinchali, et al., "Cellular network traffic scheduling with deep reinforcement learning," *Thirty-second AAAI Conf. Artificial Intell.*, 2018.

[4] X. Xiong, et al., "Resource allocation based on deep reinforcement learning in iot edge computing," in *IEEE J. Sel. Areas in Commun.*, vol. 38, no. 6, pp. 1133-1146, Jun. 2020.

[5] H. Wei, et al., "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, pp. 2496-2505, 2018.

[6] G. Zheng, et al., "Diagnosing reinforcement learning for traffic signal control," *arXiv preprint arXiv:1905.04716*, 2019.

[7] J. Joung and J. Kwon, "Zero jitter for deterministic networks without time-synchronization," in *IEEE Access*, vol. 9, pp. 49398-49414, 2021.

[8] V. Mnih, K. Kavukcuoglu, et al., "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, 2013.

[9] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," in *AAAI'16*, vol. 30, no. 1, pp. 2094-2100, 2016.

[10] M. Joo, W. Jang, and W. Lee, "Deep reinforcement learning based multipath packet scheduling," *J. KIISE*, vol. 46, no. 7, pp. 714-719, 2019.

[11] Y. Xue, C. Gedo, C. Christou, B. Liebowitz, "A framework for military precedence-based assured services in GIG IP networks," *IEEE MILCOM*, pp. 1-7, 2007.

류 지 혜 (Jihye Ryu)

2020년 8월 : 상명대학교 휴먼지능정보공학과 학사

2020년 9월~현재 : 상명대학교 지능정보공학과 석사

<관심분야> 네트워크, 강화학습, 딥러닝

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]