논문 20-45-05-09 The Journal of Korean Institute of Communications and Information Sciences '20-05 Vol.45 No.05
https://doi.org/10.7840/kics.2020.45.5.820

SDN에서 강화학습을 통한 멀티캐스트 트리 생성 기법 분석

채 지 훈◆, 김 남 기°, 이 병 대*

The Analysis of Multicast Tree Construction Scheme Using Reinforcement Learning in SDN

Jihun Chae◆, Namgi Kim°, Byoung-Dai Lee*

요 약

최근 비디오 트래픽과 같은 멀티미디어 데이터를 빠르게 주고 받아야 하는 OTT(Over The Top) 서비스, 비디오 플랫폼 서비스들의 활성화로 효과적인 멀티캐스트 전송 방식에 대한 요구가 높아지고 있다. 그 결과 중앙 집중적 구조를 가지는 SDN(Software-Defined Network)에서의 멀티캐스트 전송 방식에 대한 연구가 진행되었다. SDN 환경에서의 멀티캐스트 트리 생성은 스타이너 트리 생어의 문제로 귀결되며 NP-hard의 복잡도를 가진다. 따라서 본 논문에서는 강화학습 기법을 적용한 휴리스틱 멀티캐스트 트리 생성 기법을 소개하고 그 성능을 분석하다. 이를 위해 우리는 다양한 밀집도와 목적지 노드를 가지는 토폴로지 상에서 멀티캐스트 생성을 실험하였다. 분석 결과 토폴로지 링크의 밀집도가 증가할수록 생성되는 멀티캐스트 트리의 길이가 증가하고 도착지 노드의 개수가 늘어날수록 최적 멀티캐스트 트리의 길이와의 차이가 감소하는 것을 알 수 있었다.

Key Words : SDN(Software-Defined Network), Multicast tree, NP-hard, Machine Learning, Reinforcement Learning, MDP(Markov Decision Process)

ABSTRACT

Recently, the activation of OTT service (Over The Top) and video platform services, which require rapid transfer of multimedia data such as video traffic, has raised the need for effective multicast transmission. As a result, The research on the method of multicast transmission has been conducted in SDN(Software-Defined Network) having a centralized structure. The generation of multicast tree in SDN has the same complexity as creating a Steiner tree and is a problem with NP-hard. Therefore, in this paper, we introduced a technique for heuristic multicast tree construction and analyzed the performance. In addition, we experiment multicast tree generation on each topology according to the multiple density of topology links and the number of destination nodes. The result of analysis showed that the length of the generated multicast tree increased as the density of topology link increase and the difference of length of the optimal multicast tree decreased as the number of destination of nodes increased.

※ 본 연구는 2020년 경기대학교 대학원 연구원장학생 장학금 지원에 의하여 수행되었습니다.
※ 본 연구는 한국연구재단 논문연구과제(NRF-2017R1D1A1B04027874) 지원을 받아 수행되었습니다.
◆ First Author : Kyonggi University Department of Computer Science, blesk0111@gmail.com, 학생(석사), 학생회원
° Corresponding Author : Kyonggi University Department of Computer Science, ngkim@kyonggi.ac.kr, 교수, 정회원
* Kyonggi University Department of Computer Science, blee@kgu.ac.kr, 교수
논문번호 : 202001-005-B-RN, Received January 9, 2020; Revised February 18, 2020; Accepted February 24, 2020

Ⅰ. 서 론

최근 기업, 통신 사업 및 멀티미디어 서비스의 확대, IPTV, 스마트TV 등과 같은 방송 통신 융합 서비스의 발전으로 인해 서비스 가입자와 인터넷 트래픽량이 기하급수적으로 증가하고 있다[1]. 이러한 이유로 여러 서비스에서 효과적이고 유연한 멀티미디어 트래픽 전송 방식에 대한 요구가 높아지고 있다. 멀티미디어 서비스의 효과적인 전송을 위해서 기존의 네트워크 구조에서는 분산되어 있는 네트워크 장비의 수정이 필요하다. 하지만 기존의 분산적이고 폐쇄적인 네트워크 구조는 새로운 프로토콜의 구현이 어려웠으며 네트워크 장비 회사에서 제공하는 기능만을 사용해야 하는 문제가 존재했다[2]. 이후, 네트워크 관리 측면에서 변화하는 네트워크 트래픽 및 구조에 맞추어 개방성이 요구됨과 동시에 이를 관리 및 유지 가능하도록 하는 기술의 필요성이 대두되어 SDN (Software-Defined Network) 기술이 제안되었다. 멀티캐스트 전송 기법에서 네트워크 노드들은 멀티캐스트 트리를 구성하게 된다. 하지만 분산된 네트워크 구조에서는 멀티캐스트 트리 생성은 최적의 트리를 생성하기 어렵고 지역적 최적에 집중하게 된다. 따라서 네트워크 구조가 중앙 집중화되어 있는 SDN에서는 멀티캐스트 트리의 전역적 최적 문제를 다루는 연구들이 진행되었다[3,4]. SDN 환경에서 멀티캐스트 트리 생성은 스타이너 트리 생성 문제로 대표될 수 있다[5]. 하지만 멀티캐스트 트리 생성을 위한 스타이너 트리의 생성은 NP-hard의 문제로 증명되었다[6]. 그러므로 네트워크 노드의 개수가 증가할수록 최적의 멀티캐스트 트리를 생성하는 것은 어렵다. 이를 해결하기 위해 근사 알고리즘을 활용하여 멀티캐스트 트리를 생성하려는 연구들이 존재하였지만 만족스러운 최적의 멀티캐스트 트리를 생성하지는 못하였다[7]. 본 논문에서는 머신 러닝의 한 갈래인 강화 학습에서의 DQN(Deep-Q-Network) 기법을 적용하여 멀티캐스트 트리를 생성하는 기법을 소개하고 그 성능을 분석하는 실험을 진행하였다. 실험에서는 서로 다른 링크 밀집도를 가지는 여러 토폴로지들을 생성하여 멀티캐스트 트리를 생성하고 그 결과를 분석하였다. 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구인 SDN, 스타이너 트리의 근사 알고리즘과 강화학습에 대해서 설명하고, 3장에서는 DQN 기법을 이용한 멀티캐스트 트리 생성에 대해서 설명한다. 4장에서는 실험에 대해서 설명하고, 5장에서 실험결과를 분석한다. 마지막으로 6장에서 향후 연구 및 결론을 맺는다.

Ⅱ. 관련 연구

기존의 네트워크 구조에서는 하나의 네트워크 장비 내에 데이터 전송을 담당하는 데이터 평면과 데이터를 어디로 보낼지를 결정하는 라우팅 제어 부분인 제어 평면이 함께 존재한다. 이로 인해 네트워크 정보가 분산되어 있는 현재의 네트워크 구조가 만들어졌을 뿐만 아니라 네트워크 장비를 생산하는 벤더 회사 마다 네트워크 장비 제어 방식이 공개되지 않아 폐쇄적인 특성까지 지니게 되었다. SDN은 이와 같은 문제점을 해결하기 위해 제어 평면과 데이터 평면을 분리하면서 소프트웨어 어플리케이션을 사용하여 중앙의 컨트롤러에서 제어 및 프로그래밍이 가능하도록 하는 구조를 가진다. 따라서 SDN 환경에서 멀티캐스트 트리를 생성할 때 SDN 환경에서 분산되어 있는 네트워크 정보를 중앙의 컨트롤러가 관찰할 수 있기 때문에 SDN 컨트롤러가 전체적인 네트워크 정보를 고려하여 멀티캐스트 트리를 생성 하는 방식은 기존 네트워크 구조에서보다 효과적이다. SDN 컨트롤러가 멀티캐스트 트리를 만드는 것은 그래프를 구성하는 노드와 링크 정보를 통해서 트리를 구성하는 것이므로 이는 스타이너 트리를 생성하는 문제로 표현될 수 있다[5]. 따라서 멀티캐스트 트리를 구성하는 노드들은 스타이너 트리를 구성하는 부분집합 노드들로 표현 가능하고 SDN 네트워크 내에 존재하는 새로운 노드를 추가하여 스타이너 트리를 생성할 수 있다. 하지만 스타이너 트리 문제는 NP-hard의 문제로 증명되어 있다[6].

$$Performance\ ratio = \frac{Constructed\ Tree\ Length}{Optimal\ Tree\ Length} \quad (1)$$

이 스타이너 트리를 근사하기 위한 여러 연구들이 존재하는데 수식 1과 같이 performance ratio의 값으로 성능이 평가되었다[7]. performance ratio의 값은 근사 알고리즘이 생성한 트리의 길이를 최적의 트리 길이로 나누어 계산하도록 한다. 이러한 기존의 근사 알고리즘들은 표 1과 같이 최적의 트리에 비해 만족스러운 결과를 얻지는 못하였고 최적화 문제에 사용되는 머신러닝 기법을 적용한 연구는 미미한 것이 사실이다.

대부분의 강화학습 기법들은 상태에 대해서 얼마나 좋은지를 가치 함수를 사용하여 추정한다. 가치 함수는 어떠한 정책에 따라 기대 보상 값을 통해 상태를

표 1. 스타이너 트리 근사 알고리즘[7]
Table 1. Steiner tree approximation algorithms.

Year	Performance Ratio	Authors
1980	2.000	Takahashi, Matsuyama
1993	1.834	Zelikovsky
1994	1.734	Berman, Ramaiyer
1995	1.694	Zelikovsky
1997	1.667	Promel, Steger
1997	1.644	Karpinski, Zelikovsky
1998	1.598	Hougrardy, Promel

추정한다. 강화학습 기법 중에서 대표적인 Q-learning 기법은 상태와 행동 쌍에 대한 가치 함수로 q 함수를 사용하고 q 함수의 값을 최대화 하도록 한다. 이러한 q 함수의 값을 Q-table에 저장하여 모든 상태와 행동에 대한 q 함수 값을 가지고 있어 어떠한 정책에 대한 누적 보상 값을 찾을 수 있으므로 최대의 누적 보상 값을 가지는 정책을 구할 수 있다. 하지만 기존의 Q-table을 사용하는 Q-learning 기법은 주어진 상태 공간의 크기가 클수록 커다란 Q-table을 유지해야하고 Q-table의 탐색 시간 또한 증가한다. 이러한 이유로 Q-learning 기법은 SDN 네트워크와 같은 동적인 네트워크 문제에 적용하는 것은 적절하지 않았다. 그러나 최근 DeepMind 사에서 발표한 DQN 기법은 Q-table 대신에 DNN(Deep-Neural-Network)을 사용하여 상태 공간의 제한 및 Non-Stationary 문제와 데이터 연관성(Data Correlation) 문제를 해결하였다[8].

Ⅲ. DQN 기법을 이용한 멀티캐스트 트리 생성

본 논문에서 소개하는 DQN 기법을 적용하여 멀티캐스트 트리를 생성하는 기법은 문제 상황에 대해서 MDP(Markov Decision Process)로 표현해야 한다. 따라서 멀티캐스트 트리 생성 문제는 MDP의 구성요소인 상태, 행동, 보상, 감가율이 다음과 같이 정의된다. 상태는 SDN 컨트롤러가 관측하는 네트워크 노드 연결 정보이고 멀티캐스트 트리를 생성하는 DQN 에이전트는 전달받은 상태 정보를 이용한다. 상태 정보는 SDN 네트워크의 노드 연결 정보를 의미하고 인접 행렬 형태로 표현 가능하다. 인접 행렬 형태의 노드 연결 정보는 전체 노드 개수의 제곱 크기를 가지며 링크 정보가 존재하는 노드와 없는 노드는 각각 1, 0의 값을 갖도록 하였다. 또한 멀티캐스트 트래픽의 출발, 목적 노드의 대각 행렬은 각각 1, -1의 값을 가진다. 행동은 DQN 에이전트가 현재 연결된 트리의 말단 노드에서 이동 가능한 모든 노드들이다. 각 노드마다 번호를 부여하여 연결 가능한 노드들의 번호를 행동으로 설정하였다. 보상은 DQN 에이전트가 모든 목적 노드들을 포함하여 트리를 생성했을 때 부여한다. 따라서 위에서 설정한 목적지 노드의 대각행렬의 -1값을 모두 찾으면 종료상태가 되어 보상이 부여된다. 또한 0~1 사이의 값을 가지는 감가율을 사용하여 미래의 보상에 대해 패널티를 부여함으로써 최단 경로를 찾도록 학습하였다. 감가율은 에이전트의 행동 횟수에 따라 감가율을 거듭제곱하여 보상 값과 곱하기 때문에 연결되는 노드의 개수가 적어질수록 보상 값은 증가하게 된다. 따라서 DQN 에이전트가 최소 비용을 가지는 멀티캐스트 트리를 생성하도록 학습이 가능하다.

DQN 에이전트는 학습 시에 위의 MDP 요소들을 사용하게 된다. 학습 과정은 다음과 같다. 우선 DQN 에이전트는 SDN 컨트롤러로부터 네트워크 노드 연결 정보를 전달받는다. 인접행렬 형태인 노드 연결 정보를 신경망에 입력으로 전달하여 연결 가능한 노드들에 대한 q 함수 값을 결과로 얻는다. 연결 가능한 노드들을 선택하는 방법은 탐험과 탐구의 문제로 본 논문에서는 decaying e-greedy 방법을 사용하여 행동을 선택하도록 하였다. 학습 초반에는 탐험의 비중을 높여 가능한 행동들 중에서 무작위로 선택하게 하고 반면 탐구의 경우에는 가능한 행동들 중에서 가장 q 함수 값이 큰 행동을 선택하여 노드를 연결하였다. 이런 과정을 거쳐 모든 목적지 노드가 연결이 되면 보상 값을 받게 되고 종료 상태가 된다. DQN 에이전트의 신경망을 업데이트 하는 방법은 학습 시에 상태, 행동, 보상, 다음 상태의 정보를 저장하게 되는데 저장된 정보에서 일정 주기 마다 무작위로 정보를 추출하여 받은 보상 값을 최대화하도록 신경망의 가중치를 갱신한다. 따라서 학습이 진행될수록 DQN 에이전트는 최적의 멀티캐스트 트리를 생성하게 된다.

이러한 과정을 거쳐 DQN 에이전트의 신경망이 학습되면 그림 1과 같이 멀티캐스트 트리를 생성하도록 하는 것이 가능하다. 그림 1은 학습된 DQN 에이전트가 멀티캐스트 트리를 생성하는 것을 나타낸 것이다. 먼저 출발 노드에서 멀티캐스트 트래픽이 발생하면 SDN 컨트롤러는 현재 네트워크의 노드 연결 정보를 관측하여 DQN 에이전트에게 전달한다. DQN 에이전트는 노드 연결 정보를 신경망의 입력으로 사용하여 현재 연결 가능한 노드들, 즉 행동에 대한 q 함수 값을 결과 값으로 내보낸다. 연결 가능한 노드들 중에서 q 함수 값이 가장 큰 노드를 선택하게 되고 목적지 노드를 모두 연결할 때 까지 이 과정을 반복한다. 그 후

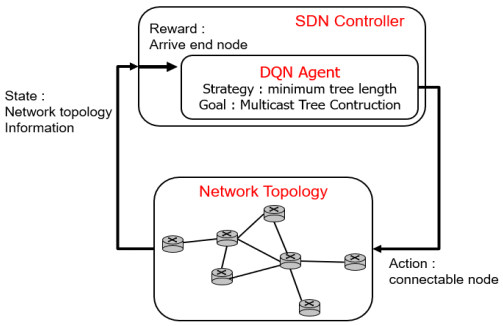

그림 1. 강화학습 에이전트의 멀티캐스트 트리 생성 과정
Fig. 1. Multicast tree construction process of RL agent.

SDN 컨트롤러는 생성된 트리에 따라 플로우 테이블을 작성하고 각 네트워크 노드에게 전달하여 플로우 테이블을 수정하도록 한다. 마지막으로 멀티캐스트 패킷은 수정된 플로우 테이블에 따라 목적 노드들까지 전달된다.

Ⅳ. 실 험

4.1 데이터 생성

본 실험은 SDN 환경에서 컨트롤러가 DQN 기법을 통해 멀티캐스트 트리를 생성하는 것이다. 따라서 출발지가 하나의 노드이고 도착지는 출발지를 제외한 노드에서 하나의 노드부터 출발지를 제외한 모든 노드가 될 수 있다. 표 2에서는 토폴로지 노드 개수에 따른 생성 가능한 출발 노드, 목적 노드 쌍의 개수를 보여준다. 표 2에서 토폴로지 노드의 개수가 증가할수록 생성 가능한 데이터의 개수가 기하급수적으로 증가한다. 따라서 표 3과 같이 중복을 제외하고 무작위로 일부를 추출하여 학습과 테스트 데이터로 사용하였다. 표 3에서 토폴로지 노드의 개수가 10개 일 때는 사용가능한 모든 데이터를 사용하였다. 반면 20개, 30개 인 경우에는 학습 데이터, 테스트 데이터를 각각 80,000개, 20,000개를 추출하여 사용하였다.

표 2. 노드 개수에 따른 데이터 개수
Table 2. The number of dataset, according to the number of node.

The number of nodes	dataset
10	5,020
20	10,485,360
30	16,106,126,460

표 3. 노드 개수에 따른 학습 데이터, 테스트 데이터 개수
Table 3. The number of train set, test set, according to the number of nodes

The number of nodes	Train	Test
10	4,010	1,010
20	80,000	20,000
30	80,000	20,000

4.2 네트워크 토폴로지 생성

$$p(u,v) = \alpha e^{-d/(\beta L)} \quad (2)$$

본 실험은 BRITE Tool[10]을 사용하여 그림 2와 같은 네트워크 토폴로지를 생성하였다. 네트워크 토폴로지 생성에 필요한 실험 변수들은 표 4과 같다. 토폴로지 생성 모델은 Waxman[9] 모델을 사용하여 노드의 위치가 랜덤으로 자리하도록 하였고 수식 2와 같은 확률로 노드가 연결되도록 하였다. α와 β는 최소 0에서 최대 1로 제한되는 범위를 가진다. 또한 d는 노드 u와 노드 v 사이의 유클리디안 거리를 나타낸다. L은 두 노드 사이의 최대 거리를 의미한다. 실험에 사용한 네트워크 토폴로지들은 Waxman 모델의 확률 변수 α와 β를 0.3, 0.2로 설정하였다. 노드의 개수는 10, 20, 30개를 가지도록 하였고 노드 개수에 따른 토폴로지 별로 링크의 밀집도를 30, 50, 70 퍼센트를 가지도록 생성하였다. 밀집도는 각 토폴로지에서 최대로 가질

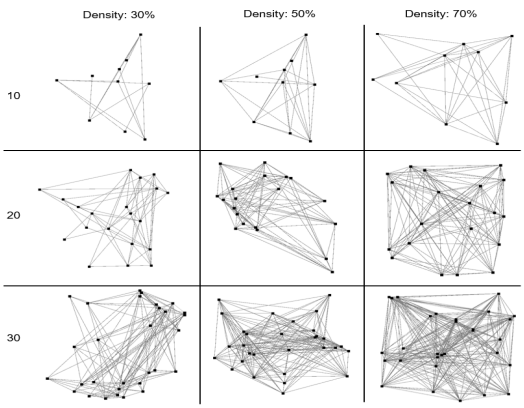

그림 2. 네트워크 토폴로지
Fig. 2. Network topology.

표 4. 토폴로지 생성에 사용한 실험 변수
Table 4. The number of dataset, according to the number of node.

Experiment parameter	setting
Random graph model	Waxman model[9]
Waxman parameter(alpha, beta)	(0.3, 0.2)
The Number of nodes	10, 20, 30
link density(%)	30, 50, 70

수 있는 링크 개수에 대한 연결된 링크의 비율이다. 또한 멀티캐스트 트리를 생성하기 위해서는 모든 노드가 연결 가능해야 하므로 노드 고립이 존재하지 않아야 한다. 따라서 그림 2과 같이 노드 고립이 존재하지 않으면서 10, 20, 30개의 노드 개수에 대해 3개의 다른 밀집도를 가지는 총 9개의 네트워크 토폴로지를 생성하여 실험에 사용하였다.

4.3 DQN 에이전트 학습

표 5는 실험에서 DQN 에이전트의 학습을 위해서 사용한 하이퍼 파라미터를 나타낸 것이다. 에피소드(episode)는 에이전트가 한번 시작 상태에서 종료 상태에 도달 할 때까지를 말하며 에이전트가 출발 노드에서부터 목적 노드들 까지 멀티캐스트 트리를 생성한 경우를 나타낸다. 토폴로지 노드 개수에 따라 이전에 생성된 데이터 집합을 통해 각각 200,000번의 에피소드를 학습 시켰다. 행동을 정하기 위해서 사용한 decaying e-greedy 방법의 epsilon 값은 현재 에피소드 값을 사용하여 학습이 진행될수록 표 5의 수식 3과 같이 epsilon 값이 감소하도록 하였다. 그러므로 0과 1 사이의 랜덤 값이 epsilon 값보다 작으면 무작위로 행동을 선택하도록 하고 랜덤 값이 크면 q 함수의 값이 가장 큰 행동을 선택하도록 하였다. 그 결과 학습이 진행될수록 랜덤으로 선택하는 행동이 감소하게 된다. 학습 시에 오차 함수의 가중치 갱신의 크기를 조절하기 위한 learning rate는 0.0001로 설정하였고 에이전트의 신경망 크기는 10개의 계층을 가진다. 미래 보상에 대해서 패널티를 주어 최소 비용 트리를 생성하기 위해서 사용한 감가율, γ는 0.9를 사용하였다. 모든 목적지를 포함하는 멀티캐스트 트리를 생성한 경우에 주어진 보상 값은 1의 값을 가진다. DQN의 데이터 연관성 문제를 위해 사용한 Reply Buffer의 크기는 10,000으로 설정하였다. 또한 Reply Buffer에 저장된 현재 상태, 행동, 보상 값, 이전 상태, 종료 상태의 정보를 10 에피소드 마다 추출하여 DQN 에이전트의 학습 신경망의 가중치를 학습시켰다. 동시에 Non-Stationary 문제를 해결하기 위해서 사용한 목표 신경망은 10 에피소드 마다 학습 신경망의 가중치로 복사하였다.

표 5. DQN의 하이퍼 파라미터
Table 5. The Hyper parameter of DQN

Hyper parameter	setting
episode	200,000
epsilon	2/((episode/10) + 1) (3)
learning rate	0.0001
the number of layer	10
r	0.9
Reply Buffer	10,000
reward	1
Update episode period	10 episode

V. 실험 결과

본 논문에서는 링크의 밀집도를 다르게 한 네트워크 토폴로지에서 멀티캐스트 트리 생성 비교 실험을 진행하였다. 멀티캐스트 트리 생성의 결과를 비교하기 위해서 테스트 데이터에 대해서 최적의 멀티캐스트 트리를 생성하고 DQN 에이전트가 생성한 멀티캐스트 트리와 비교 실험을 하였다. 최적의 멀티캐스트 트리와 DQN 에이전트가 생성한 멀티캐스트 트리를 비교하기 위해서 수식 1을 사용하였다[7]. 수식 1의 분모는 각 테스트 케이스에 대한 최적의 멀티캐스트 트리 길이이고 분자는 DQN 에이전트가 생성한 멀티캐스트 트리의 길이이다. 따라서 DQN 에이전트가 최적의 멀티캐스트 트리를 생성할수록 성능 비율(performance ratio)의 값은 1에 가까운 값이 된다.

그림 3은 링크의 밀집도에 따른 토폴로지의 전체 테스트 케이스에 대한 성능 비율의 평균을 나타낸 그림이다. 그림 3에서 링크의 밀집도가 작을수록 평균 성능 비율의 값이 1의 가까운 것을 확인 할 수 있다. 하지만 그림 3에서 노드의 개수가 많아질수록 최적 멀티캐스트 트리를 생성하기 어려워 토폴로지 노드의 개수가 30개인 경우에 대해서는 성능 비율을 구할 수

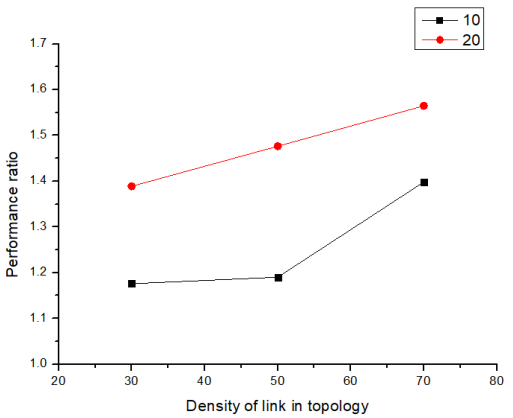

그림 3. 링크 밀집도에 따른 토폴로지의 평균 성능 비율
Fig. 3. The performance ratio of constructed multicast tree according to the density of link.

없었다. 따라서 노드 개수가 30개인 경우에도 DQN 에이전트가 생성한 멀티캐스트 트리가 최적인 멀티캐스트 트리를 따르는 트리인지 확인하기 위해 노드 개수에 따른 DQN 에이전트가 생성한 전체 멀티캐스트 트리 길이의 평균을 비교하는 실험을 진행하였다.

그림 4는 노드 개수에 따른 DQN 에이전트가 생성한 멀티캐스트 트리의 전체 평균 길이를 나타낸 것이다. 그림 4에서 노드 30개인 경우에도 노드 10개, 20개인 경우와 같은 추세를 가지는 것을 확인 할 수 있었다. 그러므로 그림 3과 4을 통해 노드 개수가 30개인 경우에도 DQN 에이전트가 생성한 멀티캐스트 트리의 길이가 최적의 멀티캐스트 트리에 따르는 길이를 가지는 것을 확인할 수 있었다. 그림 4에서 노드의 개수가 같을 때 링크의 밀집도가 높아질수록 DQN 에이전트가 생성한 멀티캐스트 트리의 평균 길이가 증가한다. 링크의 밀집도가 높으면 DQN 에이전트가 다음 노드를 선택할 때 선택 가능한 링크의 개수가 많아지기 때문에 최적의 다음 노드를 선택할 확률이 낮아지게 된다.

그림 5, 6은 노드의 개수가 각각 10개, 20개, 30개인 경우 30%, 50%, 70%의 밀집도를 가지는 토폴로지에서 도착지 노드의 개수에 따라 생성된 멀티캐스트 트리의 전체 평균 길이와 최적 멀티캐스트 트리의 전체 평균 길이를 나타낸 것이다. 그림 5, 6에서 도착지 노드의 개수가 증가할수록 최적의 멀티캐스트 트리의 평균 길이와 DQN 에이전트가 생성한 트리의 평균 길이와의 차가 감소하게 된다. 도착지 노드의 개수가 많을 때는 다음 노드를 선택할 때 선택한 노드가 도착지 노드일 확률이 높지만 도착지 노드의 개수가

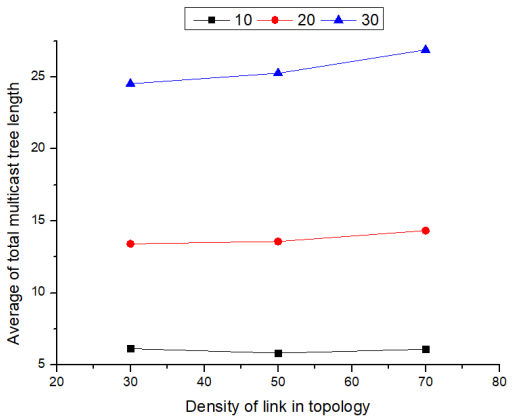

그림 4. 링크 밀집도에 따른 생성된 멀티캐스트 트리의 전체 평균 길이
Fig. 4. The average of total length of constructed multicast tree according to the density of link.

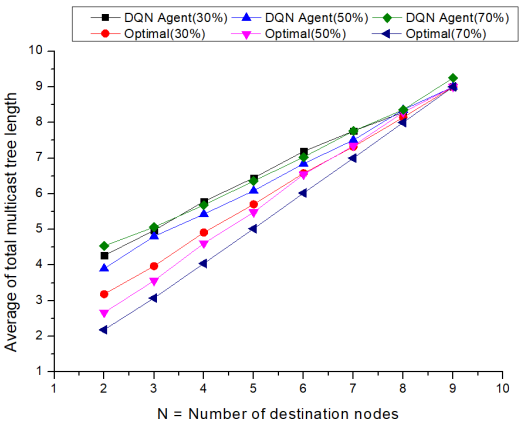

그림. 5. 노드 개수가 10개일 때 생성된 멀티캐스트 트리와 최적 멀티캐스트 트리의 전체 평균 길이
Fig. 5. The average of total length of constructed multicast tree and optimal multicast tree when the number of node is 10.

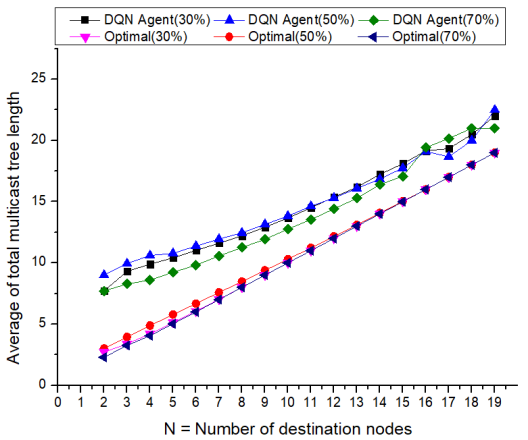

그림. 6. 노드 개수가 20개일 때 생성된 멀티캐스트 트리와 최적 멀티캐스트 트리의 전체 평균 길이
Fig. 6. The average of total length of constructed multicast tree and optimal multicast tree when the number of node is 20.

적을 때는 선택한 노드가 도착지 노드일 확률이 낮기 때문에 도착지 노드의 수가 작은 것은 각 노드들을 연결하는 링크와 노드에 대한 충분한 학습이 DQN 에이전트가 이루어져야 한다는 의미이다. 따라서 도착지 노드의 개수가 작을 때 충분한 학습이 이루어지지 않는다면 최적을 따르는 멀티캐스트 트리를 생성하는 것은 어렵다.

그림 7은 그림 5, 6에서와 같이 노드의 개수가 각각 10개, 20개, 30개인 경우 30%, 50%, 70%의 밀집도를 가지는 토폴로지에서 도착지 노드의 개수에 따른 생성된 멀티캐스트 트리의 전체 평균 길이를 나타

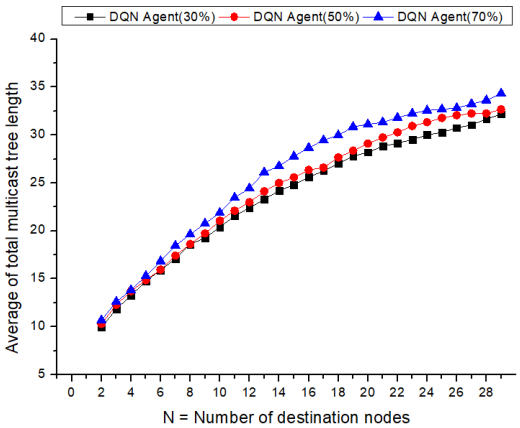

그림 7. 노드 개수가 30개일 때 생성된 멀티캐스트 트리의 전체 평균 길이
Fig. 7. The average of total length of constructed multicast tree when the number of node is 30.

낸 것이다. 그림 7에서 생성된 멀티캐스트 트리 전체 평균 길이는 토폴로지 링크의 밀집도가 높을수록 증가한다. 그림 5, 6, 7에서 모두 도착지 노드의 개수가 증가할수록 DQN 에이전트가 생성한 멀티캐스트 트리의 전체 평균 길이가 증가한다. 도착지 노드의 개수가 늘어나면 연결해야 하는 링크의 개수 또한 마찬가지로 늘어나야하기 때문에 도착지 노드의 개수가 증가할수록 멀티캐스트 트리의 길이가 증가하게 된다.

Ⅵ. 결론 및 향후 연구

본 논문에서는 강화학습 기법 중 DQN 기법을 적용하여 SDN 환경에서 네트워크 토폴로지의 밀집도에 따른 멀티캐스트 트리 생성 결과를 분석하였다. 실험에는 노드의 개수와 링크의 밀집도를 다르게 하여 총 9개의 토폴로지를 사용하였다. 각 토폴로지에서 DQN 에이전트에게 학습 데이터를 통해 멀티캐스트 트리를 생성하도록 하였고 그 결과를 확인하기 위해서 테스트 데이터를 통해 성능을 분석하였다.

실험 결과 링크의 밀집도가 상대적으로 높은 토폴로지는 밀집도가 낮은 토폴로지보다 생성된 멀티캐스트 트리의 길이가 증가함을 알 수 있었다. 이는 DQN 에이전트가 충분한 학습이 이루어지지 않은 경우 밀집도가 높으면 다음 노드를 선택할 때 최적의 멀티캐스트를 따르는 노드를 선택할 확률이 낮아지기 때문이다. 따라서 밀집도가 높을수록 최적의 멀티캐스트 트리를 생성하는 것은 어려운 것으로 판단된다. 또한 도착지 노드의 개수가 증가하면 최적의 멀티캐스트 트리와 생성된 멀티캐스트 트리의 길이의 차이가 감소하였다. 이는 도착지 노드의 개수가 많을수록 다음 노드를 선택할 때 선택한 노드가 목적지 노드일 확률이 낮기 때문이다. 그러므로 도착지 노드의 개수가 작을수록 최적의 노드를 선택할 만큼의 충분한 학습이 필요하다고 생각된다.

논문에서 가정한 환경은 한 네트워크 토폴로지 내에서 멀티캐스트 트래픽이 여러 출발지에서 발생하는 환경이다. 하지만 많은 경우 멀티캐스트 트래픽은 하나의 출발지에서 발생한다. 따라서 향후 연구로는 실제 멀티캐스트 트래픽이 발생하는 네트워크 환경을 가정하여 실험을 진행할 예정이다. 또한 더 많은 수의 네트워크 노드를 가지는 토폴로지를 사용하여 보다 크고 다양한 네트워크 환경에서 사용 기법의 유효성을 판단할 수 있을 것이다.

References

[1] FORECAST, Cisco VNI, *Cisco Visual Networking Index: Forecast and Trends, 2017 -2022*, White paper, Cisco Public Information, 2019.

[2] D. Kim, et al., "An efficient flow table management scheme in SDN," *JKIIT*, vol. 17, no. 9, pp. 65-74, Sep. 2019.

[3] J. R. Jiang and S. Y. Chen, "Constructing multiple Steiner trees for software-defined networking multicast," in *Proc. 11th Int. Conf. Future Internet Technologies(CFI)*, pp. 1-6, Nanjing, China, 2016.

[4] M. Sun, et al., "A multiple multicast tree optimization solution based on software defined network," *ICICS*, pp. 168-173, Irbid, Jordan, Apr. 2016.

[5] P. Winter, "Steiner problem in networks: A survey," *Networks*, vol. 17, no. 2, pp. 129-167, 1987.

[6] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Comput. Computations*, pp. 85-103, Springer, Boston, MA, 1972.

[7] S. Hougardy and H. J. Priomel, "A 1.598 approximation algorithm for the steiner problem in graphs," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, pp. 448-453, 1999.

[8] V. Mnih, et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529-533, Feb. 2015.
[9] B. M. Waxman, “Routing of multipoint connections,” *IEEE J. Sel. Areas in Commun.*, vol. 6, no. 9, pp. 1617-1622, Aug. 1988.
[10] A. Medina, et al., “BRITE: An approach to universal topology generation,” in *Proc. MASCOTS 2001*, pp. 346-353, OH, USA, Aug. 2001.

채 지 훈 (Jihun Chae)

2019년 2월 : 경기대학교 컴퓨터과학과 졸업
2019년 3월~현재 : 경기대학교 컴퓨터과학과 석사과정
<관심분야> 네트워크, 통신시스템, 강화학습

김 남 기 (Namgi Kim)

1997년 : 서강대학교 컴퓨터학과
2000년 : KAIST 전산학과 공학석사
2005년 : KAIST 전산학과 공학박사
2005년~2007년 : 삼성전자 통신연구소 책임연구원
2007년 3월~현재 : 경기대학교 컴퓨터공학부 교수
<관심분야> 통신시스템, 네트워크

이 병 대 (Byoung-Dai Lee)

1996년 2월 : 연세대학교 전산학과
1998년 2월 : 연세대학교 전산학과(공학석사)
2003년 2월 : Univ. of Minnesota CSE(공학박사)
2003년 3월~2010년 2월 : 삼성전자 책임연구원
2010년 3월~현재 : 경기대학교 컴퓨터과학과 교수
<관심분야> 기계학습, 딥러닝, 의료 영상 분석