https://doi.org/10.7840/kics.2022.47.9.1353

전술망의 라우팅 성능 개선을 위한 성능 지표 분석 기반 정책 엔진 설계

윤 대 건•, 노 병 희*, 오 상 윤°

Performance-Index Based Rule Engine for Improving Routing Performance of a Tactical Network

Daegun Yoon•, Byeong-hee Roh*, Sangyoon Oh°

요 약

컴퓨팅 관련 기술 발달에 따라 군 작전 수행에서 발생하는 데이터의 규모가 매우 커지고 있으며, 이에 따라 이를 처리하기 위한 군 전술망의 성능 향상에 대한 요구 또한 점점 늘어나고 있다. 군 전술망의 특성 상 다양한 장비로 구성된 네트워크를 활용해야 하며, 이러한 상황에서 민간에서 활발히 적용되는 Software-Defined Network (SDN) 기술을 적용한다면 장비를 제공 벤더로부터 자유로운 손쉬운 네트워크 관리가 가능하다. 본 논문에서는 SDN 기반 네트워크 환경에서 패킷 전송 성능 향상을 목적으로 하는 네트워크 정책 엔진 구조 설계를 소개한다. 정책 엔진은 Flow table의 Flow들이 나타내는 라우팅 경로를 수정하도록 하는 알고리즘을 포함하며 성능 개선 여부는 본 연구에서 정의한 종합 성능 지표를 통해 판단한다. 추후 본 연구에서 제안하는 전술망 라우팅 성능 개량을 위한 성능 지표 분석 기반 정책 엔진 기반의 소프트웨어를 실제 네트워크 운용 상황에 적용하고, 네트워크 성능 향상을 검증하도록 할 계획이다.

Key Words : Software-defined network, rule engine, routing performance

ABSTRACT

The scale of data generated from the military operations as well as the demand of processing the large-scale data are increasing. A military network is comprised of diverse devices from different vendors. Thus, a software-defined network (SDN) is proposed to make the network management simple regardless of device vendors. In this study, we design a rule engine that enhances the performance of packet transmission in SDN environment. The rule engine includes an algorithm that modifies a routing path of each flow in flow table, and determines whether the performance is improved or not by looking at the overall performance index. For future works, we adopt the rule engine in real network environment to improve the network performance.

※ 본 연구는 방위사업청과 국방과학연구소가 지원하는 미래전투체계 네트워크기술 특화연구센터 사업의 일환으로 수행되었습니다. (UD190033ED)
• First Author : Ajou University Department of Artificial Intelligence, kljp@ajou.ac.kr, 학생회원
° Corresponding Author : Ajou University Department of Artificial Intelligence, syoh@ajou.ac.kr, 정회원
* Ajou University Department of Artificial Intelligence, bhroh@ajou.ac.kr, 종신회원
논문번호 : 202205-082-0-SE, Received April 30, 2022; Revised June 8, 2022; Accepted June 12, 2022

Ⅰ. 서 론

군사 및 컴퓨팅 기술의 발달로 전장에서의 작전 수행 과정에서 발생하는 데이터의 규모가 점점 커지고 있다. 실시간으로 대규모 데이터를 처리하기 위해 군 전술망 (Tactical network)은 하드웨어뿐만 아니라 네트워크에서의 데이터 전송 관련한 소프트웨어도 개선하여 탑재해야 한다. 네트워크를 구성하는 하드웨어를 교체하는 소요 비용은 매우 크므로, 소프트웨어 수준에서 성능을 효과적으로 최적화하는 것은 저비용으로 전체 성능을 극대화할 수 있기 때문이다.

하지만 네트워크는 운영체제, 사용자 인터페이스, 프로토콜 등이 구성요소들이 벤더마다 상이할 수 있기 때문에, 새로운 네트워크 프로토콜을 적용하거나 네트워크 구조를 바꾸는 것이 어렵다[1]. 이러한 제한점을 극복하기 위해 소프트웨어 정의 네트워크 (Software-defined network, 이하 SDN) 패러다임이 제안되었다. SDN은 네트워크의 제어부를 전송부로부터 분리하고 네트워크를 가상화함으로써 더욱 효율적으로 리소스를 관리하고 벤더에 독립적으로 네트워크 프로그래밍이 가능하도록 한다[2].

SDN은 패킷을 처리하기 위해 flow라는 단위를 사용한다. Flow는 패킷을 전송하기 위해 정해야 하는 규칙들로 구성되며 송신 주소, 착신 주소, 포트, 액션 등의 필드들이 이러한 규칙에 해당한다[3]. SDN의 각 스위치는 이러한 flow들의 집합으로 이루어진 Flow table을 가지고 있으며 네트워크 패킷의 라우팅은 스위치들의 flow들에 따라 결정된다. 따라서 SDN 스위치들의 flow들을 어떻게 구성하는지에 따라 네트워크 패킷의 라우팅 경로가 달라진다. 예를 들어, 착신 주소로 패킷을 보내기 위해 거치는 스위치들이 각각 어느 포트로 패킷을 내보내는지에 따라 라우팅 경로가 달라진다. 즉, flow들로부터 정의되는 모든 경로들을 종합하였을 때 스위치 간 링크 사용률이 도출되고 링크들 간의 부하 차이가 발생함을 확인할 수 있다. 패킷들의 라우팅 거리가 길어지면 네트워크에서의 데이터 전송 시간이 증가하는 것이기 때문에 라우팅 거리를 줄이는 것이 네트워크 성능 향상에 매우 중요하다. 또한, 라우팅 경로는 링크들의 연속으로 이루어지는데, 특정 링크들이 다른 링크들에 비해 라우팅 경로들에 많이 포함된다면 해당 링크들에 부하가 집중되어 부하 분산이 불균형해진다. 그 결과로 네트워크 리소스 활성에 불균형이 생겨 전체 네트워크 성능이 저하될 가능성이 크다.

SDN의 성능을 측정하기 위해 Wang 외 3인[6]의 연구에서는 패킷의 개수, 바이트 수, 쿼리 시간 등을 통계 내어 네트워크 트래픽의 특성을 나타냈다. 하지만 해당 연구는 네트워크 트래픽의 특성을 time series로 나타내어 이후의 트래픽을 예측하는 것이 목표이기 때문에 SDN의 flow table을 직접적으로 수정하여 성능을 개선하는 것과는 거리가 있다.

본 논문에서는 네트워크의 성능을 향상을 목표로 라우팅 거리와 링크 사용률 차이를 줄이기 위한 성능 지표 기반 정책 엔진 (Rule engine) 설계를 소개한다. 본 연구에서의 성능 지표 요소는 라우팅 거리 평균, 라우팅 거리 최댓값, 링크 사용률 분산, 링크 사용률 최댓값으로 구성되며 이 요소들을 이용하여 성능을 나타내는 종합 지표를 도출한다. 제안하는 정책 엔진은 종합 성능 지표의 수치를 증가시킴으로써 전체 네트워크의 성능을 향상시키는 것을 목표로 하며, 이 수행 과정은 주기적으로 SDN 스위치에 속한 flow들의 필드를 수정하여 패킷 라우팅 경로들을 수정하는 것으로 이루어진다.

본 연구의 정책 엔진은 flow 수정 내역을 유향 비순환 그래프 (Directed acyclic graph, 이하 DAG) 형태의 자료 구조를 이용하여 관리한다. 즉, flow의 내용이 수정되어 DAG을 갱신 (Update)할 때는 수정 내역을 새로운 정점 (Vertex)으로 생성하여 그래프에서 연결되도록 한다. 반면, 수정 내역을 폐기 (Revert)하여 이전의 flow로 되돌리기 위해서는 추가된 정점에 연결된 간선 (Edge)을 제거한다. 정책 엔진은 DAG을 이용한 자료 구조와 갱신, 폐기 연산을 통해 flow 수정 내역을 히스토리로 보관하므로 flow 수정으로 인한 성능 변화에 따라 언제든지 원하는 시점의 상태 (State of flow tables), 즉 체크포인트 (Checkpoint)로 되돌아갈 수 있다.

본 연구의 타당성을 입증하기 위해 성능 분석 예제를 기반으로 case study를 진행하였으며, 예제에서는 시간 흐름에 따른 각 성능 지표 요소들의 수치 변화를 측정하여 라우팅 경로의 효율성과 링크 부하 분산의 개선을 확인함으로써 정책 엔진 적용의 효과를 평가하였다. 본 연구에서 설계한 정책 엔진은 SDN 환경의 Application plane (AP)[3]에 배치하여 SDN 컨트롤러와 Northbound API[2]를 통해 통신하도록 함으로써 실제 네트워크 운용 상황에서 사용할 수 있다. 구체적으로, 정책 엔진은 SDN 컨트롤러[4,5]로부터 flow table을 전달받고, 컨트롤러로 flow의 변경 사항을 전달하는 역할을 수행하도록 실제 네트워크에 적용 가능하다.

Ⅱ. 성능 지표 분석 기반 정책 엔진

2.1 종합 성능 지표 정의

본 연구에서는 라우팅 효율성을 기반으로 네트워크의 성능을 평가하기 위해 라우팅 거리 평균, 라우팅 거리 최댓값, 링크 사용률 분산, 링크 사용률 최댓값으로 이루어진 총 네 개의 성능 지표 요소를 활용한다.

성능 지표 요소를 정의하기 위해 필요한 라우팅 거리는 다음과 같이 링크 가중치들의 합으로 정의한다.

$$d_k = \sum_{i=0}^{m_k - 1} w_i \tag{1}$$

수식 (1)에서 d_k는 k 번째 경로의 라우팅 거리이고, m_k는 해당 경로에서 거치는 링크의 수 (라우팅 홉 수)이며, w_i는 각 링크의 가중치이다. 특히 모든 가중치가 1일 때는 라우팅 거리가 라우팅 홉 수와 같아진다. 수식 (1)을 기반으로 라우팅 거리 관련 성능 지표 요소들의 정의가 도출되며, 첫 번째 요소인 라우팅 거리 평균 d_{avg}의 정의는 다음과 같다:

$$d_{avg} = \frac{\sum_{k=0}^{n-1} d_k}{n} \tag{2}$$

수식 (2)에서 n은 경로의 총 개수이다. 만약 라우팅 거리 평균이 크다면 전체적으로 네트워크의 Flow들에 정의된 라우팅 경로들이 길다는 것을 의미하므로 패킷 라우팅이 비효율적일 가능성이 크다. 따라서 라우팅 거리 평균의 값을 줄이는 것이 중요하다. 두 번째 성능 지표 요소인 라우팅 거리 최댓값 $d_{\max}$의 정의는 다음과 같다:

$$d_{\max} = \max_{0 \le k < n} d_k \tag{3}$$

라우팅 거리 최댓값이 라우팅 거리 평균보다 커질수록 특정 라우팅 경로가 다른 라우팅 경로들보다 길어진다는 것이므로 라우팅 거리가 최대인 경로의 거리를 줄이는 방안이 필요하다.

성능 지표 요소들은 라우팅 거리뿐만 아니라 링크 사용률로부터도 정의된다. 먼저 경로 k가 링크 i를 지나는 지를 나타내는 지표인 p_i^k의 정의는 다음과 같다:

$$p_i^k = \begin{cases} 1 \, passed \\ 0 \, else \end{cases} \tag{4}$$

즉, 해당 링크를 지날 경우 1, 지나지 않을 경우 0이 된다. p_i^k를 이용하여 링크 사용률은 다음과 같이 정의된다:

$$u_i = \frac{\sum_{k=0}^{n-1} p_i^k}{n} \tag{5}$$

수식 (5)에서 u_i는 i 번째 링크 사용률이고 전체 경로들 중 해당 링크를 지나는 경로의 비율을 의미한다. 수식 (5)를 기반으로 링크 사용률 관련 성능 지표 요소들의 정의가 도출되며, 세 번째 성능 지표 요소인 링크 사용률 분산 u_{var}의 정의는 다음과 같다:

$$u_{var} = \frac{\sum_{i=0}^{m-1} (u_i - u_{avg})^2}{m} \tag{6}$$

수식 (6)에서 m은 링크의 총 개수이고 u_{avg}는 링크 사용률 평균이다. 링크 사용률이 커질수록 링크 간 부하 분산이 고르지 않다는 것을 의미하기 때문에 링크 사용률 수치를 낮추는 것이 중요하다. 네 번째 성능 지표 요소인 링크 사용률 최댓값 $u_{\max}$은 다음과 같이 정의된다:

$$u_{\max} = \max_{0 \le i < m} u_i \tag{7}$$

패킷들의 라우팅 경로가 특정 링크로 집중될 경우 링크 사용률 최댓값이 증가한다. 따라서 링크 사용률 분산과 마찬가지로 부하 균형을 고르게 하기 위해 링크 사용률 최댓값 또한 낮추는 것이 네트워크 성능을 향상시키는 데에 유리하다.

본 연구에서는 상기 정의된 성능 지표 요소들을 이용하여 종합 성능 지표를 정의한다. 종합 성능 지표 S는 라우팅 거리 평균 d_{avg}, 라우팅 거리 최댓값 $d_{\max}$, 링크 사용률 분산 u_{var}, 링크 사용률 최댓값

u_{max}를 고려하여 설계되었으며 각 요소의 값에 반비례하는 지표로 정의된다:

$$S = \frac{\theta_0}{d_{avg}} + \frac{\theta_1}{d_{max}} + \frac{\theta_2}{u_{var}} + \frac{\theta_3}{u_{max}} \quad (8)$$

본 연구에서 제안하는 정책 엔진의 성능 평가는 종합 성능 지표의 수치를 기준으로 이루어지며 이 지표의 수치는 네 개의 성능 지표 요소 값들이 작아질수록 커진다. 따라서 flow table들의 flow들을 수정하여 라우팅 거리를 줄이고 링크 사용률 간 차이를 줄이는 것이 전체 네트워크 성능 최적화에 매우 중요하다. 수식 (8)의 θ_0,θ_1,θ_2,θ_3은 하이퍼파라미터로서 각각 100, 100, 1, 1로 고정된 값을 가진다. 각 하이퍼파라미터의 값은 상황에 따라 변경될 수 있으며, 라우팅 거리 관련 성능 지표 요소 값들이 링크 사용률 관련 성능 지표 요소 값들보다 크기가 매우 작기 때문에 가중치를 주기 위해 θ_0과 θ_1의 크기를 크게 설정하였다. 하이퍼파라미터들과 종합 성능 지표 수식을 최적화하기 위한 연구들이 추후에 추가적으로 진행될 필요가 있다.

2.2 Flow table 성능 개선 알고리즘

본 절에서는 네트워크의 성능을 개선하기 위해 정책 엔진에서 성능 지표 요소들을 기반으로 도출한 종합 성능 지표를 이용하여 flow의 라우팅 경로를 변경하는 알고리즘에 대해 자세히 소개한다. 알고리즘 1은 flow의 라우팅 경로를 변경하여 성능을 개선하기 위한 워크플로우를 나타낸다. 알고리즘 1의 워크플로우는 네트워크 운용이 중단되기 전까지 실행이 종료되지 않으며, line 3부터 line 11까지의 nested loop에 표현되어 있는 바와 같이 flow table 내의 flow를 하나씩 선택하여 해당 flow의 라우팅 경로를 수정하도록 하는 iterative 알고리즘이다. Nested loop에서는 flow table의 flow들이 한 번씩 조회되어 라우팅 경로를 수정할지 말지 결정되며, 모든 flow가 조회되어 nested loop을 빠져나가면 다시 모든 flow를 대상으로 nested loop를 통해 flow 개선 과정을 반복한다. Line 7은 Nested loop 내의 각 iteration에서 flow 하나를 선택하는 것을 나타낸다. Line 9는 선택된 flow를 백업함으로써 flow 업데이트를 이후에 취소할 수 있도록 하는 과정이며, line 10과 line 11은 각각 선택된 flow의 라우팅 경로를 수정하고 flow table에 업데이트하는 과정을 나타낸다. Line 12는 업데이트된 flow table을 이용하여 네트워크 상태를 정해진 시간동안 모니터링

알고리즘 1 Flow-improving workflow

```
1  while true do
2    flows ← FT;
3    while !flows.size() do
4      if !(flows.size == FT.size()) then
5        if !analyzeStatus(FT) then
6          FT.revert(temp);
7      flow = flows[rand() % flows.size()];
8      flows.delete(flow);
9      temp = flow;
10     modifyRoutingPath(flow);
11     FT.update(flow);
12     monitor();
```

하는 과정이며, line 5부터 line 6은 모니터링 기간 동안 네트워크 성능이 향상되었는지를 분석하여 향상되었다면 해당 flow의 업데이트를 유지하고, 향상되지 않았다면 업데이트를 revert하는 과정이다. 이 때 revert 결정이 내려지면 해당 flow는 백업된 flow 정보를 이용하여 업데이트 이전으로 돌아간다. 본 연구에서는 종합 성능 지표의 증감 여부를 이용하여 성능 향상 여부를 판단한다. flow table 성능 개선 알고리즘은 이러한 과정들을 통해 네트워크 운용 기간 동안 끊임없이 성능을 개선하기 위해 작동한다.

그림 1은 알고리즘 1에서 flow 하나를 추출하여 라우팅 경로를 수정하는 과정을 도식화한 것이다. Flow의 라우팅 경로는 그림 1의 flow table의 column 중 하나인 action의 port를 변경함으로써 수정할 수 있는데, 이는 각 SDN 스위치에서 패킷이 나가게 될 포트가 변경됨에 따라 라우팅 경로가 변경될 수 있기 때문이다. 각 flow가 처음 선택되었을 때는 라우팅 경로는 시작점부터 도착점까지 최단 경로가 선택된다. 모든 flow가 한 번씩 선택되어 수정 내역이 갱신 또는 폐기

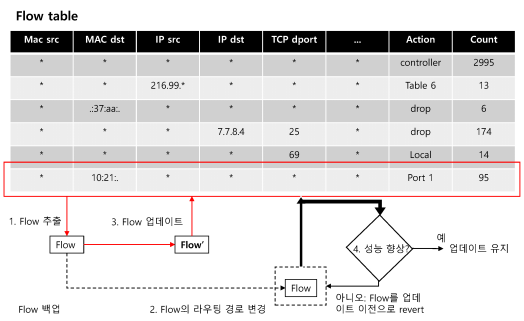


그림 1. Flow table 라우팅 경로 변경 워크플로우
Fig. 1. Flow table routing path modification workflow

된 후, 다시 한 번씩 선택될 때 (다음 라운드, 즉 알고리즘 1의 line 1에서의 loop에 해당)는 지난 라운드에서 선택했던 경로 다음으로 짧은 경로를 선택하여 flow 라우팅 경로 수정을 시도한다. 즉, 본 논문에서 제안하는 flow table 성능 개선 알고리즘은 종합 성능 지표 수치의 실제 상승이 이루어진 경우에만 flow의 업데이트를 진행하므로, 알고리즘이 적용되는 시간이 경과함에 따라 성능 향상이 이루어 질 것으로 예측된다.

2.3 Flow table 수정내역 및 체크포인트 관리

본 절에서는 정책 엔진이 flow 수정 내역을 관리하기 위한 방법에 대해 소개한다. 본 연구의 정책 엔진은 flow 수정 내역을 DAG 형태의 자료 구조로 나타낸다. Flow가 갱신되면 수정 내역은 새로운 정점으로 생성되어 그래프에 연결되고, flow를 수정 이전 버전으로 되돌릴 때는 해당 정점에 연결된 간선을 제거한다.

그림 2는 flow 수정 내역 관리의 예시를 나타낸다. 예시에서는 iteration 0부터 시작하고, 선택된 flow의 수정 내역을 새로운 정점으로 생성하여 그래프에 연결함으로써 flow 수정 내역을 관리한다. 이 때 수정 내역의 업데이트가 유지되려면 알고리즘 1의 line 5에 나타낸 성능 향상 여부에서 종합 성능 지표 수치가 상승한 것이 확인되어야 한다. 만약 종합 성능 지표 수치가 하락했다면 그림 2의 예시 중 iteration 2에서와 같이 수정 내역을 폐기하는 과정이 이루어진다. 이 때 iteration 2에서 flow 수정 내역을 나타내는 정점인 n_c는 삭제되지 않고 유지되는데, 이는 성능 향상 여부와 상관없이 원하는 시점의 그래프로 복원하기 위한 체크포인트의 기능을 지원하기 위한 것이다. 그 다음으로 iteration 3에서는 폐기된 수정 내역의 정점을 건너뛰고 다음 flow의 수정 내을 나타내는 정점이 그래프에 연결된다. 이와 같이 flow table의 히스토리를 그래프로 기록하여 버전 관리를 가능하게 함으로써 네트워크의 상황에 따라 적절한 라우팅 경로를 통해 패킷 전송을 용이하게 할 수 있다.

Ⅲ. 성능 분석 예제

본 장에서는 예제를 기반으로 정책 엔진의 flow 개선 알고리즘에 의해 성능 지표 요소들과 그에 따라 도출되는 종합 성능 지표의 수치가 어떻게 변화하는지를 관찰한다.

그림 3은 성능 분석 예제에서 사용되는 네트워크 상태를 그래프로 나타낸 것이며, 간선의 가중치는 거리를 의미한다. Flow 수정에 따라 변하는 종합 성능 지표 수치를 분석하기 위해 총 네 개의 flow를 정의하였으며, 이 예제는 표 1에 나타낸 바와 같다. 이 중 f_0은 송신 노드인 정점 A로부터 착신 노드인 정점 G까지 패킷을 전송하기 위한 라우팅 경로를 나타내는 flow이며 총 거리는 17이다. Flow가 총 네 개이므로 한 라운드 내에 총 네 번의 iteration이 존재하며, 본 예제에서는 한 번의 라운드 내에서 일어나는 종합 성능 지표 수치의 변화만을 관찰한다.

각 iteration에서 flow는 임의로 선택되므로 본 예제에서는 iteration 0부터 iteration 3까지 f_0, f_2, f_3, f_1 순으로 선택되는 것으로 가정한다. Flow 개선 알

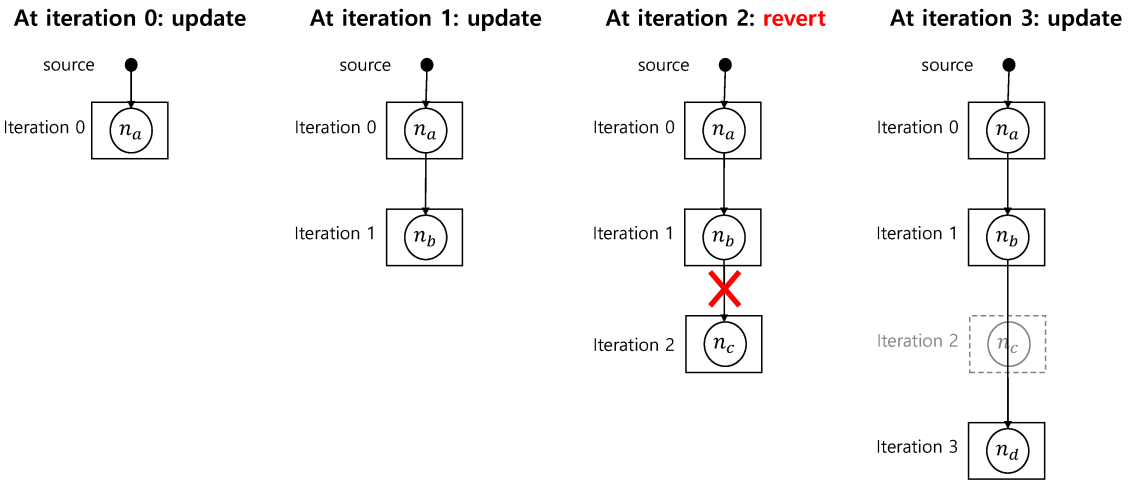


그림 2. Flow 수정내역 관리 예시
Fig. 2. Example of managing modification history

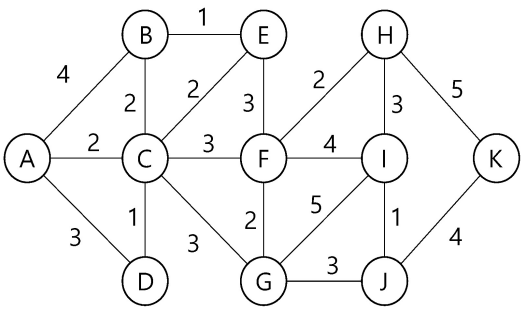


그림 3. 네트워크 예시
Fig. 3. Example of network

표 1. Flow 초기 상태
Table 1. Initial state of flow table

Flow	Path	Distance
f_0	ABEFHIJG	17
f_1	CGI	8
f_2	BACGJK	16
f_3	ECFIH	12

고리즘에 의해 iteration 0에서 f_0가 선택되어 라우팅 경로가 기존의 ABEFHIJG (거리 17)에서 최단거리를 나타내는 경로인 ACG (거리 5)로 변경된다. 라우팅 경로 수정에 의해 종합 성능 지표 수치는 표 2에 나타낸 바와 같이 47.62에서 61.43으로 상승하며 알고리즘은 성능이 향상되었다고 판단하여 flow 업데이트를 유지한다.

다음으로, iteration 1에서는 f_2을 선택하여 라우팅 경로를 BACGJK (거리 16)에서 BEFHK (거리 11)로 변경한다. 이에 따라 종합 성능 지표 수치가 61.43에서 84.24로 상승하여 flow 업데이트가 유지된다. 하지만 iteration 2에서 변경되는 f_3 업데이트는 반영되지 않고 폐기되는데, 이는 라우팅 경로가 ECFIH (거리 12)에서 EFH (거리 5)로 변경되지만 종합 성능 지표

표 2. Iteration 별 성능 지표 요소 및 종합 성능 지표 수치
Table 2. Detailed values of performance index elements and overall performance index for each iteration

Iter.	d_{avg}	$d_{\max}$	u_{var}	$u_{\max}$	S
초기 값	13.25	17	0.031	0.5	47.62
0	10.25	16	0.020	0.5	61.43
1	9.00	12	0.016	0.25	84.24
2	9.00	12	0.016	0.25	84.24
3	9.00	12	0.016	0.25	84.24

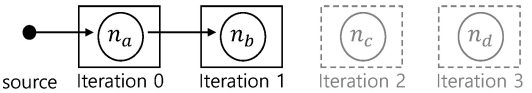


그림 4. 한 라운드 후의 flow 수정 내역을 나타내는 DAG
Fig. 4. DAG representing the modification history of flow table after a round

수치가 84.24에서 58.25로 하락하기 때문이다. 마찬가지로, iteration 3에서 f_1의 라우팅 경로가 CGI (거리 8)에서 CFI (거리 7)로 수정되지만 종합 성능 지표 수치가 84.24에서 54.73으로 하락하여 수정 내역이 폐기된다. 따라서 flow 개선 알고리즘에 의해 한 라운드 내에 종합 성능 지표 수치가 초기 값인 47.62에서 84.24로 상승하였으며 이를 통해 네트워크 성능 향상을 기대할 수 있다.

알고리즘 실행 결과로 한 라운드 진행 후의 flow 수정 내역을 나타내는 DAG는 그림 4와 같이 만들어진다. Iteration 2와 iteration 3에서의 flow 수정 내역은 간선 삭제로 업데이트에 반영되지 않았지만 체크포인트를 사용하여 언제든 다시 연결될 수 있다.

Ⅳ. 결론 및 향후 연구 계획

본 논문에서는 SDN 환경에서의 네트워크 성능 향상을 위해서, 네트워크 성능을 반영하는 지표를 정의하고 flow의 패킷 라우팅 경로를 수정하여 이 지표의 수치를 상승시키기 위한 알고리즘을 포함하는 네트워크 정책 엔진을 제안하였다.

본 논문에서는 SDN 환경에서의 네트워크 성능 향상을 위해서, 네트워크 성능을 반영하는 지표를 정의하고 flow의 패킷 라우팅 경로를 수정하여 이 지표의 수치를 상승시키기 위한 알고리즘을 포함하는 네트워크 정책 엔진을 제안하였다.

추후 본 연구에서 제안한 정책 엔진을 이용한 프로그램을 개발하도록 하여 실제 SDN 환경에서의 패킷 전송 성능 개선을 달성하도록 할 계획이다. 이에 대한 실험 계획으로 시간 흐름에 따른 각 성능 지표 요소들의 수치 변화를 측정하여 라우팅 경로의 효율성과 링크 부하 분산의 개선을 확인함으로써 정책 엔진의 효과를 평가할 계획이다. 즉, 본 연구에서 제안한 정책 엔진을 구현하여 실제 SDN 환경에 탑재함으로써 네트워크 상황 모니터링 및 flow table 수정을 자동화하지 않은 설정의 네트워크들에 비해 더 성능 개선을 용이하게 하고 다양한 상황에 유연하게 대처할 수 있을 것으로 기대한다.

References

[1] H. Kang, S. Cho, and S. Shin, "Safe communication under vulnerable network with software-defined networking," in *Proc. KICS Autumn Conf. 2019*, pp. 54-55, Seoul, Korea, Nov. 2019.

[2] P. V. Tijare and D. Vasudevan, "The northbound APIs of software defined networks," *Int. J. Eng. Sci. and Res. Technol.*, vol. 5, no. 10, pp. 501-513, Oct. 2016. (https://doi.org/10.5281/zenodo.160891)

[3] B. Isyaku, M. S. Mohd Zahid, M. Bte Kamat, K. Abu Bakar, and F. A. Ghaleb, "Software defined networking flow table management of openflow switches performance and security challenges: A survey," *Future Internet*, vol. 12, no. 9, pp. 147-176, Aug. 2020. (https://doi.org/10.3390/fi12090147)

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69-74, Apr. 2008. (https://doi.org/10.1145/1355734.1355746)

[5] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proc. IEEE Int. Symp. World of Wirel., Mob. and Multimedia Netw.*, pp. 1-6, Sydney, Australia, Jun. 2014. (https://doi.org/10.1109/WoWMoM.2014.6918985)

[6] Y. Wang, D. Jiang, L. Huo, and Y. Zhao, "A new traffic prediction algorithm to software defined networking," *Mob. Netw. and Appl.*, vol. 26, no. 2, pp. 716-725, Dec. 2019. (https://doi.org/10.1007/s11036-019-01423-3)

윤 대 건 (Daegun Yoon)

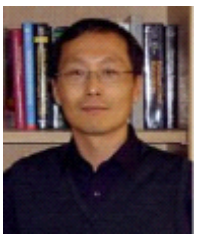
2018년 8월 : 아주대학교 소프트웨어학과 학사

2018년 9월~현재 : 아주대학교 인공지능학과 석박사통합과정

<관심분야> 분산 딥 러닝, GPU 기반 그래프 알고리즘

[ORCID:0000-0002-7520-1144]

노 병 희 (Byeong-hee Roh)



1998년 2월 : 한국과학기술원 전기및전자공학과

1989년 3월~1994년 2월 : 한국통신 연구원

1998년 2월~2000년 3월 : 삼성전자 연구원

2014년 3월~2015년 2월 : 국방과학연구소 객원 연구원

2000년 3월~현재 : 아주대학교 소프트웨어학과 교수

<관심분야> 유/무선 멀티미디어 통신 응용, 트래픽 제어, IoT 플랫폼, 미래인터넷, 네트워크보안, 국방전술통신 네트워크, 증강현실, 혼합현실(Mixed Reality), 재난통신

[ORCID:0000-0003-2509-4210]

오 상 윤 (Sangyoon Oh)

2006년 : 인디애나대학교 컴퓨터공학과 박사

2006년~2007년 : SK 텔레콤

2007년~현재 : 아주대학교 소프트웨어학과 교수

<관심분야> 분산/병렬 컴퓨팅, HPC, 빅데이터처리, 클라우드

[ORCID:0000-0001-5854-149X]