SDN 환경에서 동적 흐름 제어를 위한 네트워크 인지 어플리케이션 구현

이동규°, 홍충선*
경희대학교 컴퓨터공학과
{lidoobil, cshong}@khu.ac.kr

Network-aware Based Dynamic Flow Control Application On Software Defined Network Environment

Dongkyu Lee°, ChoongSeon Hong*
Department of Computer Science and Engineering, Kyung Hee University

요 약

Software Defined Network (SDN)이라는 기술은 현재 네트워크 구조의 유동성 및 확장성 등의 문제들을 해결할 수 있는 대안으로 떠오른다. 본 논문에서는 SDN 환경의 테스트 베드에서 네트워크 상황을 인지 할 수 있는 네트워크 인지 어플리케이션을 구현하고, 이를 통해 효율적인 흐름 제어 기법에 대해 연구를 진행한다. 본 논문에서 제안한 어플리케이션은 컨트롤러를 추상화하여 사용자의 편리성을 높이고, 스위치의 port들을 실시간 모니터링하여 동적 라우팅 환경을 제공한다. 또한 어플리케이션은 각 port의 트래픽 상황과 해당 링크의 대역폭을 고려한 흐름 제어 기법을 제안하여 기존 컨트롤러의 라우팅 알고리즘과 비교 분석 및 평가한다.

1. 서 론

클라우드, 가상화, 빅데이터, 사물인터넷 등 다양한 트렌드와 신기술이 네트워크 환경에 빠르게 흡수되고 있는 상황에 적응하기 위해서는 네트워크 시장은 폭발적으로 증가하는 트래픽을 감당해야 한다. 그러나 현재 네트워크 구조는 완제품으로 제공되는 네트워크 장비로 구성되어 있다. 이런 구조는 새로운 기능을 구현하기 위한 초기 도입 단가나 운용을 위한 비용이 높아 변화에 대한 대처가 유연하지 못하다[1]. 이에 따라 효율적인 네트워크 구조 개선을 위한 OpenFlow기반의 Software Defined Networking(SDN) 기술이 관심을 받고 있다. SDN은 네트워크 제어 기능이 패킷 포워딩 기능과 분리되어 직접적으로 프로그래밍 가능한 네트워크 구조이며, 이를 통해 네트워크 운영자는 네트워크 설계 및 운용을 간략화 할 수 있다. 또한 SDN에서 네트워크 장치들은 기존의 복잡한 프로토콜의 구현 및 동작 없이 단순히 제어 장치로부터의 명령만을 수행할 수 있다. 이처럼 SDN은 네트워크의 세부 구성 정보와 독립적으로 요구 사항에 따라 소프트웨어를 통해 통신망을 효율적으로 제어 · 관리 할 수 있다. 뿐만 아니라 비즈니스 요구에 따라 인프라 정책, 트래픽 전달 경로 등의 차별화하는 서비스를 손쉽게 개발하여 적용할 수 있게 된다. 이를 통해 SDN은 네트워크 서비스의 혁신을 가속화할 수 있다는 점에서 주목을 받고 있다[2].

본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학 ICT연구센터 육성 지원사업의 연구결과로 수행되었음(IITP-2016-(H8501-16-1015))

*Dr. CS Hong is the corresponding author

본 논문에서는 SDN 환경을 제공할 수 있는 OpenFlow 프로토콜과 OpenDayLight 컨트롤러를 통해 보다 효율적으로 흐름 제어를 할 수 있는 어플리케이션에 대하여 연구한다.

2. 관련 연구

2.1 SDN Architecture and Interfaces

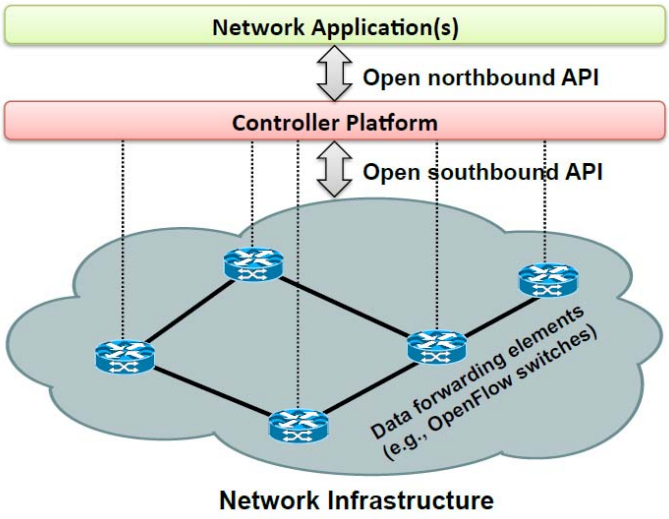

그림 1. SDN Architecture

그림 1[2]은 SDN Architecture와 인터페이스를 보여준다. “Southbound API” 는 Infrastructure 계층과 컨트롤러 계층 사이의 인터페이스를 나타낸다. 현재 SDN 구현에 쓰이는 대표적인 Southbound Interface는 OpenFlow protocol이다. “Northbound API” 는 컨트롤러와 어플리

케이션 계층 간의 인터페이스를 나타낸다. 이 인터페이스를 통해 네트워크 상태와 어플리케이션에 관한 정보 교환이 가능하다[3]. 본 논문에서는 어플리케이션 구현을 위해 컨트롤러 플랫폼으로 OpenDayLight 컨트롤러를 사용하였으며, Northbound-API로는 OpenDayLight Rest NBI[4]를 이용하였다.

2.2 네트워크 인지 어플리케이션 (Network-aware Application)

네트워크 인지 어플리케이션은 네트워크 성능을 측정하고 그 결과에 따라 행동한다. 이런 어플리케이션의 전형적인 예로는 Skype나 MPEG-DASH가 있다. 이는 네트워크 문제점을 극복하기 위해 어플리케이션 계층에서 비디오 코덱이나 품질을 변경하여 성능 조정을 한다. SDN에서도 비디오 스트림에서 QoS향상 방안에 대한 어플리케이션 연구는 많이 진행되었지만, 아직 개별 TCP Flow에 대한 동적 라우팅 연구는 많이 이루어지지 않았다[5]. 따라서 본 연구에서는 프로토타입으로 SDN에서 네트워크 상황을 인지하여 동적 흐름 제어가 가능한 어플리케이션을 구현한다.

3. 본 론

3.1 기존 연구의 문제점 및 제안 사항

기존 OpenDayLight는 기본적으로 Dijkstra Algorithm을 기반으로 경로를 설정한다. OpenDayLight의 Dijkstra SPF 모듈은 경로 결정을 할 때, 링크의 대역폭 상황을 고려하지 않는다.[6] 따라서 처음 경로 설정 시에 대역폭과 고려하지 않고 최단 경로를 찾는다. 또한 특정 링크에 트래픽이 급증하여 경로 변경이 요구되는 상황에서도 가능한 다른 경로로 변경되지 않는다. 이는 네트워크 자원의 비효율적 사용을 야기하고, 링크의 혼잡도 초래할 수 있다. 이를 해결하기 위해 본 연구는 두 가지 이슈를 제안한다. 첫 번째로, 어플리케이션은 처음 경로를 설정 시, Constrained Shortest Path First(CSPF) 알고리즘[7]을 통해 최소 대역폭을 고려한 최단 경로로 설정한다. 두 번째로, 실시간 port monitoring하여 트래픽 상황에 따른 동적인 흐름 제어를 운용한다.

3.2 제안 알고리즘

제안하는 알고리즘은 네트워크 토폴로지의 각 링크 대역폭들을 고려하여 초기 경로를 설정한다. 또한 스위치의 각 port의 단위시간당 Data rate에 기반하여 트래픽 상황을 고려하고 이를 통해 동적 흐름 제어를 수행한다. 그림 2는 본 논문에서 제안하는 동적 흐름 제어에 관한 의사코드이다.

그림 2를 보면 모니터링 할 스위치 ID와 링크 대역폭의 임계값을 정한다. 그 후 해당 스위치의 port에 흐르는 Data Bytes를 시간으로 나누어 단위시간당 Data rate를 구한다. 이렇게 스위치의 모든 port에 대해 현재 트래픽 상황을 알 수 있게 Data rate를 구한 후, 정렬하여 저장한다. 실시간으로 모니터링은 하지만 특정한 상황이 아닌 경우에도 Flow를 자주 변경하는 것은 오버헤드가 발생할 위험이 크다. 따라서 임계점을 설정하여 port의 Data rate가 설정한 임계점을 초과하였을 때만 Flow 변경이 이루어진다. 이때 Flow 변경은 최소의 네트워크 자원이 사용되는 port를 output port로 설정하여 자동으로 이루어질 수 있게 설계한다.

```
Input : nodeID, flowSpec, threshold
Output : rate, FlowEntry
1. JSONObject portData;
2. set threshold of bandwidth
3. while(1)
4.     sleep(5000)
5.     portData = portsInfo(nodeID)
6.     for(int i=0; i<portNum; i++)
7.         rtBytes = portData.get(i)
8.         rate = rtBytes/ durationTime
9.         outputArray[portNum], priority[portNum] = rate
10.    end for
11.    priority.sort()
12.    min = priority[1], max = priority[end]
13.    if (max> threshold)
14.        if(min == outputArray[1])
15.            flowSpec.output = 1
16.            FlowInstall(nodeID, flowSpec)
17.        else if(min == outputArray[2])
18.            flowSpec.output = 2
19.            FlowInstall(nodeID, flowSpec)
20.        ...
21.        else if(min == outputArray[portNum])
22.            flowSpec.output = portNum
23.            FlowInstall(nodeID, flowSpec)
24.        end if
25.    end if
26. end while
```

그림 2. port monitoring 기반 라우팅 알고리즘

4. 평가

4.1 테스트 베드 환경

본 테스트 베드에서는 VMWare을 통해 Mininet 2.2.1로 표 1과 같은 네트워크를 구성하였다.

표 1. 테스트 베드 환경

Controller	OpenDayLight Hydrogen Base 1.0
Bandwidth on link	100, 300, 500Mbit
File Sizes	1.5GB, 2.0GB, 2.5GB
Language	Java

4.2 시나리오

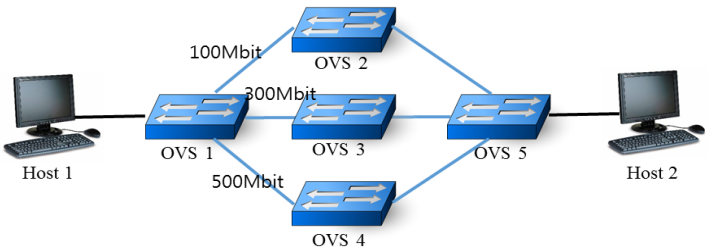

그림 3. 네트워크 토폴로지

그림 3과 같은 토폴로지에서 Host 1에서 Host 2까지의 경로는 총 3가지 (OVS1→2→5/ OVS1→3→5/ OVS1→4→5) 가 존재한다. 컨트롤러는 어플리케이션을 통해 최소 대역폭을 고려하는 CSPF 알고리즘을 기반으로 초기 경로를 지정한다. 그림3과 같은 토폴로지에서 최소 대역폭을 200Mbit으로 설정하면 CSPF 알고리즘에 따라 초기 경로는 (OVS1→3→5)가 된다. 이 상황에서 Host1에서 Host2로 대용량의 파일을 전송한다. 이때 링크에 설정한 임계점을 넘는지 확인하기 위하여 어플리케이션은 OVS1의 output port 각각을 모니터링 한다. 그리고 만약 대역폭에 설정한 임계점이 지나게 되면 다른 output port의 트래픽 상황을 고려하여 최적의 경로로 Flow가 변경되어 파일을 전송한다.

4.3 실험 결과

Node Connector	Pkts	Pkts	Rx Bytes	Tx Bytes
OF|3@00:00:00:00:00:00:00:01	322529	456632	21287338	1573106348
OF|1@00:00:00:00:00:00:00:01	511967	322542	2031838545	21287940
OF|2@00:00:00:00:00:00:00:01	5	55330	490	458732379
OF|4@00:00:00:00:00:00:00:01	5	4	490	392
SW|0@00:00:00:00:00:00:00:01	0	5	0	500

그림 4. OVS1의 Output Port 정보

시나리오를 통한 시뮬레이션 결과, 그림 3과 같이 2GB의 파일이 여러 port로 분산되어 전송된 것을 볼 수 있다. 또한 기존의 OpenDayLight와 비교했을 때, 다양한 파일 사이즈 모두에서 기존보다 효율적으로 패킷이 전송되었음을 알 수 있었다.

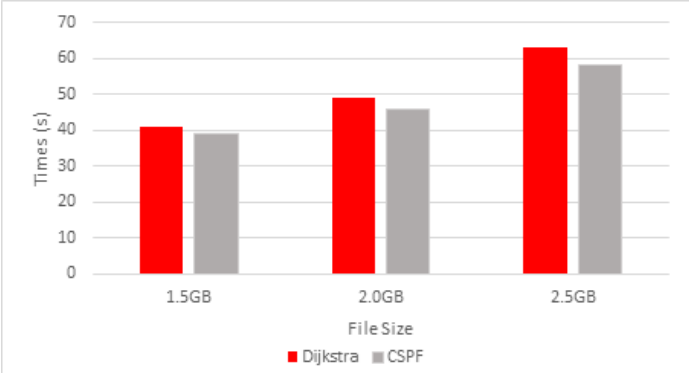

그림 5. 기존 알고리즘과 비교

5. 결론

본 연구는 네트워크 자원 중 대역폭과 트래픽 상황을 인지하여 사용자가 편리하게 경로 설정 및 흐름 제어를 할 수 있는 어플리케이션을 제안한다. 어플리케이션은 기존 OpenDayLight의 Dijkstra SPF와 다르게 동적 환경을 통한 CSPF 알고리즘 기반으로 경로 설정을 제안한다. 또한 스위치의 port들을 실시간 모니터링하고 트래픽 상황을 고려하여 Flow를 변경한다. 실험 결과, 기존보다 네트워크의 자원 사용률을 높일 수 있었고, 트래픽 상황을 고려하여 보다 효율적인 패킷 포워딩이 가능할 수 있다는 결과를 얻었다. 하지만 제안 사항에서 Flow를 변경하는 기준이 단순하다는 점과 모니터링 할 스위치 ID를 설정을 해줘야 한다는 점 등이 실제 네트워크 상황에 적용하기에 어렵다는 점이 있다. 따라서 향후 연구에서는 다양하고 실제 네트워크 환경에서도 활용할 수 있는 테스트 베드 구축과 보다 구체적인 판단 기준을 정리하는 데 초점을 둔다. 또한 나아가 단순히 현재 상황에 따른 대처뿐만 아니라 예측할 수 없는 상황에서도 최적의 흐름 제어를 할 수 있는 방안에 대한 연구도 필요하다.

참 고 문 헌

[1] 오픈소스를 활용한 OpenFlow 이해하기:SDN 입문, 서영석, 이미주 공저. 2014.

[2] D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C.Esteve, Rothenberg, S.Azodolmolky, S.uhling, “Software Defined Networking: A Comprehensive Survey,” proceedings of the IEEE, Vol. 103,pp.14−76, January 2015.

[3] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, and P. Tran−Gia, “SDN−based Application−Aware Networking on the Example of YouTube Video Streaming,” in 2nd European Workshop on Software Defined Networks (EWSDN 2013), (Berlin, Germany), Oct. 2013.

[4] OpenDayLight Rest NBI 사이트 :http://opendaylight.nbi.sdngeeks.com/

[5] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, W. Kellerer, “Dynamic Application−Aware Resource Management Using Software−Defined Networking: Implementation Prospects and Challenges“ in 2014 IEEE Network Operations and Management Symposium (NOMS), May. 2014.

[6] Z.K. Khattak, M. Awais, A. Iqbal, “Performance evaluation of OpenDaylight SDN controller”, Paralleland Distributed Systems (ICPADS), 2014 20th IEEE Conference on, pp.671−676, Dec. 2014.

[7] Das A, Sharafat A R, Parulkar G, McKeown N. "MPLS with a Simple OPEN Control Plane," Proc. of the Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference, Mar. 2011.