최소한의 에이전트 배치를 통한 비용 효율적인 SFC 모니터링 방식

이지수, 염상길, 추현승
성균관대학교 소프트웨어대학
e-mail:{jisoo49, sanggil12, choo}@skku.edu

A Cost-effective SFC Monitoring Approach with Minimum Agent Deployment

Jisoo Lee, Sanggil Yeoum, Hyunseung Choo
College of Software, Sungkyunkwan University

요 약

최근 다양한 네트워크 서비스에 대한 수요가 증가함에 따라 Service Function (SF)의 동적 구성을 위한 유연한 모델이 요구된다. Service Function Chaining (SFC)은 일련의 SF 로 구성된 새로운 네트워크 서비스 배포 모델을 정의한다. Software Defined Networking (SDN)은 제어 평면을 중앙 집중화함으로써 네트워크 트래픽 제어를 단순화하여 SFC 동작에 중요한 역할을 한다. SDN 기반 SFC (SD-SFC)는 SF 장애를 감지하기 위한 모니터링 시스템이 필요하다. 그러나 기존의 모니터링 방식은 모든 SF 에 Monitoring Agent (MA)를 배치하기 때문에 높은 시그널링 비용을 가진다. 본 논문에서는 최소한의 SF 에 MA 를 배치함으로써 시그널링 비용을 줄이는 SFC 모니터링 방식을 제안한다. 제안하는 SF selection 알고리즘은 최적화 된 SF 집합을 사용하여 오버로드 된 SF 를 반환하여 MA 를 배치한다. 우리는 제안 시스템의 효율성을 평가하기 위해 테스트베드 구현을 통해 실험하였다. 실험 결과에 따르면 우리는 기존 방식에 비해 시그널링 비용을 59.2 % 절감하였다.

1. 서론

인터넷 사용자에게 제공되는 네트워크 서비스는 운영자의 특정 요구 사항 및 정책을 충족하도록 정적으로 구성된다. 최근 트래픽의 꾸준한 증가로 서비스 관리가 복잡해짐에 따라 Service Function (SF)의 동적 구성을 위한 유연한 배치 모델이 요구된다. [1] Service Function Chaining (SFC)은 사전 정의된 SF 들의 집합으로 구성된 새로운 네트워크 서비스 배치 모델이다. SF는 트래픽 전송 경로인 Service Function Path (SFP)의 특정 위치에 배치된다. Software Defined Networking (SDN)은 데이터 평면과 제어 평면의 분리를 통해 SFC 간의 네트워크 트래픽을 제어하여 SDN 기반 SFC (SD-SFC)에서 중요한 역할을 한다. 제어 평면의 중앙 컨트롤러는 데이터 평면 내 가상 또는 물리적 SF의 정렬된 리스트를 결정한다.

그러나 SD-SFC에서 중요한 난제 중 하나는 동적인 트래픽 상황에서 적절한 SFP를 선정하는 것이다. [2] SFC의 성능은 관련된 모든 SF의 성능에 따라 달라진다. (즉, 단일 SF에 결함이 있거나 과부하가 발생하면 전체 SFC에 영향을 미칠 수 있다.) 따라서, SD-SFC에서 SF의 성능을 기반으로 SFP를 동적으로 변경하기 위해 모니터링 시스템이 요구된다. 일반적으로 모니터링 방식은 agent-less 방식과 agent-based 방식의 두 가지 범주로 나뉜다. agent-less 방식은 인터페이스 또는 개방형 프로토콜을 통해 특정 플로우에 대한 플로우 통계를 수집한다. agent-based 방식에서는 에이전트 형태의 활성 프로브가 모니터링 대상에 배포되어 모니터링 정보를 수집한다.

agent-less 방식은 플로우 통계에서 모니터링 정보를 얻기 때문에 상대적으로 가볍고 구현하기 쉽다. 이 방식은 중앙 컨트롤러를 통해 모든 스위치의 플로우 통계를 폴링한다. 플로우 메시지를 통해 특정 플로우의 트래픽 정보를 얻기 때문에 대역폭 병목 현상이 발생할 수 있고 실시간 정보를 얻을 수 없다. 반대로 agent-based 방식은 메모리, CPU 및 네트워크에 대한 다양한 정보를 실시간으로 측정하고 수집할 수 있다. 따라서 빠른 SFP 복구를 위해 즉각적인 모니터링 및 문제 감지가 필요한 SD-SFC 환경에서는 agent-based 방식이 더 유리하다. 그러나 이 방식은 에이전트 배포 및 시그널링 비용 최적화를 고려해야 한다.

본 논문에서는 최소한의 에이전트를 배치하여 시그널링 비용을 줄이는 SFC 모니터링 방식을 제안한다. SF selection 알고리즘은 ingress SF의 서비스율와 egress SF의 도착률 사이의 차이를 임계값과 비교하여 SF 오버로드의 가능성을 판단한다. 제안 방식은 모든 SF에 에이전트를 배치하는 기존 방식에 비해 59.2% 정도 시그널링 비용을 절감하였다.

2. 배경

2.1 오픈 소스 모니터링 솔루션: Zabbix

Zabbix는 확장성이 뛰어난 오픈 소스 모니터링 솔루션으로 다양한 메트릭을 실시간으로 모니터링하고 수집한다. [3] Zabbix는 크게 에이전트와 서버로 구성된다. 에이전트는 모니터링 대상에 배포되어 대상의 모니터링 정보를 측정하여 서버로 보낸다. 서버는 에이전트와의 상호 작용을 위한 가용성 및 무결성 정보를 보고 받는 중앙 구성 요소이며 모니터링 대상에 문제가 발생할 경우 관리자에게 알린다. 모든 구성 정보와 에이전트로부터 받은 데이터는 Zabbix 데이터베이스에 저장된다. Zabbix 프론트-엔드는 관리자가 서버에 쉽게 접근하기 위한 웹 기반 인터페이스이며, 모니터링 데이터 기반 실시간 그래프, 이벤트 기반 트리거, 액션 등 다양한 기능을 제공한다.

트리거 기능은 모니터링 데이터를 분석하여 모니터링 대상의 상태를 분석한다. 액션 기능은 트리거 발생 시 다양한 동작을 취할 수 있도록 한다. 관리자는 이러한 기능을 사용하여 모니터링 대상에 문제가 발생하였을 때 신속하게 대처할 수 있다. 예를 들어 관리자는 모니터링 대상의 CPU 사용률이 높아지면 이에 대한 경고 통지를 받을 수 있다. 트리거는 관리자에게 통지를 보내는 액션을 구성하기 위해 사전 정의된 임계값과 비교하여 대상의 상태를 관찰한다. 트리거 표현식이 참이면 상태가 OK에서 PROBLEM으로 변경되고 액션이 수행된다. 액션은 메시지 보내기, 원격 명령 실행, 호스트 추가 및 삭제, 호스트 활성화 및 비활성화 등 다양한 작업을 제공한다. 제안 시스템은 Zabbix의 트리거 및 액션 기능을 이용하여 SF selection 알고리즘을 구현한다.

2.2 관련 연구

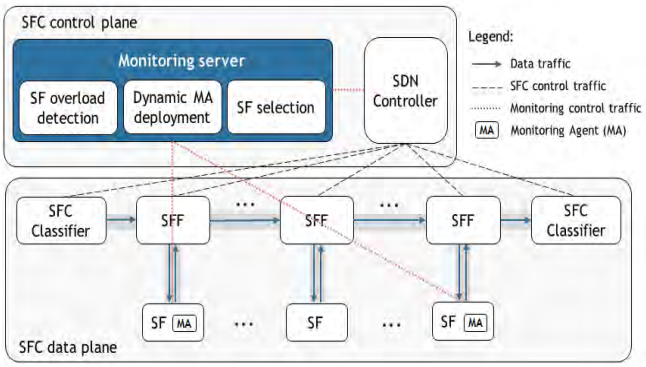
모니터링 접근 방식은 크게 두 가지 범주로 나뉘는데 agent-less 방식과 다른 하나는 agent-based 방식이다. agent-less 방식은 OpenFlow [4]와 같은 오픈 프로토콜이나 인터페이스를 사용한다. 이 방식은 SDN의 플로우 통계를 사용하기 때문에 가볍고 구현하기 쉽다. OpenTM [5]은 OpenFlow 컨트롤러에서 얻은 라우팅 정보를 사용하여 플로우 통계를 얻기 위한 스위치를 지능적으로 선택한다. 활성 플로우의 수에 따라 통신 비용이 증가한다. FlowSense [6]는 스위치의 플로우 통계를 전달받아 성능 변화를 모니터링 하는 푸시 기반 방식이다. 이러한 agent-less 방식은 특정 플로우 조건에서 모니터링 데이터의 정확성을 보장할 수 없으며 실시간 요구 사항을 충족시킬 수 없다.

반면에 agent-based 방식은 메모리, CPU 및 네트워크에 대한 다양한 정보를 실시간으로 측정하고 수집할 수 있다. 따라서, SD-SFC 환경에서는 세분화 된 성능 모니터링을 통한 경고 통지 기능을 제공하는 이 방식이 더 유리하다. Zabbix와 같은 기존의 agent-based 방식은 다양한 모니터링 메트릭을 지원하며 불필요한 네트워크 오버 헤드를 발생시키지 않는다. 그러나 이러한 방식은 모든 대상에 에이전트를 배포하기 때문에 일반적으로 시그널링 비용이 높다. 기존 연구는 시그널링 비용 면에서 효율적인 모니터링 솔루션을 제공하지 않는다.

3. 제안 시스템

3.1 시스템 설계

그림 2는 SD-SFC 아키텍처에 모니터링 시스템을 통합 한 제안 시스템 설계이다. SFC Specification (draft-ietf-sfc-control-plane-06) [7]에 따르면, SD-SFC는 SFC 제어 평면과 SFC 데이터 평면으로 구성된다. SFC 제어 평면은 SDN 컨트롤러와 세 개의 모듈(SF overload detection, SF selection, Dynamic MA deployment)을 가지는 모니터링 서버로 구성된다. 제어 평면은 SF의 상태에 따라 SFP를 구성하고 Service Function Forwarder (SFF) 정책을 관리한다. 데이터 평면은 SFC 분류자, SFF 및 SF가 포함된다.



(그림 1) 시스템 설계

SDN 컨트롤러는 SFC 데이터 평면 내 구성 요소에 플로우 규칙을 삽입하여 SFP를 구성한다. SF에 문제가 있거나 스케일링 작업을 수행한 경우 해당 SF을 포함하는 SFP를 업데이트한다. 네트워크 트래픽이 들어오면 SFC 분류자는 각 네트워크 서비스를 식별하고 특정 SFF 및 SFP의 정보를 할당한다. SFF는 SFP의 정책을 기반으로 SF나 다른 SFF로 트래픽을 전송한다. SF는 방화벽, DPI (Deep Packet Inspection) 및 NAT (Network Address Translation)와 같은 미들 박스 기능을 수행하고 트래픽을 이전 SFF로 반환한다.

모니터링 서버는 MA로부터 SF에 대한 모니터링 정보를 수집하고 문제가 발생하면 SDN 컨트롤러로 알린다. 각 SF의 서비스율 및 도착률을 모니터링하고 이를 기반으로 트리거 및 액션 기능을 수행한다. SF Overload Detection 모듈은 ingress SF와 egress SF를 통과하는 트래픽의 속도를 이용해 오버로드 된 SF가 존재하는지 판단한다. 두 속도의 차가 임계값보다 큰 경우, 현재 SFC에 오버로드가 발생한 것으로 판단한다. Dynamic MA deployment 모듈은 추가 SF에 대한 모니터링이 요구될 때 동적으로 MA를 배치한다. SF Selection 모듈은 SFC 내 오버로드 된 SF가 감지되면 해당 SF에 대해 탐색한다.

3.2 SF Selection 알고리즘

SF selection 알고리즘은 최소한의 에이전트를 배치하여 오버로드 된 SF를 선정한다. 우리는 복잡성을 줄이기 위해 이진 탐색을 사용하기 때문에 내림차순으로 정렬된 서비스율 및 도착률 집합을 이용한다. 제안 알고리즘은 필요에 따라 동적으로 에이전트를 배치함으로써 배치 비용과 시그널링 비용을 줄이는 것을 목표로 한다.

Algorithm 1: Overloaded SF selection algorithm

Input: μ_i, λ_i, $threshold\ \alpha$, S
Output: sf_x
Initialisation : $s = 1,\ e = n,\ mid = 0,\ x = 0,$
$A = \{sf_1, sf_n\}$

```
1: function SelectSF(s, e)
2: if e − s > 1 then
3:     if μ_s − λ_e > α then
4:         mid = ⌊(s + e)/2⌋
5:         A = A + {sf_mid}
6:         if λ_mid − μ_mid > α then
7:             x = mid
8:         else
9:             if μ_s − λ_mid > α then
10:                return SelectSF(s, mid)
11:            else
12:                return SelectSF(mid, e)
13:            end if
14:        end if
15:    end if
16: end if
17: return sf_x
```

알고리즘 1은 제안하는 SF selection 알고리즘에 대한 의사 코드를 보인다. SFC 토폴로지는 $S = \{sf_i \mid i \in 1,2,..,n\}$와 같이 표현되며 S는 SF의 집합이다. 모니터링 에이전트가 배치된 SF의 집합인 A 는 $A = \{sf_k \mid k \le n,\ sf_k \in S\}$ 과 같이 표현된다. $M = \{\mu_i \mid i \in 1,2,\ldots,n\}$은 서비스 속도의 집합으로, μ_i는 1초 동안 sf_i로부터 나가는 트래픽의 통계량을 나타낸다. $\wedge = \{\lambda_i \mid i \in 1,2,\ldots,n\}$은 도착률의 집합으로, λ_i는 1초 동안 sf_i로 들어오는 트래픽의 통계량을 나타낸다. 우리는 ingress SF (즉, $sf_s \in S$)의 서비스율 λ_e와 egress SF (즉, $sf_e \in S, (e > s)$)의 도착률 λ_e를 사용한다. $\mu_s - \lambda_e$가 α로 표현되는 임계값보다 크면 동일한 플로우에 대한 SFP 내 SF이 오버로드 된 것을 의미한다. SF selection 알고리즘의 시간 복잡도는 $O(log\ n)$이며 여기서 n은 SFC 내 SF의 수를 나타낸다.

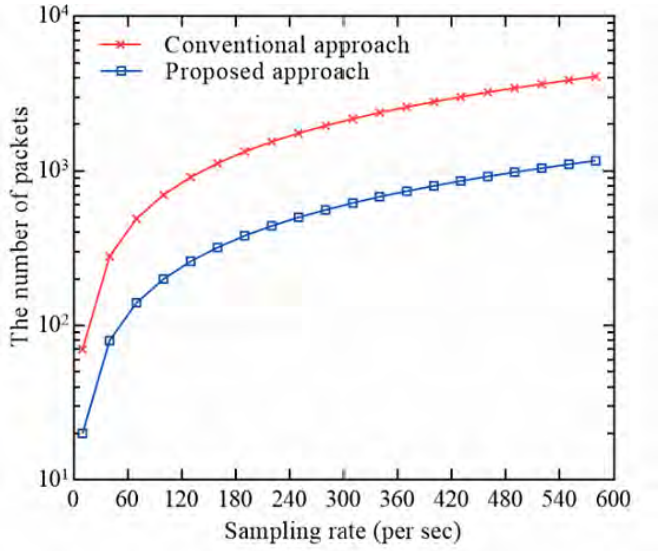
우리는 각 SF 로부터 얻은 서비스율, 도착률, 임계값을 입력으로 사용한다. 또한 집합 S 는 SF 요소를 집합 A 에 포함시키기 위해 사용된다. 오버로드 된 SF (즉, sf_x)는 출력으로 얻는다. 이 알고리즘의 반복적인 절차는 두 변수 s 와 e로 탐색 범위의 경계를 추적한다. 탐색 범위에서 첫 번째 SF 의 위치를 나타내는 변수 s는 1로 초기화되고 마지막 SF 의 위치를 나타내는 변수 e는 n으로 초기화된다. 중간 SF 의 위치를 나타내는 변수 mid와 오버로드 된 SF 의 위치를 나타내는 변수 x는 0으로 초기화된다. 집합 A는 초기에 두 개의 요소인 sf_1 와 sf_n 만을 포함한다.

우리는 다음 탐색 단계로 진행할지에 대한 여부를를 결정하기 위해 $e - s$ 값이 1 보다 큰지를 확인해야 한다. sf_s 와 sf_e 사이에 SF 이 없다면 탐색은 종료되어야 한다. (line 2) μ_s와 λ_e의 차이가 임계값 α 보다 큰 경우 mid 변수를 다시 계산하고 집합 A 에 sf_{mid}를 포함시켜 MA 를 배포한다. (line 3 에서 4) λ_{mid}와 μ_{mid}의 차이가 임계값 α 보다 크다면 sf_{mid}에 오버로드가 발생한 것을 의미한다. (line 5 에서 6) 그렇지 않으면 알고리즘은 다음 범위에 대해 계속해서 탐색한다. μ_s와 λ_{mid}의 차이가 임계값 α 보다 큰 경우 다음 범위의 첫 번째 위치와 마지막 위치인 매개 변수로 s 및 mid를 갖는 재귀 함수를 호출한다. (line 8 에서 9) 그렇지 않으면, 매개 변수로 mid와 e를 갖는 재귀 함수를 호출한다. (line 11) 오버로드 된 SF 가 존재하면, SF (즉, sf_x)이 반환되고, 그렇지 않으면, 변수 x는 탐색의 실패를 전달하는 존재하지 않는 SF (즉, sf_0)을 반환한다. (line 16)

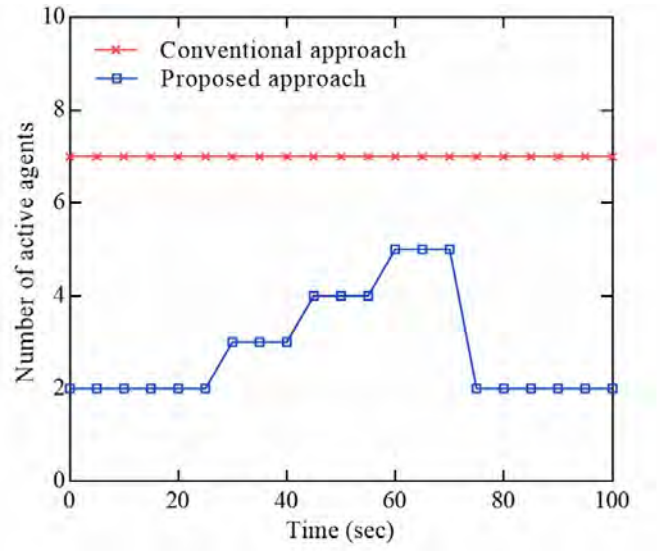
4. 성능평가

우리는 제안 방식의 성능을 평가하기 위해 테스트베드 기반의 실험 환경을 구현하였다. 테스트 토폴로지는 하나의 리눅스 서버와 7 개의 테스트베드로 구성되어있다. 우리는 Host A 에서 Host B 로 트래픽을 전송하기 위해 Iperf [8]라는 트래픽 생성 툴을 사용한다. 우리는 SDN 컨트롤러로 ONOS [9]라는 오픈 소스 SDN 컨트롤 플랫폼을 사용한다. 우리는 16G RAM 와 Intel(R) CPU 3.30GHz 4 코어 프로세서를 가지는 리눅스 서버에 ONOS 컨트롤러와 Zabbix 서버를 실행한다. 테스트베드에는 네트워크 플로우를 제어하기 위한 Open vSwitch (OvS) 2.5.2 와 모니터링 데이터를 측정하는 Zabbix 에이전트를 설치한다. 실험에서는 모니터링 데이터 및 네트워크 트래픽 전송을 위해 2 개의 브릿지를 사용한다. 리눅스 브릿지는 리눅스 서버와 테스트베드간의 이더넷 인터페이스를 연결한다. OvS Bridge 는 OvS 가 IP 트래픽이 일반적인 스위치처럼 동작할 수 있도록 플로우 규칙을 추가한다.

우리는 제안 방식의 성능을 평가하기 7 개의 SF 로 구성된 SFC 환경에서 단일 SF 에 오버로드가 발생한 시나리오를 고려한다. 만약 오버로드 문제가 발생한다면 트래픽 병목 현상이 발생하여 이후 트래픽의 속도가 감소할 것이다. 우리는 이와 같은 도착률의 변화를 구성하기 위해 out-port 의 대역폭을 변경하여 네트워크 혼잡을 발생시켰다. 우리는 시그널링 비용을 성능평가 지표로 사용하며, Zabbix 서버가 MA 를 폴링하는 횟수와 활성 MA 의 수를 통해 얻는다. 모든 SF 에 MA 를 배치하는 기존 모니터링 방식보다 낮은 시그널링 비용을 갖는 것을 확인한다.



(그림 3) 샘플링 속도에 따른 시그널링 비용 비교



(그림 4) 시간에 따른 활성 에이전트 수 비교

그림 4 는 혼잡을 발생시키기 전에 샘플링 속도에 따른 패킷의 수에 대해 제안 방식과 기존 방식을 비교한 그래프이다. 2 개의 SF 에만 MA 를 배치한 제안 방식은 7 개의 SF 모두에 MA 를 배치한 기존 방식보다 적은 패킷의 수를 가진다. 이 결과는 제안하는 모니터링 방식이 기존의 방식의 시그널링 비용을 71.4% 정도 줄인 것을 의미한다. 그림 5 는 활성 에이전트의 수에 대해 제안 방식과 기존 방식을 비교한 그래프이다. 제안 방식은 혼잡 발생 가능성을 감지한 후 에이전트를 동적으로 추가하고 오버로드 된 SF 를 발견하면 초기 상태로 돌아온다. 전체 과정에서 제안 방식의 최대 활성 에이전트 수는 5 개로 기존 방식과 비교하여 여전히 효율적이다. 이 결과는 제안 방식이 MA 를 최대로 배치하더라도 기존 방식의 시그널링 비용을 28.6% 정도 줄인 것을 의미한다. 결론적으로 제안 방식은 전체 혼잡 회복 과정에서 평균적으로 59.2% 정도 시그널링 비용을 절감하였으며 이를 통해 기존 방식보다 비용 효율적인 것을 확인하였다.

5. 결론

본 논문에서는 SD-SFC 에서 최소한의 에이전트만을 배치하여 비용 효율적인 SFC 모니터링 방식을 제안하였다. 우리는 Zabbix 모니터링 솔루션을 사용해 SF selection 알고리즘을 구현하여 모니터링 제어 시그널링 비용을 줄였다. 테스트베드 기반의 실험에서는 제안 방식이 시그널링 비용에 있어 모든 SF 에 에이전트를 배치한 기존 방식보다 약 59.1% 줄인 것을 보였다. 우리의 향후 연구는 오버로드 감지 정확도와 지연시간 간의 균형을 고려하여 적절한 임계값을 결정하는 것이다. 또한 스위치 및 SF 를 추가함으로써 보다 현실적인 SFC 환경을 구성하고 실험할 계획이다.

ACKNOWLEDGEMENT

본 논문은 기초연구사업 (NRF-2010-0020210)과 과학기술정보통신부 및 정보통신기술진흥센터의 Grand ICT연구센터지원사업 (IITP-2018-2015-0-00742), 방송통신인프라 원천기술개발사업 (2014-3-00547, 자율 제어 네트워킹 및 자율 관리 핵심 기술 관리)의 연구결과로 수행되었음

참고문헌

[1] ONF White Paper. (2015 June) ONF TS-027: L4-L7 Service Function Chaining Solution Architecture, [online] Available: https://www.opennetworking.org/

[2] J. Lee, H. Ko, D. Suh, S. Jang, and S. Pack, "Overload and failure management in service function chaining," Network Softwarization (NetSoft), 2017 IEEE Conference on. IEEE, 2017.

[3] https://en.wikipedia.org/wiki/Zabbix

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

[5] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic matrix estimator for OpenFlow networks," international Conference on Passive and Active Network Measurement. Springer, Berlin, Heidelberg, 2010.

[6] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: monitoring network utilization with zero measurement cost," international Conference on Passive and Active Network Measurement. Springer, Berlin, Heidelberg, 2013.

[7] M. Boucadair, “Service function chaining (sfc) control plane components and requirements,” IETF SFC WG, Tech. Rep., may 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-sfccontrol-plane-06.

[8] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs, “Iperf-The TCP/UDP bandwidth measurement tool. [Online],” Available: https://iperf.fr/

[9] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayshi, “ONOS: Towards an Open, Distributed SDN OS," Proc. of ACM HotSDN 2014, Aug. 2014.