소프트웨어 정의 네트워킹에서 플로우 프로토콜을 고려한 플로우 엔트리 방출 전략

최한힘나라*, Syed Muhammad Raza**, 김문성****, 추현승***
*성균관대학교 인공지능학과
**성균관대학교 전자전기컴퓨터공학과
***성균관대학교 소프트웨어대학
****서울신학대학교 교양학부
{hanhimy, choo, s.moh.raza}@skku.edu, moonseong@stu.ac.kr

Flow Protocol based Flow Entry Eviction Strategy for Software Defined Networking

Hanhimnara Choi*, Syed Muhammad Raza**, Moonseong Kim****, and Hyunseung Choo***
*Dept. of Artificial Intelligence, Sungkyunkwan University
**Dept. of Electrical and Computer Engineering, Sungkyunkwan University
***College of Software, Sungkyunkwan University
****Dept. of Liberal Arts, Seoul Theological University

요 약

소프트웨어 정의 네트워킹(SDN)은 기존 네트워크의 컨트롤 기능을 컨트롤러에 중앙 집중화하여 프로그램 가능하며 유연한 네트워크 관리 방식을 제공한다. 컨트롤러-스위치 간 인터페이스 표준 기술인 OpenFlow에서는 스위치 내부의 플로우 테이블을 이용하여 네트워크에 접근하는 패킷을 처리한다. 하지만 OpenFlow 스위치에 주로 사용되는 TCAM의 부족한 용량으로 인해, 많은 트래픽이 발생하는 환경에서 플로우 테이블에 충분한 양의 플로우 엔트리를 설치하지 못하는 문제가 발생한다. 이 경우 플로우 테이블 오버플로우가 발생하는데, 네트워크에 새로이 진입하는 플로우를 위하여 기존의 플로우 엔트리를 방출시킬 필요가 있다. 이때, 활성화된 플로우 엔트리를 방출하게 되면 네트워크 컨트롤 오버헤드가 크게 증가할 위험이 있다. 따라서 오버플로우가 발생했을 때 어떤 플로우 엔트리를 방출시킬지 정하는 것이 중요하다. 이에 본 논문은 플로우 프로토콜 타입에 기반한 플로우 엔트리 방출 정책을 제안하여 효율적인 플로우 테이블 사용을 목표로 한다.

1. 서론

소프트웨어 정의 네트워킹(SDN)은 스위치의 데이터 계층과 컨트롤 계층을 분리하여 컨트롤 기능을 컨트롤러에 중앙 집중화하는 차세대 네트워킹 기술이다. 이를 통하여 SDN은 프로그램 가능한 네트워크 애플리케이션을 통한 유연한 네트워크 관리 방법을 제공한다. SDN에서는 컨트롤러와 스위치 간의 통신을 통하여 네트워크 관리를 수행하며 오픈플로우(OpenFlow)는 이 통신 표준 프로토콜 중 가장 대중적인 프로토콜이다. OpenFlow 기반의 SDN에서는 스위치 내에 하나 혹은 둘 이상 정의된 플로우 테이블을 통해 패킷을 처리한다. 플로우 테이블은 각 플로우의 패킷 처리 방식을 담고 있는 플로우 엔트리의 집합이다. SDN 컨트롤러는 OpenFlow에 정의된 메시지를 스위치와 주고받으며 플로우 엔트리 설치, 삭제, 변경 등을 수행할 수 있다.

OpenFlow 스위치는 통상 3진 내용 주소화 기억장치(TCAM)를 사용한다. TCAM은 빠른 데이터 탐색 속도를 제공하여 스위치에서 사용하기에 용이하지만, 데이터센터 네트워크와 같이 많은 양의 트래픽이 발생하는 곳에서 사용하기에는 그 용량이 부족하다는 문제점을 안고 있다. 스위치의 메모리가 충분하지 않을 경우, 플로우 테이블이 충분한 수의 플로우 엔트리를 수용하지 못하게 된다. 이로 인하여 플로우 테이블 오버플로우가 발생할 수 있다. 플로우 테이블이 가득 찬 경우, 새로운 플로우를 설치하기 위해서는 이미 플로우 테이블 내에 있는 플로우 엔트리를 방출시켜야한다. 이때, 플로우 테이블이 가득 찬 상황에서 처리해야할 패킷이 아직 많이 남아있는 플로우 엔트리가 새로운 플로우 엔트리 설치로 인해 방출된다면 오버플로우가 다시 발생하게 될 가능성이 높다. 따라서 테이블 미스로 인한 컨트롤

러와 스위치 간의 부가적인 통신이 일어나게 되며, 활성상태인 플로우 엔트리의 예기치 못한 방출은 전송 제어 프로토콜(TCP) 윈도우를 줄어들게 하는 문제를 야기한다. 그러므로 오버플로우 상황에 대비하여 플로우 엔트리의 방출 우선순위를 정하는 정책은 네트워크 성능에 큰 영향을 미칠 수 있다. 이에 본 논문은 TCP와 사용자 데이터그램 프로토콜(UDP) 플로우의 특성을 모두 고려하여 플로우의 방출 우선순위를 매기는 정책을 제안한다.

2. 관련 연구

오버플로우가 발생했을 때 방출해야 하는 플로우 엔트리의 우선순위를 정하는 여러 연구들이 제안되어 온 바 있다. 먼저, 다양한 분야에서 쓰이던 전통적인 알고리즘들인 FIFO 혹은 Random과 같은 알고리즘을 이용한 방출 기법들이 있다 [1]. 그 외에, 플로우의 프로토콜 타입별 특성을 고려한 테이블 엔트리 관리 기법이 제안된 바 있다 [2]. 해당 연구는 TCP의 명시적 연결 종료 신호인 FIN 플래그를 플로우 테이블에서 매칭하여 TCP 플로우 엔트리를 조기에 방출하는 방법을 제시하였다. 하지만 UDP 플로우에 대해서는 단순히 엔트리 설치를 지연하는 데 그쳤으며, TCP와 UDP 플로우 모두를 고려한 방출 정책은 제시하지 않았다. 따라서 본 논문은 효과적으로 오버플로우에 대비하기 위해 프로토콜 타입에 기반한 플로우 엔트리 방출 정책을 제안한다.

3. 제안 아이디어

3. 1 시스템 구조

먼저, TCP 플로우의 경우 연결 종료 신호로 발생하는 FIN 플래그를 이용하여 우선적으로 방출해야하는 플로우의 우선순위를 매길 수 있다. 플로우 테이블에서 실시간으로 플래그 매칭을 수행하며 FIN 플래그가 매칭된 플로우가 발생하는 경우, 스위치는 해당 플로우 엔트리의 정보를 컨트롤러에 전달한다. 컨트롤러는 이 플로우 엔트리 정보를 전달 받아, FIFO 방식을 통해 오버플로우 발생 시 해당 플로우 엔트리가 가장 우선적으로 방출되도록 방출의 우선순위를 지정한다. 이때 플로우 엔트리의 목록들은 스위치가 특정 상황에 컨트롤러로 보내는 두 가지의 메시지 - 새로운 패킷이 스위치에 진입할 때 발생하는 packet_in 메시지와 플로우 엔트리가 테이블 내에서 삭제되는 경우 발생하는 flow_removed 메시지 - 들을 통해 업데이트 된다.

한편, 오버플로우가 발생했을 때 이전까지 FIN 플래그가 매칭된 TCP 플로우들이 목격되지 않았거나 이미 전부 방출되었을 수 있다. 이 경우 활성화된 TCP 플로우 엔트리를 방출하는 것은 위험이 있으므로 비활성화된 UDP 플로우를 우선적으로 방출한다. UDP 플로우는 명시적 연결 종료 신호를 보내지는 않지만 SDN 컨트롤러는 스위치에게 플로우 테이블 내 UDP 플로우 엔트리에 대한 통계 정보를 요청할 수 있다. 이 통계 정보에는 플로우 엔트리별로 매칭된 패킷의 수(packet_count) 가 있다. 컨트롤러는 이를 바탕으로 플로우의 대략적인 추세를 확인하여, 비활성화된 것으로 판단되는 플로우를 가려낼 수 있다. 이 과정은 다음과 같이 수행된다.

컨트롤러는 스위치에게 플로우 정보 요청 메시지(flow_stats_request)를 보낸다. 이후 컨트롤러는 스위치의 플로우 정보 회신 (flow_stats_reply) 메시지를 바탕으로 UDP 플로우의 정보를 업데이트한다. 또한 매 5초마다 해당 메시지를 교환하여 플로우 정보를 주기적으로 업데이트한다. 이렇게 교환된 정보를 바탕으로 UDP 플로우 매칭 정보와 해당 플로우의 packet_count 값들이 컨트롤러에 의해 키-값 구조의 데이터에 저장된다. 새로운 플로우 정보가 수신되면 packet_count 값은 최신 값으로 업데이트 된다. packet_count 값의 변화가 없는 플로우의 경우에는 일정 시간 매칭된 패킷이 없었으므로 비활성화된 것으로 판단, 먼저 비활성화된 플로우부터 우선적인 방출 대상으로 지정된다. 이외에, 우선적인 방출 대상으로 지정된 플로우의 리스트가 비어 있는 경우, UDP 플로우 엔트리 중 packet_count 값이 이전과 비교하여 가장 작은 수준으로 인상된 엔트리가 방출된다.

3. 2 시스템 구현

본 논문의 시뮬레이션 환경은 파이썬 기반의 SDN 프레임워크인 Ryu 컨트롤러와 미니넷을 사용하여 구현되었다. Ryu 컨트롤러에는 파이썬 스크립트를 이용하여 기본적인 스위치 구동방식 및 제안 정책이 정의된다. 본 실험에서는 컨트롤러와 스위치 하나, 그리고 스무 개의 호스트로 구성된 네트워크 토폴로지를 사용하였다. 미니넷에 구현된 각 호스트들은 서로가 receiver, sender가 되어 패킷을 주고받는다. 또한 네트워크에 충분한 수의 오버플로우, 즉 과부하를 주기 위하여 플로우 테이블 사이즈는 50개의 엔트리까지 수용할 수 있도록 설정하였다. 스위치는 OVS 스위치 2.5.5 버전을 사용하였으며

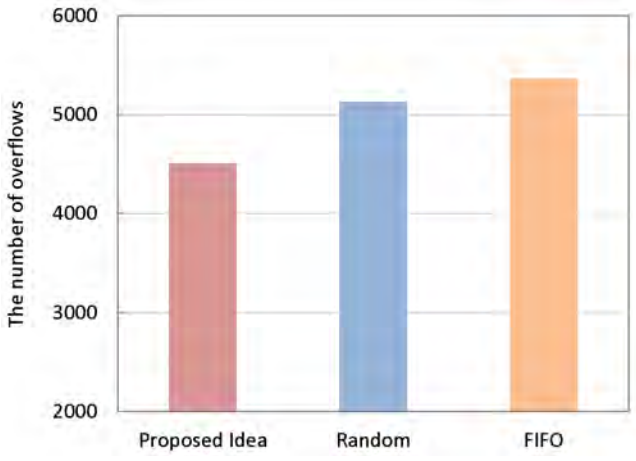
OpenFlow 1.3 버전을 사용하였다.

본 시뮬레이션에서는 플로우에 기반하여 생성된 트래픽이 사용되었다. 각 호스트 노드에서 iperf를 사용하여 발생된 실험 트래픽은 1,500개의 플로우를 생성하며 TCP와 UDP 플로우의 비율은 40:60으로 설정되었다. 1,500개의 플로우는 순차적으로 미리 설정된 간격을 가지고 생성되었으며 그 간격은 지수분포를 따른다.

3. 3 성능 평가

제안 시스템의 성능 평가를 위하여 실험 트래픽에서 제안 시스템과 비교 대상들의 시뮬레이션을 진행하였다. 성능 지표로는 오버플로우의 발생 횟수를 측정하였으며, 이를 통해 오버플로우가 발생했을 때 어떠한 정책이 더 효과적으로 플로우 엔트리를 방출하였는지 확인할 수 있다. 성능 비교 대상으로는 Random과 FIFO 기반 방출 정책을 사용하였다.

그림 1에서 실험 트래픽이 주어졌을 때 각 정책을 기반으로 한 환경에서 발생한 오버플로우의 횟수를 확인할 수 있다. 제안아이디어에서 오버플로우가 가장 적게 발생했으며 Random, FIFO 정책 순으로 오버플로우가 적게 발생한 것을 확인할 수 있었다. 우연에 크게 의존하는 Random 정책과 달리, FIFO 정책의 경우 제일 먼저 설치된 플로우 엔트를 방출하므로 지속적인 연결을 필요로 하는 플로우 엔트리를 방출하여 가장 많은 오버플로우를 발생시킨 것으로 볼 수 있다.



(그림 1) 각 정책에서 발생한 오버플로우의 횟수.

4. 결론

본 논문에서는 플로우의 프로토콜 타입을 고려하여 플로우 테이블 오버플로우에 대비한 플로우 엔트리 방출 정책을 제안하였다. 해당 정책을 통하여 기존의 플로우 엔트리 방출 정책보다 플로우 테이블 오버플로우 발생 횟수를 12% 이상 감소시킬 수 있었다. 더 나아가, UDP 플로우 엔트리 정보의 샘플링 주기와 비활성화 조건을 트래픽 환경에 따라 다르게 설정하는 등 보다 정교한 방식으로 설정하여 더욱 효율적인 방출 정책을 구성할 수 있을 것이다.

Acknowledgment

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 ICT명품인재양성사업(IITP-2020-2051-001), 글로벌핵심인재양성지원사업(IITP-2019-0-01579), 2020년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원(No.2019-0-00421, 인공지능대학원지원(성균관대학교))을 받아 수행된 연구임(NRF-2020R1A2C2008447).

참고문헌

[1] A. Zarek, Y. Ganjali, and D. Lie, “Openflow timeouts demystified,” Univ. of Toronto, Toronto, Ontario, Canada, 2012.

[2] S. Shirali-Shahreza and Y. Ganjali, "Delayed Installation and Expedited Eviction: An Alternative Approach to Reduce Flow Table Occupancy in SDN Switches," in IEEE/ACM Transactions on Networking, vol. 26, no. 4, pp. 1547-1561, Aug. 2018.