프로그래머블 데이터 플레인 언어의 확장성 비교

백호성, 김영호, 장석원, 한솔, 백상헌, 임용재
고려대학교 전기전자공학과
e-mail : {gh1emd, drkim05, imsoboy2, hs1087, shpack}@korea.ac.kr, yongjae.rim@gmail.com

Comparison of Extensibility of Programmable Data Plane Language

Hosung Baek, Youngho Kim, Seokwon Jang, Sol Han, Sangheon Pack, Yongjae Rim
School of Electrical Engineering, Korea University

요 약

본 논문에서는 현재 오픈소스 형태로 개발이 진행되고 있는 프로그래머블 데이터 플레인 언어, P4 와 Domino 를 언어의 확장성 측면에서 비교분석한다. 언어의 응용사례를 통해 각 언어의 활용 가능 범위를 살펴보고, 일반적인 스위치 모델로서의 적합성을 판단한다.

1. 서론

최근 소트프웨어 기반 네트워킹 기술 중 컨트롤 플레인을 위한 연구는 여러 글로벌 오픈소스 프로젝트들이 등장하면서 활발하게 진행되었고, 이에 따라 특정 장비에 대한 의존도가 점차적으로 낮아지고 있다. 반면, 데이터 플레인에 대한 연구는 특정 장비를 위한 개발 위주로 진행되어, 전용 장비에 대한 의존도가 여전히 높은 상태이다. 이를 해결하기 위해, 특정 장비에 종속되지 않도록 학계와 산업계에서 프로그래머블 데이터 플레인 구현을 위한 언어가 개발되고 있다. 대표적으로 오픈소스 형태로 개발되고 있는 P4[1], Domino[2] 등이 있다. P4[1]는 Barefoot Netoworks 회사를 주도로 개발되고 있는 데이터 플레인 프로그래밍을 위한 언어로, 현재 HUAWEI, CISCO 등 산업계뿐만 아니라, PRINCETON, Stanford University 와 같은 학계에서도 P4 컨소시엄에 참여하여 P4[1]를 개발하고 있다. Domino[2]는 글로벌한 데이터 플레인 state 를 요구하는 데이터 플레인 알고리즘을 구현하지 못하는 P4[1]의 제약사항을 해결하기 위한 언어로, MIT 를 주도로 개발되고 있다. 본 논문에서는 이 두 언어에 대해, 프로그래머블 데이터 플레인 언어의 요구사항 중 확장성 측면을 어느정도 충족되는지 각 언어의 응용사례를 통해 비교한다.

2. P4 의 응용사례

P4[1]의 응용사례로는 대표적으로 INT(In-band Network Telemetry)[3], HULA[4], Paxos[5], FlowRadar[6]가 있다. INT 는 메타데이터를 통해 데이터 플레인 자체적으로 모니터링 하고자 하는 스위치의 상태를 collecting 하고 보고할 수 있도록 하는 프레임워크다. 이를 통해 데이터센터 의 장애 및 성능 관리 기능을 향상시킬 수 있다.

다음으로, HULA[4]는 기존의 로드밸런싱 알고리즘인 ECMP(Equal-Cost Multi-Path routing)과 CONGA(Congestion-aware load balancing)[7]를 개선한 알고리즘이다. HULA 는 각각의 스위치에 대해 한 스위치로부터 목적지 스위치까지의 link utilization 과 swtich id 를 probe 를 통해 주기적으로 전송하고, 이를 홉단위로 저장한다. 즉, 각각의 스위치는 목적지 스위치까지 가는 경로 중 link utilization 이 제일 낮은 다음 홉에 대한 정보만 유지한다. 따라서 큰 메모리 사이즈가 요구되었던 CONGA[7]와는 대조적으로, 규모가 큰 topology 에서도 이 알고리즘을 적용할 수 있다.

Paxos[5]는 분산 네트워크 consensus 프로토콜로 데이터 플레인 상에서 구현하기는 복잡성이 높은 프로토콜이다. 이를 P4[1]를 이용하여 데이터 플레인 네트워크 서비스 형태로 구현하였고, 이를 통해 성능 개선이 가능하도록 하였다.

FlowRadar[6]는 기존의 모니터링 tool 인 Netflow 를 제한된 메모리와 패킷 처리 시간을 갖는 실제 데이터 센터 스위치에 구현하기 어렵기 때문에, 이를 해결하기 위해 한 플로우를 여러개의 해시 테이블에 저장하고 XOR 연산을 통해 한 해시 테이블에 다른 플로우도 중복으로 저장할 수 있도록 한다.

3. Domino 의 응용사례

다음으로, P4[1]의 match-action 모델에 의한 제한점을 개선하기 위해 개발되고 있는 Domino[2] 언어의 응용사례에 대해 살펴본다. Domino 의 응용사례를 보면 다음과 같다. Domino[2]는 패킷을 프로세싱하는 기능뿐만 아니라 디바이스의 스테이트를 읽고 쓰는 기능을 요구하는 stateful 프로세싱이 가능하다. 따라서, 이를 이용하여 P4[1]로는 구현하기 힘들었던 scheduling, active queue management 등이 Domino[2]를 통해 구현 가능하다. 대표적인 응용사례로, 패킷단위의 트래픽 분산 방법과 플로우단위의 트래픽 분산 방법의 장점을 결합하기 위해 패킷과 플로

우의 중간단위인 flowlet 단위로 트래픽 분산을 하는 Flowlet[8]을 Domino [2]언어로 구현하였다. 또한 CONGA[7] 등 여러 데이터 플레인 알고리즘을 P4[1]보다 더 작은 LOC(Line of Code)로 구현한 것을 확인할 수 있다.

다음으로, P4[1]로는 구현할 수 없는 스케줄러를 구현한 응용사례로, 패킷 스케줄러의 한 방법으로 제안된 PIFO(Push-In-First-Out queue)[9]가 있다. PIFO 는 다른 일반적인 스케줄링 알고리즘과 마찬가지로, 패킷을 어떤 순서로 처리할지, 언제 처리할지에 대해 스케줄링 한다. 순서에 대한 것은 priority queue 로, 시간에 대한 것은 calendar queue 를 통해 스케쥴링을 구현하였다.

위와 같이, Domino[2] 언어를 이용하여 P4[1]로 구현하기 어려웠던 여러 데이터 플레인 알고리즘을 구현한 것을 확인할 수 있다.

4. P4 와 Domino 의 확장성 비교

P4[1]는 데이터 플레인 상에 구현하기 어려웠던 알고리즘 및 프로토콜을 가능하도록 하였고, 기존의 컨트롤 플레인의 logic 을 P4[1]를 이용하여 데이터 플레인에서 수행할 수 있는 방향으로 개발되고 있다. 하지만 P4[1]를 이용하여 구현한 응용사례에는 여전히 제한사항이 있다. 현재 P4[1]를 특정 타겟에 올릴 수 있는 컴파일러가 공개되지 않았기 때문에, HULA[4]와 Paxos[5], FlowRadar [6]모두 P4 [1]언어를 이용하여 논리적으로 구현만 되어있는 상태이다. 실제 알고리즘의 시뮬레이션은 Network Simulator-2를 통해서 수행되었고, 새롭게 제시한 알고리즘이 기존의 알고리즘보다 성능적으로 더 낫다는 것만 보여진 상태이다. 즉, 실제로 P4[1]로 구현된 코드가 컴파일러를 통해 특정 타겟에 구현될 수 있는지는 확인이 되지 않은 상태이다.

또한 FlowRadar[6]의 경우, P4 로 [1]구현한 코드를 보면, 많은 And/XOR 연산과 단일 stage 로 구현되어 있다. 이는 실제 타겟이 될 하드웨어의 resource 에 대한 제약사항이 고려되어 있지 않기 때문에, 현실적으로 구현가능한지는 확인이 요구된다. 따라서 P4[1]의 활용 가능 범위는 P4[1]를 특정 타겟에 구현될 수 있도록 하는 컴파일러에 영향을 받는다. 또한 P4[1]는 match-action 모델을 기반으로 하는 언어이기 때문에, scheduling, active queue management 와 같은 복잡한 알고리즘을 구현하기에는 어렵다.

반면, Domino[2]는 기존의 match-action 모델을 기반으로 하는 P4[1]와는 달리, 여러 데이터 플레인 알고리즘을 표현하기에 충분한 프로세싱 unit 인 atom 으로 구성된 transaction 으로 보다 더 많은 알고리즘을 표현할 수 있는 것을 확인할 수 있다. 현재 기존의 알고리즘을 Domino[2]를 통해 구현한 단계이기 때문에 Domino [2]언어를 활용한 사례는 조금 더 확인이 필요하다. 하지만, Domino[2]는 P4[1]보다는 더 일반화된 구조로 구성되어 있어, 보다 더 넓은 범위에서 사용될 수 있다.

5. 결론

본 논문에서는 프로그래머블 데이터 플레인 언어, P4[1]와 Domino[2]에 대해 각각의 응용사례를 통해 각 언어의 확장성을 비교 분석하였다. P4[1]를 이용하여 더 넓은 범위에서 활용하고, L2~L3 계층뿐만 아니라, L1~L7 계층까지 적용될 수 있기 위해서는 P4[1]를 일반화시킬 수 있도록 하는 연구 방향이 제시되어야 한다. 또한 P4[1]의 제한점을 개선하기 위해 개발되고 있는 Domino[2]는 match-action 모델을 기반으로 하는 P4[1]보다 더 일반화된 구조로 되어 있어, 더 넓은 범위에서 활용될 수 있다. 하지만 아직 응용사례가 많지않기 때문에, 추후에 언어의 확장성에 대해 확인이 필요하다.

감사의 글

본 논문은 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (B0101-16-0233: 스마트 네트워킹 핵심 기술 개발, B0190-15-2012: 글로벌 SDN/NFV 공개 소프트웨어 핵심 모듈/기능 개발)

참고문헌

[1] P. Bosshart, *et al.*, "P4: Programming protocol-independent packet processors," *SIGCOMM Computer Communication Review,* Vol. 44, No. 3, pp. 87–95, July 2014.

[2] A. Sivaraman, *et al.,* “Packet transactions: High-level programming for line-rate switches,” in *Proc. SIGCOMM ‘16*, to appear.

[3] C. Kim, *et al.,* “In-band Network Telemetry via Programmable Dataplanes,” in *Proc. SIGCOMM’15*

[4] N. Katta, et al., “HULA: Scalable Load Balancing Using Programmable Data Planes,” in *Proc. SIGCOMM ‘16*, to appear.

[5] H. T. Dang, *et al.,* “Paxos Made Switch-y,” *SIGCOMM Computer Communication Review,* Vol. 46, No. 2, pp. 19–24, April 2016.

[6] Y. Li, *et al.,* “FlowRadar: A Better NetFlow for Data Centers,” in *Proc. SIGCOMM ‘16*, to appear.

[7] M. Alizadeh, *et al.,* “CONGA: Distributed Congestion-Aware Load Balancing for Datacenters,” in *Proc. SIGCOMM’14*.

[8] S. Sinha, *et al.,* “Harnessing TCP’s Burstiness with Flowlet Switching,” in *Proc. SIGCOMM’04.*

[9] A. Sivaraman, *et al.,* “Programmable Packet Scheduling,” in *Proc. SIGCOMM ‘16*, to appear.