프로그래머블 데이터 평면 기술 동향

20 July 2017

Sangheon Pack (백상헌)

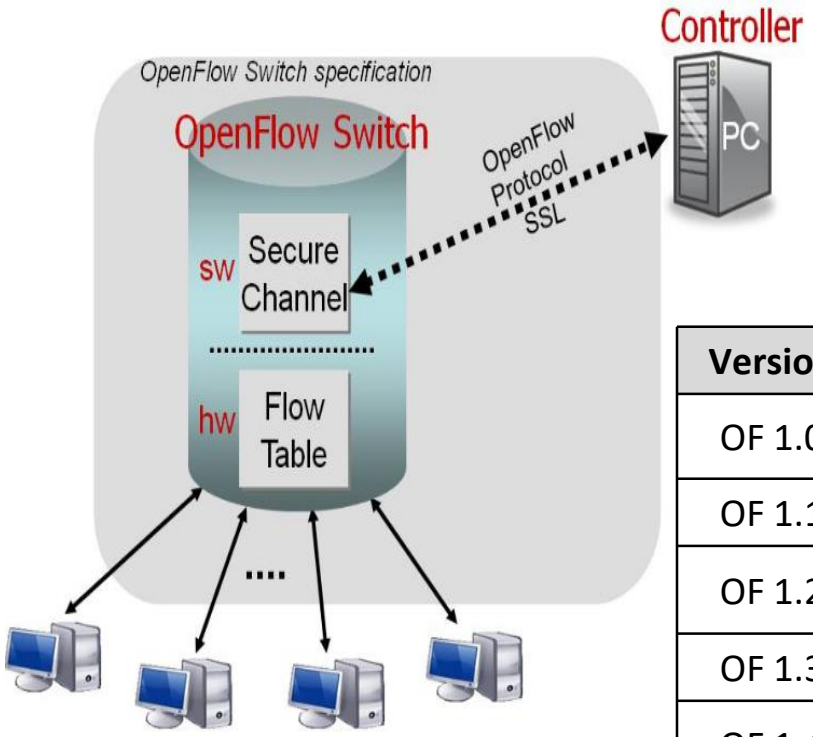
(shpack@korea.ac.kr)

Outline

- SDN 1.0 vs. SDN 2.0
- 프로그래머블 데이터 플레인 응용 분야
- 프로그래머블 데이터 플레인 하드웨어
- 프로그래머블 데이터 플레인 소프트웨어
- 결론
- 참고문헌

SDN 1.0: OpenFlow

- OpenFlow is a programmable tool.



Version	Date	Header Fields
OF 1.0	Dec 2009	12 fields (Ethernet, TCP/IPv4)
OF 1.1	Feb 2011	15 fields (MPLS, inter-table metadata)
OF 1.2	Dec 2011	36 fields (ARP, ICMP, IPv6, etc.)
OF 1.3	Apr 2012	40 fields (IPv6 Extension Header, etc.)
OF 1.4	Aug 2013	41 fields (Synchronized tables, etc.)
OF 1.5	Dec 2014	41 fields (Egress table, etc.)

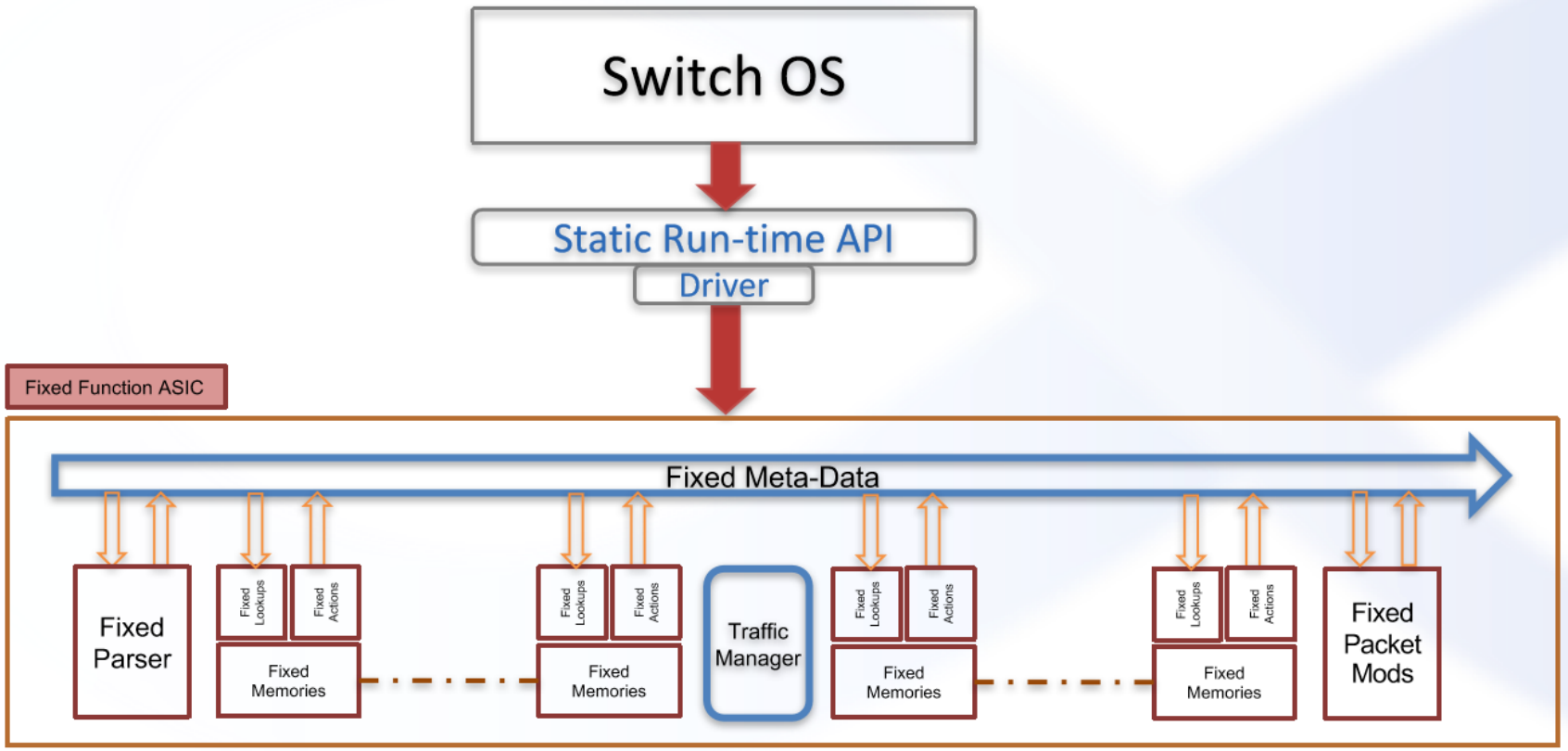
SDN 2.0

- Tenets of SDN 2.0 (from Scott Shenker at UCB)
  - 1) Software goes to the edge; the core stays dumb
  - 2) “Middleboxes” get included in SDN
  - 3) The network is opened up to third-party services
  - 4) Closed interfaces are not allowed
  - *“OpenFlow itself isn’t a bad protocol, But OpenFlow was meant to communicate new ideas to networking equipment. The limitations of ASICs — table sizes, for instance — prevented OpenFlow from getting some of those ideas across.”*

Limitations of OpenFlow

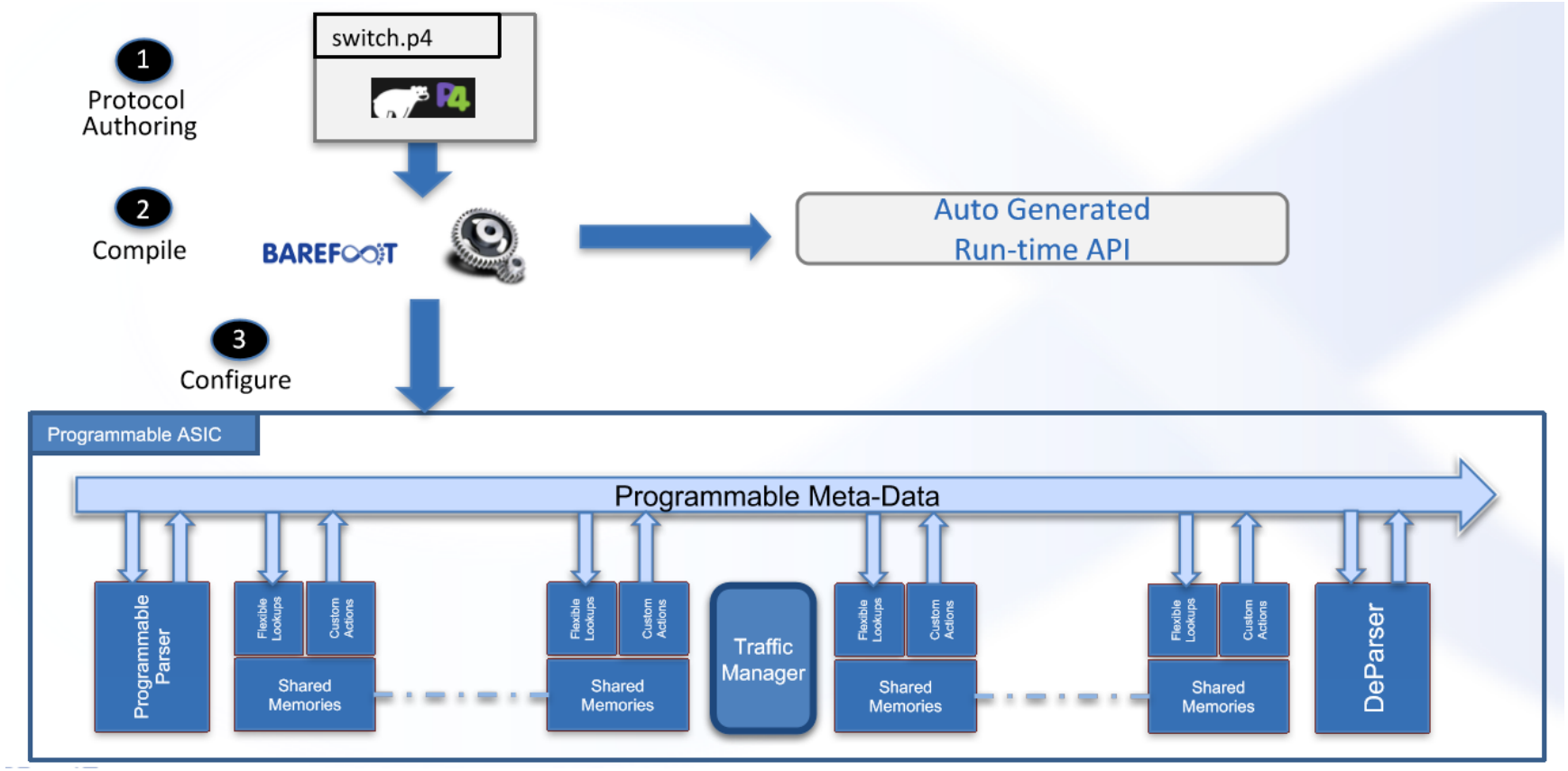
- OpenFlow is not perfect!
  - OpenFlow 버전업으로 인해 종속성이 발생
  - OpenFlow는 제한적인 확장성만을 제공하기 때문에 Parsing, match header field, action 등 전반에 걸친 유연성 확보가 필요함
  - 각 모듈에 대한 재설정 가능성 (Reconfigurability), 프로토콜 독립성 (Protocol Independence), 타켓 독립성 (Target Independence) 등을 목표로 하는 다양한 형태의 프로그래머블 데이터 플레인 기술이 출현함

As-Is



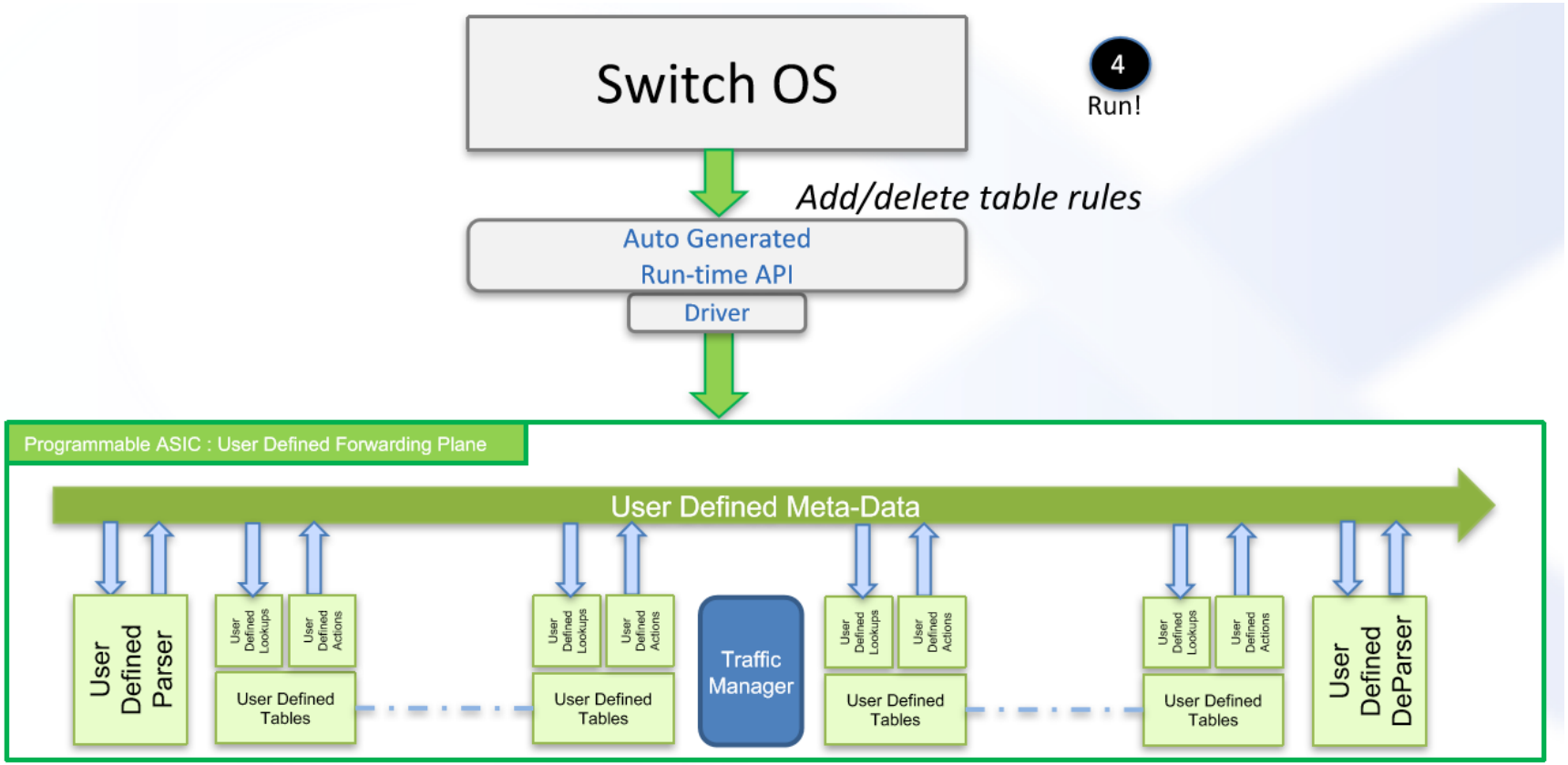
[Vladimir Gurevich, “Programmable Data Plane at Terabit Speeds”]

To-Be (1/2)



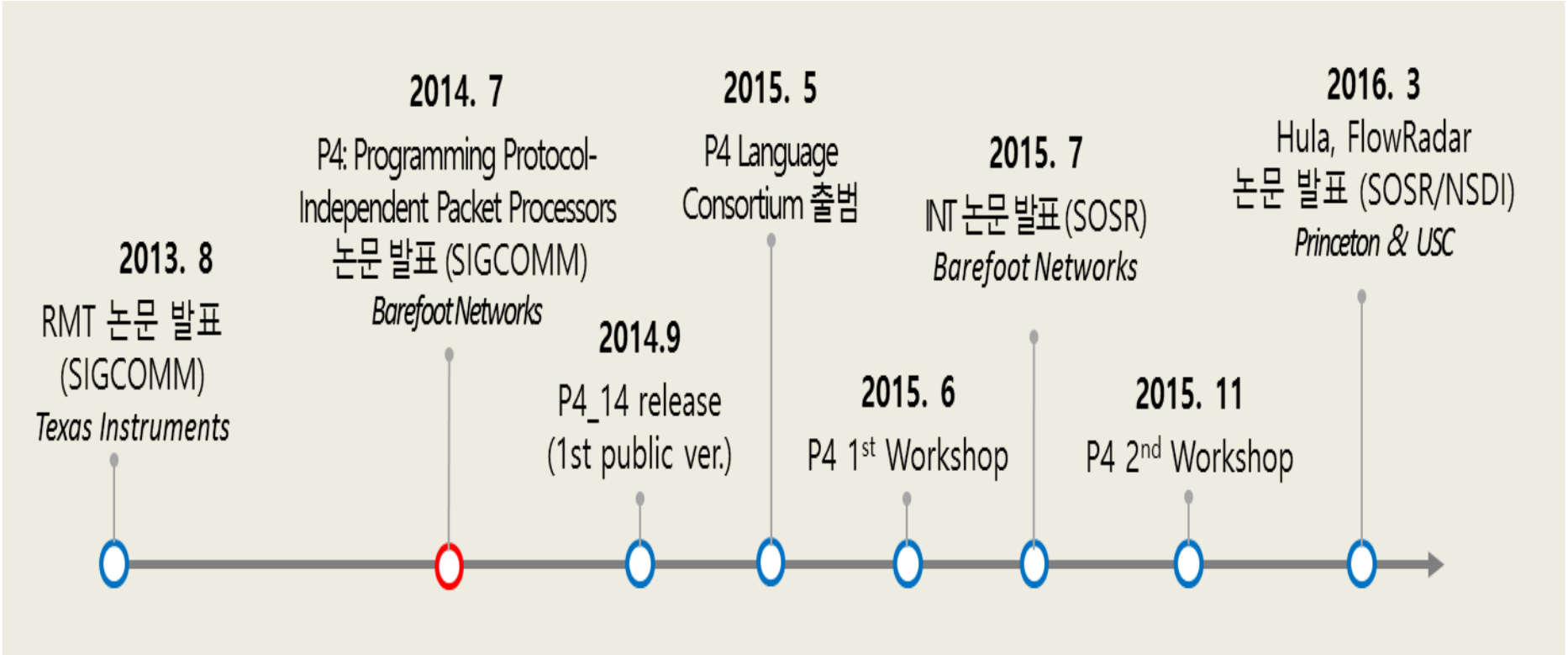
[Vladimir Gurevich, “Programmable Data Plane at Terabit Speeds”]

To-Be (2/2)

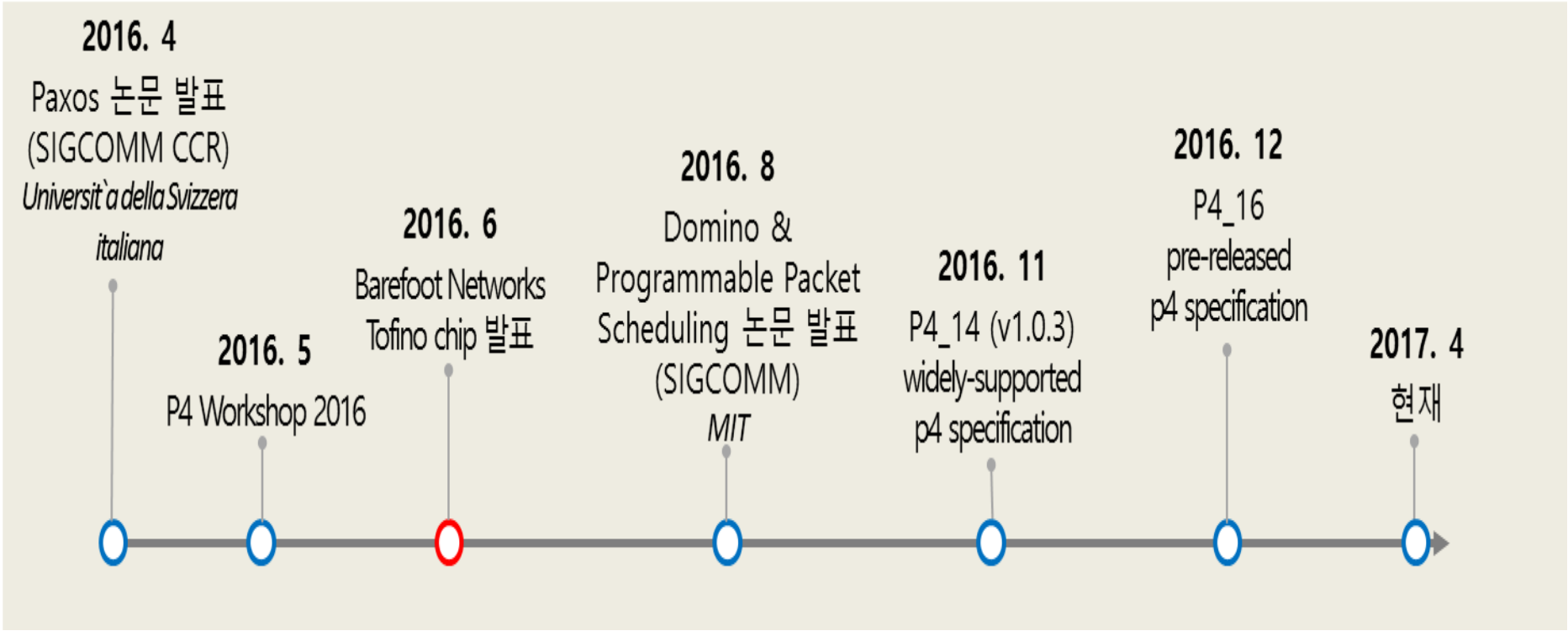


[Vladimir Gurevich, “Programmable Data Plane at Terabit Speeds”]

History (1/2)


2013. 8
RMT 논문 발표
(SIGCOMM)
Texas Instruments
2014. 7
P4: Programming Protocol-
Independent Packet Processors
논문 발표 (SIGCOMM)
Barefoot Networks
2014.9
P4_14 release
(1st public ver.)
2015. 5
P4 Language
Consortium 출범
2015. 6
P4 1st Workshop
2015. 7
INT 논문 발표 (SOSR)
Barefoot Networks
2015. 11
P4 2nd Workshop
2016. 3
Hula, FlowRadar
논문 발표 (SOSR/NSDI)
Princeton & USC

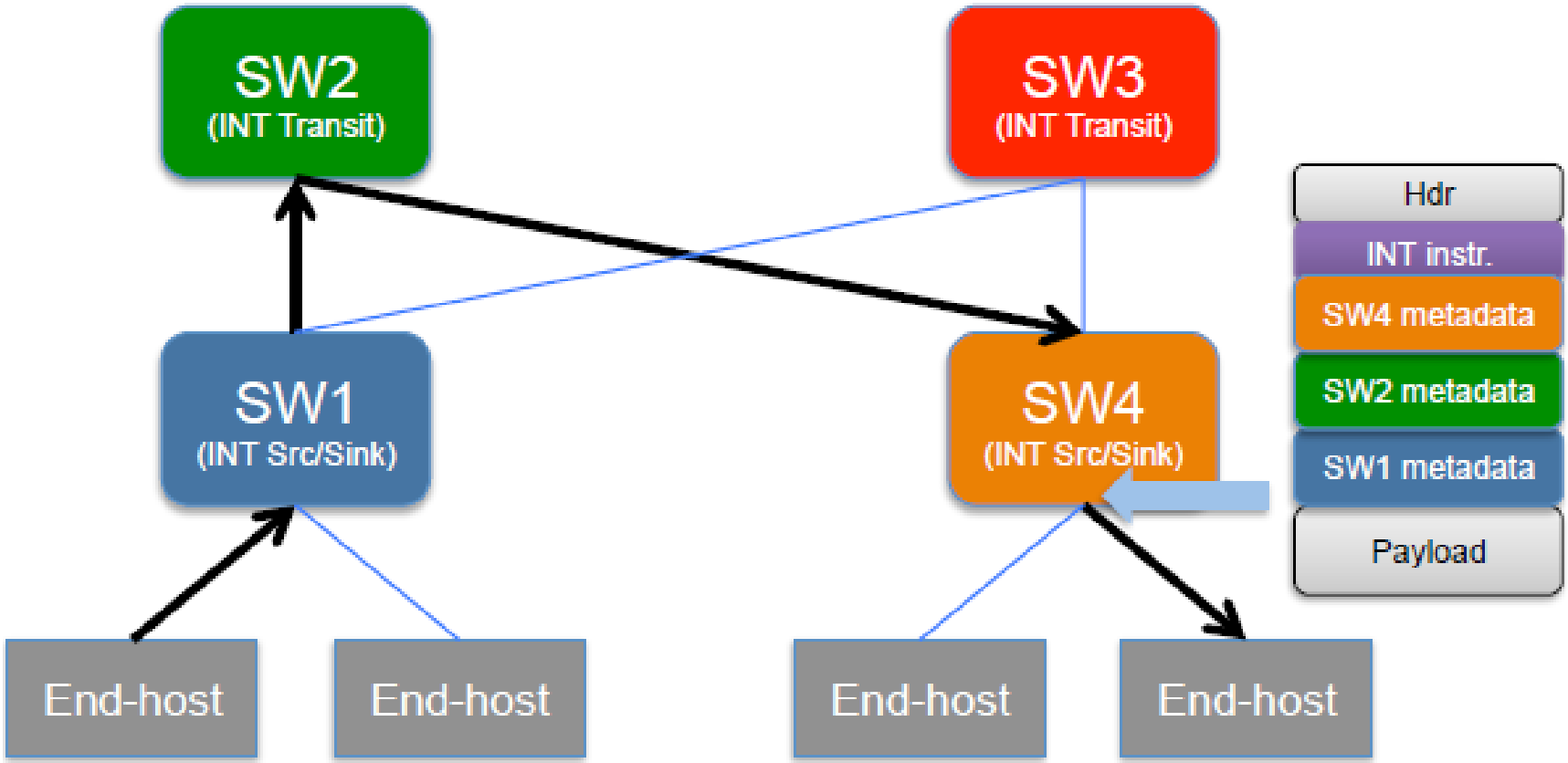
History (2/2)


2016. 4
Paxos 논문 발표
(SIGCOMM CCR)
Universit`a della Svizzera italiana
2016. 5
P4 Workshop 2016
2016. 6
Barefoot Networks
Tofino chip 발표
2016. 8
Domino &
Programmable Packet
Scheduling 논문 발표
(SIGCOMM)
MIT
2016. 11
P4_14 (v1.0.3)
widely-supported
p4 specification
2016. 12
P4_16
pre-released
p4 specification
2017. 4
현재

Application: In-band Network Telemetry (1/2)

- INT (In-band Network Telemetry) [Kim15]
  - 컨트롤러의 관여 없이 스위치를 통과하는 패킷에 INT 헤더와 스위치 메타 데이터를 삽입하여 상태 정보를 전달하는 기법; 삽입된 INT 메타데이터는 원격 모니터링 엔진에서 분석됨
  - 컨트롤러와 데이터 평면 간의 시그널링을 통해 이루어지던 주기적 제약을 없애고, 전달되는 모든 패킷에 메타 데이터 정보를 삽입하여 성능저하 없이 패킷 당 네트워크의 장애 및 성능의 관리/분석이 가능함

Application: In-band Network Telemetry (2/2)



Application: Load Balancer (1/2)

- HULA(Hop-by-hop Utilization-aware Load-balancing Architecture) [Kat16]
  - ECMP(Equal-Cost Multi-Path routing): 링크 및 스위치의 부하를 고려하지 않고 모두 같은 비용으로 분산
  - CONGA(Congestion-aware load balancing): 링크 및 스위치의 부하를 고려하지만, 이를 위해서 모든 경로에 대한 정보저장이 필요함
  - HULA는 링크 별 부하를 관리하면서도 이를 위한 정보의 양을 최소화 함으로써 성능 및 네트워크 규모에 대한 제약을 개선시킴

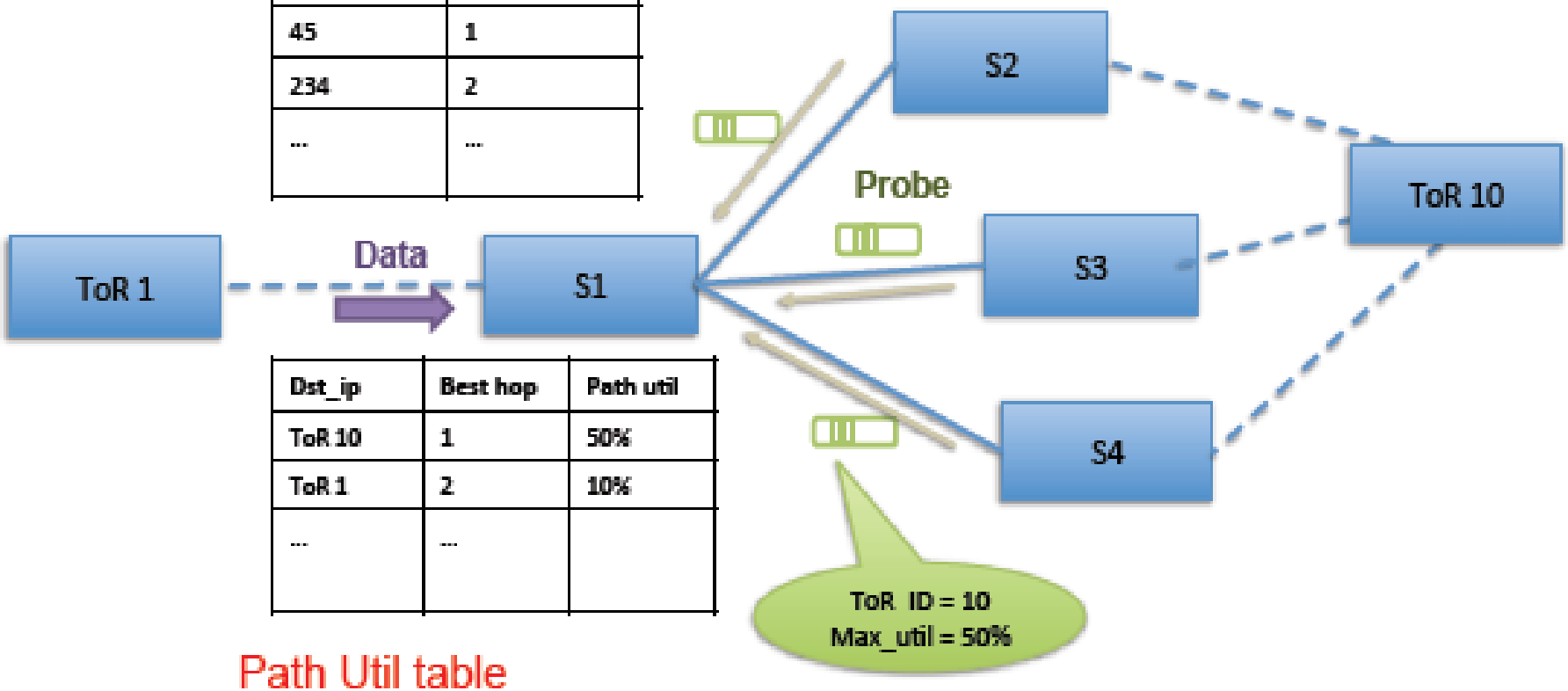
Application: Load Balancer (2/2)

P4 code

```
if(curr_time - flowlet_time[flow_hash] > THRESH) {
        flowlet_hop[flow_hash] =best_hop[packet.dst_tor];
}
metadata.nxt_hop = flowlet_hop[flow_hash];
flowlet_time[flow_hash] = curr_time;
```

Flowlet table

Flowlet_id	Next hop
45	1
234	2
…	…



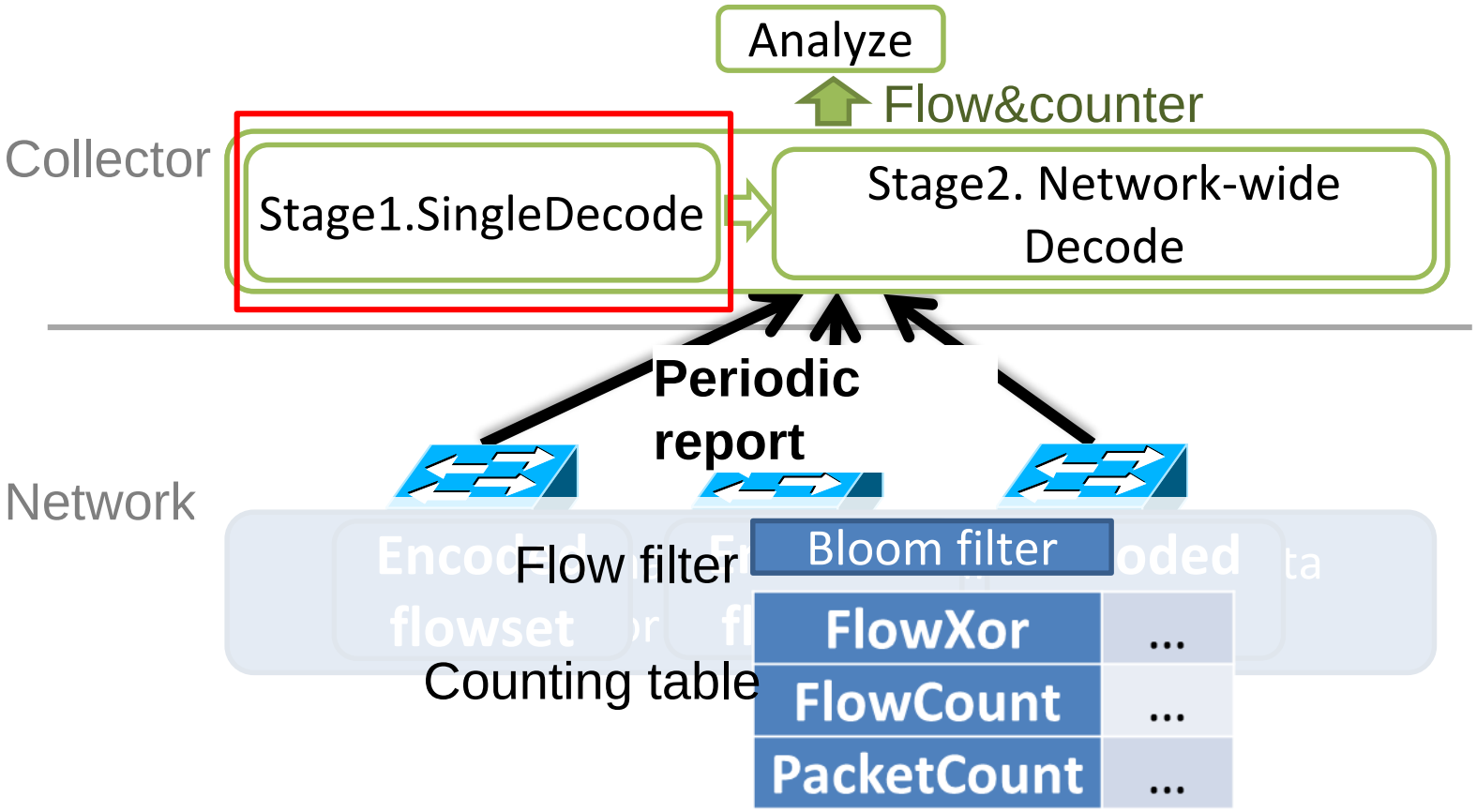
Dst_ip	Best hop	Path util
ToR 10	1	50%
ToR 1	2	10%
…	…	

Path Util table

Application: Network Monitoring (1/2)

- FlowRadar [Li16]
  - Netflow의 경우 메모리의 제한과 패킷 처리 시간 보장을 위해 데이터 센터 스위치에 구현하기 어려우며, 주기적 샘플링을 통해 모니터링을 수행함
  - FlowRadar는 데이터 평면상에 기능 구현을 통해 스위치에서 패킷 단위의 고정된 처리를 하도록 함. 메모리 사용을 줄이기 위해 하나의 flow를 여러 개의 hash 테이블에 저장하고, 중복된 hash 값으로 인해 발생하는 linked list 에 대한 불필요한 메모리 사용을 개선하기 위하여 XOR 연산을 사용

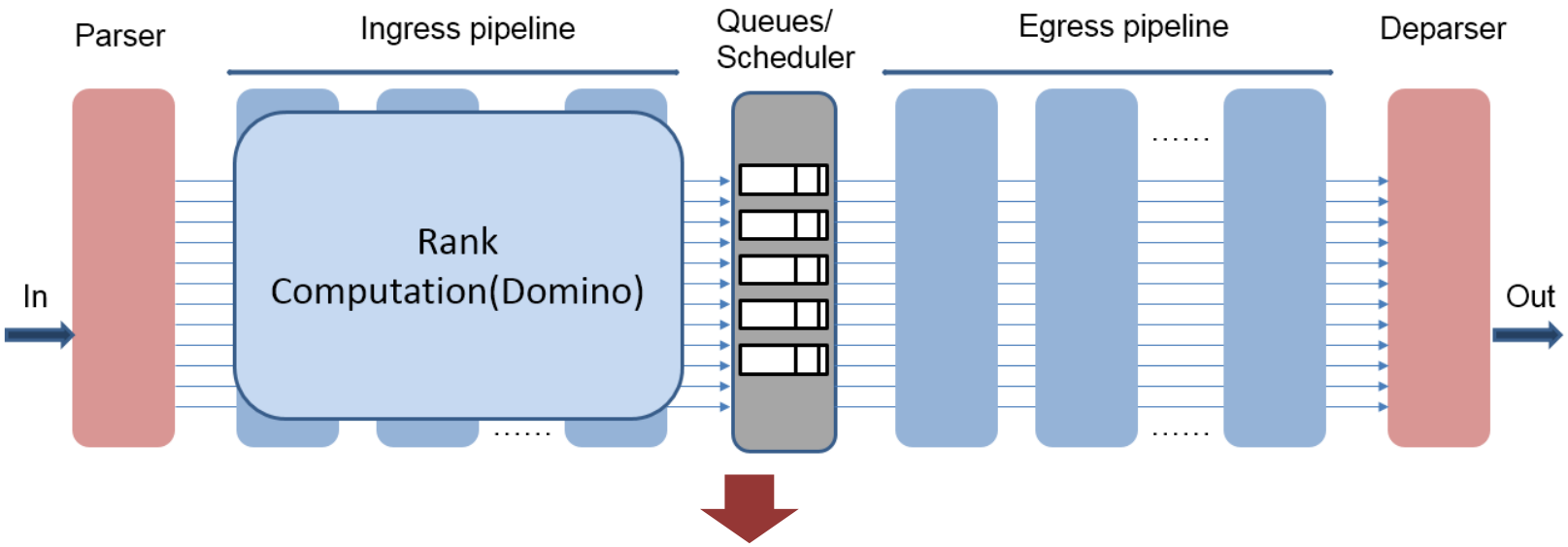
Application: Network Monitoring (2/2)



Application: Packet Scheduler (1/2)

- Abstract Scheduling Model [Siv16-2]
  - Push-in First-Out (PIFO) Queue: 미리 계산된 rank 값에 의해 패킷들이 queue안에 위치를 결정하게 되고 queueing 된 패킷들의 dequeue 과정은 rank의 순서가 높은 순서부터 처리함
  - 이를 통해, 존재하는 대다수의 scheduling 기법들을 PIFO 큐를 통해 Line-rate로 동작할 수 있음을 입증함

Application: Packet Scheduler (2/2)

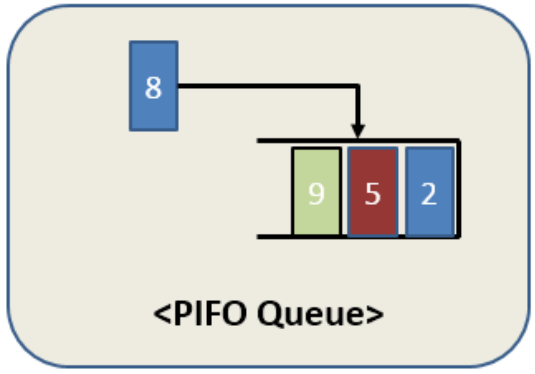


Programming Example of Fair Queueing

```
1. f = flow(p)
2. p.start = max(T[f].finish,
                 virtual_time)
3. T[f].finish= p.start + p.len
4. p.rank = p.start
```

<Packet Transaction>

Programmable



Fixed Logic

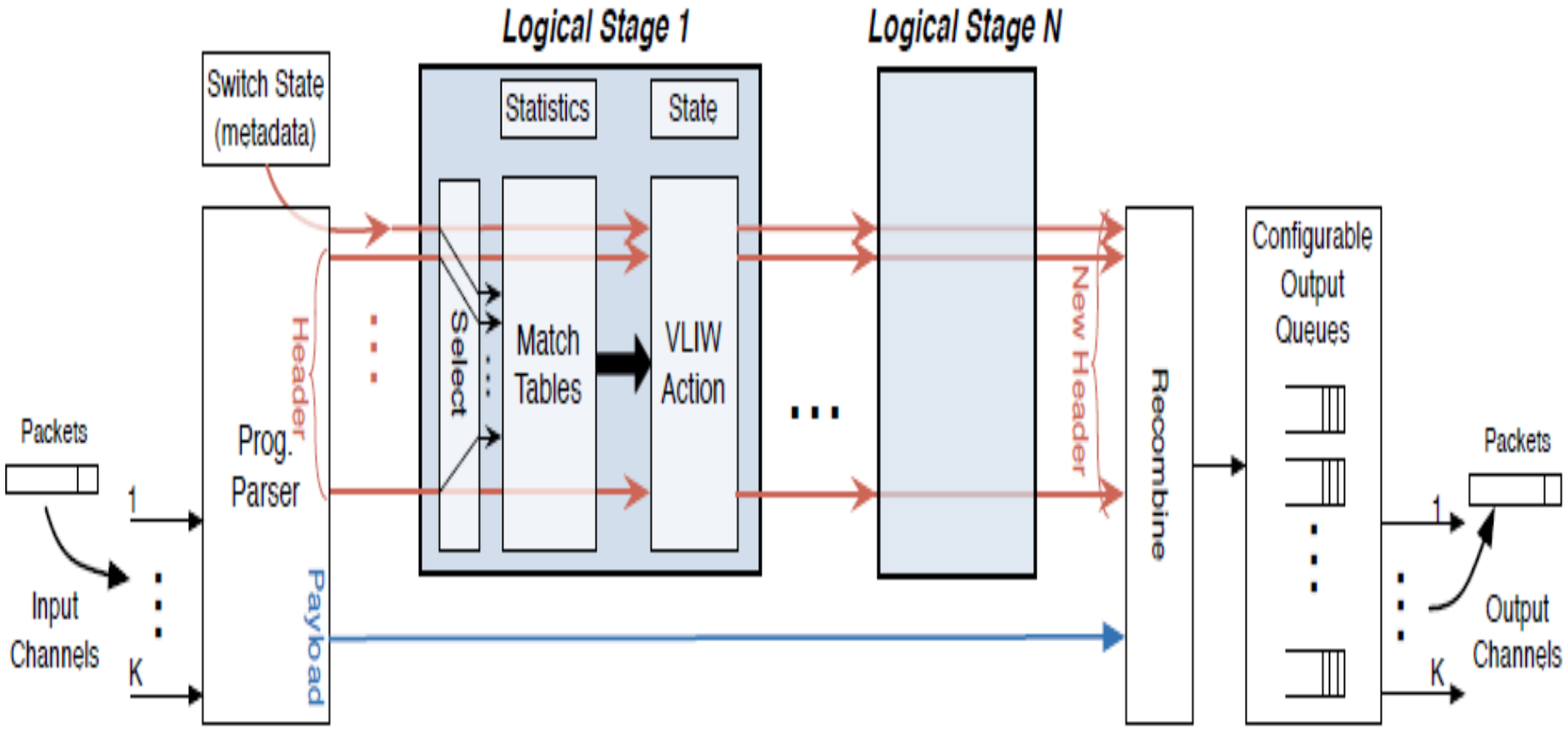
Hardware for Programmable Data Plane

Project	업체	지원 모델	지원 언어	발표시기
RMT	Texas Instruments	-	-	2013 3Q
FlexPipe	Intel	FM6000 series	자체 DPDK	2012 2Q
NFP-6xxx	Netronome	NFP-6xxx	Agilio P4C SDK (P4 & C)	2013 3Q
XPA	Cavium	CNX880xx	자체 SDK	2014 4Q
PISA	Barefoot Networks	Tofino	P4	2016 2Q
Virtex-7 FPGA	Xilinx	Virtex-7 FPGA	PX	2014 1Q

Reconfigurable Match Table (RMT) (1/2)

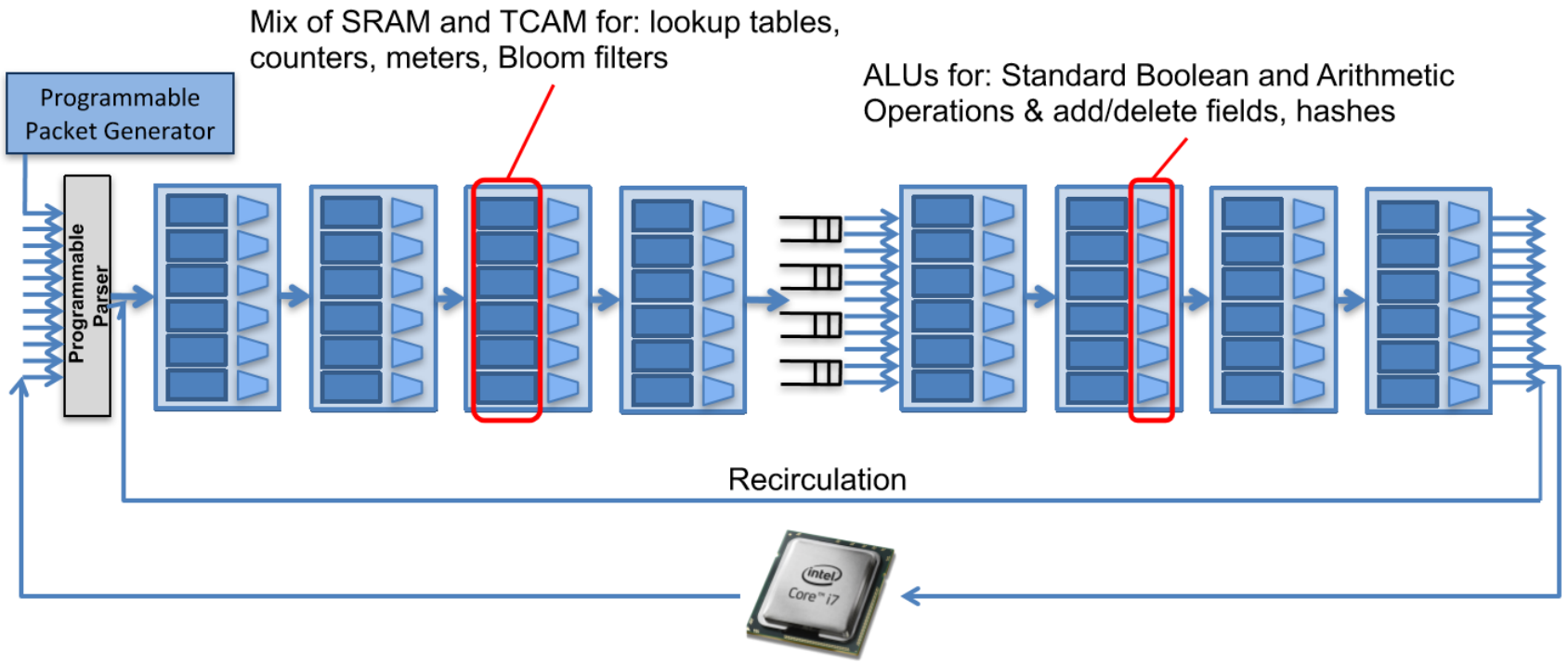
- 특성
  - (한정된 자원을 초과하지 않는 범위에서) Match Table, Action, Field를 새롭게 정의하거나 추가하는 것이 가능
  - 변경된 패킷은 정의된 queuing rule에 의해 원하는 전송 port의 queue로 위치시킬 수 있음
  - Intel의 FlexPipe, Cavium의 XPA, Barefoot의 Tofino 등에서 RMT 모델의 파이프라인 구조를 기반으로 하드웨어를 모델링하고 있으며, P4와 Domino 등의 데이터 평면 언어들도 해당 모델을 기반으로 프로그래밍 기법을 개발중

Reconfigurable Match Table (RMT) (2/2)



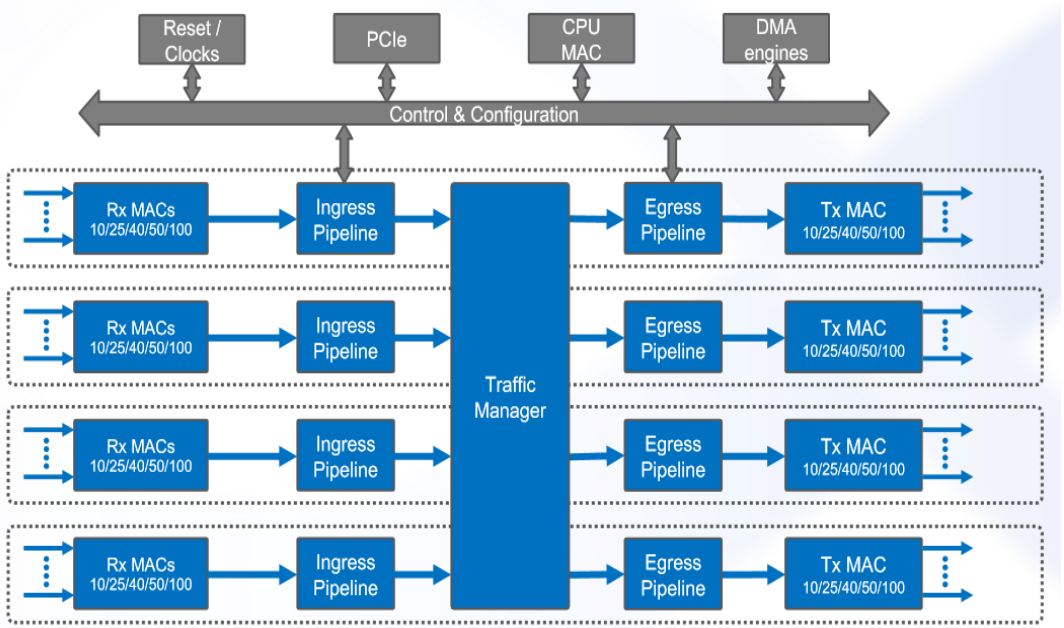
Barefoot: PISA

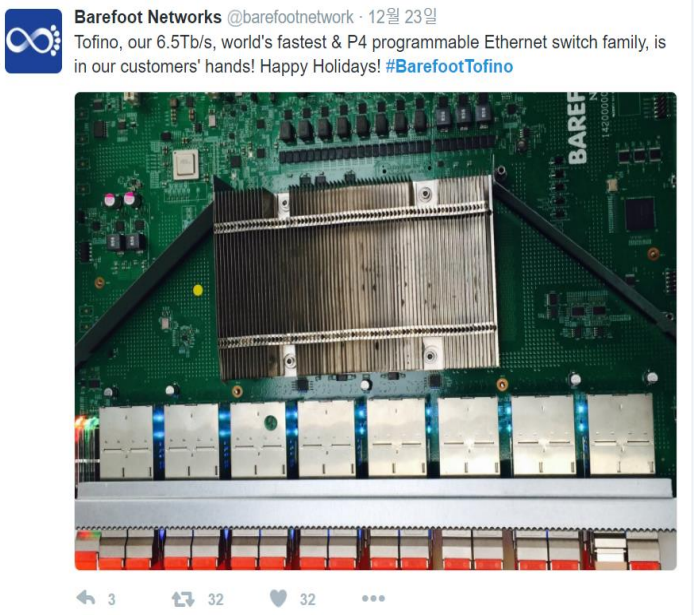
- PISA (Protocol-Independent Switch Architecture)
  - RMT 기반으로 P4를 지원



Barefoot: Tofino Chip

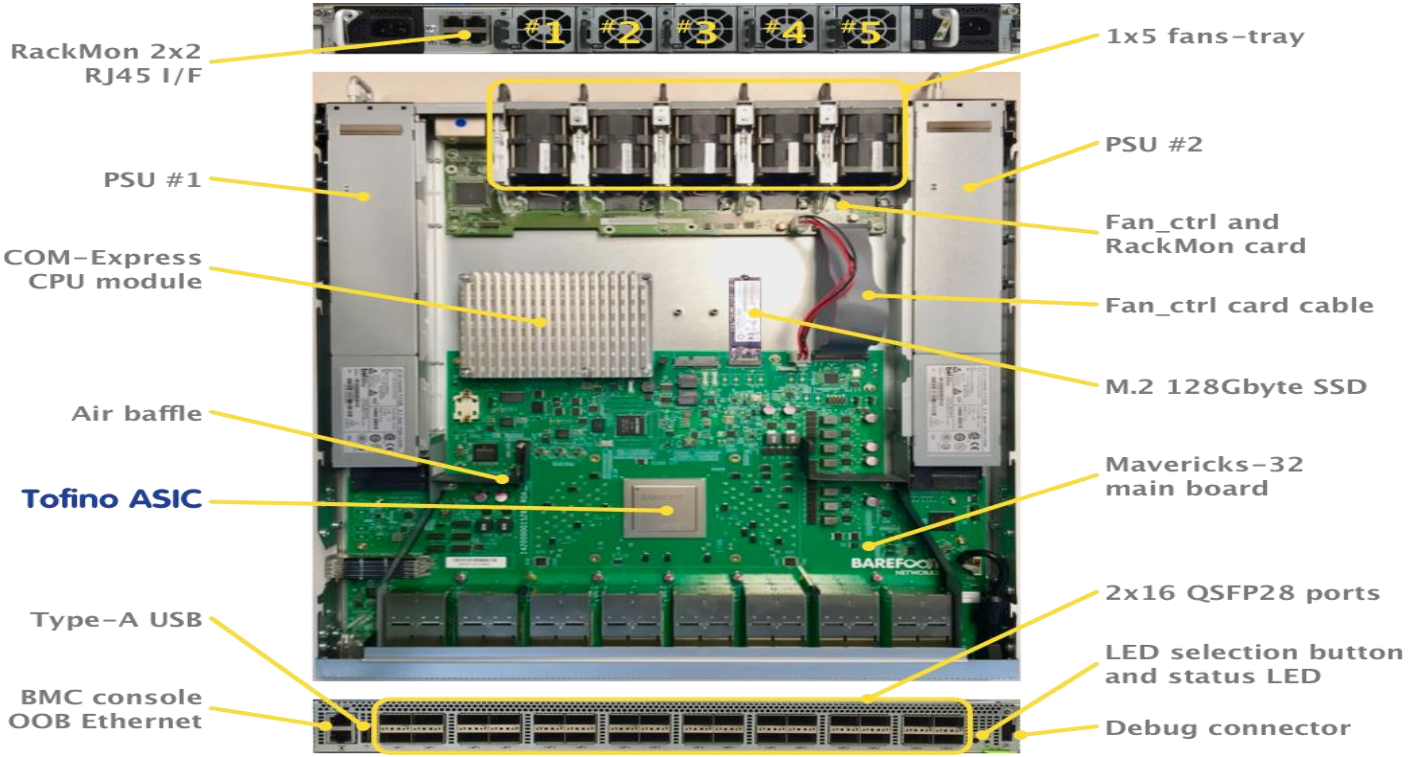
- Tofino
  - Barefoot 의 Capilano 컴파일러를 사용하여 PISA 모델에 최적화된 칩 구조 (up to 6.5 Tbps)





Barefoot/Edgecore: Switch

- OCP's Wedge100BF-32X (1RU, 3.2Tb/s)
  - https://barefootnetworks.com/blog/disaggregation-and-programmable-forwarding-planes/



Software for Programmable Data Plane

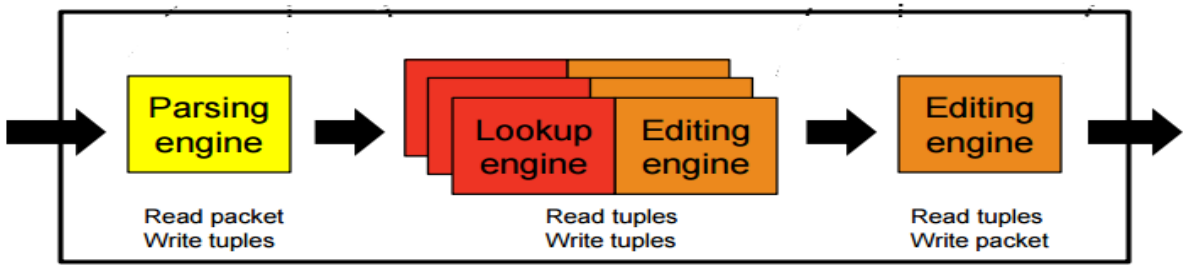
Project	업체	홈페이지	발표시기
POF	Huawei	http://www.poforwarding.org/	2013
PX/SDNet	Xilinx	https://www.xilinx.com/products/design-tools/software-zone/sdnet.html	2014
P4	P4 Consortium	http://p4.org/	2014
Domino	MIT	http://web.mit.edu/domino/	2016

Huawei POF

- POF (Protocol Oblivious Forwarding)
  - 프로토콜에 완전히 비의존적. 패킷 포맷이 알려지지 않은 경우에도 동작 가능
  - 한 개 이상의 {offset, length} tuple을 테이블 search key로 사용하고, 이렇게 search된 값에 따라 action을 취하는 구조. action은 사전에 어셈블리로 프로그래밍된 Flow Instruction Set (FIS)에 정의됨
  - 오픈 소스 형태의 POFController와 POFSwitch 프로토타입 스펙과 코드를 공개

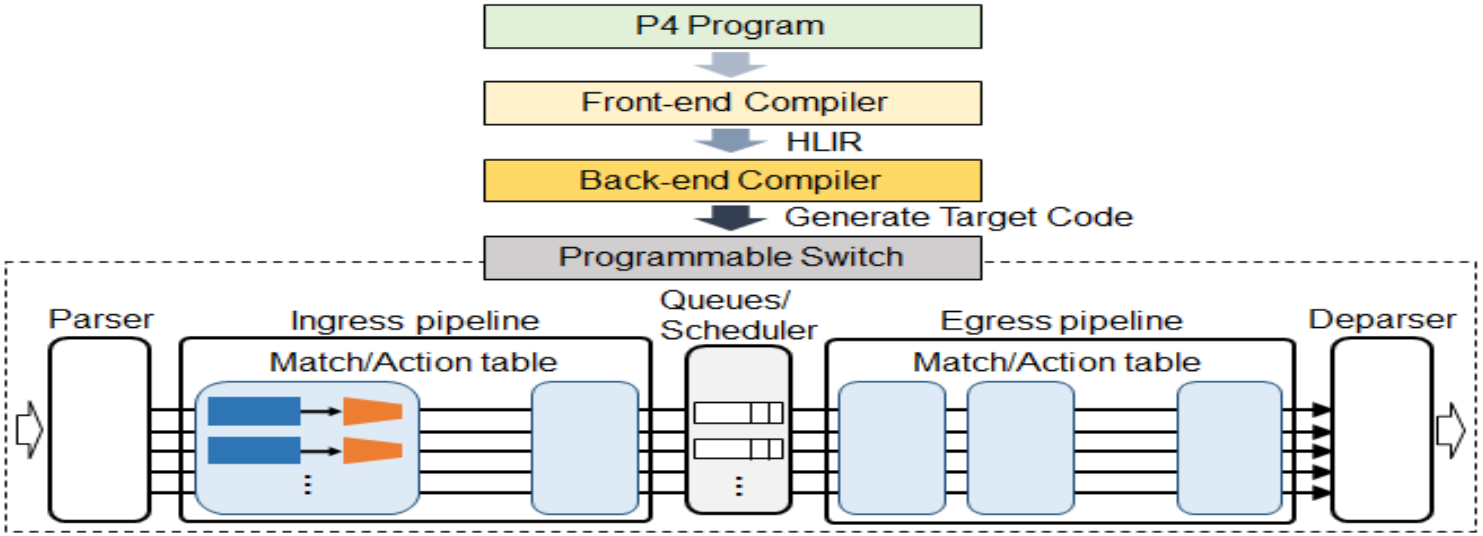
Xilinx PX

- PX
  - Parsing Engine, Lookup Engine, Editing Engine으로 데이터 평면의 기능을 구분하고 객체지향 프로그래밍 기능을 지원
  - 프로토콜에 독립적인 동작을 위해 FPGA/RMT/NPU/CPU 등의 타겟 장비에서 사용할 수 있는 Intermediate Presentation을 지원
  - P4와 협력 연구를 진행중



P4: Overview

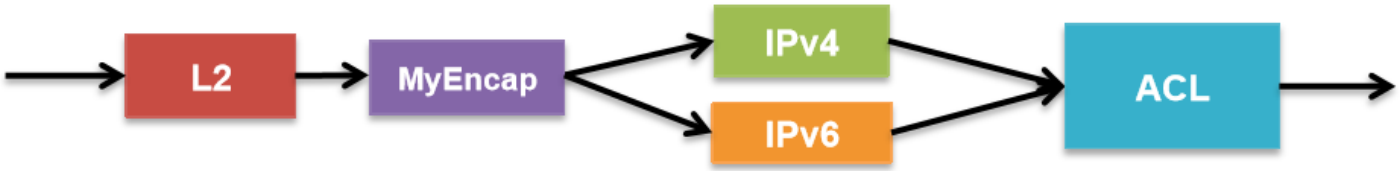
- P4 (Programming Protocol-Independent Packet Processors)
  - P4는 네트워크 장치의 하위 레벨 인터페이스에 대한 상위 수준의 추상화 기능을 제공하며 특정 프로토콜에 한정되지 않는 특성을 가짐

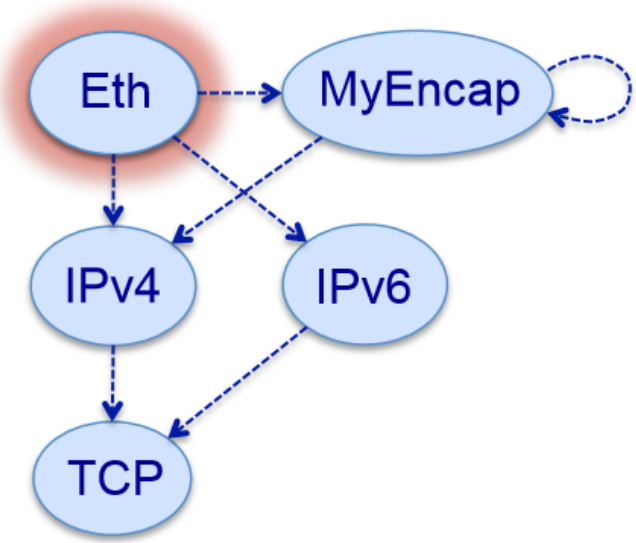


P4: Programming (1/3)

- P4 설계 과정
  - 1) 우선 설계한 프로그램에서 다루게 될 패킷의 헤더 필드를 정의
  - 2) 정의한 헤더 필드들을 어떤 순서와 규칙에 따라서 패킷에서 추출할지를 패킷 Parser를 통해 결정
  - 3) 추출한 헤더 필드의 각 부분에 대해서 어떤 규칙을 적용할지는 Match/action table 기반으로 정의

P4: Programming (2/3)

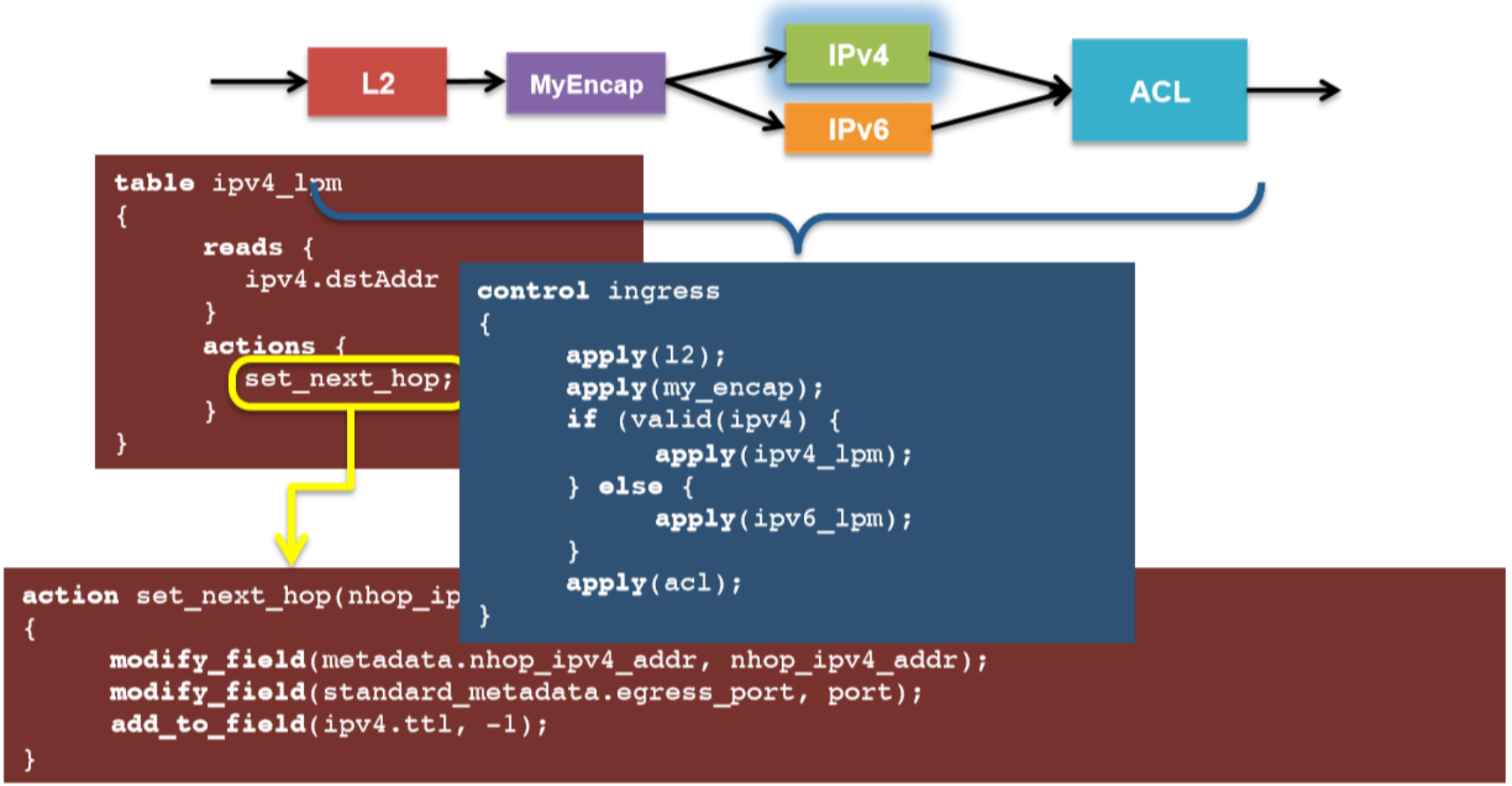





```
header_type ethernet_t {
      fields {
            dstAddr : 48;
            srcAddr : 48;
      parser parse_ethernet {
        extract(ethernet);
}       return select(latest.etherType) {
          0x8100 : parse_vlan;
header_type my_encap_t {
      fields {
            foo : 12;
            bar : 8;
            baz : 4;
            qux : 4;
            next_protocol : 4;
    }
}
```

[Changhoon Kim, “Programming The Network Data Plane in P4”]

P4: Programming (3/3)



[Changhoon Kim, “Programming The Network Data Plane in P4”]

P4: Compiling (1/2)

- Front-end 컴파일
  - 프로그램이 P4 규격에서 정의한 규칙에 맞게 설계되었는지를 검증
  - Front-end 컴파일은 P4에서 지원하는 P4-HLIR (High Level Intermediate Representation) 프로젝트를 통해 진행할 수 있으며, Front-end 컴파일의 결과물은 P4로 설계된 헤더, Parser, 테이블에 대한 정보가 Python 기반의 OrderedDict 형태로 제공됨

P4: Compiling (2/2)

- Back-end 컴파일
  - 대상이 되는 하드웨어에 대해서 Front-end 컴파일의 결과물을 매핑하는 과정
  - P4-HLIR 프로젝트에서는 Back-end 컴파일을 위한 테이블 간의 의존성 그래프 (dependency graph)에 대한 정보를 제공
  - 세부적인 Back-end 컴파일과정은 각 스테이지 별 파이프라인 구성, 메모리 할당 등과 같은 대상 하드웨어의 특성에 따라 달라지기 때문에 이에 대한 사전 지식을 필요로 함

P4_16

Term	Explanation
P4 Target	An embodiment of a specific hardware implementation
P4 Architecture	A specific set of P4-programmable components, externs, fixed components and their interfaces available to the P4 programmer
P4 Platform	P4 Architecture implemented on a given P4 Target



[Vladimir Gurevich, “P4_16 Introduction”]

Conclusion

- 멀티 서비스를 지원하는 프로그래머블 스위치 제어 기술 개발
  - 고려대학교/포항공과대학교
  - IITP 방송통신산업 기술개발사업
  - 2017년 4월-2020년 12월
  - 연구 목표: 하나의 물리적인 스위치 자원을 가상화를 통하여 다수의 서비스를 동시에 지원하는 스위치 원천 기술을 개발하고 멀티 서비스 네트워크 감시/제어/관리 기술을 연구함

References

- [Kim15] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band Network Telemetry via Programmable Dataplanes," in *Proc. ACM SOSR 2015 Demo*, June 2015.
- [Kat16] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable Load Balancing Using Programmable Data Planes," in *Proc. ACM SOSR 2016*, March 2016.
- [Li16] Y. Li, R. Miao, C. Kim, and M. Yu, "FlowRadar: A Better NetFlow for Data Centers," in *Proc. NSDI* 2016, March 2016.
- [Siv16-1] A. Sivaraman, A. Cheung, M. Budiu, C. Kim, M. Alizadeh, H. Balakrishnan, G. Varghese, N. McKeown, S. Licking, "Packet Transactions: High-Level Programming for Line-Rate Switches," in *Proc. SIGCOMM 2016*, August 2016.
- [Siv16-2] A. Sivaraman, S. Subramanian, M. Alizadeh, S. Chole, S. Chuang, A. Agrawal, H. Balakrishnan, T. Edsall, S. Katti, and N. McKeown, "Programmable Packet Scheduling at Line Rate," in *Proc. SIGCOMM 2016*, August 2016.