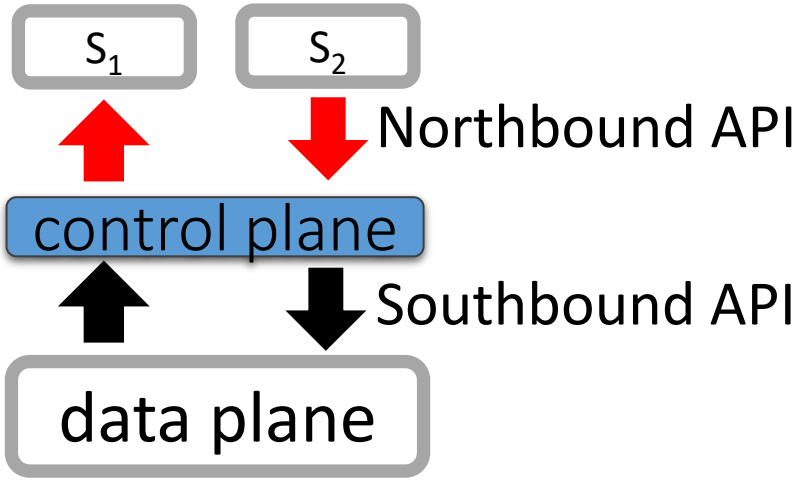
Adopting Software-Defined Networking: Challenges and Recent Developments

Kirill Kogan

IMDEA Networks Institute

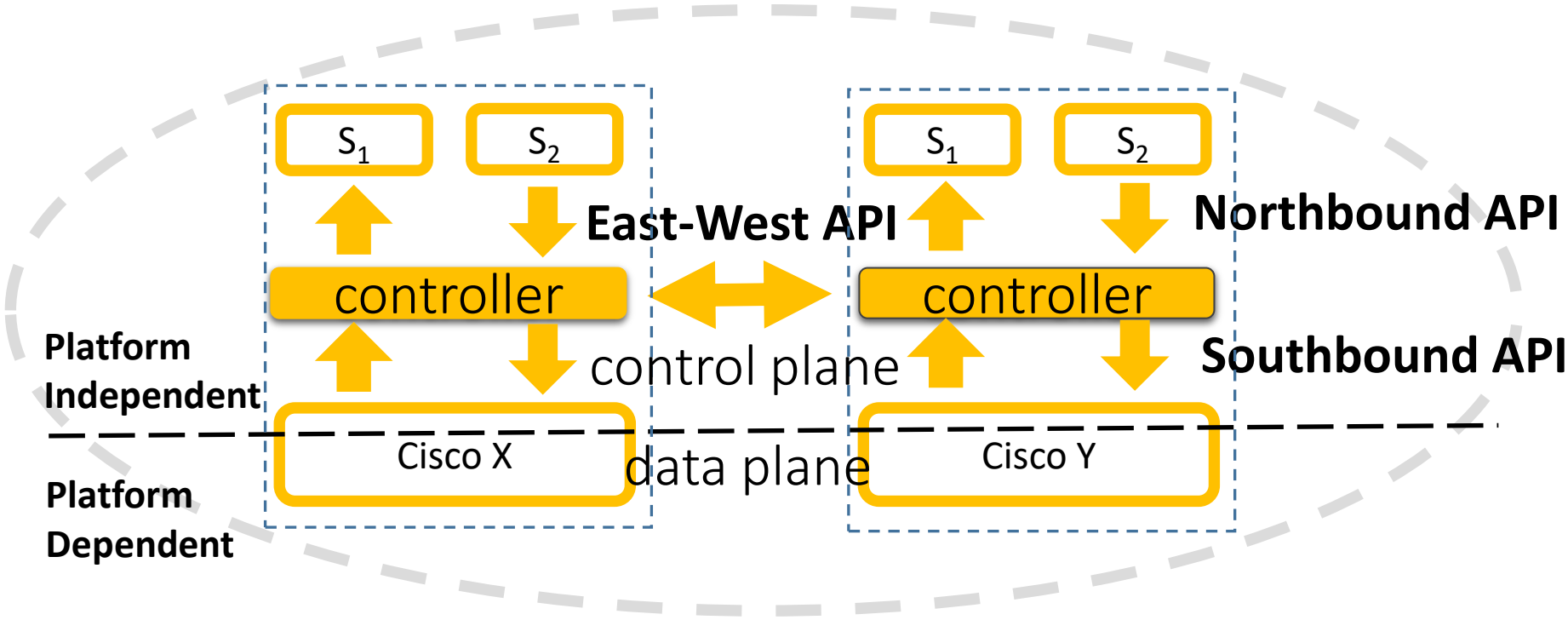
08.04.2016

Outline



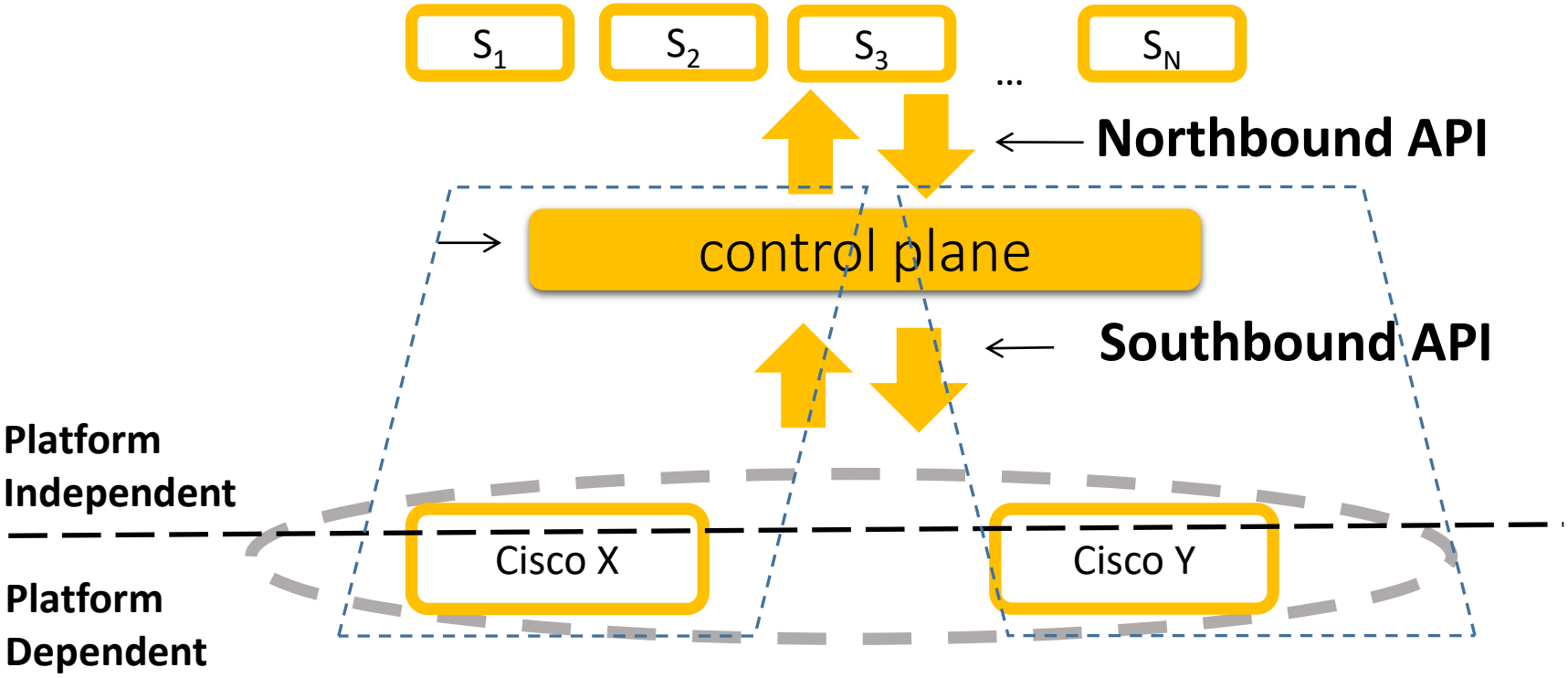
- Current state in traditional networks
- Personal insight on SDN developments
- Potentially interesting research directions

Evolution in traditional networks



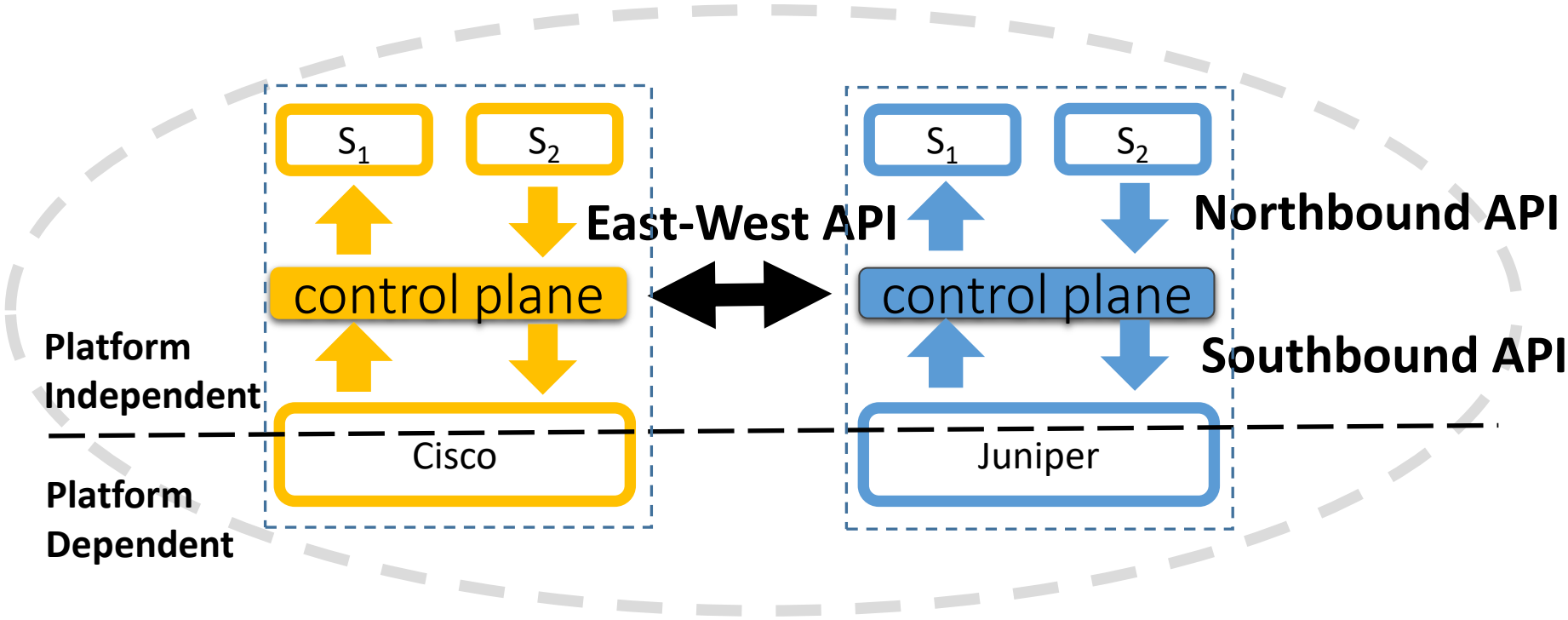
E-W API standardization **is NOT** required with per service resolution.

Traditional Network Management



Standardization of Southbound API is NOT required.

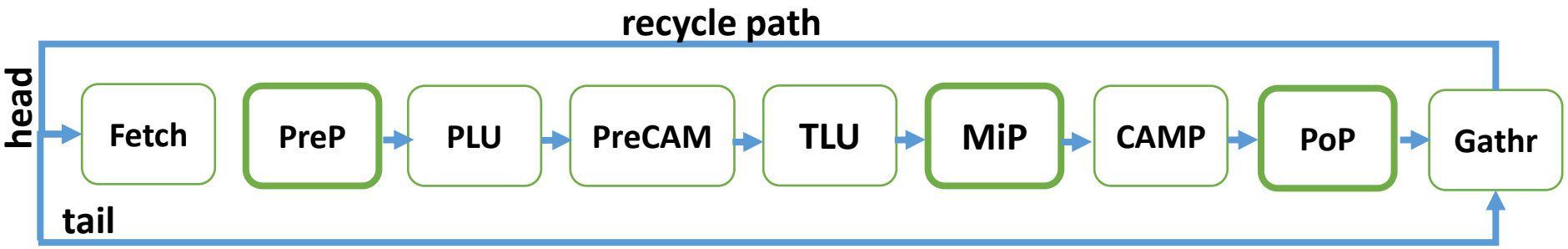
Traditional network management



E-W API standardization is required and currently it is done **with per service resolution**

This is a big problem in network management

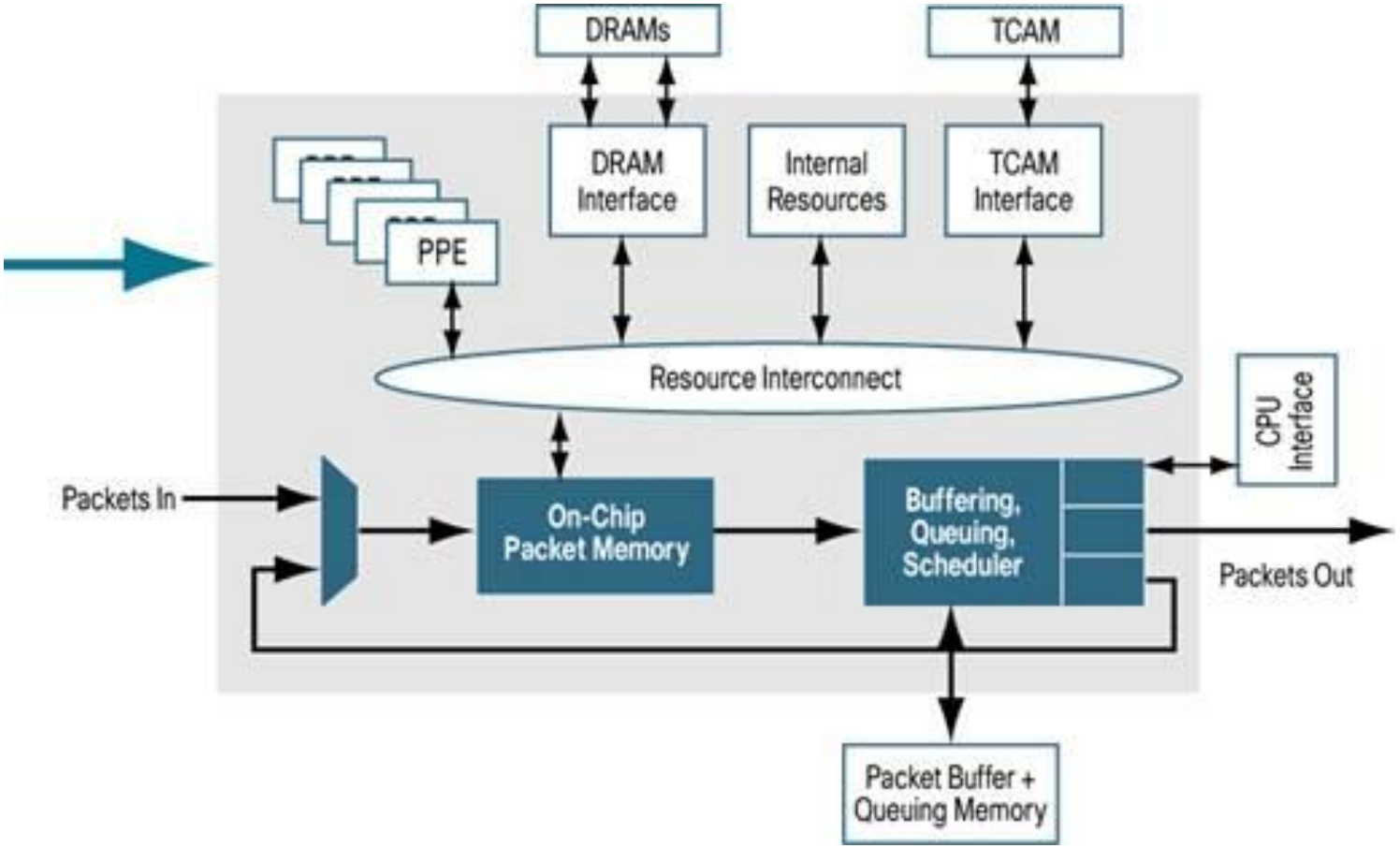
Pipeline Packet Processing Engine



Classifying L2 profiles Pre-caching fields	Fetching Services Preparation Future processing	FIB lkup	PBR Compr. lkup	L2 rewrite	Preparation Future processing QoS ACL NF	Post-proc

- **Fixed header size**
- **Hard to extend**
- **synchronous**
- **copying cost between pipeline stages**
- **difficult to port services that are implemented on general purpose CPUs**

Network processor (Cisco QFP)



- run-for-completion approach
- order of magnitude slower than specializecd ASICs

Problems in traditional networks

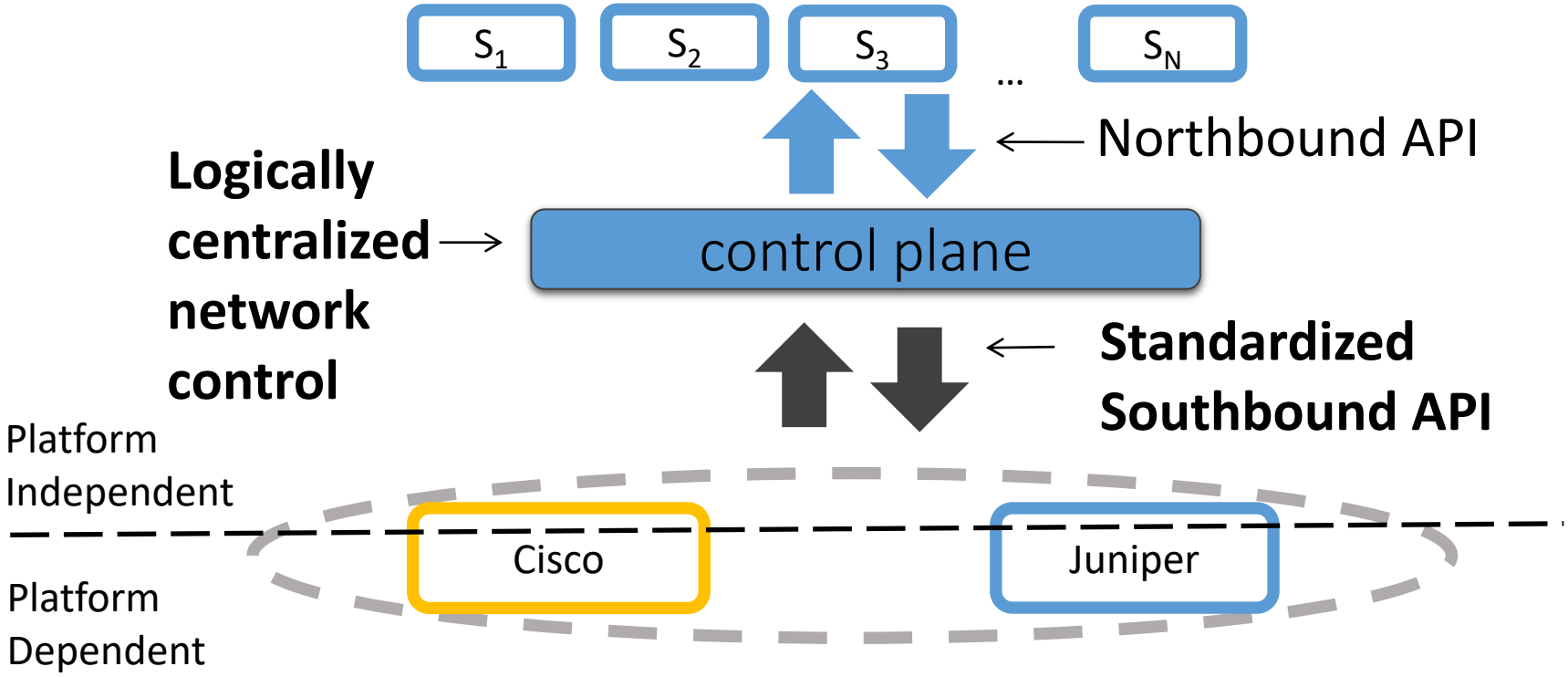
Control plane:

- Complex interoperability between CPs
- Complex distributed control for management

Data plane:

- Complex processing on data plane (network edge)
- Big manageable state on data plane

Software-defined networking view



Only a Southbound API should be standardized

OpenFlow as a language

01000 → A_1
01001 → A_2
00111 → A_3
11100 → A_4

Flow Classifiers:

- Ordered set of supported pkt fields

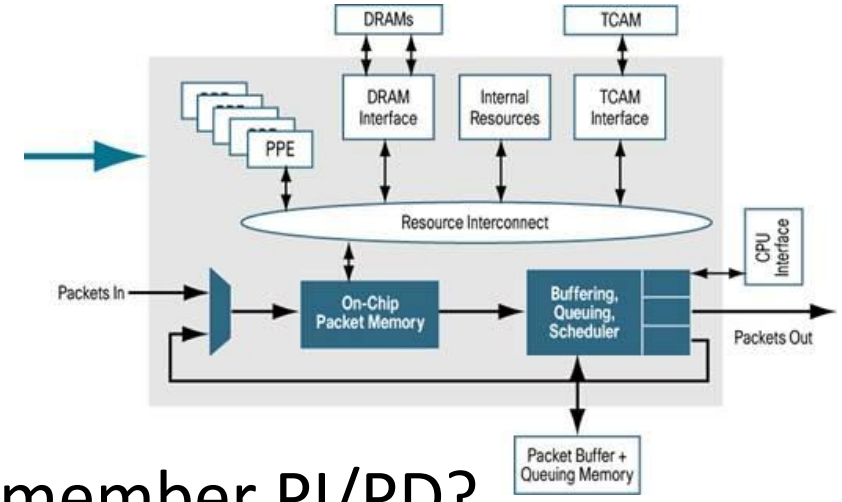
Actions:

- Header rewrite
- Set out port
- Etc.

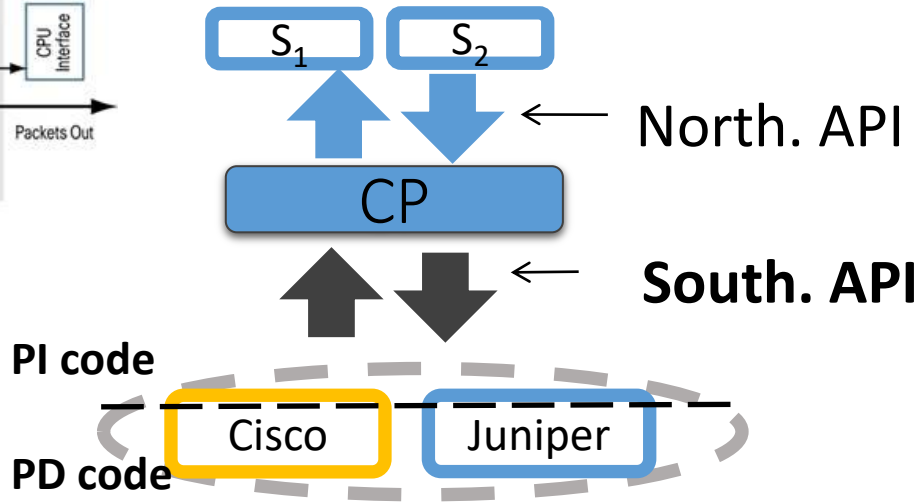
OpenFlow: Hierarchical tuple match with set actions.

Dummy switch

- Dummy switch as possible – all intelligence in CP



- Remember PI/PD?



To avoid PD code on controller – general CPU
or **delegate extra management power to controller**

General CPUs vs. NPs vs. ASICs

- 1s vs. 10s vs. 100s

- In DCs: racks of servers with CPUs

Why we cannot do the same for routers and switches?

- Major constraint in DCs: access bandwidth to storage

Do we have this constraint in networking?

Answer: on CRS-1 we had a bug – no write from DP☺

high operational cost: energy efficiency, space, etc.

- **CPUs are more flexible than specialized ASICs?**

Answer: No, if both implement the same S. API?

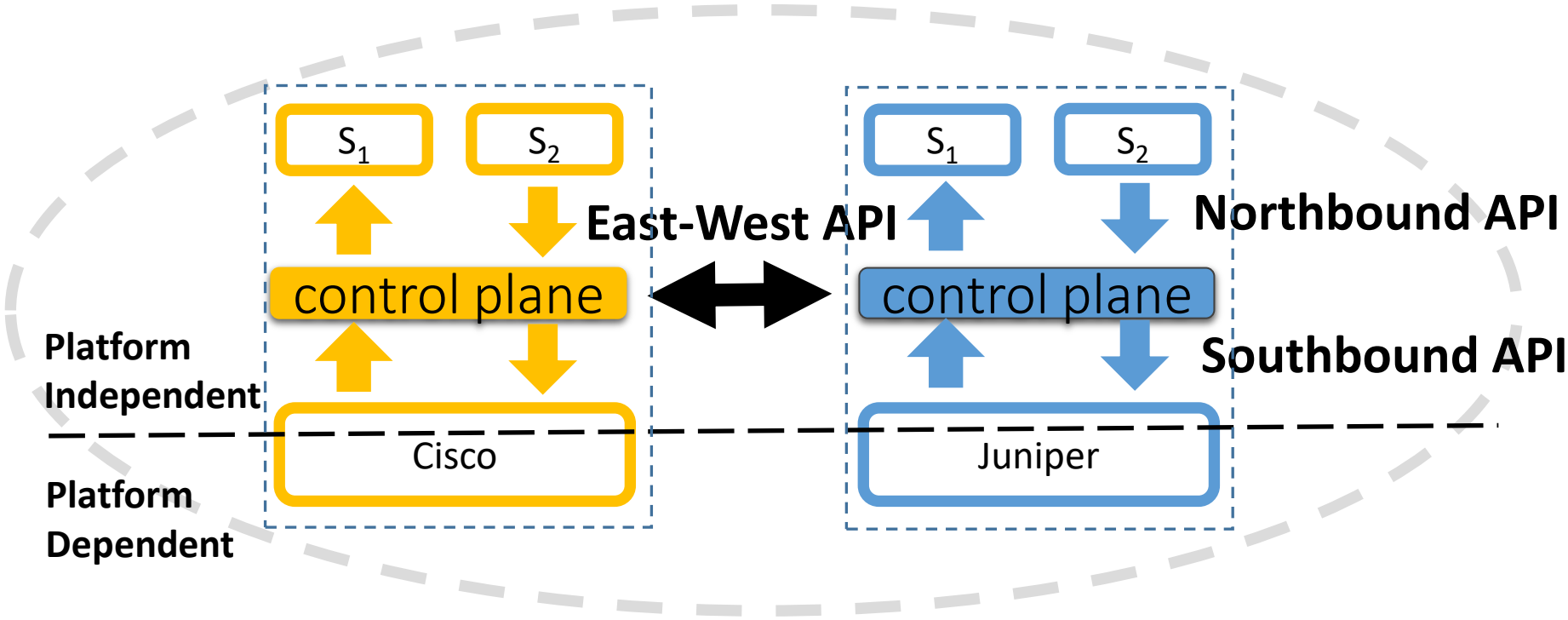
Question: Can we build general APIs for packet processing CPU + embedded GPU?

Problems in SDN CPs

Control plane:

- Complex interoperability between CPs
- Complex distributed control for management

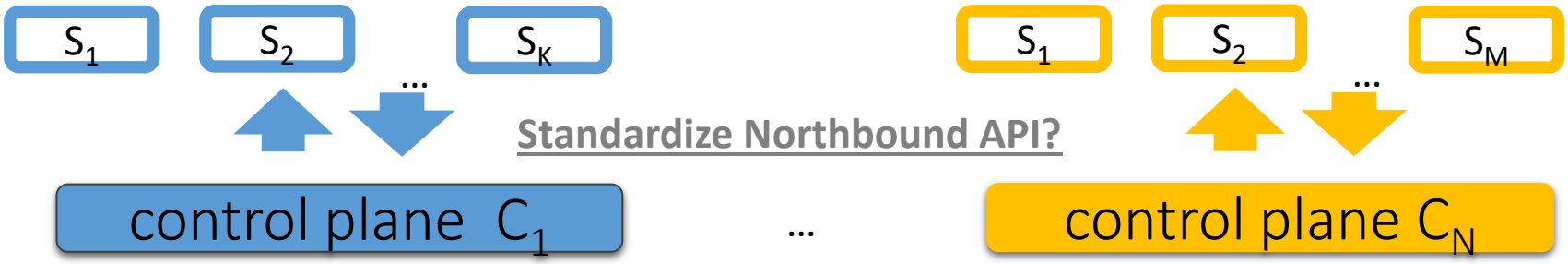
Complex interoperability between CPs



E-W API standardization is required and currently it is done **with per service resolution**

Question: Can we define E-W API NOT with per service resolution?
Do we really need to standardize S. API?

New opportunities



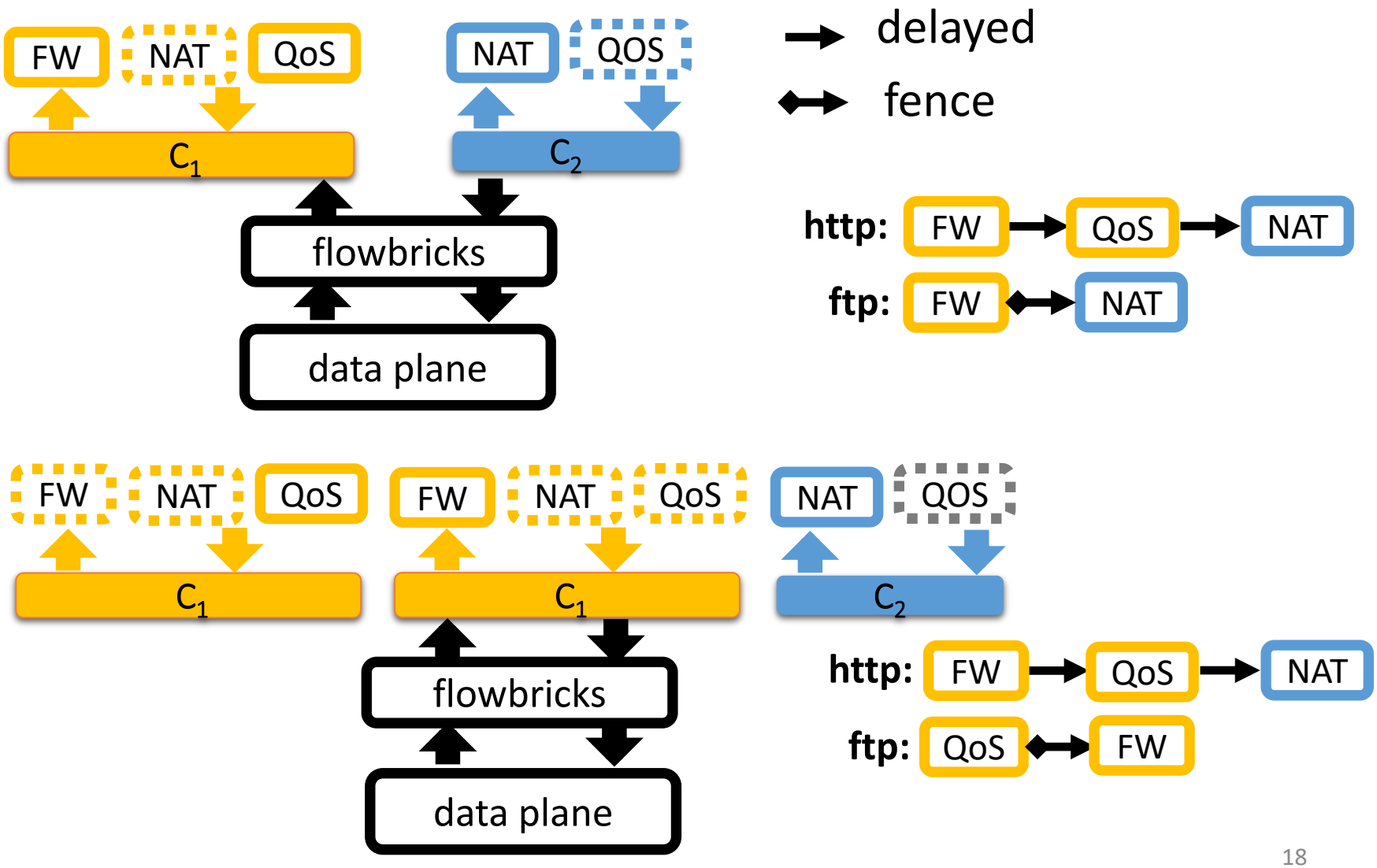
Problem: How to compose serially a subset of services?

Requirements:

1. Only the South-bound API is standardized
2. No code changes in implementing controllers
3. A composition is built at run time

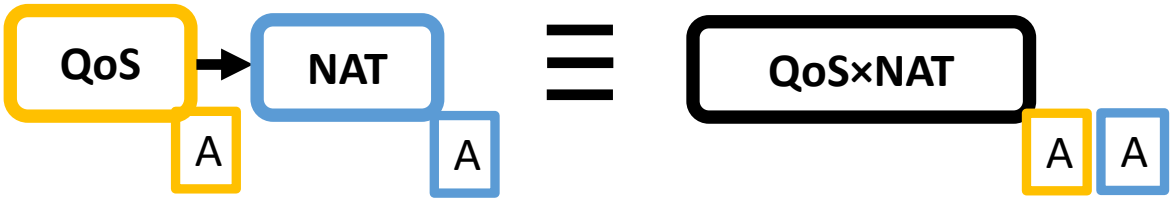
Composing Software Defined Networks *NSDI* '13 (Pyretic)

FlowBricks


FW
NAT
QoS
NAT
QOS
C1
C2
flowbricks
data plane
delayed
fence
http:
FW
QoS
NAT
ftp:
FW
NAT
FW
NAT
QoS
FW
NAT
QoS
NAT
QOS
C1
C1
C2
flowbricks
data plane
http:
FW
QoS
NAT
ftp:
QoS
FW

Flowbricks vs. Covisor

- Support of OF 1.1 vs. OF1.0



- Decreased line-rate vs. exponential memory.
- No support for dynamic field set.
- Potentially incorrect demultiplexing

Problem: E-W API is required if a state is exchanged between different services.

Composing Heterogeneous SDN Controllers with Flowbricks *ICNP* '14
CoVisor: A Compositional Hypervisor for Software-Defined Networ*ks NSDI* '15

Solutions in SDN data planes

Data plane:

- Complex processing on data plane (network edge)
- Big manageable state on data plane

Data plane problems in SDN

SW-based: $N = 4$ rules $K = 2$ fields

$$
\begin{aligned}
R_1 &= (\ 100*,\ 001*\ ) \\
R_2 &= (\ 1010,\ 0001\ ) \\
R_3 &= (\ 000*,\ ****\ ) \\
R_4 &= (\ 001*,\ ****\ )
\end{aligned}
$$

Memory	Lookup time
$O(N)$	$O(log^{k-1}N)$
$O(N^k)$	$O(logN)$

TCAM-based: $N = 3$ rules $K = 3$

$$
\begin{aligned}
R_1 &= ([1,3],[4,31],[1,28]) \\
R_2 &= ([4,4],[2,30],[4,27]) \\
R_3 &= ([7,9],[5,21],[3,18])
\end{aligned}
$$

Binary Encoding	SRGE
42+28+50=120	24+8+32=64

SAX-PAC (Scalable And eXpressive PAcket Classification *SIGCOMM* '14

Order-independence

If the rules of a classifier do not ``intersect'', their order is not important.

$$R_1 = ([1,3],[4,31],[1,28])$$
$$R_2 = ([4,4],[2,30],[4,27])$$
$$R_3 = ([7,9],[5,21],[3,18])$$

- Example: prefixes of the same length
- Implicit creation of order-dependence for service policies

	cisco1	cisco2	cisco3	fw	ipc	acl
Order-independent rules	120	249	329	39962	48294	49779
Total	148	269	364	45723	49840	49870
Order-independent %	81	93	90	87	97	99

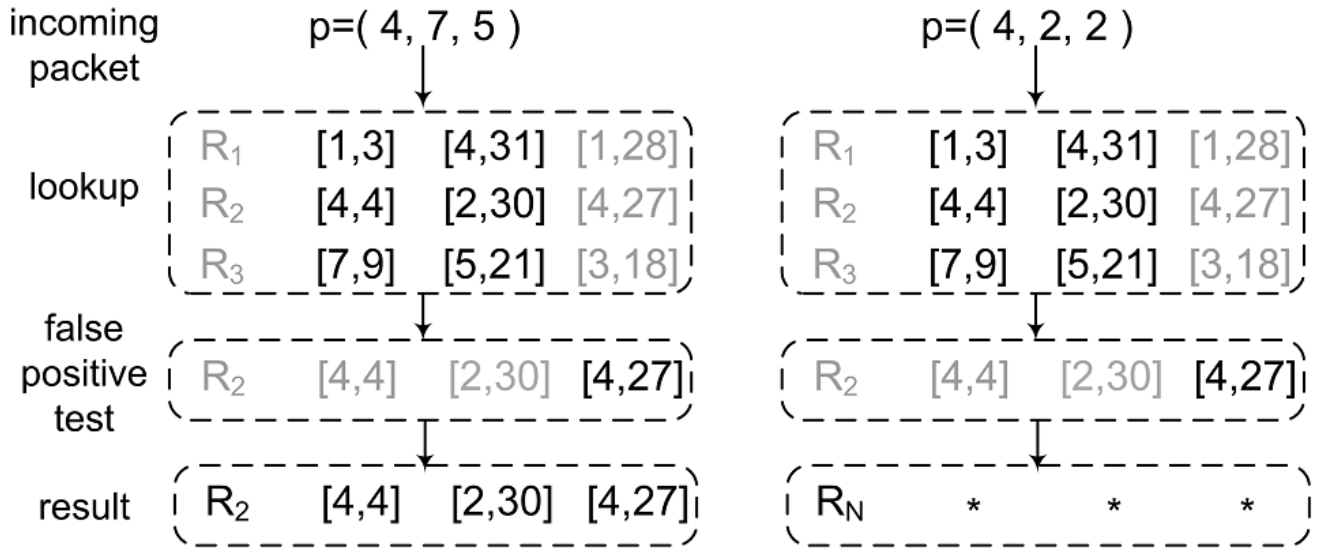
Exploiting order-independence

- Adding new fields keep order-independence
- At most one rule is matched and it can be false-positive
- We can reduce space by skipping new fields

$$
\begin{aligned}
R_1 &= ([1,3],[4,31]) \\
R_2 &= ([4,4],[2,30]) \\
R_3 &= ([7,9],[5,21])
\end{aligned}
$$

$$
\begin{aligned}
R_1^{+1} &= ([1,3],[4,31],[1,28]) \\
R_2^{+1} &= ([4,4],[2,30],[4,27]) \\
R_3^{+1} &= ([7,9],[5,21],[3,18])
\end{aligned}
$$

#Fields	Bin Encoding	Gray Encoding
2	6+7+10=23	6+4+8=18
3	42+28+50=120	24+8+32=64



Solutions in SDN Networks

Control plane:

- Complex interoperability between CPs
- Complex distributed control for management

Data plane:

- Complex processing on data plane (network edge)
- Big manageable state on data plane

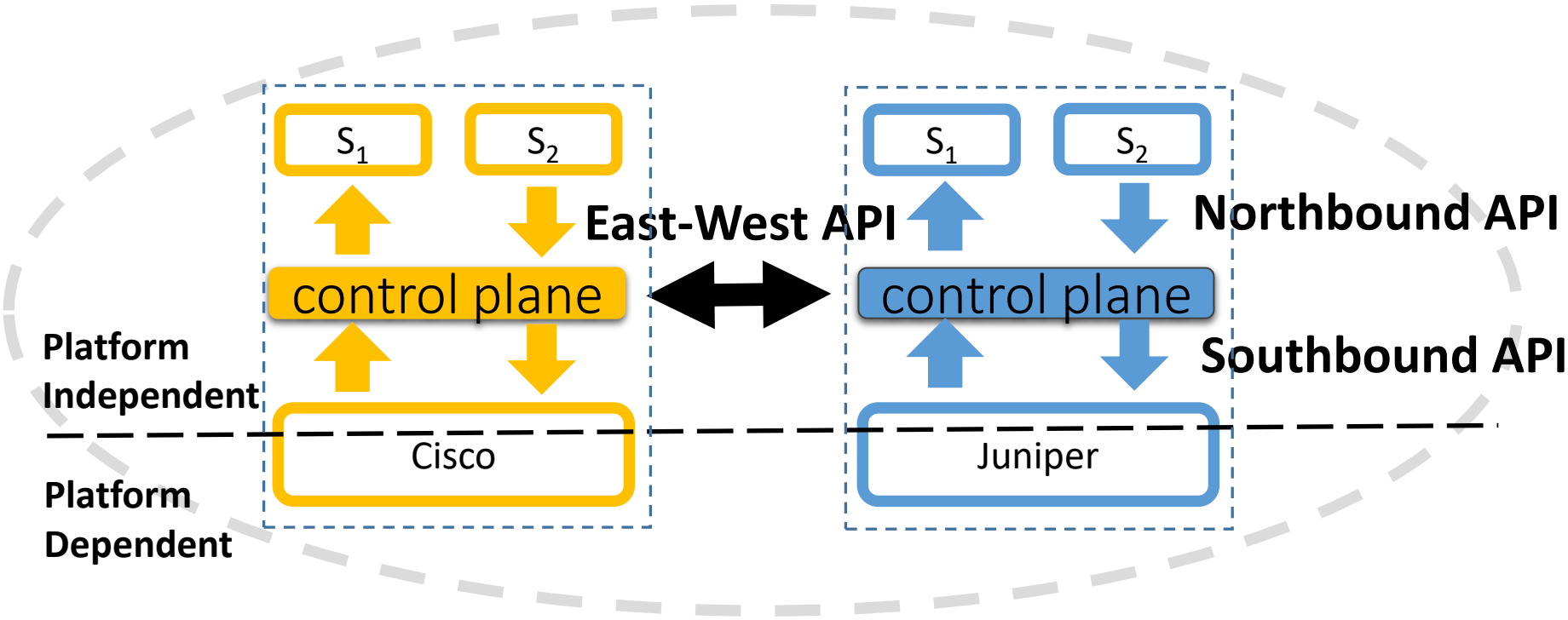
Marketing hype, just incremental step, or revolution?

What is missing/interesting in networking?

- General APIs that exploit CPU+embedded GPU, FPGA
- Specification of E-W API but not with per service resolution
- Efficient state representations on DP
- Simple and expressive buffer management
- Virtualization of computing is relatively simple,

  how to simplify network virtualization?

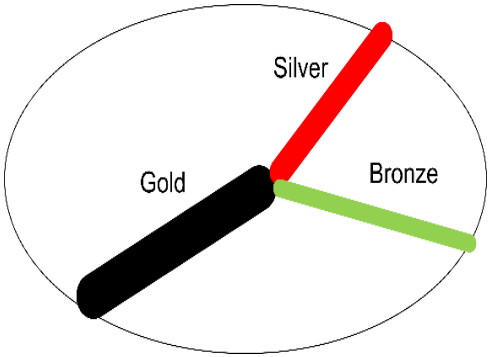
Standardization of E-W API



E-W API standardization is required and currently it is done **with per service resolution**

Sharing classifiers

1. Flow: forwarding
   (path+traffic aggregation)
2. Policies (represent economic models)



- Both can be represented as hierarchical packet match with set actions
- Can have different prioritization schemes, update requirements, etc.

Suggestion: decouple policies from flows

Strategies for Mitigating TCAM Space Bottlenecks *HOTI* '14

Virtual pipeline architecture

Problem: As SLA complexity increases (required processing)

Solutions:

- For desired line-rate, faster NPs or longer pipelines are required, resulting in a higher cost per network element.
- A controller can split processing along the path: static and dynamic.

Requirements: structured data

Question: to identify online services that require data processing during transportation.

Pros: removing boundaries between communication and computing networks - basis for new type of services.

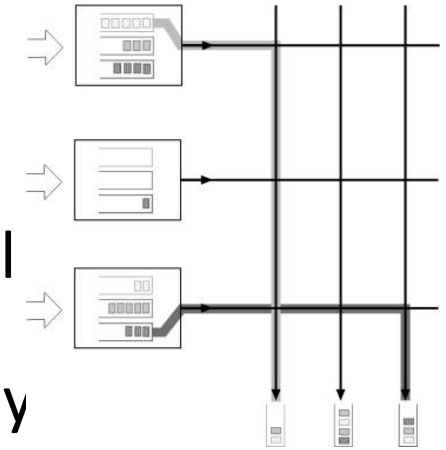
Towards an active network architecture *CCR* '96

Palette: Distributing tables in software-defined networks *INFOCOM* '13

Optimizing the "one big switch" abstraction in software-defined networks *CONEXT* '13

Software-defined buffer management

- *Buffering architectures: how inputs are connected with outputs.*
- Traditional networks allow only a predefined set of policies.



Summary:

1. Objectives beyond *fairness* and additional traffic properties lead to new challenges.
2. Current SDN and traditional networks only deal with efficient representation of packet classifiers.
3. New policies require complex changes in both CP and DP.

Achieving High Utilization with Software-Driven WAN *SIGCOMM* '13
B4: Experience with a Globally-Deployed Software Defined WAN *SIGCOMM* '13
pFabric: minimal near-optimal datacenter transport *SIGCOMM* '14
pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric *CONEXT* '15

Requirements for Software-Defined buffer managment

- **Expressivity**: should be expressive enough
- **Simplicity:** policies should be expressible concisely with a limited set of basic primitives
- **Performance:** implementations of policies should be efficient
- **Dynamism:** specification and provision of new policies should be possible at run-time without any code changes

From Single Packets to Streams

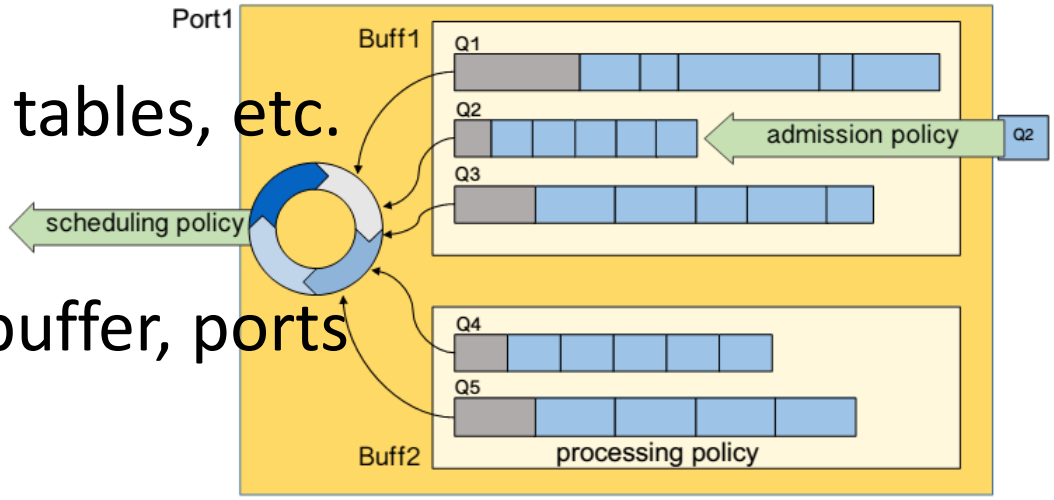
$01000 \rightarrow A_1$
$01001 \rightarrow A_2$
$00111 \rightarrow A_3$
$11100 \rightarrow A_4$

Type of services:

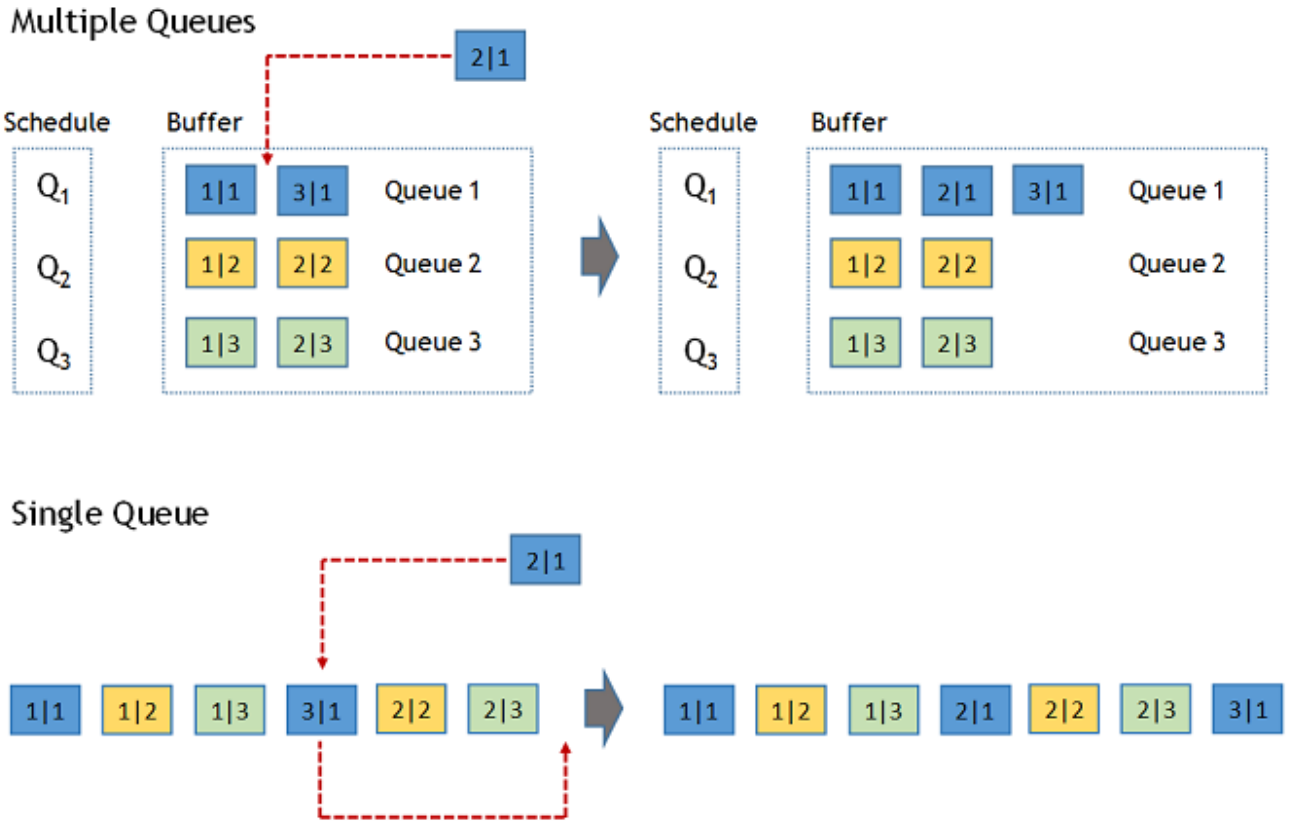
1. change properties of single packets;
2. change inter-packet properties (rate-limiting, shaping)

OF: packets, fields, flows, tables, etc.

BASEL: packets, queues, buffer, ports



What should be flexible?

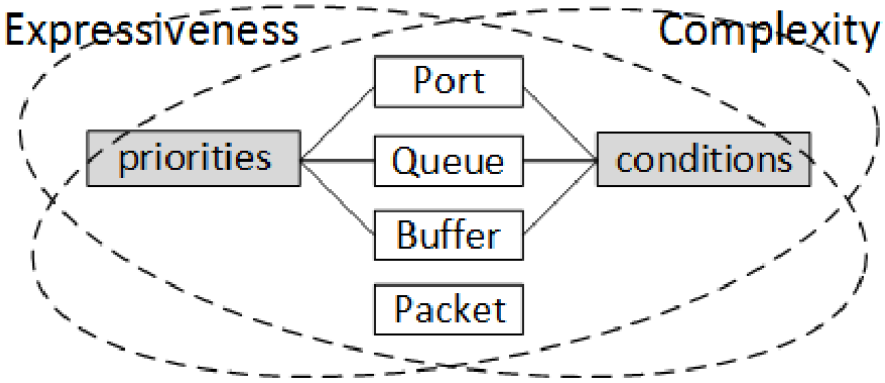


Observation: For any deterministic policy in MQ, there is a policy in SQ that can transmit the same subset of pkts.

For randomized MQ policies we consider analogous behaviors.

Expressiveness vs. simplicity

- buffer management policies are generally concerned with boundary conditions
- 2 priorities per Queue, 1 priority for Buffer and port.



Towards programmable packet scheduling *HotNets* 2015
BASEL (Buffer mAnagement SpEcification Language) *ANCS* 2016

Queue primitive

```
Queue {
 // user-specified at declaration
 size            // size in bytes          [r, cons]
 buffer          // buffer where allocat. [r, cons]

  // primitive properties
 currSize        // current size            [r, dyn]
 getHOL()        // head-of-line pkt    [Packet fun]

 // admission -  user-specified at declaration
 congestion()    // drop(P) condition    [drop cond]
 admPrio(p1,p2) // pushOut prio compar. [bool fun]
 postAdmAct()    //      [{mark,notify,modify} comp]
 weightAdm       // priority for adm.      [rw, dyn]

 // processing -  user-specified at declaration
 procPrio(p1,p2)// process. compar.  [packet comp]

 // scheduling - user-specified at declaration
 weightSched     // prio. for scheduling  [rw, dyn]
}
```

Buffer primitive

```
Buffer {
 // primitive properties
 currSize          // current size              [r, dyn]
 getBestQueue()    // on weightAdm            [Queue fun]
 getCurrQueue()    // admitted one            [Queue fun]

 // user-specified at declaration
 size              // size                      [r, cons]

 // admission -  user-specified at declaration
 congestion()      // {drop(P)}               [drop cond]
 queuePrio(q1,q2)// compare q-s               [bool fun]
 postAdmAct()      //     [{mark,notify,modify} cond]
}
```

Port and packet primitives

```
Port {
 // primitive properties
 getBestQueue()     // on weightSched    [Queue fun]
 getCurrQueue()     // scheduled one     [Queue fun]

 // scheduling user-specified at declaration
 schedPrio(q1,q2)   //  compare q-s        [bool fun]
 postSchedAct()     //  [{mark,notify,modify} cond]
}
```

```
Packet {
 size               // size in bytes          [r, cons]
 value              // virtual value          [r, cons]
 processing         // # of cycles            [r,  dyn]
 arrival            // arrival time           [r, cons]
 slack              // offset in time         [r, cons]
 queue              // target queue id        [r, cons]
 flow               // flow id                [r, cons]
}
```

Software-defined transports

pFabric with FIFO and push-out

p.value carries remaining flow size

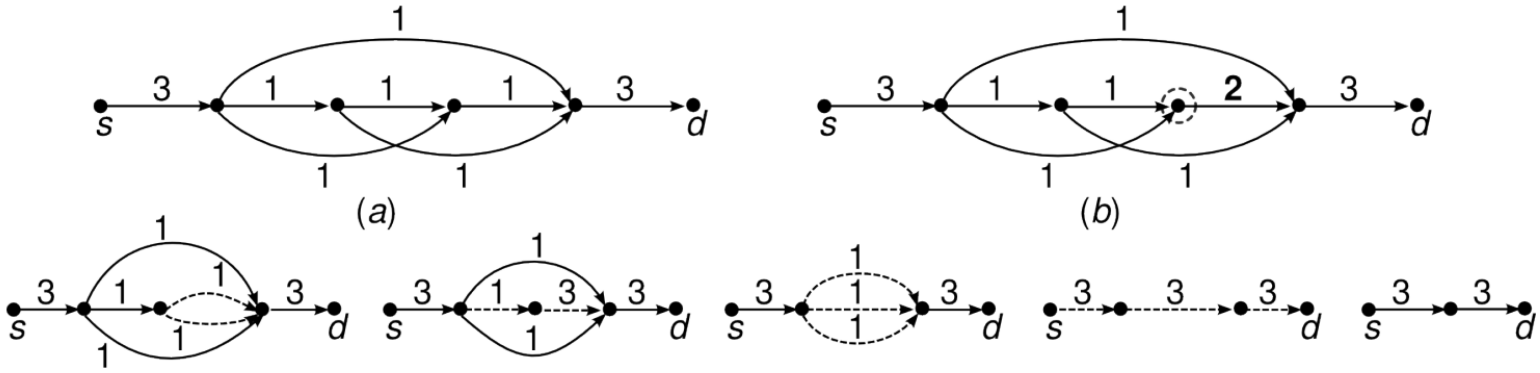
```
// Specification of the buffering architecture
q1 = Queue(B);
out = Port(q1);

// Admission control
q1.admPrio(p1,p2) = (p1.value > p2.value);
q1.congestion = defCongestion(q1);

// Processing policy: fifo()
q1.admPrio(p1,p2 )= fifo(p1,p2);
```

Integration with Pyretic

```
q1=Queue(B);...qk=Queue( B );
out=Port(q1,..,qk);
// pyretic part
split = (match(dstip=IPAddr('10.0.0.1')) >>
        modify(queue_id=q1) >> fwd( out ))
```

Network virtualization

- Currently properties of a switch are replicated to the whole network (e.g., pFabric, pHost)
- Network representations by a virtual switch do not preserve original ``routing'' capabilities.
- Tradeoffs between simplicity and exploiting resources of network infra are unclear.



CONGA: distributed congestion-aware load balancing for datacenters *SIGCOMM* '14

Thank You.

Questions?

kirill.kogan@imdea.org