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

Platform Independent

Platform Dependent

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
Fetching Services
Pre-caching fields
Preparation Future processing

• 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 specialized 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

Logically centralized network control

Northbound API

Standardized Southbound API

Platform Independent

Platform Dependent

Only a Southbound API should be standardized
OpenFlow as a language

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

C1

FW
NAT
QoS

flowbricks

data plane

C2

NAT
QOS

FW
NAT
QoS

flowbricks

data plane

→ delayed
← fence

http:
FW → QoS → NAT

ftp:
FW → NAT

C1

FW
NAT
QoS

C1

FW
NAT
QoS

C2

NAT
QOS

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 Networks *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

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

<table>
<thead>
<tr>
<th>Memory</th>
<th>Lookup time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(N)$</td>
<td>$O(\log^{k-1}N)$</td>
</tr>
<tr>
<td>$O(N^k)$</td>
<td>$O(\log N)$</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Binary Encoding</th>
<th>SRGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>42+28+50=120</td>
<td>24+8+32=64</td>
</tr>
</tbody>
</table>

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.

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

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

<table>
<thead>
<tr>
<th></th>
<th>cisco1</th>
<th>cisco2</th>
<th>cisco3</th>
<th>fw</th>
<th>ipc</th>
<th>acl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order-independent rules</td>
<td>120</td>
<td>249</td>
<td>329</td>
<td>39962</td>
<td>48294</td>
<td>49779</td>
</tr>
<tr>
<td>Total</td>
<td>148</td>
<td>269</td>
<td>364</td>
<td>45723</td>
<td>49840</td>
<td>49870</td>
</tr>
<tr>
<td>Order-independent %</td>
<td>81</td>
<td>93</td>
<td>90</td>
<td>87</td>
<td>97</td>
<td>99</td>
</tr>
</tbody>
</table>
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

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

<table>
<thead>
<tr>
<th>#Fields</th>
<th>Bin Encoding</th>
<th>Gray Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6+7+10=23</td>
<td>6+4+8=18</td>
</tr>
<tr>
<td>3</td>
<td>42+28+50=120</td>
<td>24+8+32=64</td>
</tr>
</tbody>
</table>

\[\text{p=}(4, 7, 5)\]

\[\text{p=} (4, 2, 2)\]
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?
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 management

- **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

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

```plaintext
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

```c
// 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

\[
q1 = \textbf{Queue}(B); \ldots qk = \textbf{Queue}(B);
\]
\[
\text{out} = \textbf{Port}(q1, \ldots, qk);
\]

// pyretic part
\[
\text{split} = (\text{match}(\text{dstip}=\text{IPAddr}('10.0.0.1'))) \gg
\]
\[
\text{modify}(\text{queue\_id}=q1) \gg \text{fwd(out)}
\]
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