Forwarding and Routing with Packet Subscriptions

Theo Jepsen, Ali Fattaholmanan, Masoud Moshref, Antonio Carzaniga, Nate Foster, Robert Soulé

Università della Svizzera italiana,
Cornell University, Barefoot Networks
Status quo: location-based addressing

192.168.0.3
How do applications communicate?

- **Microservices:**
  - service ID

- **Load balancers:**
  - lowest load

- **Pub/sub:**
  - topic

![Kubernetes](image1)

![Load Balancer](image2)

![Kafka](image3)
Forwarding with software middleware

Message Publisher

#art
likes=70

#art
#sports

#art and
likes>50
Forwarding with software middleware

Message Publisher

#art
likes=70

#art
likes>50

#art and
likes>50

Wastes bandwidth

Slow!
Forwarding with software middleware

Why not use the switches!?
We now have fast, programmable networks. We can use them for more expressive routing.
Packet Subscriptions

- Identify packet and indicate action
  
  \[ \text{topic} = \text{art}: \text{fwd}(1) \]

- Relational and logical operators
  
  \[ \text{topic} = \text{art} \land \text{likes} > 70: \text{fwd}(1) \]

- Multicast
  
  \[ \text{likes} > 70: \text{fwd}(1, 2, 3) \]
Packet Subscriptions challenges

- How to evaluate rules?
- How to route with rules?
Compilation overview

Application specification

Generate table pipeline

Programmable switch

Generate table entries

Subscriptions
Compiling subscription rules

likes > 100: fwd(1)
likes > 100 ∧ topic = art: fwd(2)
likes < 60 ∧ topic = sports: fwd(3)

Binary Decision Diagram (BDD)
# Compiling Subscription Rules

BDD

P4 table entries

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>likes</td>
<td></td>
</tr>
<tr>
<td>&lt; 60</td>
<td>state←1</td>
</tr>
<tr>
<td>&gt; 100</td>
<td>state←2</td>
</tr>
<tr>
<td>*</td>
<td>state←6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>topic</td>
</tr>
<tr>
<td>1</td>
<td>sports</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>2</td>
<td>art</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>fwd(3)</td>
</tr>
<tr>
<td>4</td>
<td>fwd(1,2)</td>
</tr>
<tr>
<td>5</td>
<td>fwd(1)</td>
</tr>
<tr>
<td>6</td>
<td>drop()</td>
</tr>
</tbody>
</table>
Packet Subscriptions challenges

How to evaluate rules?

How to route with rules?
Routing with Packet Subscriptions

Where to place rule?

```
topic=art
```
Routing with Packet Subscriptions

Where to place rule?

- topic=art: fwd(2)
- topic=art: fwd(3)
- topic=art: fwd(2)
- topic=art
Routing with Packet Subscriptions

Traffic reduction scheme

All switches store all rules: uses lots of memory!
Routing with Packet Subscriptions

Memory reduction scheme

- src=down: fwd(up)
- topic=art: fwd(2)
- topic=art: fwd(3)
- src=down: fwd(up)
- topic=art: fwd(2)

Uses less memory
Evaluation
Are Packet Subscriptions useful to applications?

Market Feed Filtering  Video Streaming  In-Band Network Telemetry
Is forwarding efficient, in terms of performance?

Server

Nasdaq
Feed Publisher

Subscriber:
stock = GOOGL

Switch filters messages

Baseline: host filters messages

Switch
In-network filtering reduces tail latency
Is routing efficient, in terms of FIB memory?

- Core
- Agg
- ToR

Measure memory usage

- price>70
- price=20
- price<50
Compiler uses memory efficiently

Traffic Reduction Scheme

![Graph showing the relationship between the number of filters and the number of table entries for Core, Agg, and ToR.]
Compiler uses memory efficiently

Memory Reduction Scheme

# of table entries

<table>
<thead>
<tr>
<th>Core</th>
<th>Agg</th>
<th>ToR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5M</td>
<td>5M</td>
<td>5M</td>
</tr>
<tr>
<td>10M</td>
<td>10M</td>
<td>10M</td>
</tr>
<tr>
<td>15M</td>
<td>15M</td>
<td>15M</td>
</tr>
</tbody>
</table>

# of filters

400  800  1200  1600
In conclusion, Packet Subscriptions...

- Provide the network abstraction used by applications
- Improve performance by using network resources efficiently
- Scale to large network topologies

Try it out!
https://github.com/usi-systems/camus-compiler
https://github.com/usi-systems/packet-subscriptions-demo