Hybrid Packet and Flow Processing with Flowlets

Yogesh Mundada, Rob Sherwood, Nick Feamster
yhm@cc.gatech.edu, rob.sherwood@stanford.edu, feamster@cc.gatech.edu
Motivation: Two Ways of Processing Traffic

- Process as an **individual packet**
  - Advantages: Flexible, More control, Micro-decisions
  - Disadvantages: Slow
- Process as an **aggregate flow**
  - Advantages: Faster, Macro-decisions
  - Disadvantages: Not enough control and flexibility

- **Flowlets**: Process packets on per-packet *and* per-flow granularity
  - Best of both worlds!
Example: Network Security (Flow-Based Only)

1. Routers forward traffic using only packet header fields.
2. Identify malicious traffic by examining packet data at line speed.

- Normal traffic
- Malicious traffic
- Suspicious traffic

Enterprise Network

Ingress Point

Ingress Point

Ingress Point
Example: Network Security (Packet-Based Only, Distributed)

1. Deploy DPI boxes at each Ingress point

Disadvantages
1. Distributed coordinated attacks
2. Space, Maintenance, Cost, Power

Normal traffic
Malicious traffic
Suspicious traffic

Ingress Point
Enterprise Network
Example: Network Security (Packet-Based Only, Centralized)

1. Redirect traffic through a centralized DPI

Advantages
1. Distributed attacks are detected
2. Less cost, power, maintenance

Disadvantages
1. Not scalable
2. Single point of failure

Normal traffic
Malicious traffic
Suspicious traffic
Better Solution: Both Packets and Flows

1. Redirect only suspicious traffic to the centralized DPI box.

Desired Properties
- Distributed monitoring
- Centralized decision making
- Packet by packet and Flow based processing
Neither Flow nor Packet-Based Processing is Sufficient

<table>
<thead>
<tr>
<th></th>
<th>Examples</th>
<th>Speed</th>
<th>Caching</th>
<th>“Macro” Decisions</th>
<th>“Micro” Decisions</th>
<th>Control/Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Flow-based processing</strong></td>
<td>MPLS, Forwarding tables, Openflow</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Packet-based processing</strong></td>
<td>Active Networks, DPI, Click</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

We need both !!
Main Idea: Combine Packet and Flow Processing

Flow Based Processing
- Fast
- Caching
- Macro-decisions

Packet Based Processing
- Flexible
- Control
- Micro-decisions

Flowlets
- Hybrid model
- Flexibility and Control
- Caching
- Switch between modes
- Expressive power
Talk Outline

- Motivation
- Example: Enterprise Security
- Flowlets: Hybrid Processing
- Background
  - OpenFlow
  - Click
- OpenFlow Click Element
- Demonstration
- Implementation Details and Decisions
- Future work and Summary
Flow-Based Processing: OpenFlow

- **OpenFlow Switch**
  - Secure Channel (secchan)
  - Flow Table

- **Controller**
  - Add/delete flow entry
  - Encapsulated packets
  - Controller discovery

- **PC**
  - OpenFlow Protocol
  - SSL
Packet-Based Processing: Click

- Easy to program
- Intuitive configuration language
- Provides more control
- Large elements library
Hybrid: OpenFlow Click Element

OpenFlow rules can control the output port for each flow
Example: Enterprise Security

Nox Controller

User Space

Kernel Space

Flowlets OpenFlow Element

dpctl

secchan

Worm Detector

eth0

From Device

Queue

To Device

eth0

eth1

From Device

Queue

To Device

eth1
Enterprise Security with Flowlet Processing

1. Redirect only suspicious traffic to the centralized DPI box.

Desired Properties
- Distributed monitoring
- Centralized decision making
- Packet by packet processing
- Flow based processing

Ingress Point

Normal traffic
Malicious traffic
Suspicious traffic
Demonstration

Click With OpenFlow Element

NOX Controller

Node 0

Node 5
Demonstration

Nox Controller

User Space

Kernel Space

Flowlets

OpenFlow Element

dpctl

secchan

eth8

From Device

Queue

To Device

eth8

eth10

From Device

Queue

To Device

eth10

veth0

From Device

Delay Shaper

Burster
Demo Click Configuration

q1 :: Queue;
q2 :: Queue;
q3 :: Queue;
s :: Ofswitch;

FromDevice(eth8, PROMISC true) -> [0]s;
FromDevice(eth10, PROMISC true) -> [1]s;
FromDevice(veth0, PROMISC true) -> [2]s;

s[0] -> Print("Received from eth8", MAXLENGTH 100) -> q1;
s[1] -> Print("Received from eth10", MAXLENGTH 100) -> q2;
s[2] -> Print("Received from veth0", MAXLENGTH 100) -> q3 -> DelayShaper(2) -> b::Burster(0.1) -> q2;

q1 -> ToDevice(eth8);
q2 -> ToDevice(eth10);
Implementation Decisions

1. Implement new Click element using existing OpenFlow switch source code.

2. Implement element as kernel module.

3. Minimize changes to existing OpenFlow code base.
Element Architecture

- Communication Module
  - Openflow messages
  - Openflow Message Parser
- Control Module
  - Commands from Controller
- Forwarding Engine
  - Packets to controller
  - Temporary packets
- Flow Tables (Hash and Linear)
  - Install Rules
  - Consult rules
  - Purge Entries
- Timer (Stale Entries)

Packets in & out
Other Applications of Flowlets

- Loop detection: TTL zero
- Inserting/Deleting extra bits from packet headers (Splicing)
- Packet sampling
- Duplicate packet detection
Future work

- User-space element
- Click vendor-specific action
- Dynamic port addition/deletion
- Dynamic element load/unload
- Dynamic sub-graph load/unload
Flowlets

- Hybrid model
- Flexibility with control
- Caching
- Switch between modes
- Expressive power
Challenges

• Locking
• Refactoring code in correct modules
• Memory allocation
• Multi-threaded code
• Debugging
• Mixing C/C++ code