Adaptive Flow Monitoring & Selective DPI for ONOS

ONS2016
2016. 3. 17

Taesang Choi
ETRI
Topics

• OPEN-TAM ONOS Subproject Overview
• Development Phase 1
  • Adaptive(Effective) Flow Monitoring
• Development Phase 2
  • Selective-DPI (Deep Packet Inspection)
• Demo & Use-Cases
Open-TAM Subproject Overview

• ONOS Architecture Review and OPEN-TAM Architecture Design
  • ONOS wiki page: beginner, user, developer, architecture guide
  • ONOS Sub Component: analysis of the ONOS architecture and associated source code
  • OPEN-TAM Architecture Design

• OPEN-TAM Subproject Setup
  • Kick-off Conference Call with ONOS TSR
  • Proposal of a Subproject: OPEN-TAM
  • Creation of OPEN-TAM Future Project Wiki Page

• Development and Release Plans
  • Started with Blackbird 1.1.0 release
  • Phase 1 result (AFM) was incorporated in EMU release
  • Phase 2 will be in GoldenEye release
Phase 1: Adaptive Flow Monitoring
Adaptive Flow Monitoring Motivation

• Default ONOS Flow Monitoring Issues
  • Default FlowRule service collects all flow information from all devices at every time interval (default 10 seconds)
  • This mechanism may cause performance degradation issue at each collection time in a large-scale real carrier network due to the number of switches and its associated flows (for example; WAN: ~500 Routers, ~10K ports, ~1-10M flows per port)
  • To overcome performance problem in a simple way, we can maintain collection time interval value with a large number. It then causes another critical issue: lack of accuracy

• Our proposal to this problem is an effective flow monitoring scheme called, Adaptive Flow Monitoring Service that can minimize collection computing overhead and provide more accurate flow statistics
Adaptive Flow Monitoring Algorithm

1. Initialize Variables and Setup Tasks() {
   • CAL_AND_SHORT_POLL_INTERVAL= 5, MID_POLL_INTERVAL=10, LONG_POLL_INTERVAL= 15, ENTIRE_POLL_INTERVAL=30
   • Set each tasks being executed at every corresponding time interval }

2. CAL_AND_SHORT_FLOWS_TASK() {
   • IF at first time call or ENTIRE_POLL_INTERVAL, send FlowStatsRequest message getting all flow entries.
   • Else at every call, calculates FlowLiveType and save it appropriate tables
   • Sends FlowStatsRequest message only for SHORT_FLOWS entries

3. MID_FLOWS_TASK() {
   • If at every time call and not ENTIRE_POLL_INTERVAL, sends FlowStatsRequest message only for MID_FLOWS entries }

4. LONG_FLOWS_TASK() {
   • If at every time call and not ENTIRE_POLL_INTERVAL, sends FlowStatsRequest message only for LONG_FLOWS entries }
AFM– Flow Polling Example

- **CAL_AND_SHORT_POLL_INTERVAL= 5**
- **MID_POLL_INTERVAL=10(5*2)**
- **LONG_POLL_INTERVAL= 15(5*3)**
- **ENTIRE_POLL_INTERVAL=30(5*3*2)**

**Simple Polling Count:** 44(29+15)
**Adaptive Polling count:** 29

About **51.7% Reduction**
Adaptive Flow Monitoring Design Strategy

• Re-designing of FlowStatCollector into NewAdaptiveFlowStatCollector used in OpenFlowRuleProvider for each corresponding switch
• Extended StoredFlowEntryWithType from StoredFlowEntry
• Divides FlowEntries into four groups, i.e., Immediate_Flow, Short_Flow, Mid_Flow, and Long_Flow groups
• And uses four time intervals, i.e., Short_Poll_Interval, Mid_Poll_Interval, Long_Poll_Interval, Entire_Poll_Interval
• At every Short_Poll_Interval, calculates all flows into appropriate flow groups and send FlowStatsRequest message about all Short_Flows
• And every Mid_Poll_Interval, send FlowStatsRequest message about all Mid_Flows
• And every Long_Poll_Interval, send FlowStatsRequest message about all Long_Flows
• And finally every Entire_Poll_Interval, send FlowStatsRequest message all flow entries in the switch
NewAdaptiveFlowStatsCollector

<table>
<thead>
<tr>
<th>I</th>
<th>StoredFlowEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>TypedStoredFlowEntry</td>
</tr>
<tr>
<td>Member</td>
<td>public static enum FlowLiveType { IMMEDIATE_FLOW, SHORT_FLOW, MID_FLOW, LONG_FLOW, UNKNOWN_FLOW }</td>
</tr>
</tbody>
</table>
| Method | public int flowLiveType();
|         | public void setFlowLiveType(FlowLiveType liveType); |

<table>
<thead>
<tr>
<th>C</th>
<th>DefaultFlowEntry</th>
<th>Impl</th>
<th>TypedStoredFlowEntry</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>DefaultTypedFlowEntry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Member</td>
<td>private FlowLiveType liveTyep;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Method | public DefaultTypedFlowEntry(FlowRule rule, FlowLiveType liveType);  
|         | public DefaultTypedFlowEntry(FlowEntry fe, FlowLiveType liveType);  
|         | @Override  
|         | public int flowLiveType();  
|         | @Override  
|         | public void setFlowLiveType(FlowLiveType liveType); |
**NewAdaptiveFlowStatsCollector**

<table>
<thead>
<tr>
<th>C</th>
<th>NewAdaptiveFlowStatsCollector</th>
</tr>
</thead>
</table>
| Member | private final Logger log = getLogger(getClass());  
private final OpenFlowSwitch sw;  
private ScheduledExecutorService adaptiveFlowStatsScheduler =  
newScheduledThreadPool(4, groupedThreads("onos/flow", "device-stats-collector-%d"));  
private CalAndShortFlowsTask calAndShortFlowsTask;  
private MidFlowsTask midFlowsTask;  
private LongFlowsTask longFlowsTask;  
private int calAndShortPollInterval = 5;  
private int midPollInterval = 10;  
Private int longPollInterval = 15;  
Private int entirePollInterval = 30;  
private final InternalDeviceFlowTable deviceFlowTable = new InternalDeviceFlowTable(); |
| Method | NewAdaptiveFlowStatsCollector(OpenFlowSwitch, int);  
synchronized void adjustCalPollInterval(int pollInterval);  
private class CalAndShortFlowsTask implements Runnable { }  
private class MidFlowsTask implements Runnable { }  
private class LongFlowsTask implements Runnable { }  
public synchronized void start();  
public synchronized void stop();  
public boolean addWithFlowRule(FlowRule... flowRules);  
public boolean addOrUpdateFlows(FlowEntry... flowRules);  
public boolean removeFlows(FlowEntry... flowRules);  
public boolean pushFlowMetrics(List<FlowEntry> flowEntries); |

<table>
<thead>
<tr>
<th>C</th>
<th>OpenFlowRuleProvider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member</td>
<td>Added NewAdaptiveFlowStatsCollector method call statements at every flow entry ADD, REMOVED, and UPDATED</td>
</tr>
</tbody>
</table>
| Method | public getFlowCount();  
public TypedStoredFlowEntry getFlowEntry(FlowRule rule);  
public Set<StoredFlowEntry> getFlowEntries();  
public void add(TypedStoredFlowEntry rule);  
public void addWithCalAndSetFlowLiveType(TypedStoredFlowEntry rule);  
{ getFlowEntriesInternal(rule.id()).add((StoredFlowEntry) rule); }  
public boolean remove(FlowEntry rule);  
public void checkAndMoveLiveFlowAll(); |

<table>
<thead>
<tr>
<th>Internal</th>
<th>InternalDeviceFlowTable</th>
</tr>
</thead>
</table>
| Member | private final Map<FlowId, Set<StoredEntryWithWithType>> deviceFlowEntries = Maps.newConcurrentMap();  
private final Set<StoredFlowEntry> shortFlows = new HashSet<>();  
private final Set<StoredFlowEntry> midFlows = new HashSet<>();  
private final Set<StoredFlowEntry> longFlows = new HashSet<>(); |
| Method | public getFlowCount();  
public TypedStoredFlowEntry getFlowEntry(FlowRule rule);  
public Set<StoredFlowEntry> getFlowEntries();  
public void add(TypedStoredFlowEntry rule);  
public void addWithCalAndSetFlowLiveType(TypedStoredFlowEntry rule);  
{ getFlowEntriesInternal(rule.id()).add((StoredFlowEntry) rule); }  
public boolean remove(FlowEntry rule);  
public void checkAndMoveLiveFlowAll(); |
FlowStatisticManager Class Hierarchy

<table>
<thead>
<tr>
<th>Class</th>
<th>AbstractShellCommand</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>GetFlowStatistics</td>
<td>Member</td>
</tr>
<tr>
<td></td>
<td>Method</td>
</tr>
<tr>
<td></td>
<td>execute()</td>
</tr>
<tr>
<td></td>
<td>{ flowStatsService.load(device, inLiveType, inInstructionType, topn); }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>FlowStatisticManager</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Member</td>
</tr>
<tr>
<td></td>
<td>Method</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>InternalFlowRuleStatsListener</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TypedStatistics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DistributedFlowStatisticStore</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>FlowStatisticStore</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FlowStatisticService</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Post FlowRuleEvent Add/Update/Remove
- GetCurrentFlowStatistic (cp);
- GetPreviousFlowStatistic (cp);
- Add/Update/Remove flowRule

12
FlowStatisticManager Class Design

<table>
<thead>
<tr>
<th>Class</th>
<th>AbstractShellCommand</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>GetFlowStatistics</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Member</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>execute()</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```java
execute() {
    statisticFlowService.loadSummary(device, port);
    statisticFlowService.loadAllByType(device, port, flow_type, inst_type);
    statisticFlowService.loadTopnByType(device, port, flow_type, inst_type, int topn);
}
```

<table>
<thead>
<tr>
<th>Class</th>
<th>TypedFlowEntryWithLoad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Member</td>
<td>ConnectionPoint cp;</td>
</tr>
<tr>
<td></td>
<td>TypedFlowEntry typedFlowEntry;</td>
</tr>
<tr>
<td></td>
<td>Load load;</td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>SummaryFlowEntryWithLoad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Member</td>
<td>ConnectionPoint cp;</td>
</tr>
<tr>
<td></td>
<td>Load totalLoad;</td>
</tr>
<tr>
<td></td>
<td>Load immediateLoad;</td>
</tr>
<tr>
<td></td>
<td>Load shortLoad;</td>
</tr>
<tr>
<td></td>
<td>Load midLoad;</td>
</tr>
<tr>
<td></td>
<td>Load longLoad;</td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface</th>
<th>FlowStatisticService</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>FlowStatisticManager</td>
</tr>
<tr>
<td>Member</td>
<td>@Override</td>
</tr>
<tr>
<td>Method</td>
<td>Map&lt;ConnectionPoint, SummaryFlowEntryWithLoad&gt; loadSummary(Device device) { }</td>
</tr>
<tr>
<td></td>
<td>SummaryFlowEntryWithLoad loadSummary(Device device, PortNumber pNumber) { }</td>
</tr>
</tbody>
</table>

```java
Map<ConnectionPoint, List<TypedFlowEntryWithLoad>> loadAllByType(Device device, FlowLiveType liveType, InstructionType instType) { } |
List<TypedFlowEntryWithLoad> loadAllByType(Device device, PortNumber pNumber, FlowLiveType liveType, InstructionType instType) { } |
```

<table>
<thead>
<tr>
<th>Interface</th>
<th>FlowStatisticStore</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>DistributedFlowStatisticStore</td>
</tr>
<tr>
<td>Member</td>
<td>private Map&lt;ConnectPoint, Set&lt;FlowEntry&gt;&gt; previous = new ConcurrentHashMap&lt;&gt;();</td>
</tr>
<tr>
<td></td>
<td>private Map&lt;ConnectPoint, Set&lt;FlowEntry&gt;&gt; current = new ConcurrentHashMap&lt;&gt;();</td>
</tr>
<tr>
<td>Method</td>
<td>void removeFlowStatistic(FlowRule rule) { }</td>
</tr>
<tr>
<td></td>
<td>void addFlowStatistic(FlowEntry rule) { }</td>
</tr>
<tr>
<td></td>
<td>void updateFlowStatistic(FlowEntry rule) { }</td>
</tr>
<tr>
<td></td>
<td>Set&lt;FlowEntry&gt; getCurrentFlowStatistic(ConnectPoint connectPoint) { }</td>
</tr>
<tr>
<td></td>
<td>Set&lt;FlowEntry&gt; getPreviousFlowStatistic(ConnectPoint connectPoint) { }</td>
</tr>
</tbody>
</table>
Advanced Adaptive Flow Sampling

• To reduce performance overhead further,
  • At every polling interval, instead of getting all flow stats of corresponding flow entries (SHORT, MID, and LONG_FLOWS)
  • Apply one of the following sampling methods for each flow table groups, especially for SHORT or MID groups with large number of flow entries
    • No sampling (get all, same as current method)
    • Random sampling (at every count (default=100) flow entry number)
    • Top-n sampling (only top-n (default=1000) flows based on Bytes per Second rates)
    • Probabilistic sampling (only probabilistic (default=1/2) success flow entry number)
    • ...

Phase 2: Selective Deep Packet Inspection
Selective-DPI Motivation

• Current Problem
  • Current ONOS flow can be classified and selected by lower-level FlowSelection criteria based on FlowRule entry (e.g., ports, ether_type, vlan_id, 5-tuple, etc.)
  • There is **no application classification service** for ONOS data plane user-data

• We propose to add a **Selective DPI service** that can filter data plane user-data from controller traffic and classify them with application level granularity by using a open source DPI s/w
Selective DPI: OpenDPIManager Architecture – On-ONOS Platform

OpenDPIApp
Component

DPIManager

DPIEngine

DPIEngine

DPIEngine

RawPacketDumpManager
Component

RawPacketDumpAdminService

RawPacketDumpService
(Flow)

RawPacketDumper

wPacketDumper

wPacketDumper

RuleStore

RawPacketDumpService
(Flow)

RawPacketDumper

wPacketDumper

wPacketDumper

Capture
Interface

Neighbor
OpenDPIApps

DPIManager

DPIEngine

DPIEngine

DPIEngine

RawPacketDumpManager
Component

RawPacketDumpAdminService

RawPacketDumpService
(Flow)

RawPacketDumper

wPacketDumper

wPacketDumper

RuleStore

RawPacketDumpService
(Flow)

RawPacketDumper

wPacketDumper

wPacketDumper

Capture
Interface

17
Selective DPI: OpenDPIManager Architecture - off-ONOS Platform

- OpenDPIApp Component
- DPI Service Manager
- RawPacketDumpManager Component
- RawPacketDumpProvider Component
- DPI Engine
- DPI Engine
- DPI Engine
- EXternal OpenDPIEngine Stores
- Packet Dumper
- Packet Dumper
- Packet Dumper
- Intel DPDK 2.2.0
- ntop NDPI-1.7
- Capture Interface
Selective-DPI: Scalability and Performance

- On-ONOS Platform solution can be used for limited number flows that don’t influence performance and reliability of ONOS
  - The scalability limitation of on-ONOS platform can be determined with field tests
  - It is a native ONOS DPI solution which can be used for certain premium service applications or ONOS control & management traffic trouble shooting purposes

- Off-ONOS Platform solution is a full-blown scalable DPI solution
  - It is a non native solution
  - Require a dedicated stand-alone DPI engine which can be a hardware-based solution for the performance assurance or a container-based virtual software function for service agility

- Issue seems to be
  - Not choosing one of them
  - But finding an appropriate ratio of deployment of both solutions depending on the service requirements
Demo and Use Cases
GetFlowStatistics Command

• CLI:onos> **get-flow-stats Device[/Port] [--summary | --all | --top number] [--flow type | --instruction type] [--help]**
  • Default: onos> get-flow-stats Device –summary
  • -s, --summary: show summary flow rule stats based on flow live type
  • -a, --all: show all flow rule stats
  • -t, --top N : show only top N flow rule stats, 0 < N <= 1000, default = 100
  • -f, --flow type: show only flow rule live type = [IMMEDIATE| SHORT | MID | LONG]
  • -i, --instruction type: show flow rule instruction type = [DROP | OUTPUT | GROUP | L0MODIFICATION | L2MODIFICATION | TABLE | L3MODIFICATION | METADATA]
CLI:onso> get-flow-stats

get-flow-stats --help

get-flow-stats --summary

get-flow-stats --all

get-flow-stats --topn 10
CLI:onso> dpsis & -j

- Lists the DPI results
- Received from external DPI engine with table or json type.

```
dpis --help
```

```
3. saycon@raptor:~$onos (ssh)

```

```
23
```
CLI:onso> dpis –n 2

• Lists the latest n (MAX=100) DPI result entries
Selective DPI Use Case for Function Service Chaining Classification Application (off-Platform)

1: DPI Start Request: Identification of a specific flows for Service Chaining Classification

2.1: Create and RawPacket Dumper and Start to Capture
2.2 Then enforce an instance of DPI Engine
2.3 Returns Classification Results

3: Return the result and notify DPI External PacketDumper Access Information

4: Request Redirection of flow for DPI with Access Info

5: Setup Flow Rule for Redirection of Requested flow(s)

6: Redirection of Requested flow(s) to DPI Subsystem

7: Packet dumps into a destined DPI engine for analytics

8: DPI Classification Results for Service Chaining Classification
Thanks for your Attention

choits@etri.re.kr