ETRI openTAM Subproject

ONOS Day

2015. 9. 15

SangSik Yoon

Chunglae Cho

Taesang Choi

SDN기술연구실

ETRI

ETRI-openTAM Planning Overview

- Preparing and Setup Subproject (~June 30)
- Development Phase 1 (~Aug. 31)
  - Adaptive Flow Sampling
- Development Phase 2 (~Oct. 31)
  - Open Selective-DPI(Deep Packet Inspection)

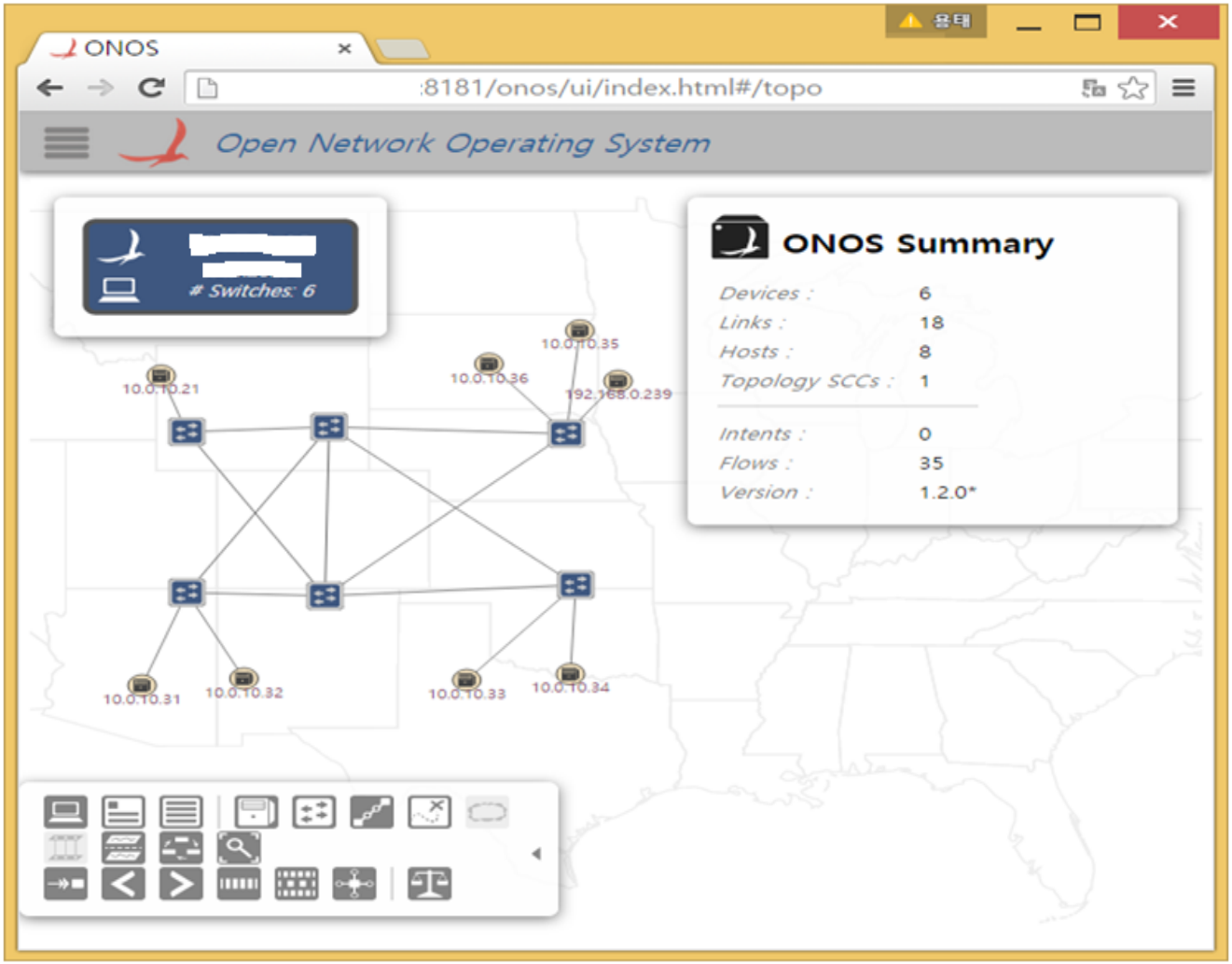
Preparing and Setting-up the Subproject

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

- Setting up Development Environment
  - Downloaded Blackbird 1.1.0 release and installed at Ubuntu 14.04.2
  - Setting Environment & Hands on examples : mininet and OVS

- OpenTAM Subproject Setup
  - Kick-off Conference Call
  - Proposal of a Subproject: OpenTAM
  - Creation of OpenTAM Project Wiki Page

ONOS GUI – ETRI SDN Testbed Env.



Development Phase 1
-Adaptive Flow Sampling Service

- Current Problem
  - Current 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 generates another critical issue: **lack of accuracy**

  - Our proposal to this problem is an effective flow monitoring scheme called, **Adaptive(Effective) Flow Sampling Service** that can minimize collection computing overhead and provide more accurate flow statistics

Development Phase 1
-Adaptive Flow Sampling Service

- OpenTAM AFS Service Development Completed
  - OpenTAM : NewAdativeFlowStatsCollector Implementation

  - First committed when Aug.17 to onos gerrit repository
  - completed 11 patches reviewed by the ONOS QA team
  - System Integration tests are underway

Adaptive Flow Sampling Service

- Basic Design Strategy
  - Re-designing 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_INTERAVAL, MID_POLL_INTERVAL, LONG_POLL_INTERVAL, ENTIRE_POLL_INTERVAL
  - At every SHORT_POLL_INTERAL, 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

Adaptive Flow Sampling 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 about SHORT_FLOWs entries
- 3. MID_FLOWS_TASK() {
  - If at every time call and not ENTIRE_POLL_INTERVAL, sends FlowStatsRequest message only about SHORT_FLOWs entries }
- 4. LONG_FLOWS_TASK() {
  - If at every time call and not ENTIRE_POLL_INTERVAL, sends FlowStatsRequest message only about LONG_FLOWs entries }

Adaptive Flow Sampling Design
- NewAdaptiveFlowStatsCollector -1

C	NewAdaptiveFlowStatsCollector

Member	pivate 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 InteralDeviceFlowTable deviceFlowTable = new InteralDeviceFlowTable();

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);

C	OpenFlowRuleProvider
Member	
Method	Added NewAdaptiveFlowStatsCollector method call statements at every flow entry ADD, REMOVED, and UPDATED

Internal C	InteralDeviceFlowTable

Member	private final Map<FlowId, Set<StoredFlowEntryWithType>> 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();

Adaptive Flow Sampling Design
- NewAdaptiveFlowStatsCollector -2

I	StoredFlowEntry

I	TypedStoredFlowEntry
Member	public static enum FlowLiveType { IMMEDIATE_FLOW, SHORT_FLOW, MID_FLOW, LONG_FLOW, UNKNOWN_FLOW }
Method	public int flowLiveType(); public void setFlowLiveType(FlowLiveType liveType);

C	DefaultFlowEntry	Impl	TypedStoredFlowEntry

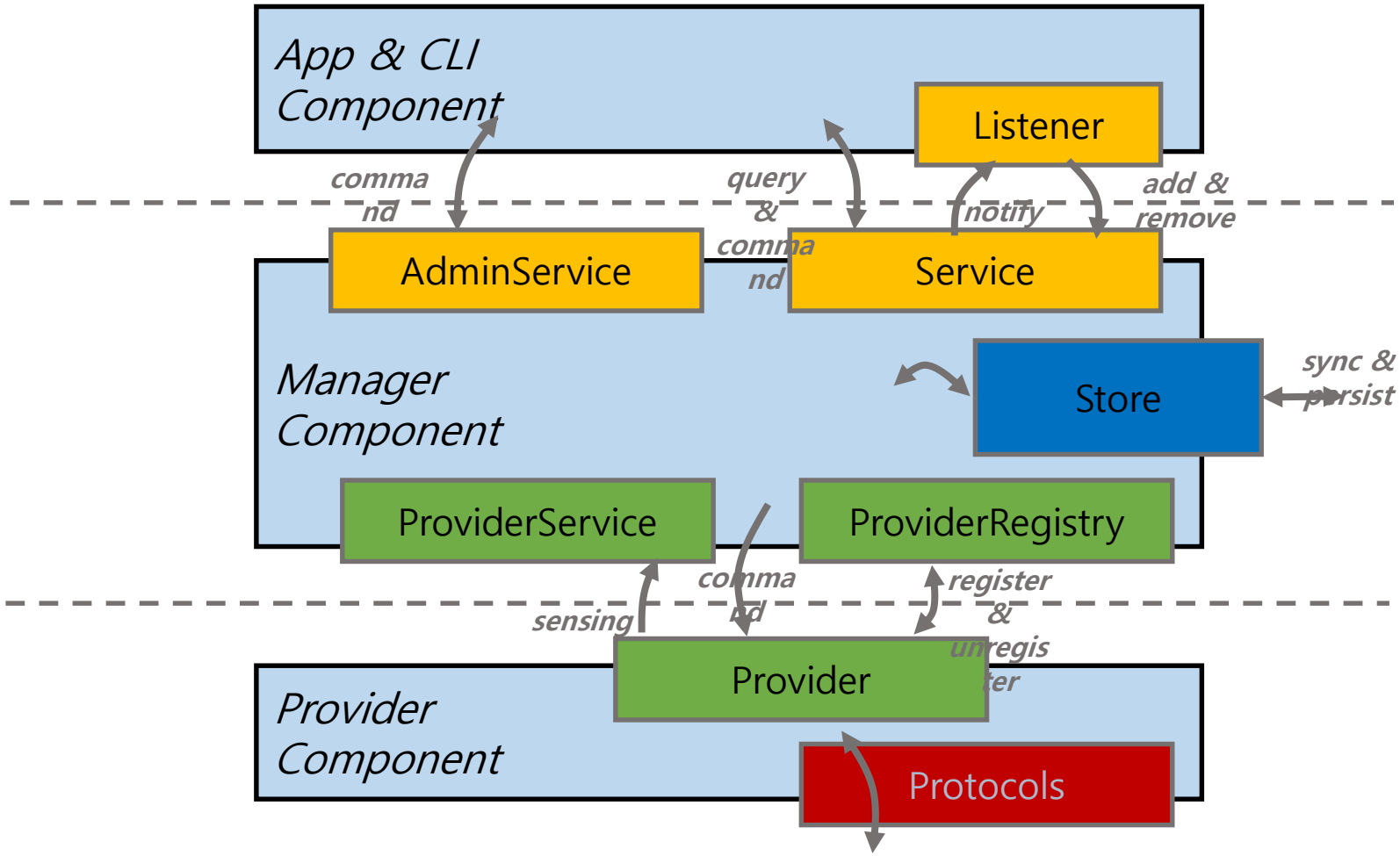
C	DefaultTypedFlowEntry
Member	private FlowLiveType liveTyep;

Method	public DefaultTypedFlowEntry(FlowRule rule, FlowLiveType liveType);
	public DefaultTypedFlowEntry(FlowEntry fe, FlowLiveType liveType);
	
	@Override
	public int flowLiveType();
	@Override
	public void setFlowLiveType(FlowLiveType liveType);

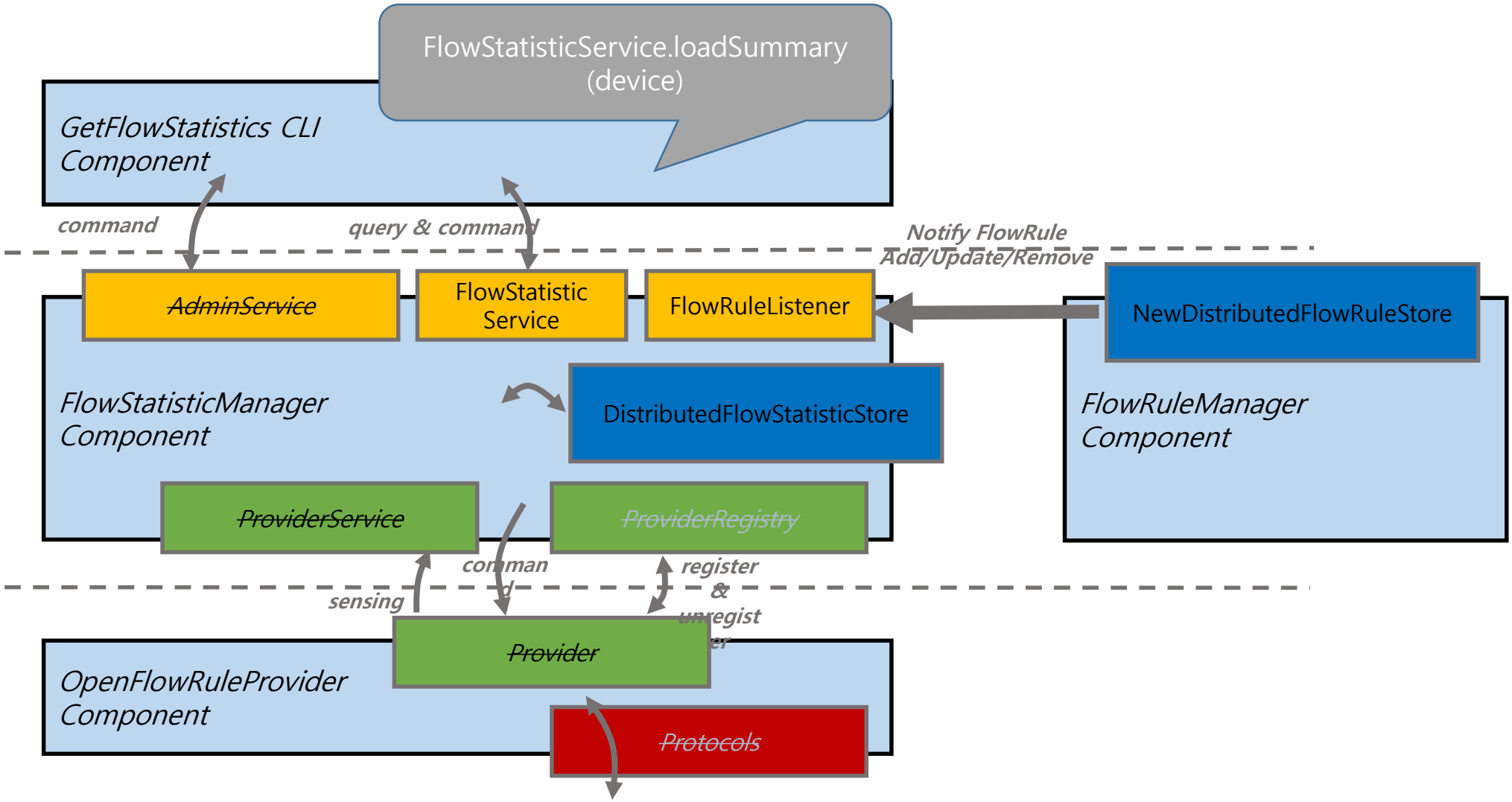
Advanced Adaptive Flow Samling

- In the basic sampling concept,
  - At every polling interval, instead of getting all flow stats of corresponding flow entries (SHORT, MID, and LONG_FLOWs)
  - Adopts one of following sampling methods for each flow table groups
    - 1. no sampling (get all, same as current method)
    - 2. random sampling (at every count(default=100) flow entry number, get it only)
    - 3. top-n sampling (only top-n(default=1000) flows based Bytes per Second rates, get them all)
    - 4. probability sampling (only probabilistic(default=1/2) success flow entry number
    - 5. …

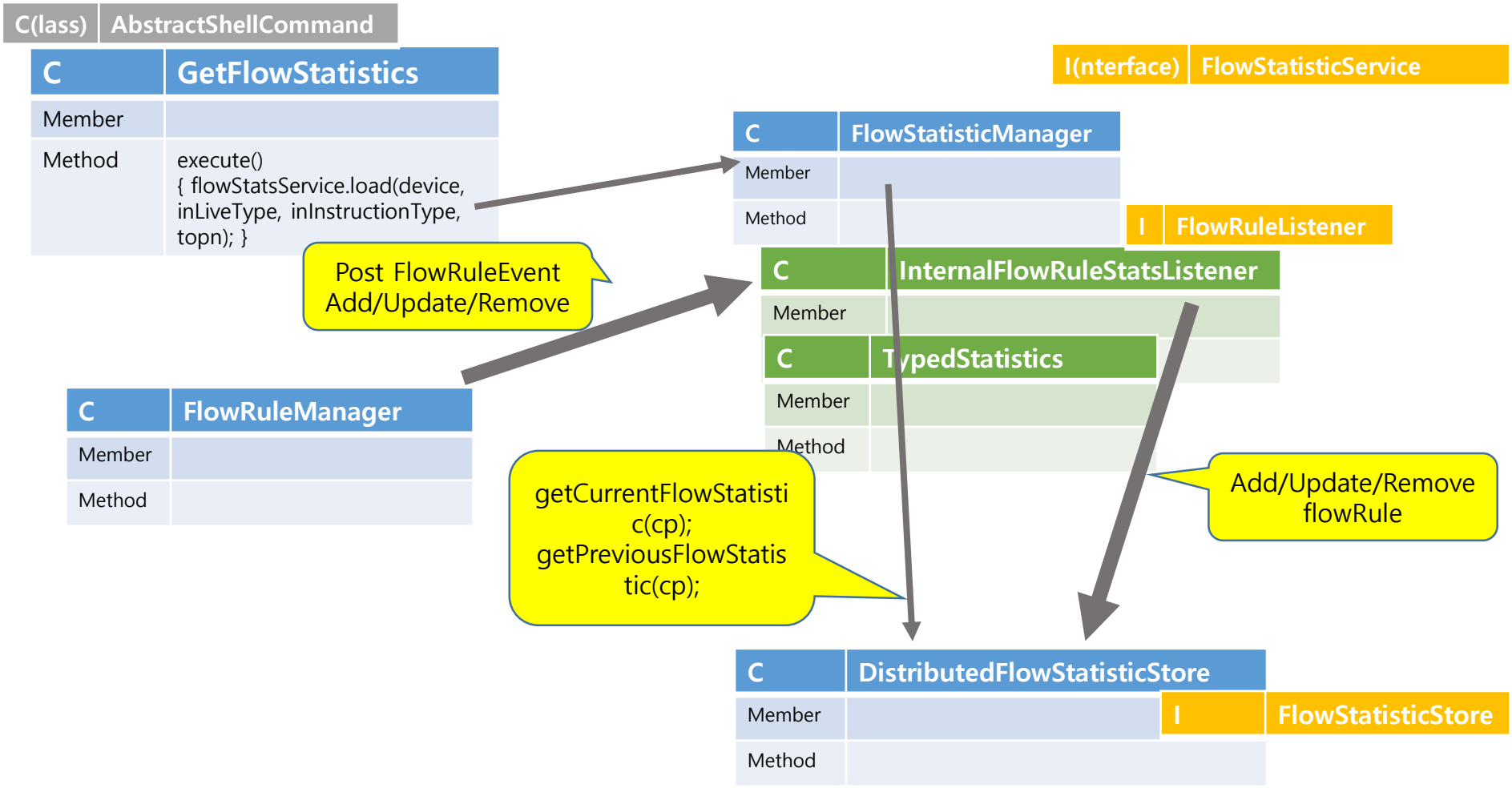
ONOS Subsystem Structure



FlowStatisticManager Subsystem Design



FlowStatisticManager Class Hierarchy



FlowStatisticManager Class Design

C(lass) AbstractShellCommand

C	GetFlowStatistics
Member	

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

I(nterface) FlowStatisticService

C	FlowStatisticManager
Member	
Method	@Overrride Map<ConnectionPoint, SummaryFlowEtnryWithLoad> loadSummary(Device device) { } SummaryFlowEtnryWithLoad loadSummary(Device device, PortNumer pNumber) { } Map<ConnectionPoint, List<TypedFlowEntryWithLoad>> loadAllByType(Device device, FlowLiveType liveType, InstructionType instType) { } List<TypedFlowEntryWithLoad> loadAllByType(Device device, PortNumer pNumber, FlowLiveType liveType, InstructionType instType) { } Map<ConnectionPoint, List<TypedFlowEntryWithLoad>> loadTopnByType(Device device, FlowLiveType liveType, InstructionType instType, int topn) { } List<TypedFlowEntryWithLoad> loadTopnByType(Device device, PortNumer pNumber, FlowLiveType liveType, InstructionType instType, int topn) { }

C	TypedFlowEntryWithLoad
Member	ConnectionPoint cp; TypedFlowEntry typedFlowEntry; Load load;
Method	

C	SummaryFlowEntryWithLoad
Member	ConnectionPoint cp; Load totalLoad; Load immediateLoad; Load shortLoad; Load midLoad; Load longLoad;
Method	

I(nterface) FlowStatisticStore

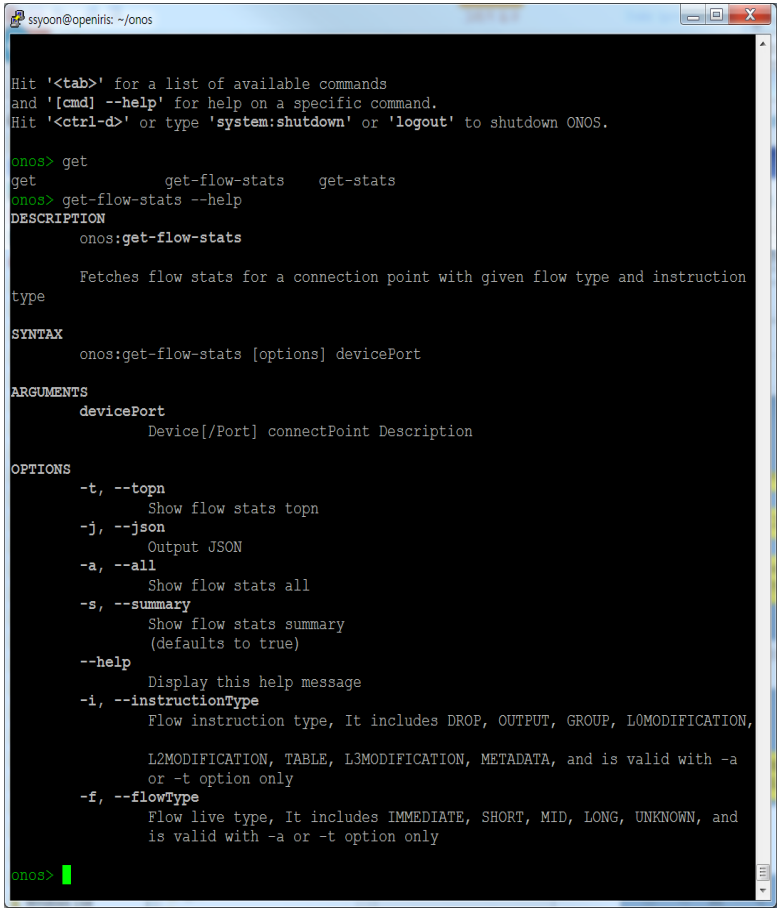
C	DistributedFlowStatisticStore
Member	private Map<ConnectPoint, Set<FlowEntry>> previous = new ConcurrentHashMap<>(); private Map<ConnectPoint, Set<FlowEntry>> current= new ConcurrentHashMap<>();
Method	void removeFlowStatistic(FlowRule rule) { } void addFlowStatistic(FlowEntry rule) { } void updateFlowStatistic(FlowEntry rule) { } Set<FlowEntry> getCurrentFlowStatistic(ConnectPoint connectPoint) { } Set<FlowEntry> getPreviousFlowStatistic(ConnectPoint connectPoint) { }

GetFlowStatistics Command

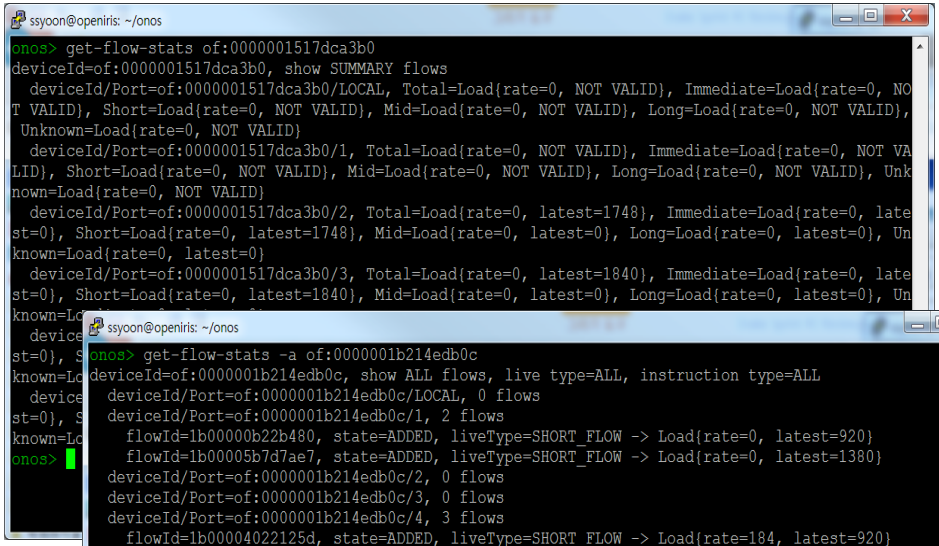
- CLI:onos> **get-flow-stats** *Device*[/*Port*] [--summary | --all | --top *number*] [--flow *type* | --instruction *type*] [--help]
  - Default: onos> get-flow-stats Device –summary
  - -s, --summary: show summary flow rule stats based on flow live type
  - -a, --all: show all flow rule stats
  - -t, --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

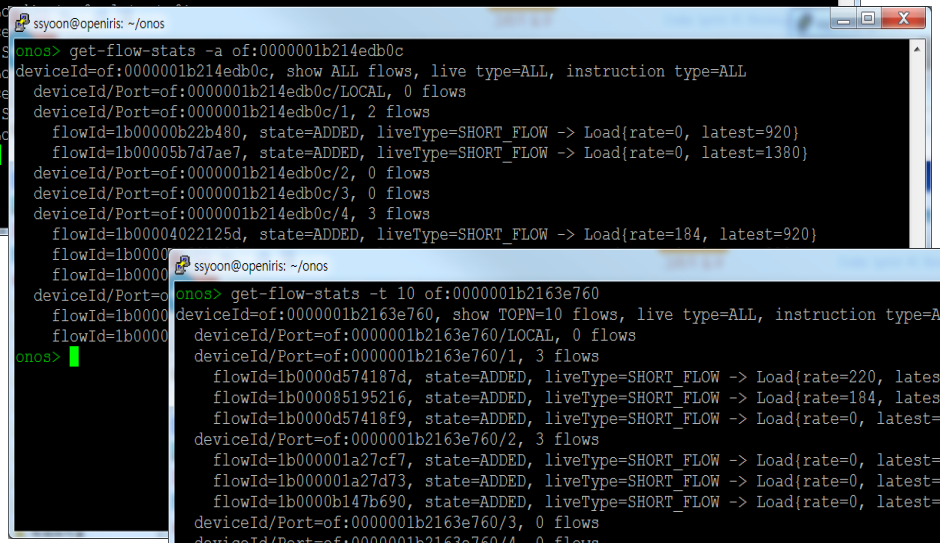
get-flow-stats --help



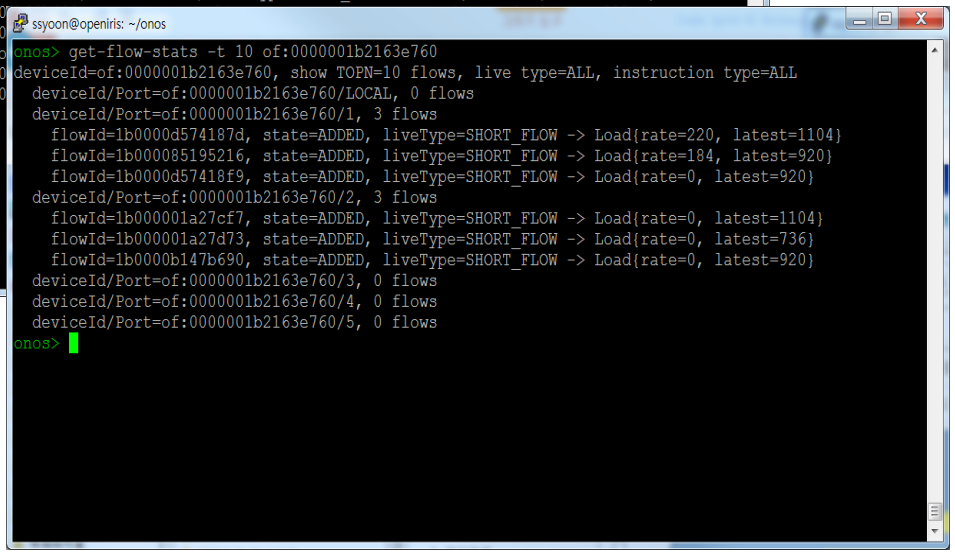
get-flow-stats --summary



get-flow-stats --all



get-flow-stats --topn 10



NewAdaptiveFlowStatsCollector Sources

- A: core/api/src/main/java/org/onosproject/net/flow/DefaultTypedFlowEntry.java
- A: core/api/src/main/java/org/onosproject/net/flow/TypedStoredFlowEntry.java
- A: providers/openflow/flow/src/main/java/org/onosproject/provider/of/flow/impl/NewAdaptiveFlowStatsCollector.java
- M: providers/openflow/flow/src/main/java/org/onosproject/provider/of/flow/impl/OpenFlowRuleProvider.java

- M: Modified, A: Added

FlowStatisticService Sources

- M: cli/src/main/java/org/onosproject/cli/Comparators.java
- A: cli/src/main/java/org/onosproject/cli/net/GetFlowStatistics.java
- M: cli/src/main/resources/OSGI-INF/blueprint/shell-config.xml
- A: core/api/src/main/java/org/onosproject/net/flow/DefaultTypedFlowEntry.java
- A: core/api/src/main/java/org/onosproject/net/flow/TypedStoredFlowEntry.java
- A: core/api/src/main/java/org/onosproject/net/statistic/FlowStatisticService.java
- A: core/api/src/main/java/org/onosproject/net/statistic/FlowStatisticStore.java
- A: core/api/src/main/java/org/onosproject/net/statistic/SummaryFlowEntryWithLoad.java
- A: core/api/src/main/java/org/onosproject/net/statistic/TypedFlowEntryWithLoad.java
- M: core/net/pom.xml
- A: core/net/src/main/java/org/onosproject/net/statistic/impl/FlowStatisticManager.java
- A: core/store/dist/src/man/java/org/onosproject/store/statistic/impl/DistributedFlowStatisticStore.java

- M: Modified, A: Added

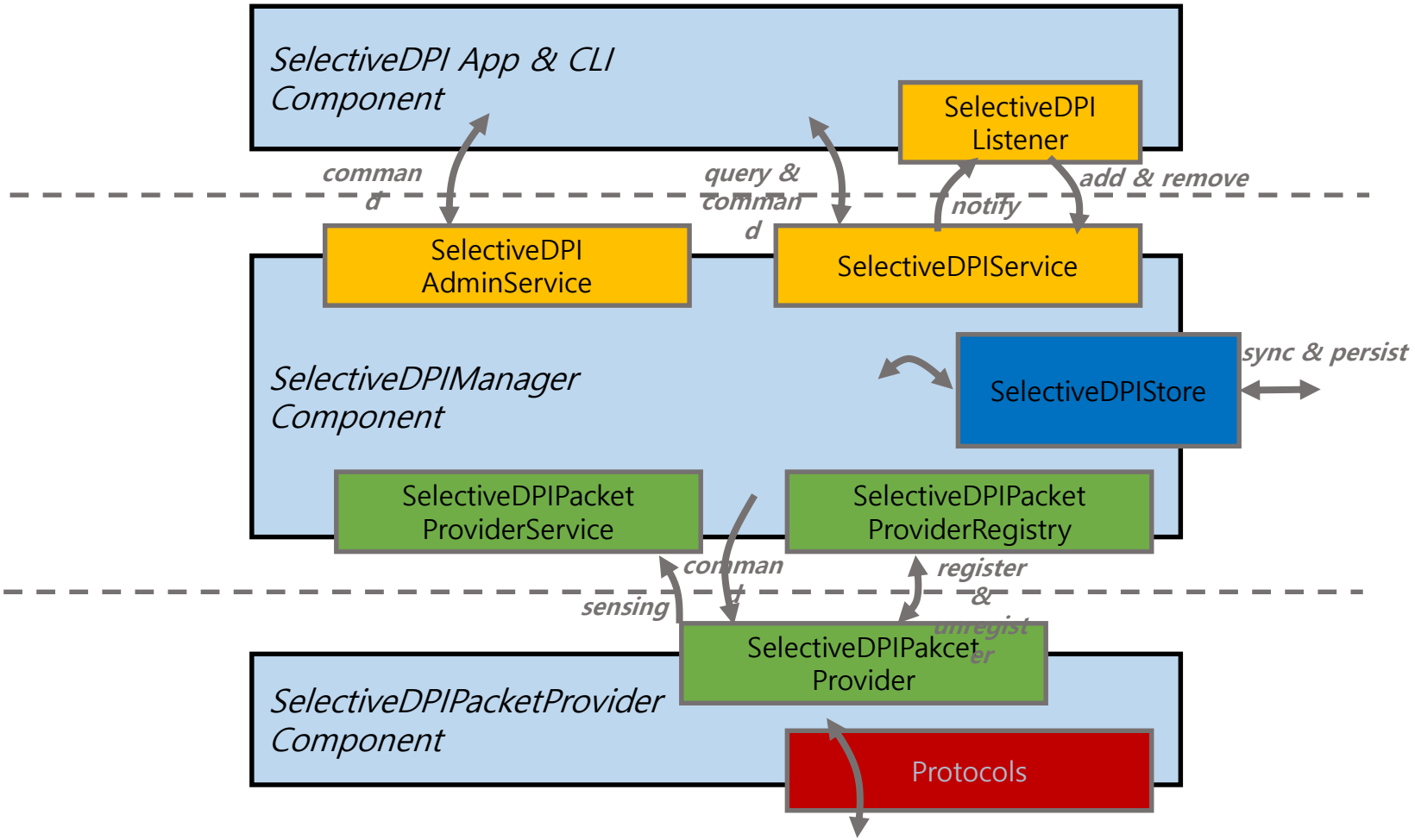
Development Phase 2 (~Oct. 31)
-Open Selective-DPI(Deep Packet Inspection)

- Current Problem
  - Current ONOS flow can be classified and selected by lower-level FlowSelection criteria based on FlowRule entry (eg., 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
  - Application of S-DPI can be application traffic analysis, service chaining classification, etc. We will take phased approach from a simple application to complex ones

Development Phase 2
-Open Selective-DPI(Deep Packet Inspection)

- Service Architecture and Development Module



Development Phase 2
-Open-DPI(Deep Packet Inspection)

- Development Modules
  - SelectiveDPI CLI App Component: SelectiveDPIListener
  - SelectiveDPIManager Component: SelectiveDPIAdminService, SelectiveDPIService, SelectiveDPIStore, SelectiveDPIPacketProviderServie, SelectiveDPIPacketProviderRegistry
  - SelectiveDPIPacketProvider Component: OpeSelectivenDPIPacketProvider

- Detailed Development Plan
  - Architecture & Interface Module Design (~Sept. 11)
  - Development of SelectiveDPIPacketProvider Interface (~Sept. 25)
  - Integration to existing OpenDPI s/w with our Developed Provider and Testing (~Oct. 9)
  - Development of SelectiveDPIManager and CLI application (~Oct. 23)
  - Integration Testing and Performance Evaluation (~Oct. 31)
  - Code contribution and wiki update (~Oct. 31)

Thank You
Q&A
choits@etri.re.kr