Introducing ONOS Blackbird

SDN Network OS for Service Provider Networks

Charles M.C. Chan

Mar. 28, 2015

SDN Developer Society, Taipei

Who Am I ?

- **Charles** Min-Cheng Chan / 詹珉誠 / @rascov
- **Ph.D. Candidate**, National Chiao Tung University
- **Team Lead**, D-Link NCTU Joint Research Center
- **Individual Contributor**, ONOS Project
  - 14 commits / **4,861 ++** / **667 --**
  - **IPv6**: initial planning and development
  - **CVE-2015-1166**: denial-of-service due to exception handling while deserializing malformed packets

Outline

- **Introducing ONOS Blackbird**
  - **Motivation**
  - **Objective**
  - **Key Features**
  - **Use Cases**
- ONOS Architecture
- Performance Evaluation
- How to write an ONOS application
- ONOS Toward IPv6

Motivation

- Why are **service providers** interested in SDN
  - Reduce **CAPEX** and **OPEX**
  - Cloud-style **agility**, **flexibility**, **scalability**
  - Roll out services rapidly
  - Reduce operational complexity, increase visibility

Objective

- Strict requirements on SDN control plane
  - Handle **hundreds of millions** of end points
  - **Five nines** availability, high performance, low latency
  - Easily create and deliver services
  - Seamless migration of existing networks

➡ ONOS is designed for these strict requirements

Key Features - Avocet

- High-availability, scalability, performance
  - **Distributed Core**
- Northbound abstractions
  - **Application Intent Framework**
- Southbound abstractions
  - Protocol adapters, OF 1.0/1.3 for now
    - Based on Loxigen
- GUI
- Open source
  - Apache 2.0 License

Key Features - Blackbird

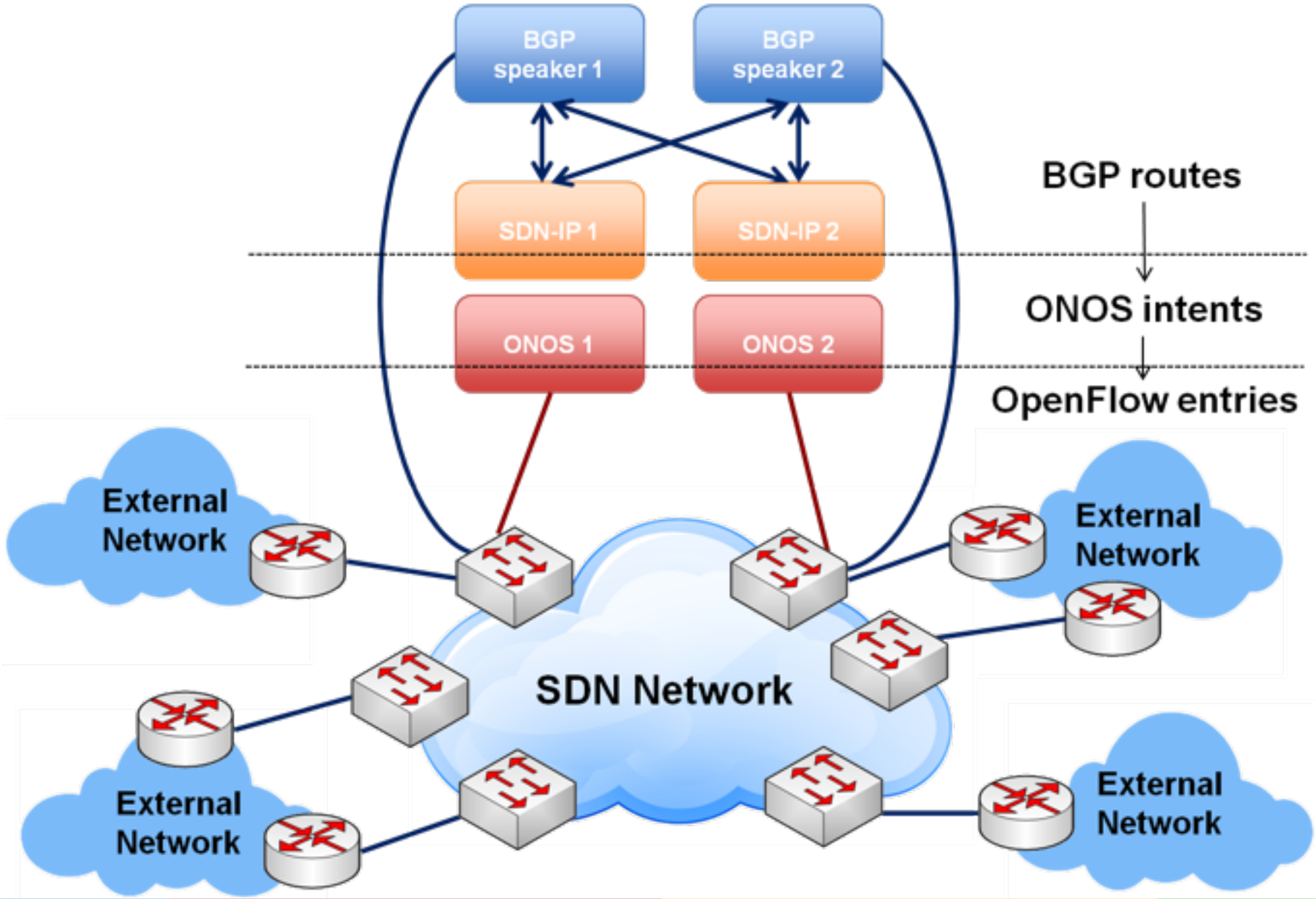
- **IPv6 Support (experimental)**
- Distributed Application Framework
- Internet2 deployment / SDN-IP
- **A lot of testing**
- **Performance evaluation / enhancement**
- **Hazelcast -> RAFT**
  - Eventually consistent map
  - Strongly consistent map
- REST API
- Modular and extensible GUI
  - Angular JS

Key Features - Cardinal

- Security mode
  - Application permissions
- Configuration Model
- Multicast
  - SinglePointToMultiPoint (S2M) intent
- IPv6
- NETCONF
- IP RAN (ONS Demo)
  - L3 VPN
- Internet2 deployment (ONS Demo)

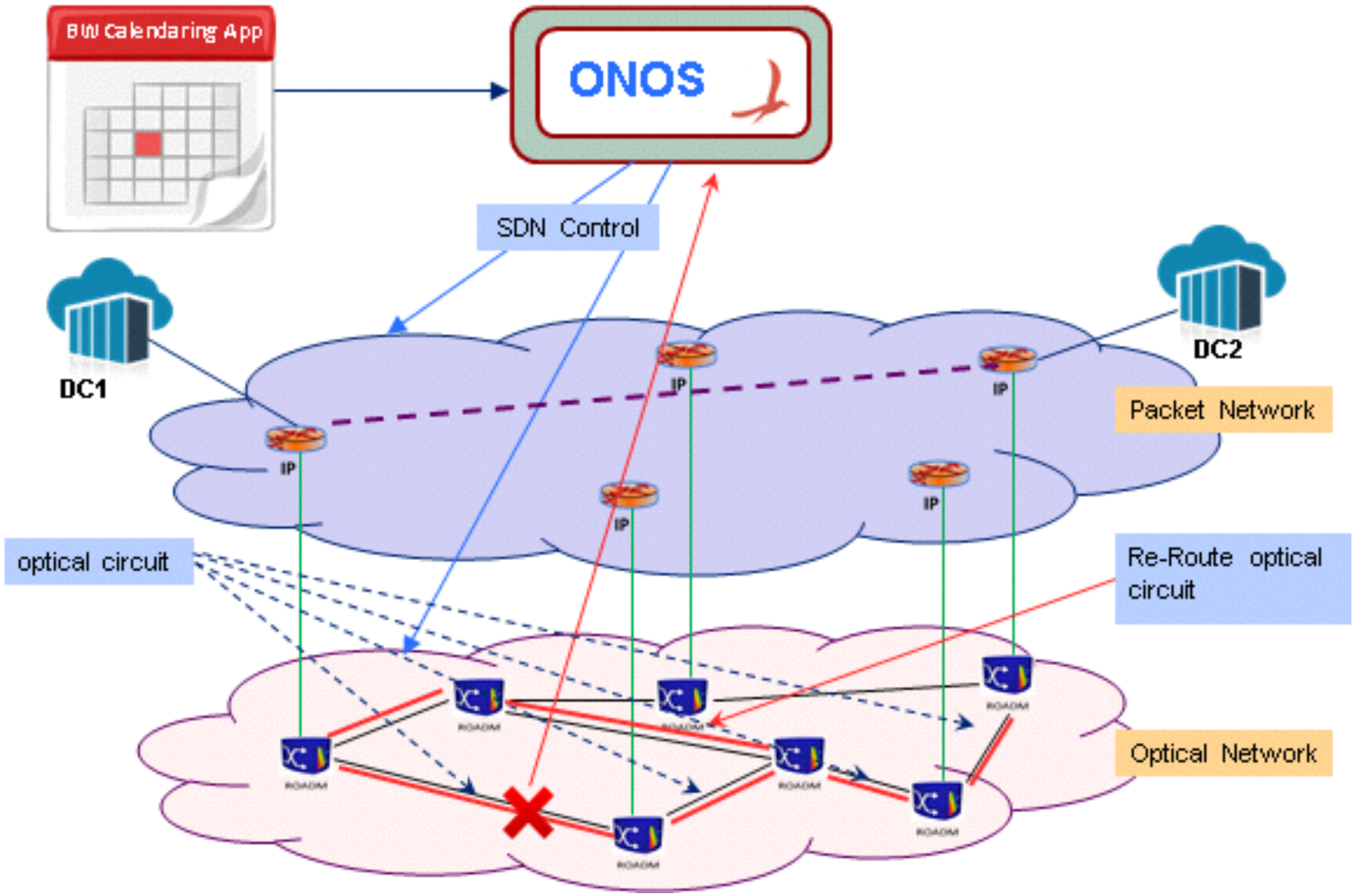
Use Case: SDN-IP

- Talk to external network using **BGP**
- **Challenge:** Real-world development (500k+ routes)
  - Flow entry query between controller and switch paralyzes the control plane

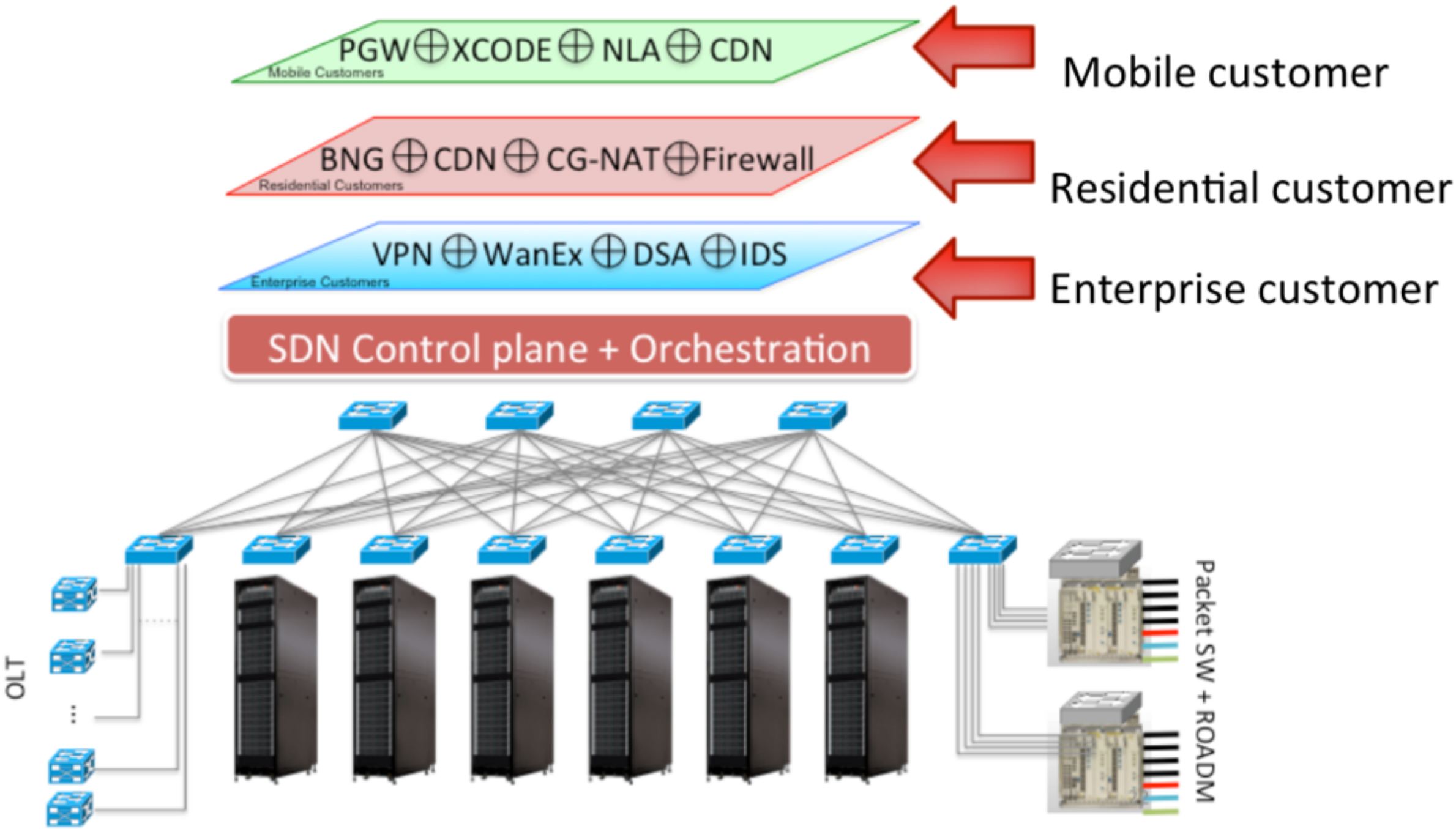


Use Case: Packet / Optical Network

- **On-demand provisioning** of bandwidth (calendar app)
- Automated handling of failures and seamless restoration

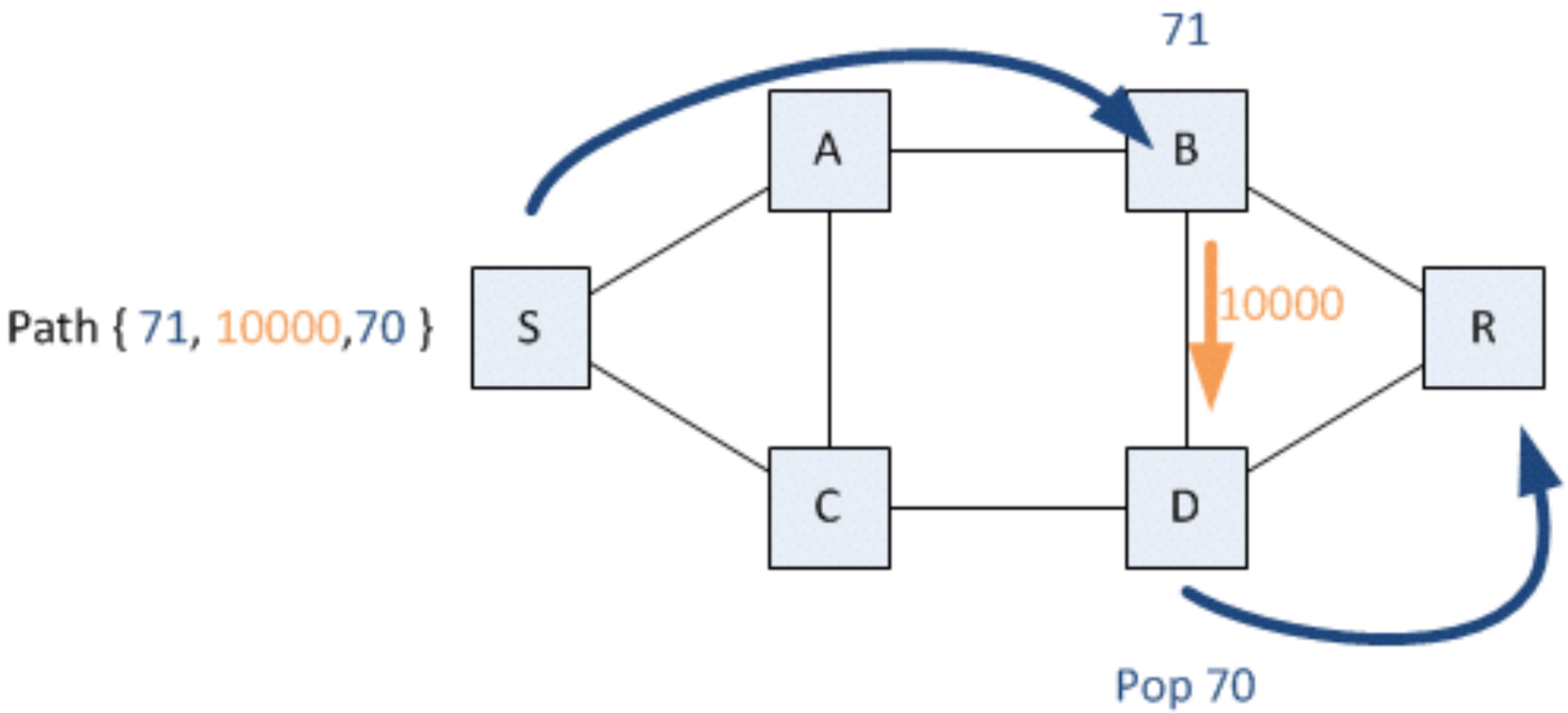


Use Case: NFaaS



Use Case: Segment Routing

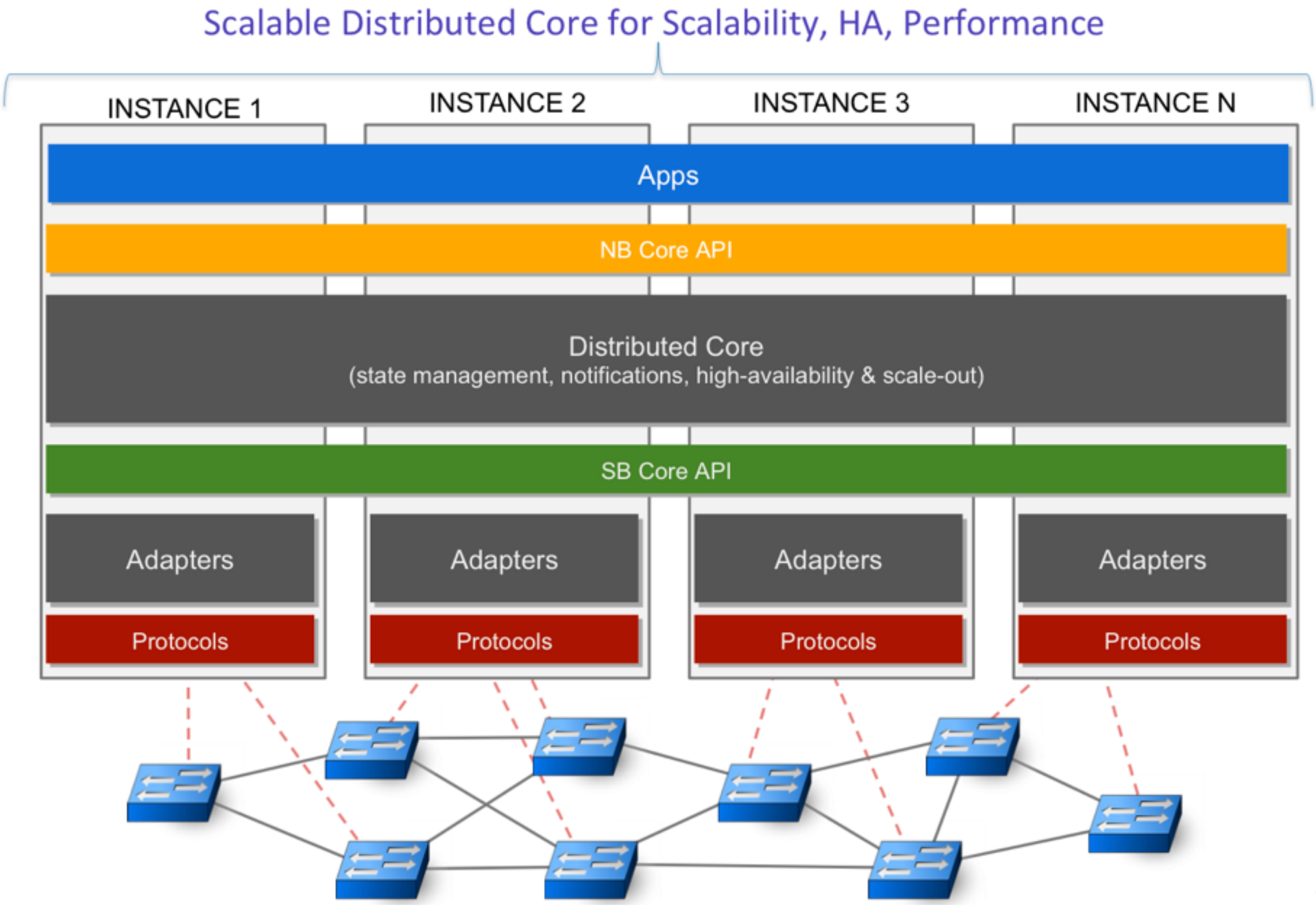
- Allows to enforce a flow through any **topological path** and service chain
  - Per-flow state is maintained only at the ingress node
- Can be directly applied to
  - **MPLS**, using labels
  - IPv6, using routing extension headers



Outline

- Introducing ONOS Blackbird
- **ONOS Architecture**
  - **Application Intent Framework**
  - **Distributed Core**
- Performance Evaluation
- How to write an ONOS application
- ONOS Toward IPv6

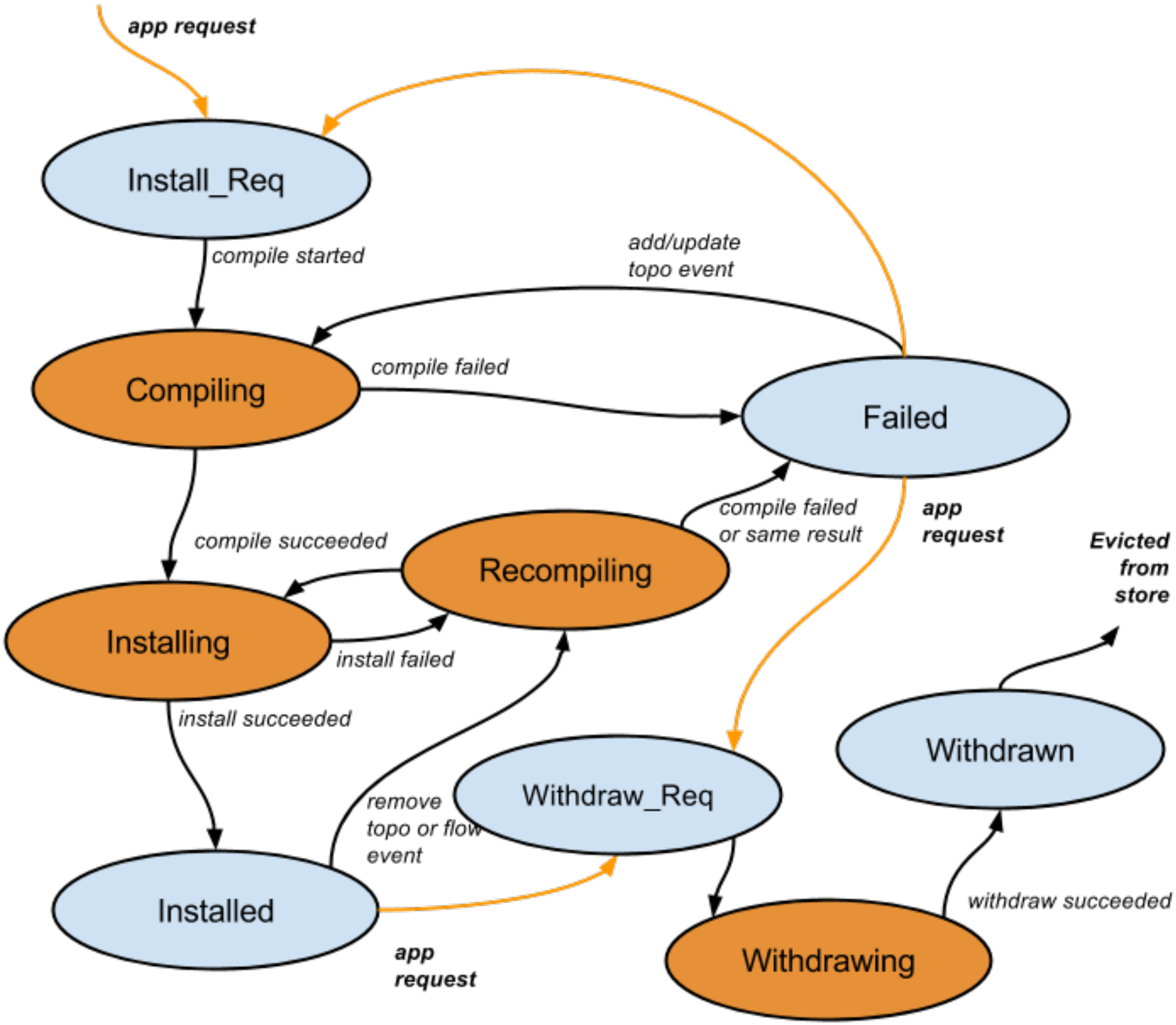
System Architecture



Application Intent Framework (1/3)

- Assign what to do (intent) instead of how to do (flow)
- Intent consists of
  - **Network Resource**, e.g. link
  - **Constraints**, e.g. bandwidth
  - **Criteria**, header fields or patterns that describe a slice of traffic
  - **Instruction**, e.g. header mod, output to port
- Intent can be compiled into other well-known intents by ***IntentCompiler***
  - *HostToHostIntent -> PathIntent*
- Intent can be converted into *BatchOperation* by ***IntentInstaller***
  - *PathIntent -> FlowRuleBatchOperations*

Application Intent Framework (2/3)



Application Intent Framework (3/3)

- Intent framework in Blackbird
  - No priority, first request first allocate
  - No conflict resolution
  - Will be in Cardinal
    - #2977: Add priority to remaining intent types
- Bandwidth constraint
  - Currently works in packet-optical networks only
  - Will be enforced when OVSDB adapter is finished

Distributed Core

- Mastership
  - None, Standby (Slave), Master
- Synchronization
  - Hazelcast (In-memory software data grid)
    - Distributed java.util.{Queue, Set, List, Map}
    - Distributed event and listener
    - Scale, fail-over…etc.
  - By default
    - Multicast 224.2.2.3:54327
  - Moving **from Hazelcast to RAFT**

Outline

- Introducing ONOS Blackbird
- ONOS Architecture
- **Performance Evaluation**
  - **Tested Hardware Switches**
  - **Flow Install Throughput**
  - **Intent Latency**
  - **Intent Throughput**
  - **Link Event Throughput**
  - **Port Event Throughput**
  - **Switch Event Throughput**
- How to write an ONOS application
- ONOS Toward IPv6

Tested Hardware Switch

- **Pica8 3290**
  - OpenFlow 1.0
  - Small office network
  - Reactive forwarding

Test Environment

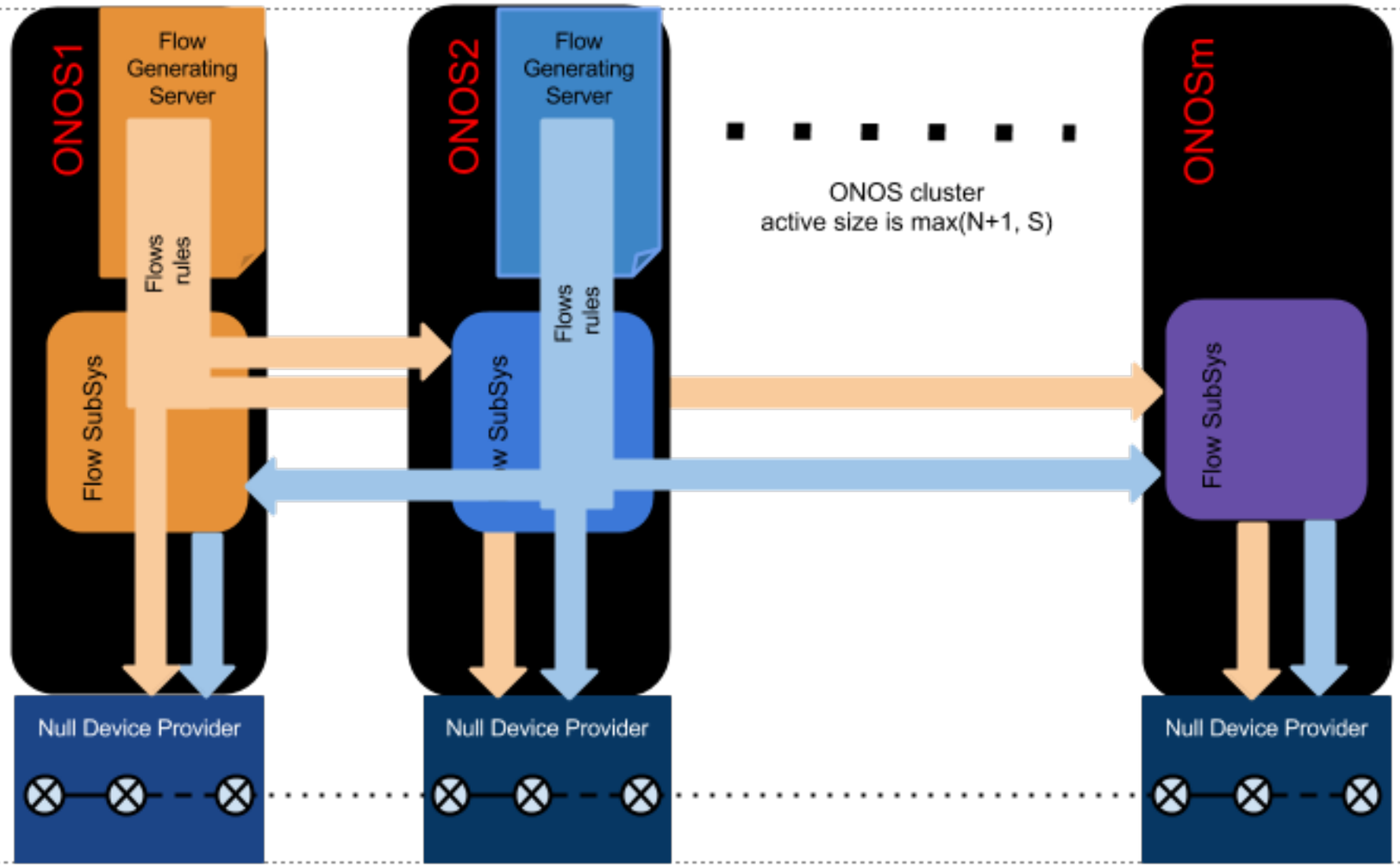
- Bare-metal controller
  - Xeon E5-2670 / 32G DDR3 RAM / SSD / 1Gbps NIC
  - JAVA_OPTS = -Xms8G -Xmx8G
- **NullProvider**
  - Fake switches, not even a virtual one

Flow Install Throughput - Test Plan

Burst flow installation test command: python3 flow-tester.py -f FL -n N -s [Servers], where
FL: the number of flows to be installed PER DEVICE;
[Servers]: list of "S" number of ONOS servers where flows are generated;
N: for the flows generated from each Server, the number of neighboring ONOS nodes where flows need to be sent to due to device mastership. ex. this diagram depicts S=2, and N=2.

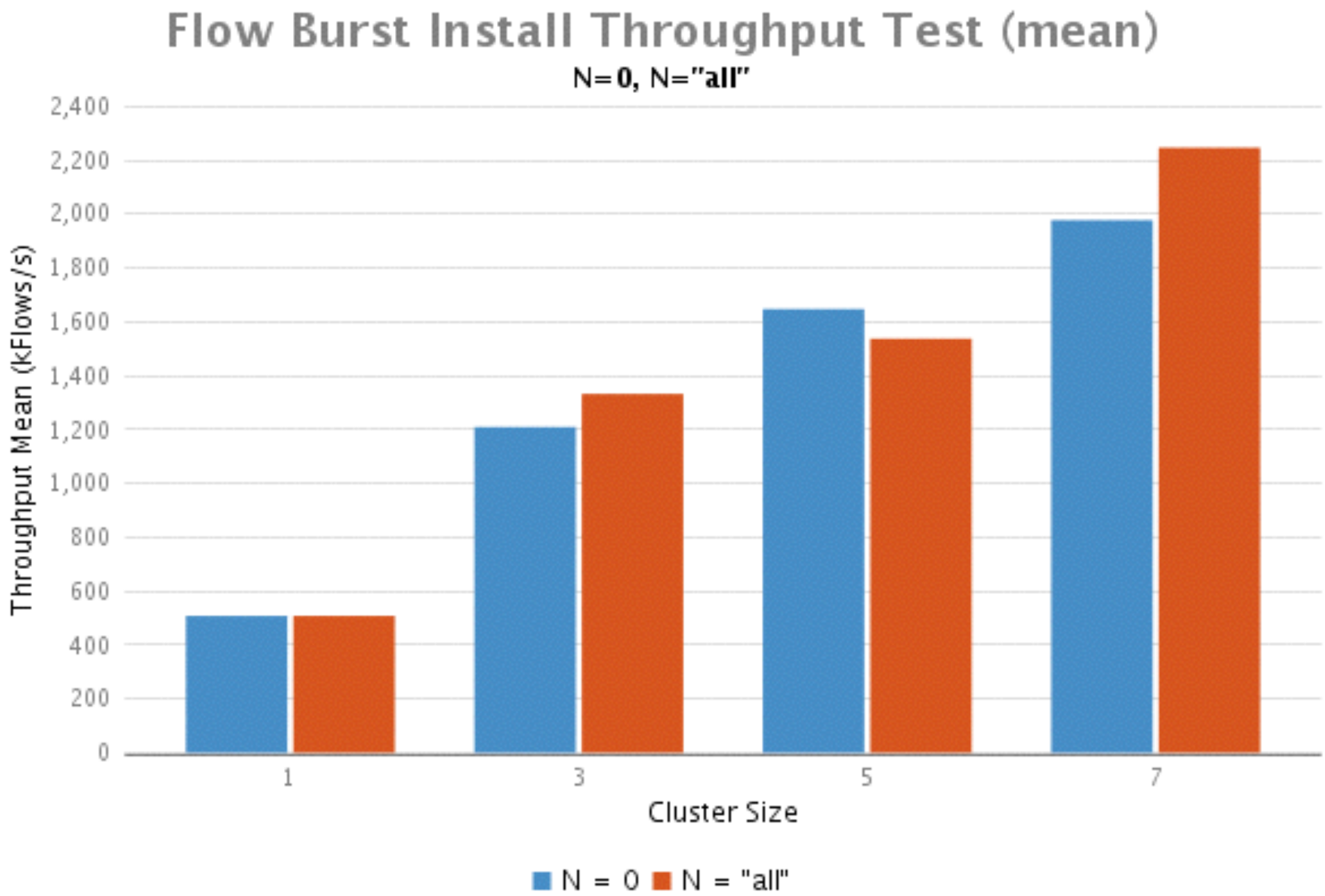
In order to make scale-out test results comparisons, total number of flows generated is fixed.

Total # of flows = S * FL * (N+1) * SW/max(N+1, S)



In order to make scale-out test results comparisons, the number of devices "SW" is fixed. Their mastership are "evenly" assigned to all active nodes in the cluster.

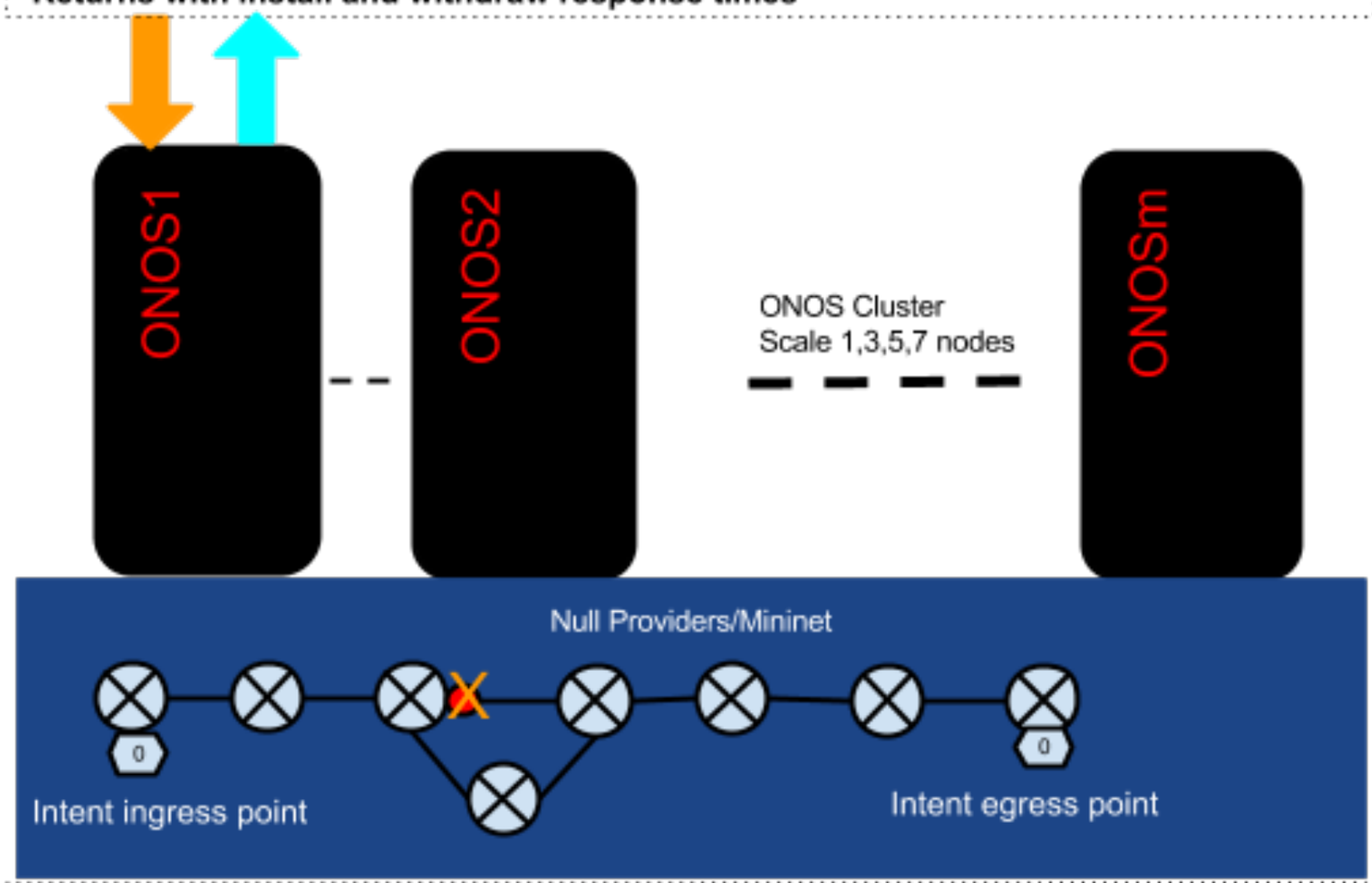
Flow Install Throughput - Test Result



- SW = **35** - total **# of switches** (Null Devices) connected to ONOS cluster evenly distributed to active ONOS nodes

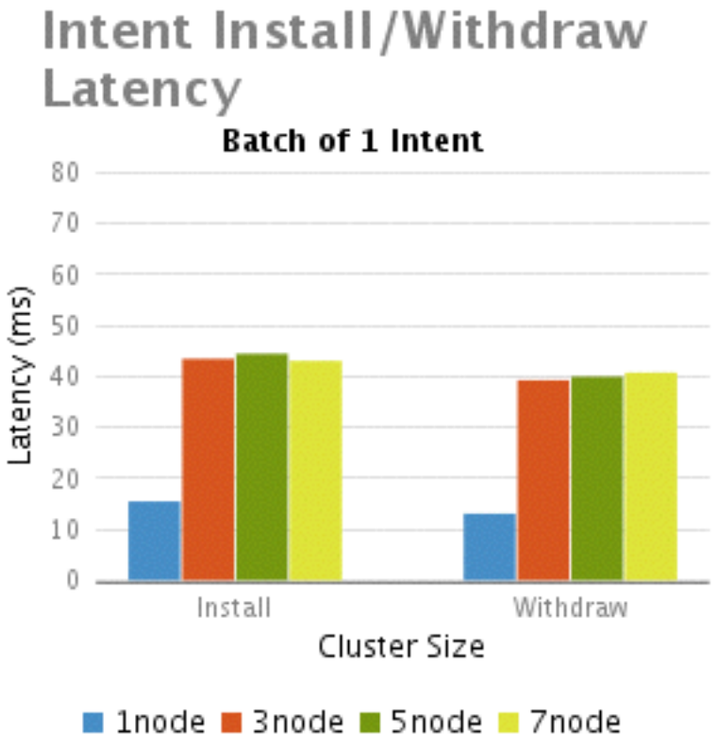
Intent Latency - Test Plan

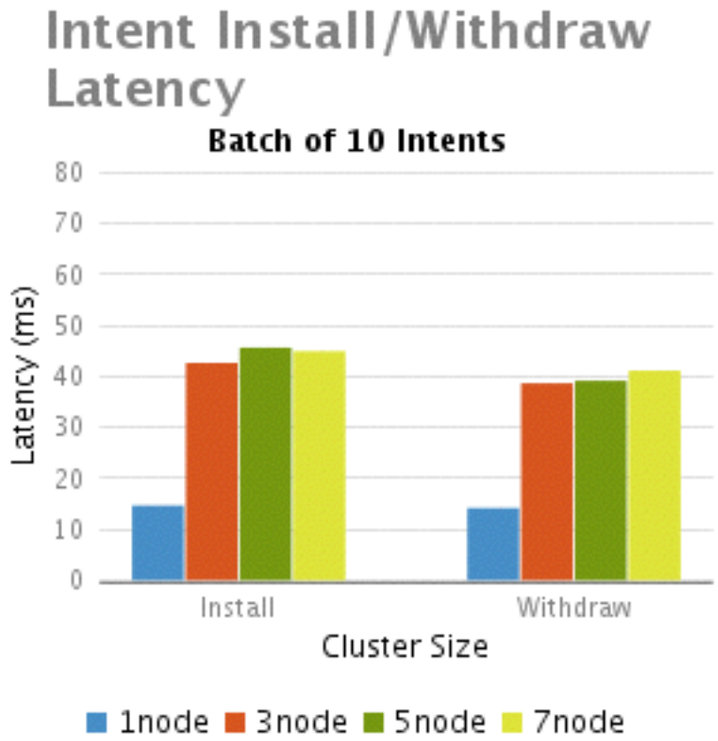
Cmd - “push-test-intents” to install, withdraw various sizes of intents;
Intents are end to end from first switch to last switch;
Returns with install and withdraw response times

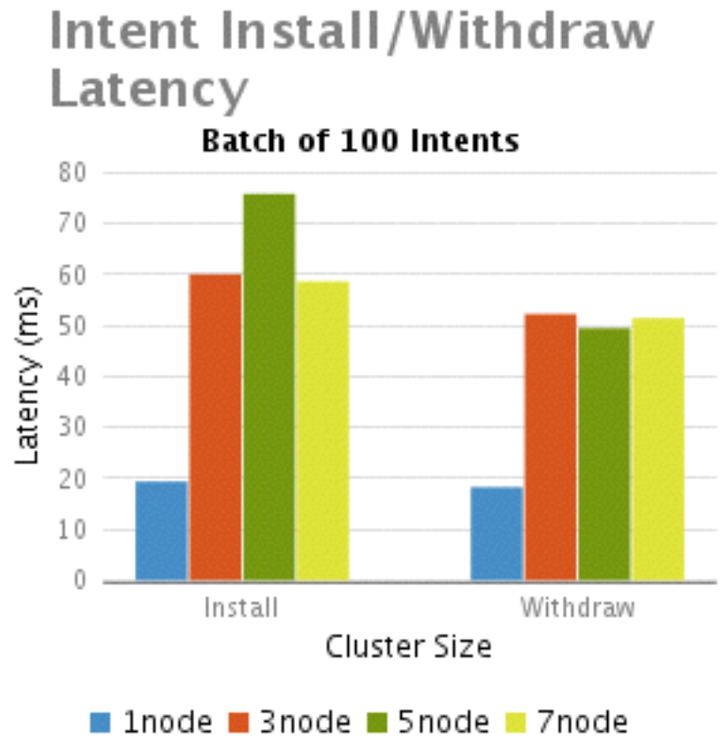


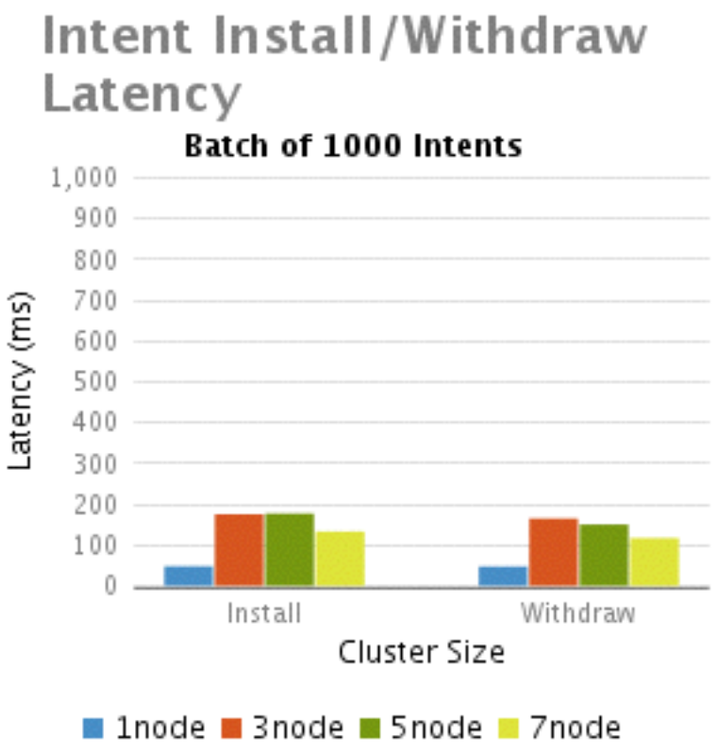
Seven (7) devices connected in a linear topology.
Device masterships are “evenly” assigned to all active nodes in the cluster.

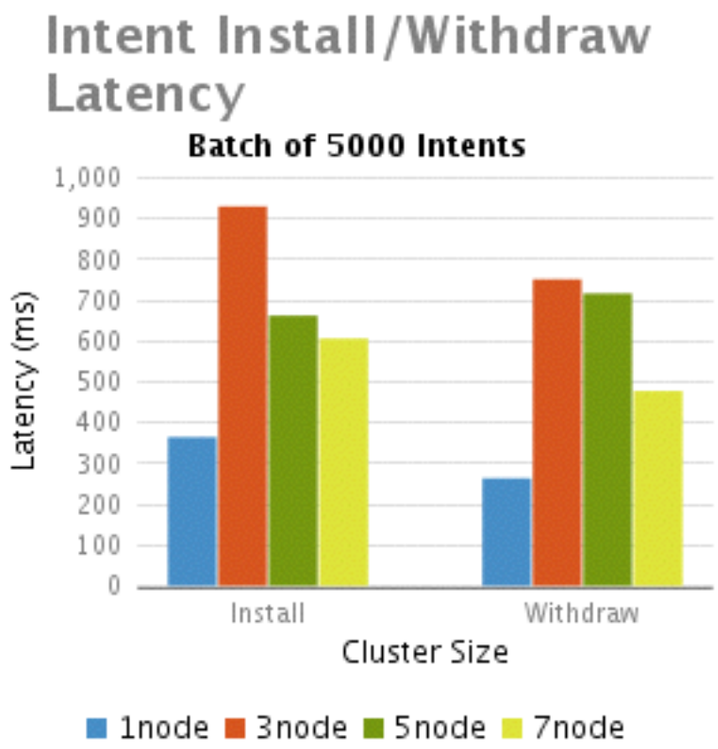
Intent Latency - Test Result







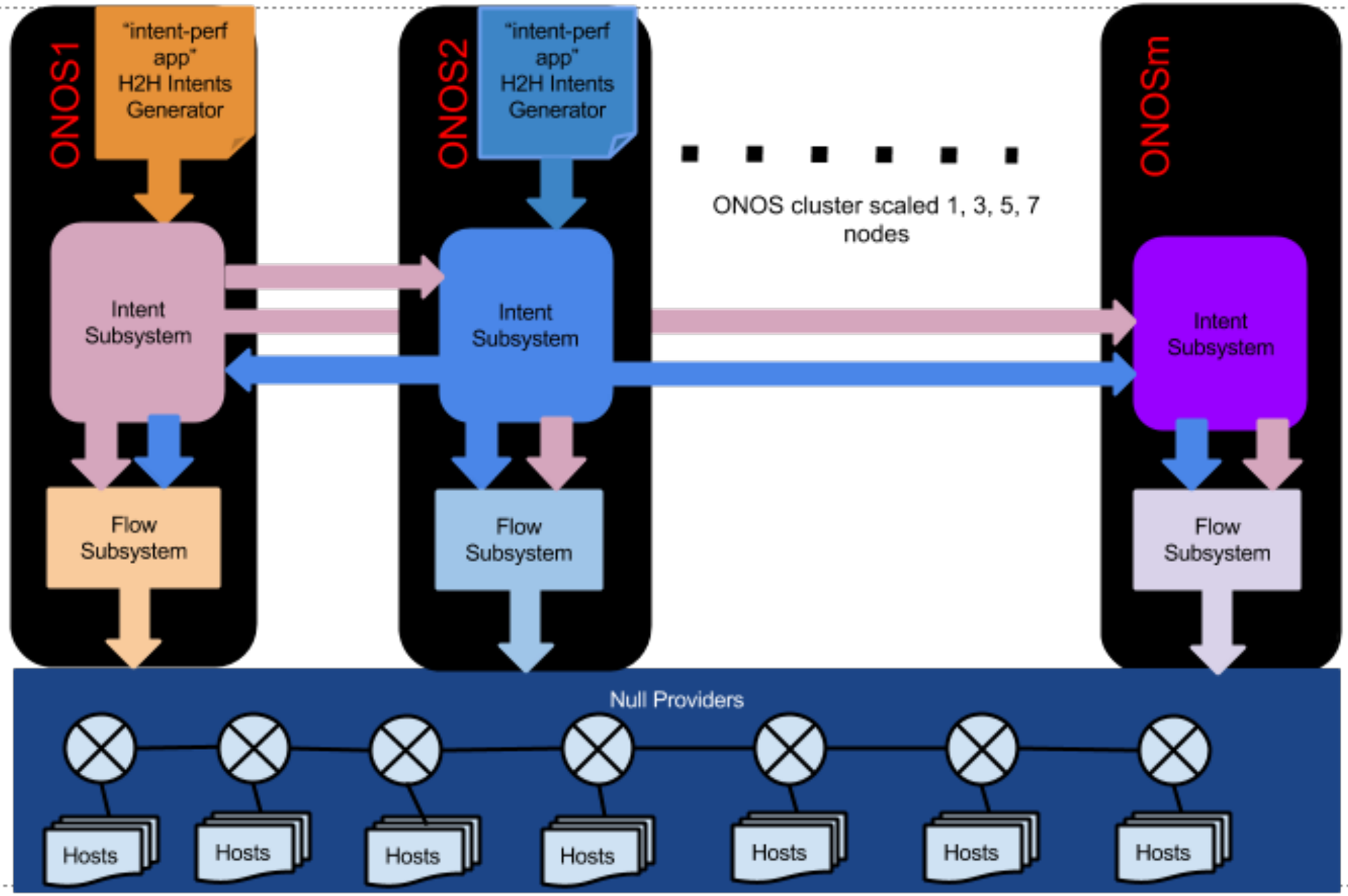




- 1 -> 3 node(s)
  - EW overhead
- >3 nodes
  - Large # intents (>1000)
    - Size ↑, latency ↓
  - Small # intents
    - Process overhead

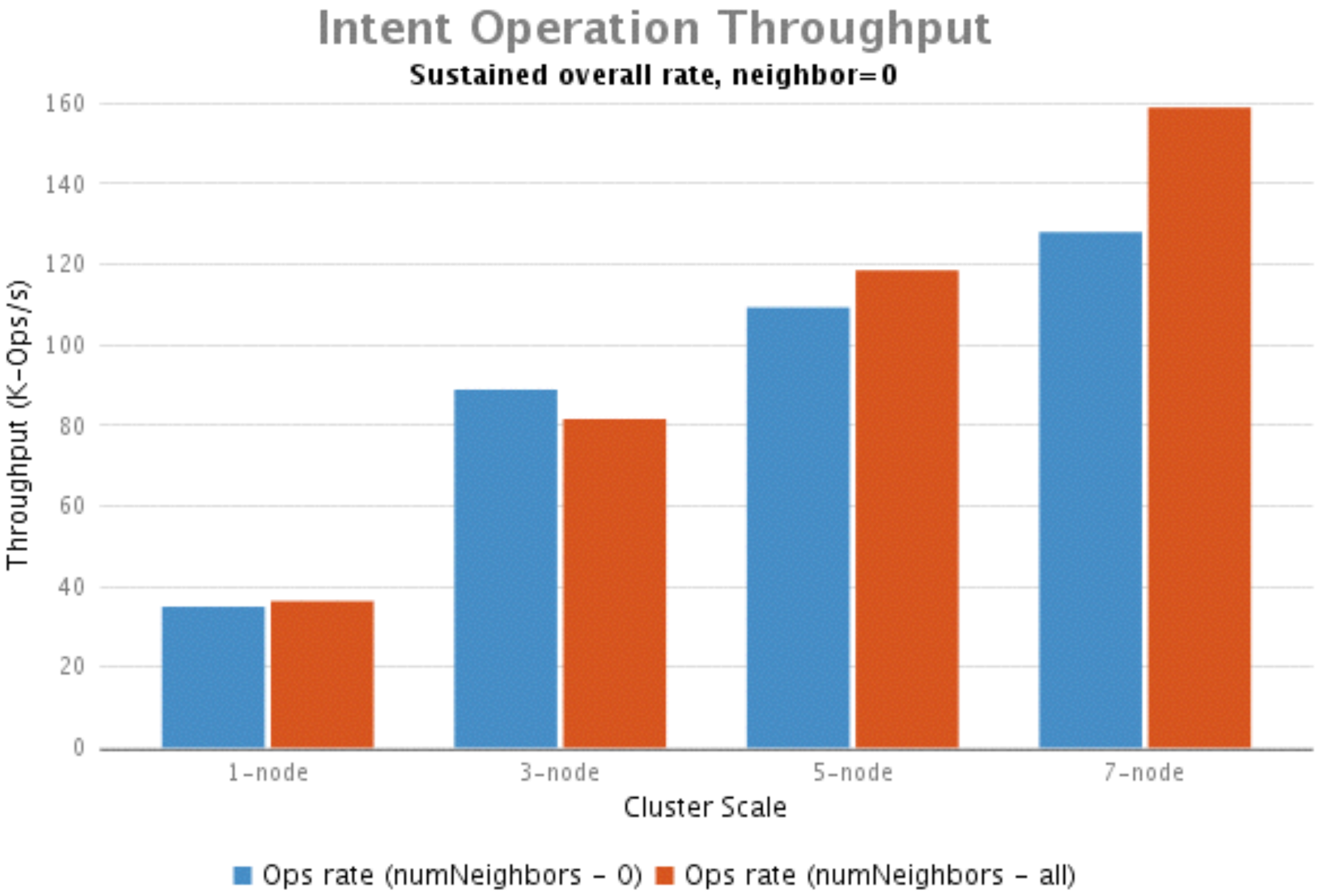
Intent Throughput - Test Plan

Intents generated are one-hop intents.
Neighboring pattern to show how many neighbors intents are propagating before reaching local flow subsystem.



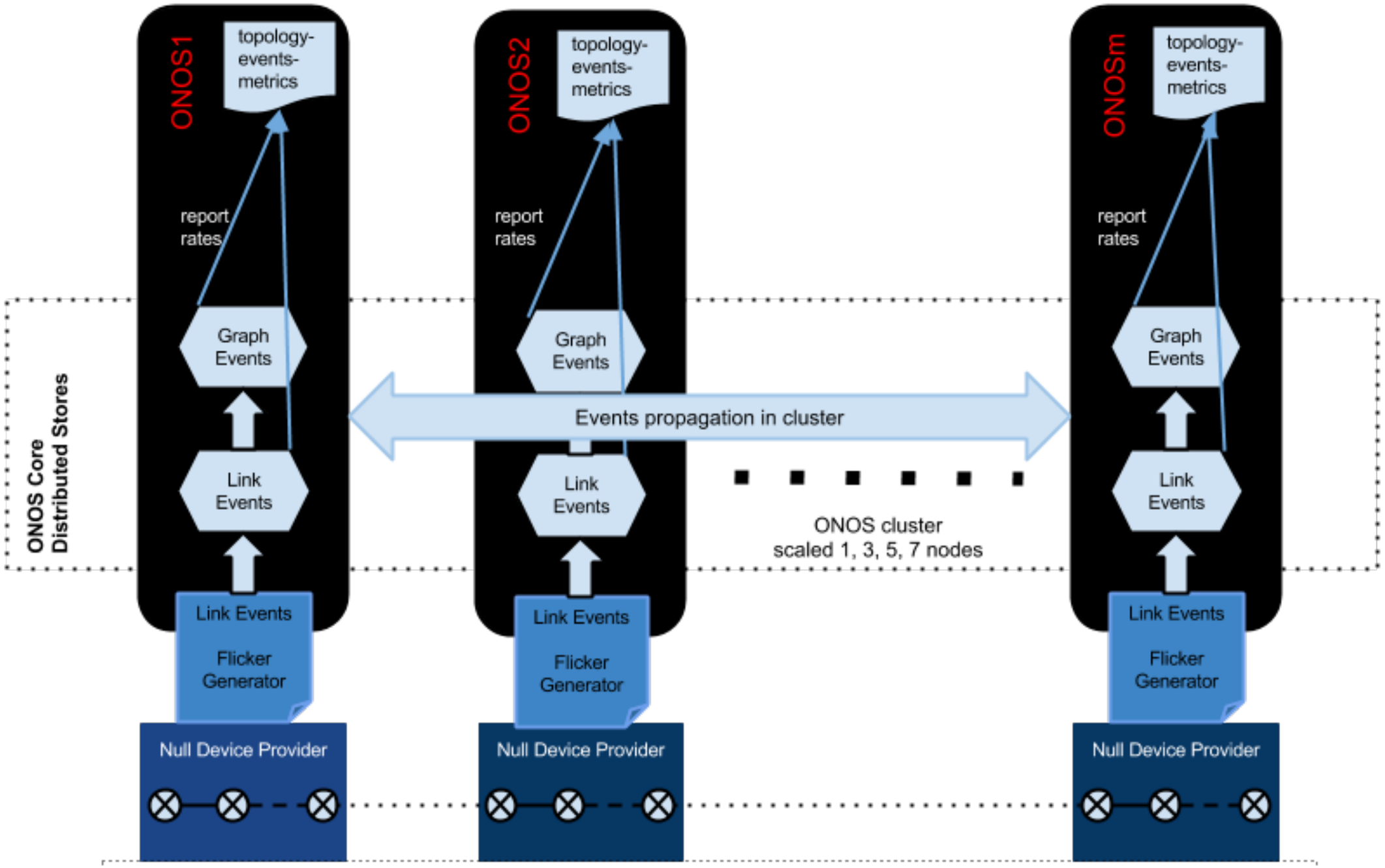
Seven (7) devices, each with a set of Hosts attached, connected in a linear topology. Device masterships are "evenly" assigned to all active nodes in the cluster.

Intent Throughput - Test Result



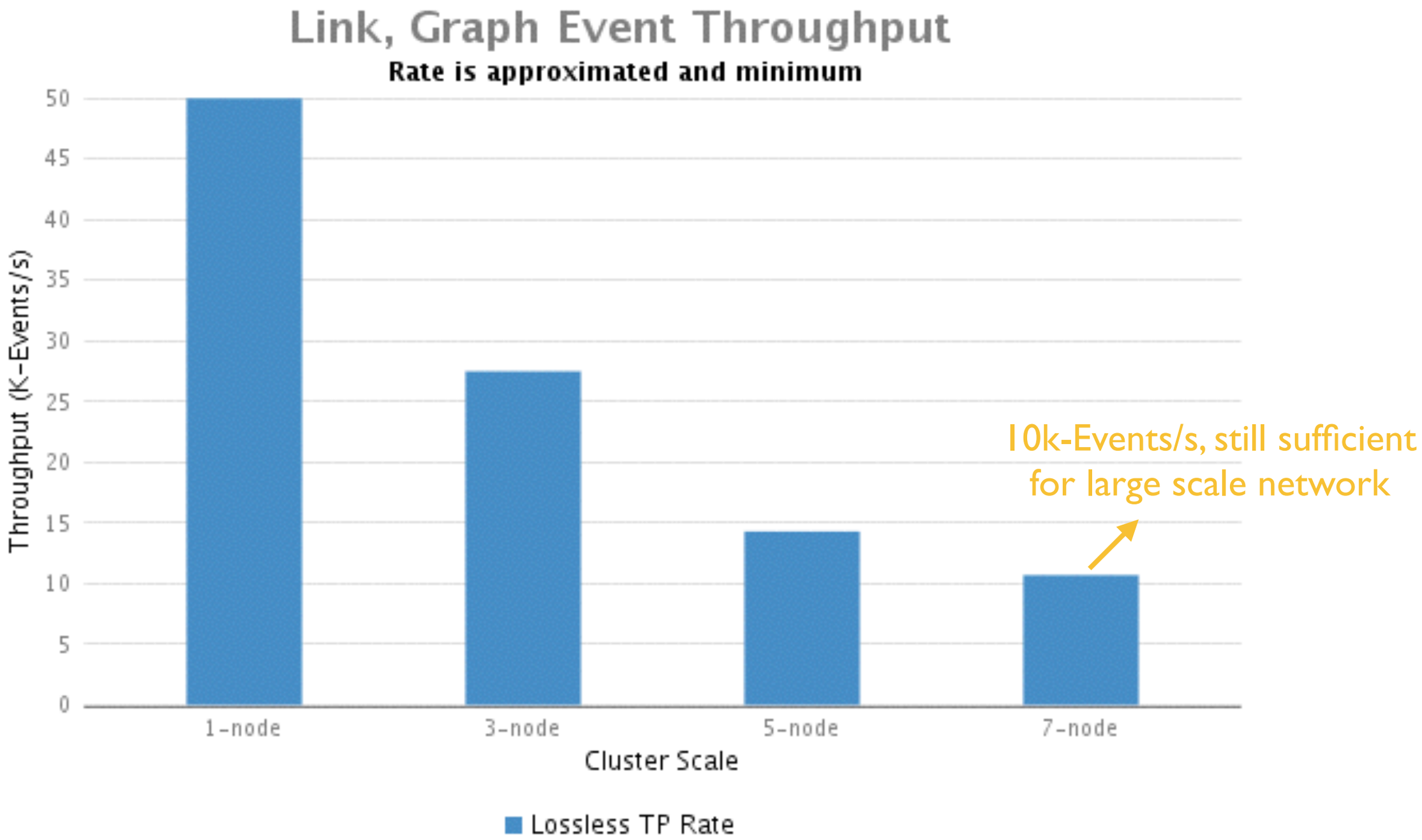
Link Event Throughput - Test Plan

Use “topology-events-metrics” to obtain time-averaged event throughput.



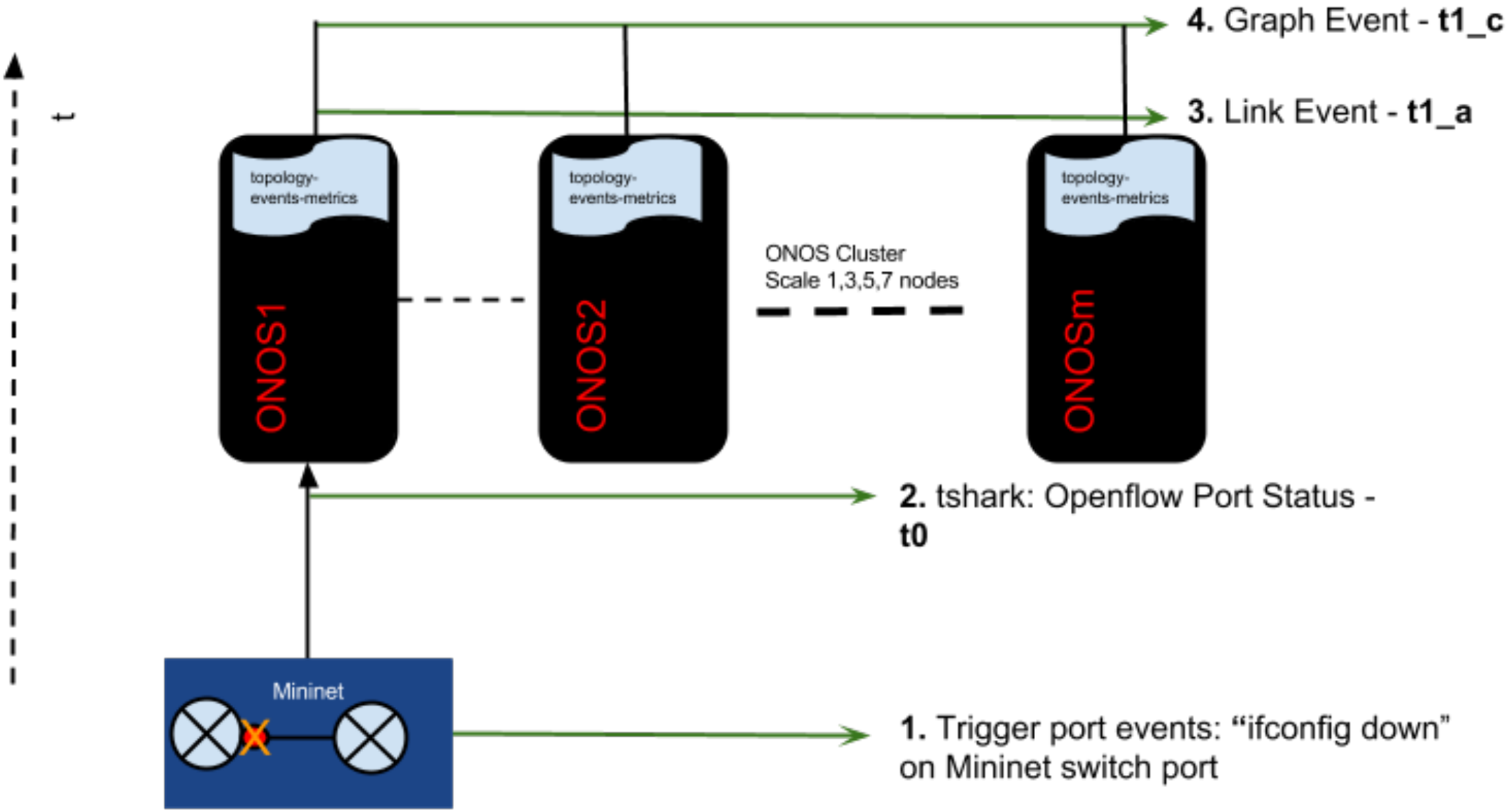
Load generated depends on the “eventRate” and number of Null links (each will start one thread of flicker @eventRate.

Link Event Throughput - Test Result

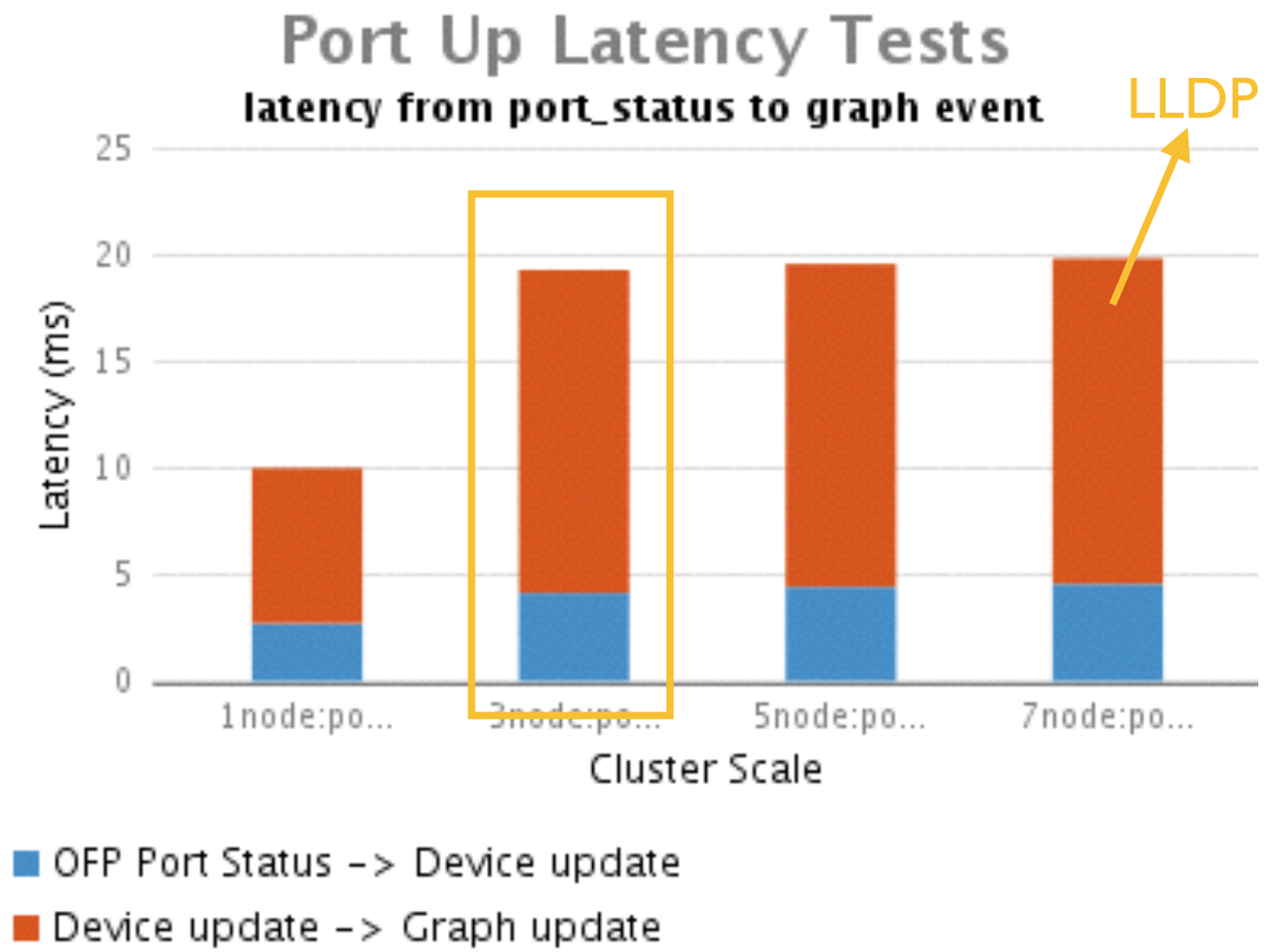


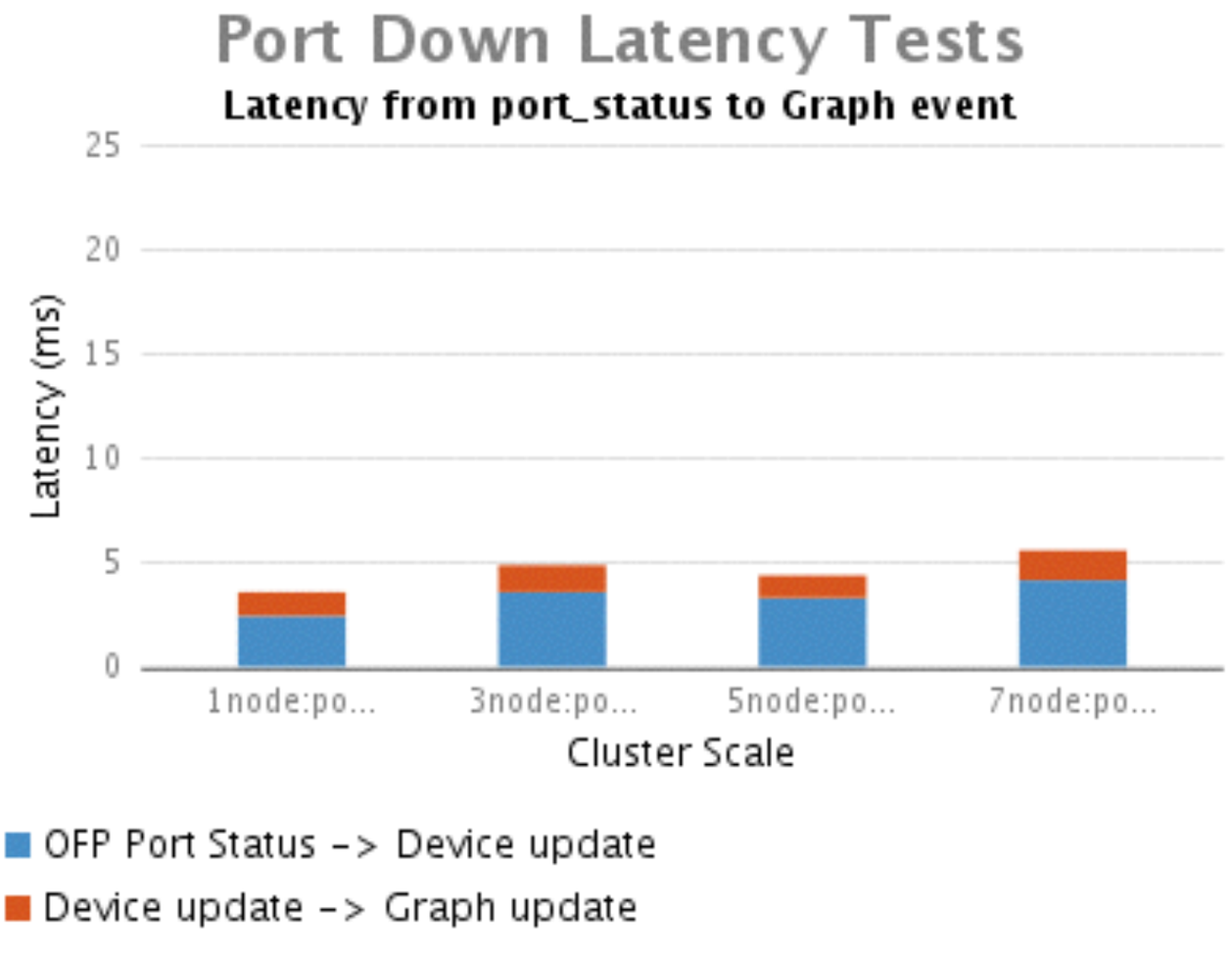
- **40** Null Devices (linear topology) on **each ONOS node**
- **32** effective **flicker threads**
- Flicker eventRate varies from **4000 to 500**

Port Event Latency - Test Plan



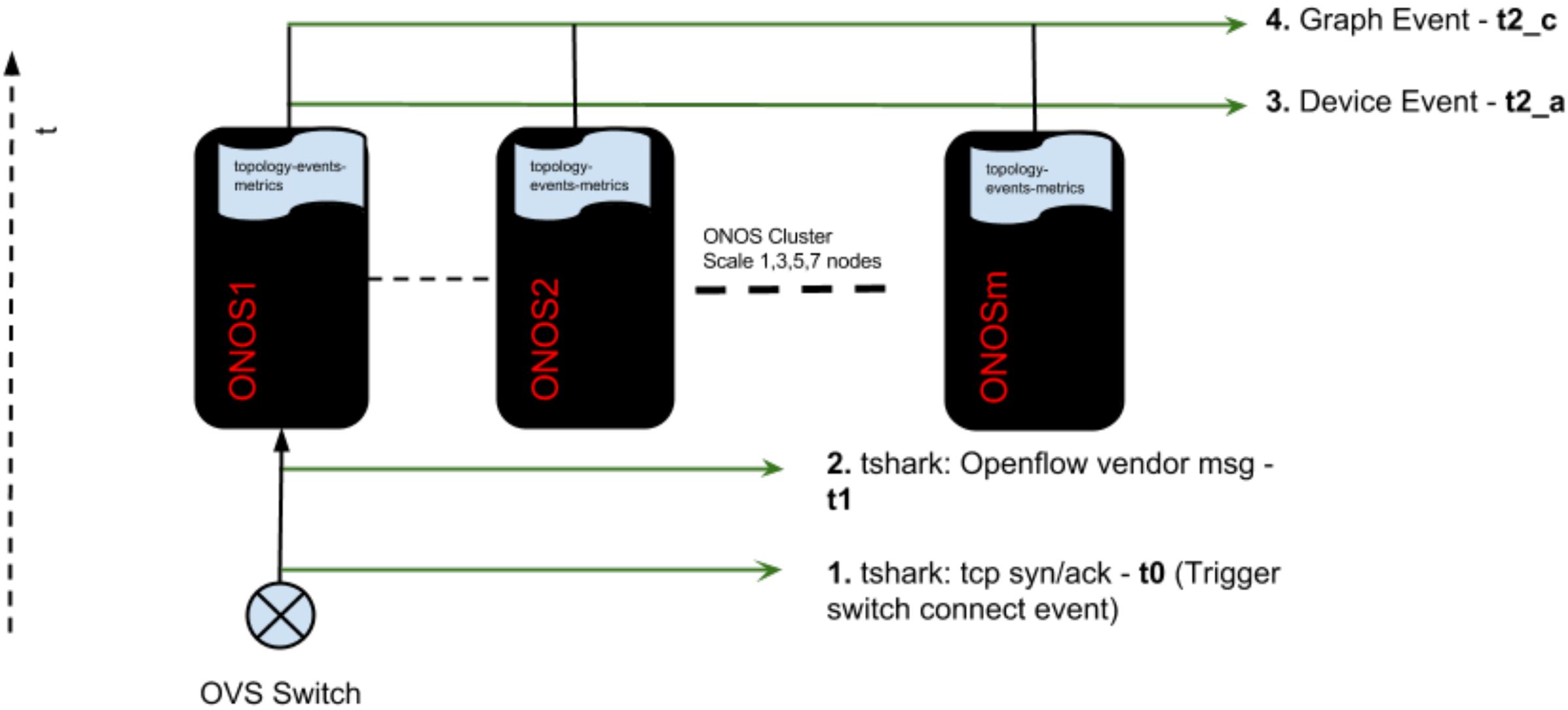
Port Event Latency - Test Result



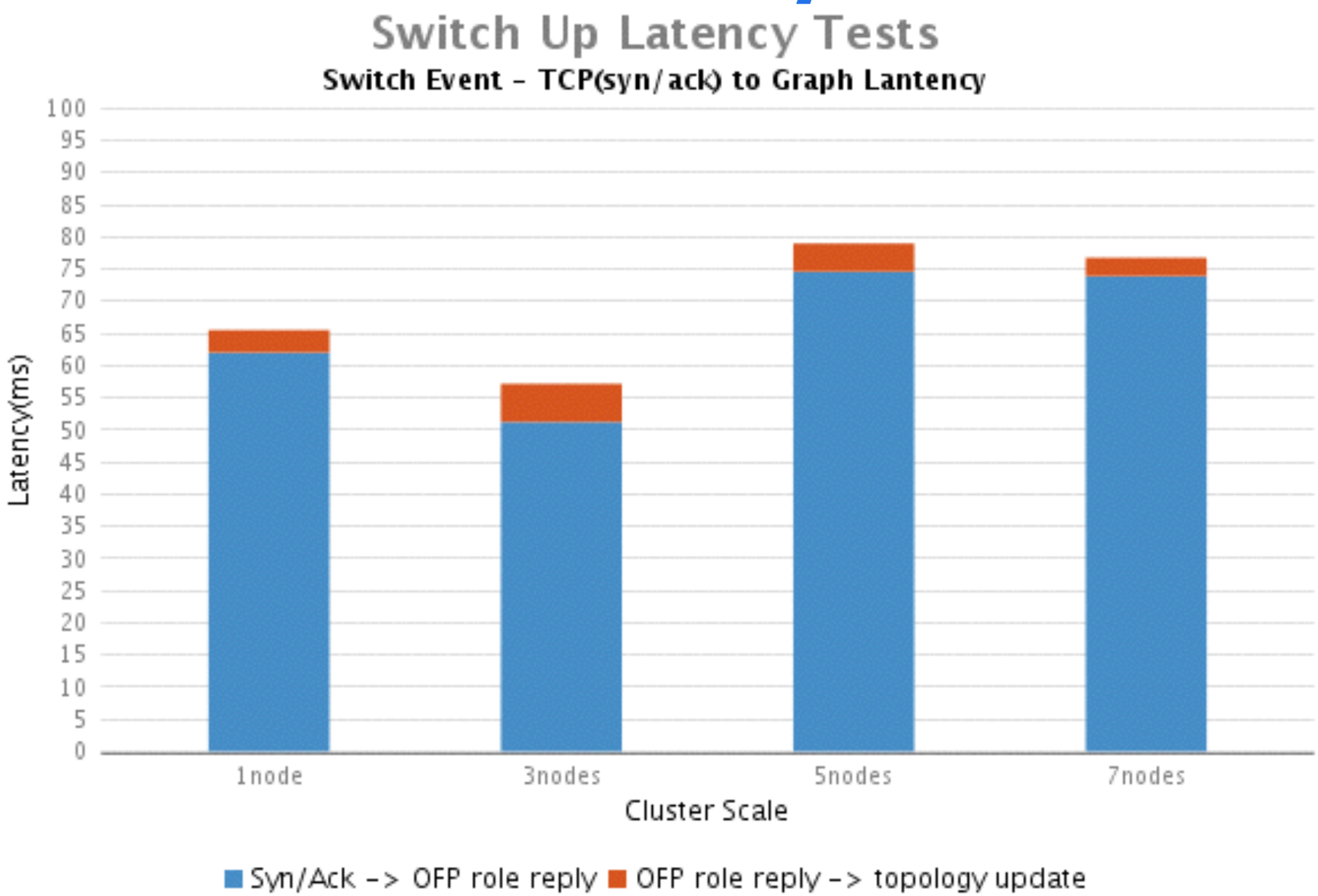


- **Port-up** timing breakdowns for a 3-node cluster
  - OFP of_port status -> complete Packet-In/Out for link discovery: **11 ms**
  - OFP of_port status -> device event: **5~11 ms**
  - complete Packet-In/Out for link discovery -> link event: **5~6 ms**
  - Link event -> graph event: **1~2 ms**

Switch Event Latency - Test Plan



Switch Event Latency - Test Result



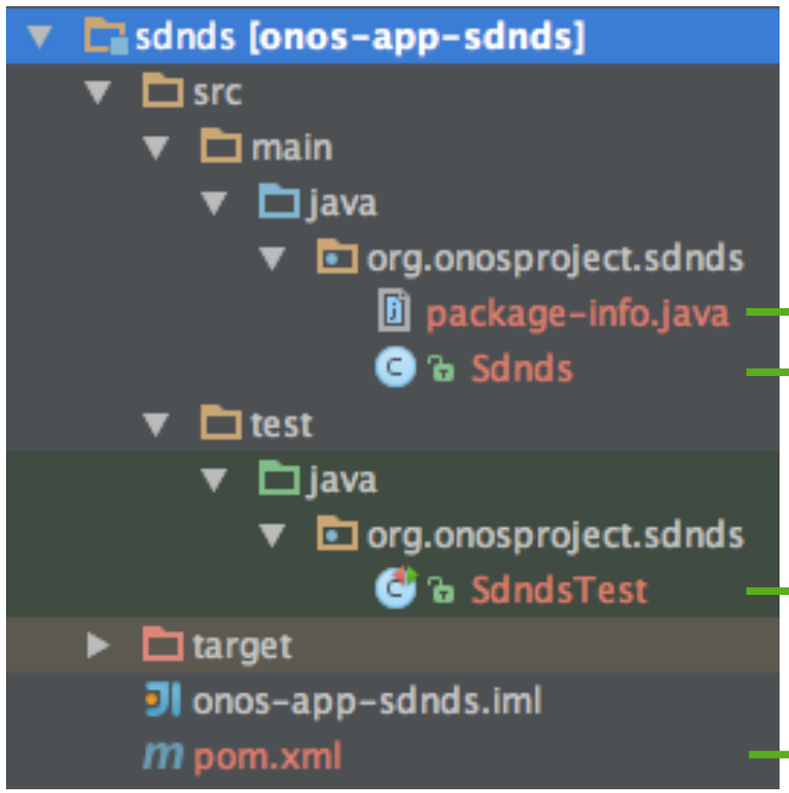
- **Syn/Ack -> OFP role reply** timing breakdowns for example of 58.1 ms
  - TCP syn -> OFP Hello (from ovs): **0.3 ms**
  - OFP Hello (from ovs) -> OFP of_features_request: **2.6 ms**
  - OFP of_features_request -> OFP of_features_reply: **47.0 ms**
  - OFP of_feature_reply -> OFP role_request: **8.0 ms**
  - OFP role_request -> OFP role_reply: **0.2 ms**

Outline

- Introducing ONOS Blackbird
- ONOS Architecture
- Performance Evaluation
- **How to write an ONOS application**
  - **Setup directory layout**
  - **Add pom.xml (app)**
  - **Edit pom.xml (parent)**
  - **Register application**
  - **Write application**
  - **Write unit test**
  - **Build application**
  - **Load application**
- ONOS Toward IPv6

Setup Directory Layout

```
cd ${ONOS_ROOT}
mkdir -p apps/sdnds
mkdir -p apps/sdnds/src/main/java/org/onosproject/sdnds
mkdir -p apps/sdnds/src/test/java/org/onosproject/sdnds
```



package-wide documentation/annotation

application goes here

unit test goes here

Project Object Model (app)

Add pom.xml (/apps/sdnds/pom.xml)

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.onosproject</groupId>
        <artifactId>onos-apps</artifactId>
        <version>1.1.0-SNAPSHOT</version>
        <relativePath>../pom.xml</relativePath>
    </parent>

    <artifactId>onos-app-sdnds</artifactId>
    <packaging>bundle</packaging>

    <description>Demo application for SDNDS</description>

    <dependencies>
        <dependency>
            <groupId>org.osgi</groupId>
            <artifactId>org.osgi.compendium</artifactId>
        </dependency>

        <dependency>
            <groupId>org.onosproject</groupId>
            <artifactId>onlab-junit</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

for unit test

Edit pom.xml (/apps/pom.xml)

```
<modules>
    <module>tvue</module>
    <module>fwd</module>
    <module>ifwd</module>
    <module>mobility</module>
    <module>proxyarp</module>
    <module>config</module>
    <module>sdnip</module>
    <module>calendar</module>
    <module>optical</module>
    <module>metrics</module>
    <module>oecfg</module>
    <module>demo</module>
    <module>election</module>
    <module>routing</module>
    <module>routing-api</module>
    <module>bgprouter</module>
    <module>intent-perf</module>
    <module>database-perf</module>
    <module>sdnds</module>
</modules>
```

Add app.xml (/apps/sdnds/app.xml)

```xml
<?xml version="1.0" encoding="UTF-8"?>

<app name="org.onosproject.app.sdnds" origin="SDNDS" version="1.1.0"
        features="onos-app-sdnds">
    <description>Demo app for SDNDS</description>
</app>
```

Register Application (/features/features.xml)

```
<feature name="onos-app-sdnds" version="@FEATURE-VERSION"
         description="Demo app for SDNDS">
    <feature>onos-api</feature>
    <bundle>mvn:org.onosproject/onos-app-sdnds/@ONOS-VERSION</bundle>
</feature>
```

Write Application (1/5)

- package and imports

```
1   package org.onosproject.sdnds;
2
3   import org.apache.felix.scr.annotations.Activate;
4   import org.apache.felix.scr.annotations.Component;
5   import org.apache.felix.scr.annotations.Deactivate;
6   import org.apache.felix.scr.annotations.Reference;
7   import org.apache.felix.scr.annotations.ReferenceCardinality;
8   import org.onlab.packet.Ethernet;
9   import org.onlab.packet.IPacket;
10  import org.onlab.packet.IPv4;
11  import org.onosproject.core.ApplicationId;
12  import org.onosproject.core.CoreService;
13  import org.onosproject.net.flow.DefaultTrafficSelector;
14  import org.onosproject.net.flow.TrafficSelector;
15  import org.onosproject.net.packet.InboundPacket;
16  import org.onosproject.net.packet.PacketContext;
17  import org.onosproject.net.packet.PacketPriority;
18  import org.onosproject.net.packet.PacketProcessor;
19  import org.onosproject.net.packet.PacketService;
20  import org.osgi.service.component.ComponentContext;
21  import org.slf4j.Logger;
22
23  import static org.slf4j.LoggerFactory.getLogger;
24
```

Write Application (2/5)

- @Reference and @Activate

```
25  /**
26   * Demo application for SDNDS.
27   */
28  @Component(immediate = true)
29  public class Sdnds {
30      private ReactivePacketProcessor processor = new ReactivePacketProcessor();
31      private ApplicationId appId;
32      private final Logger log = getLogger(getClass());
33
34      @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
35      protected CoreService coreService;
36
37      @Reference(cardinality = ReferenceCardinality.MANDATORY_UNARY)
38      protected PacketService packetService;
39
40      @Activate
41      public void activate(ComponentContext context) {
42          appId = coreService.registerApplication("org.onosproject.sdnds");
43
44          packetService.addProcessor(processor, PacketProcessor.OBSERVER_MAX);
45
46          TrafficSelector.Builder selector = DefaultTrafficSelector.builder();
47          selector.matchEthType(Ethernet.TYPE_IPV4);
48          packetService.requestPackets(selector.build(), PacketPriority.REACTIVE,
49                  appId);
50          selector.matchEthType(Ethernet.TYPE_ARP);
51          packetService.requestPackets(selector.build(), PacketPriority.REACTIVE,
52                  appId);
53
54          log.info("Started with Application ID {}", appId.id());
55      }
```

(prior)
↓ Advisor
↓ Director
↓ Observer
(subsequent)

ask for packet in
default: drop

Write Application (3/5)

- @Deactivate

```
57      @Deactivate
58      public void deactivate() {
59          packetService.removeProcessor(processor);
60          processor = null;
61          log.info("Stopped");
62      }
63
```

Write Application (4/5)

- Packet processor

```
64      /**
65       * Packet processor prints all packet_in packets.
66       */
67      private class ReactivePacketProcessor implements PacketProcessor {
68
69          @Override
70          public void process(PacketContext context) {
71              // Stop processing if the packet has been handled, since we
72              // can't do any more to it.
73              /*
74              if (context.isHandled()) {
75                  return;
76              }
77              */
78
79              InboundPacket pkt = context.inPacket();
80              Ethernet ethPkt = pkt.parsed();
81              IPv4 ipv4 = isIPv4(ethPkt);
82              if (ipv4 != null) {
83                  log.info("IPv4, {}->{}",
84                           ipv4.getSourceAddress(),
85                           ipv4.getDestinationAddress()
86                  );
87              }
88          }
89      }
90
```

Write Application (5/5)

- Helper class (optional)

```java
    /**
     * Determine if the packet is IPv4.
     *
     * @param ethPkt the packet to be checked
     * @return the IPv4 packet if true, else return null
     */
    public static IPv4 isIPv4(Ethernet ethPkt) {
        if (ethPkt != null) {
            IPacket ipkt = ethPkt.getPayload();
            if (ipkt != null) {
                if (ipkt instanceof IPv4) {
                    return (IPv4) ipkt;
                }
            }
        }
        return null;
    }
}
```

Write Unit Test

```java
package org.onosproject.sdnds;

import org.junit.Test;
import org.onlab.packet.Ethernet;
import org.onlab.packet.IPv4;
import org.onlab.packet.IPv6;

import static org.junit.Assert.assertTrue;

/**
 * Unit test for Sdnds.
 */
public class SdndsTest {
    /**
     * Tests isIPv4.
     */
    @Test
    public void testIsIpv4() {
        Ethernet ethpkt = new Ethernet();
        IPv4 ipv4 = new IPv4();
        IPv6 ipv6 = new IPv6();

        ethpkt.setPayload(ipv4);
        assertTrue(Sdnds.isIPv4(ethpkt) instanceof IPv4);

        ethpkt.setPayload(ipv6);
        assertTrue(Sdnds.isIPv4(ethpkt) == null);

    }
}
```

Build Application

```
alias ob='onos-build'
alias obd='onos-build-docs'
alias obi='onos-build -Dmaven.test.failure.ignore=true'
alias obs='onos-build-selective'

alias op='onos-package'

onos-install
onos-install -nf (OS X)
```

Load Application

```
/opt/onos/apache-karaf-3.0.2/bin/karaf clean
onos> feature:install onos-app-sdnds

alias ol='onos-log'
```

Outline

- Introducing ONOS Blackbird
- ONOS Architecture
- Performance Evaluation
- How to write an ONOS application
- **ONOS Toward IPv6**

IPv6 Support in ONOS

- First community-driven feature
  - Community did the planning, development and testing
- No meeting
  - All coordination are done by email / jira / gerrit

IPv6 Status - Blackbird

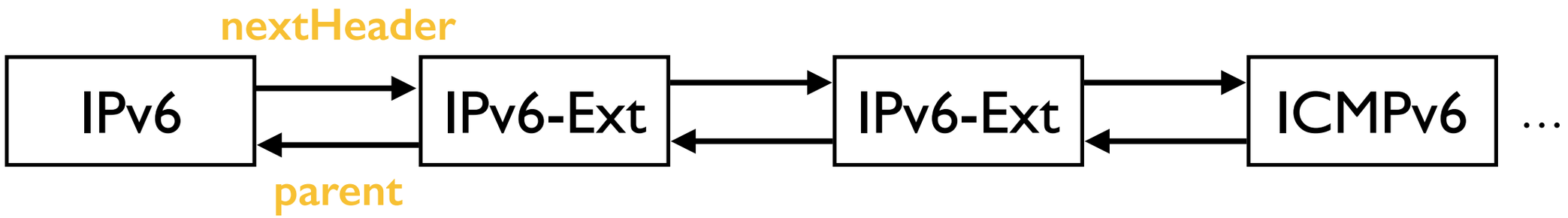
- Use cases
  - SDN-IP
    - **Exchanging IPv6 routing** information with BGP routers
  - Reactive forwarding
    - **Forward IPv6 packets** in SDN
- Experimental feature
  - Without Q/A approval

ONOS IPv6 Task Force

- **Charles M.C. Chan**
  - Ph.D. student, National Chiao Tung University, Taiwan
  - Initial planning and development
- **Kunihiro Ishiguro**
  - Co-founder, IPInfusion
  - Development and IPv6 tutorial script
- **Dusan Pajin**
  - Network engineer, Academic Network of Serbia
  - Testing and development
- **Pavlin Radoslavov**
  - (Former) Member of Technical Staff, ON.Lab
  - Coordinator and leftover tasks processor

What Have Been Done (1/4)

- Packet **serializer / deserializer**
- Why
  - Need the class to parse IPv6 packet header
    - E.g. source IP, destination IP
- Challenge
  - Extension headers
    - Treated as upper layer header



  - Upper layer checksum
    - Pseudo header (TODO)

What Have Been Done (2/4)

- Criteria, Selector, Treatment
- Why
  - To support IPv6-related matching and actions

- **Criteria**
  - Matching fields
    - E.g. src_ipv6, dst_ipv6
- **Selector**
  - Matches
    - E.g. src_ipv6=fe80::1, src_mac=00:00:00:00:00:01
- **Treatment**
  - Actions
    - E.g. set_dst_ipv6=fe80::2, output=3

What Have Been Done (3/4)

- **Neighbor Discovery Protocol** (NDP)
- Why
  - Similar to ARP in IPv4
  - Need to parse NDP contents
    - E.g. Link-layer address

What Have Been Done (4/4)

- Host service
- Why
  - Need to track / monitor the **location of IPv6 hosts**
- IPv4 / IPv6
  - Update location
- ARP / NDP
  - Update location and IP/mac mapping

Future Work - Cardinal

- Expose IPv6 intents in CLI / REST
- SDN-IP: verify receiving of IPv6 routes over IPv6 peering
- More testing and bug fixes
- Obtain Q/A approval

Demo

It’s show time!

Thank you!

Q&A

http://wiki.onosproject.org

You can find almost everything here