Path recovery behavior comparison between ODL and ONOS controllers

Speaker: Diamanti Alessio

Contributors: Alessio Diamanti (Orange/Cnam), José Manuel Sanchez Vilchez (Orange), Mamadou Tahirou Bah (LIP6)

June 17, 2019

Agenda

- Introduction
- Topology update reactivity
  - Target topology
  - Link up event controller reaction time
  - Variability in reaction times
- Topology discovery traffic
  - LLDP traffic volume
- Conclusion and future work

Introduction

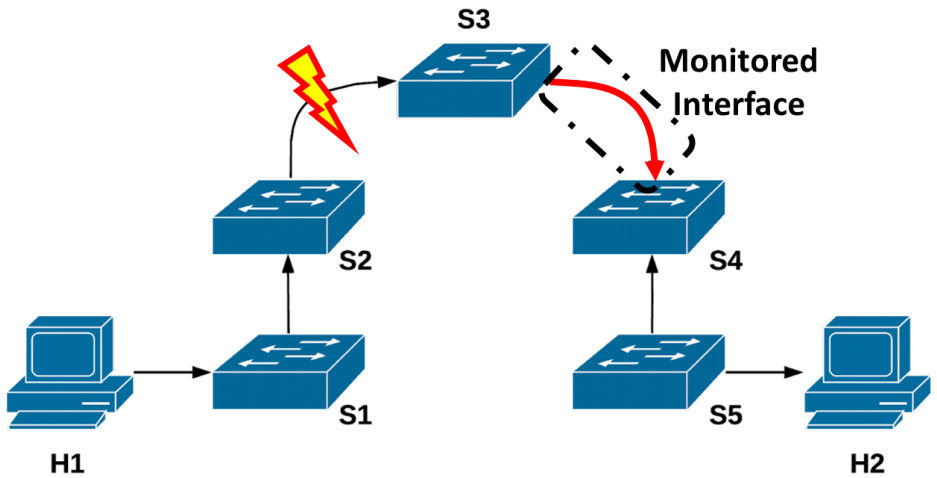
Topology discovery and update are implemented through an event-driven publish-subscriber pattern:

- **First discovery** is solicited by Openflow switches (`OFPT_HELLO`);
- The controller periodically checks network state through **LLDP** protocol;
- Switches notify **link disruption/establishment** through `OFPT_PORT_STATUS`

Event's subscribers update network topology representation in a distributed store. Finally, path computation applications react to store representation changes.

Target topology

Target topology was deployed through a **developed python module** able to inject faults and degradations on each of the simulated network elements(not Mininet-based). Two hosts connected by a single path exchanging UDP packets through an **Iperf** session.



- ONOS Quali and ODL Oxygen;
- ***org.onosproject.fwd*+ *org.onosproject.openflow*** for ONOS and ***odl-l2switch-all*** ODL

Reaction time computation

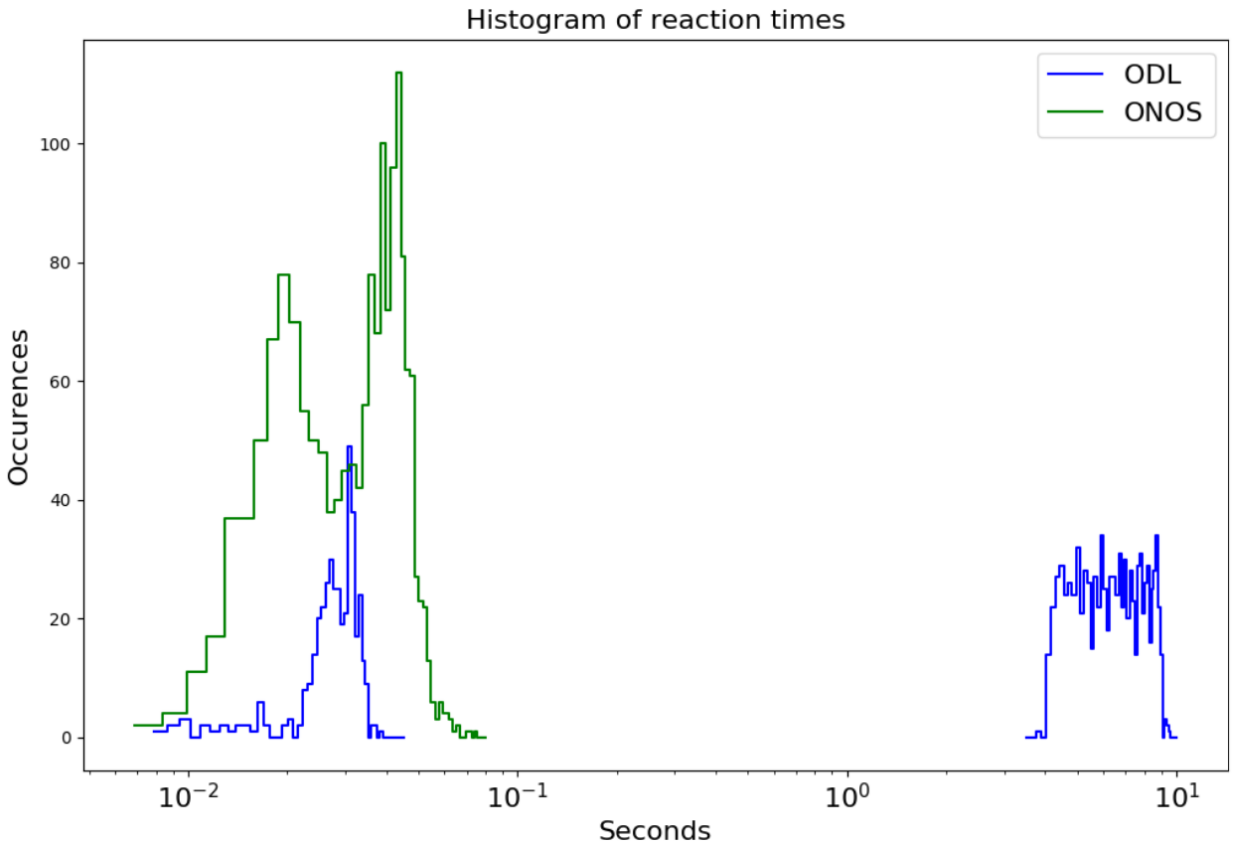
To gather statistically sound data, we performed 1400 iteration of the following steps:

- At $\mathbf{t_0}$ UDP Iperf session is started. H1 is client and H2 the server
- **Tshark** captures packet on link S3-S4
- At $\mathbf{t_0} + \tau = \mathbf{T_{start}}$ the link S2-S3 fails
- After T seconds the link is restored$\rightarrow \mathbf{T_{stop}} = \mathbf{T_{start}} + \mathbf{T}$
- First packet on S3-S4 after $\mathbf{T_{start}} + \mathbf{T}$ is recorded $\rightarrow \mathbf{T_{first}} = \mathbf{T_{stop}} + \mathbf{T_{react}}$

Reaction time

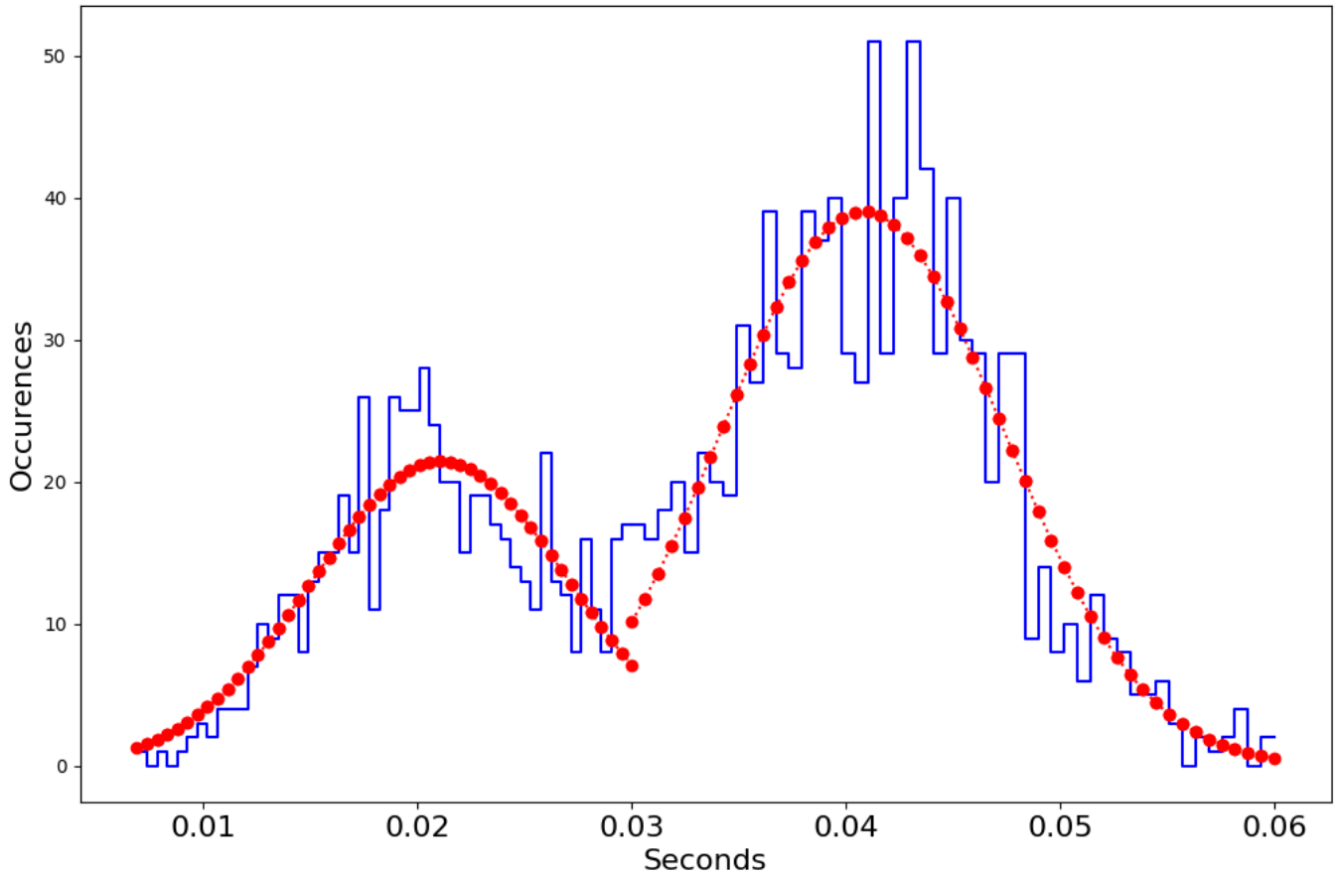
$$\mathbf{T_{react}} = \mathbf{T_{first}} - \mathbf{T_{stop}}$$

ONOS vs ODL reaction times



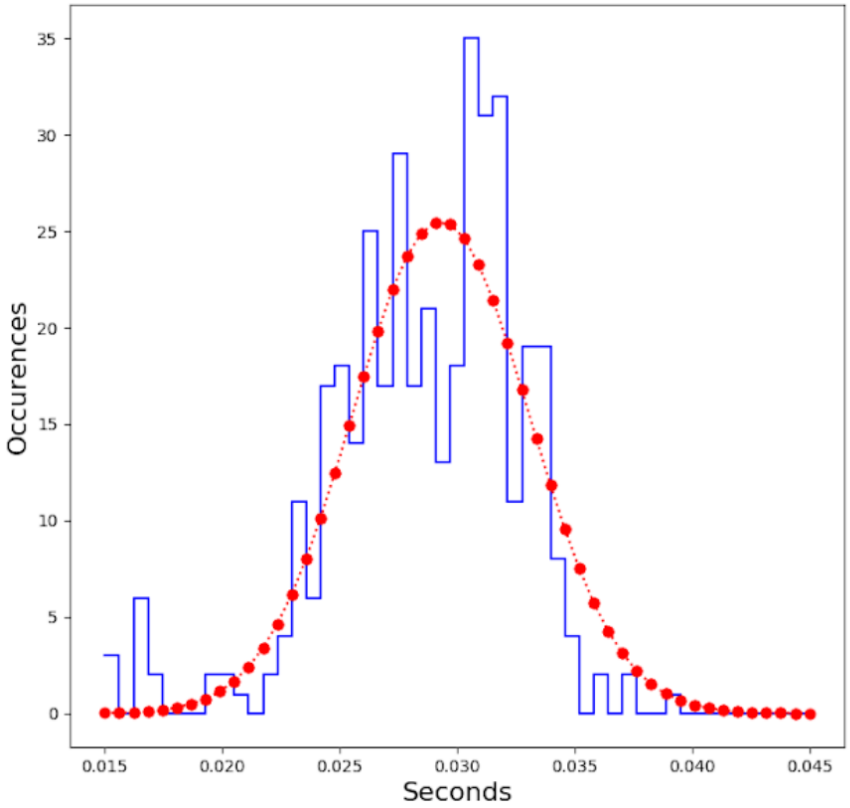
- ODL is **unstable**: in **30,1%** $\mathbf{T_{react} \in [0, 0.04]}$, while in the remaining **69.91%** of cases $\mathbf{T_{react} \in [3, 10]}$.

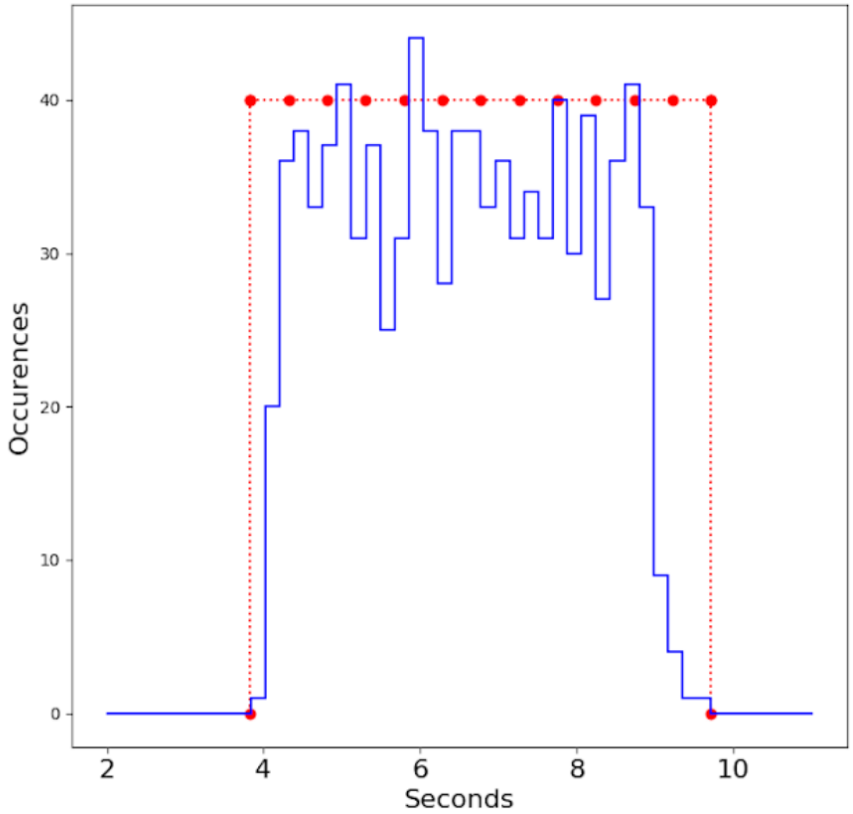
ONOS insights



Two no-gapped modes: first centered near 0.02 secs, the second centered at 0.045 secs

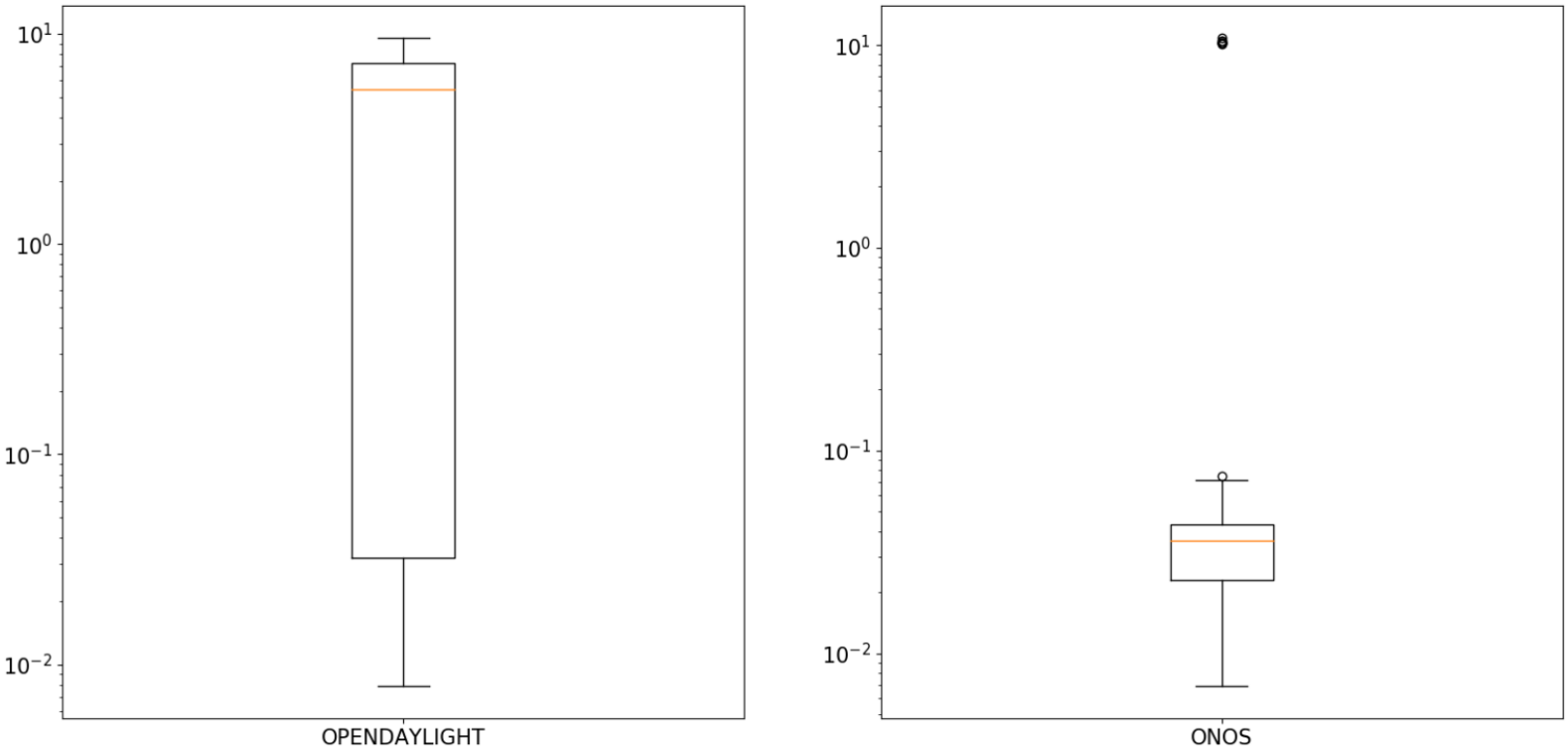
ODL insights





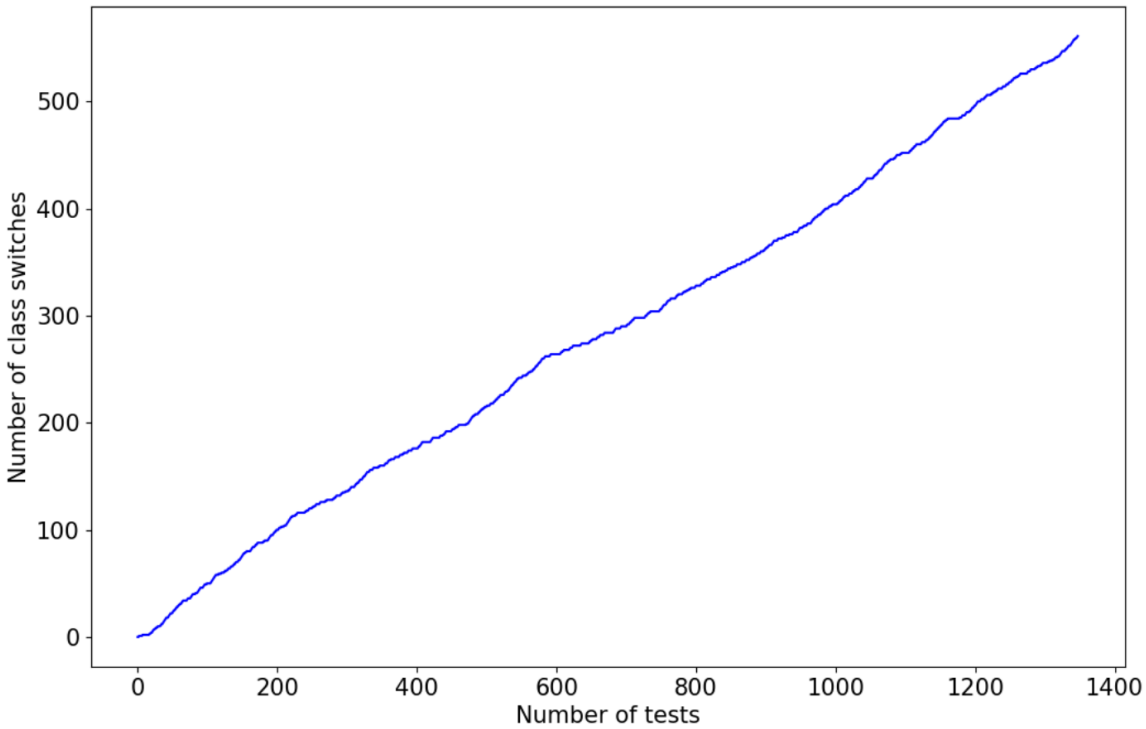
Two gapped modes: first centered near **0.03** secs, the second around **7** secs

Variability



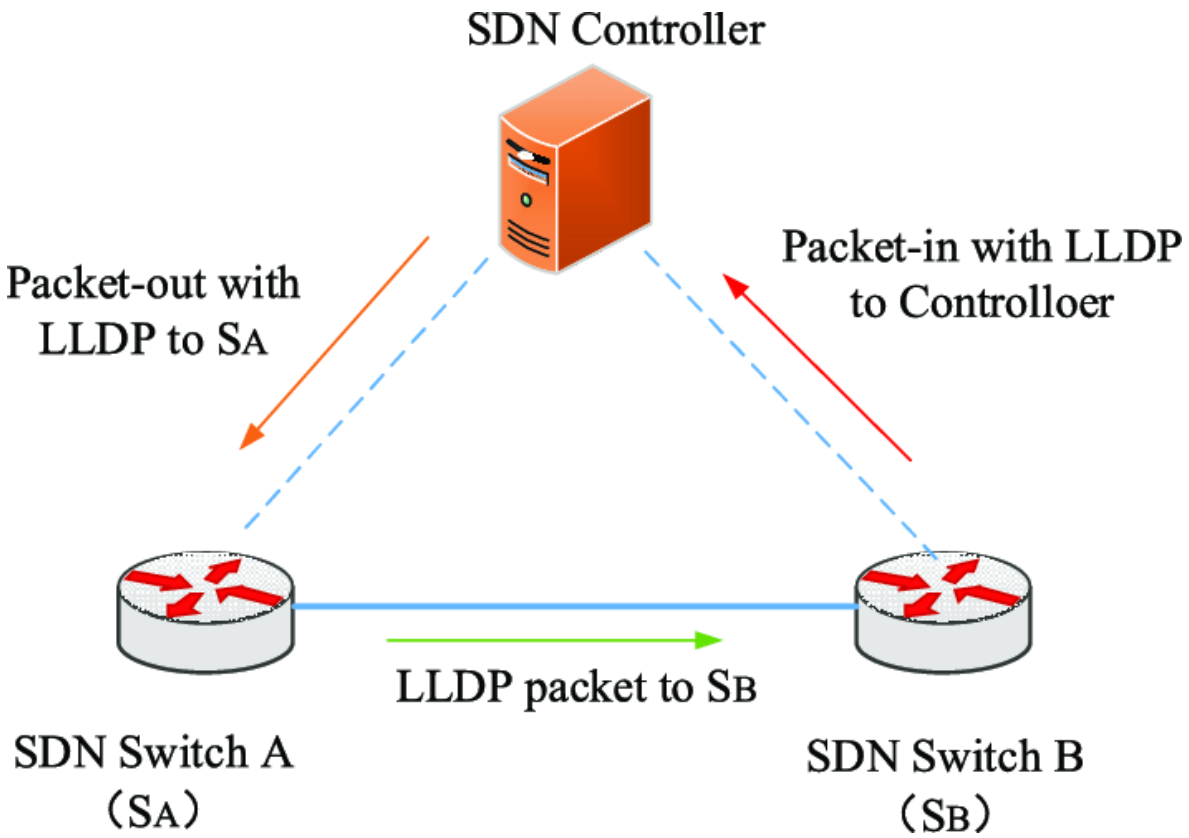
- ONOS median reaction time is **0.036** secs
- ODL median reaction time is **5.45** secs

ODL stability

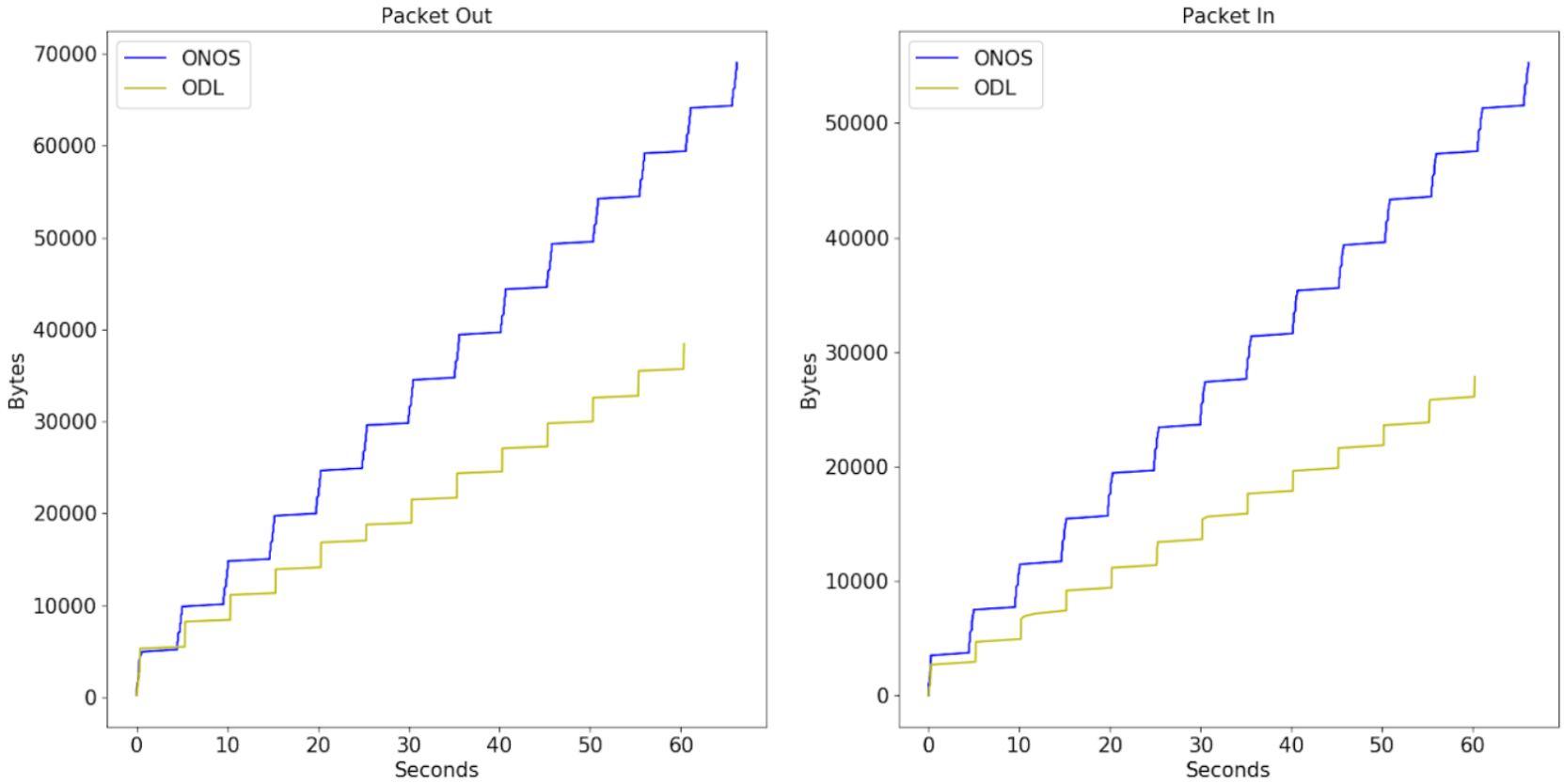


If at the i-th test the reaction time is in first [second] class in the subsequent i+1-th test the reaction time fall in the second [first] class $\rightarrow$ ODL instability is "predictable".

LLDP protocol



LLDP traffic volumes



ONOS produces a bigger amount of LLDP traffic (PACKET_OUT and thus PACKET_IN)

Conclusion and future work

- **ONOS is faster and more stable** reacting to link-up event
- **ODL is unpredictable** when reacting to link-up event

Future work

Why is there a two order of magnitude difference between the two reaction time classes? It would be interesting to study ODL core mechanism triggered by `OFPT_PORT_STATUS`.

- **ODL produces less LLDP traffic**

Future work

Deeply inspect ODL's and ONOS' LLDP implementation

QUESTIONS?