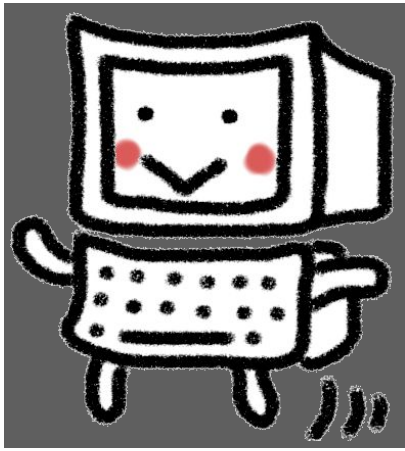


Lab 4

Path Application

Date : 2020/04/16 (THU) 12:00

Deadline : 2020/05/06 (WED) 23:55

Outline

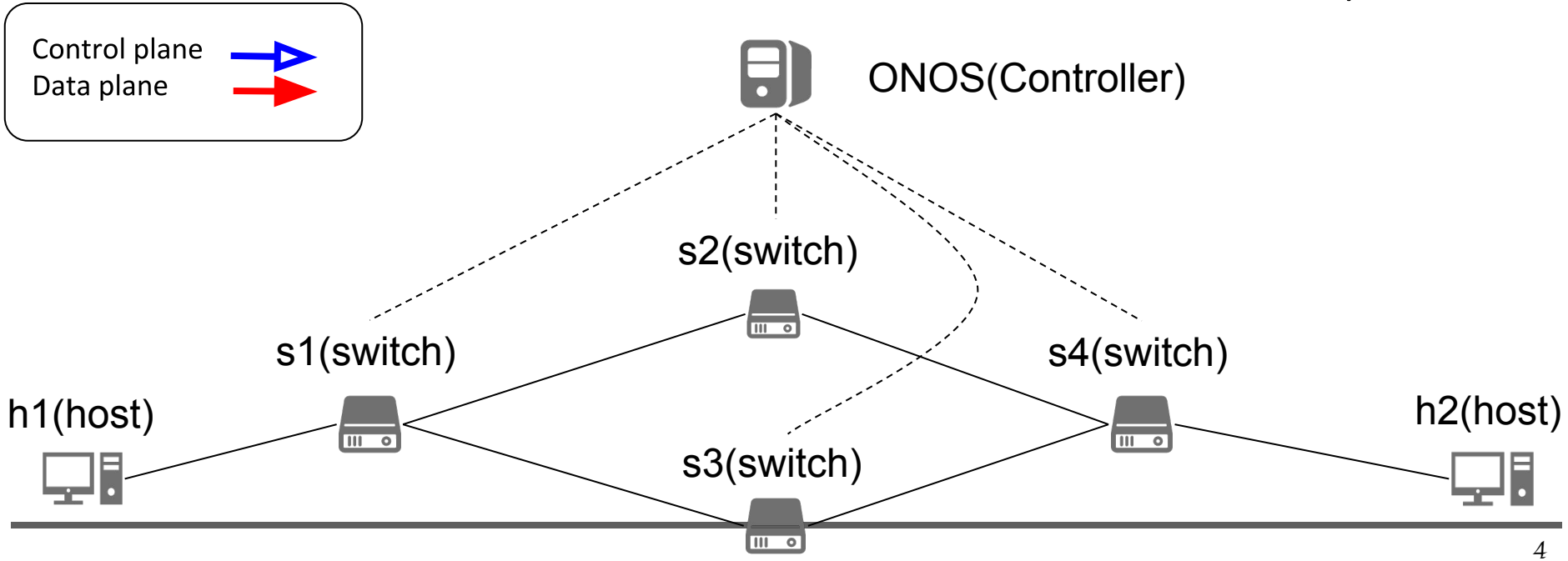
- Lab 4 - Path Application
- Lab 4 Requirements
- Demonstration
- About Submission

Outline

- Lab 4 - Path Application
- Lab 4 Requirements
- Demonstration
- About Submission

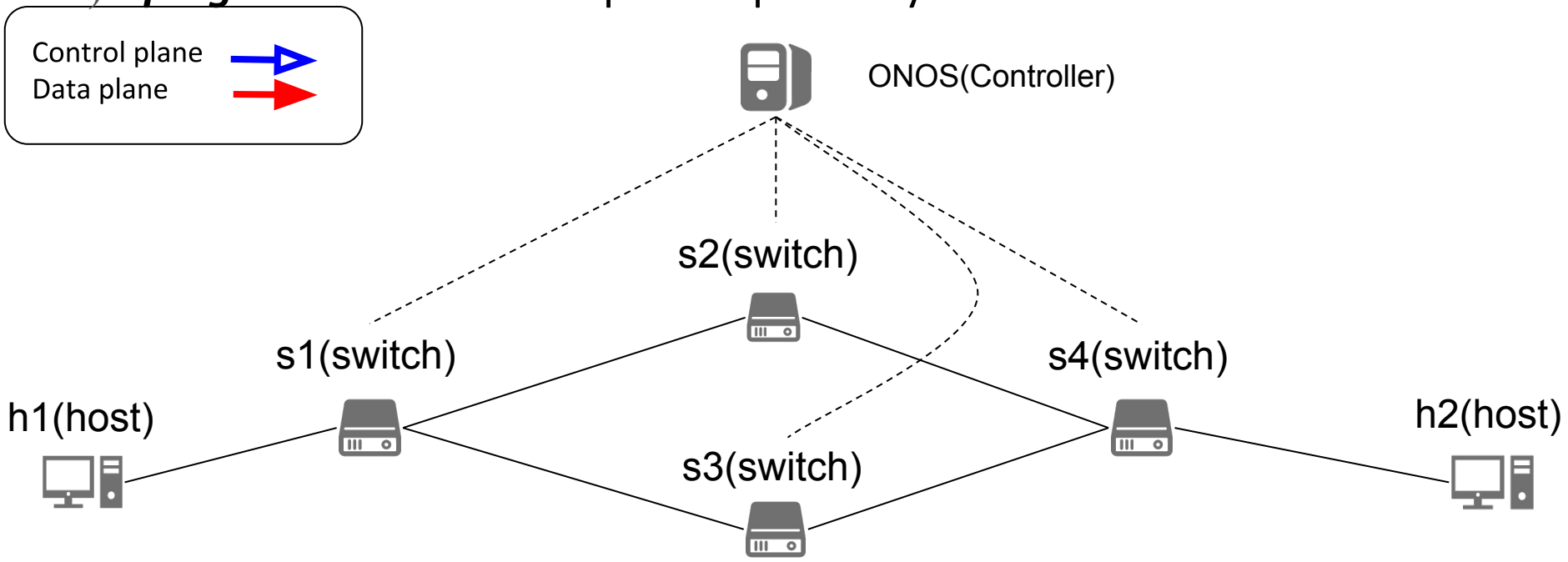
Path Application - Introduction

- ***Application for requesting computation of paths using the current topology snapshot.***
  a. Find path
     i. Retrieve a network topology graph when the controller receives Packet-in
     ii. Find a path from the source to destination
  b. Install flow rules
     i. Proactive install flow rules on each network device in the path



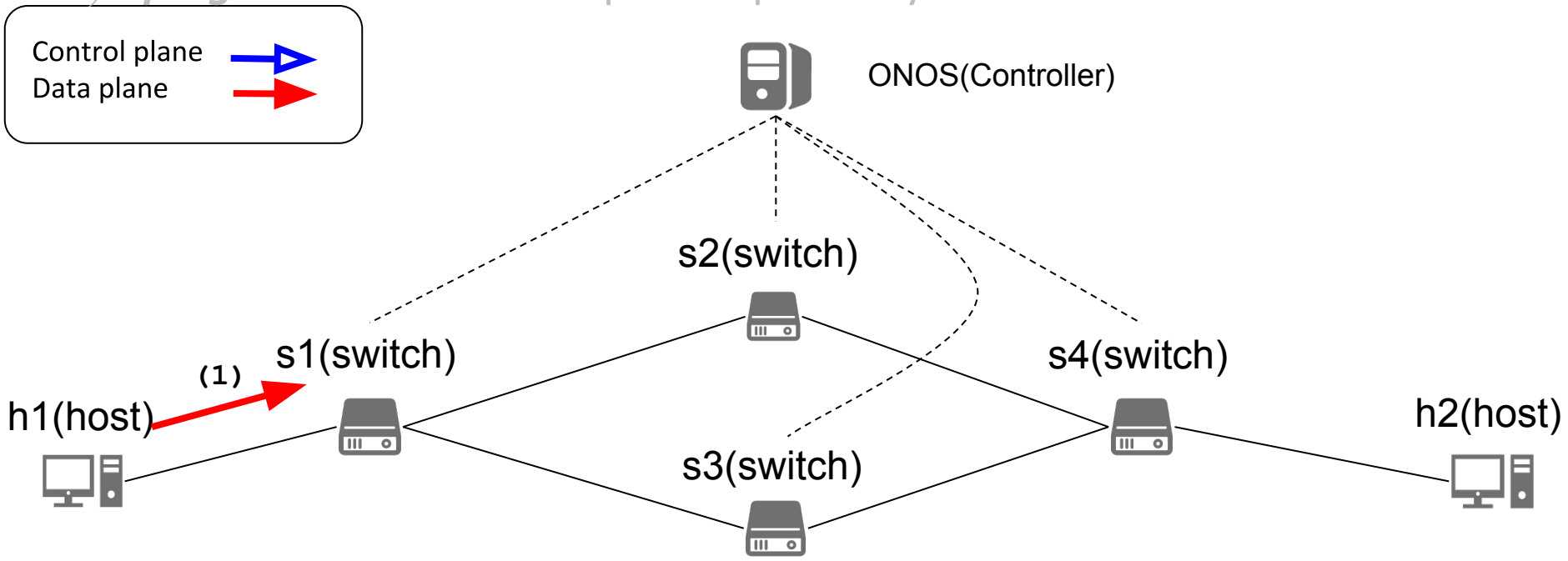
Path Application - Workflow

1) h1 uses ***ping*** to send ICMP request to h2
2) No existing flow matching for ICMP request on s1, so s1 sends Packet-in to ONOS
3) Retrieve the graph of current network topology
4) Use path algorithm (e.g. Dijkstra) to find a path from h1 to h2
5) Proactive install flow rules on each switch in the path
6) ***ping*** will send ICMP request repeatedly to h2



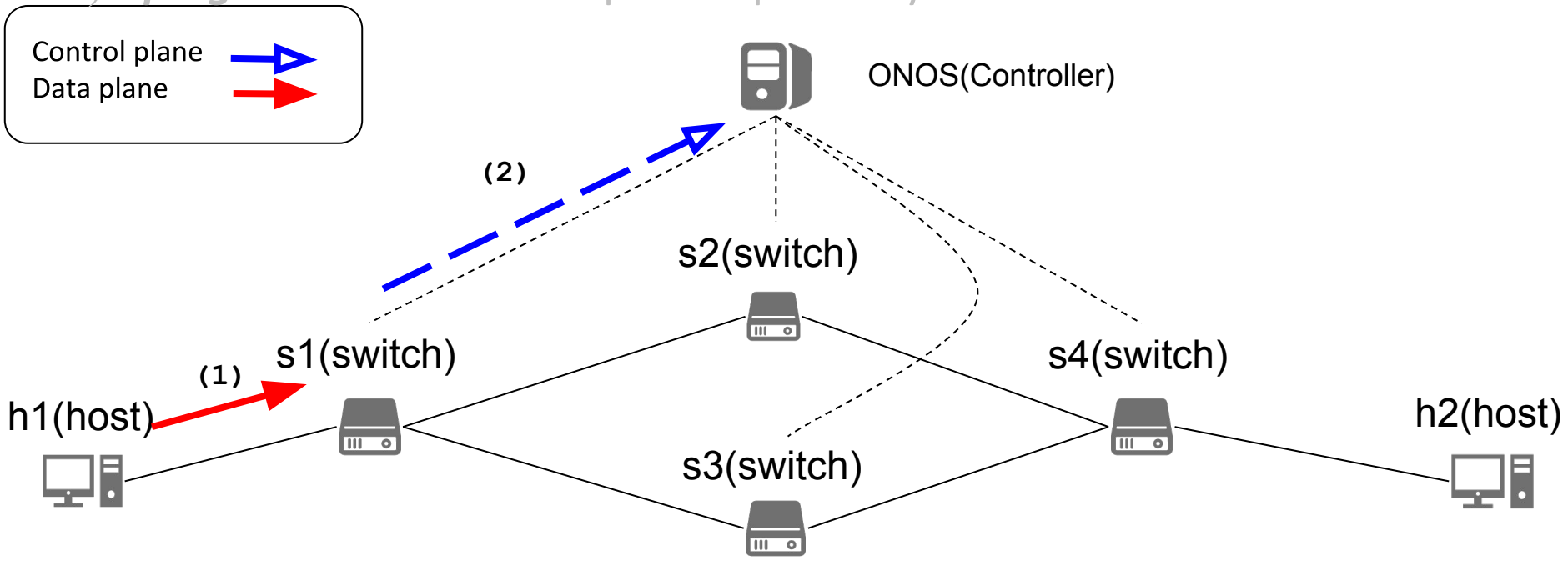
Path Application - Workflow

1) h1 uses ***ping*** to send ICMP request to h2
2) No existing flow matching for ICMP request on s1, so s1 sends Packet-in to ONOS
3) Retrieve the graph of current network topology
4) Use path algorithm (e.g. Dijkstra) to find a path from h1 to h2
5) Proactive install flow rules on each switch in the path
6) ***ping*** will send ICMP request repeatedly to h2



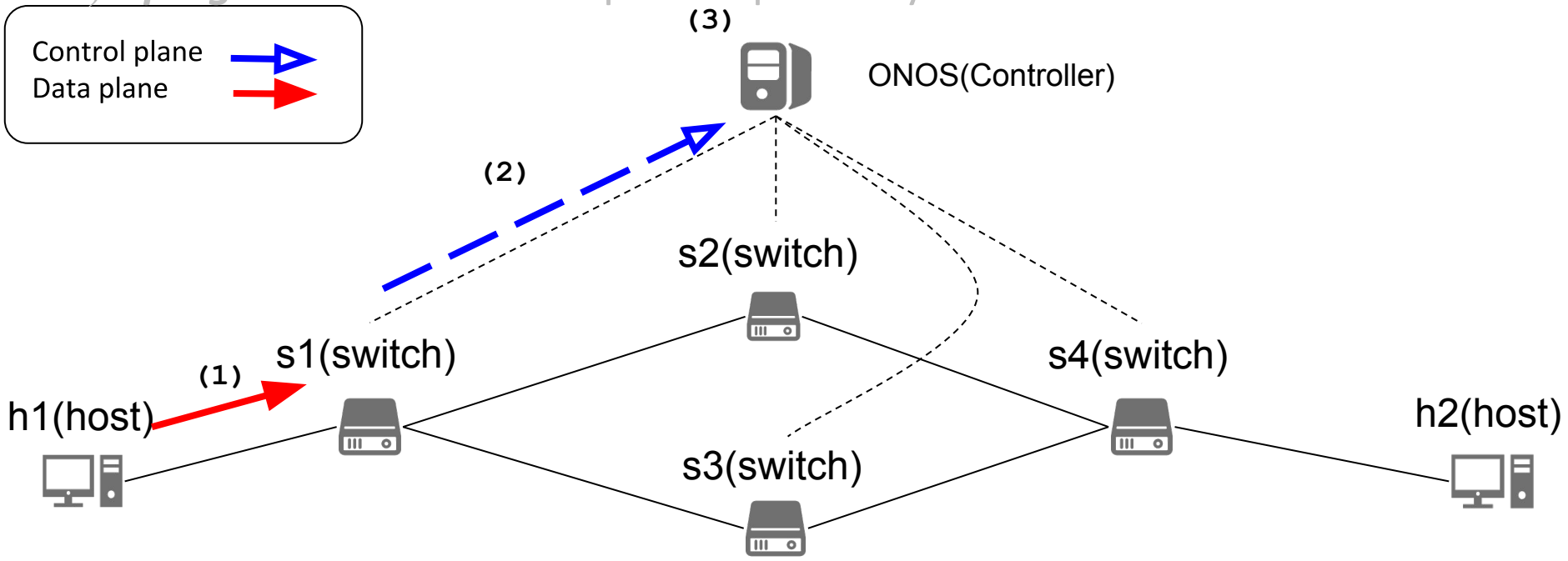
Path Application - Workflow

1) h1 uses ***ping*** to send ICMP request to h2
2) No existing flow matching for ICMP request on s1, so s1 sends Packet-in to ONOS
3) Retrieve the graph of current network topology
4) Use path algorithm (e.g. Dijkstra) to find a path from h1 to h2
5) Proactive install flow rules on each switch in the path
6) ***ping*** will send ICMP request repeatedly to h2



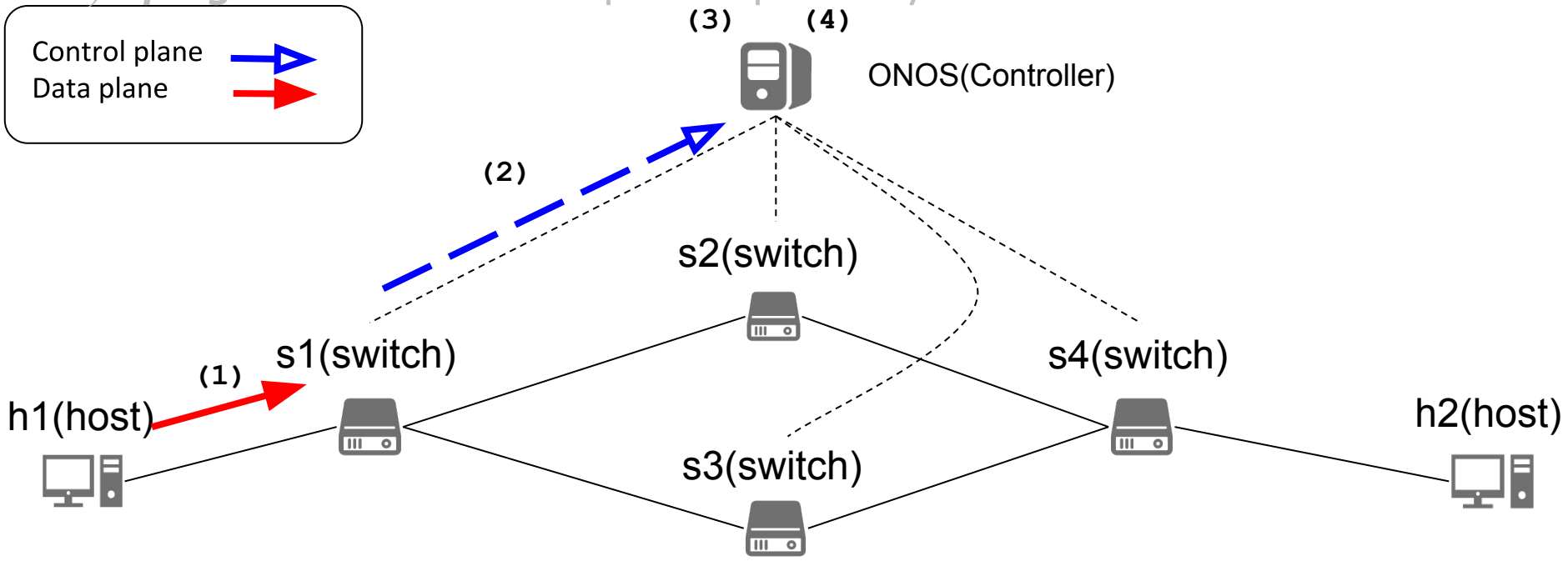
Path Application - Workflow

1) h1 uses ***ping*** to send ICMP request to h2
2) No existing flow matching for ICMP request on s1, so s1 sends Packet-in to ONOS
3) Retrieve the graph of current network topology
4) Use path algorithm (e.g. Dijkstra) to find a path from h1 to h2
5) Proactive install flow rules on each switch in the path
6) ***ping*** will send ICMP request repeatedly to h2



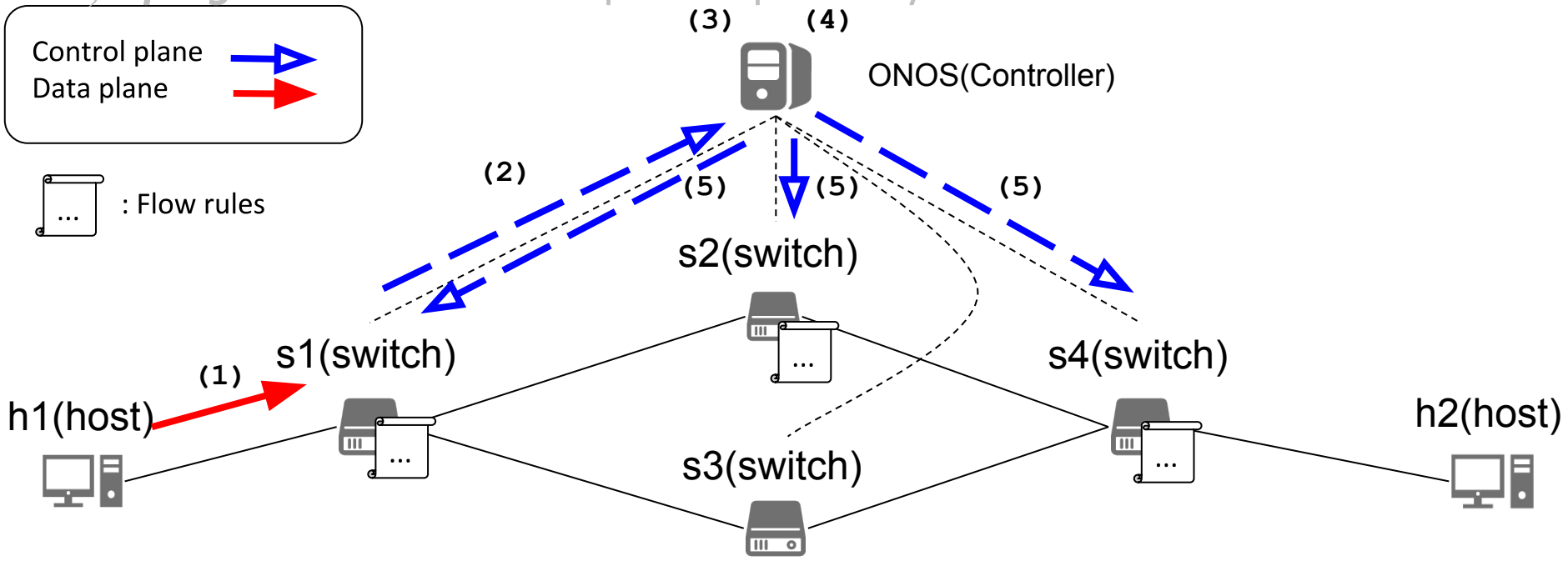
Path Application - Workflow

1) h1 uses ***ping*** to send ICMP request to h2
2) No existing flow matching for ICMP request on s1, so s1 sends Packet-in to ONOS
3) Retrieve the graph of current network topology
4) Use path algorithm (e.g. Dijkstra) to find a path from h1 to h2
5) Proactive install flow rules on each switch in the path
6) ***ping*** will send ICMP request repeatedly to h2



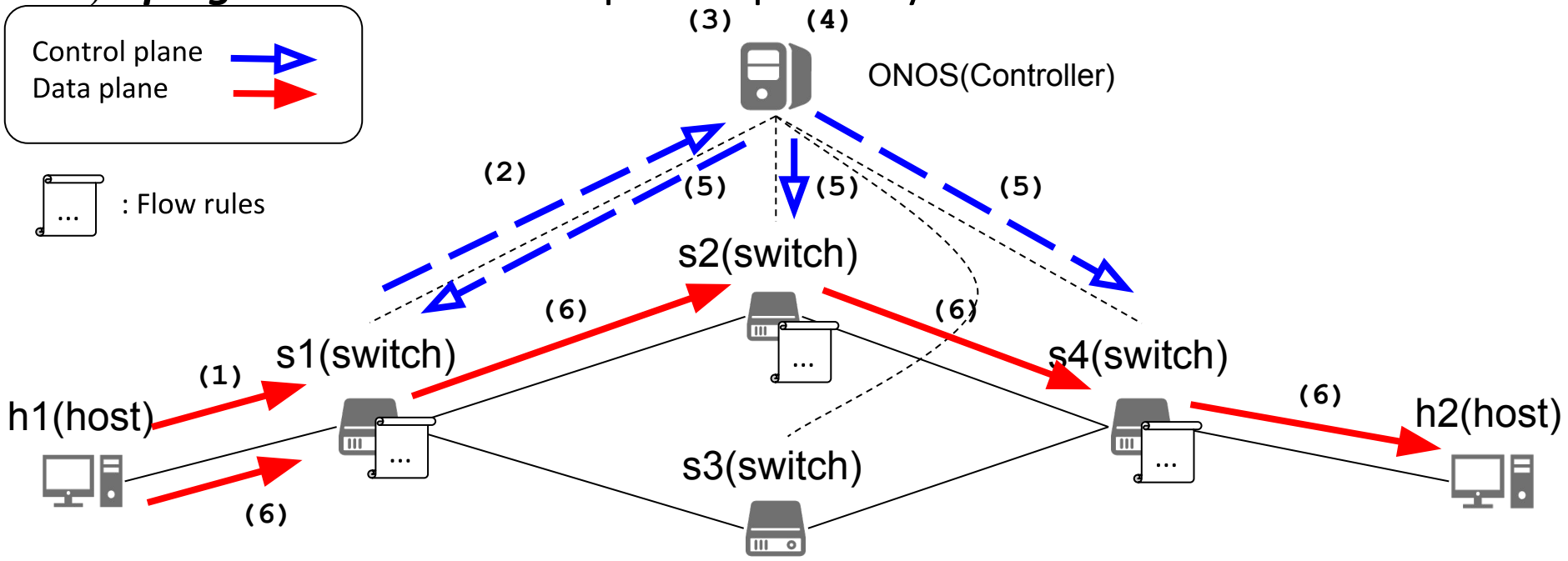
Path Application - Workflow

1) h1 uses ***ping*** to send ICMP request to h2
2) No existing flow matching for ICMP request on s1, so s1 sends Packet-in to ONOS
3) Retrieve the graph of current network topology
4) Use path algorithm (e.g. Dijkstra) to find a path from h1 to h2
5) Proactive install flow rules on each switch in the path
6) ***ping*** will send ICMP request repeatedly to h2



Path Application - Workflow

1) h1 uses ***ping*** to send ICMP request to h2
2) No existing flow matching for ICMP request on s1, so s1 sends Packet-in to ONOS
3) Retrieve the graph of current network topology
4) Use path algorithm (e.g. Dijkstra) to find a path from h1 to h2
5) Proactive install flow rules on each switch in the path
6) ***ping*** will send ICMP request repeatedly to h2



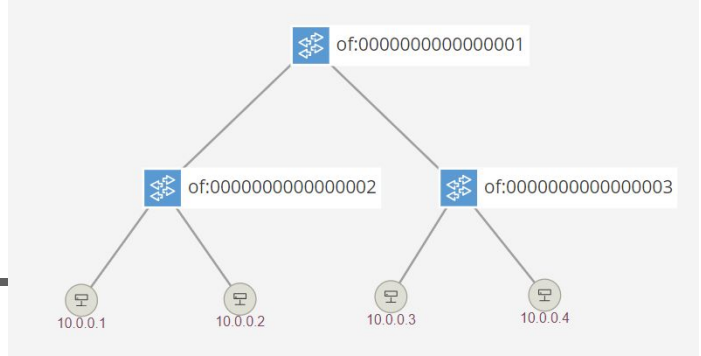
Outline

- Lab 4 - Path Service
- Lab 4 Requirements
- Demonstration
- About Submission

Part I: Path Application for tree topology

- **Find paths and install flow rules**
  - For mininet ***tree*** topology with depths 2 to 5
  - **DO NOT** use ONOS built-in APIs to find paths (e.g. TopologyService.getPath())
  - Shortest path algorithms are not required
  - Please use Logger (Java API) to print out the following information
    i. **Packet-in received**
       - *'Packet-in from device <deviceId>'*
    ii. **Successfully found a path between two hosts**
       - *'Start to install path from <dstMac> to <srcMac>'*
    iii. **Flow rules installed**
       - *'Install flow rule on <deviceId>'*

```
| 196 | Started
| 166 | Application nctu.winlab.myfwd has been activated
| 196 | Packet-in from device of:0000000000000002
| 196 | Start to install path from 26:B0:0E:BE:DB:F9/None to B6:AE:52:E5:04:23/None
| 196 | Install flow rule on of:0000000000000003
| 196 | Install flow rule on of:0000000000000001
| 196 | Install flow rule on of:0000000000000002
| 196 | Packet-in from device of:0000000000000003
| 196 | Start to install path from B6:AE:52:E5:04:23/None to 26:B0:0E:BE:DB:F9/None
| 196 | Install flow rule on of:0000000000000002
| 196 | Install flow rule on of:0000000000000001
| 196 | Install flow rule on of:0000000000000003
```



Part II: Path Application for topology given by TA

- **Find paths and install flow rules**
  - For topology given by TA, which may have loops.
  - **DO NOT** use ONOS built-in APIs to find paths (e.g. TopologyService.getPath())
  - Shortest path algorithms are not required
  - Please use Logger (Java API) to print out the following information
    i. **Packet-in received**
       - *'Packet-in from device <deviceId>'*
    ii. **Successfully found a path between two hosts**
       - *'Start to install path from <dstMac> to <srcMac>'*
    iii. **Flow rules installed**
       - *'Install flow rule on <deviceId>'*

```
| 196 | Started
| 166 | Application nctu.winlab.myfwd has been activated
| 196 | Packet-in from device of:0000000000000002
| 196 | Start to install path from 26:B0:0E:BE:DB:F9/None to B6:AE:52:E5:04:23/None
| 196 | Install flow rule on of:0000000000000003
| 196 | Install flow rule on of:0000000000000001
| 196 | Install flow rule on of:0000000000000002
| 196 | Packet-in from device of:0000000000000003
| 196 | Start to install path from B6:AE:52:E5:04:23/None to 26:B0:0E:BE:DB:F9/None
| 196 | Install flow rule on of:0000000000000002
| 196 | Install flow rule on of:0000000000000001
| 196 | Install flow rule on of:0000000000000003
```

Other Requirements (1/3)

- **ONOS Applications activation**
  - **DO NOT** use ONOS built-in apps to activate a forwarding path. (e.g. fwd)
  - Please activate ***proxyarp*** to handle ARP request
  - You may activate your ***path-app*** and the following ONOS applications only.

```
winlab@root > apps -a -s
*  12 org.onosproject.optical-model        2.2.0    Optical Network Model
*  13 org.onosproject.drivers              2.2.0    Default Drivers
*  83 org.onosproject.openflow-base        2.2.0    OpenFlow Base Provider
*  84 org.onosproject.lldpprovider         2.2.0    LLDP Link Provider
*  85 org.onosproject.hostprovider         2.2.0    Host Location Provider
* 115 org.onosproject.proxyarp             2.2.0    Proxy ARP/NDP
* 156 org.onosproject.openflow            2.2.0    OpenFlow Provider Suite
* 172 org.onosproject.gui2                2.2.0    ONOS GUI2
winlab@root >
```

Other Requirements (2/3)

- **Flow rule regulation**
  - Match field (selector): ETH_TYPE, IPV4_SRC, IPV4_DST
  - Action field (treatment): OUTPUT
  - Flow priority: 10
  - Flow timeout: 30
- Only the first ICMP request and the first ICMP reply will trigger Packet-in.
  - ***ping*** will send ICMP request repeatedly from source to destination
- Proactively install flow rules on network devices from destination to source along the path
- **DO NOT** use Intent to install flow rules.

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	2	52	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	4	80	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	4	15	10	0	ETH_TYPE:ipv4, IPV4_SRC:10.0.0.4/32, IPV4_DST:10.0.0.2/32	imm[OUTPUT:3], cleared:false	nctu.winlab.pathapp
Added	5	16	10	0	ETH_TYPE:ipv4, IPV4_SRC:10.0.0.2/32, IPV4_DST:10.0.0.4/32	imm[OUTPUT:2], cleared:false	nctu.winlab.pathapp
Added	6	26	10	0	ETH_TYPE:ipv4, IPV4_SRC:10.0.0.3/32, IPV4_DST:10.0.0.1/32	imm[OUTPUT:3], cleared:false	nctu.winlab.pathapp
Added	7	27	10	0	ETH_TYPE:ipv4, IPV4_SRC:10.0.0.1/32,	imm[OUTPUT:1], cleared:false	nctu.winlab.pathapp

Other Requirements (3/3)

- **Project naming convention**
  - You should follow the Maven project naming format below, or your project will not be scored.
    - <groupId>: **nctu.winlab**
    - <artifactId>: **path-app**
    - <version>: (arbitrary)
    - <package>: **nctu.winlab.pathapp**

Project 4 Scoring Criteria

- (40%) Path Service for tree topology
- (30%) Path Service for topology given by TA
- (20%) Flow rule regulation
- (10%) Project naming convention

Outline

- Project 4 - Path Service
- Project 4 Requirements
- Demonstration
- About Submission

Demonstration (1/3) - Environment setup

- Start ONOS

```
$ bazel run onos-local -- clean
```

- Activate ***proxyarp*** to handle ARP packet

```
$ onos localhost app activate proxyarp
```

- Compile your project

```
$ cd <artifactId> && mvn clean install -DskipTests
```

- Install and activate your application on ONOS

```
$ cd <artifactId> && onos-app localhost install!
target/<artifactId>-<version>.oar
```

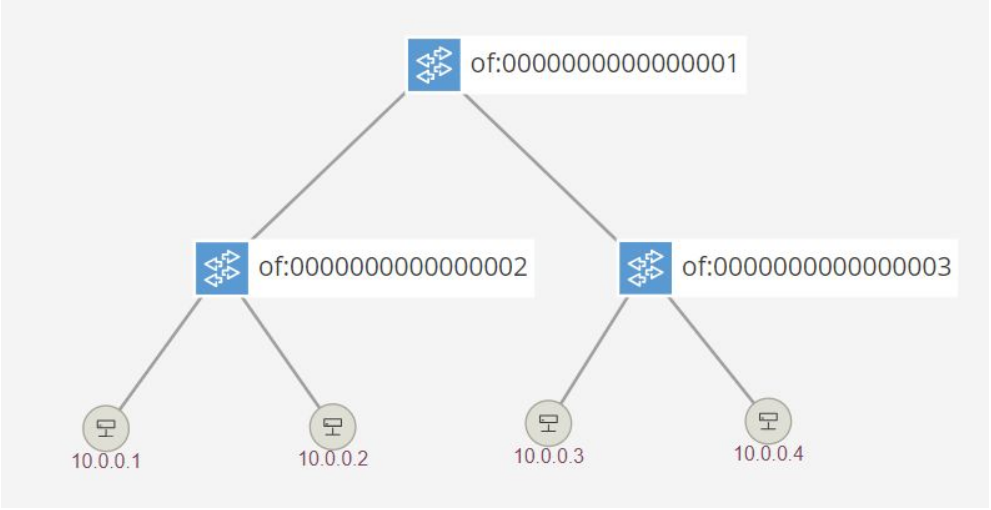
Demonstration (2/3) - ping

- Using Mininet tree topology to test your application
  - Ex: ***tree*** topology, depth=***2***

```
$ sudo mn --controller=remote,ip=127.0.0.1:6653 --topo=tree,depth=2
```

- Ping from h1 to h3

```
mininet> h1 ping h3
```




```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.395 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.072 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.083 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.069 ms
^C
--- 10.0.0.3 ping statistics ---
7 packets transmitted, 5 received, 28% packet loss, time 6142ms
rtt min/avg/max/mdev = 0.063/0.136/0.395/0.129 ms
```

Demonstration (3/3) - ONOS log

```
| 196 | Started
| 166 | Application nctu.winlab.myfwd has been activated
| 196 | Packet-in from device of:0000000000000002
| 196 | Start to install path from 26:B0:0E:BE:DB:F9/None to B6:AE:52:E5:04:23/None
| 196 | Install flow rule on of:0000000000000003
| 196 | Install flow rule on of:0000000000000001
| 196 | Install flow rule on of:0000000000000002
| 196 | Packet-in from device of:0000000000000003
| 196 | Start to install path from B6:AE:52:E5:04:23/None to 26:B0:0E:BE:DB:F9/None
| 196 | Install flow rule on of:0000000000000002
| 196 | Install flow rule on of:0000000000000001
| 196 | Install flow rule on of:0000000000000003
```

Hints

- You can use path algorithms like BFS, Dijkstra or spanning tree to find a path between two hosts.
- Forwarding packets in data plane is much faster than installing flow rules in control plane
- The first two ***ping***s will fail
- Use ONOS TopologyService.getGraph(topology) to retrieve the network topology graph.
- Use ONOS TopologyGraph.getEdgeFrom(vertex) to acquire all edges leading out from the specified vertex (switch).

Outline

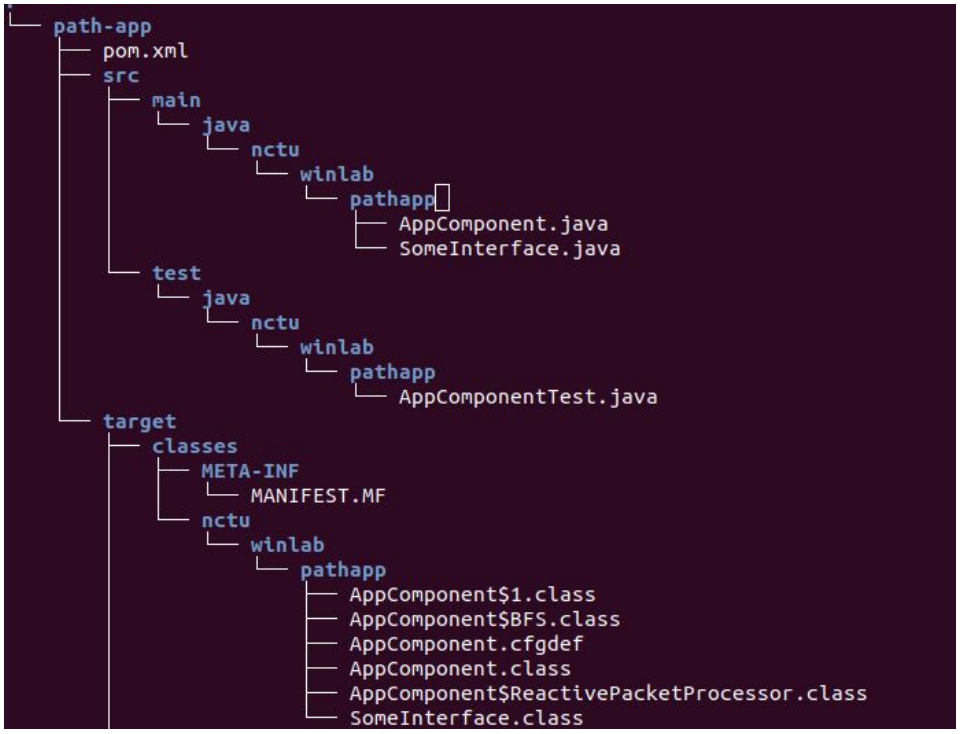
- Project 4 - Path Service
- Project 4 Requirements
- Demonstration
- About Submission

Path Service

About Submission

About submission

- Files
  - You need to submit all files under your project directory
  - Zip the whole ***path-app*** folder into a *.zip* file
    - Named: **project4_<studentID>.zip**
- Submit
  - Upload ".zip" file to New e3
    - Named: **project4_<studentID>.zip**
  - Incorrect naming convention or format subjects to not scoring.



Q & A

Thank you

References

- ONOS Java API 2.2.0
  - http://api.onosproject.org/2.2.0/apidocs/