

Project 3

ONOS Application Development: SDN-enabled Learning Bridge

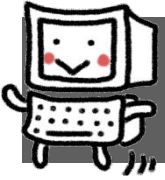
Date : 2020/03/26 (THR) 12:00

Deadline : 2020/04/15 (WED) 23:55



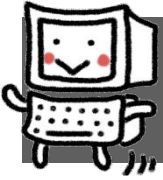
Outline

- ☐ Overview
- ☐ Create ONOS Application
- ☐ Learning Bridge Function
- ☐ Project 3 Requirements

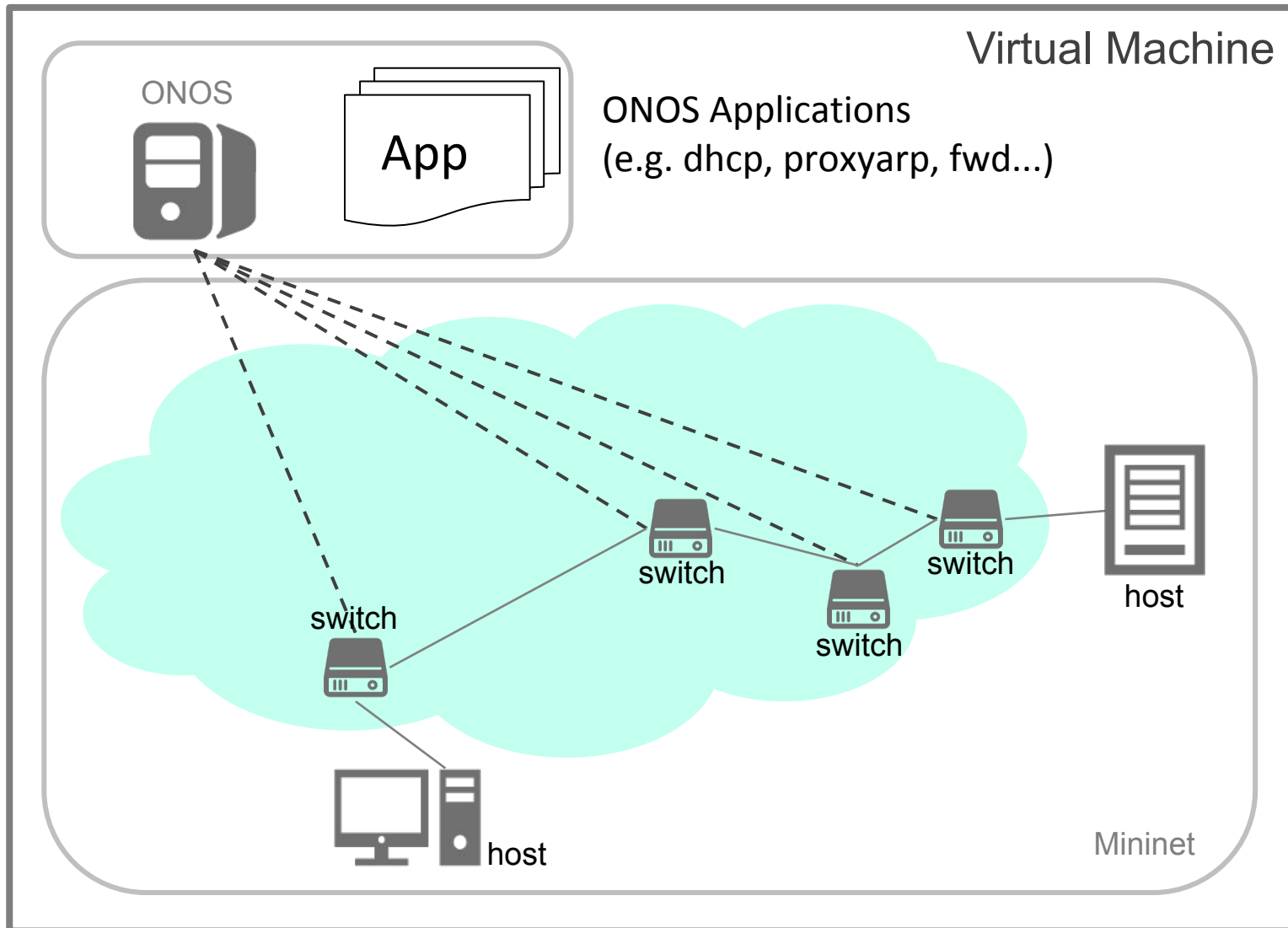


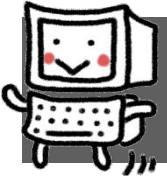
Outline

- ☒ Overview
- ☐ Create ONOS Application
- ☐ Learning Bridge Function
- ☐ Project 3 Requirements



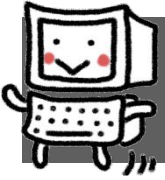
Overview





Outline

- ☐ Overview
- ☒ Create ONOS Application
- ☐ Learning Bridge Function
- ☐ Project 3 Requirements



Developing ONOS Applications







Create ONOS Application

Environment Setup - JDK installation (1/4)

- ❑ Download Oracle JDK 11 (JDK: Java Development Kit)
 - Java SE Development Kit 11- - Downloads

Java SE Development Kit 11.0.6

This software is licensed under the [Oracle Technology Network License Agreement for Oracle Java SE](#)

Product / File Description	File Size	Download
Linux Debian Package	147.99 MB	 jdk-11.0.6_linux-x64_bin.deb
Linux RPM Package	154.65 MB	 jdk-11.0.6_linux-x64_bin.rpm
Linux Compressed Archive	171.8 MB	 jdk-11.0.6_linux-x64_bin.tar.gz
macOS Installer	166.45 MB	 jdk-11.0.6_osx-x64_bin.dmg
macOS Compressed Archive	166.77 MB	 jdk-11.0.6_osx-x64_bin.tar.gz
Solaris SPARC Compressed Archive	188.51 MB	 jdk-11.0.6_solaris-sparcv9_bin.tar.gz



Environment Setup - JDK installation (2/4)

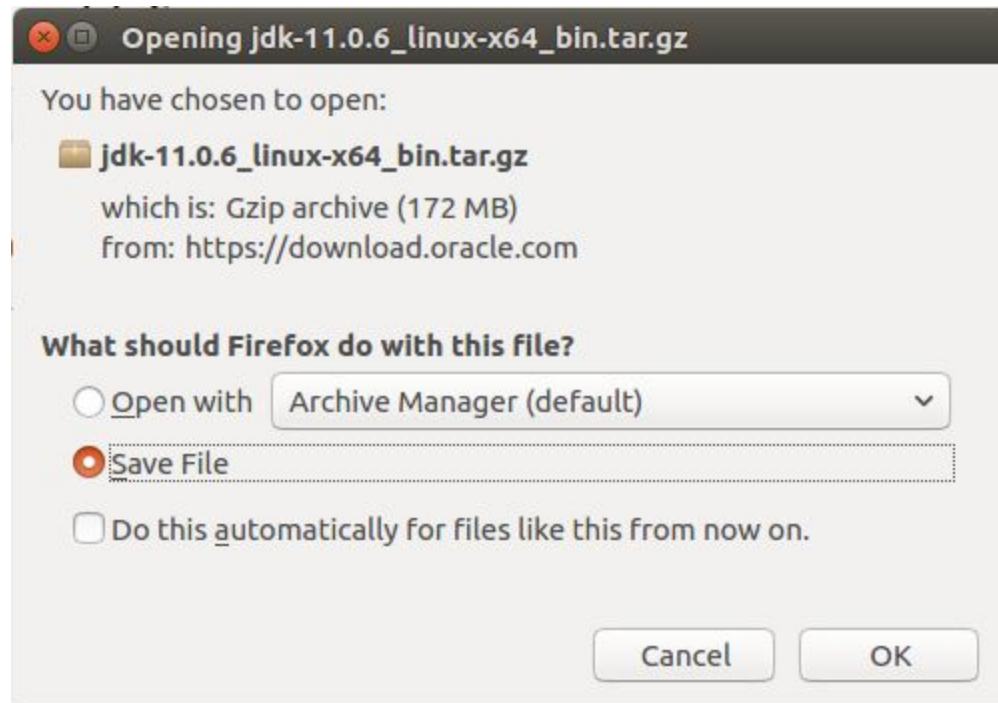
- ❑ Download Oracle JDK 11 (JDK: Java Development Kit)
 - You will be asked to create an Oracle account to download this software.

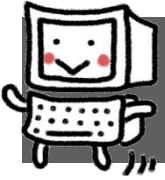
A screenshot of the Oracle JDK 11 download page. It shows a license agreement window with a close button (X) in the top right corner. The text reads: "You must accept the Oracle Technology Network License Agreement for Oracle Java SE to download this software." Below this is a checkbox with a green checkmark and the text "I reviewed and accept the Oracle Technology Network License Agreement for Oracle Java SE". At the bottom is a green button labeled "Download jdk-11.0.6_linux-x64_bin.tar.gz" with a download icon.



Environment Setup - JDK installation (3/4)

- ❑ Download Oracle JDK 11 (JDK: Java Development Kit)
 - After creating the Oracle account and login, you can download this file now.





Environment Setup - JDK installation (4/4)

- ❑ Untar JDK in `/opt`

```
$ sudo tar -zxf ~/Downloads/jdk-11.0.6_linux-x64_bin.tar.gz -C /opt
```

- ❑ Setting Oracle JDK as the default JVM

```
$ sudo update-alternatives --install /usr/bin/java java /opt/jdk-11.0.6/bin/java 2000  
$ sudo update-alternatives --install /usr/bin/javac javac /opt/jdk-11.0.6/bin/javac 2000
```

- ❑ Check the result by running these command

```
$ java -version  
$ javac -version
```

```
sdnfv@sdnfv-VirtualBox:~$ java -version  
java version "11.0.6" 2020-01-14 LTS  
Java(TM) SE Runtime Environment 18.9 (build 11.0.6+8-LTS)  
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.6+8-LTS, mixed mode)  
sdnfv@sdnfv-VirtualBox:~$ javac -version  
javac 11.0.6  
sdnfv@sdnfv-VirtualBox:~$
```



Environment Setup - Maven

- ☐ Install Maven

```
$ sudo apt install maven
```

- ☐ **Maven**: a software project management and comprehension tool.

- ☐ Indicate the version of ONOS API

```
$ export ONOS_POM_VERSION=2.2.0
```

- ☐ Build the current version of ONOS application archetypes.

- ONOS version: 2.2.0

```
$ cd $ONOS_ROOT/tools/packages/archetypes  
$ mvn clean install -DskipTests
```



Create ONOS Application (1/3)

- ❑ Create ONOS application (Red words are what you type)

```
$ onos-create-app
```

```
...
```

```
[INFO] ...
```

```
Define value for property 'groupId': nctu.winlab
```

```
Define value for property 'artifactId': bridge-app
```

```
Define value for property 'version' 1.0-SNAPSHOT: : <enter>
```

```
Define value for property 'package' nctu.winlab: : nctu.winlab.bridge
```

```
Confirm properties configuration:
```

```
groupId: nctu.winlab
```

```
artifactId: bridge-app
```

```
version: 1.0-SNAPSHOT
```

```
package: nctu.winlab.bridge
```

```
Y: : <enter>
```

```
[INFO] ...
```

```
...
```

```
[INFO] BUILD SUCCESS
```



Create ONOS Application (2/3)

- ❑ Project created successful
 - A folder named <artifactId> is created by *onos-create-app*.

```
sdnfv@sdnfv-VirtualBox: ~/bridge-app$ tree
.
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── nctu
│   │   │   │   ├── winlab
│   │   │   │   │   ├── bridge
│   │   │   │   │   │   ├── AppComponent.java
│   │   │   │   │   │   └── SomeInterface.java
│   │   └── test
│   │       ├── java
│   │       │   ├── nctu
│   │       │   │   ├── winlab
│   │       │   │   │   ├── bridge
│   │       │   │   │   └── AppComponentTest.java
└──
```

11 directories, 4 files



Create ONOS Application (3/3)

- ❑ Describe your project
 - Modify pom.xml (pom: Project Object Model)

```
$ cd <artifactId>  
$ vim pom.xml      # Uncomment line 32~37  
                   # Modify line 32~34
```

Before

```
28     <properties>  
29         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
30         <onos.version>2.2.0</onos.version>  
31         <!-- Uncomment to generate ONOS app from this module.  
32         <onos.app.name>org.foo.app</onos.app.name>  
33         <onos.app.title>Foo App</onos.app.title>  
34         <onos.app.origin>Foo, Inc.</onos.app.origin>  
35         <onos.app.category>default</onos.app.category>  
36         <onos.app.url>http://onosproject.org</onos.app.url>  
37         <onos.app.readme>ONOS OSGi bundle archetype.</onos.app.readme>  
38         -->  
39     </properties>
```

After

```
28     <properties>  
29         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
30         <onos.version>2.2.0</onos.version>  
31         <!-- Uncomment to generate ONOS app from this module -->  
32         <onos.app.name>nctu.winlab.bridge</onos.app.name>  
33         <onos.app.title>Learning Bridge App</onos.app.title>  
34         <onos.app.origin>Winlab, NCTU</onos.app.origin>  
35         <onos.app.category>default</onos.app.category>  
36         <onos.app.url>http://onosproject.org</onos.app.url>  
37         <onos.app.readme>ONOS OSGi bundle archetype.</onos.app.readme>  
38         -->  
39     </properties>
```



Write ONOS Application (1/2)

- ❑ A template code should be placed under
`<artifactId>/src/main/java/nctu/winlab/bridge/`

```
sdnfv@sdnfv-VirtualBox:~/bridge-app$ tree
.
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── nctu
│   │   │   │   ├── winlab
│   │   │   │   │   ├── bridge
│   │   │   │   │   │   ├── AppComponent.java
│   │   │   │   │   │   └── SomeInterface.java
│   │   └── test
│   │       ├── java
│   │       │   ├── nctu
│   │       │   │   ├── winlab
│   │       │   │   │   ├── bridge
│   │       │   │   │   └── AppComponentTest.java
└──
```

11 directories, 4 files



Write ONOS Application (2/2)

- ❑ After *onos-create-app* has created a template code (*AppComponent.java*), you can simply use it to write ONOS application.

```
$ cd <artifactId>
```

```
$ vim src/main/java/nctu/winlab/bridge/AppComponent.java
```

```
public class AppComponent implements SomeInterface {

    private final Logger log = LoggerFactory.getLogger(getClass());

    /** Some configurable property. */
    private String someProperty;

    @Reference(cardinality = ReferenceCardinality.MANDATORY)
    protected ComponentConfigService cfgService;

    @Activate
    protected void activate() {
        cfgService.registerProperties(getClass());
        log.info("Started");
    }

    @Deactivate
    protected void deactivate() {
        cfgService.unregisterProperties(getClass(), false);
        log.info("Stopped");
    }

    @Modified
    public void modified(ComponentContext context) {
        Dictionary<?, ?> properties = context != null ? context.getProperties() : new Properties();
        if (context != null) {
            someProperty = get(properties, "someProperty");
        }
        log.info("Reconfigured");
    }

    @Override
    public void someMethod() {
        log.info("Invoked");
    }

}
```




Compile, Install and Activate ONOS Application

- ☐ Compile ONOS application

```
$ cd <artifactId>  
$ mvn clean install -DskipTests  
# option '-DskipTests' to skip running the tests for our project
```

- ☐ Run ONOS

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

- ☐ Install and activate ONOS application

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

- ☐ Note: By using an **exclamation mark** with ``install`` parameter, we can activate the application immediately after the application being installed on ONOS.



Deactivate, Uninstall ONOS Application

- ❑ Reinstall your application
 - If you have done some modification on your project, you need to recompile your project and reinstall your project on ONOS.
 - i. Re-compile

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

- ii. Deactivate application

<onos-app-name> is indicated in your pom.xml

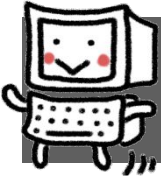
```
$ onos localhost app deactivate <onos-app-name> #e.g. nctu.winlab.bridge-app
```

- iii. Uninstall application

```
$ onos-app localhost uninstall <onos-app-name>
```

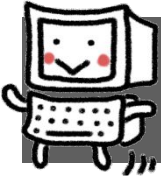
- iv. Install & Activate application

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



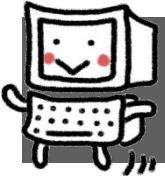
References

- ❑ ONOS Wiki – Template Application Tutorial
 - <https://wiki.onosproject.org/display/ONOS/Template+Application+Tutorial>
- ❑ ONOS Application Subsystem
 - <https://wiki.onosproject.org/display/ONOS/Application+Subsystem>
- ❑ ONOS Java API (2.2.0)
 - <http://api.onosproject.org/2.2.0/apidocs/>
- ❑ JDK installation
 - <https://www.digitalocean.com/community/tutorials/how-to-manually-install-oracle-java-on-a-debian-or-ubuntu-vps>



Outline

- ☐ Overview
- ☐ Create ONOS Application
- ☒ Learning Bridge Function
- ☐ Project 3 Requirements



Developing ONOS Applications

Learning Bridge Function



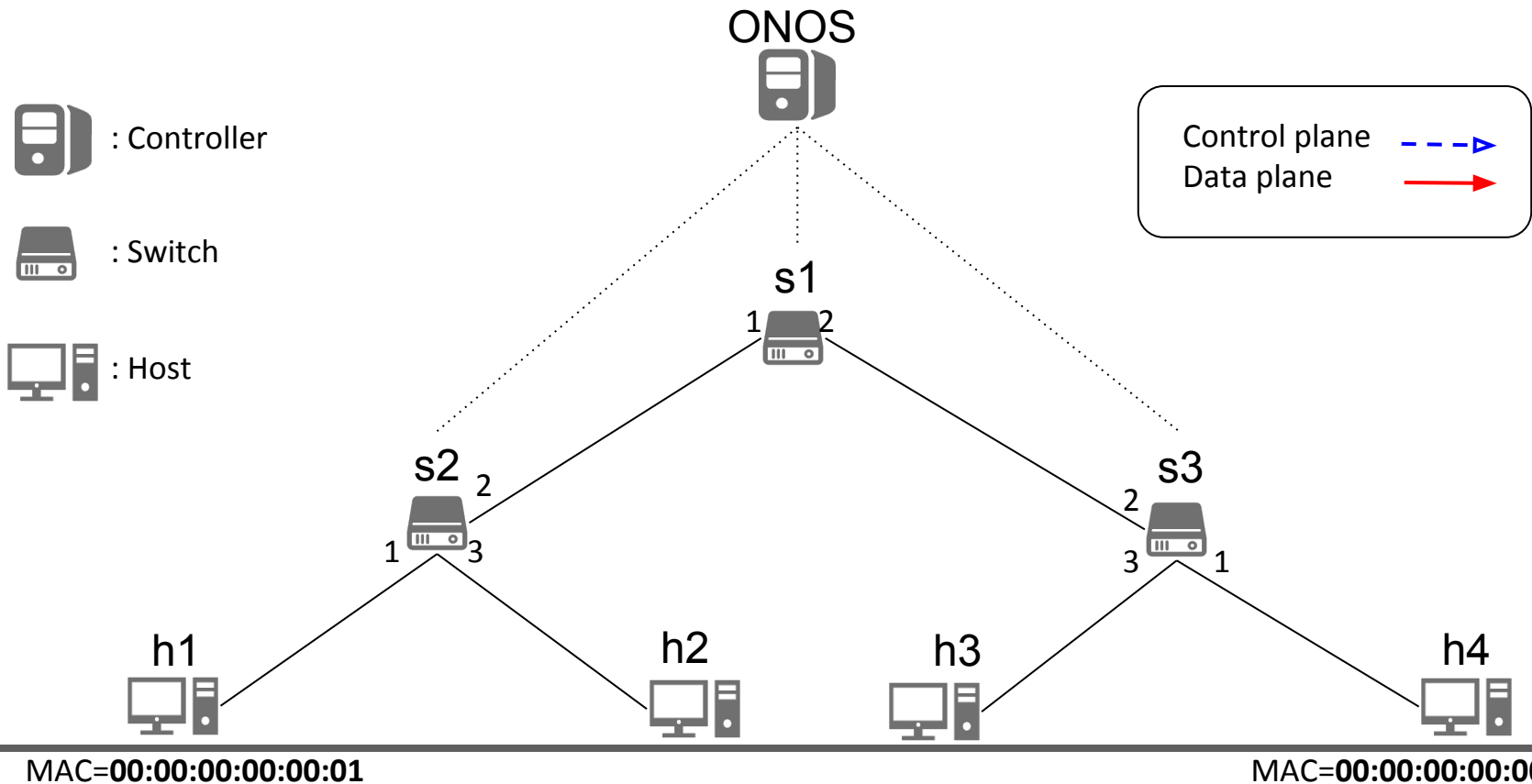
Introduction of Learning Bridge Function

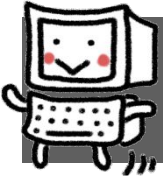
1. Forwarding information learning

- Associate the source MAC address with incoming port

2. Packets forwarding

- Use destination MAC address as index to look up the MAC address table and forward the packet to the proper output port

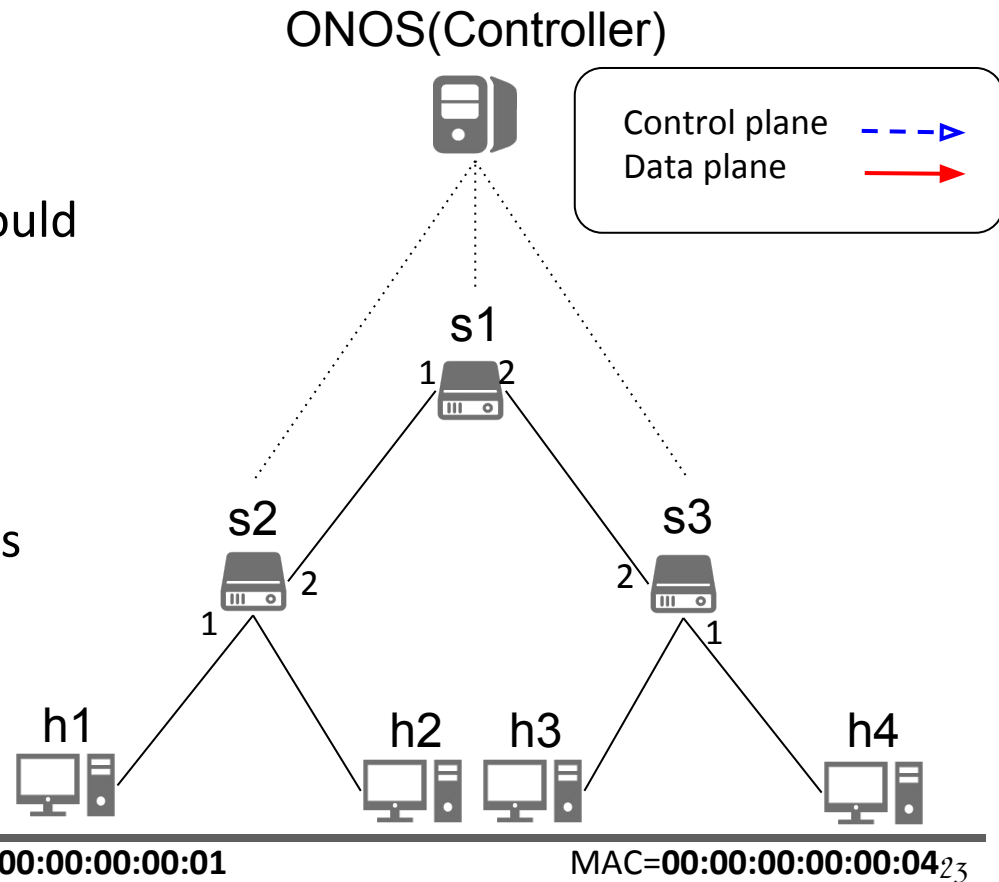


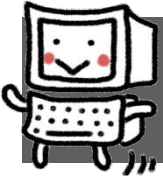


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h4 receives the ICMP packet which is send by h1

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port

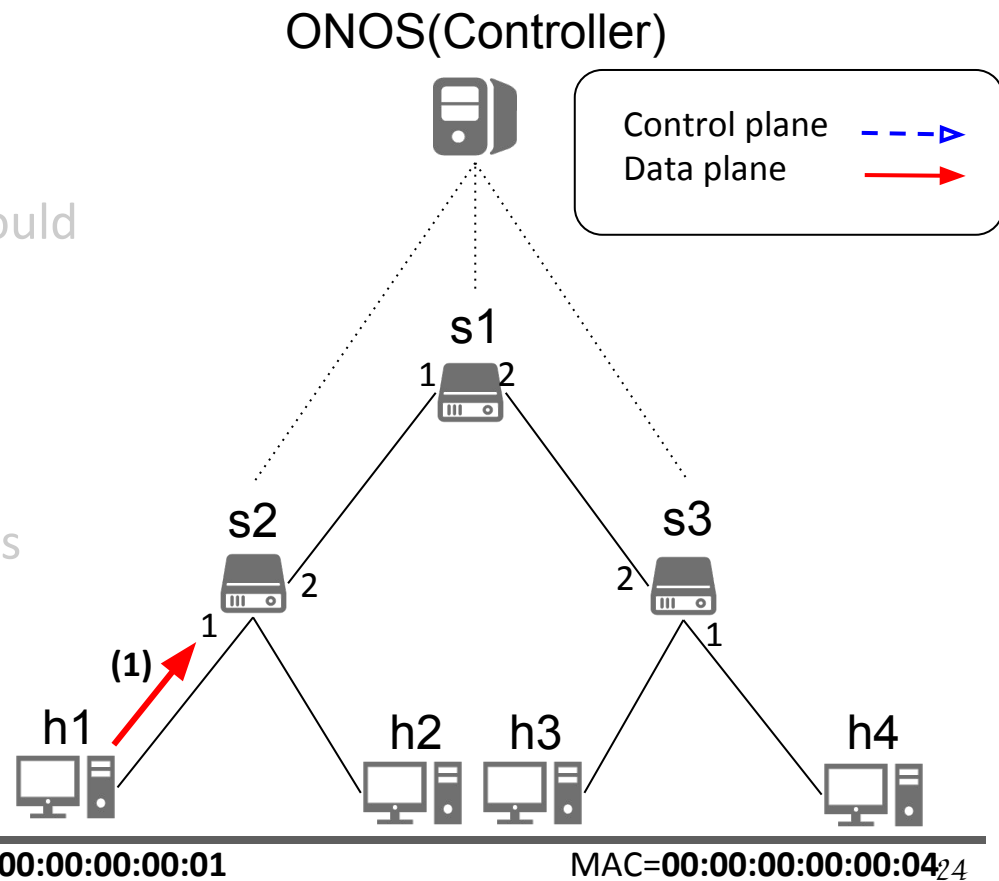


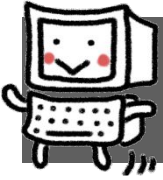


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet
- (5) h4 receives the ICMP packet which is send by h1

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port

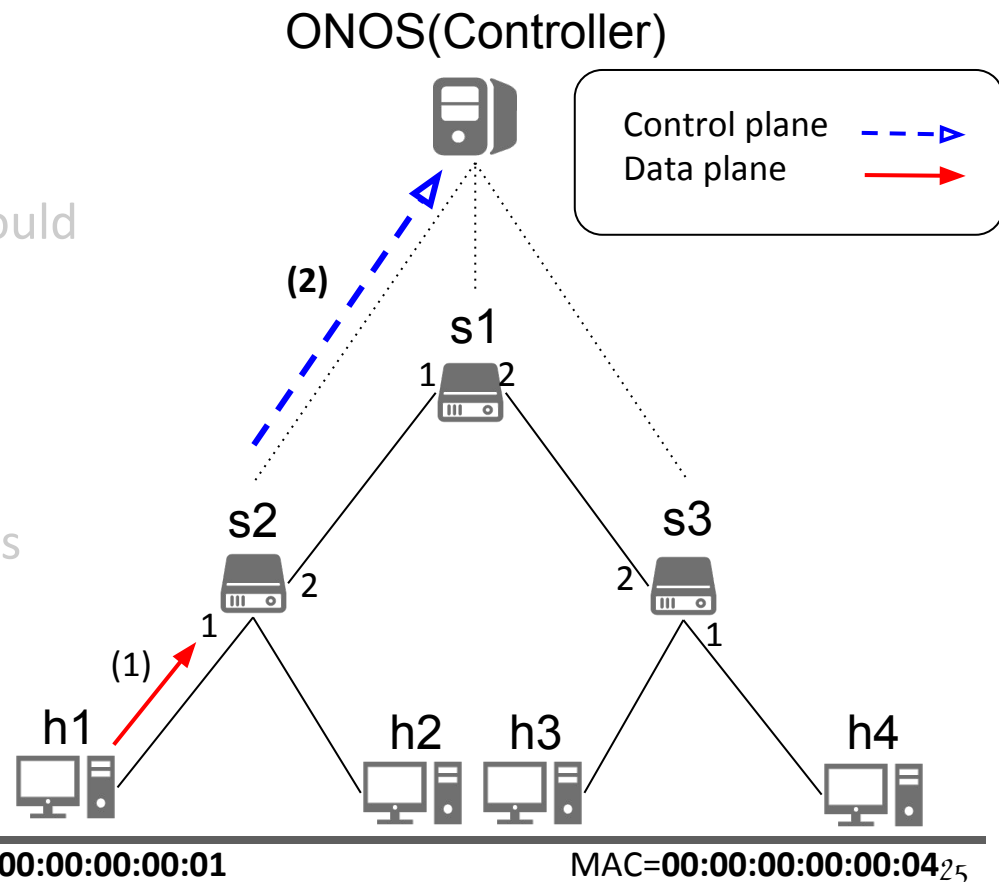




Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h4 receives the ICMP packet which is send by h1

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port

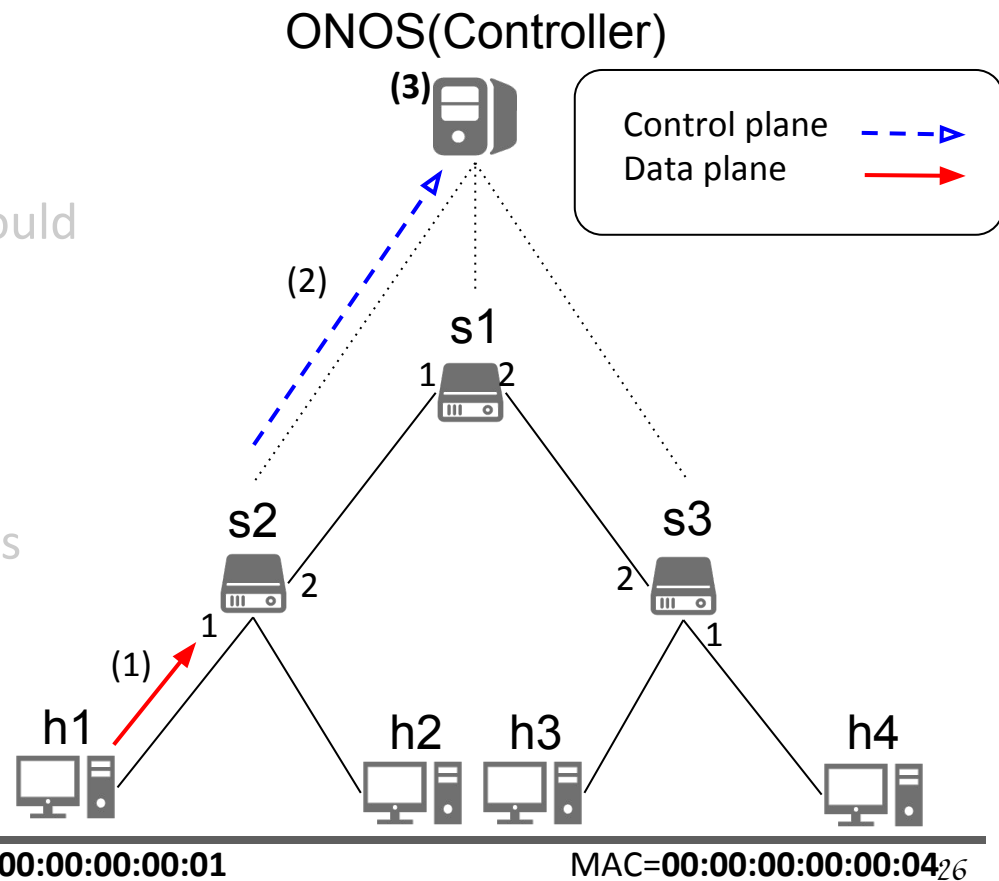


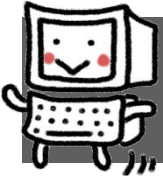


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h4 receives the ICMP packet which is send by h1

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
		00:.....:01	1		

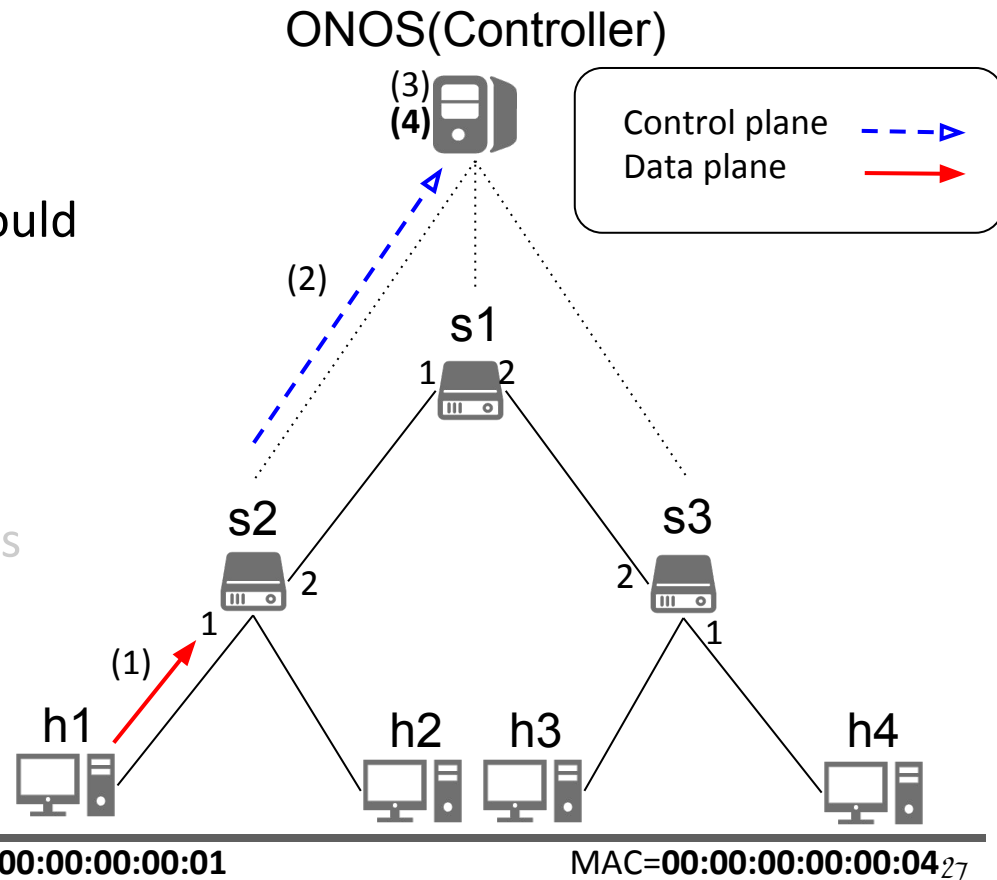


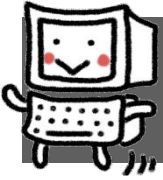


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h4 receives the ICMP packet which is send by h1

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
		00:.....:01	1		

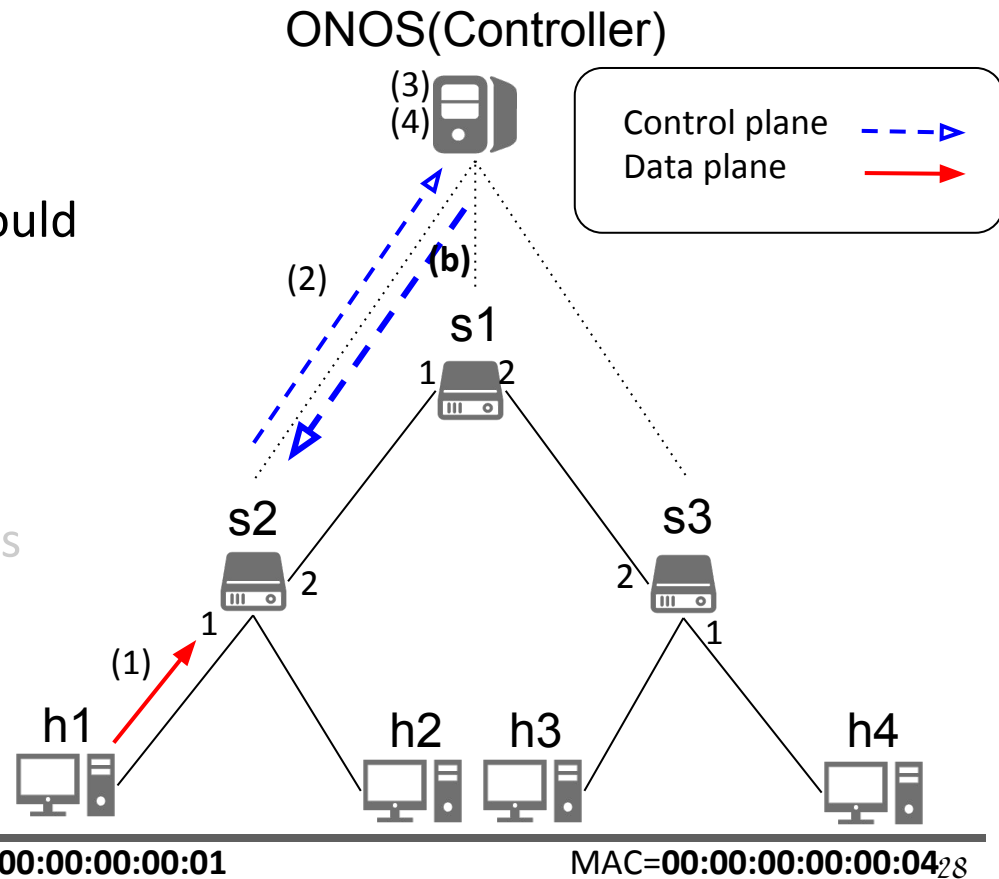


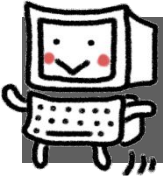


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
		00:.....:01	1		

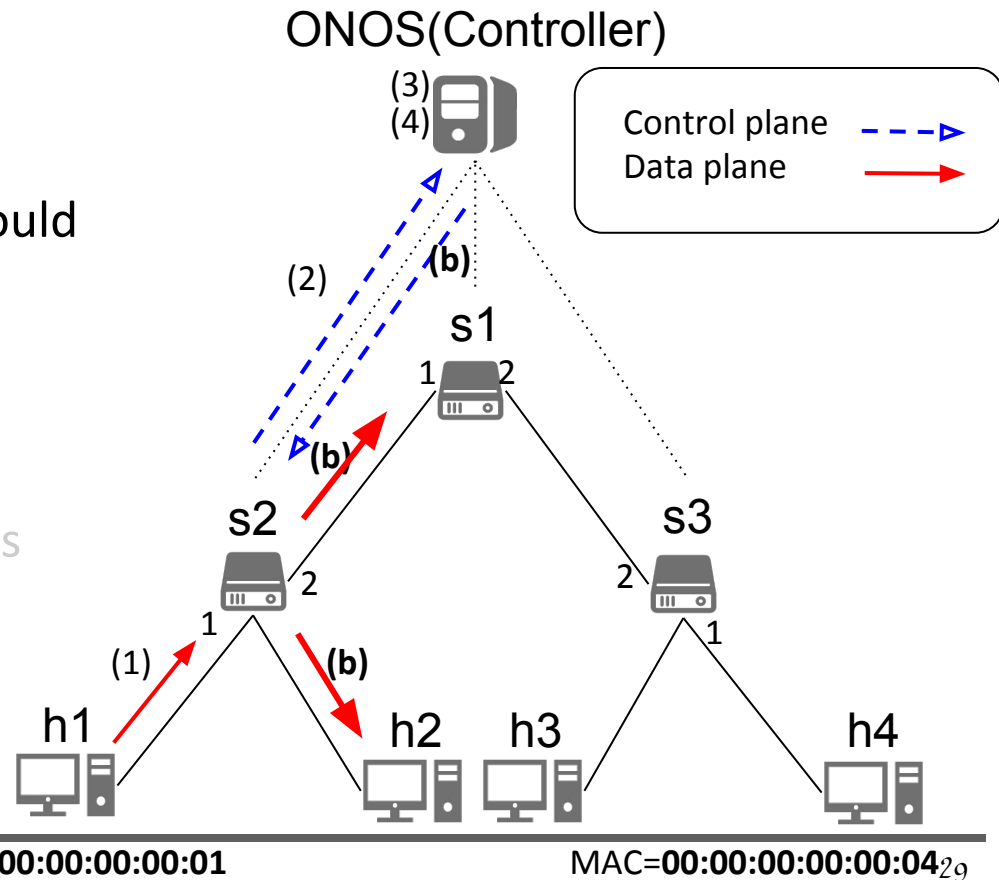




Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
		00:.....:01	1		

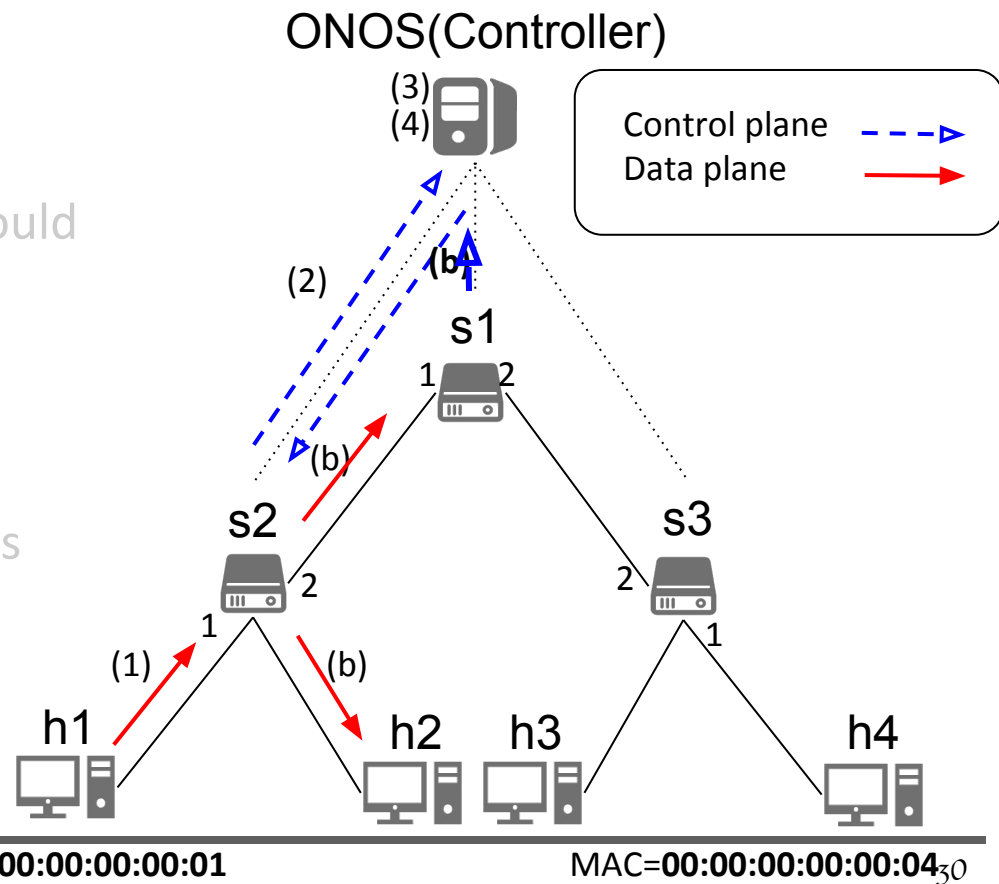


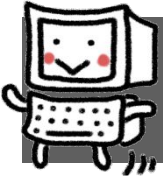


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h4 receives the ICMP packet which is send by h1

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
		00:.....:01	1		



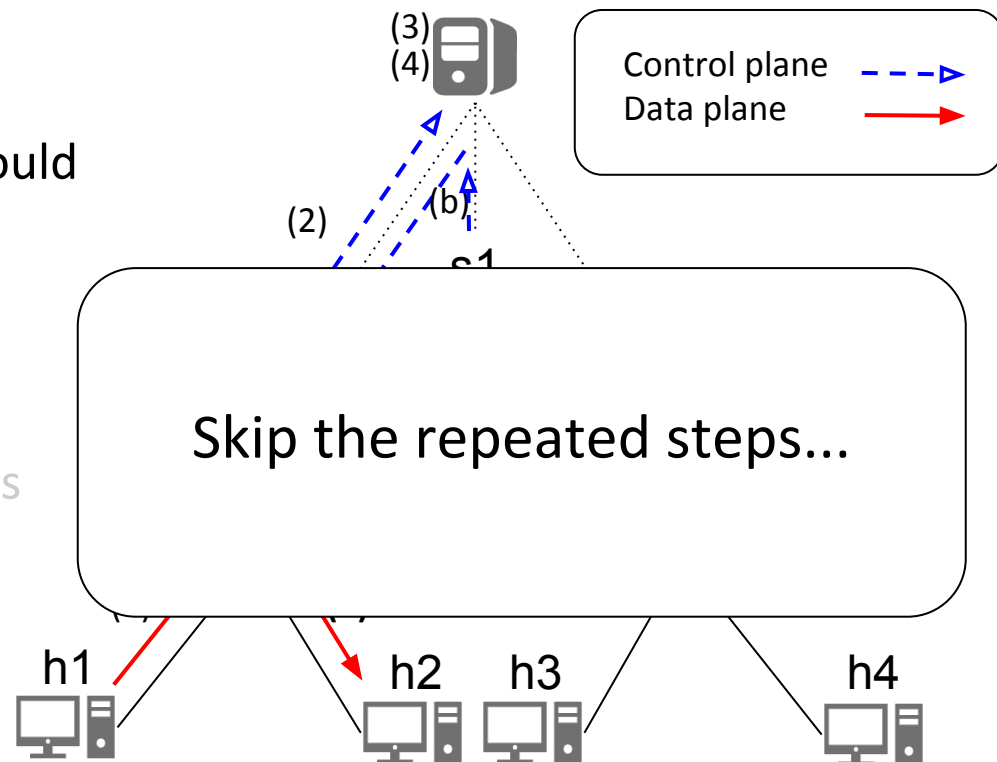


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet
- (5) h4 receives the ICMP packet which is send by h1

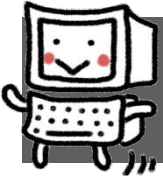
s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
		00:.....:01	1		

ONOS(Controller)



MAC=00:00:00:00:00:01

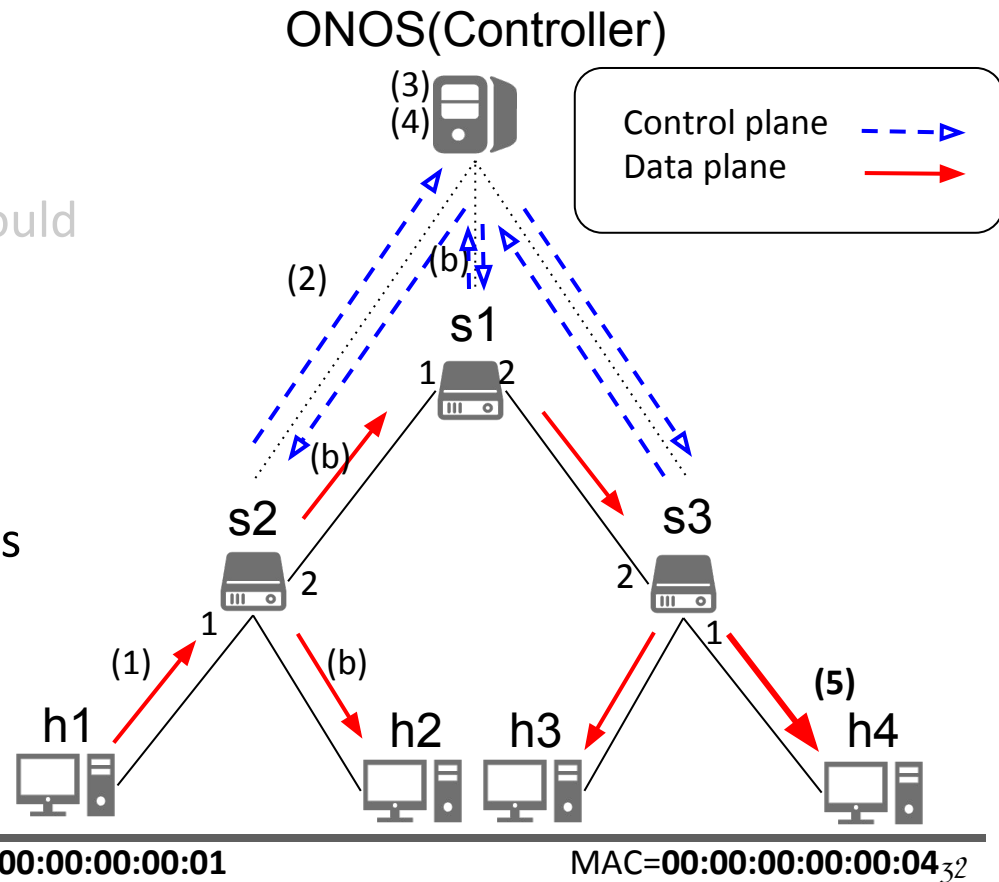
MAC=00:00:00:00:00:04 31

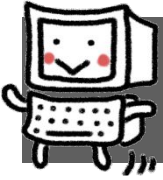


Workflow of Learning Bridge Function

- (1) h1 uses ping to send ICMP request to h4.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2

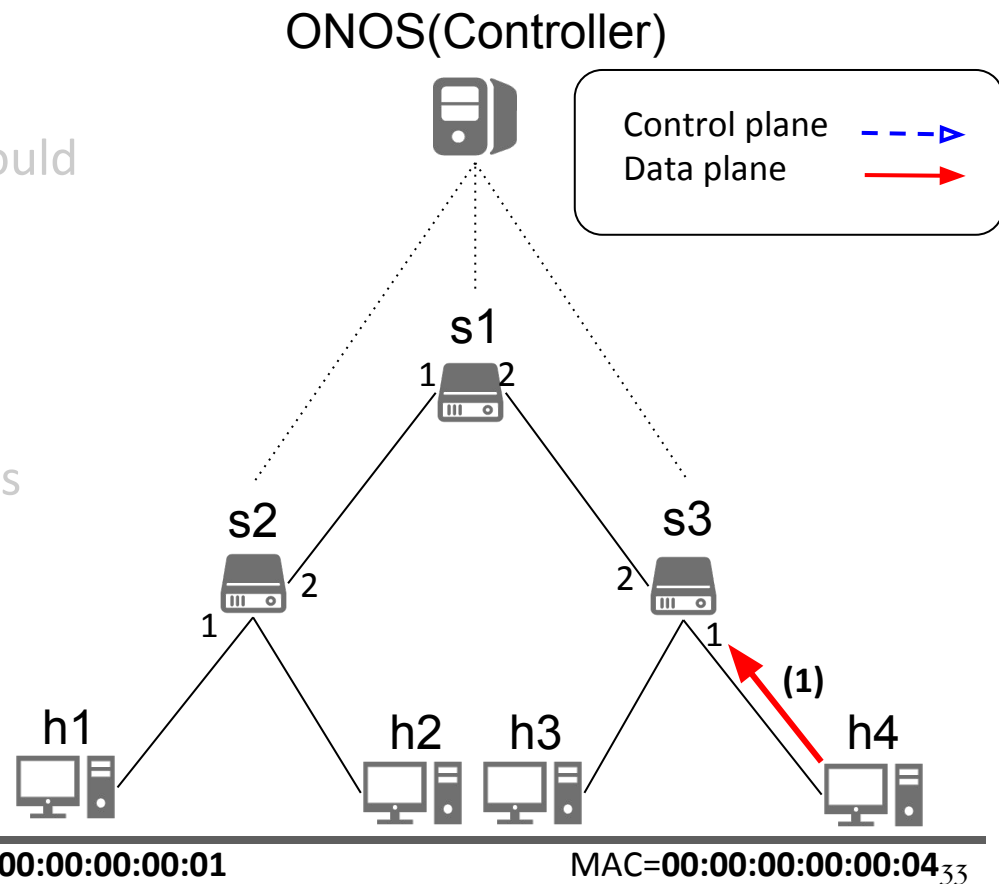




Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2

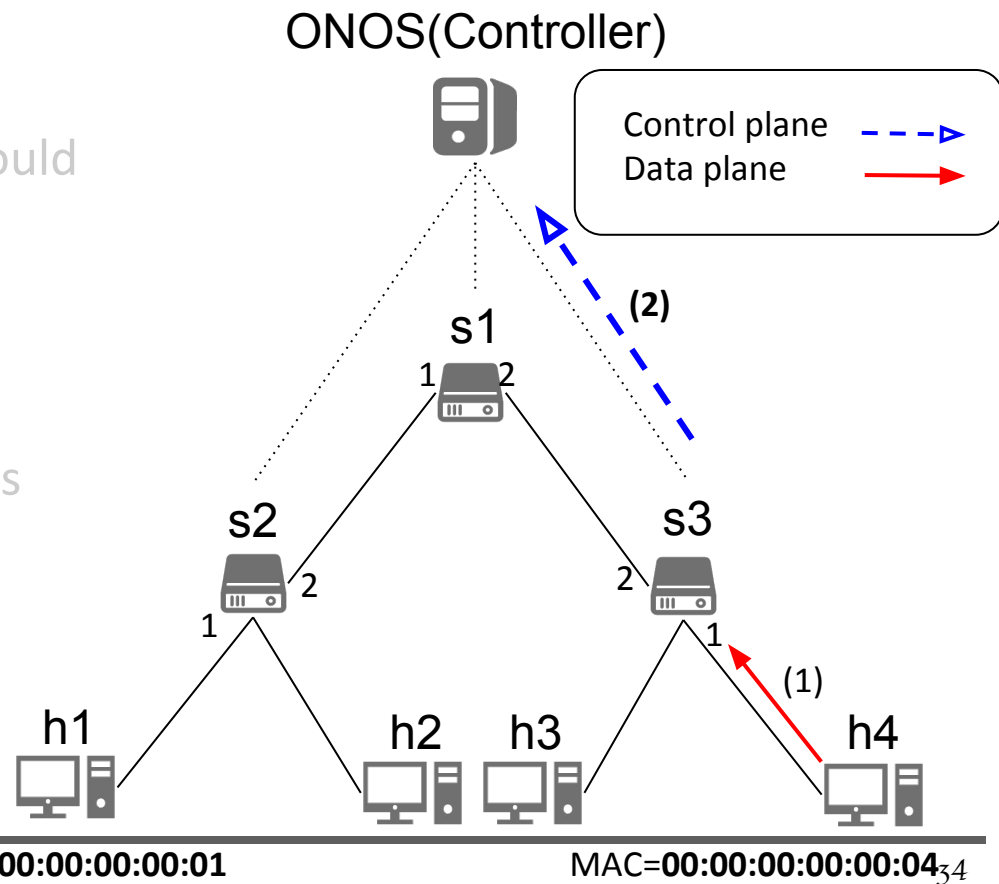


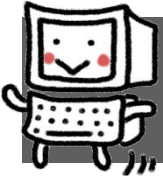


Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2

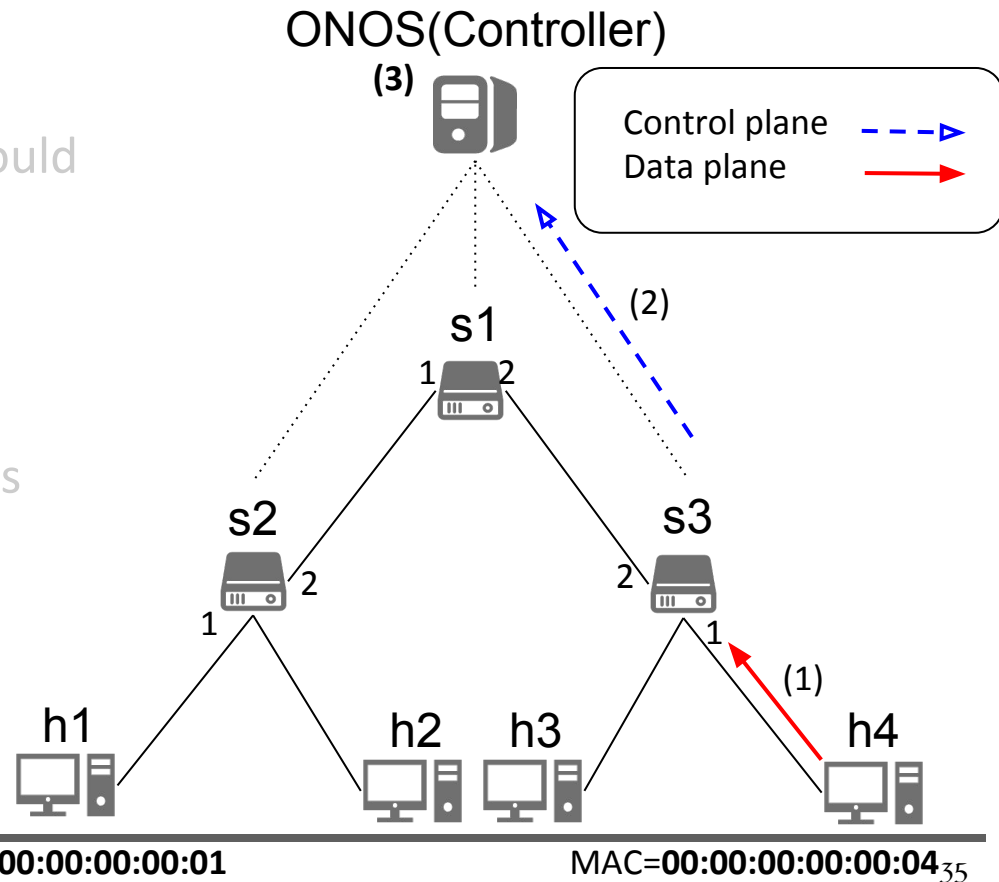


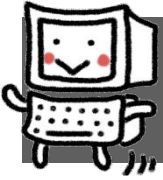


Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2
				00:.....:04	1

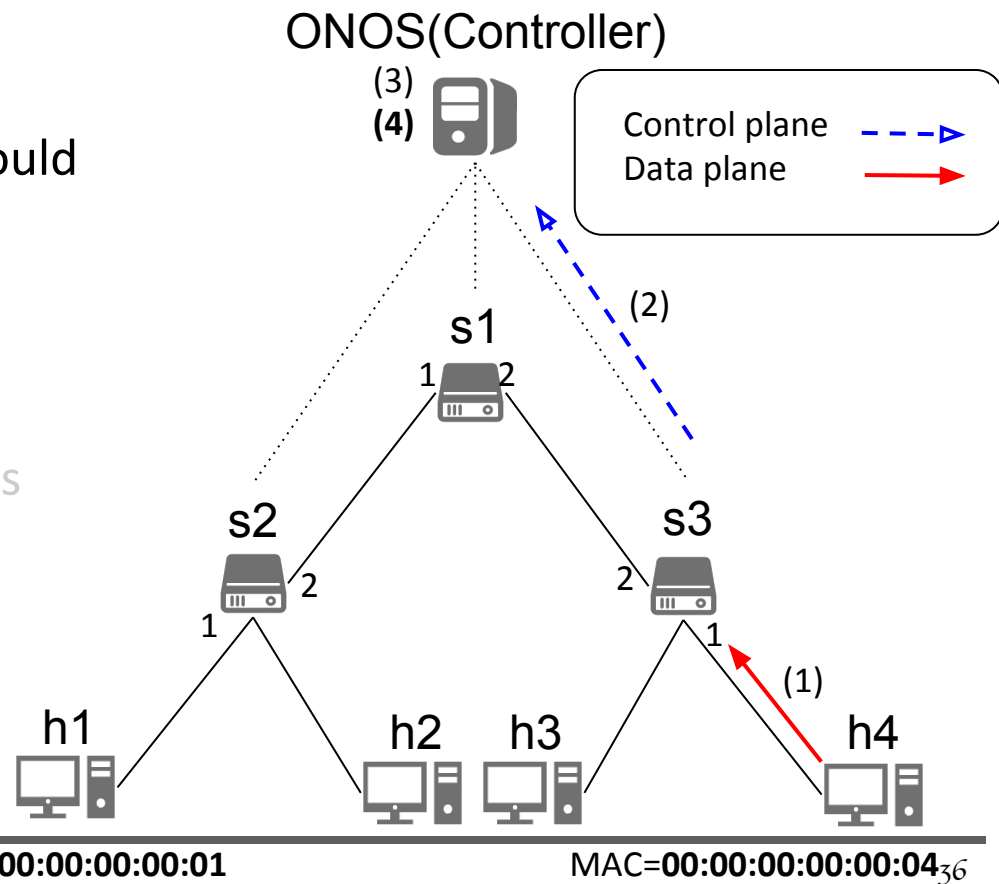


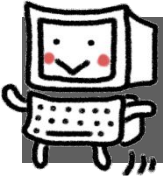


Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2
				00:.....:04	1

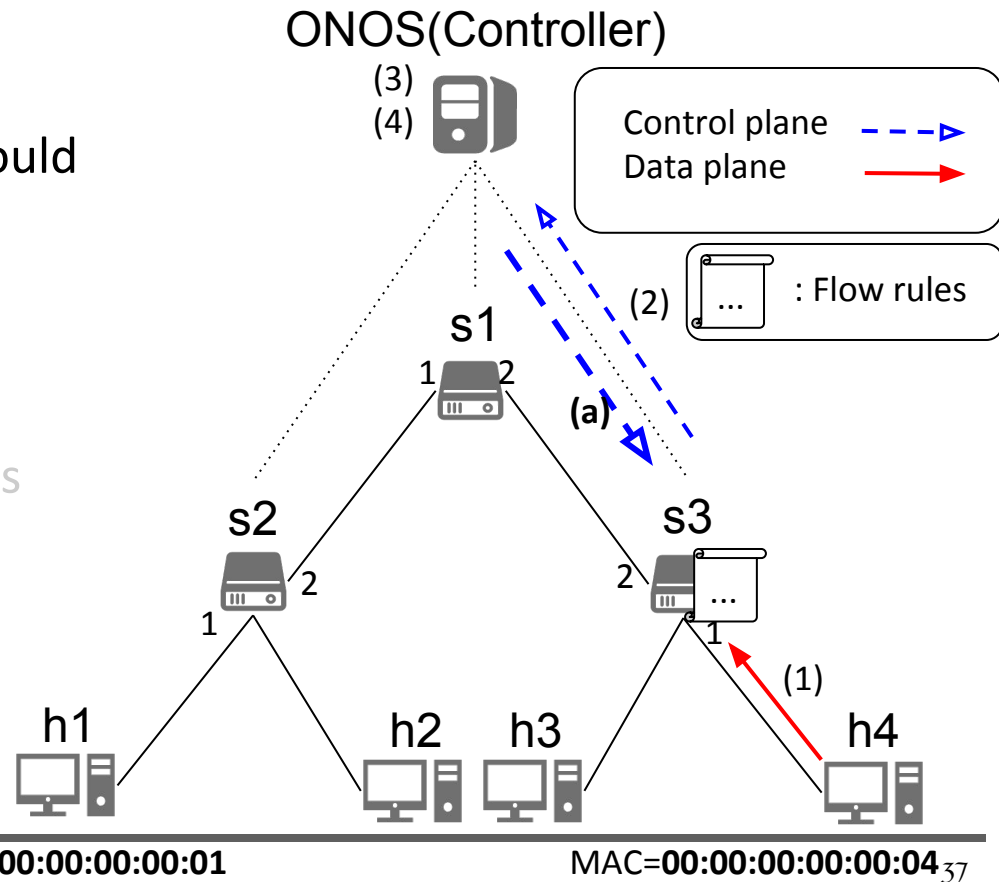




Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2
				00:.....:04	1

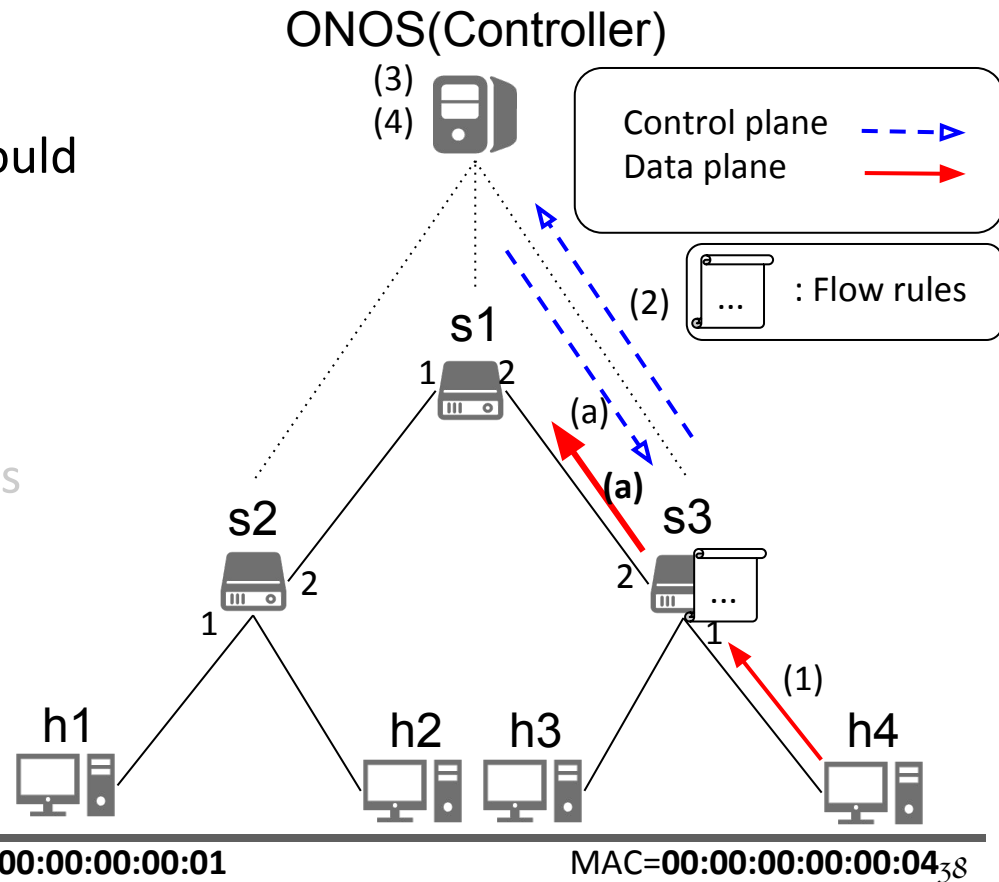


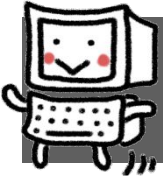


Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2
				00:.....:04	1

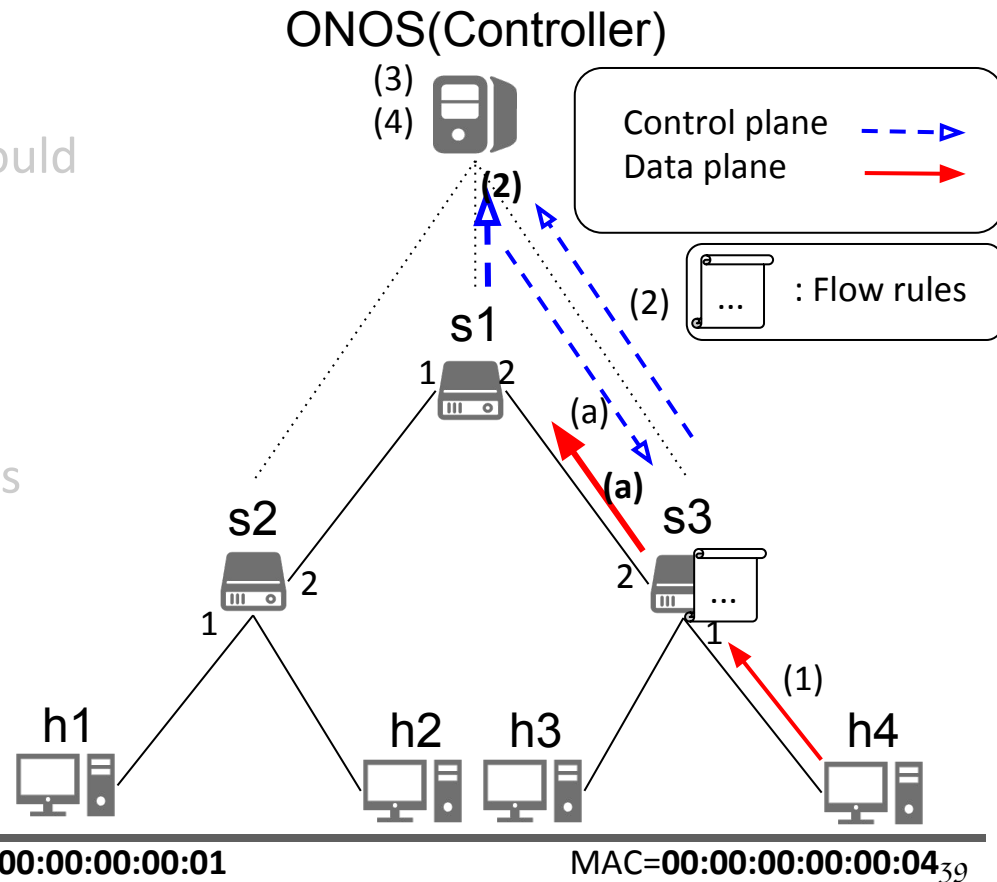


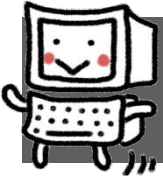


Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2
				00:.....:04	1

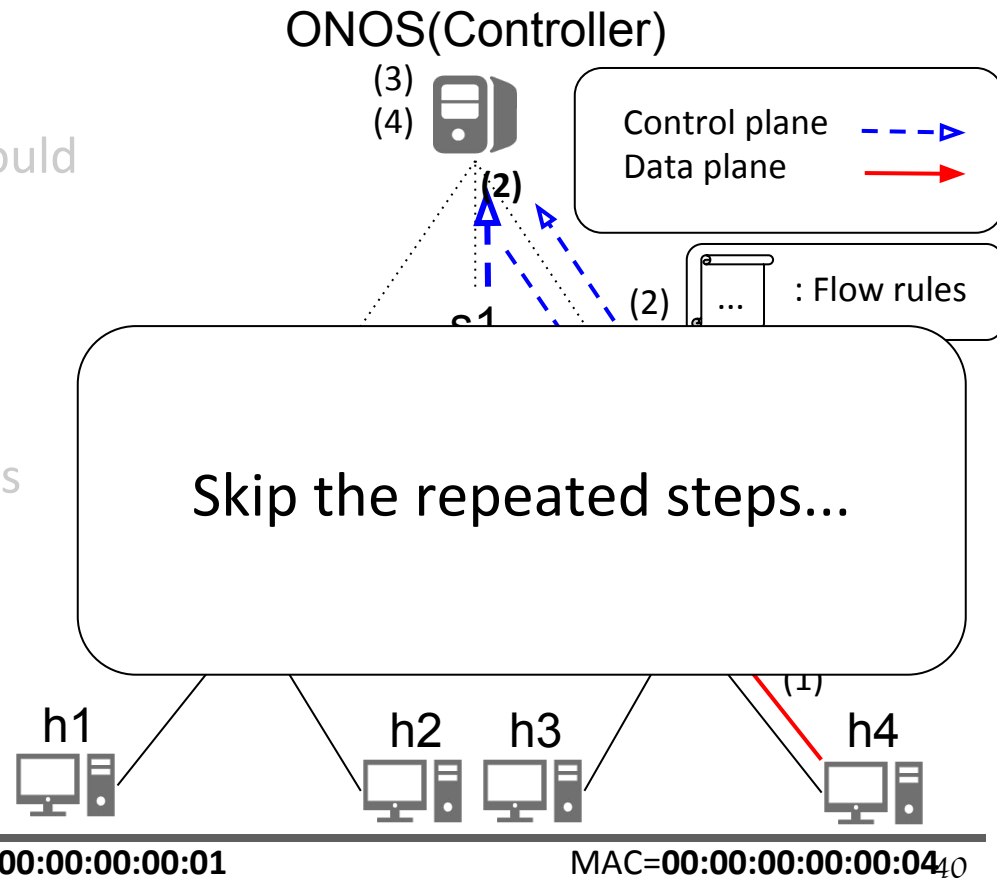




Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit**: install flow rule, then send Packet-out to the switch
 - (b) **Table miss**: flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2
				00:.....:04	1

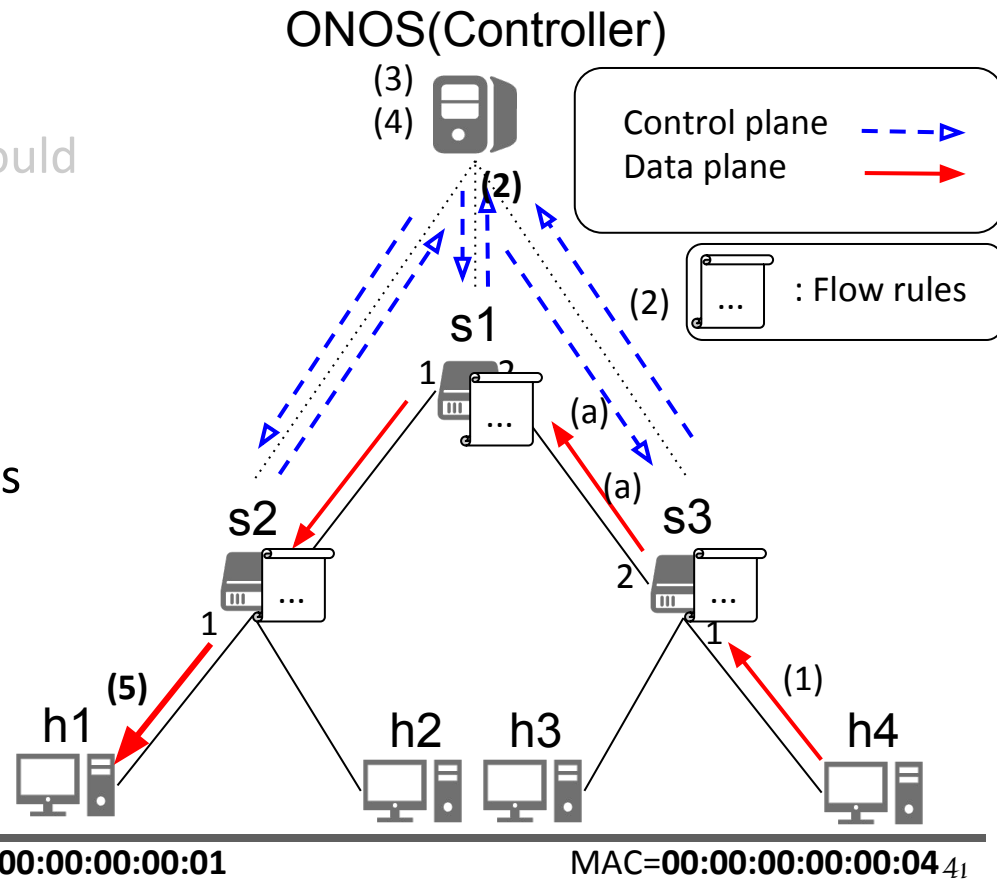


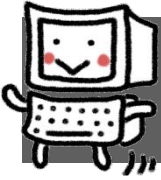


Workflow of Learning Bridge Function

- (1) h4 sends ICMP reply to h1.
- (2) The packet doesn't match any flow rules on the switch. The switch sends Packet-in to the controller.
- (3) Parse the Packet-in and update the MAC address table.
- (4) Look up the MAC address table to decide which port the controller should send to.
 - (a) **Table hit:** install flow rule, then send Packet-out to the switch
 - (b) **Table miss:** flood packet
- (5) h1 receives the ICMP packet which is send by h4

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port
00:.....:01	1	00:.....:01	1	00:.....:01	2
00:.....:04	2	00:.....:04	2	00:.....:04	1





The details of Learning Bridge Function

1. Forwarding information learning

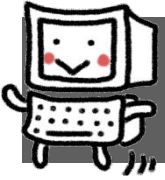
- Associate the source MAC address with incoming port
 - (1) Parse the Packet-in to get the source MAC address.
 - (2) Update the MAC address table (source MAC address \leftrightarrow incoming port) when the controller receives Packet-in.

2. Packets forwarding

- Use destination MAC address as index to look up the MAC address table and forward the packet to the proper output port
 - (1) Parse the Packet-in to get the destination MAC address.
 - (2) Check if the destination MAC address is in the MAC address table.
 - Table hit:
 - I. Install ***flow rule** on the switch which send Packet-in.
 - II. Send Packet-out to the port (related to the destination MAC address) on the switch which send Packet-in.
 - Table miss:
 - Send Packet-out to every port on the switch which send Packet-in, except for the Packet-in port. (This step is also known as “flood packet”)

***flow rule**

Match	src MAC dst MAC
Action	set output port
Priority	10
Timeout	10



Outline

- ☐ Overview
- ☐ Create ONOS Application
- ☐ Learning Bridge Function
- ☒ Project 3 Requirements



Developing ONOS Applications

Project 3 Requirements



Part 1: Create an ONOS application

❑ Maven project naming convention

- Incorrect naming convention or format subjects to not scoring
 - <groupId>: **nctu.winlab**
 - <artifactId>: **bridge-app**
 - <version>: **(arbitrary)**
 - <package>: **nctu.winlab.bridge**

```
sdnfv@sdnfv-VirtualBox:~/bridge-app$ tree
.
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── nctu
│   │   │   │   ├── winlab
│   │   │   │   │   ├── bridge
│   │   │   │   │   │   ├── AppComponent.java
│   │   │   │   │   │   └── SomeInterface.java
│   │   └── test
│   │       ├── java
│   │       │   ├── nctu
│   │       │   │   ├── winlab
│   │       │   │   │   ├── bridge
│   │       │   │   │   │   └── AppComponentTest.java
└──
```

11 directories, 4 files



Part 2: Learning Bridge Function

- ❑ **Learning Bridge & Forwarding Packet**
 - a. Learning Bridge Function with **tree** (depth=2) topology
 - b. Learning Bridge Function with **tree** (depth=3~5) topology

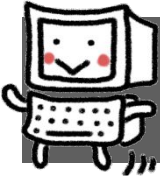
```
$ sudo mn --controller=remote,ip=127.0.0.1:6653 --topo=tree,depth=2
```

- ❑ Ping should work between all hosts

```
mininet> pingall
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

(Ex. mininet **tree**
topology with depth=2)

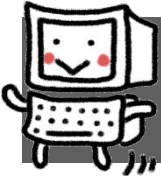


Other Requirements (1/2)

☐ ONOS Applications activation

- You are only allowed to activate your **bridge-app** and the following ONOS applications:

```
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
* 156 org.onosproject.openflow           2.2.0    OpenFlow Provider Suite
* 172 org.onosproject.gui2               2.2.0    ONOS GUI2
```



Other Requirements (2/2)



Flow rule regulation

- Match field (selector): **ETH_SRC, ETH_DST**
- Action field (treatment): **OUTPUT**
- Flow priority: **10**
- Flow timeout: **10**



DO NOT use ONOS built-in apps to install flow rules. (e.g. fwd, proxyarp), you are only allowed to use Java API FlowObjective or FlowRule to install flow rules on the network devices

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	69	5	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1	0	10	0	ETH_DST:A2:F1:7D:87:3B:2D, ETH_SRC:6A:AC:C0:78:16:66	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge
Added	1	0	10	0	ETH_DST:6A:AC:C0:78:16:66, ETH_SRC:A2:F1:7D:87:3B:2D	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	3	0	10	0	ETH_DST:B2:45:45:F1:52:7C, ETH_SRC:A2:F1:7D:87:3B:2D	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	3	0	10	0	ETH_DST:A2:F1:7D:87:3B:2D, ETH_SRC:B2:45:45:F1:52:7C	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge
Added	3	0	10	0	ETH_DST:7E:33:57:61:0E:35, ETH_SRC:6A:AC:C0:78:16:66	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge
Added	3	0	10	0	ETH_DST:6A:AC:C0:78:16:66, ETH_SRC:7E:33:57:61:0E:35	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	3	17	10	0	ETH_DST:7E:33:57:61:0E:35, ETH_SRC:B2:45:45:F1:52:7C	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge
Added	3	17	10	0	ETH_DST:B2:45:45:F1:52:7C, ETH_SRC:7E:33:57:61:0E:35	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	20	69	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	66	3,965	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,558	3,965	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core



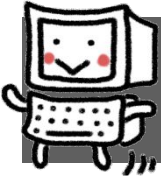
Project 3 Scoring Criteria

- ☐ (30%) Create ONOS application
- ☐ (30%) Learning Bridge Function with tree(depth=2) topology
- ☐ (20%) Learning Bridge Function with tree(depth=2~5) topology
- ☐ (10%) Flow rule regulation
- ☐ (10%) Submission naming convention



Hints

- ☐ Make sure to Packet-out when you send Flow-mod
 - Since flow modification message only install flow rule on the switch
- ☐ Make sure to cancel request for Packet-in when you deactivate your app
- ☐ Use Java API FlowObjective or FlowRule to send Flow-mod
 - You can trace *ReactiveForwarding.java* to find out how can we use Java API to install flow rules
- ☐ How to debug:
 - (1) Use Logger (Java API) to print out some information on your terminal
 - (2) Use Wireshark to capture your packet



About Submission

Files

- You need to submit all files under your project directory.
- Zip the whole **bridge-app** folder into a .zip file.
 - Named: **project3_<studentID>.zip**

```
sdnfv@sdnfv-VirtualBox:~/bridge-app$ tree
.
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── nctu
│   │   │   │   ├── winlab
│   │   │   │   │   ├── bridge
│   │   │   │   │   │   ├── AppComponent.java
│   │   │   │   │   │   └── SomeInterface.java
│   │   └── test
│   │       ├── java
│   │       │   ├── nctu
│   │       │   │   ├── winlab
│   │       │   │   │   ├── bridge
│   │       │   │   │   │   └── AppComponentTest.java
└──
```

11 directories, 4 files

Submit

- Upload “.zip” file to New e3
 - Named: **project3_<studentID>.zip**
- Your project will not be scored if you type wrong file name or wrong format.



Q & A

Thank you



References

- ❑ ONOS Reactive Forwarding application
 - <https://github.com/opennetworkinglab/onos/blob/master/apps/fwd/src/main/java/org/onosproject/fwd/ReactiveForwarding.java>
- ❑ ONOS Java API
 - <http://api.onosproject.org/2.2.0/apidocs/>



Appendix - Reactive Forwarding Application

- ☐ Application Entry Point
 - activate()

- ☐ Request Packet-in
 - requestIntercepts()

- ☐ Handle Packet-in
 - PacketProcessor{process()}

- ☐ Packet-out
 - packetOut() // PacketContext.setOutput().send()

- ☐ Flow-mod
 - installRule() // FlowObjectiveService.forward()

Refer to ONOS v2.2 ReactiveForwarding