



Lab 3

ONOS Application Development:
SDN-enabled Learning Bridge and Proxy ARP

助教: 惠品嘉 黃琮閔

sdnta@win.cs.nycu.edu.tw

Deadline: 2024/10/16 23:59



Outline

- Build ONOS Application Project
 - Environment Setup
 - Create an ONOS Application
 - Build, Install, Activate, and Reinstall ONOS Application
- ARP
 - Introduction
 - ARP Request/Reply Format
- Learning Bridge Function
 - Introduction
 - Workflow
- Proxy APR
 - Introduction
 - Workflow
- Lab 3 Requirements

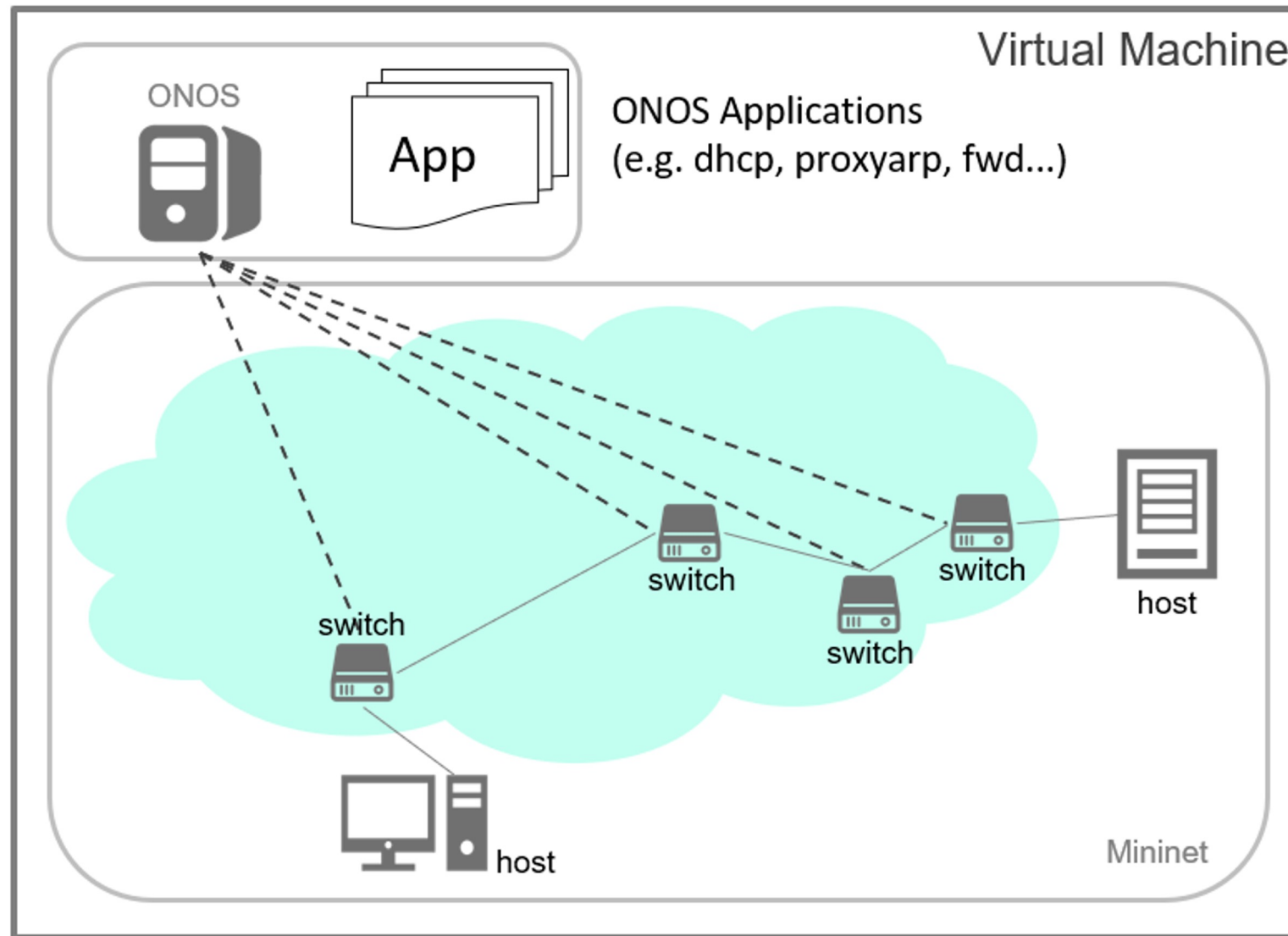


Outline

- Build ONOS Application Project
 - Environment Setup
 - Create an ONOS Application
 - Build, Install, Activate, and Reinstall ONOS Application
- ARP
 - Introduction
 - ARP Request/Reply Format
- Learning Bridge Function
 - Introduction
 - Workflow
- Proxy APR
 - Introduction
 - Workflow
- Lab 3 Requirements



Overview





JDK Installation

1. Download the “install_jdk” script from [E3](#).
2. Add execution permission to the script:

```
$ chmod +x install_jdk
```

3. Execute the script:

```
$ ./install_jdk
```

4. Once the installation finishes, you will see a success message:

```
Setting up zulu11-jdk (11.0.16.1-1) ...
*****
* Installation of Azul Zulu JDK 11 finished successfully! *
*****
```

5. Check the installed JDK version:

```
$ java -version
```

```
demo@SDN-NFV:~$ java -version
openjdk version "11.0.16.1" 2022-07-19 LTS
OpenJDK Runtime Environment Zulu11.58+23-CA (build 11.0.16.1+1-LTS)
OpenJDK 64-Bit Server VM Zulu11.58+23-CA (build 11.0.16.1+1-LTS, mixed mode)
```



Apache Maven

- A software project management and comprehension tool.
- Based on the concept of a Project Object Model (POM).
- Manage a project's build, reporting and documentation from a **central piece** of information.
- It has been installed in your VM by the “env_setup” script in Lab 1.
- Official website: <https://maven.apache.org/>



Build ONOS Application Archetypes

- We will use ***onos-create-app*** command to generate an ONOS application template.
- ***onos-create-app*** command relies on the ONOS archetypes.
- We need to build ONOS archetypes first.
- Steps:
 - Specify ONOS version:

```
$ export ONOS_POM_VERSION=2.7.0
```
 - Build archetypes:

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

 - ***-DskipTests***: Skip running tests of the project.



Build ONOS Application Template

- Run *onos-create-app*.

```
$ onos-create-app
...
[INFO] ...
Define value for property 'groupId': nycu.winlab
Define value for property 'artifactId': bridge-app
Define value for property 'version' 1.0-SNAPSHOT: : <enter>
Define value for property 'package' nycu.winlab: : nycu.winlab.bridge

Confirm properties configuration:
onosVersion: 2.7.0
groupId: nycu.winlab
artifactId: bridge-app
version: 1.0-SNAPSHOT
package: nycu.winlab.bridge
Y: : <enter>
[INFO] ...
...
[INFO] BUILD SUCCESS
```

→ Archive ID for the created ONOS application.



Folder Structure of Created ONOS Application

- *onos-create-app* command creates a folder named **bridge-app** (artifactId).
- Structure of **bridge-app** folder:

```
demo@SDN-NFV:~/bridge-app$ tree
.
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── nctu
│   │   │   │   ├── winlab
│   │   │   │   │   ├── bridge
│   │   │   │   │   │   ├── AppComponent.java
│   │   │   │   │   │   ├── package-info.java
│   │   │   │   │   │   └── SomeInterface.java
│   │   └── test
│   │       ├── java
│   │       │   ├── nctu
│   │       │   │   ├── winlab
│   │       │   │   │   ├── bridge
│   │       │   │   │   │   └── AppComponentTest.java
└── test
```

11 directories, 5 files



Modify ONOS Application Properties

- Modify Project Object Model file **pom.xml** to describe your project.

pom.xml
Before

```
34     <properties>
35     <!-- Uncomment to generate ONOS app from this module.
36         <onos.app.name>org.foo.app</onos.app.name>
37         <onos.app.title>Foo App</onos.app.title>
38         <onos.app.origin>Foo, Inc.</onos.app.origin>
39         <onos.app.category>default</onos.app.category>
40         <onos.app.url>http://onosproject.org</onos.app.url>
41         <onos.app.readme>ONOS OSGi bundle archetype.</onos.app.readme>
42     <!-->
43     </properties>
```

pom.xml
After

```
34     <properties>
35         <onos.app.name>nctu.winlab.bridge</onos.app.name>
36         <onos.app.title>Learning Bridge App</onos.app.title>
37         <onos.app.origin>Winlab, NCTU</onos.app.origin>
38         <onos.app.category>default</onos.app.category>
39         <onos.app.url>http://onosproject.org</onos.app.url>
40         <onos.app.readme>ONOS OSGi bundle archetype.</onos.app.readme>
41     </properties>
```



Overview of AppComponent.java

- AppComponent.java code template.

```
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(), clear: false);  
        log.info("Stopped");  
    }  
  
    @Modified  
    public void modified(ComponentContext context) {  
        Dictionary<?, ?> properties = context != null ? context.getProperties()  
        if (context != null) {  
            someProperty = get(properties, propertyName: "someProperty");  
        }  
        log.info("Reconfigured");  
    }  
  
    @Override  
    public void someMethod() { log.info("Invoked"); }  
}
```

Inject a dependent service in ONOS Core.

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

Executed when app activated.

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

Executed when app deactivated.

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



Build, Install and Activate ONOS Application

- Build ONOS application:

```
# In the root of your application folder.  
$ mvn clean install -DskipTests
```

```
demo@SDN-NFV:~/bridge-app$ ls target  
bridge-app-1.0-SNAPSHOT.jar      checkstyle-header.txt      generated-test-sources  
bridge-app-1.0-SNAPSHOT.oar      checkstyle-result.xml      maven-archiver  
bridge-app-1.0-SNAPSHOT-tests.jar checkstyle-suppressions.xml maven-status  
checkstyle-cachefile            classes                    oar  
checkstyle-checker.xml          generated-sources          test-classes
```

} built results

- Run ONOS:

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

- Install and activate ONOS application:

```
# In the root of your application folder.  
$ onos-app localhost install! target/<artifactId>-<version>.oar  
                                     (bridge-app-1.0-SNAPSHOT.oar)
```

- **install!**: Install and activate application immediately.



Reinstall ONOS Application

If you modify your application, you need to rebuild and reinstall it on ONOS.

1. Rebuild application of new version:

```
# In the root of your application folder.  
$ mvn clean install -DskipTests
```

2. Deactivate application of old version on ONOS:

```
$ onos-app localhost deactivate <onos.app.name>
```

- **<onos.app.name>** is set in your pom.xml. e.g. nycu.winlab.bridge

3. Uninstall application of old version:

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

4. Install and activate application of new version:

```
# In the root of your application folder.  
$ onos-app localhost install! target/<artifactId>-<version>.oar
```



References

- [Install Azul Zulu on Debian-based Linux](#)
- [ONOS Wiki – Template Application Tutorial](#)
- [ONOS Application Subsystem](#)
- [ONOS Java API \(2.7.0\)](#)



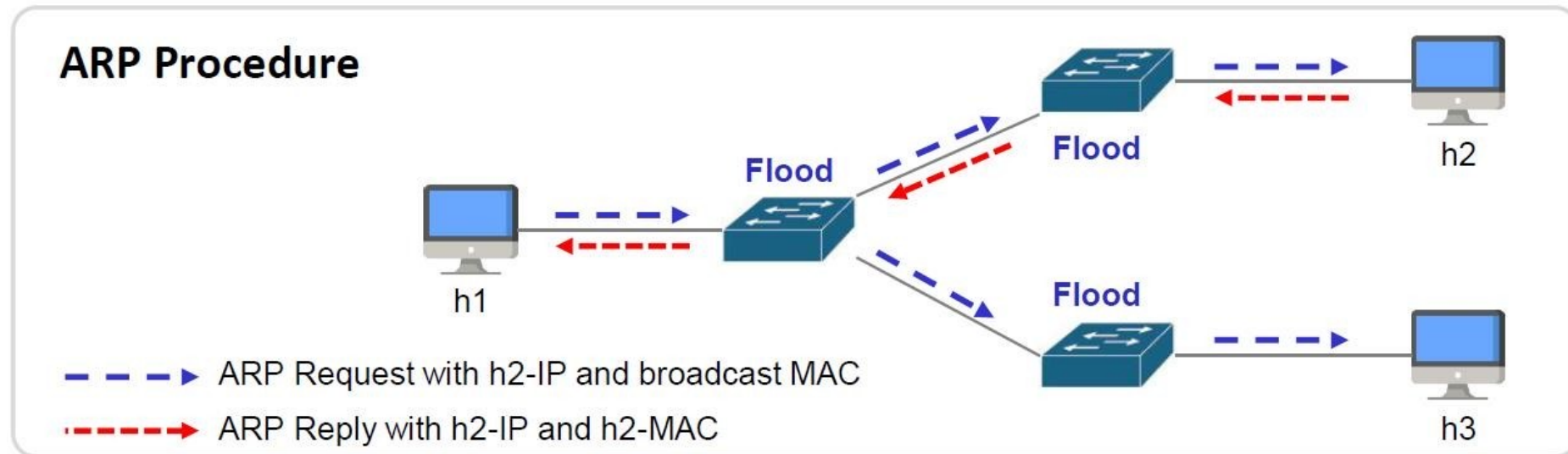
Outline

- Build ONOS Application Project
 - Environment Setup
 - Create an ONOS Application
 - Build, Install, Activate, and Reinstall ONOS Application
- ARP
 - Introduction
 - ARP Request/Reply Format
- Learning Bridge Function
 - Introduction
 - Workflow
- Proxy APR
 - Introduction
 - Workflow
- Lab 3 Requirements



What is Address Resolution Protocol (ARP)

- Used to discover Link Layer address (e.g. MAC) with the given Network Layer address (e.g. IPv4)
- Use flooding to discover devices
 - Destination Ethernet address of ARP Request is broadcast address
- Hosts maintain an ARP table for mapping IP address to MAC





ARP Request Packet Format

- e.g. h1 sends ARP request to h2

Hardware Type (Ethernet = 1)		Protocol Type (IPv4 = 0x0800)
Hardware Length (Ethernet = 6)	Protocol Length (IPv4 = 4)	Operation Code (Request = 0x1)
Sender Hardware Address (h1-MAC)		
Sender Protocol Address (h1-IP)		
Target Hardware Address (00:00:00:00:00:00)		
Target Protocol Address (h2-IP)		



ARP Reply Packet Format

- e.g. h2 sends ARP reply to h1

Hardware Type (Ethernet = 1)		Protocol Type (IPv4 = 0x0800)
Hardware Length (Ethernet = 6)	Protocol Length (IPv4 = 4)	Operation Code (Reply = 0x2)
Sender Hardware Address (h2-MAC)		
Sender Protocol Address (h2-IP)		
Target Hardware Address (h1-MAC)		
Target Protocol Address (h1-IP)		



Outline

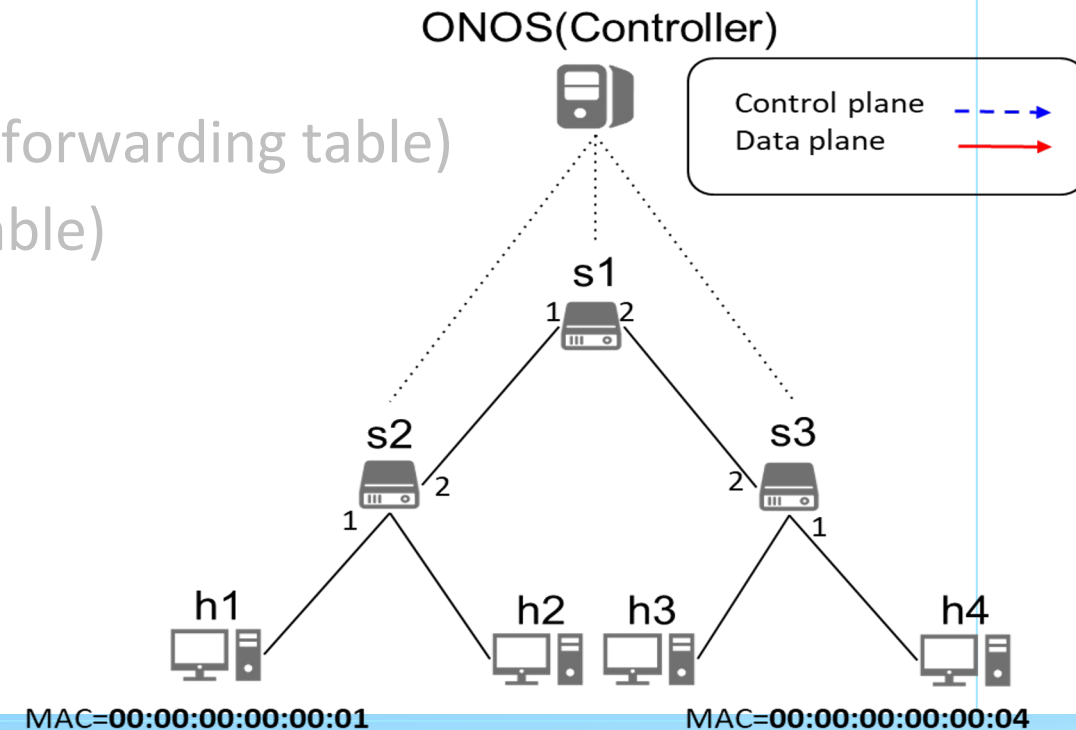
- Build ONOS Application Project
 - Environment Setup
 - Create an ONOS Application
 - Build, Install, Activate, and Reinstall ONOS Application
- ARP
 - Introduction
 - ARP Request/Reply Format
- Learning Bridge Function
 - Introduction
 - Workflow
- Proxy ARP
 - Introduction
 - Workflow
- Lab 3 Requirements



Learning Bridge Functionality

- Switch functionality:
 - When receives a packet, matches Destination MAC
 - Matched: Forwards packet via specified port
 - Not matched: Packet-in
- ONOS App functionality:
 - When receives a Packet-in
 - Records Source MAC and incoming port (in forwarding table)
 - Looks up Destination MAC (in forwarding table)
 - a. Not found:
Floods Packet-out.
 - b. Found:
Sends Packet-out via designated port.
Installs flow rule on switch.

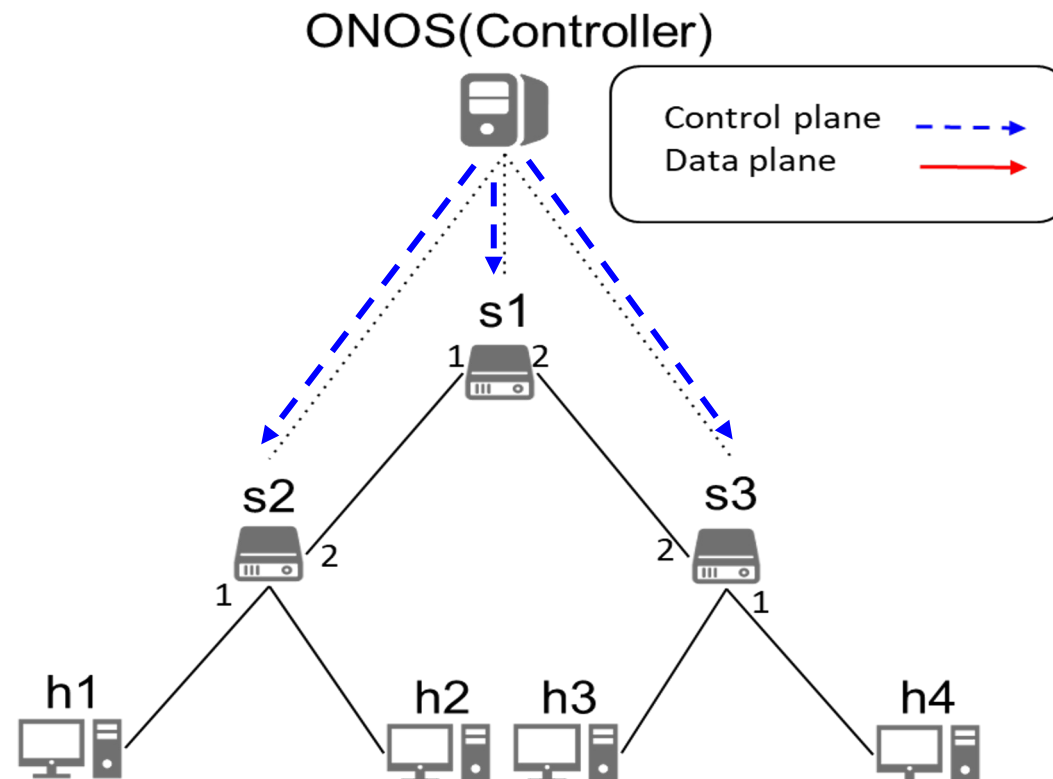
s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port





Request for Packet-in

- When App is activated, it installs a rule on each switch.
 - To request Packet-in for IPv4 packets.
 - With very low priority.
- Don't forget to cancel the request for Packet-in when your App is deactivated.

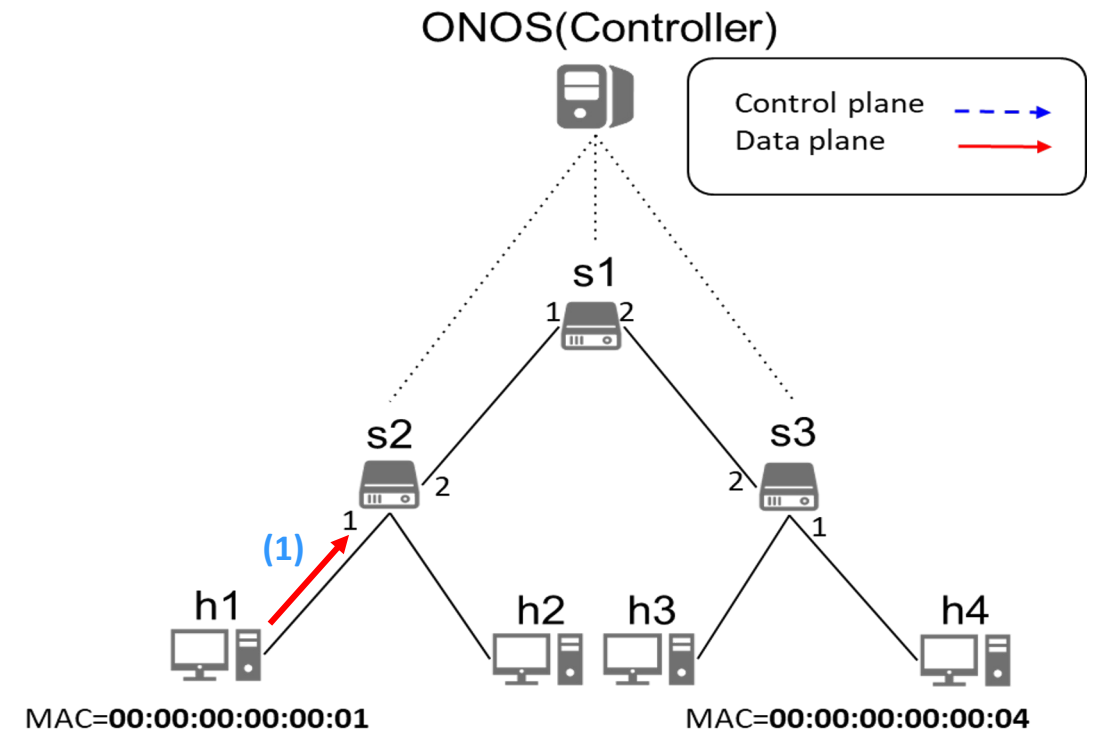




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port

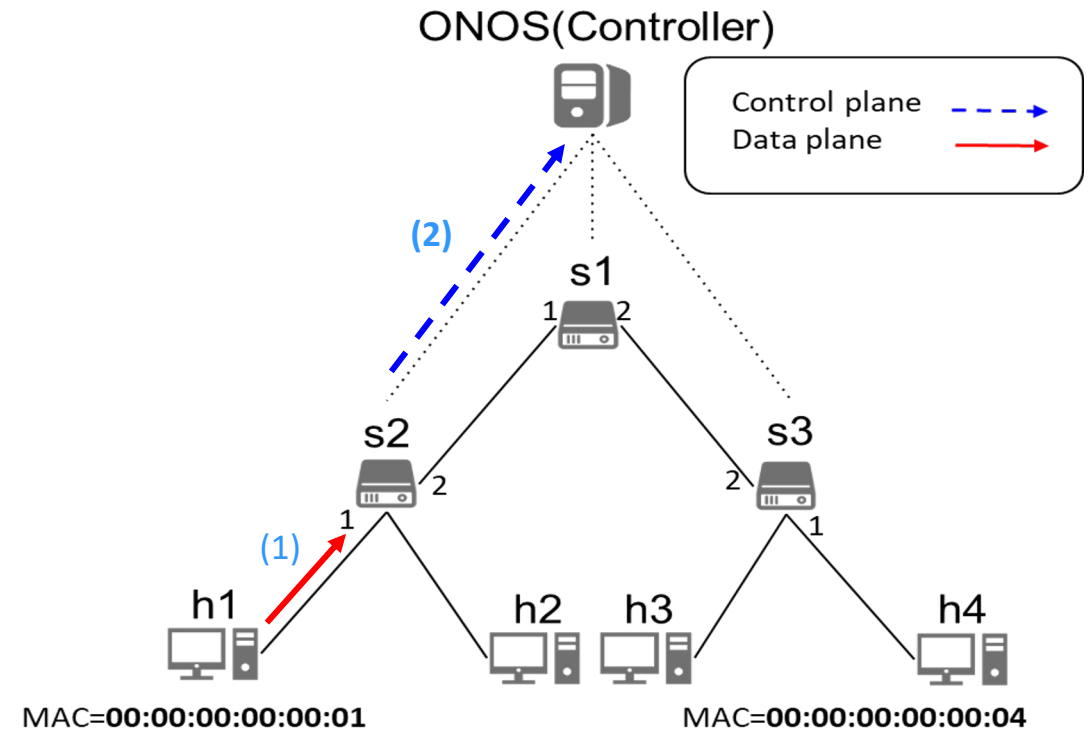




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch (s2) sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

s1		s2		s3	
MAC	Port	MAC	Port	MAC	Port

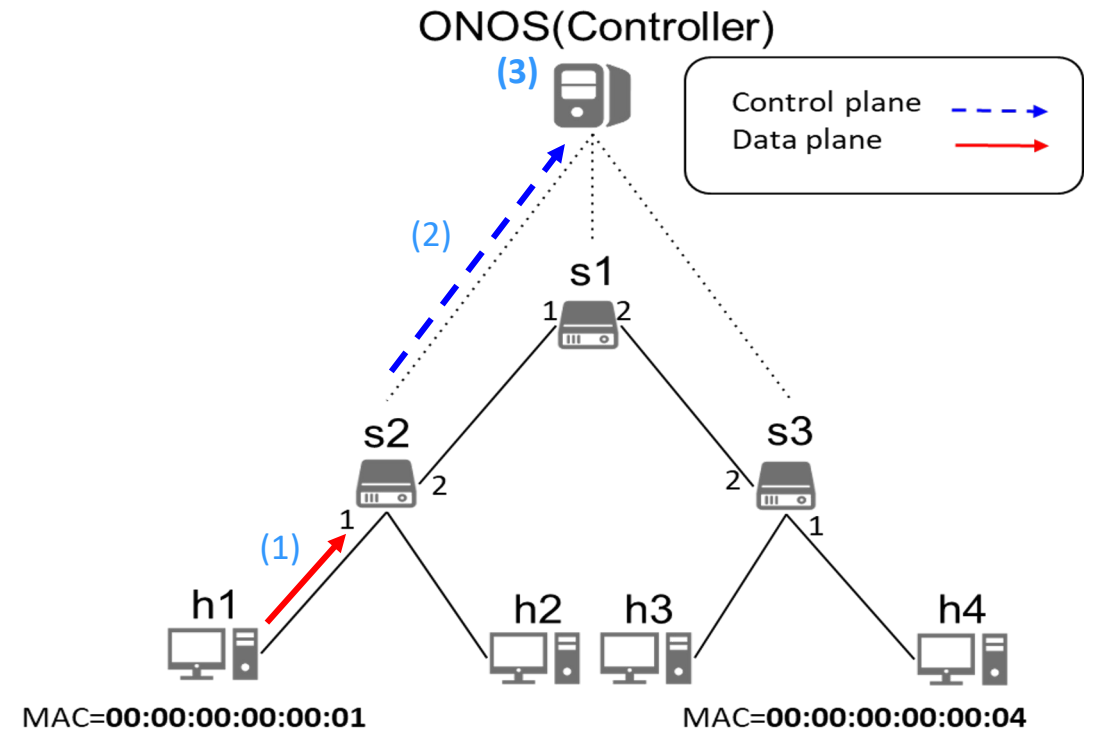




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch sends Packet-in to Controller.
3. **Controller updates MAC address table with source MAC and incoming port.**
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

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

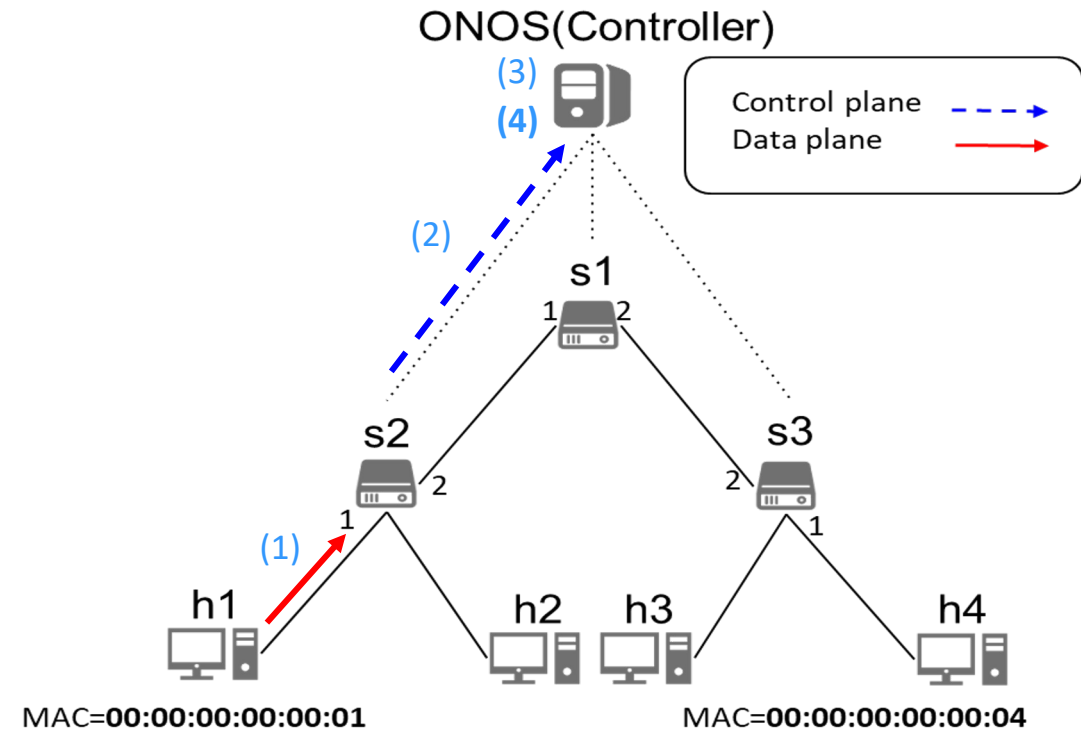




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. **Controller looks up MAC address table for destination MAC:**
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

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

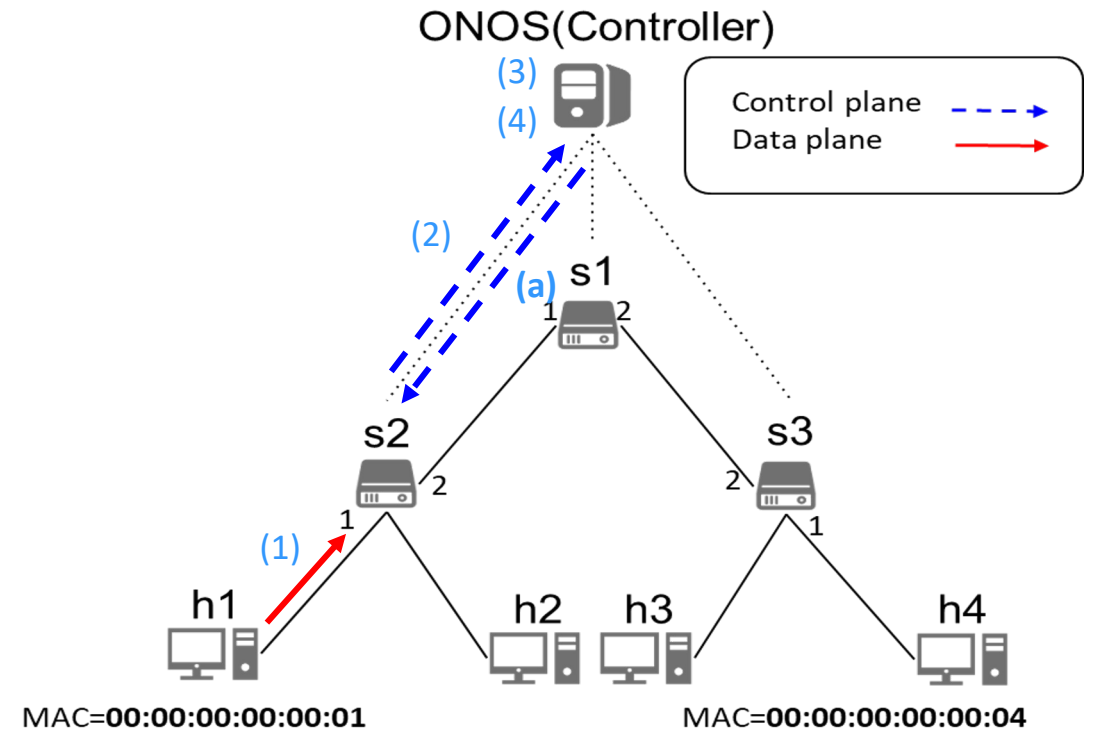




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

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

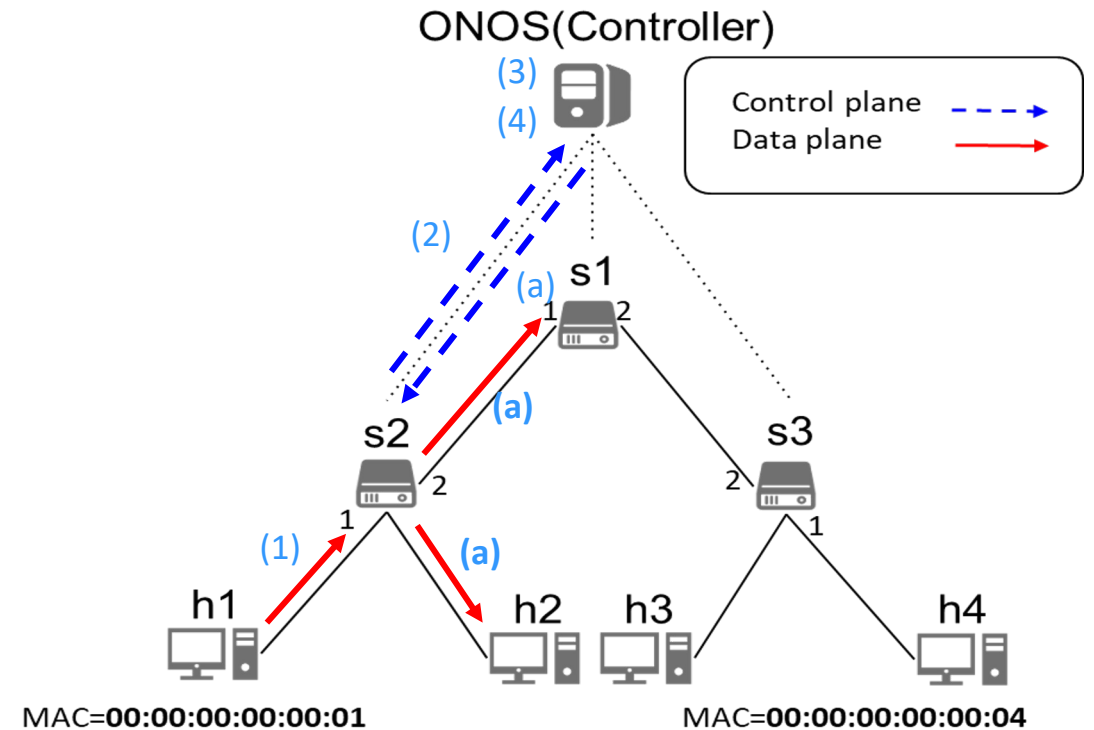




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

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

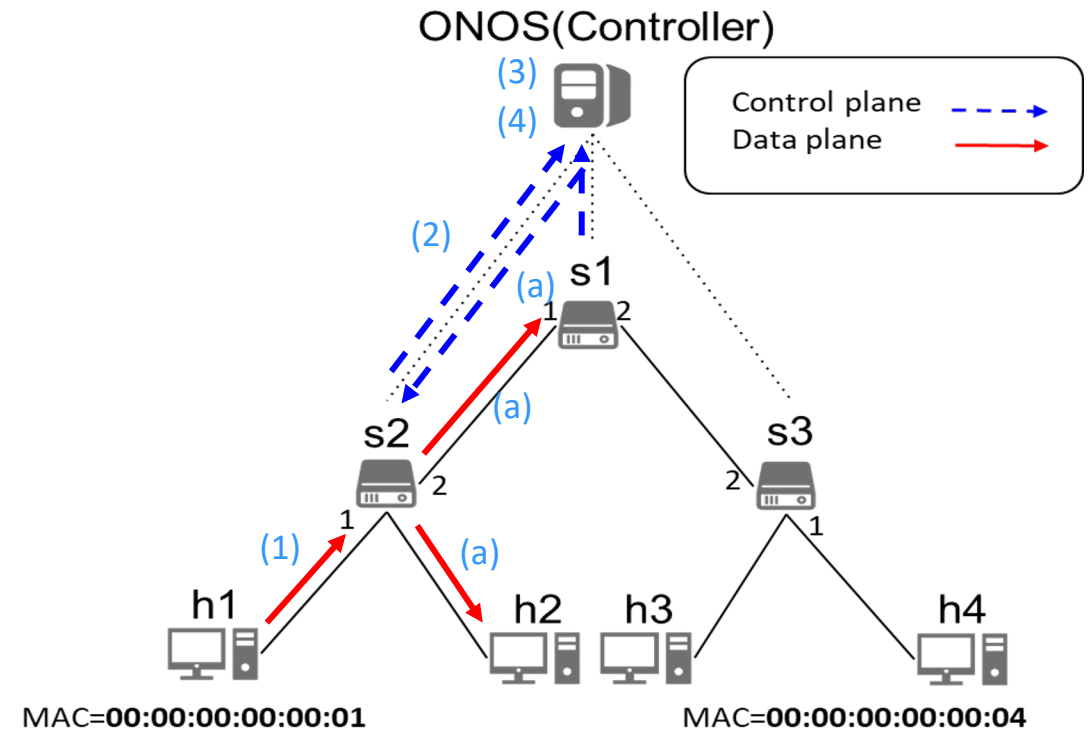




Workflow (h1 -> h4)

1. h1 pings h4.
2. **Switch (s1) sends Packet-in to Controller.**
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

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

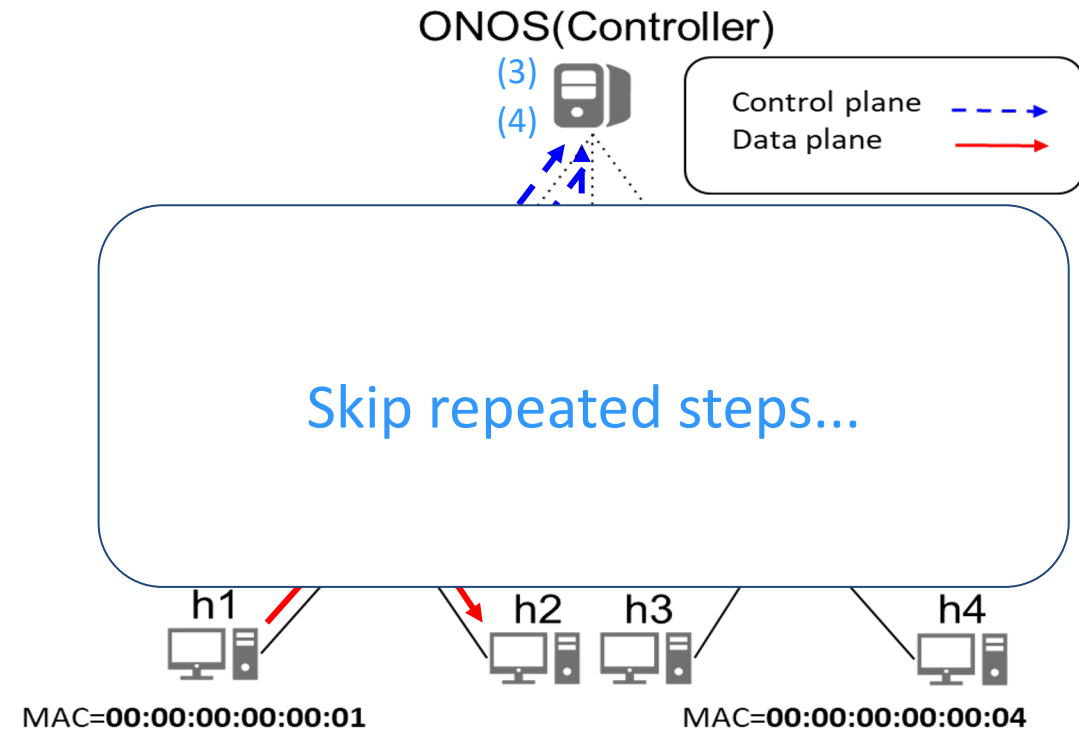




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch sends Packet-in to Controller.
3. **Controller updates MAC address table with source MAC and incoming port.**
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

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

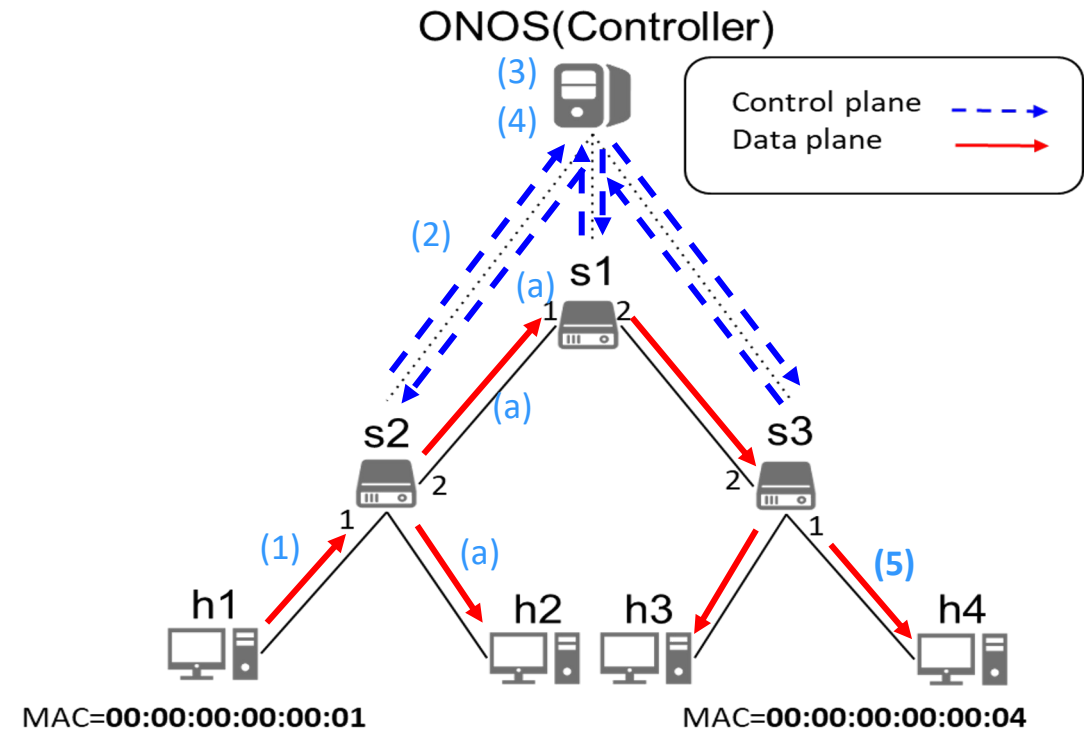




Workflow (h1 -> h4)

1. h1 pings h4.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h4 receives packet from h1.

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

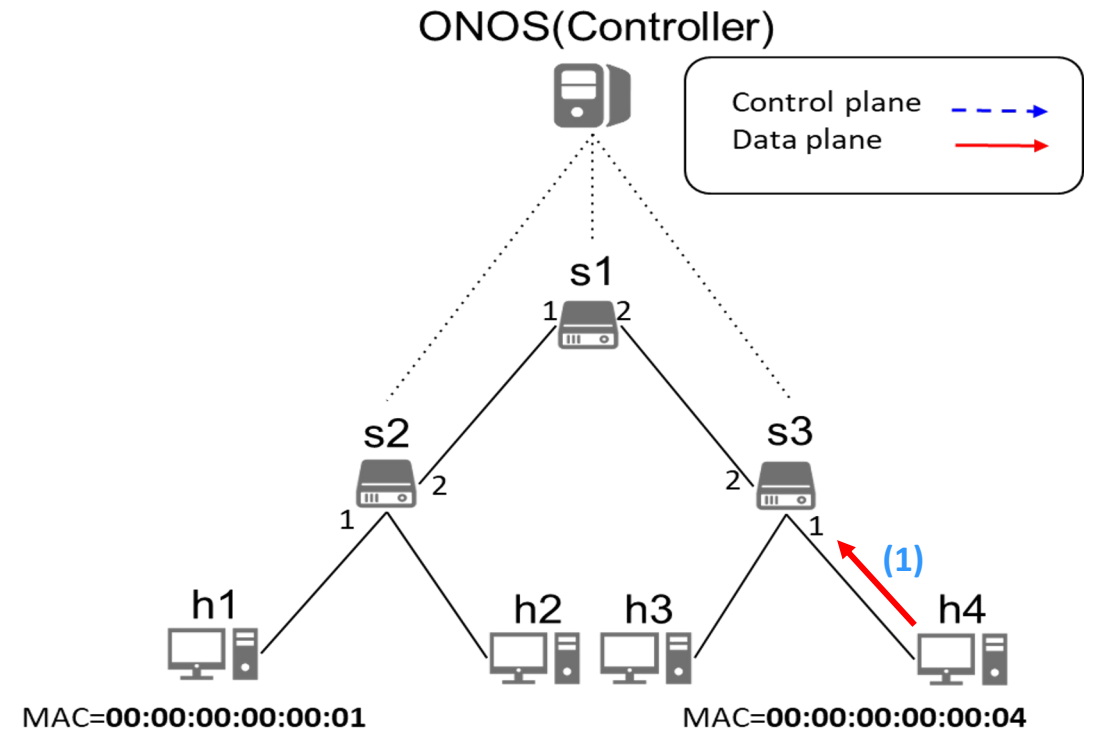




Workflow (h4 -> h1)

1. h4 replies to h1.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from h4.

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

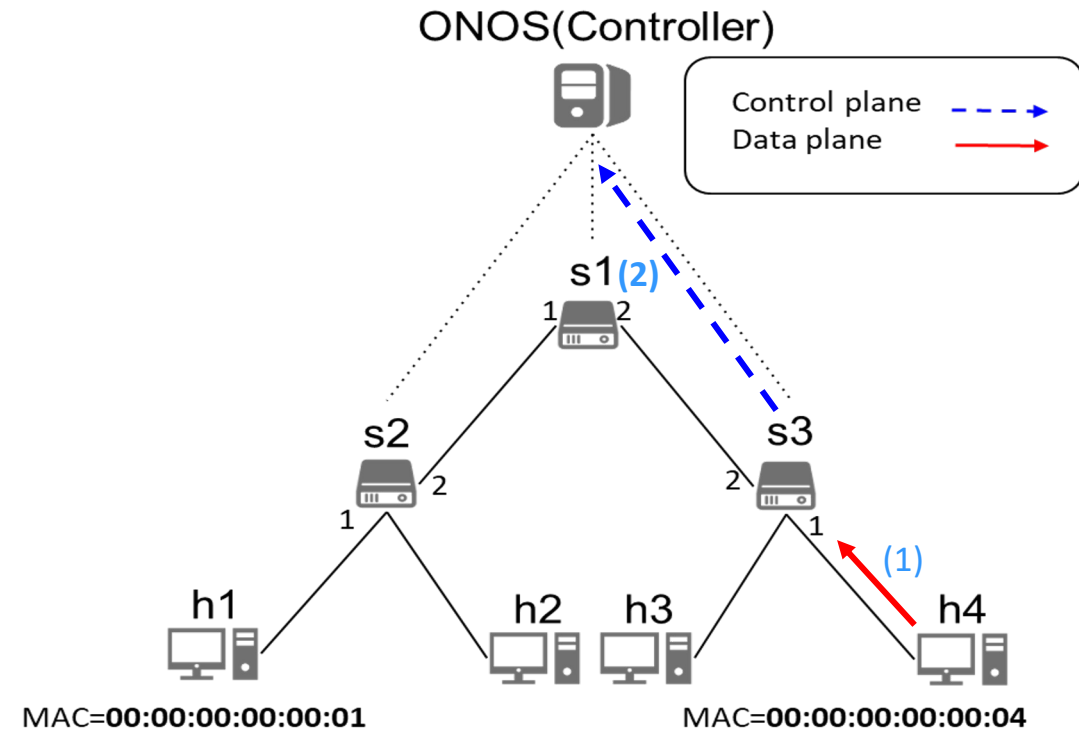




Workflow (h4 -> h1)

1. h4 replies to h1.
2. **Switch (s3) sends Packet-in to Controller.**
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from h4.

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





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

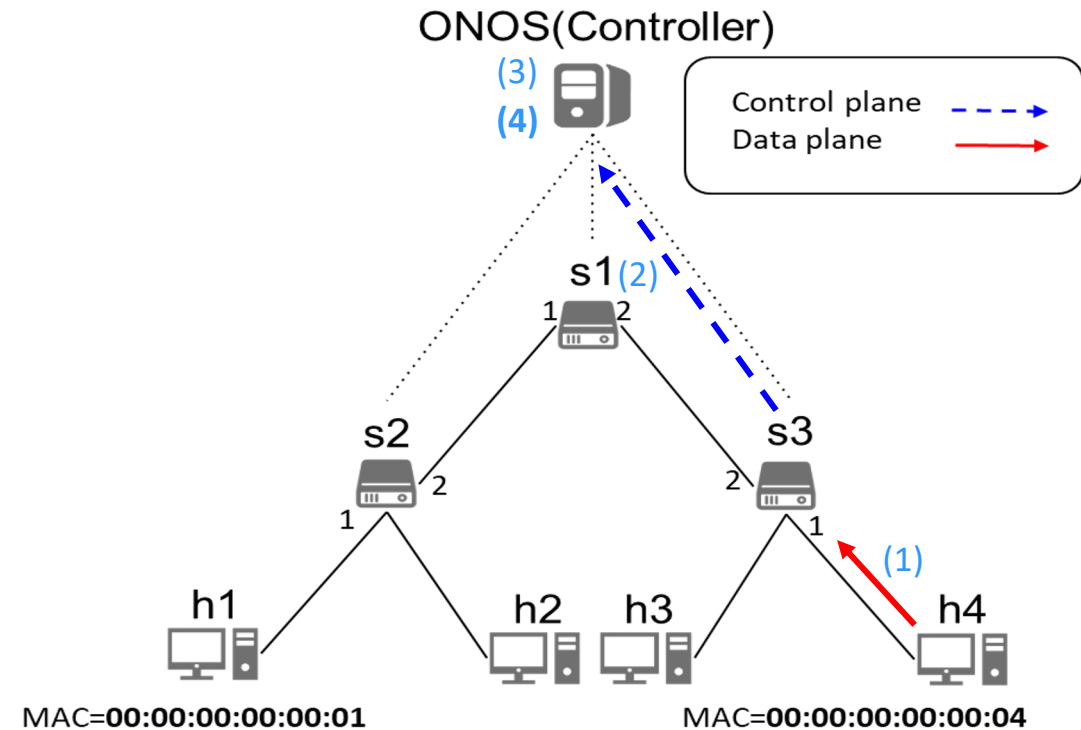




Workflow (h4 -> h1)

1. h4 replies to h1.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. **Controller looks up MAC address table for destination MAC:**
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from h4.

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

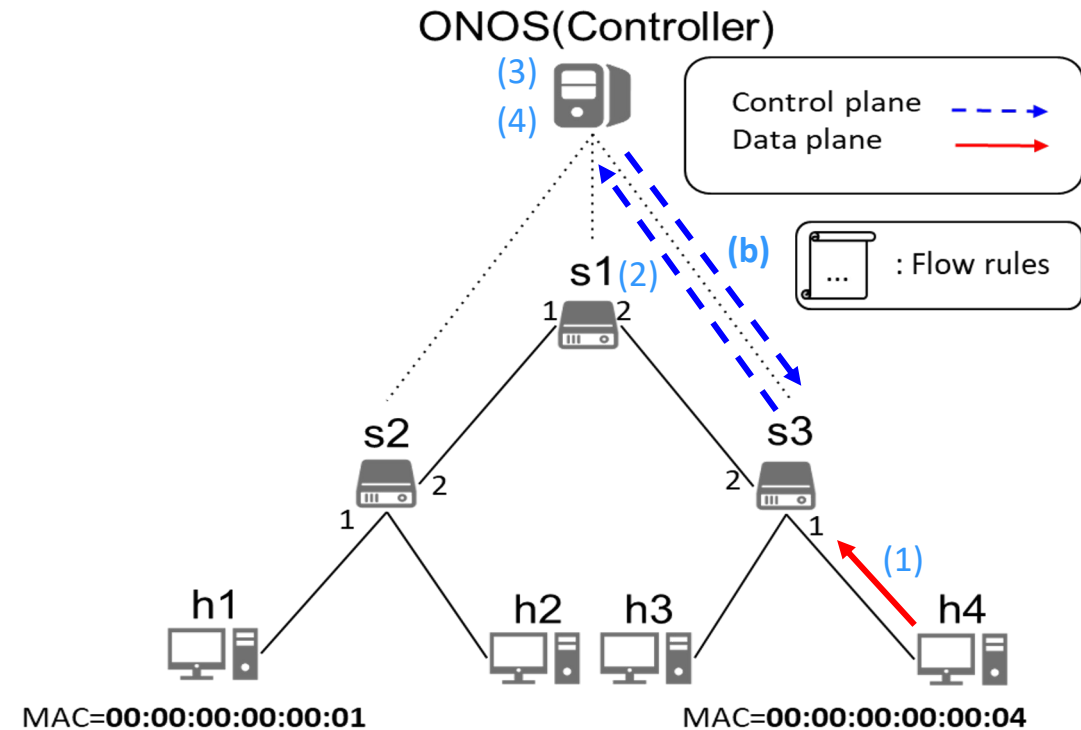




Workflow (h4 -> h1)

1. h4 replies to h1.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from h4.

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

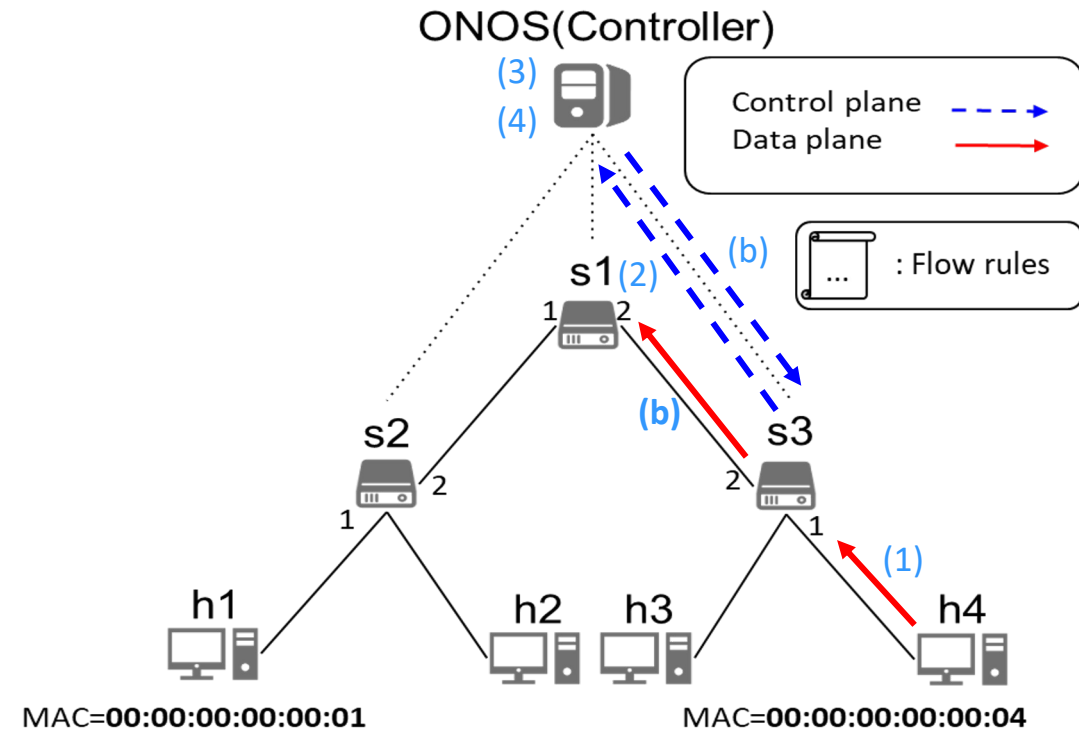




Workflow (h4 -> h1)

1. h4 replies to h1.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from h4.

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

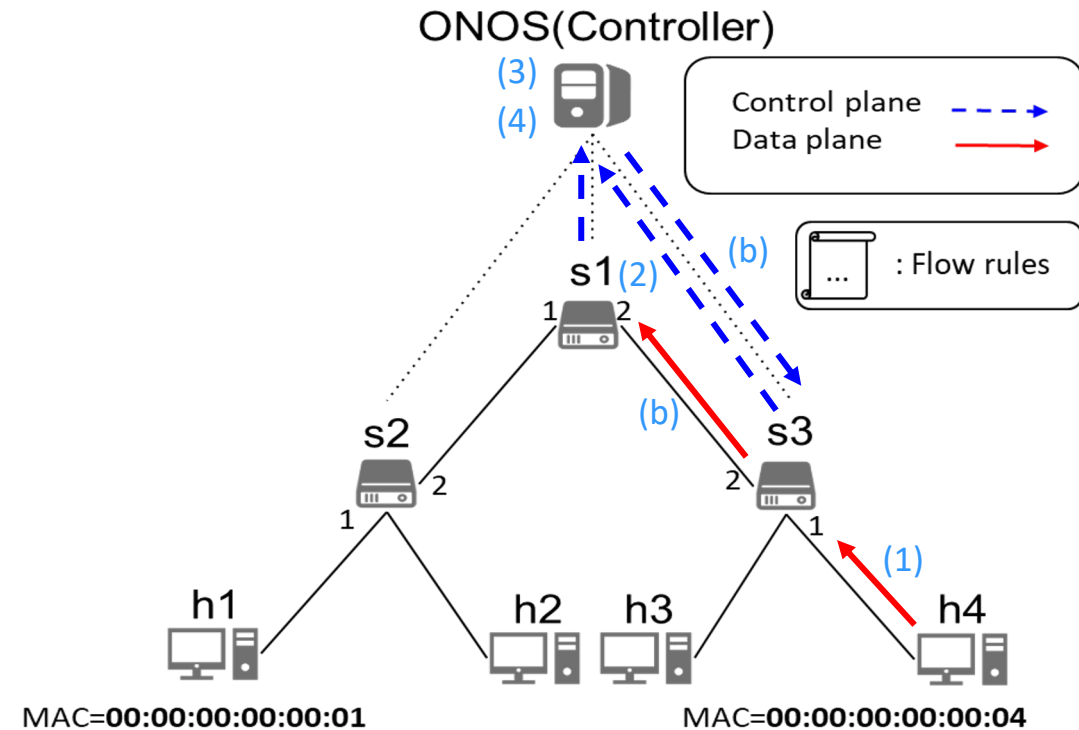




Workflow (h4 -> h1)

1. h4 replies to h1.
2. **Switch (s1) sends Packet-in to Controller.**
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from h4.

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

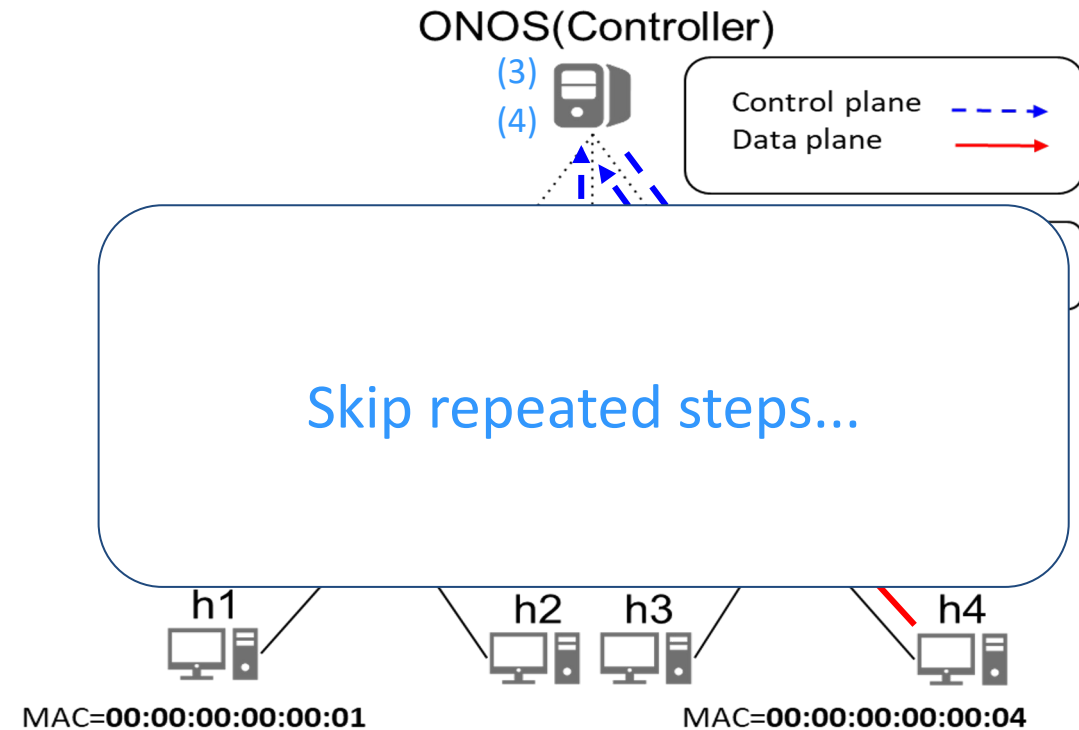




Workflow (h4 -> h1)

1. h4 replies to h1.
2. Switch sends Packet-in to Controller.
3. **Controller updates MAC address table with source MAC and incoming port.**
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from h4.

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

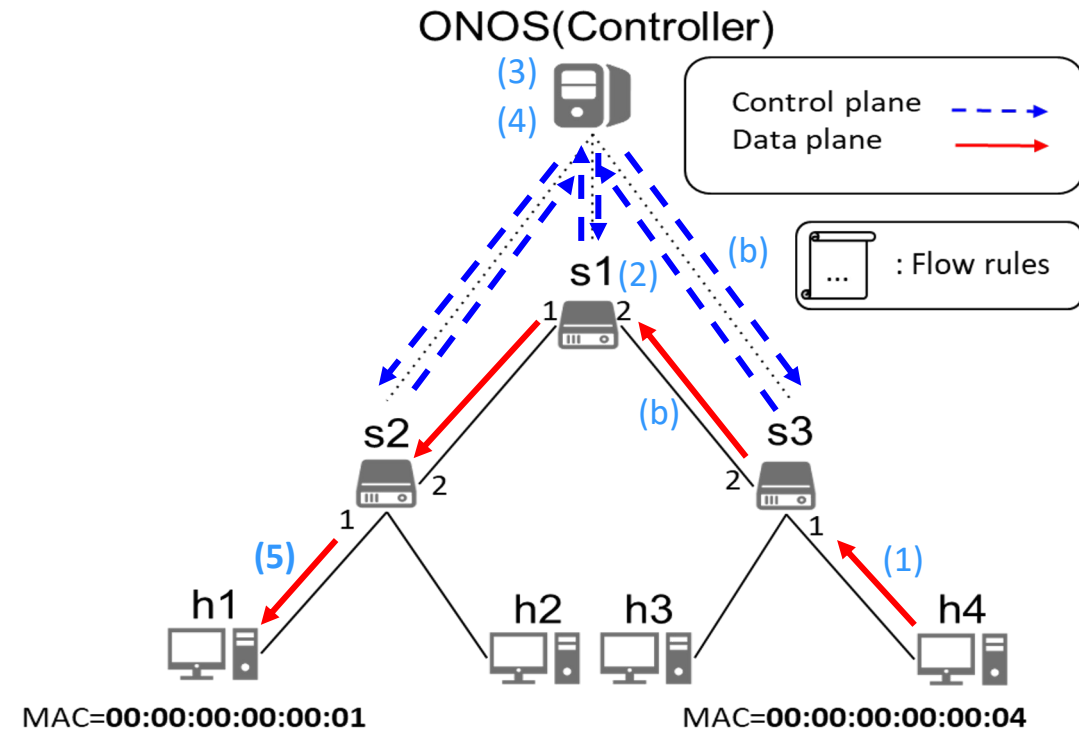




Workflow (h4 -> h1)

1. h4 replies to h1.
2. Switch sends Packet-in to Controller.
3. Controller updates MAC address table with source MAC and incoming port.
4. Controller looks up MAC address table for destination MAC:
 - a. Destination MAC not found:
 - Floods Packet-out.
 - b. Destination MAC found:
 - Sends Packet-out via designated port.
 - Installs flow rule on switch.
5. h1 receives packet from 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





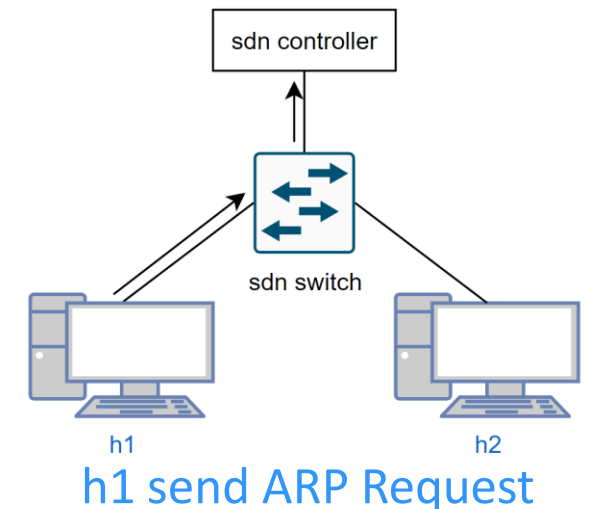
Outline

- Build ONOS Application Project
 - Environment Setup
 - Create an ONOS Application
 - Build, Install, Activate, and Reinstall ONOS Application
- ARP
 - Introduction
 - ARP Request/Reply Format
- Learning Bridge Function
 - Introduction
 - Workflow
- Proxy APR
 - Introduction
 - Workflow
- Lab 3 Requirements



What is Proxy ARP

- A Proxy device answers ARP Requests for IP address on behalf of other devices
 - The Proxy device could be router, firewall, etc.
 - The replied MAC belongs to the **Proxy device**
- In SDNs, controller can serve as Proxy device
 - However, the replied MAC belongs to the **target host**
 - Benefits:
 - Decreases workload of network devices
 - Prevent issues like broadcast storm





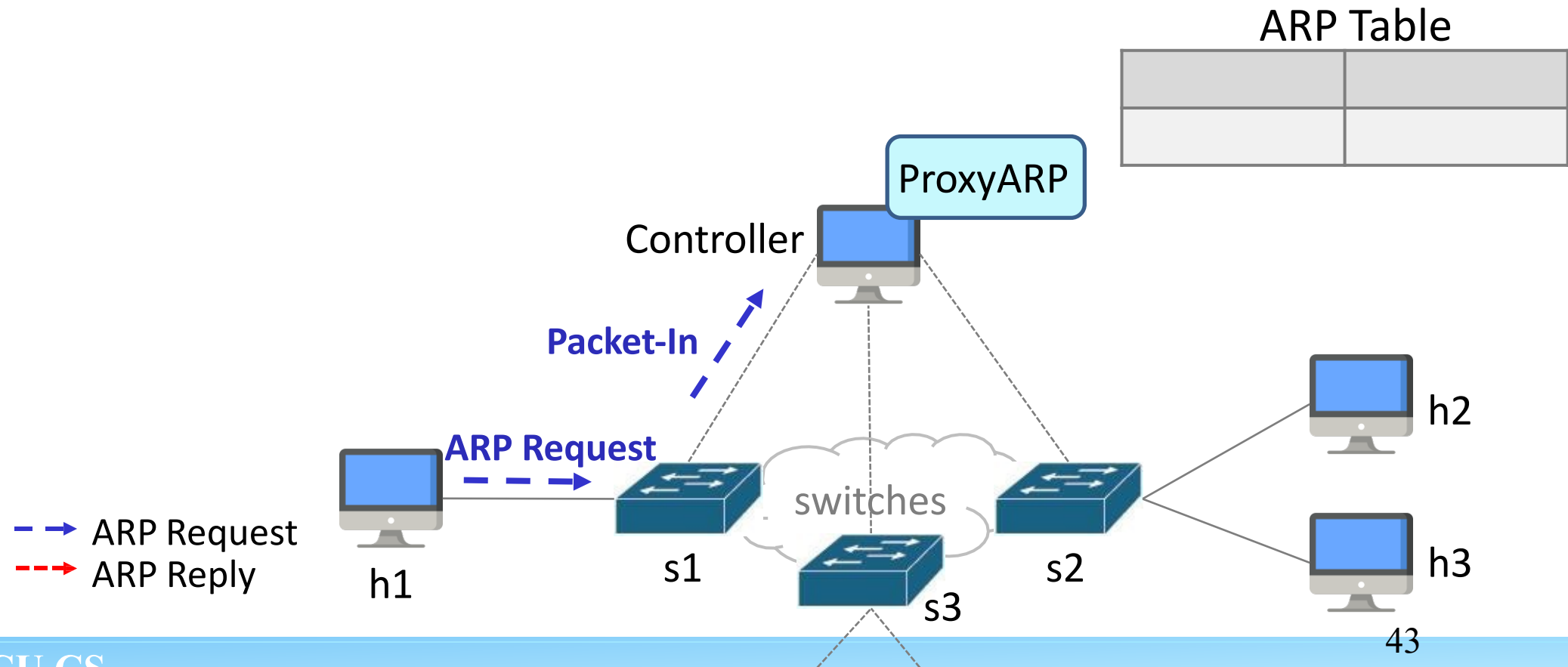
Workflow of Proxy ARP in SDN

1. Sender sends ARP Request
2. Edge switch Packet-Ins the Request to controller
3. Proxy ARP learns IP-MAC mappings of the sender
4. Proxy ARP looks up ARP table (For target IP-MAC mapping)
 - If mapping exist:
 - Fetch target MAC
 - 5a. Packet-Outs ARP Reply (with target MAC) to the sender
 - Else (mapping not exist):
 - 5b. **Floods** ARP Request to **edge ports** except the port receiving ARP Request
6. When h2 receives ARP Request, h2 will Reply ARP packet.
7. Edge switch Packet-Ins the Reply to controller
8. Proxy ARP learns IP-MAC mapping from h2



First ARP Request

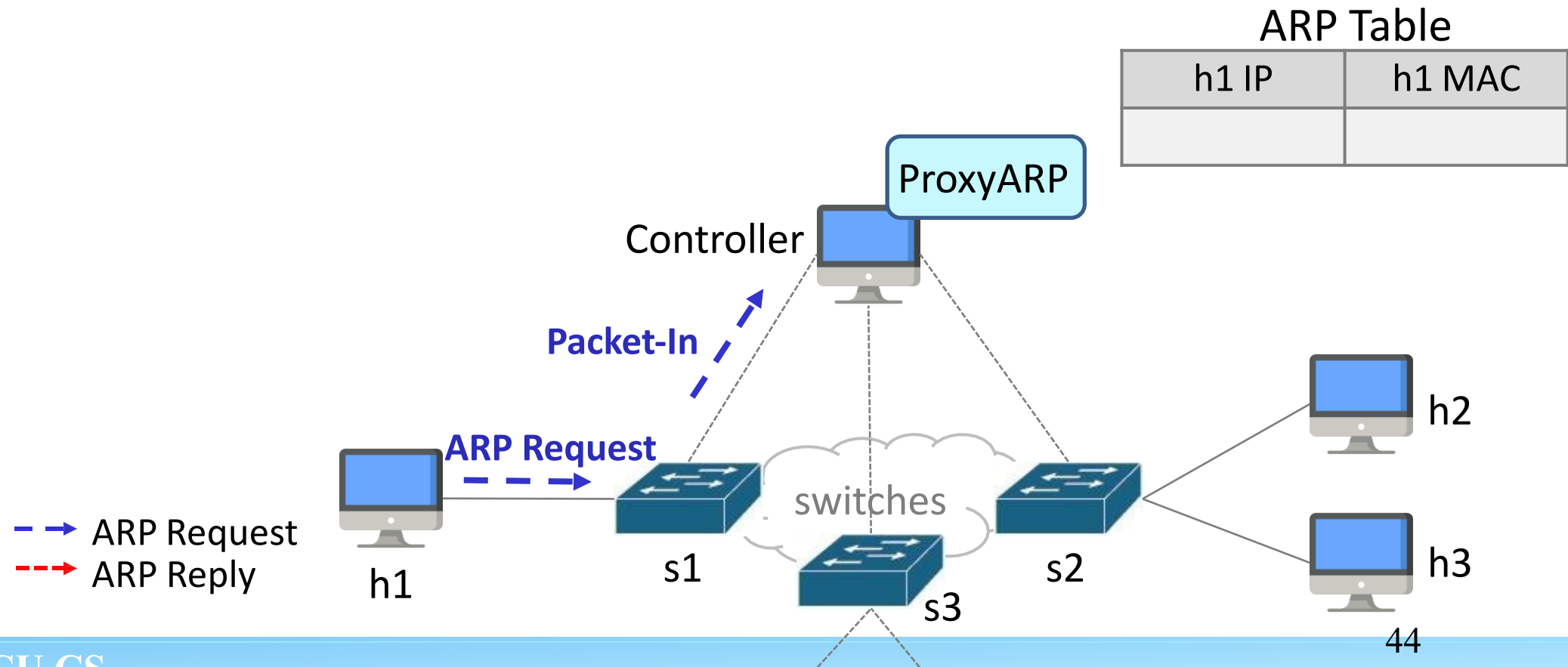
1. h1 sends ARP Request
2. Edge switch Packet-Ins the Request to controller





Proxy ARP learns IP-MAC

3. Controller learns mapping of IP to MAC of h1
4. Proxy ARP looks up ARP table (For target IP-MAC mapping)

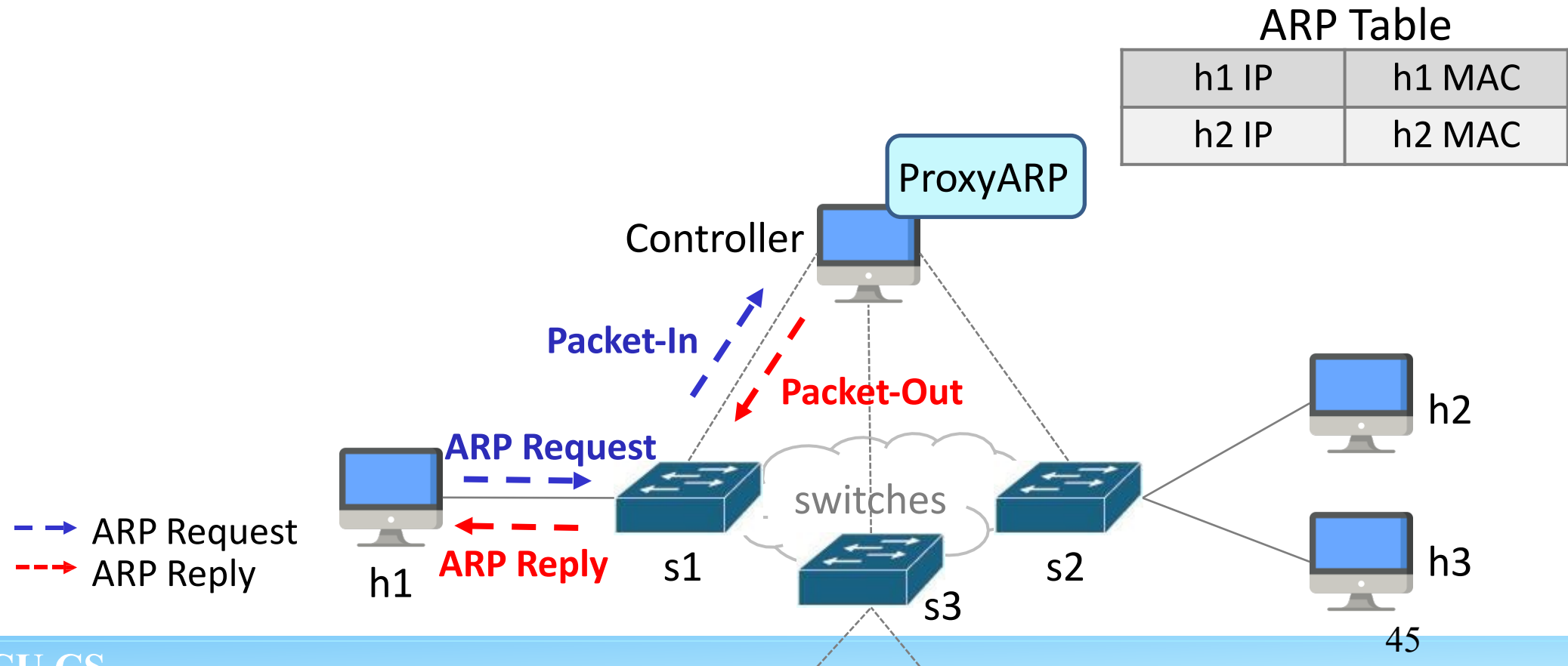




If mapping exist

- Fetch target MAC

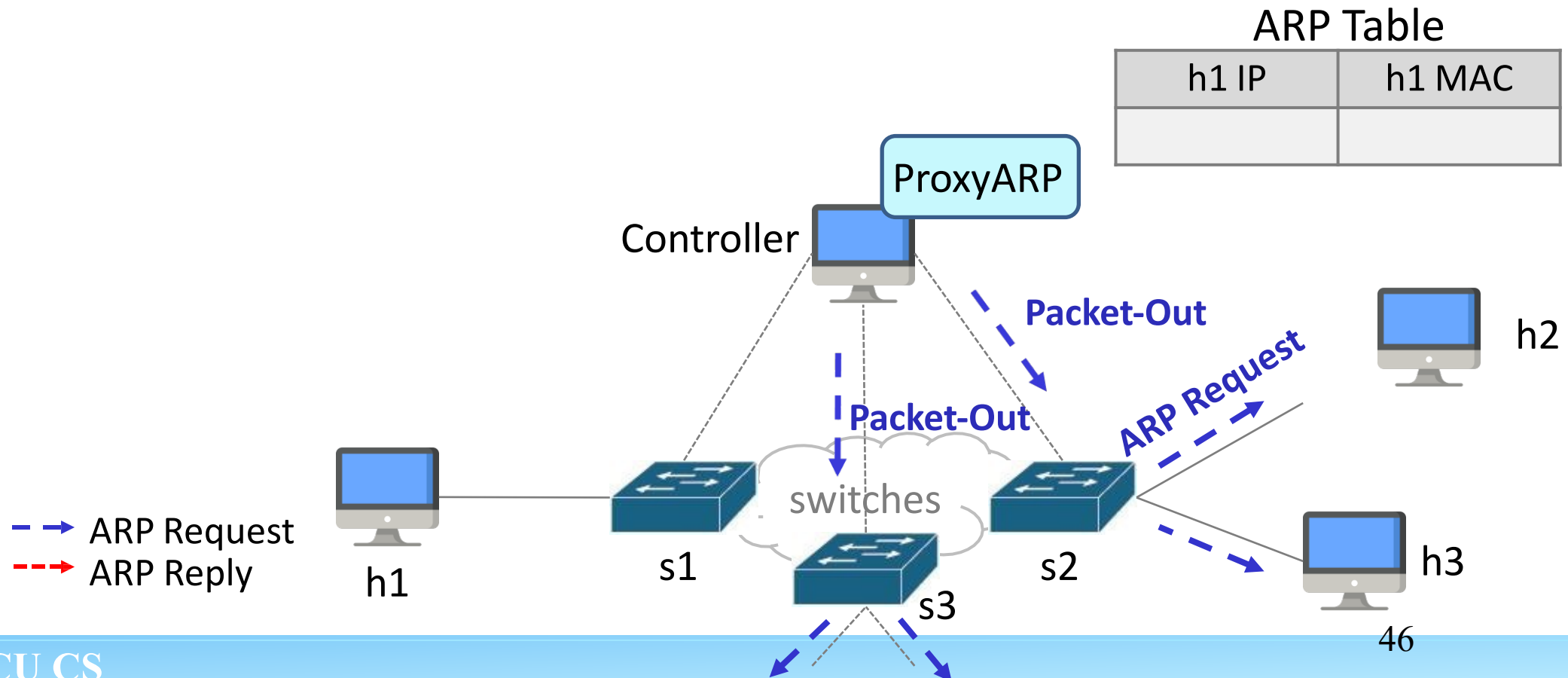
5a. Proxy ARP simply generates and Packet-Outs ARP Reply (with target MAC) to the sender





If mapping not exist

- 5b. Floods ARP Request to edge ports except the port receiving ARP Request via Packet-Outs ARP Request

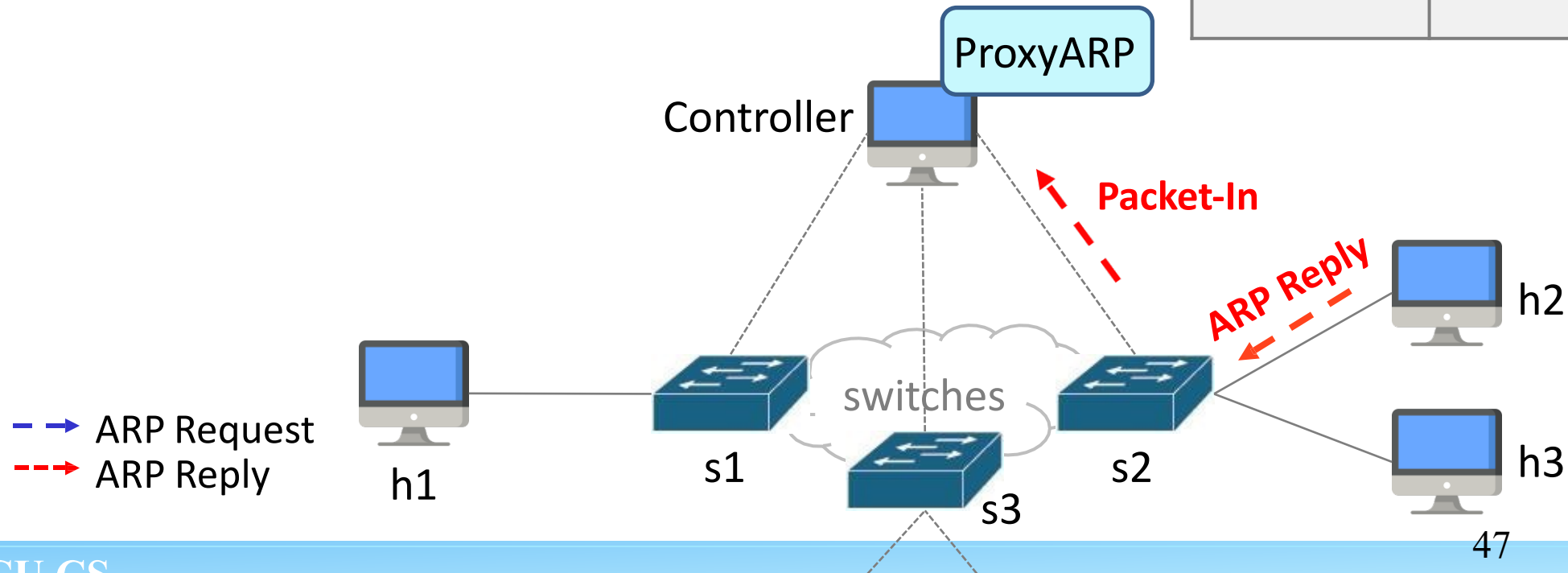




Reply ARP packet

6. When h2 receives ARP Request, h2 will Reply ARP packet.
7. Edge switch Packet-Ins the Reply to controller

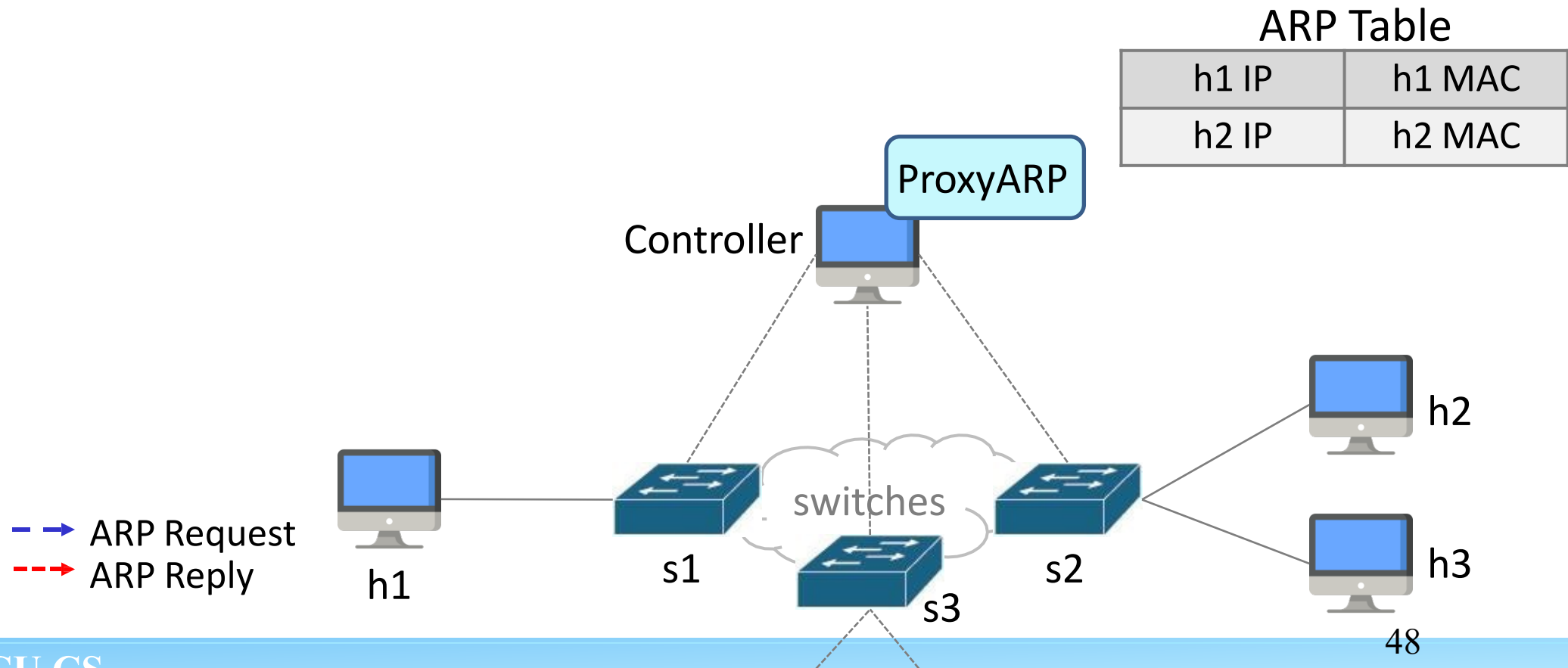
ARP Table	
h1 IP	h1 MAC





Proxy ARP learns IP-MAC

8. Proxy ARP learns IP-MAC from h2





Outline

- Build ONOS Application Project
 - Environment Setup
 - Create an ONOS Application
 - Build, Install, Activate, and Reinstall ONOS Application
- ARP
 - Introduction
 - ARP Request/Reply Format
- Learning Bridge Function
 - Introduction
 - Workflow
- Proxy APR
 - Introduction
 - Workflow
- Lab 3 Requirements



Lab 3 Descriptions

- **Complete the given code for learning bridge function (40%)**
 - Correct naming convention in pom (5%)
 - Learning bridge function is available (20%)
 - Logs are in the correct format (10%)
 - Flow rules in hosts are comply with the regulations (5%)
- **Create an ONOS application for Proxy ARP (60%)**
 - Correct naming convention in pom (10%)
 - Proxy ARP is available (40%)
 - Logs are in the correct format (10%)
- **The two apps will be tested separately**
- **If there's a difference between the image and the text, just go with the text.**
- **We'll be providing a template for the learning bridge app at E3**



Learning Bridge Function

- You must set values in the **pom.xml** file as the following: (5%)
 - **<groupId>**: **nycu.winlab**
 - **<artifactId>**: **bridge-app**
 - **<version>**: **(default)**
 - **<onos.app.name>**: **nycu.winlab.bridge**
- You earn credits only if **all** settings are correct.

```
26     <groupId>nctu.winlab</groupId>
27     <artifactId>bridge-app</artifactId>
28     <version>1.0-SNAPSHOT</version>
29     <packaging>bundle</packaging>
30
31     <description>ONOS OSGi bundle archetype</description>
32     <url>http://onosproject.org</url>
33
34     <properties>
35         <onos.app.name>nctu.winlab.bridge</onos.app.name>
36         <onos.app.title>Learning Bridge App</onos.app.title>
37         <onos.app.origin>Winlab, NCTU</onos.app.origin>
38         <onos.app.category>default</onos.app.category>
39         <onos.app.url>http://onosproject.org</onos.app.url>
40         <onos.app.readme>ONOS OSGi bundle archetype.</onos.app.readme>
41     </properties>
```



Learning Bridge Function

- Ping should work for all host pairs.

```
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> 
```

- Learning Bridge Function with tree (depth=2) topology. **(10%)**

```
$ sudo mn --controller=remote,127.0.0.1:6653 \
  --topo=tree,depth=2 \
  --switch=ovs,protocols=OpenFlow14
```

- Learning Bridge Function with tree (depth=3~5) topology. **(10%)**
 - You earn credits only if your application works for **all** depths.



Learning Bridge Function

- We will test your application with only the following applications activated:

```
demo@root > apps -a -s 23:1
* 12 org.onosproject.optical-model 2.7.0 Optical Network Model
* 13 org.onosproject.drivers 2.7.0 Default Drivers
* 52 org.onosproject.openflow-base 2.7.0 OpenFlow Base Provider
* 72 org.onosproject.hostprovider 2.7.0 Host Location Provider
* 73 org.onosproject.lldpprovider 2.7.0 LLDP Link Provider
* 74 org.onosproject.openflow 2.7.0 OpenFlow Provider Suite
* 81 org.onosproject.gui2 2.7.0 ONOS GUI2
```

- You must only use classes under [org.onosproject.net.flowobjective](#) or [org.onosproject.net.flow](#) package to install flow rules on network devices.
 - If you don't follow this, you won't get any points.
 - **Everyone should really understand how to use both packages to install flow rules.**



Learning Bridge Function

- Use *log.info()* to record actions done by your application. (10%)
 1. New entry is added into the forwarding table.
 2. Destination MAC address is missed. Flood the packet.
 3. Destination MAC address is matched. Install a flow rule.
- You earn credits only if each log pattern is **exactly the same as the given one**.

```
1. 2022-09-29T01:58:41,115 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | Add an entry to the port table of `of:0000000000000002`. MAC address: `2E:D1:D4:8A:B1:90` => Port: `1`.
    2022-09-29T01:58:41,116 | INFO | onos-of-dispatcher-127.0.0.1:53624 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | Add an entry to the port table of `of:0000000000000001`. MAC address: `2E:D1:D4:8A:B1:90` => Port: `1`.
2. 2022-09-29T01:58:41,116 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | MAC address `FF:FF:FF:FF:FF:FF` is missed on `of:0000000000000002`. Flood the packet.
    2022-09-29T01:58:41,116 | INFO | onos-of-dispatcher-127.0.0.1:53624 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | MAC address `FF:FF:FF:FF:FF:FF` is missed on `of:0000000000000001`. Flood the packet.
    2022-09-29T01:58:41,117 | INFO | onos-of-dispatcher-127.0.0.1:53632 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | Add an entry to the port table of `of:0000000000000003`. MAC address: `2E:D1:D4:8A:B1:90` => Port: `3`.
    2022-09-29T01:58:41,117 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | Add an entry to the port table of `of:0000000000000002`. MAC address: `A2:66:19:A6:1D:0F` => Port: `2`.
    2022-09-29T01:58:41,117 | INFO | onos-of-dispatcher-127.0.0.1:53632 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | MAC address `FF:FF:FF:FF:FF:FF` is missed on `of:0000000000000003`. Flood the packet.
3. 2022-09-29T01:58:41,121 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | MAC address `2E:D1:D4:8A:B1:90` is matched on `of:0000000000000002`. Install a flow rule.
    2022-09-29T01:58:41,122 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | Add an entry to the port table of `of:0000000000000002`. MAC address: `2E:D1:D4:8A:B1:90` => Port: `1`.
    2022-09-29T01:58:41,123 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | MAC address `A2:66:19:A6:1D:0F` is matched on `of:0000000000000002`. Install a flow rule.
    2022-09-29T01:58:41,128 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | Add an entry to the port table of `of:0000000000000002`. MAC address: `2E:D1:D4:8A:B1:90` => Port: `1`.
    2022-09-29T01:58:41,129 | INFO | onos-of-dispatcher-127.0.0.1:53644 | LearningBridge | 215 - nctu.winlab.bridge-app - 1.0.0.SNAPSHOT
    | MAC address `FF:FF:FF:FF:FF:FF` is missed on `of:0000000000000002`. Flood the packet.
```



Learning Bridge Function

1. New entry is added into the MAC address table.
 - Pattern: **“Add an entry to the port table of `{device ID}`. MAC address: `{MAC}` => Port: `{port}`.”**
 - Example: “Add an entry to the port table of `of:000000000000000002`. MAC address: `2E:D1:D4:8A:B1:90` => Port: `1`.”
2. Destination MAC address is missed. Flood the packet.
 - Pattern: **“MAC address `{MAC}` is missed on `{device ID}`. Flood the packet.”**
 - Example: “MAC address `FF:FF:FF:FF:FF:FF` is missed on `of:000000000000000002`. Flood the packet.”
3. Destination MAC address is matched. Install a flow rule.
 - Pattern: **“MAC address `{MAC}` is matched on `{device ID}`. Install a flow rule.”**
 - Example: “MAC address `2E:D1:D4:8A:B1:90` is matched on `of:000000000000000002`. Install a flow rule.”



Learning Bridge Function

- Here are examples of how to use *log.info()* to print log information.

```
log.info("Add an entry to the port table of `" + inDevice +  
|      "` . MAC address: `" + srcMac + "` => Port: `" + inPort + "`.");
```

```
log.info("MAC address `" + dstMac + "` is missed on `" + inDevice + "` . Flood the packet.");
```

```
log.info("MAC address `" + dstMac + "` is matched on `" + inDevice + "` . Install a flow rule.");
```

```
log.info("MAC address `{}` is matched on `{}` . Install a flow rule.",  
|      |      dstMac, recDevId);
```




Learning Bridge Function

- Rule requirements: (5%)
 - Match field (selector): **ETH_SRC, ETH_DST**
 - Action field (treatment): **OUTPUT**
 - Flow priority: **30**
 - Flow timeout: **30**
 - You earn credits only if **all** flow rules are correct.

STATE ▼	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	0	2,945	1	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	1	7	30	0	ETH_DST:A2:66:19:A6:1D:0F, ETH_SRC:3E:0B:9F:F9:EF:D9	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	1	8	30	0	ETH_DST:A2:66:19:A6:1D:0F, ETH_SRC:9A:E8:EA:DF:AD:88	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	1	8	30	0	ETH_DST:2E:D1:D4:8A:B1:90, ETH_SRC:3E:0B:9F:F9:EF:D9	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	1	7	30	0	ETH_DST:3E:0B:9F:F9:EF:D9, ETH_SRC:A2:66:19:A6:1D:0F	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge
Added	1	8	30	0	ETH_DST:3E:0B:9F:F9:EF:D9, ETH_SRC:2E:D1:D4:8A:B1:90	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge
Added	1	8	30	0	ETH_DST:9A:E8:EA:DF:AD:88, ETH_SRC:A2:66:19:A6:1D:0F	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge
Added	1	8	30	0	ETH_DST:2E:D1:D4:8A:B1:90, ETH_SRC:9A:E8:EA:DF:AD:88	imm[OUTPUT:1], cleared:false	nctu.winlab.bridge
Added	1	8	30	0	ETH_DST:9A:E8:EA:DF:AD:88, ETH_SRC:2E:D1:D4:8A:B1:90	imm[OUTPUT:2], cleared:false	nctu.winlab.bridge



Learning Bridge Function

- You can trace [ReactiveForwarding.java](#) to figure out how to install flow rules.
- When receives Packet-in, your application need to send Packet-out to switch, in addition to installing flow rule.
- How to debug:
 - Use [Logger](#) to print runtime information.
 - Use [Wireshark](#) to capture your packets.



ARP Proxy

- You must set values in the `pom.xml` file as the following: (10%)
 - `<groupId>`: `nycu.winlab`
 - `<artifactId>`: `ProxyArp`
 - `<version>`: `(default)`
 - `<onos.app.name>`: `nycu.winlab.ProxyArp`
- You earn credits only if **all** settings are correct.



ARP Proxy

- Work properly at least in **tree (depth=3, fanout=3)** topology **(40%)**
 - - All hosts are able to **arping** to each other
- Once you activate your application and Mininet, execute **arping** in Mininet to check ARP functionality

```
mininet> h1 arping h2
```

- Correct result would look like:

```
mininet> h1 arping h2 -c 3
ARPING 10.0.0.2 from 10.0.0.1 h1-eth0
Unicast reply from 10.0.0.2 [D6:B5:82:B5:23:0E] 15.850ms
Unicast reply from 10.0.0.2 [D6:B5:82:B5:23:0E] 4.267ms
Unicast reply from 10.0.0.2 [D6:B5:82:B5:23:0E] 4.370ms
Sent 3 probes (1 broadcast(s))
Received 3 response(s)
mininet>
```



ARP Proxy

- Print messages in following events: (10%)

- ARP table miss

```
| 209 - nctu.winlab.ProxyArp - 1.0.0.SNAPSHOT | TABLE MISS. Send request to edge ports
```

- ONOS receives ARP Reply from host

```
| 209 - nctu.winlab.ProxyArp - 1.0.0.SNAPSHOT | RECV REPLY. Requested MAC = 06:4F:F1:84:A5:EA
```

- ARP table hit

```
| 209 - nctu.winlab.ProxyArp - 1.0.0.SNAPSHOT | TABLE HIT. Requested MAC = 06:4F:F1:84:A5:EA
```

- You earn credits only if each log pattern is **exactly the same as the given one.**
- We also use *log.info()* to print messages.



ARP Proxy

- ONOS application activation
 - You are only allowed to activate **your *ProxyARP*** and the following ONOS applications:

```
brian@root > apps -a -s
* 6 org.onosproject.drivers 2.2.0 Default Drivers
* 7 org.onosproject.optical-model 2.2.0 Optical Network Model
* 39 org.onosproject.gui2 2.2.0 ONOS GUI2
* 52 org.onosproject.openflow-base 2.2.0 OpenFlow Base Provider
* 84 org.onosproject.hostprovider 2.2.0 Host Location Provider
* 85 org.onosproject.lldpprovider 2.2.0 LLDP Link Provider
* 86 org.onosproject.openflow 2.2.0 OpenFlow Provider Suite
* 192 nctu.winlab.ProxyArp 1.0.SNAPSHOT ONOS OSGi bundle archetype
```



Submission Naming Convention

- Move your bridge-app and Proxy ARP into directory **lab3_<student ID>**
- Compress the directory into **zip** file named as **lab3_<student ID>**
- Wrong file name or format will result in **10 points deduction**
- **20% deduction for late submission in one week**
 - Won't accept submissions over **one week**

```
sdn@sdn-virtual-machine:~/lab3_313552034$ tree
.
├── bridge-app
│   ├── pom.xml
│   └── src
│       ├── main
│       │   ├── java
│       │   │   ├── nycu
│       │   │   └── winlab
│       │   └── bridge
│       │       ├── AppComponent.java
│       │       ├── package-info.java
│       │       └── SomeInterface.java
│       └── test
│           ├── java
│           │   ├── nycu
│           │   └── winlab
│           └── bridge
│               └── AppComponentTest.java
└── ProxyArp
    ├── pom.xml
    └── src
        ├── main
        │   ├── java
        │   │   ├── nycu
        │   │   └── winlab
        │   └── ProxyArp
        │       ├── AppComponent.java
        │       ├── package-info.java
        │       └── SomeInterface.java
        └── test
            ├── java
            │   ├── nycu
            │   └── winlab
            └── ProxyArp
                └── AppComponentTest.java

24 directories, 10 files
```



Lab 3 Demo

- TA has opened a demo time-reserved table.
 - Lab 3 Demo Time-reserved Table
 - Open Period: 9/26 ~ 10/16 23:59
 - The demo dates will be in the week after Lab 3 deadline.
- Demo questions will show when demo starts.
- The demo score will be **40%** of total score.
 - e.g. If you earn 100% credits for submission and 80% credits for demo, then your total score of Lab3 will be **$100 \times 60\% + 80 \times 40\% = 92$** .



About help!

- For lab problem, ask at e3 forum
 - Ask at the e3 forum
 - TAs will help to clarify Lab contents instead of giving answers!
 - Please describe your questions with sufficient context,
 - e.g. Environment setup, Input/Output, Screenshots, ...
- For personal problem mail to sdnta@win.cs.nycu.edu.tw
 - You have special problem and you can't meet the deadline
 - You got weird score with lab
- No Fixed TA hours