Project 4

Unicast DHCP Application

Deadline: 2022/11/09 (WED) 23:59

Outline

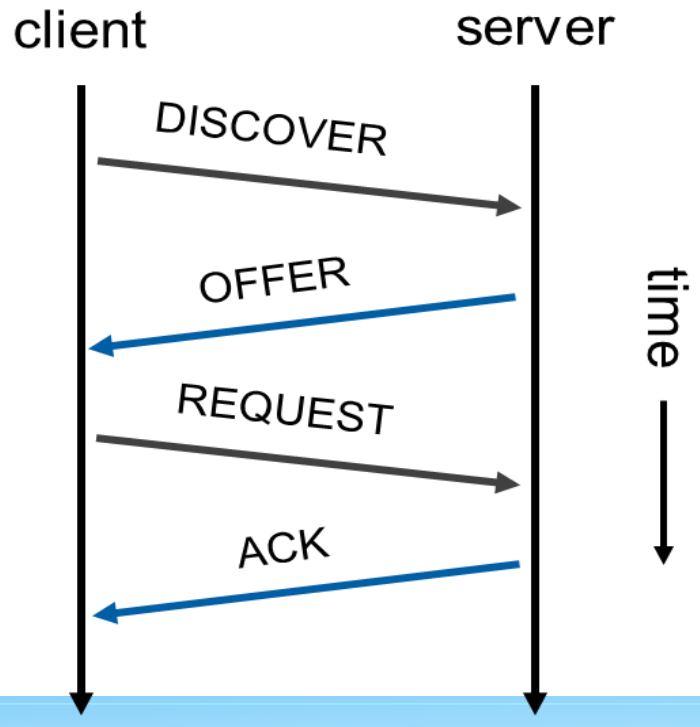
- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
- Submission & Scoring Criteria
- References

Outline

- Introduction to DHCP
  - What is DHCP?
  - DHCP Workflow
  - DHCP Utility Setup
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
- Submission & Scoring Criteria
- References

What is DHCP?

- Dynamic Host Configuration Protocol
- Provide necessary information for a host to access network
  - IP address, gateway, DNS (Domain Name Server) and etc.
- Client and server use UDP port 68 and 67, respectively
- A DHCP transaction is completed by 4 messages:



Outline

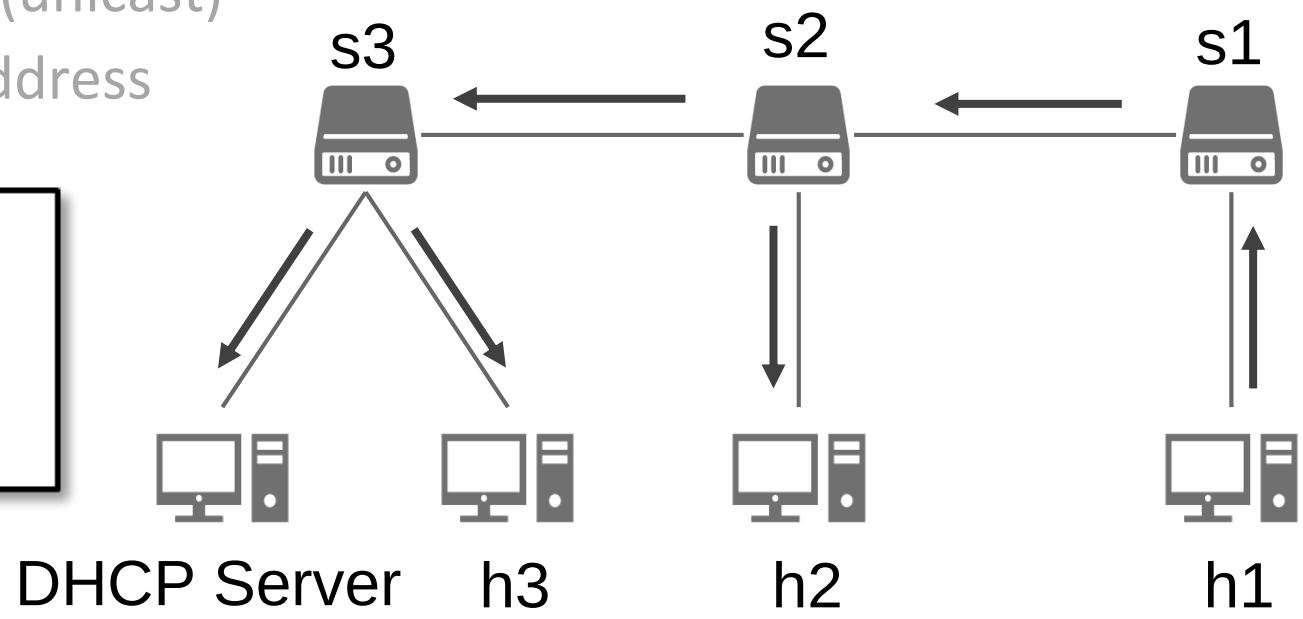
- Introduction to DHCP
  - What is DHCP?
  - DHCP Workflow
  - DHCP Utility Setup
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
- Submission & Scoring Criteria
- References

DHCP Discover

- h1 attaches to network
  - Issue DHCPDISCOVER to locate available DHCP server (broadcast)
- A DHCP server receives DHCPDISCOVER
  - Reply DHCPOFFER (unicast or broadcast)
- h1 chooses a server to reply DHCPREQUEST (broadcast)
- The server replies with DHCPACK (unicast)
  - h1 now owns the assigned IP address

```
Src IP:  0.0.0.0
Dst IP:  255.255.255.255
Src MAC: <MAC of h1>
Dst MAC: ff:ff:ff:ff:ff:ff
```

DHCP DISCOVER

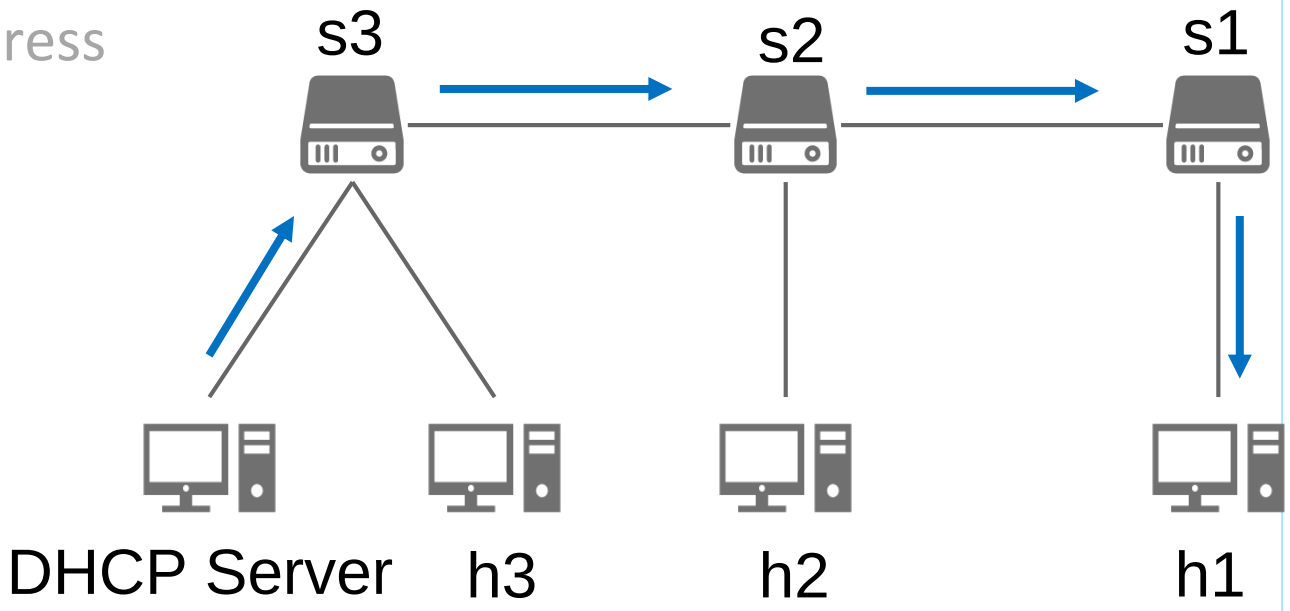


DHCP Offer

- h1 attaches to network
  - Issue DHCPDISCOVER to locate available DHCP server (broadcast)
- A DHCP server receives DHCPDISCOVER
  - Reply DHCPOFFER (unicast or broadcast)
- h1 chooses a server to reply DHCPREQUEST (broadcast)
- The server replies with DHCPACK (unicast)
  - h1 now owns the assigned IP address

```
Src IP:  <IP of server>
Dst IP:  <IP of h1>
Src MAC: <MAC of server>
Dst MAC: <MAC of h1>
Your IP address: 10.0.0.2
Subnet Mask: 255.255.255.0
IP Address Lease Time: 3600
```

DHCP OFFER

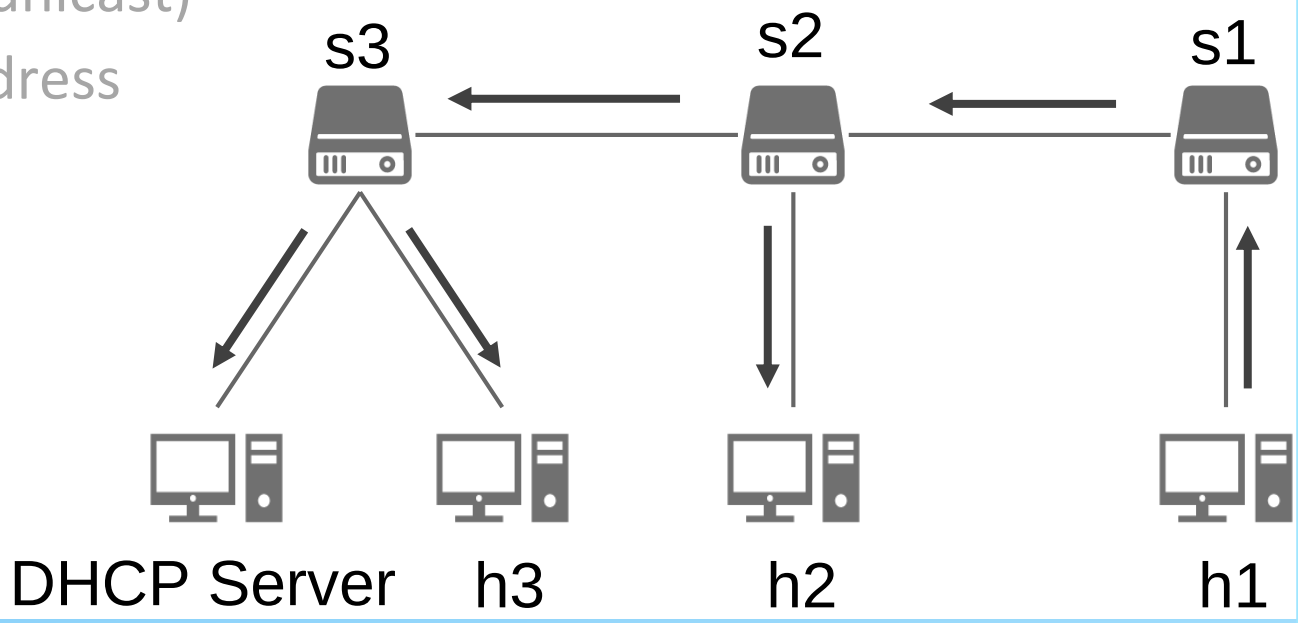


DHCP Request

- h1 attaches to network
  - Issue DHCPDISCOVER to locate available DHCP server (broadcast)
- A DHCP server receives DHCPDISCOVER
  - Reply DHCPOFFER (unicast or broadcast)
- h1 chooses a server to reply DHCPREQUEST (broadcast)
- The server replies with DHCPACK (unicast)
  - h1 now owns the assigned IP address

```
Src IP:  0.0.0.0
Dst IP:  255.255.255.255
Src MAC: <MAC of h1>
Dst MAC: ff:ff:ff:ff:ff:ff
Requested IP address: 10.0.0.2
DHCP Server Identifier: <server IP>
```

DHCP REQUEST

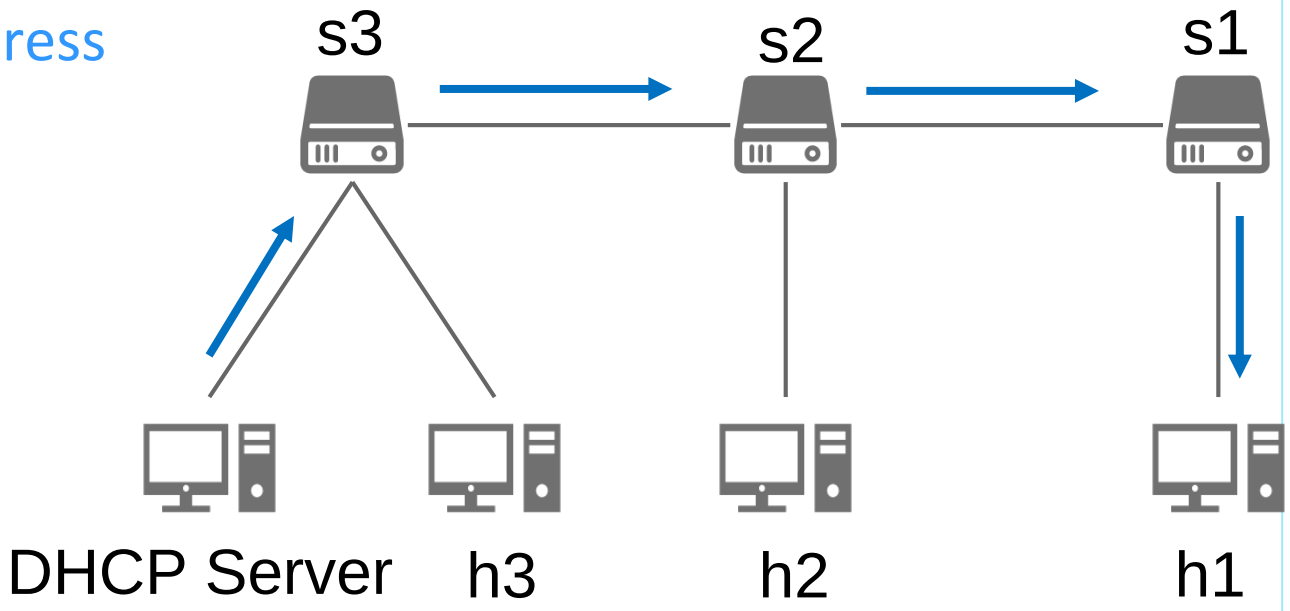


DHCP Ack

- h1 attaches to network
  - Issue DHCPDISCOVER to locate available DHCP server (broadcast)
- A DHCP server receives DHCPDISCOVER
  - Reply DHCPOFFER (unicast or broadcast)
- h1 chooses a server to reply DHCPREQUEST (broadcast)
- The server replies with DHCPACK (unicast)
  - h1 now owns the assigned IP address

```
Src IP:  <IP of server>
Dst IP:  <IP of h1>
Src MAC: <MAC of server>
Dst MAC: <MAC of h1>
Your IP address: 10.0.0.2
Subnet Mask: 255.255.255.0
IP Address Lease Time: 3600
```

DHCP OFFER



Outline

- Introduction to DHCP
  - What is DHCP?
  - DHCP Workflow
  - DHCP Utility Setup
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
- Submission & Scoring Criteria
- References

DHCP Utility Setup

- Install DHCP utility (isc-dhcp-server) before starting this project

```
bash$ sudo apt update && sudo apt install isc-dhcp-server
```

- To use dhcpd inside mininet host properly, you should modify AppArmor settings (**only need to be done for the first time**)
  - For server

```
bash$ sudo ln -s /etc/apparmor.d/usr.sbin.dhcpd \
                 /etc/apparmor.d/disable/
bash$ sudo apparmor_parser -R /etc/apparmor.d/usr.sbin.dhcpd
```

  - For client

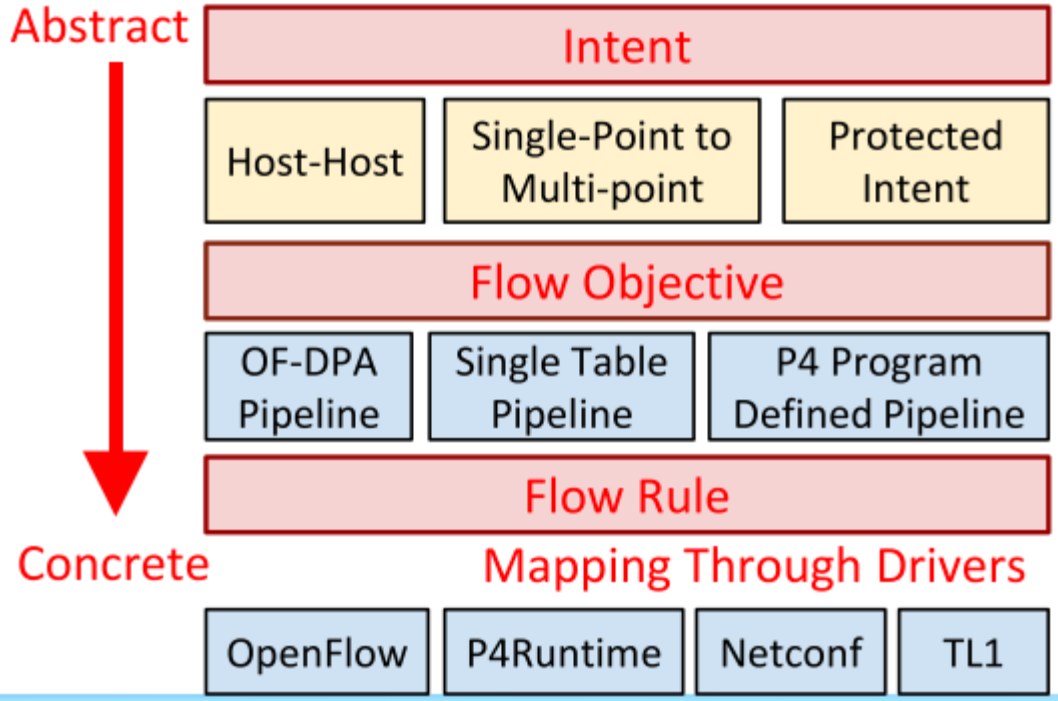
```
bash$ sudo /etc/init.d/apparmor stop
bash$ sudo sed -i '30i /var/lib/dhcp{,3}/dhcpclient* lrw,' \
                       /etc/apparmor.d/sbin.dhclient
bash$ sudo /etc/init.d/apparmor start
```

Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
- Submission & Scoring Criteria
- References

Introduction to Intent Service (1/2)

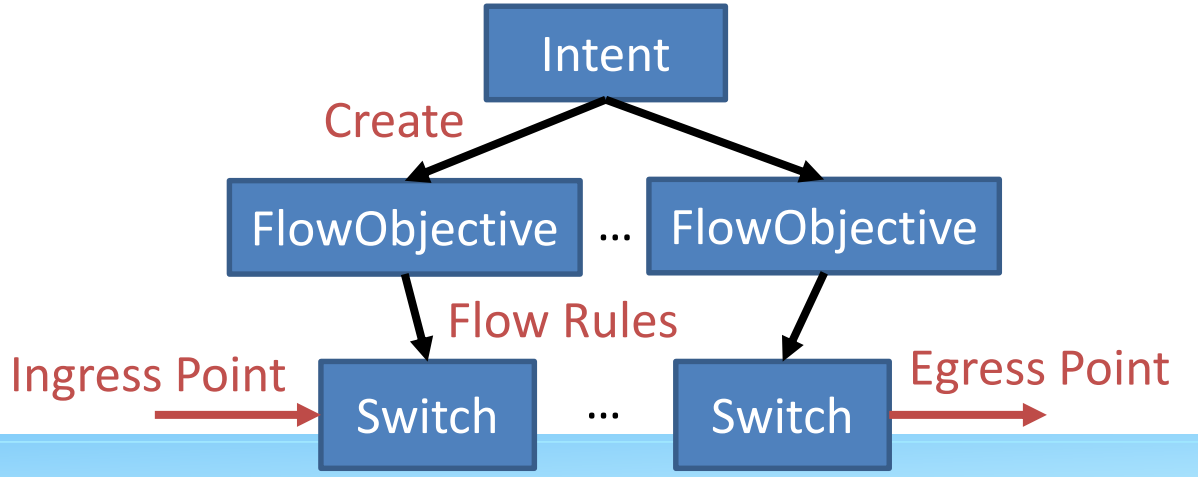
- An intent is used to define flow rules in traffic view and global way
- Provide a high-level, network-centric abstraction
  - Focuses on ***what*** should be done
    - Rather than ***how*** it is specifically programmed



ONOS Reference

Introduction to Intent Service (2/2)

- For each intent, we need to define
  - **Ingress Point**: ConnectPoint where packets enter the SDN network
    - One point for one point to one point intent
    - A set of points for multi point to one point intent
  - **Egress Point**: ConnectPoint where packets leave the SDN network
  - **Traffic Selector**: Define what kind of packet this intent processes
  - **Traffic Treatment**: Define how to modify the packet
  - **Priority**: Priority for every flow rule this intent creates



Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
  - Overview
  - Example ONOS Application
- Unicast DHCP Function
- Submission & Scoring Criteria
- References

ONOS Network Config Service

- Purpose
  - Configure ONOS apps that provide network services
  - Add information about devices, links, and device configuration into ONOS's network view
- Functionality
  - Provide an extendable configuration database
  - Provide a restful API endpoint for configuration upload

ONOS wiki page

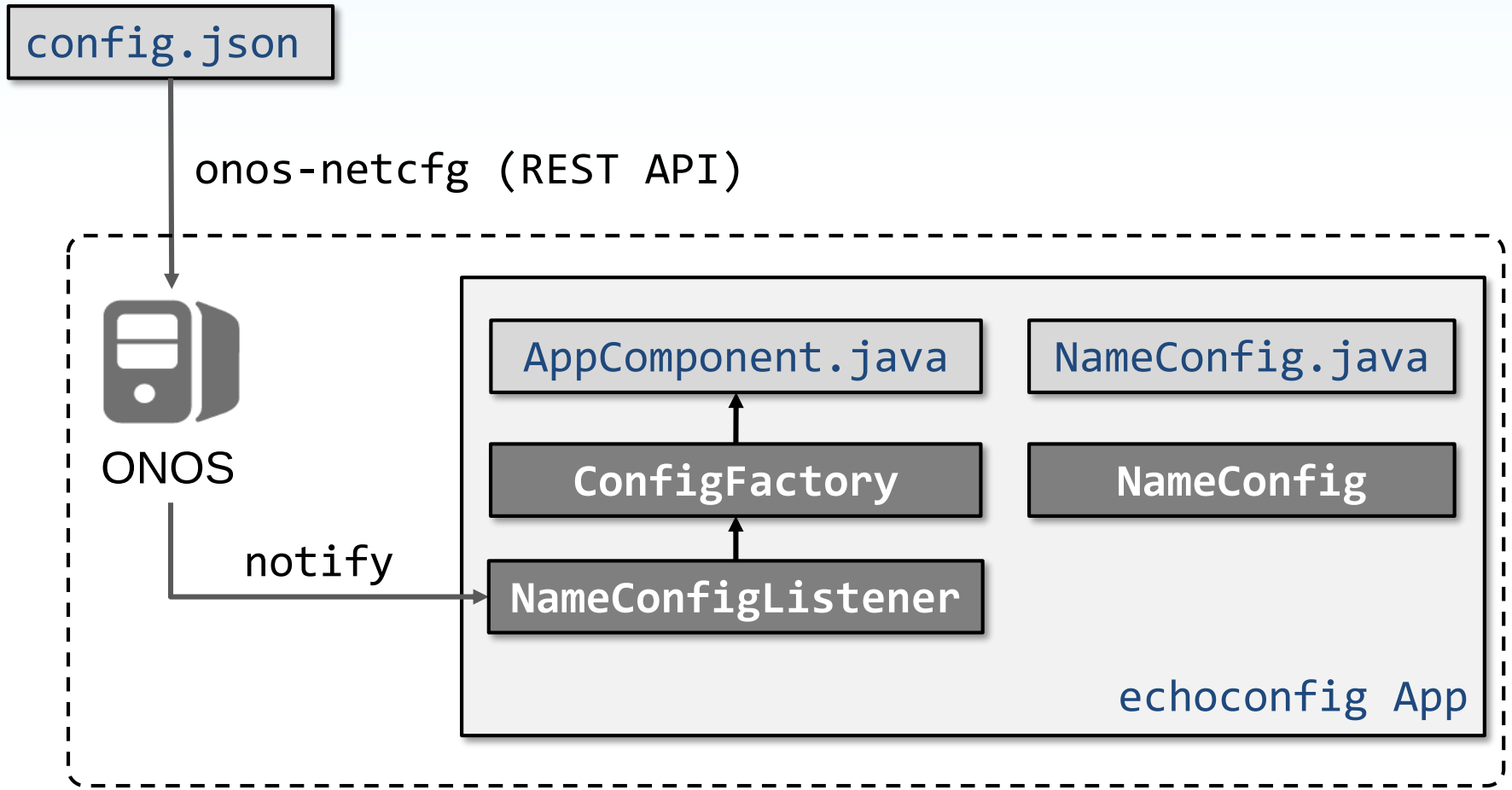
Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
  - Overview
  - Example ONOS Application
- Unicast DHCP Function
- Submission & Scoring Criteria
- References

Example Application – echoconfig

- ***echoconfig*** App :
  echoes (prints out) a name specified in the configuration file
- Components of ***echoconfig***
  1. *AppComponent.java*: main program of ***echoconfig*** that
     - Listens to configuration file uploaded event
     - Instantiates a ***NameConfig*** object
     - Prints value of name specified in configuration file
  2. *NameConfig.java*
     - Provides functions to validate and retrieve data from ***config.json***
- Configuration file of ***echoconfig***
  3. *config.json*
     - Provides some information

echoconfig APP and Configuration Uploading



NameConfig.java – NameConfig Class

- Provide functions to:
  - Validate contents of ***config.json***
    - e.g. Check presence of required fields
  - Retrieve "name" value from ***config.json***

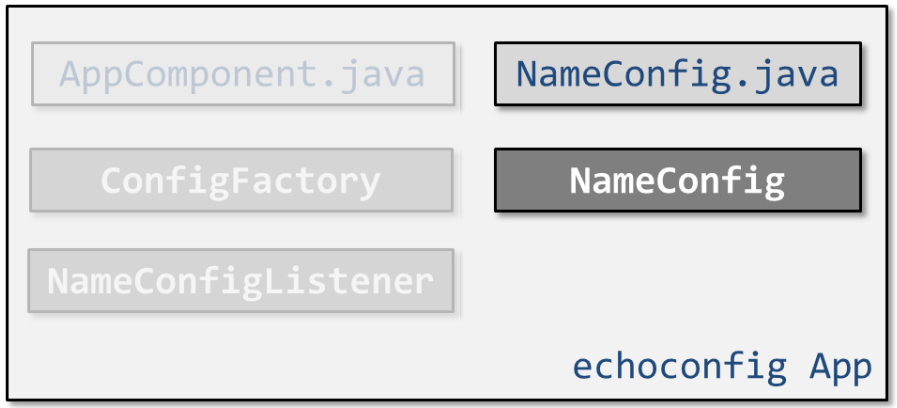
```java
public class NameConfig extends Config<ApplicationId> {

  public static final String NAME = "name";

  @Override
  public boolean isValid() {
    return hasOnlyFields(NAME);
  }

  public String name() {
    return get(NAME, defaultValue: null);
  }
}
```

Retrieve "name" value

NameConfig.java

```json
{
  "apps": {
    "nctu.winlab.echoconfig": {
      "whoami": {
        "name": "Magikarp"
      }
    }
  }
}
```

config.json



AppComponent.java – ConfigFactory

1. Instantiate a ***factory*** for creating a ***NameConfig*** object
   - The arguments serve as key for ONOS to select the correct factory

```java
private final ConfigFactory<ApplicationId, NameConfig> factory =
    new ConfigFactory<ApplicationId, NameConfig>(
        APP_SUBJECT_FACTORY, NameConfig.class, "whoami") {
    @Override
    public NameConfig createConfig() {
        return new NameConfig();
    }
};
```

Instantiate a factory

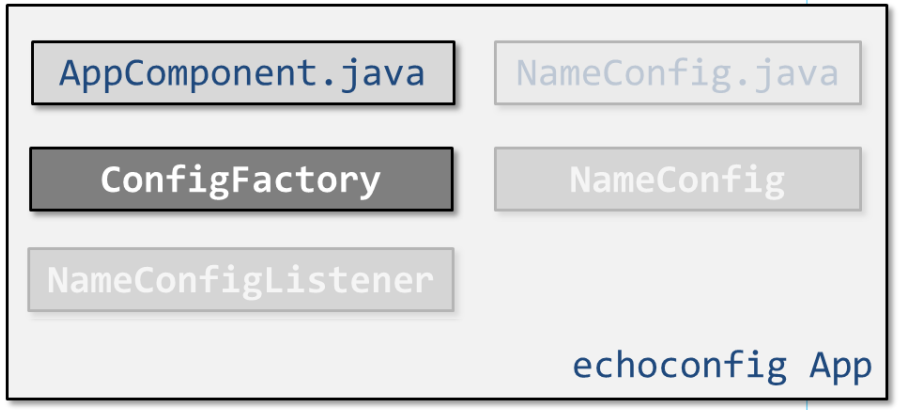
```json
{
  "apps": {
    "nctu.winlab.echoconfig": {
      "whoami": {
        "name": "Magikarp"
      }
    }
  }
}
```

config.json

2. Register ***factory*** with ONOS

```java
@Activate
protected void activate() {
    appId = coreService.registerApplication(name: "nctu.winlab.echoconfig");
    cfgService.addListener(cfgListener);
    cfgService.registerConfigFactory(factory);
    log.info(msg: "Started");
}
```

Register factory



AppComponent.java – NameConfigListener

1. Implement ***NameConfigListener*** Class and instantiate a listener
   - Listen to network configuration event (e.g. A config file is uploaded)
   - ONOS will call event() when it receives event

```
private class NameConfigListener implements NetworkConfigListener {
  @Override
  public void event(NetworkConfigEvent event) {
    if ((event.type() == CONFIG_ADDED || event.type() == CONFIG_UPDATED)
        && event.configClass().equals(NameConfig.class)) {
      NameConfig config = cfgService.getConfig(appId, configClass: NameConfig.class);
      if (config != null) {
        log.info(format: "It is {}!", config.name());
      }
```

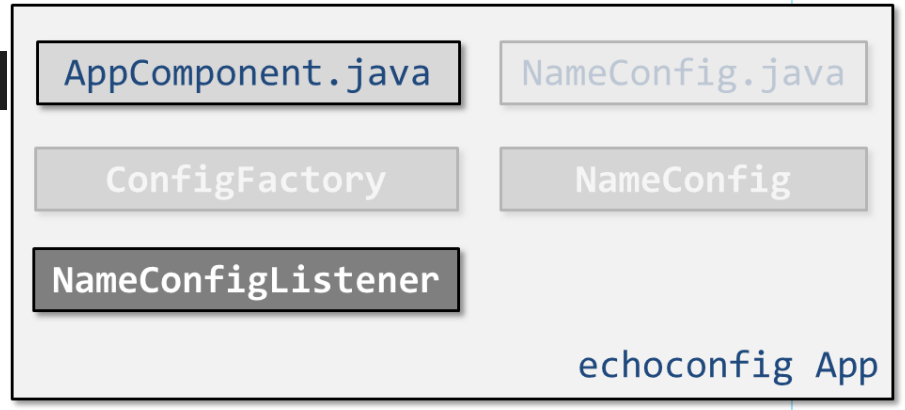
Implement Listener

2. Instantiate a listener

```
private final NameConfigListener cfgListener = new NameConfigListener();
```

3. Register the listener object with ONOS

```
@Activate
protected void activate() {
  appId = coreService.registerApplication(
  cfgService.addListener(cfgListener);
```



echoconfig Demonstration

1. Build, install and activate the app
2. Upload ***config.json***

```
bash$ onos-netcfg localhost config.json
```

3. ONOS log will show following message:

```
{
  "apps": {
    "nctu.winlab.echoconfig": {
      "whoami": {
        "name": "Magikarp"
      }
    }
  }
}
```

config.json

```
| 11 - org.apache.karaf.features.core - 4.2.9 | Changes to perform:
| 11 - org.apache.karaf.features.core - 4.2.9 |   Region: root
| 11 - org.apache.karaf.features.core - 4.2.9 |     Bundles to install:
| 11 - org.apache.karaf.features.core - 4.2.9 |       mvn:nctu.winlab/echoconfig/1.0-SNAPSHOT
| 11 - org.apache.karaf.features.core - 4.2.9 | Installing bundles:
| 11 - org.apache.karaf.features.core - 4.2.9 |   mvn:nctu.winlab/echoconfig/1.0-SNAPSHOT
| 11 - org.apache.karaf.features.core - 4.2.9 | Starting bundles:
| 11 - org.apache.karaf.features.core - 4.2.9 |   nctu.winlab.echoconfig/1.0.0.SNAPSHOT
| 219 - nctu.winlab.echoconfig - 1.0.0.SNAPSHOT | Started
| 11 - org.apache.karaf.features.core - 4.2.9 | Done.
          | 193 - org.onosproject.onos-core-net - 2.7.0 | Application nctu.winlab.echoconfig has been activated
         | 219 - nctu.winlab.echoconfig - 1.0.0.SNAPSHOT | It is Magikarp!
```

ONOS log

Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
  - Overview & Workflow
  - How to Test Your App
  - Supplements
- Submission & Scoring Criteria
- References

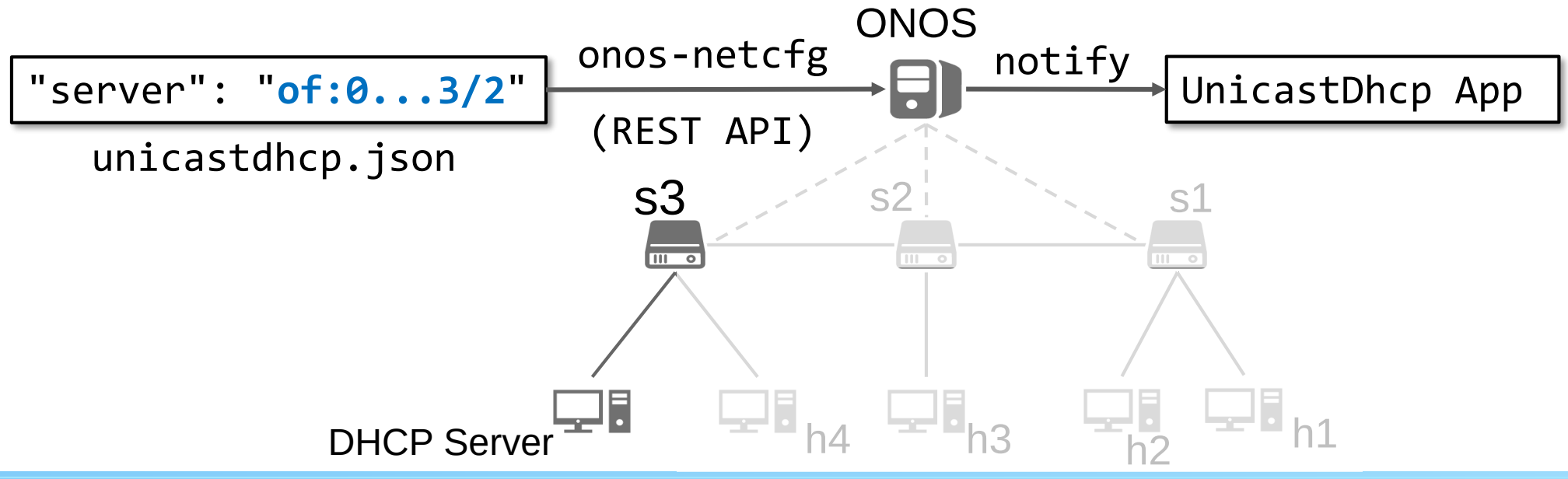
Overview

- DHCP protocol generates **broadcast** packets in the network
- You need to implement a **unicast** DHCP application
- Workflow for this application
  1. Configure a DHCP server location
  2. Install flow rules to packet-in DHCP packets
  3. Create intent for DHCP client and server to communicate

Step 1 – Configure DHCP Server Location

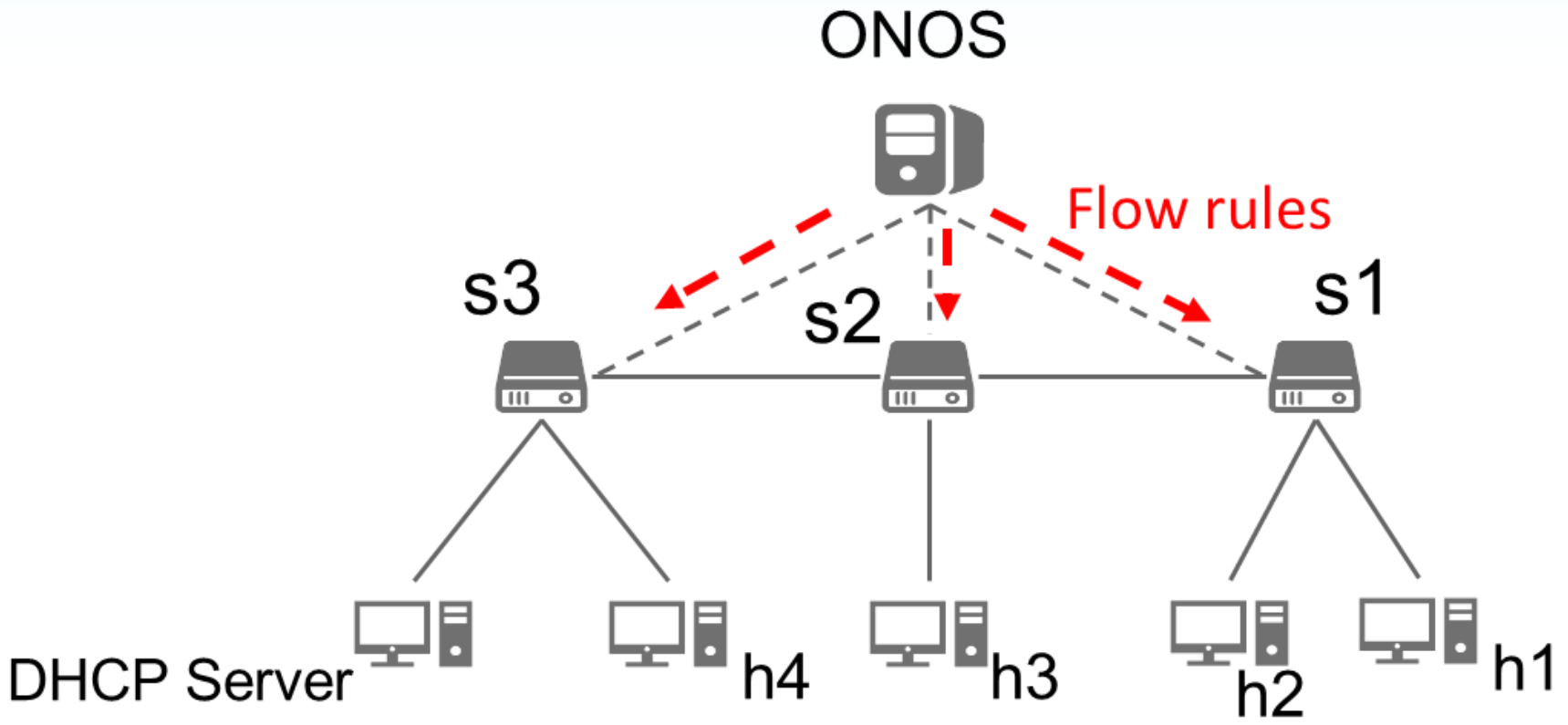
- Describe the ConnectPoint of DHCP server in ***unicastdhcp.json***
- Upload the file to ONOS configuration service via REST API
- Should print configured location to ONOS log when notified

```
bash$ onos-netcfg localhost unicastdhcp.json
```



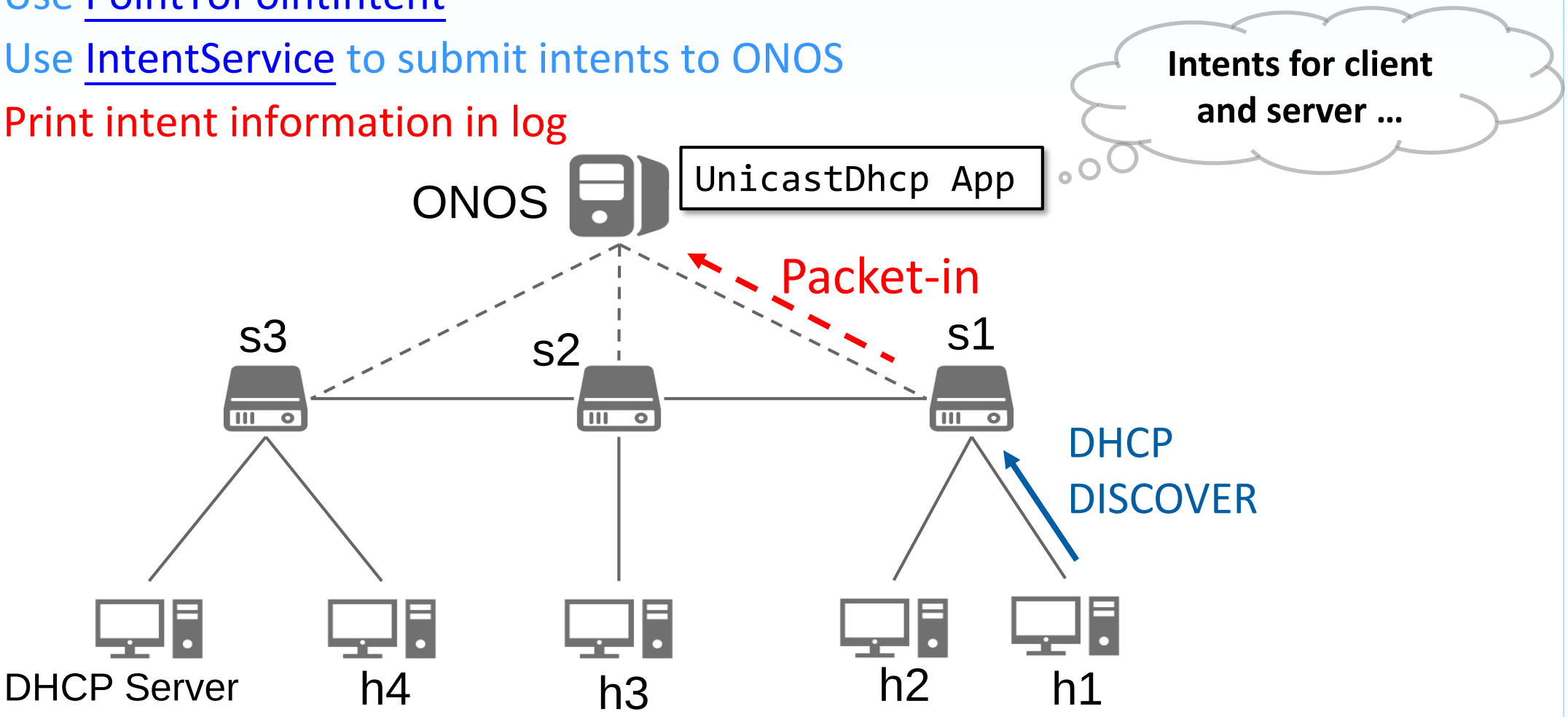
Step 2 – Packet-In DHCP Packets

- Request switches to packet-in DHCP packets



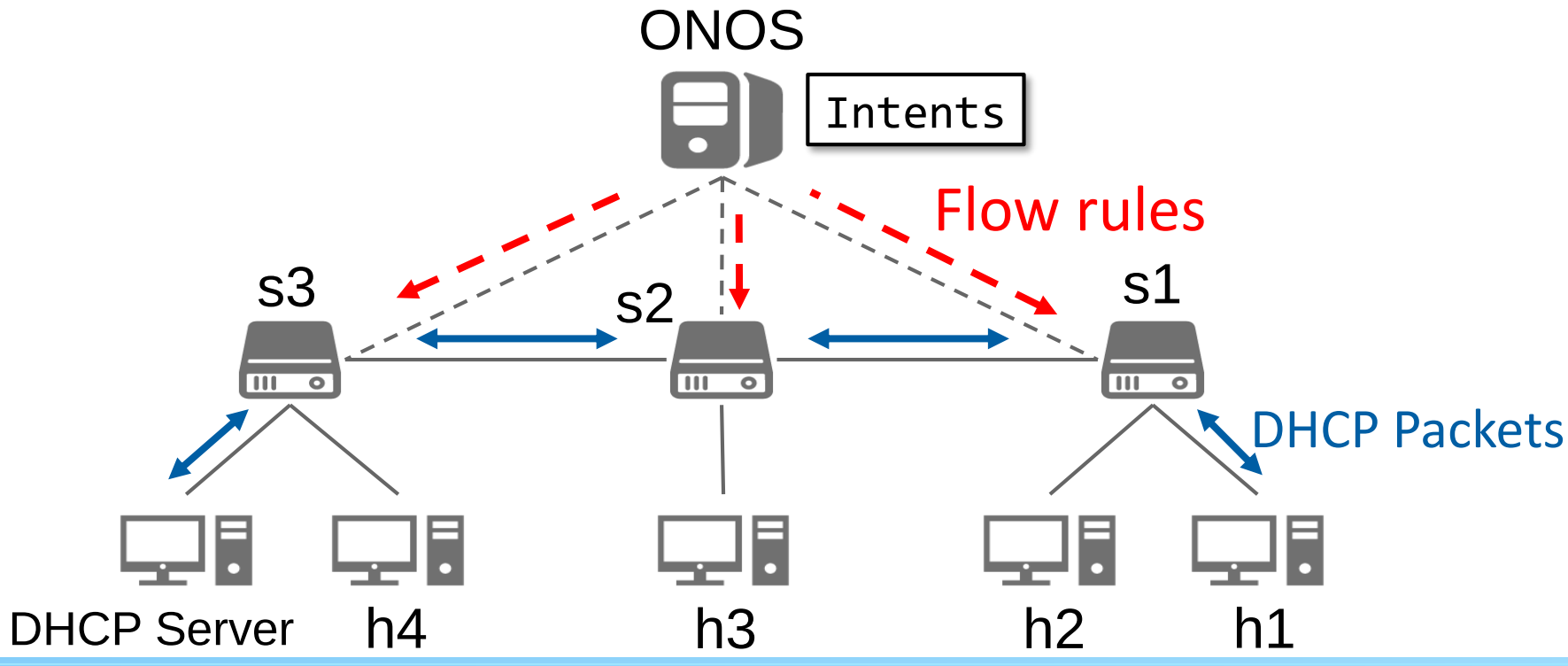
Step 3 – Create Intents (1/2)

- Create **intents** for DHCP client and server to communicate
  - Use PointToPointIntent
  - Use IntentService to submit intents to ONOS
  - Print intent information in log



Step 3 – Create Intents (2/2)

- Intents will install flow rules to forward DHCP packets
- Subsequent DHCP packets should all become unicast
  - DISCOVER, OFFER, REQUEST, ACK
- Interfaces not on the path should not receive any DHCP packet



Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
  - Overview & Workflow
  - How to Test Your App
  - Supplements
- Submission & Scoring Criteria
- References

How to Test Your App (1/2)

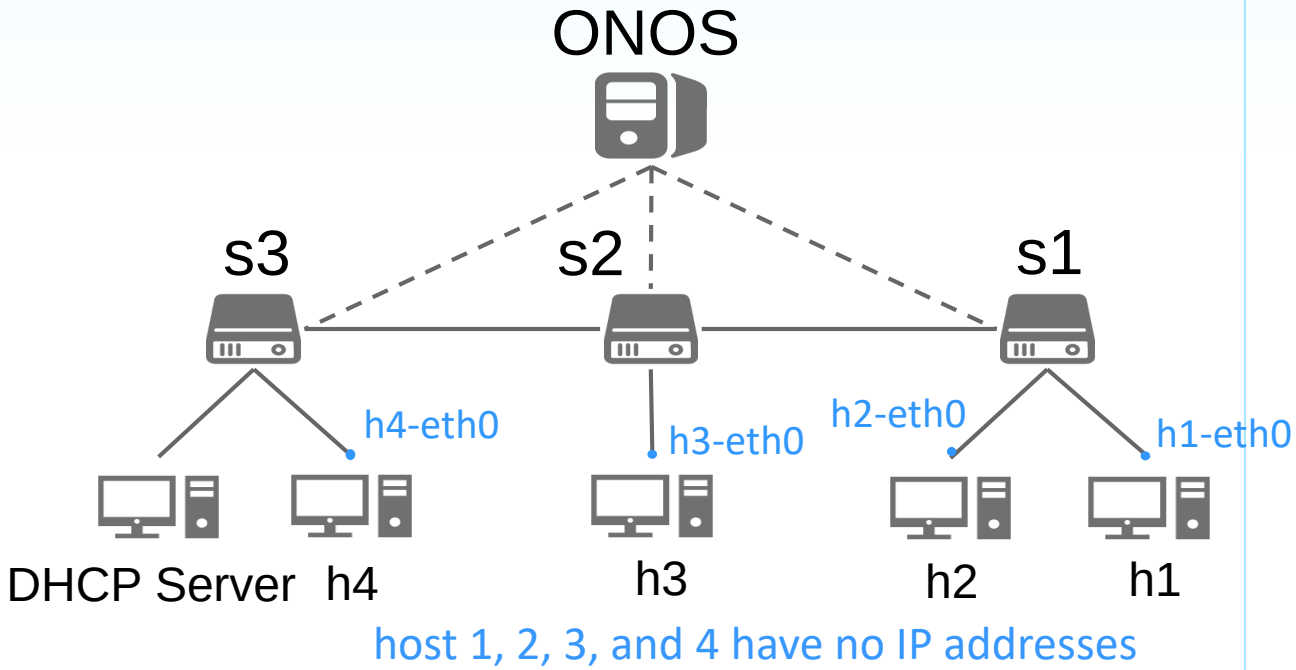
1. Setup DHCP utilities in p.11
2. Build, install and activate your app

```
bash$ mvn clean install -Dskiptests
bash$ onos-app localhost install! \
      <path to .oar>
```

3. Run ***topo.py*** to build the topology and enter mininet CLI

```
bash$ chmod +x topo.py
bash$ sudo ./topo.py
```

(Please run ***topo.py*** in the directory containing ***dhcpd.conf***)



4. Upload your config file to ONOS

```
bash$ onos-netcfg localhost unicastdhcp.json
```

How to Test Your App (2/2)

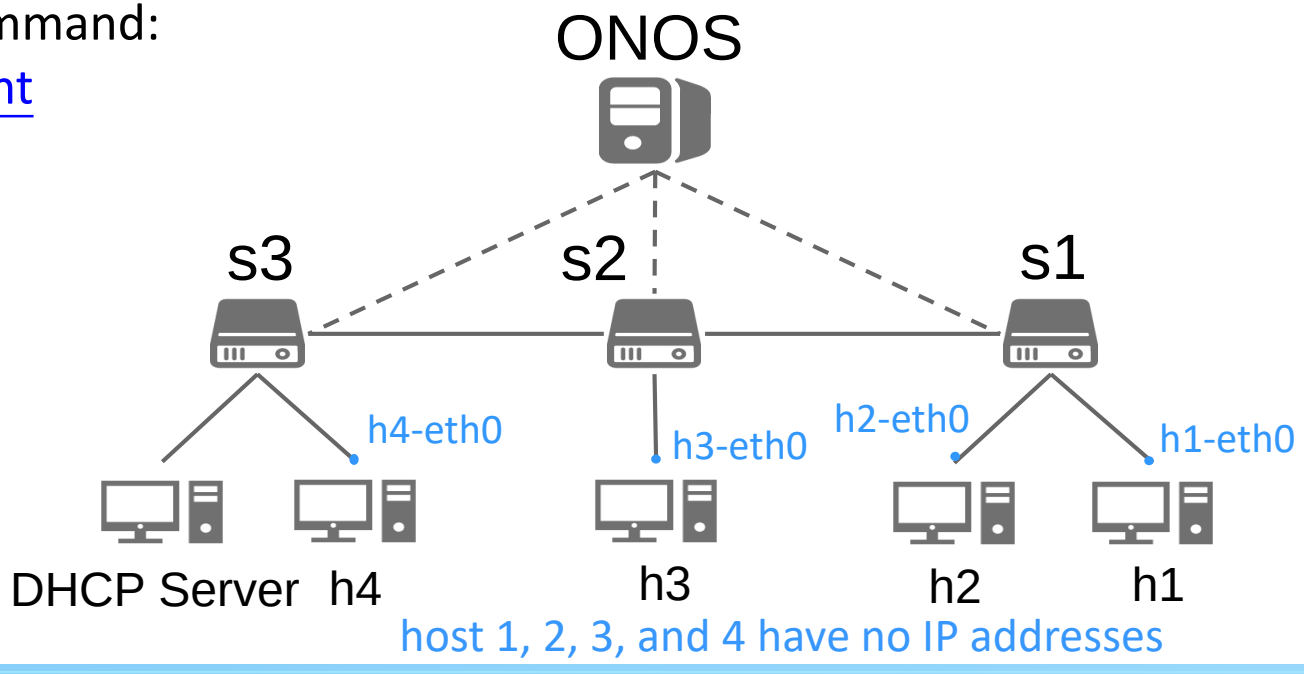
5. Try to acquire an IP address on a host

```
mininet> h1 dhclient -v h1-eth0 (-v: Observe all messages of a DHCP transaction)
```

✔ Note: Release current lease before re-issuing a DHCP request on an interface

```
mininet> h1 dhclient -r h1-eth0 (-r: Release current lease)
```

More about the command:
[Man page of dhclient]



Demonstration

1. h1-eth0 does not have an IPv4 address yet

```
mininet> h1 ifconfig h1-eth0
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::e8e9:78ff:fefb:fd01  prefixlen 64  scopeid 0x20<link>
        ether ea:e9:78:fb:fd:01  txqueuelen 1000  (Ethernet)
```

2. Observe DHCP procedure on h1-eth0

```
mininet> h1 dhclient -v h1-eth0
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/h1-eth0/ea:e9:78:fb:fd:01
Sending on   LPF/h1-eth0/ea:e9:78:fb:fd:01
Sending on   Socket/fallback
DHCPDISCOVER on h1-eth0 to 255.255.255.255 port 67 interval 3 (xid=0xd74d5b7c)
DHCPDISCOVER on h1-eth0 to 255.255.255.255 port 67 interval 3 (xid=0xd74d5b7c)
DHCPREQUEST of 10.1.11.100 on h1-eth0 to 255.255.255.255 port 67 (xid=0x7c5b4dd7)
DHCPOFFER of 10.1.11.100 from 10.1.11.3
DHCPACK of 10.1.11.100 from 10.1.11.3
bound to 10.1.11.100 -- renewal in 232 seconds.
```

DHCP Messages

3. h1-eth0 now has an IPv4 address

```
mininet> h1 ifconfig h1-eth0
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.11.100  netmask 255.255.255.0  broadcast 10.1.11.255
        inet6 fe80::e8e9:78ff:fefb:fd01  prefixlen 64  scopeid 0x20<link>
        ether ea:e9:78:fb:fd:01  txqueuelen 1000  (Ethernet)
```

Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
  - Overview & Workflow
  - How to Test Your App
  - Supplements
- Submission & Scoring Criteria
- References

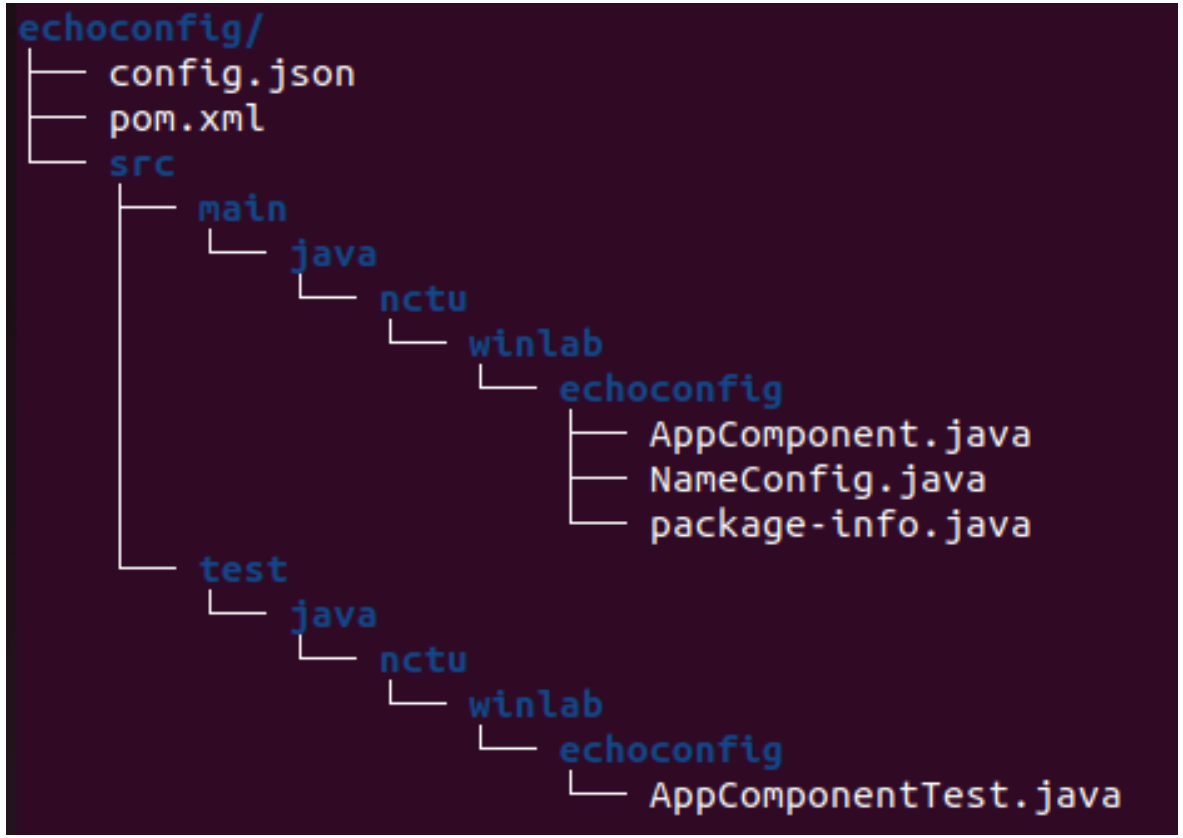
Supplements (1/3)

- These files will be provided to you:

```
SDNFV_project4/
├── echoconfig
│   ├── config.json
│   ├── pom.xml
│   └── src
│       ├── main
│       │   └── java
│       │       └── nctu
│       │           └── winlab
│       │               └── echoconfig
│       │                   ├── AppComponent.java
│       │                   ├── NameConfig.java
│       │                   └── package-info.java
│       └── test
│           └── java
│               └── nctu
│                   └── winlab
│                       └── echoconfig
│                           └── AppComponentTest.java
├── topo
│   ├── dhcpd.conf
│   └── topo.py
└── unicastdhcp.json

13 directories, 9 files
```

Supplements (2/3)

1. Program and configuration files of a sample application ***echoconfig***
   - You can directly build the app with these files



Supplements (3/3)

2. Network Topology files for Unicast DHCP App
   - ***topo.py***: mininet topology
     - Executing this python script will directly enter mininet CLI
   - ***dhcpd.conf***: DHCP server configuration used by topo.py
3. Configuration file for Unicast DHCP App
   - ***unicastdhcp.json***: configuration file for unicast DHCP app

Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
- Submission & Scoring Criteria
- References

Scoring Criteria (1/3)

- **(10%)** Project naming convention
  - <groupId>: nctu.winlab
  - <artifactId>: unicastdhcp
  - <version>: <use default> (1.0-SNAPSHOT)
  - <package>: nctu.winlab.unicastdhcp
- **(30%)** Acquire DHCP server location from the configuration file
  - You must use classes under org.onosproject.net.config
    - Otherwise, ZERO point is given in this section
  - Print the correct DHCP server location in ONOS log
    - Message format: "DHCP server is connected to `{device ID}`, port `{port}`"
    - Wrong format or no output is a 10-point deduction

Scoring Criteria (2/3)

- **(40%)** Use ***IntentService*** to install flow rules
  - You must NOT use any class under org.onosproject.net.flowobjective
  - You must use PointToPointIntent and IntentService
    - If any of the two above rules is violated, ZERO points are given in this section
  - IP address acquisition using ***dhclient***
    - Every host in the topology should be able to acquire an IP-address
    - For every host that can't acquire an IP address, 5 points are deducted
  - Print intent information and submission in ONOS log
    - Message format:

    "Intent `{ingress device ID}`, port `{ingress port}` => `{egress device ID}`, port `{egress port}` is submitted."

    - Wrong format or no output is a 10-point deduction

Scoring Criteria (3/3)

- **(20%)** DHCP transaction packets must be unicasting
  - Hosts that are not involved in the transaction should not receive any packet
- Reminders
  - Your application should be functional on a different topology as well
  - In this project, it is not required to use ping to check connectivity.
    - For simplicity, you should not activate fwd application
    - We will not activate fwd before testing your app

```
sclin@root > apps -a -s
*   4 org.onosproject.gui2                  2.7.0     ONOS GUI2
*  18 org.onosproject.hostprovider          2.7.0     Host Location Provider
*  20 org.onosproject.lldpprovider          2.7.0     LLDP Link Provider
*  21 org.onosproject.optical-model         2.7.0     Optical Network Model
*  22 org.onosproject.openflow-base         2.7.0     OpenFlow Base Provider
*  23 org.onosproject.openflow              2.7.0     OpenFlow Provider Suite
*  43 org.onosproject.drivers               2.7.0     Default Drivers
* 177 nctu.winlab.unicastdhcp               1.0.SNAPSHOT Unicastdhcp App
sclin@root >
```

Submission Naming Convention

- Rename your unicast DHCP app directory as **project4_<student ID>**.
- Compress the directory into a **zip** file named as **project4_<student ID>.zip.**
- Upload your zip file to E3.
- 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

```
project4_ID/
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── nctu
    │           └── winlab
    │               └── unicastdhcp
    │                   ├── AppComponent.java
    │                   ├── DhcpConfig.java
    │                   ├── package-info.java
    │                   └── SomeInterface.java
    └── test
        └── java
            └── nctu
                └── winlab
                    └── unicastdhcp
                        └── AppComponentTest.java

11 directories, 6 files
```

Demo

- TA will open a demo time-reserved table one week before demo
- The dates will be chosen after the deadline
- Demo questions will appear at the start of the demo
- The score of demo will occupy **40%** total score of this project
  - For example:
    - You earn 100% of the credits for submission
    - You earn 80% of the credits for demo
    - Then your total score of this project will be:
      100 x 60% + 80 x 40% = 92.

About help!

- For any project problem, ask at e3 forum
  - Ask at the e3 forum
  - TAs will help to clarify project 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.nctu.edu.tw**
  - You have special problem and you can't meet the deadline
  - You got weird score with project
- No Fixed TA hour

Outline

- Introduction to DHCP
- Introduction to Intent Service
- Introduction to Network Configuration Service
- Project 4 Overview
- Submission & Scoring Criteria
- References

References

- ONOS Java API 2.7.0
- Intent Framework
- The Network Configuration Service