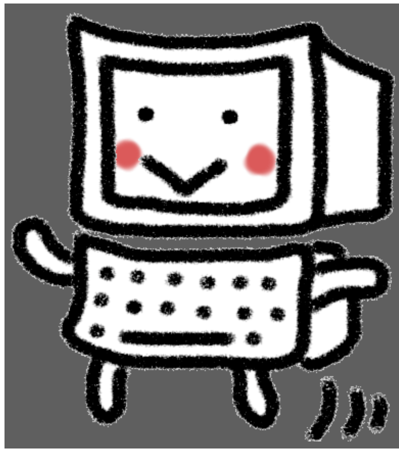


Project 6

Unicast DHCP Application

Release : 2020/05/21 (Thu.)

Deadline: 2020/06/10 (Wed.)

Outline

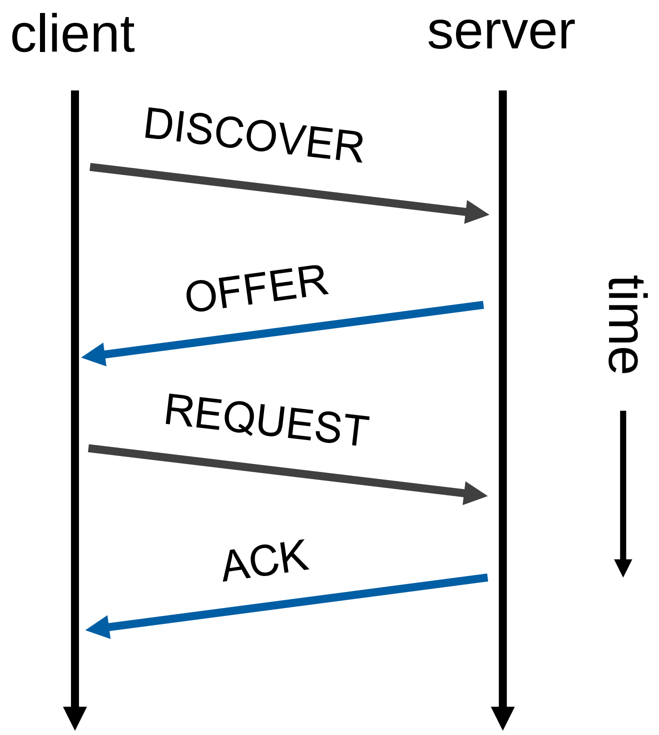
- ❑ Introduction to DHCP
- ❑ Project 6 Overall Procedure
- ❑ Environment Setup & Sample Application
- ❑ Submission

Outline

- Introduction to DHCP
- Project 6 Overall Procedure
- Environment Setup & Sample Application
- Submission

About DHCP (Dynamic Host Configuration Protocol)

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

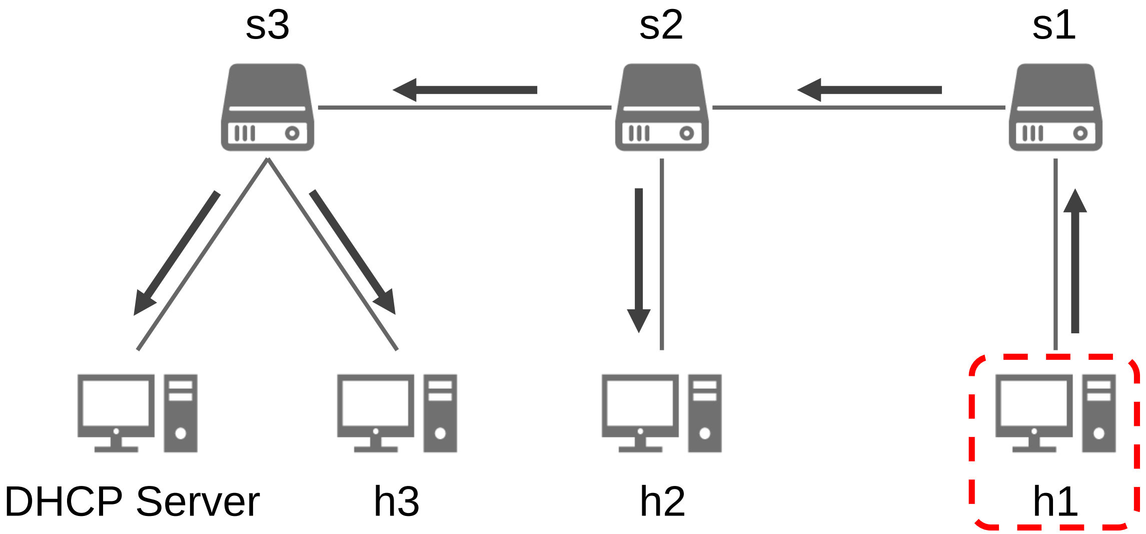


DHCP Discover

- When h1 attaches to a network, it may issue DHCPDISCOVER to locate available DHCP servers
- Upon receiving the DHCPDISCOVER, server reply DHCPOFFER
  - More than one servers may reply the request
  - Could be unicast or broadcast (depending on broadcast flag of DHCPDISCOVER)
- h1 chooses a server, replies with a DHCPREQUEST
- Server replies with DHCPACK, h1 now owns the assigned 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

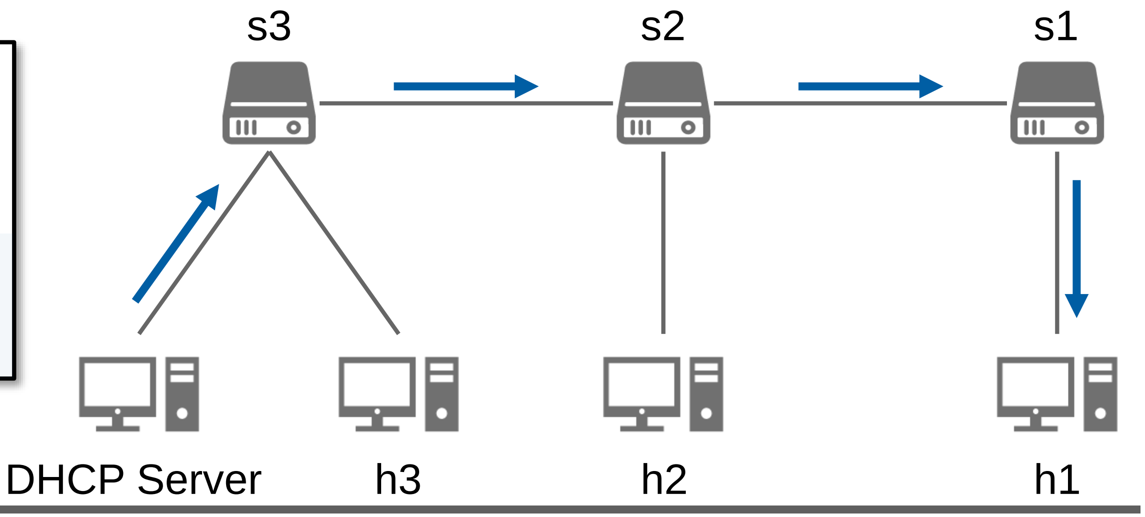


DHCP Offer

- When h1 attaches to a network, it may issue DHCPDISCOVER to locate available DHCP servers
- Upon receiving the DHCPDISCOVER, server reply DHCPOFFER
  - More than one servers may reply the request
  - Could be unicast or broadcast (depending on broadcast flag of DHCPDISCOVER)
- h1 chooses a server, replies with a DHCPREQUEST
- Server replies with DHCPACK,
  - h1 now owns the assigned address

```
Src IP:  <IP of server>
Dst IP:  <IP of h1>
Src MAC: <MAC of server>
Dst MAC: <MAC of h1>
Your IP address: 10.0.0.1
Subnet Mask: 255.255.255.0
IP Address Lease Time: 3600
```

DHCP OFFER

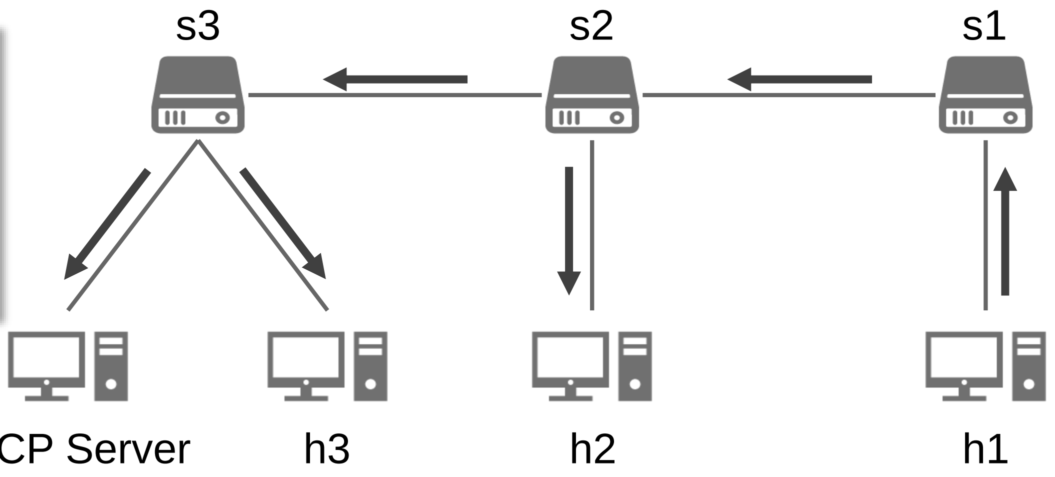


DHCP Request

- When h1 attaches to a network, it may issue DHCPDISCOVER to locate available DHCP servers
- Upon receiving the DHCPDISCOVER, server reply DHCPOFFER
  - More than one servers may reply the request
  - Could be unicast or broadcast (depending on broadcast flag of DHCPDISCOVER)
- h1 chooses a server, replies with a DHCPREQUEST
- Server replies with DHCPACK,
  - h1 now owns the assigned 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: 10.0.0.1
DHCP Server Identifier: <server IP>
```

DHCP REQUEST

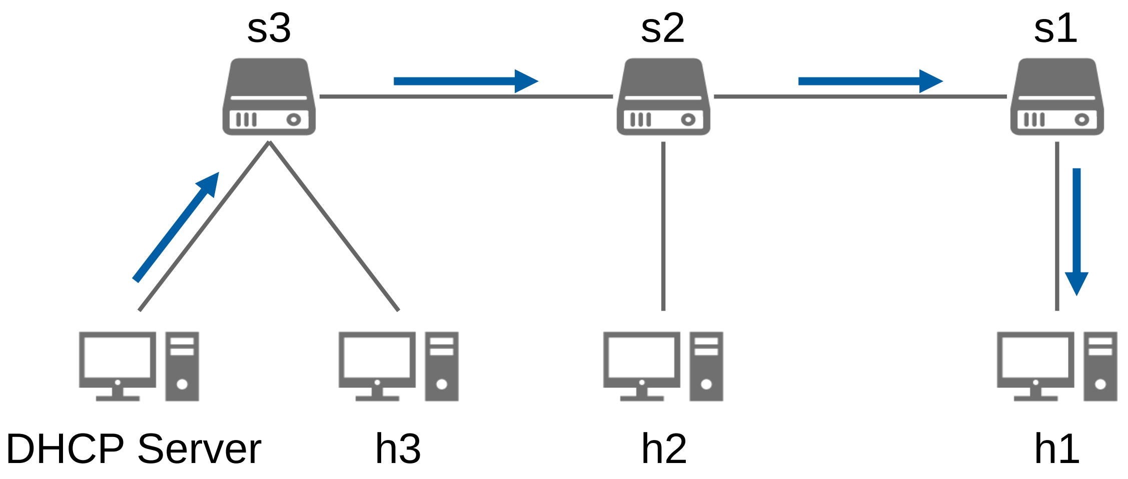


DHCP Ack

- When h1 attaches to a network, it may issue DHCPDISCOVER to locate available DHCP servers
- Upon receiving the DHCPDISCOVER, server reply DHCPOFFER
  - More than one servers may reply the request
  - Could be unicast or broadcast (depending on broadcast flag of DHCPDISCOVER)
- h1 chooses a server, replies with a DHCPREQUEST
- Server replies with DHCPACK
  - h1 now owns the assigned address

```
Src IP:  <IP of server>
Dst IP:  <IP of h1>
Src MAC: <MAC of server>
Dst MAC: <MAC of h1>
Your IP address: 10.0.0.1
Subnet Mask: 255.255.255.0
IP Address Lease Time: 3600
```

DHCP ACK



Outline

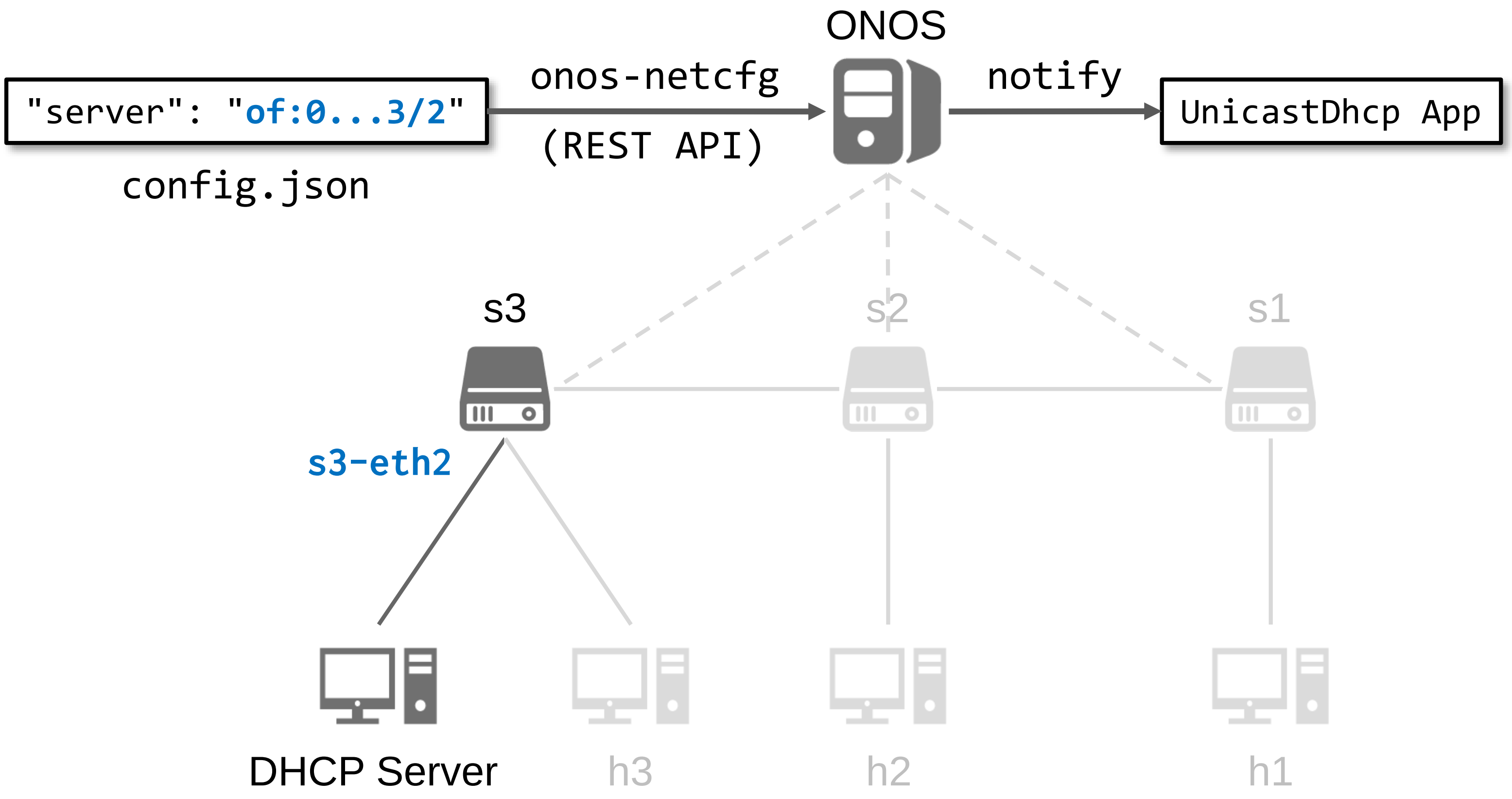
Project 6 Overall Procedure

- In this project, you need to implement an unicast DHCP application
  1. Configure a DHCP server location
  2. Install flow rules to packet-in DHCP packets
  3. Compute path between a DHCP client and the DHCP server
  4. Install flow rules to forward DHCP packets via unicast

Step 1/4 – Configure DHCP Server Location

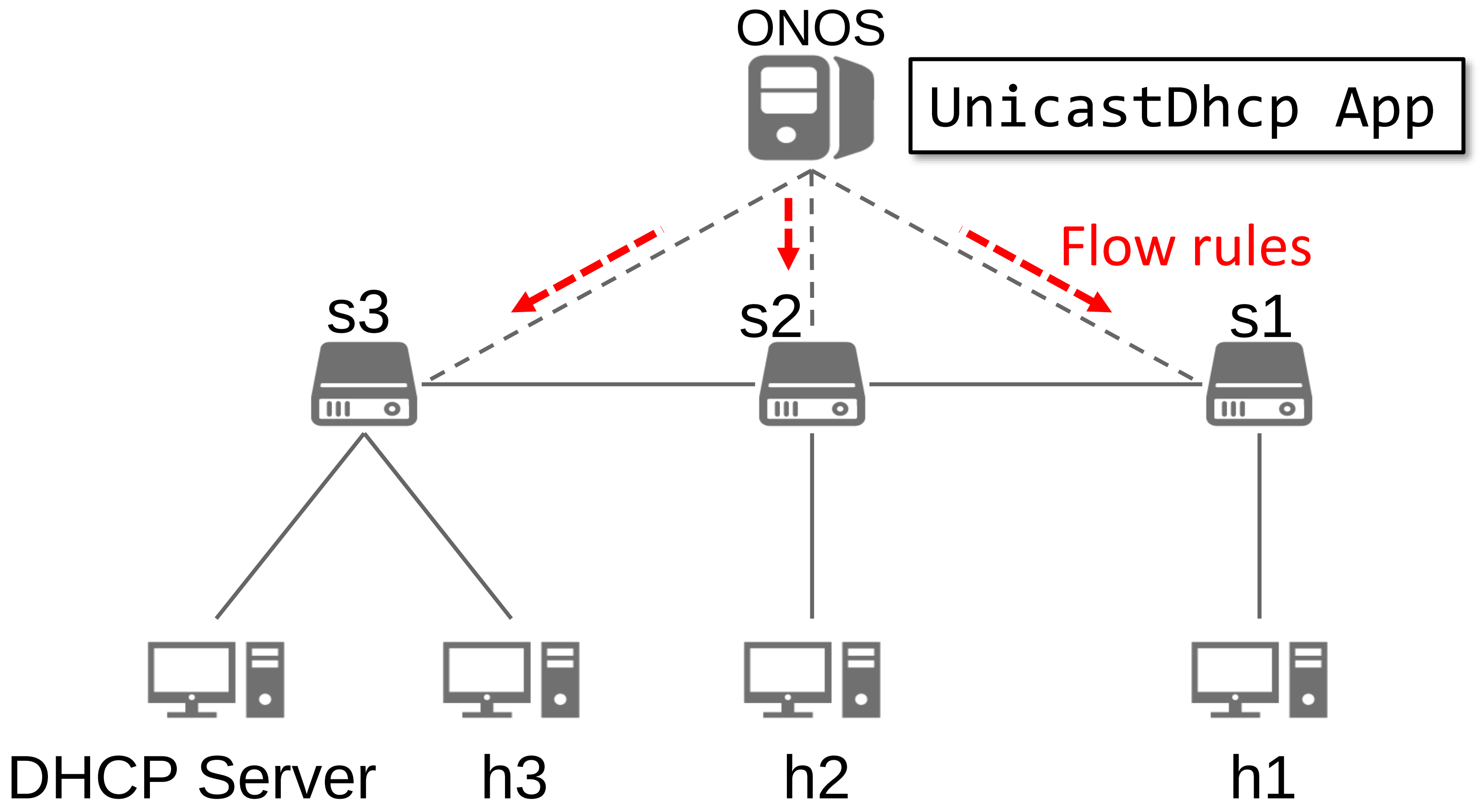
- ❑ Write a JSON file describing the `ConnectPoint` of DHCP server
- ❑ Upload the file to ONOS configuration service via REST API
- ❑ You **should** print the configured location to ONOS log when notified

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



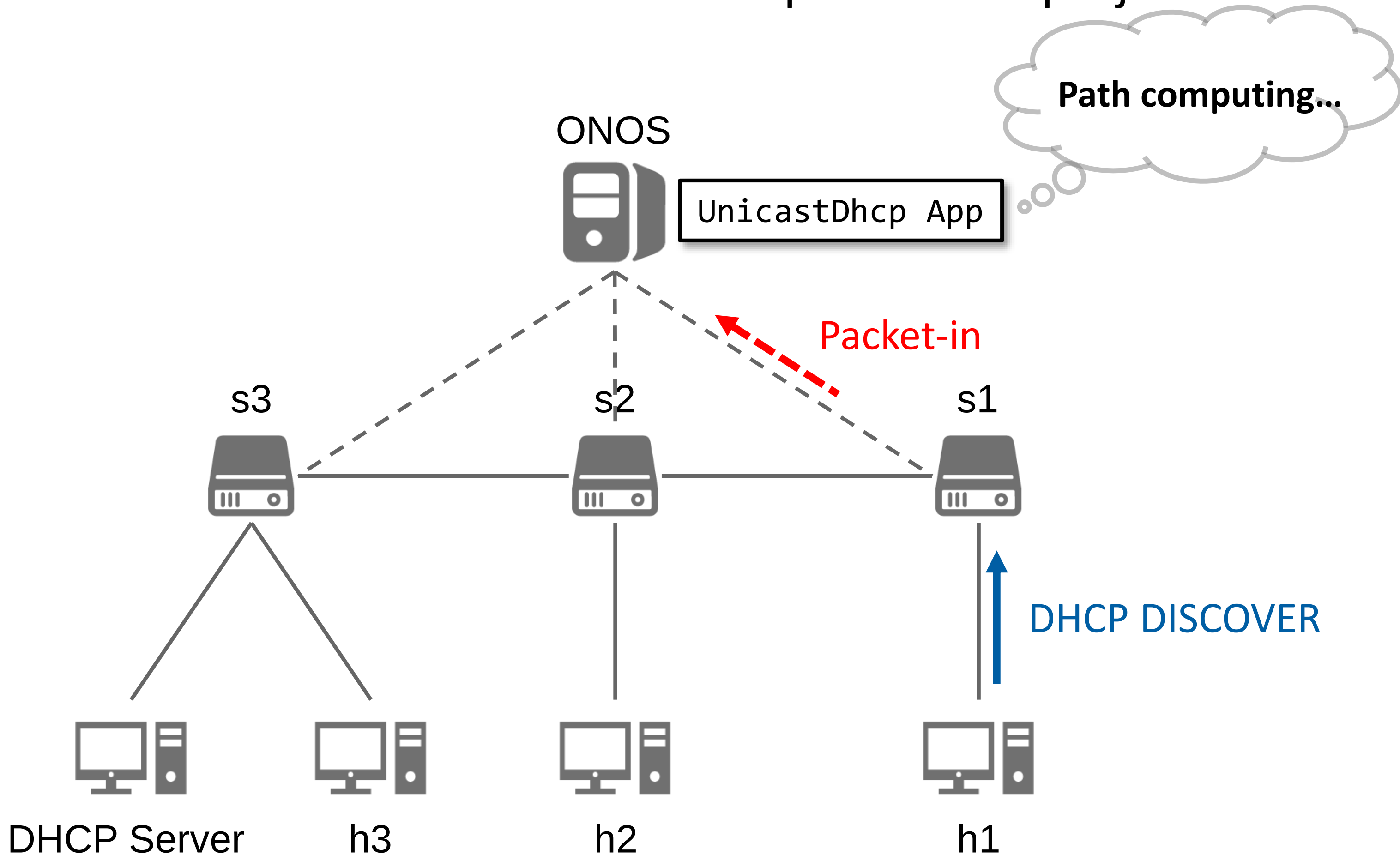
Step 2/4 – Packet-in DHCP Request

- Request switches to packet-in DHCP packets
  - Ask ONOS to delivery the packet-in DHCP packets



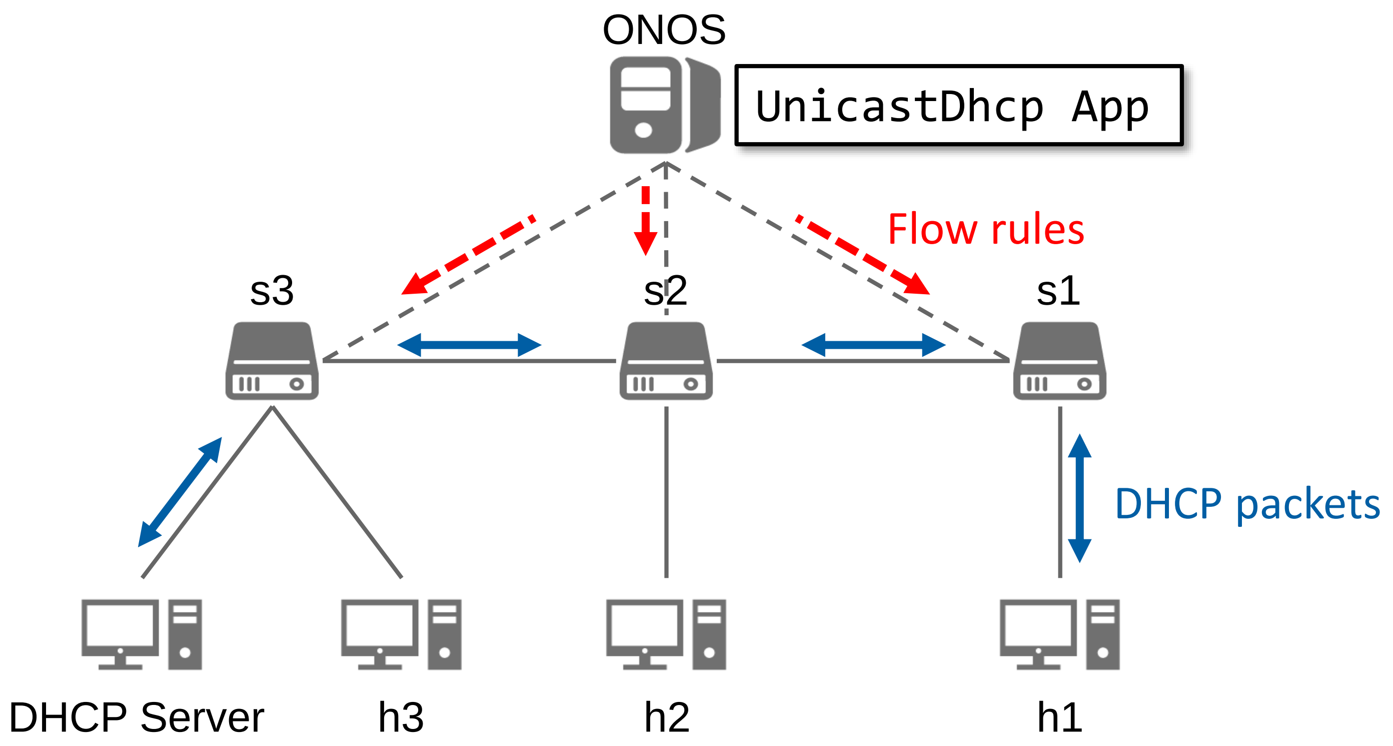
Step 3/4 – Compute Path

- Make switches packet-in DHCP packets
- Compute the path between requester and DHCP server
  - You **can** use ONOS PathService to find the path in this project



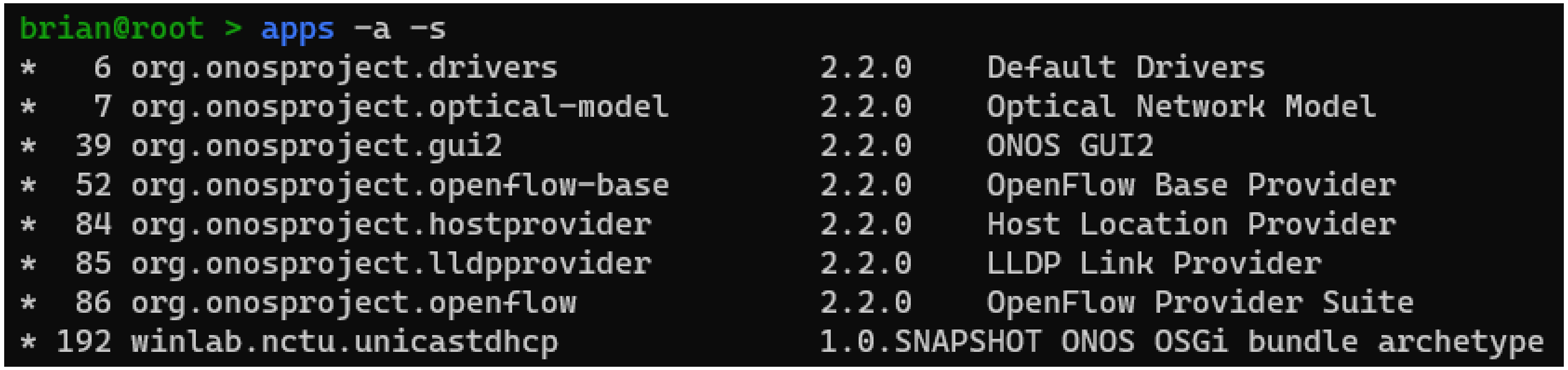
Step 4/4 – Install DHCP Packet Forwarding Rules

- Install flow rules to forward DHCP packets between client and server
- Subsequent DHCP packets (DISCOVER, OFFER, REQUEST, ACK) **should** all become unicast
  - Interfaces not on the path **should not** receive these packets



Suggestion

- In this project, it is not required to use ping to check connectivity.
  - For simplicity, you should deactivate **fwd** application
  - We will deactivate fwd when testing your App



```
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 winlab.nctu.unicastdhcp                  1.0.SNAPSHOT ONOS OSGi bundle archetype
```

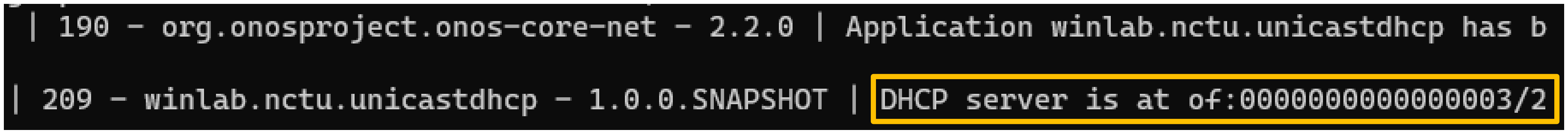
Naming Requirement

- You should follow the Maven project naming format below, or your project will not be scored

`<groupId>:`	`nctu.winlab`
`<artifactId>:`	`unicastdhcp`
`<version>:`	`<use default> (1.0-SNAPSHOT)`
`<Package>:`	`nctu.winlab.unicastdhcp`

Project 6 Requirements and Scoring Criteria

- **(10%)** Project naming convention
- **(30%)** Print DHCP location in ONOS log after uploading config file



```
 | 190 - org.onosproject.onos-core-net - 2.2.0 | Application winlab.nctu.unicastdhcp has b

| 209 - winlab.nctu.unicastdhcp - 1.0.0.SNAPSHOT | DHCP server is at of:0000000000000003/2
```

- **(30%)** Host(s) can get IP address after using `dhclient`
- **(30%)** DHCP transaction packets should be forwarded by **unicast**

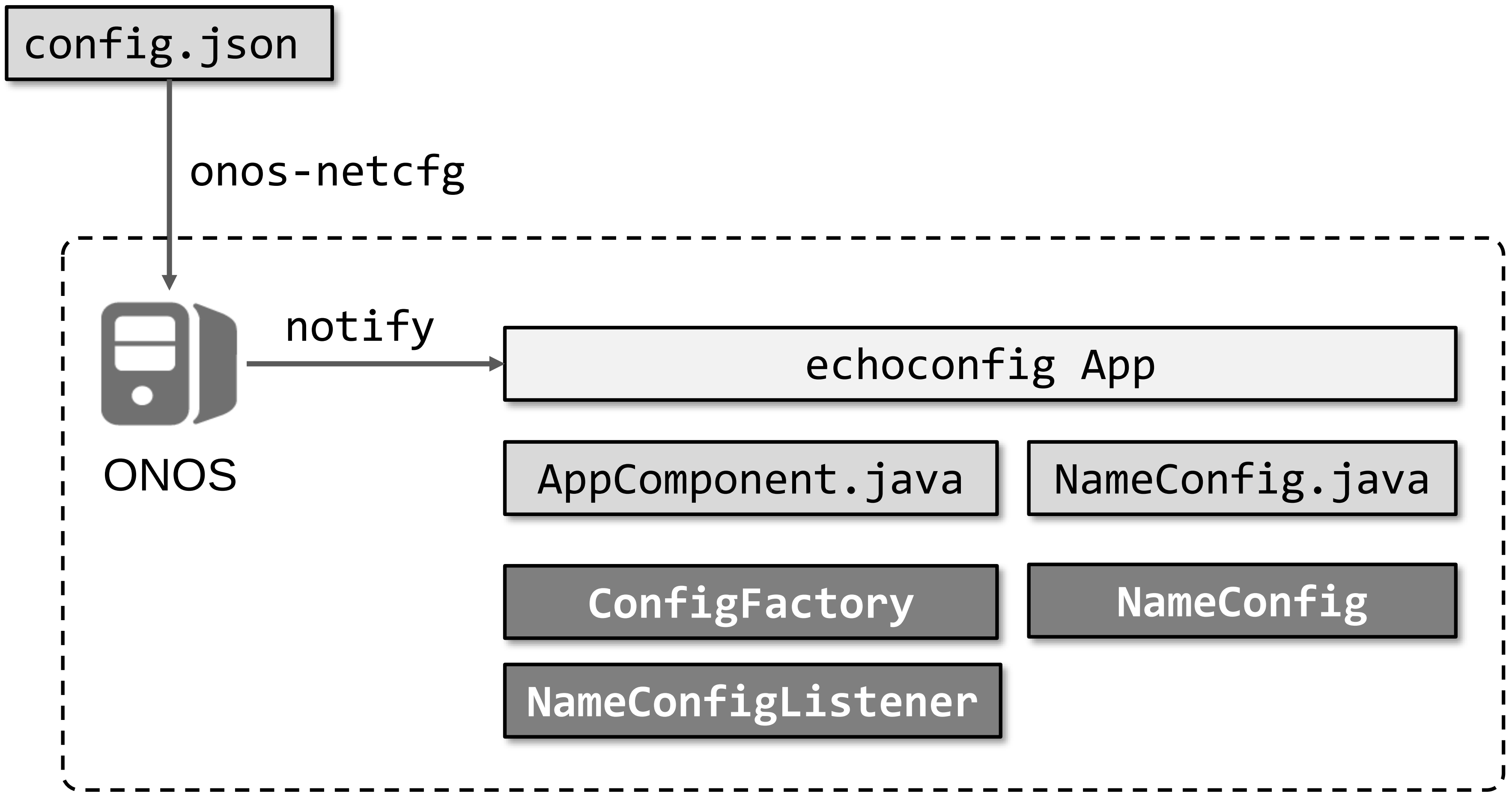
Outline

- ❑ Introduction to DHCP
- ❑ Project 6 Overall Procedure
- ❑ Environment Setup & Sample Application
- ❑ Submission

Provided File

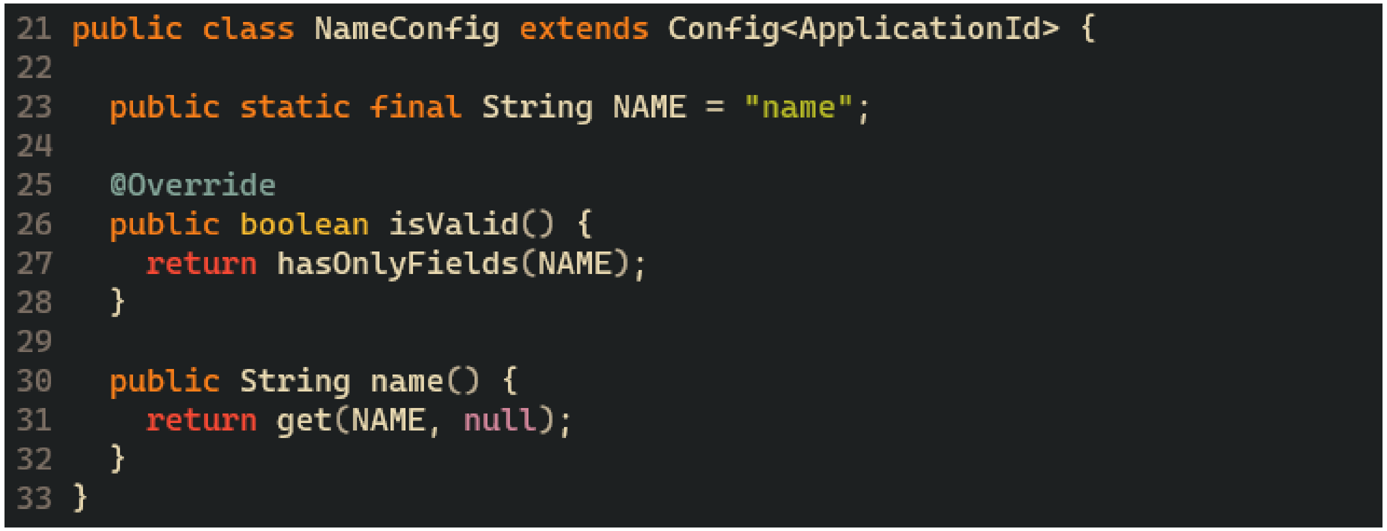
- "project6-supplement.zip" includes the following files:
  1. A sample application **echoconfig and its configuration files**
     - **AppComponent.java**: the sample application (**echoconfig**) that
       - Receives configuration value(s) from NameConfig.java
       - Prints value(s) of configuration file
     - **NameConfig.java**
       - Holds the key-value data extracted from config.json
     - **config.json**: configuration file for **echoconfig**
  2. Configuration file for Unicast DHCP app
     - **unicastdhcp.json**: configuration file for unicast DHCP app
  3. **Topology environment files**
     - **topo.py**: mininet topology
     - **dhcpd.conf**: DHCP configuration file used by topo.py

echoconfig Architecture

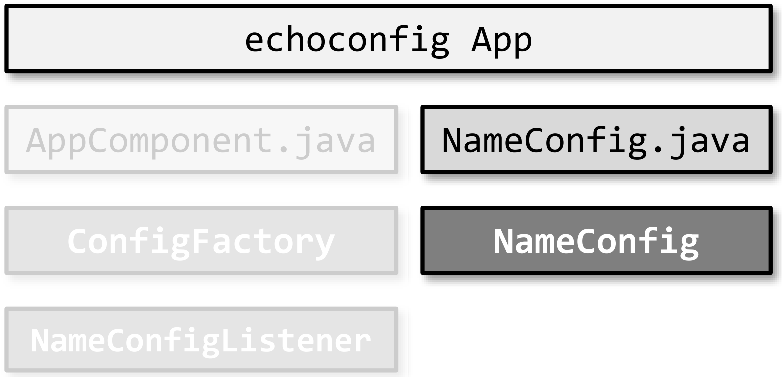


Component 1/3 – NameConfig

- A customized configuration for the echoconfig application
  - ONOS will use `isValid()` to check your uploaded JSON file
  - You can get the configuration value from an instance of this class

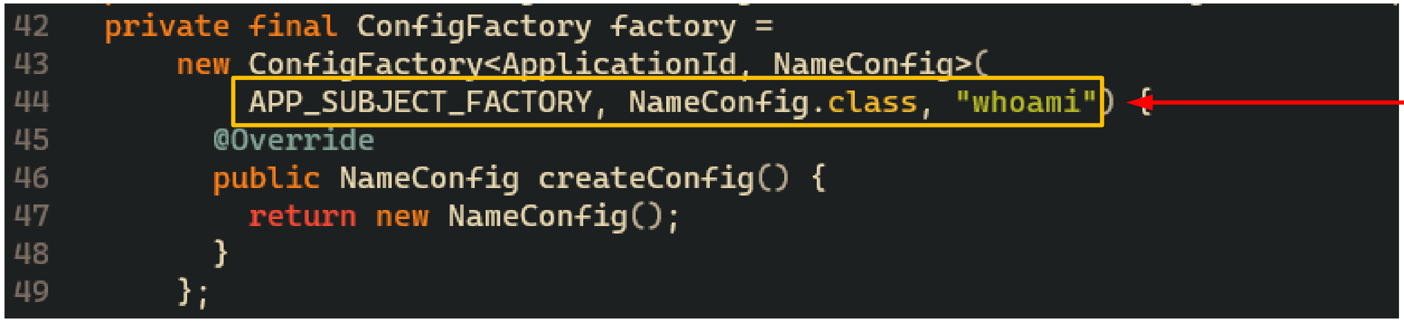


```java
public class NameConfig extends Config<ApplicationId> {

  public static final String NAME = "name";

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

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

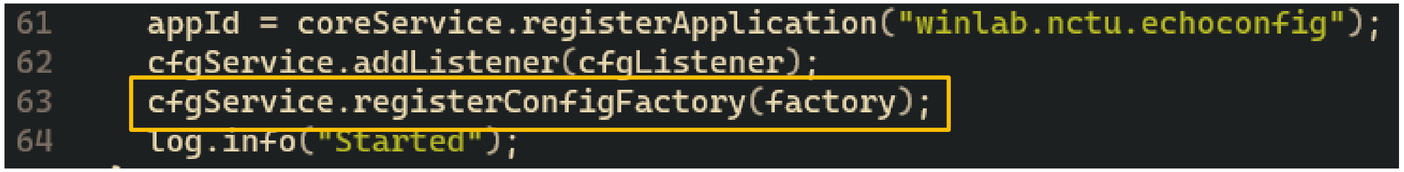


Component 2/3 – ConfigFactory

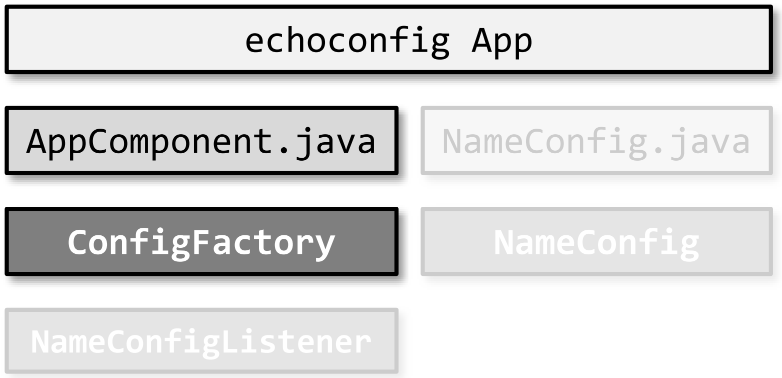
- Tell ONOS how to create a **NameConfig** instance
  - The arguments serve as key for ONOS to fetch the correct factory
  - ONOS will call createConfig()



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

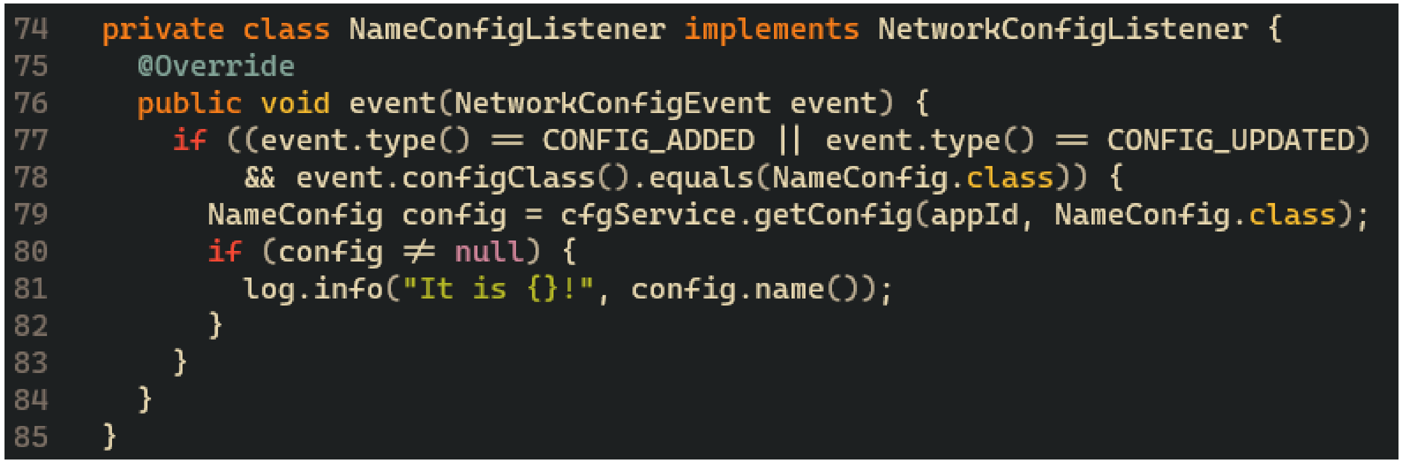


```java
        appId = coreService.registerApplication("winlab.nctu.echoconfig");
        cfgService.addListener(cfgListener);
        cfgService.registerConfigFactory(factory);
        log.info("Started");
```

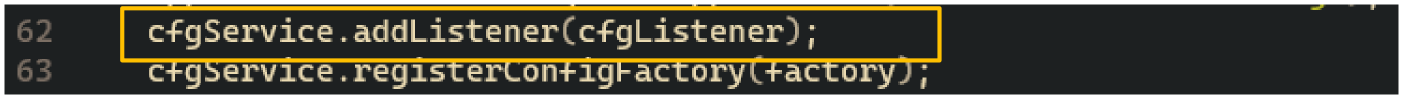


Component 3/3 – NameConfigListener

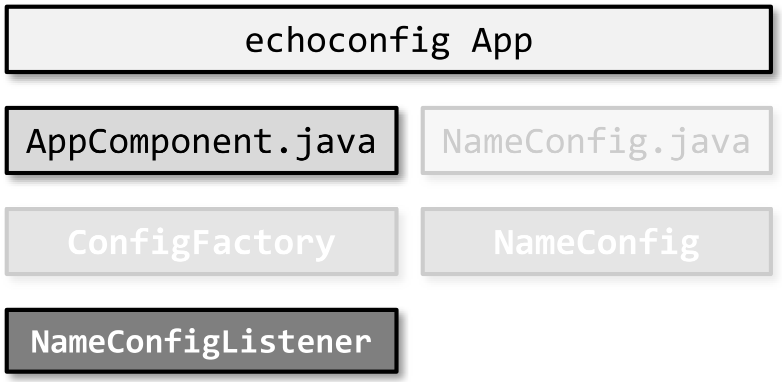
- Listen to network configuration event (e.g. A config file is uploaded)
- ONOS will call `event()` when event happens



```java
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, NameConfig.class);
   if (config != null) {
    log.info("It is {}!", config.name());
   }
  }
 }
}
```

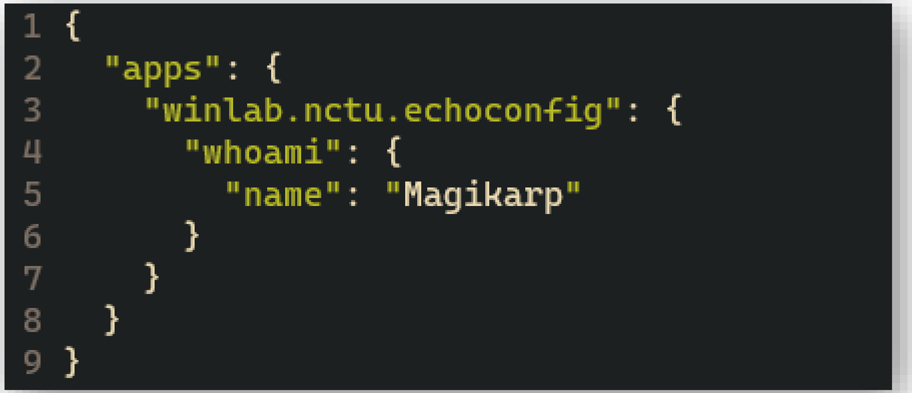


```java
cfgService.addListener(cfgListener);
cfgService.registerConfigFactory(factory);
```



echoconfig Demonstration

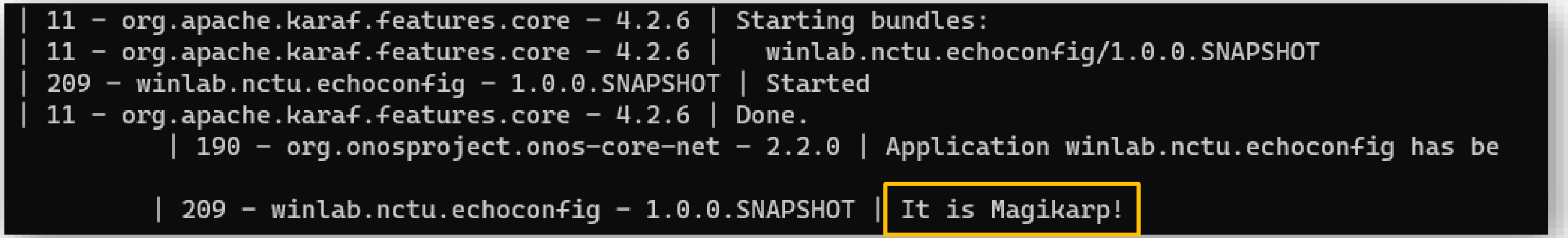
- Upload config.json



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

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

- ONOS log will show following message



```
| 11 - org.apache.karaf.features.core - 4.2.6 | Starting bundles:
| 11 - org.apache.karaf.features.core - 4.2.6 |   winlab.nctu.echoconfig/1.0.0.SNAPSHOT
| 209 - winlab.nctu.echoconfig - 1.0.0.SNAPSHOT | Started
| 11 - org.apache.karaf.features.core - 4.2.6 | Done.
          | 190 - org.onosproject.onos-core-net - 2.2.0 | Application winlab.nctu.echoconfig has be

         | 209 - winlab.nctu.echoconfig - 1.0.0.SNAPSHOT | It is Magikarp!
```

Provided File

- "project6-supplement.zip" includes following files:
  1. A sample application **echoconfig and its configuration files**
     - **AppComponent.java**: the sample application (**echoconfig**) that
       - Receives configuration value(s) from NameConfig.java
       - Prints value(s) of configuration file
     - **NameConfig.java**
       - Holds the key-value data extracted from config.json
     - **config.json**: configuration file for **echoconfig**
  2. Configuration file for Unicast DHCP app
     - **unicastdhcp.json**: configuration file for unicast DHCP app
  3. **Topology environment files**
     - **topo.py**: mininet topology
     - **dhcpd.conf**: DHCP configuration file used by topo.py

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
```

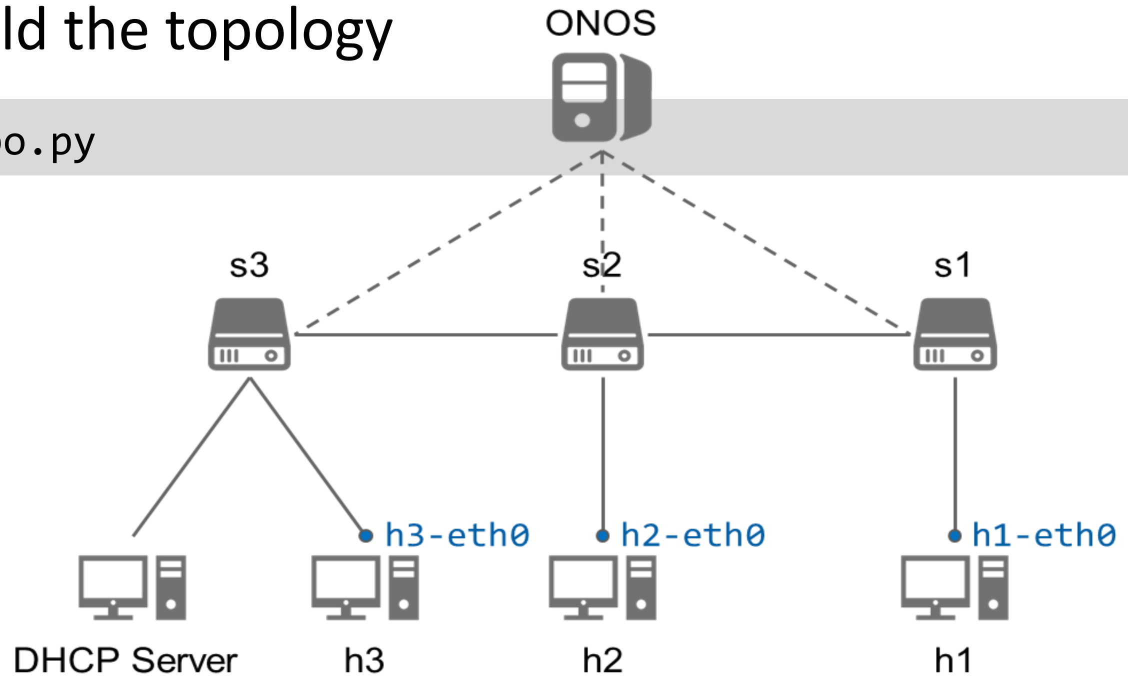
  - For client

```
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
```

How to Test Your App

- Use **topo.py** to build the topology

```
bash$ sudo python topo.py
```



- 3 hosts without IP addresses in the provided topology

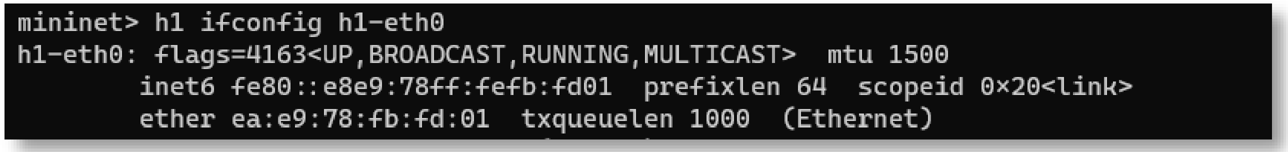
```
mininet> h1 dhclient -v h1-eth0
```

✓ Note: Release current lease before re-issue DHCP request on an interface (to observe all packets of a DHCP transaction)

```
mininet> h1 dhclient -r h1-eth0
```

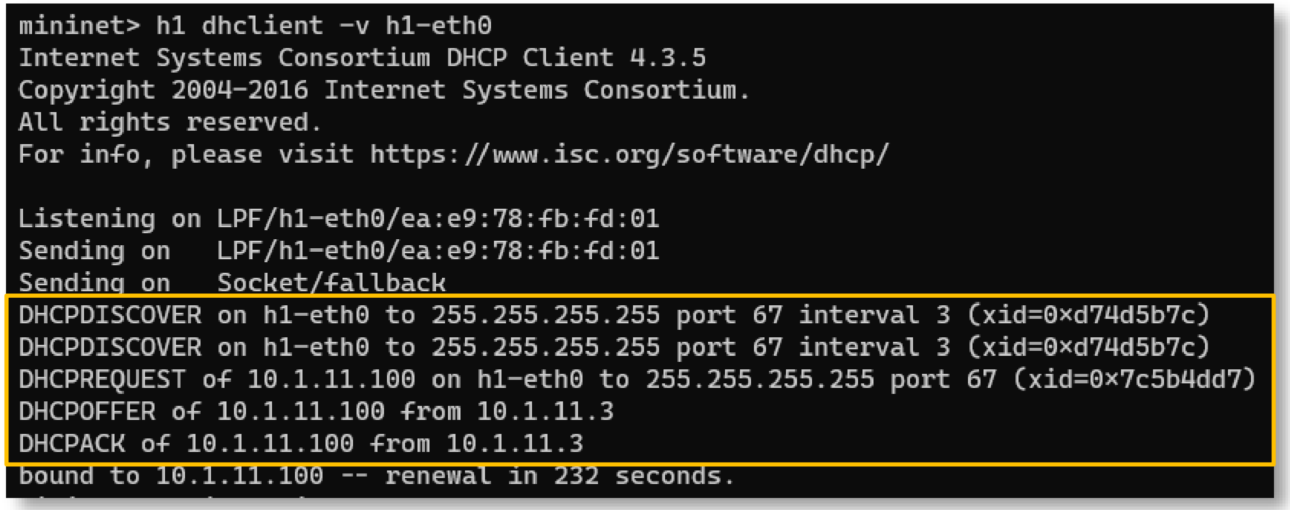
Demonstration

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



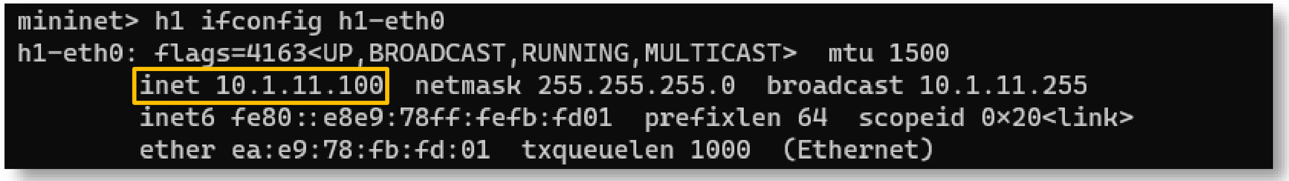
```
mininet> h1 ifconfig h1-eth0
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::e8e9:78ff:fefb:fd01  prefixlen 64  scopeid 0x20<link>
        ether ea:e9:78:fb:fd:01  txqueuelen 1000  (Ethernet)
```

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.
```

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
- ❑ Project 6 Overall Procedure
- ❑ Environment Setup & Sample Application
- ❑ Submission

Submit to e3

- Files
  - All files of your application
- Submission
  - Upload ".zip" file to e3
    - Named: **project6_<studentID>.zip**
  - Incorrect naming convention or format will not be scored

Reference

- ONOS Java API (2.2.0)
- The Network Configuration Service