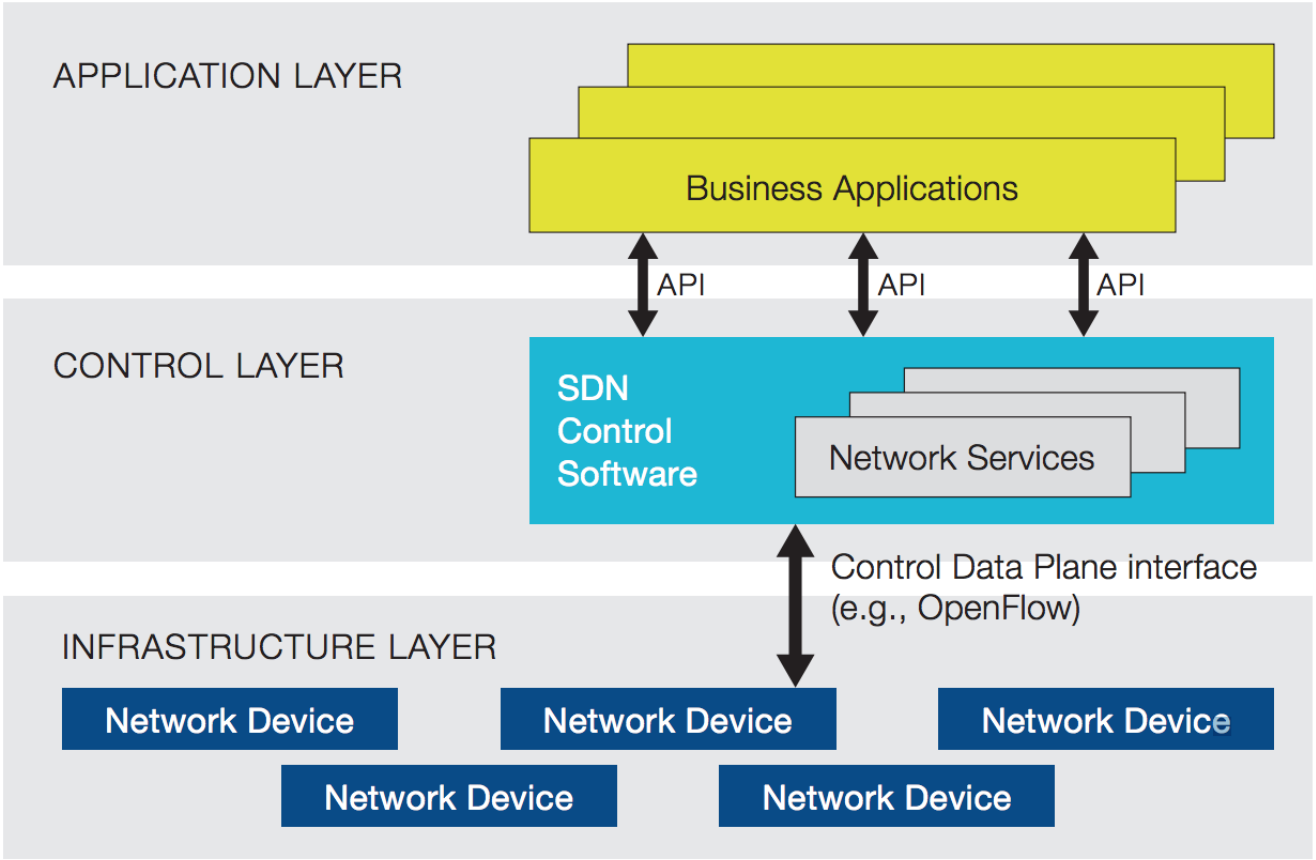
A Software Stack for SDN Programming Automation

林潇 (Shawn Lin)
Systems Networking Lab (SNLab)
Tongji-Yale Network Center

Dec 12, 2016

What is SDN?

- Software Defined Networking (SDN) is an emerging network architecture where network control is decoupled from forwarding and is directly programmable.



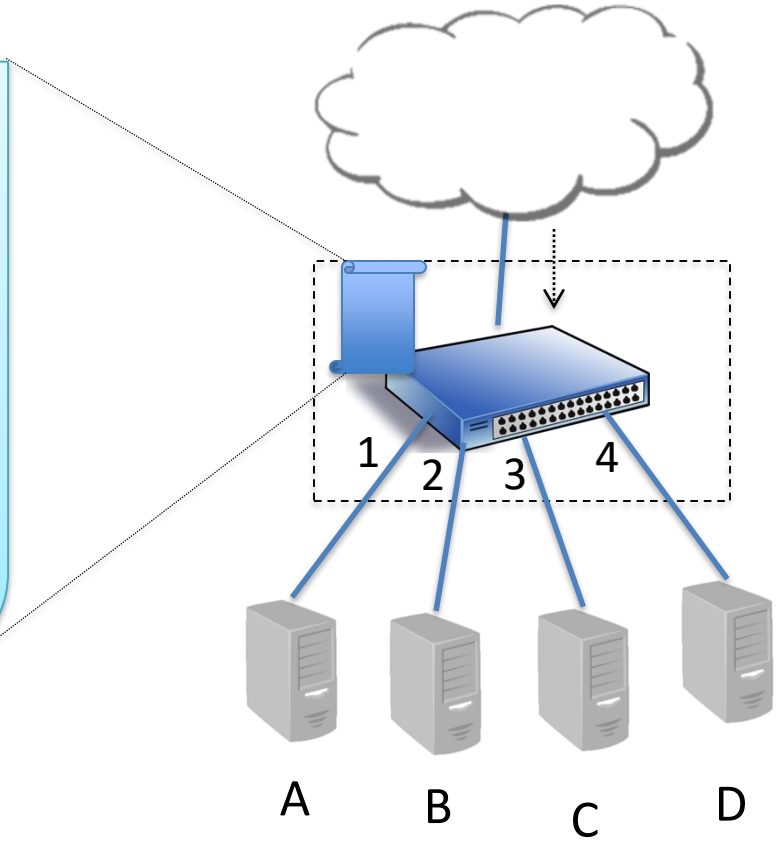
Source: *SDN White Paper*

What does SDN bring to us?

Is it easy enough to implement this policy?

Policy: block traffic to port 22

```
prohibitedPort = 22        // Policy
hostTbl = {A:1,B:2,C:3,D:4}

def onPacketIn(p):
    if prohibitedPort == p.tcp_dst:
        drop
    else:
        forward([hostTbl(p.eth_dst)])
```



Implement it with Flow Rule Manager (SalFlowService)

```
hostTbl = {A:1,B:2,C:3,D:4}

def onPacketIn(p):
 if 22 == p.tcp_dst:
   # drop

   installRule({‘match':{'tcp_dst':22},
              ‘action’:[]})

 else:
   # forward([hostTbl(p.eth_dst)])

   installRule({‘match’: {‘eth_dst’:p.eth_dst,
                      ’tcp_dst’!=22},
              ‘action’:[hostTbl(p.eth_dst)]})
```

match does not support logic negation

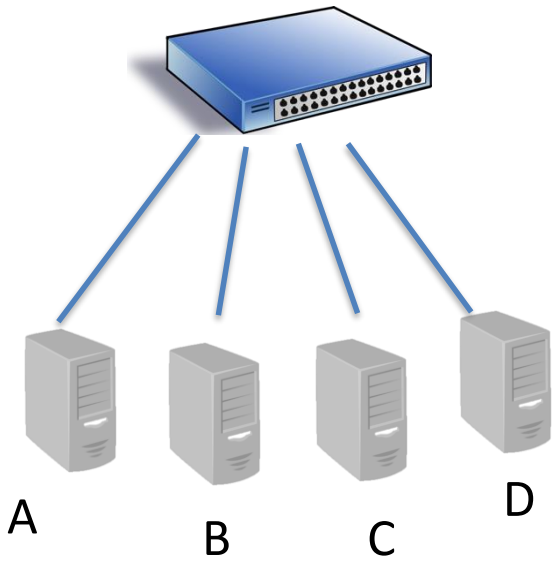
What can we do for logical negation?

```
hostTbl = {A:1,B:2,C:3,D:4}

def onPacketIn(p):
 if 22 == p.tcp_dst:
   # drop

   installRule({'priority':1,
              'match':{'tcp_dst':22},
              'action':[]})

 else:
   # forward([hostTbl(p.eth_dst)])

   installRule({'priority':0,
              'match': {'eth_dst':p.eth_dst},
              'action':[hostTbl(p.eth_dst)]})
```

Here comes the problem

EthDst:A,
TcpDst:80

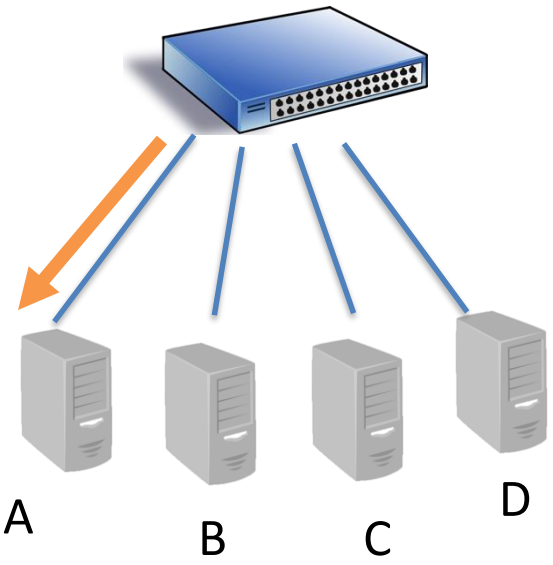
Match	Action	Priority



Here comes the problem

EthDst:A,
TcpDst:80

Match	Action	Priority
EthDst == A	output = port 1	0

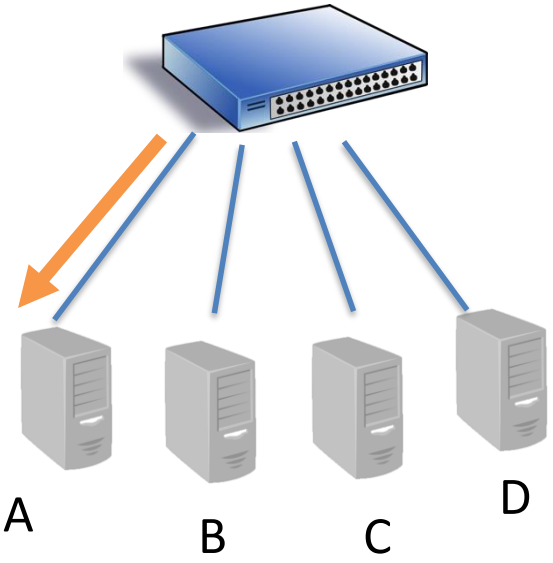


Here comes the problem

EthDst:A,
TcpDst:80

EthDst:A,
TcpDst:22

Match	Action	Priority
EthDst == A	output = port 1	0

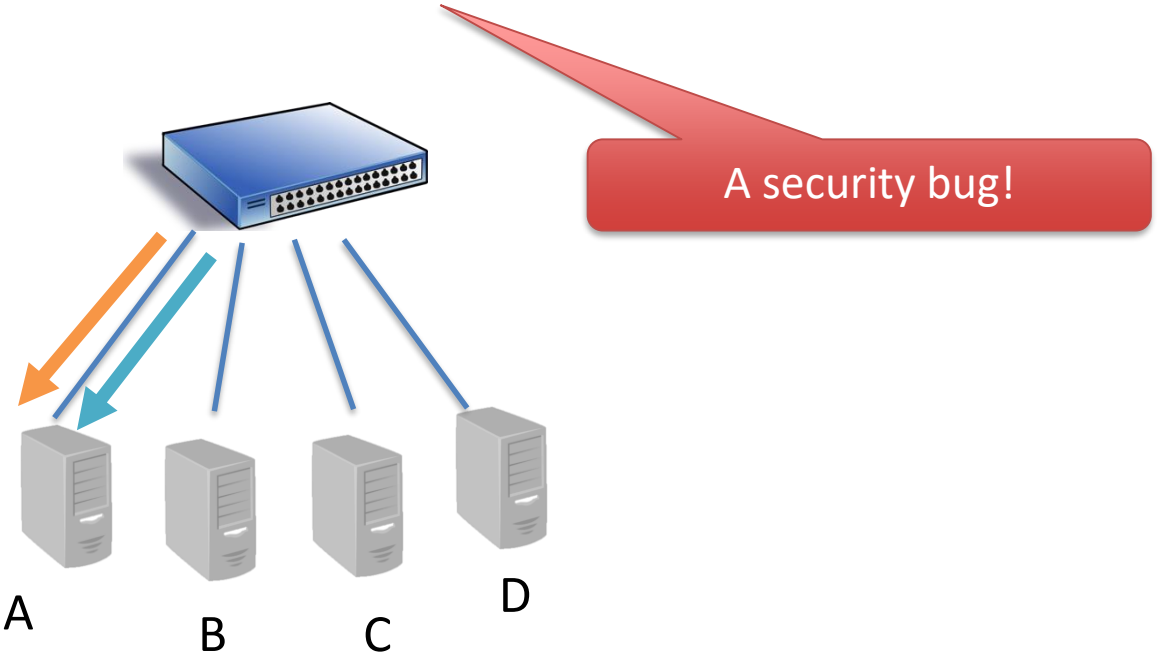


Here comes the problem

EthDst:A,
TcpDst:80

EthDst:A,
TcpDst:22

Match	Action	Priority
EthDst == A	output = port 1	0



Issues in Current SDN Programming

Issue 1: SDN programmers have to explicitly manage low-level forwarding rules, but the programming model of low-level forwarding rules is limited.

For example:

1. Flow rules do not support logic negation

2. OpenFlow supports only layer 2 to layer 4, but many decisions depend on higher layers

Easy to make a mistake when handling data changes!

```
public void findPath(Ipv4Address src, Ipv4Address dst,
                     int requiredBw) {
        srcNode, dstNode = mapToNode(src, dst);
        topology = readTopology();
        bw_src(x) = (x == srcNode ? +inf : 0);

        pq.add(srcNode);
        while (!pq.isEmpty()) {
            n = pq.pop();
            if (bw_src(n) < requiredBw)
                return failure
            if (n == dstNode)
                return setupFlowRules(path);
            for (e(n->m): n.neighbors) {
                if (min(e.bw, bw_src(n)) > bw_src(m)) {
                    updateBw(m, min(e.bw, bw_src(n)));
                    ...
                }
            }
        }
        return failure
}
```

```
...
            broker.registerDataChangeListener(..., parameters, listener, ...);
            broker.registerDataChangeListener(..., nodeMap, listener, ...);
            broker.registerDataChangeListener(..., topology, listener, ...);
...

public void onDataChanged(event) {
            findPath(currentSrc, currentDst, currentRequiredBw);
            // with current IP->AP mapping, current topology
    // and current link bandwidth
}
```

Easy to make a mistake when handling data changes!

```
public void findPath(Ipv4Address src, Ipv4Address dst,
                     int requiredBw) {
        srcNode, dstNode = mapToNode(src, dst);
        topology = readTopology();
        bw_src(x) = (x == srcNode ? +inf : 0);

        pq.add(srcNode);
        while (!pq.isEmpty()) {
            n = pq.pop();
            if (bw_src(n) < requiredBw)
                return failure
            if (n == dstNode)
                return setupFlowRules(path);
            for (e(n->m): n.neighbors) {
                if (min(e.bw, bw_src(n)) > bw_src(m)) {
                    updateBw(m, min(e.bw, bw_src(n)));
                    ...
                }
            }
        }
        return failure
}
```

```
...
        broker.registerDataChangeListener(..., parameters, listener, ...);
        broker.registerDataChangeListener(..., nodeMap, listener, ...);
        broker.registerDataChangeListener(..., topology, listener, ...);
...

public void onDataChanged(event) {
        removeFlowRulesIfExist(odlpath);
        findPath(currentSrc, currentDst, currentRequiredBw);
}
```

Issues in Current SDN Programming

Issue 2: SDN programmers manually setup listeners for data changes and handle data change events

```
InstanceIdentifier<HostNode> hostNodes = InstanceIdentifier.builder(NetworkTopology.class)//
        .child(Topology.class, new TopologyKey(new TopologyId(topologyId)))//
        .child(Node.class)
        .augmentation(HostNode.class).build();
this.hostNodeListerRegistration = dataService.registerDataChangeListener(LogicalDatastoreType.OPERATIONAL, hostNodes,

@Override
public void onDataChanged(final AsyncDataChangeEvent<InstanceIdentifier<?>, DataObject> change) {

   for (InstanceIdentifier<?> iid : deletedData) {
       if (iid.getTargetType().equals(Node.class)) {
           Node node = ((Node) originalData.get(iid));
           InstanceIdentifier<Node> iiN = (InstanceIdentifier<Node>) iid;
           HostNode hostNode = node.getAugmentation(HostNode.class);
           if (hostNode != null) {
               synchronized (hosts) {
                   try {
                       hosts.removeLocally(iiN);
                   } catch (ClassCastException ex) {
                       LOG.debug("Exception occurred while remove host locally", ex);
                   }
               }
           }
```

A bug in a basic service

This code is from the l2switch project in ODL

Maven and Karaf bring convenience as well as complexity

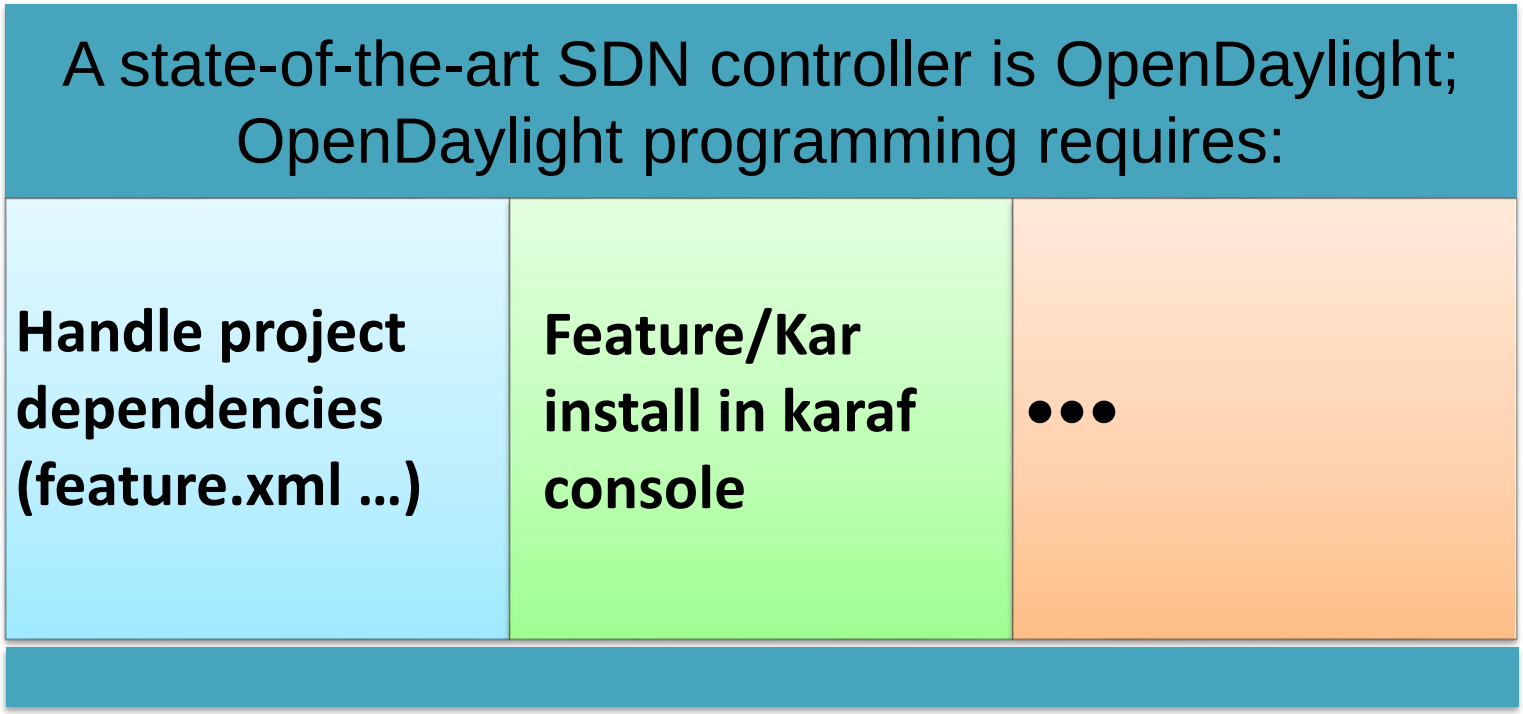
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://w
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.opendaylight.l2switch</groupId>
    <artifactId>l2switch-parent</artifactId>
    <version>0.3.3-Beryllium-SR3</version>
    <relativePath>../../parent</relativePath>
  </parent>
  <groupId>org.opendaylight.l2switch.main</groupId>
  <artifactId>main-impl</artifactId>
  <packaging>bundle</packaging>

  <dependencies>
    <dependency>
      <groupId>org.opendaylight.controller</groupId>
      <artifactId>sal-binding-api</artifactId>
    </dependency>
    <dependency>
      <groupId>org.opendaylight.openflowplugin.model</groupId>
      <artifactId>model-flow-service</artifactId>
    </dependency>
    <dependency>
      <groupId>org.opendaylight.controller.model</groupId>
      <artifactId>model-topology</artifactId>
    </dependency>
    <dependency>
      <groupId>org.opendaylight.yangtools</groupId>
      <artifactId>yang-common</artifactId>
    </dependency>
    <dependency>
      <groupId>org.opendaylight.l2switch.packethandler</groupId>
      <artifactId>packethandler-model</artifactId>
    </dependency>
    <dependency>
      <groupId>org.opendaylight.l2switch.packethandler</groupId>
      <artifactId>packethandler-impl</artifactId>
    </dependency>
    <dependency>
      <groupId>org.opendaylight.l2switch.addresstracker</groupId>
      <artifactId>addresstracker-model</artifactId>
    </dependency>
    <dependency>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<features name="l2switch-${project.version}" xmlns="http://karaf.apache.org/xmlns/features/v1.2.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://karaf.apache.org/xmlns/features/v1.2.0 http://karaf.apache.org/xmlns/features/v1.2
  <repository>mvn:org.opendaylight.openflowplugin/features-openflowplugin/${openflow.plugin.version}/xml/features</repo
  <repository>mvn:org.opendaylight.netconf/features-restconf/${restconf.version}/xml/features</repository>
  <repository>mvn:org.opendaylight.dlux/features-dlux/${dlux.version}/xml/features</repository>
  <feature name='odl-l2switch-all' description="OpenDaylight :: L2Switch :: All" version='${project.version}'>
    <feature version="${project.version}">odl-l2switch-switch</feature>
  </feature>
  <feature name='odl-l2switch-switch' description="OpenDaylight :: L2Switch :: Switch" version='${project.version}'>
    <feature version="${project.version}">odl-l2switch-hosttracker</feature>
    <feature version="${project.version}">odl-l2switch-arphandler</feature>
    <bundle>mvn:org.opendaylight.l2switch.main/main-impl/{{VERSION}}</bundle>
    <configfile finalname="${config.configfile.directory}/${config.l2switchmain.configfile}">mvn:org.opendaylight.l2swi
  </feature>
  <feature name='odl-l2switch-switch-rest' description="OpenDaylight :: L2Switch :: Switch" version='${project.version}
    <feature version="${project.version}">odl-l2switch-switch</feature>
    <feature version="${restconf.version}">odl-restconf</feature>
  </feature>
  <feature name='odl-l2switch-switch-ui' description="OpenDaylight :: L2Switch :: Switch" version='${project.version}'>
    <feature version="${project.version}">odl-l2switch-switch-rest</feature>
    <feature version="${restconf.version}">odl-mdsal-apidocs</feature>
    <feature version="${mdsal.version}">odl-mdsal-xsql</feature>
    <feature version="${dlux.version}">odl-dlux-core</feature>
  </feature>
  <feature name='odl-l2switch-hosttracker' description="OpenDaylight :: L2Switch :: HostTracker" version='${project.ver
    <feature version="${project.version}">odl-l2switch-addresstracker</feature>
    <bundle>mvn:org.opendaylight.l2switch.hosttracker/hosttracker-model/{{VERSION}}</bundle>
    <bundle>mvn:org.opendaylight.l2switch.hosttracker/hosttracker-impl/{{VERSION}}</bundle>
    <configfile finalname="${config.configfile.directory}/${config.hosttracker.configfile}">mvn:org.opendaylight.l2swit
  </feature>
  <feature name='odl-l2switch-addresstracker' description="OpenDaylight :: L2Switch :: AddressTracker" version='${proje
    <feature version="${project.version}">odl-l2switch-packethandler</feature>
    <bundle>mvn:org.opendaylight.l2switch.addresstracker/addresstracker-model/{{VERSION}}</bundle>
    <bundle>mvn:org.opendaylight.l2switch.addresstracker/addresstracker-impl/{{VERSION}}</bundle>
    <configfile finalname="${config.configfile.directory}/${config.addresstracker.configfile}">mvn:org.opendaylight.l2s
  </feature>
  <feature name='odl-l2switch-arphandler' description="OpenDaylight :: L2Switch :: ArpHandler" version='${project.versi
    <feature version="${project.version}">odl-l2switch-packethandler</feature>
    <feature version="${project.version}">odl-l2switch-loopremover</feature>
    <bundle>mvn:org.opendaylight.l2switch.addresstracker/addresstracker-model/{{VERSION}}</bundle>
    <bundle>mvn:org.opendaylight.l2switch.arphandler/arphandler-impl/{{VERSION}}</bundle>
    <configfile finalname="${config.configfile.directory}/${config.arphandler.configfile}">mvn:org.opendaylight.l2switc
  </feature>
  <feature name='odl-l2switch-loopremover' description="OpenDaylight :: L2Switch :: LoopRemover" version='${project.ver
```

```
/Users/shawn/Application/distribution-karaf-0.4.3-Beryllium-SR3/etc  ls                                                    23:43:51
all.policy                                jmx.acl.org.apache.karaf.security.jmx.cfg  org.apache.karaf.command.acl.scope_bundle.cfg  org.opendaylight.aaa.authn.cfg
config.properties                         jmx.acl.osgi.compendium.cm.cfg             org.apache.karaf.command.acl.shell.cfg         org.opendaylight.aaa.federation.cfg
custom.properties                         jre.properties                             org.apache.karaf.command.acl.system.cfg        org.opendaylight.aaa.tokens.cfg
distribution.info                         keys.properties                            org.apache.karaf.features.cfg                  org.opendaylight.controller.cluster.datastore.cfg
equinox-debug.properties                  odl.java.security                          org.apache.karaf.features.obr.cfg              org.ops4j.pax.logging.cfg
idmtool                                   opendaylight                               org.apache.karaf.features.repos.cfg            org.ops4j.pax.url.mvn.cfg
java.util.logging.properties              org.apache.felix.fileinstall-deploy.cfg    org.apache.karaf.jaas.cfg                      regions-config.xml
jetty.xml                                 org.apache.karaf.command.acl.bundle.cfg    org.apache.karaf.kar.cfg                       shell.init.script
jmx.acl.cfg                               org.apache.karaf.command.acl.config.cfg    org.apache.karaf.log.cfg                       shiro.ini
jmx.acl.java.lang.Memory.cfg              org.apache.karaf.command.acl.feature.cfg   org.apache.karaf.management.cfg                startup.properties
jmx.acl.org.apache.karaf.bundle.cfg       org.apache.karaf.command.acl.jaas.cfg      org.apache.karaf.shell.cfg                     system.properties
jmx.acl.org.apache.karaf.config.cfg       org.apache.karaf.command.acl.kar.cfg       org.apache.karaf.webconsole.cfg                users.properties
```

Issues in Current SDN Programming

Issue 3: Complex, manual project deployment



What is Our Solution?

Low level, limited programming model

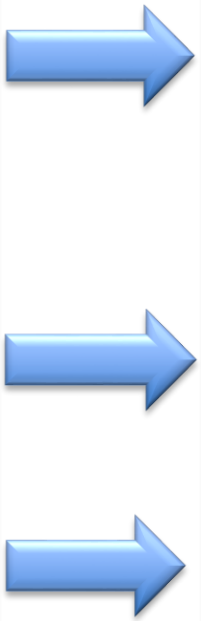
- Low-level, complex, limited (L2-L4) OpenFlow rule programming
- Programmer can define only at flow level
- Specific access control allowing only hosts partition

Raw data store

- Complex, manual tracking of execution dependency
- Manual cleanup, re-execute
- Designed directly on raw data store

Manual programming

- Complex, manual maven programming



Maple

- High-level, completely south-bound agnostic, cross-layer (**L2-L7**) programming
- Programmer sees (logically) each and every packet
- Integrated access control supporting per-user or role based programming

FAST

- Automatic execution dependency tracking
- Automatic cleanup, re-execution (intent ++)
- Host generic network functions

Web-based Integrated Dev Env (IDE)

- Web-based automatic generation of projects
- Programmer focuses only on policy decision

Our software stack for SDN programming

Developer
Operator

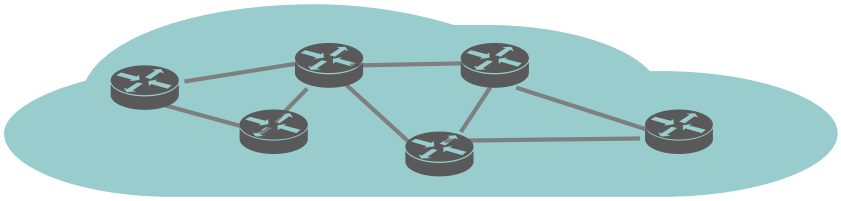
Web SDN IDE: A GUI-driven unified framework for developing/operating the whole network

Control
Plane

Maple: A programming framework enables users to write algorithmic policy to decide the behaviors of an entire network

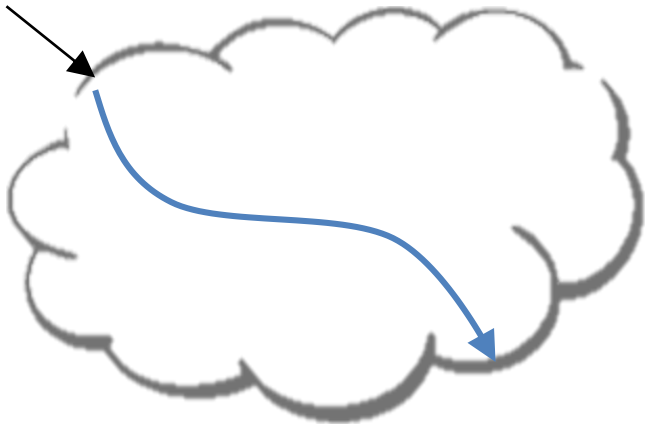
FAST: Track runtime data dependency and re-execute when data changes happen

Infrastructure



Maple: High-level SDN Programming Language

Consider each pkt
as a request



- Network as a single virtual server

- Network functions expressed in general purpose language

- Network functions (***logically***) invoked on each new pkt, returning how net handles that request

- Access pkt virtual attributes as well as physical attributes, so that network decision can depend on L2 to L7

Example

Control the traffic based on high-level science data info

```
1. private static final String[] H12_BRO = { H1, "openflow:1:2", "open
2. private static final String[] H12_SLOW = { H1, "openflow:1:2
3. private static final String[] H12_FAST =                    openflow:1:3", "openflow:3:2",  "openflow:4:1" }
4.void f(Packet pkt) {
5.     if ( pkt.ethType == Ethernet.TYPE_IPv4 ) {
6.         if ( pkt.srcIP == H1 && pkt.dstIP == H2 ) {
7.             if ( pkt.flow == null ) {
8.                 pkt.addRoute( H12_BRO )
9.                 pkt.addRoute( H12_SLOW )
10.            } else if ( pkt.flow.http && isScience(pkt.flow.http.mediaType) ) {
11.                pkt.addRoute( H12_FAST );
12.            } else {
13.                pkt.addRoute( H12_SLOW );
14.            }
15.         } else if ( pkt.srcIP == H2 && pkt.dstIP == H1 ) {
16.         …
17.     } else pkt.addRoute( Route.drop );
18. }
```

Per-packet programming model

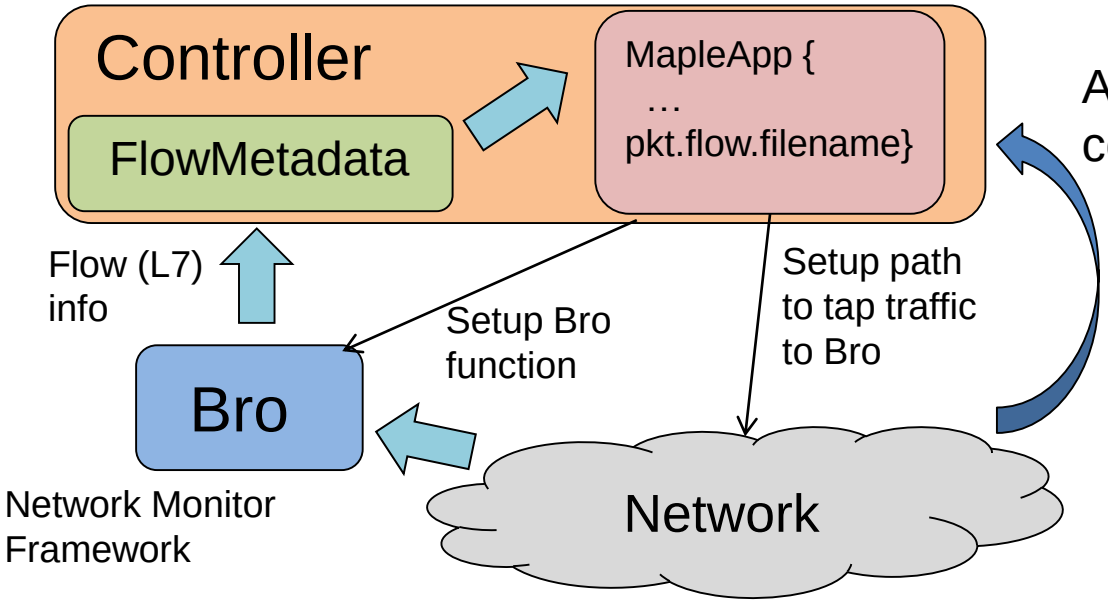
Unified packet attributes (virtual + physical)

Control packets with L7 info

South-band agnostic

Seamless L2-L4 to L7

If an MapleApp accesses L7 flow info (e.g., file name), system automatically sets up L7 inspection to extract info
Bro logical concept, transparent



How to change the path automatically when FlowMetadata updated?

Our software stack for SDN programming

Developer
Operator

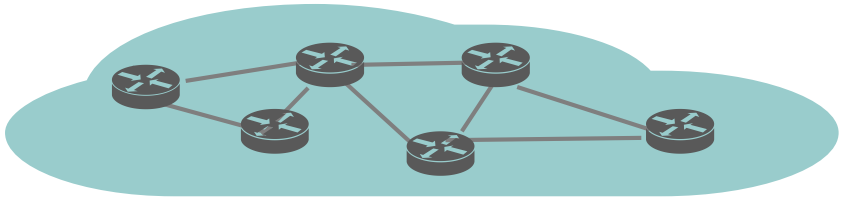
Web SDN IDE: A GUI-driven unified framework for developing/operating the whole network

Control
Plane

Maple: A programming framework enables users to write algorithmic policy to decide the behaviors of an entire network

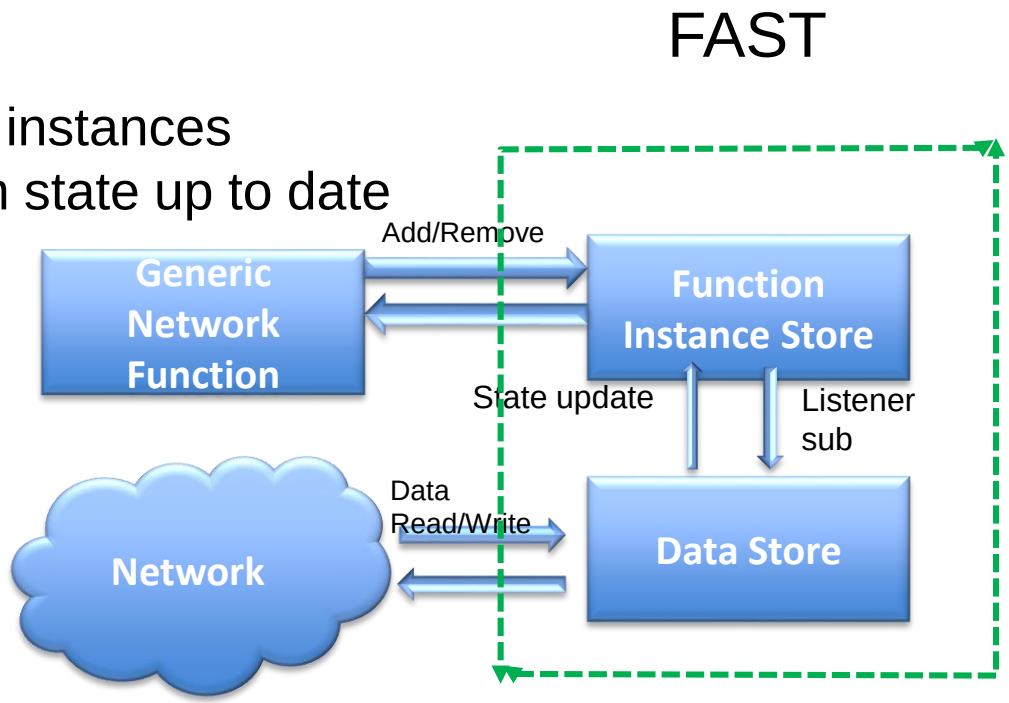
FAST: Track runtime data dependency and re-execute when data changes happen

Infrastructure



FAST: A Function Automation System

- Hosting generic data-driven network functions
  - Easy-to-use programming paradigm
- Automatic dependency management
  - Track fine-grained runtime dependency
  - Subscribe to data changes automatically
- Enforced data consistency
  - Rollback outdated function instances
  - Re-execute to keep system state up to date



Our software stack for SDN programming

Developer
Operator

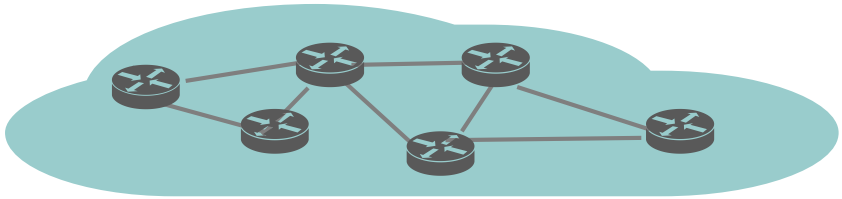
Web SDN IDE: A GUI-driven unified framework for developing/operating the whole network

Control
Plane

Maple: A programming framework enables users to write algorithmic policy to decide the behaviors of an entire network

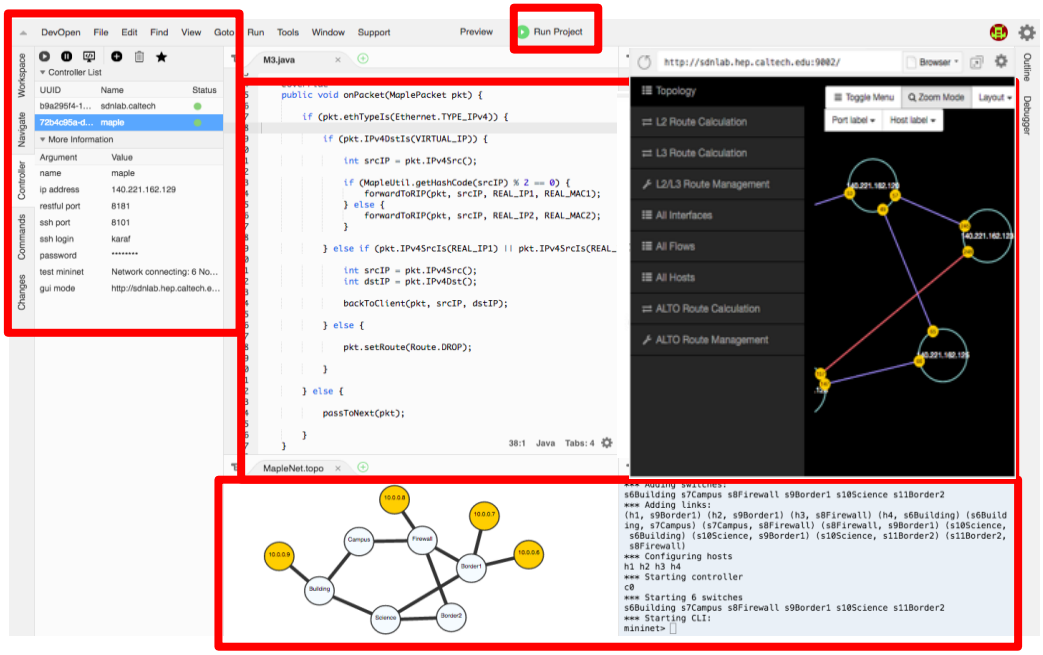
FAST: Track runtime data dependency and re-execute when data changes happen

Infrastructure



Web SDN IDE

- [illegible]
- [illegible]



Controller management

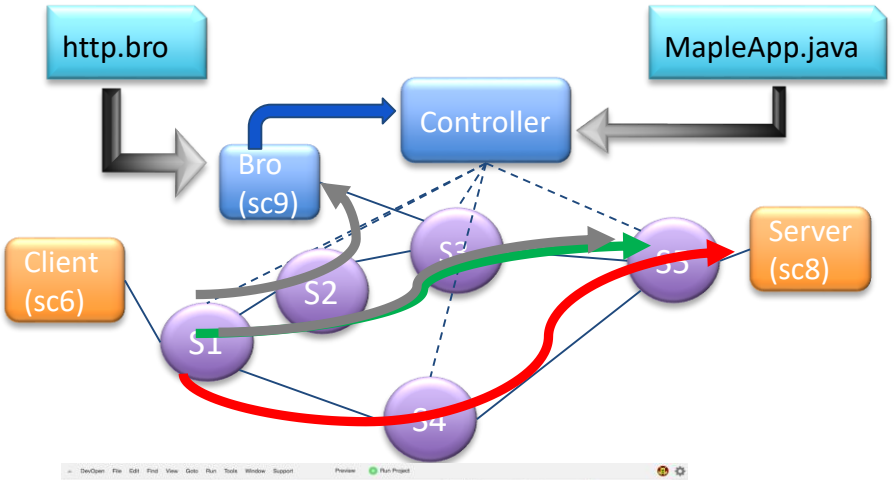
Network simulator

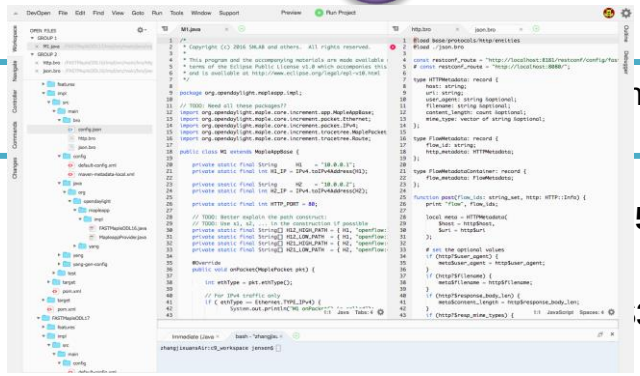
One-click deployment

Development and debugging

Topology and routes management

DEMO: Control Traffic based on L7 (HTTP Info)





Step of Demo:

1. User writes & deploys MapleApp program in Web IDE.

Bro is a network monitor framework, in which we can get HTTP info for flows.

2. Client sends a request for science data at the Server.

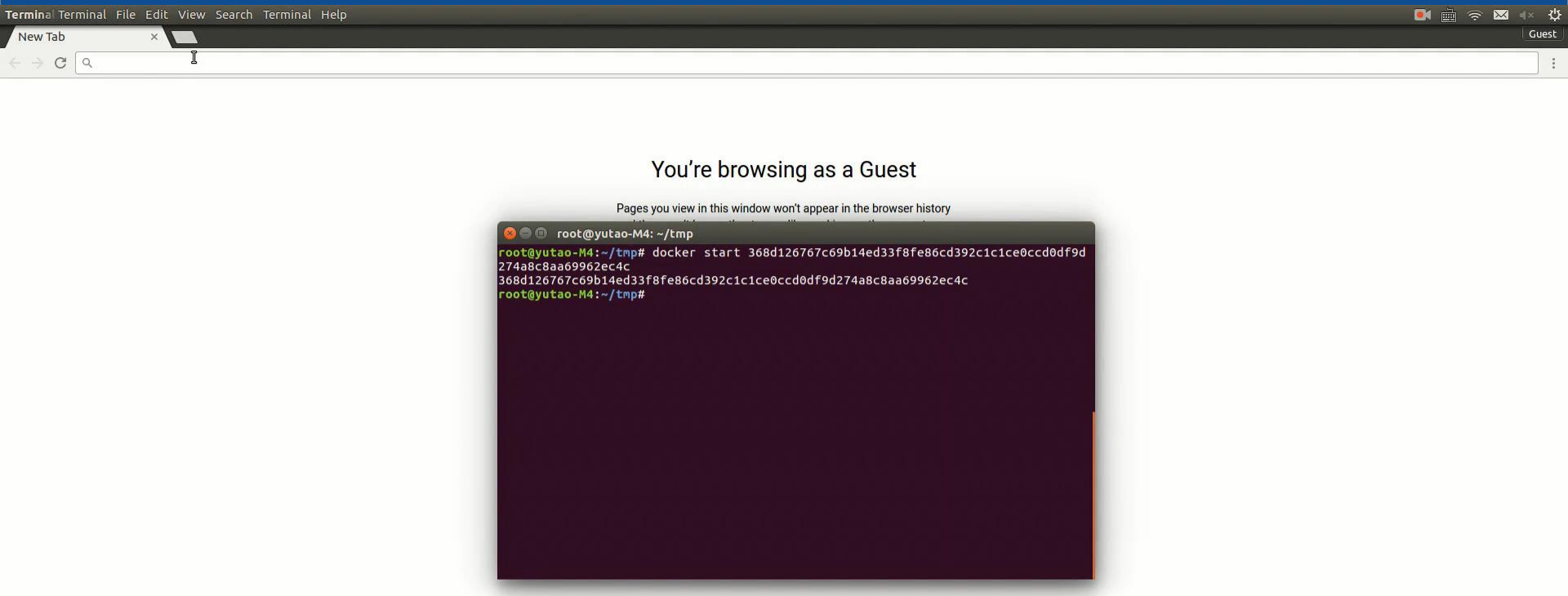
System generates a path to forward the traffic to Bro to get HTTP info (also forward the traffic to Dst by a default path).

Bro will update the controller with the HTTP info.

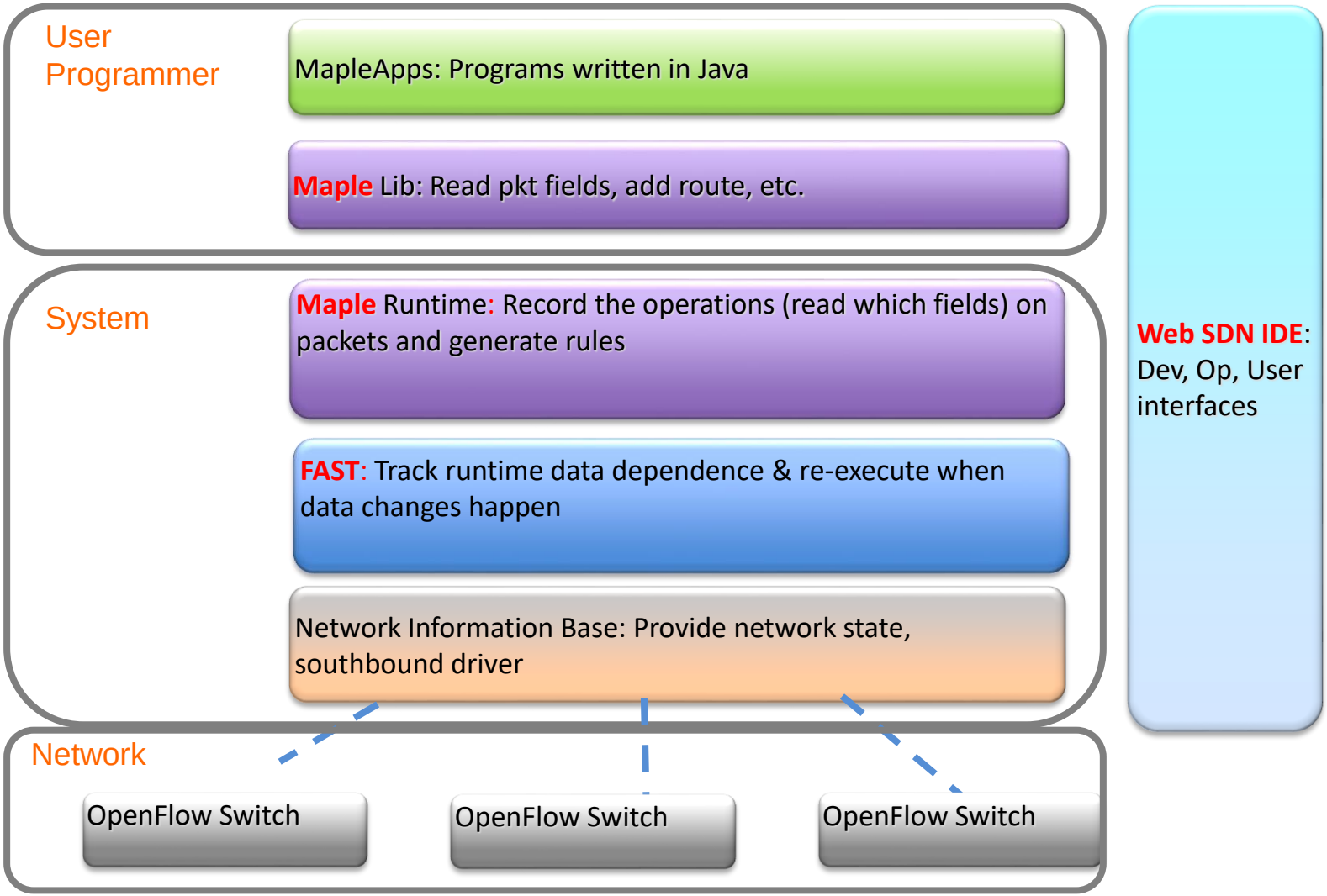
System will compute the correct path based on HTTP info.

3. Client sends a request for other type data at the Server.

Demo


Terminal Terminal File Edit View Search Terminal Help
New Tab
Guest
You're browsing as a Guest
Pages you view in this window won't appear in the browser history
root@yutao-M4: ~/tmp
root@yutao-M4:~/tmp# docker start 368d126767c69b14ed33f8fe86cd392c1c1ce0ccd0df9d
274a8c8aa69962ec4c
368d126767c69b14ed33f8fe86cd392c1c1ce0ccd0df9d274a8c8aa69962ec4c
root@yutao-M4:~/tmp#

Summary: A SDN Programming Software Stack



Past Milestones

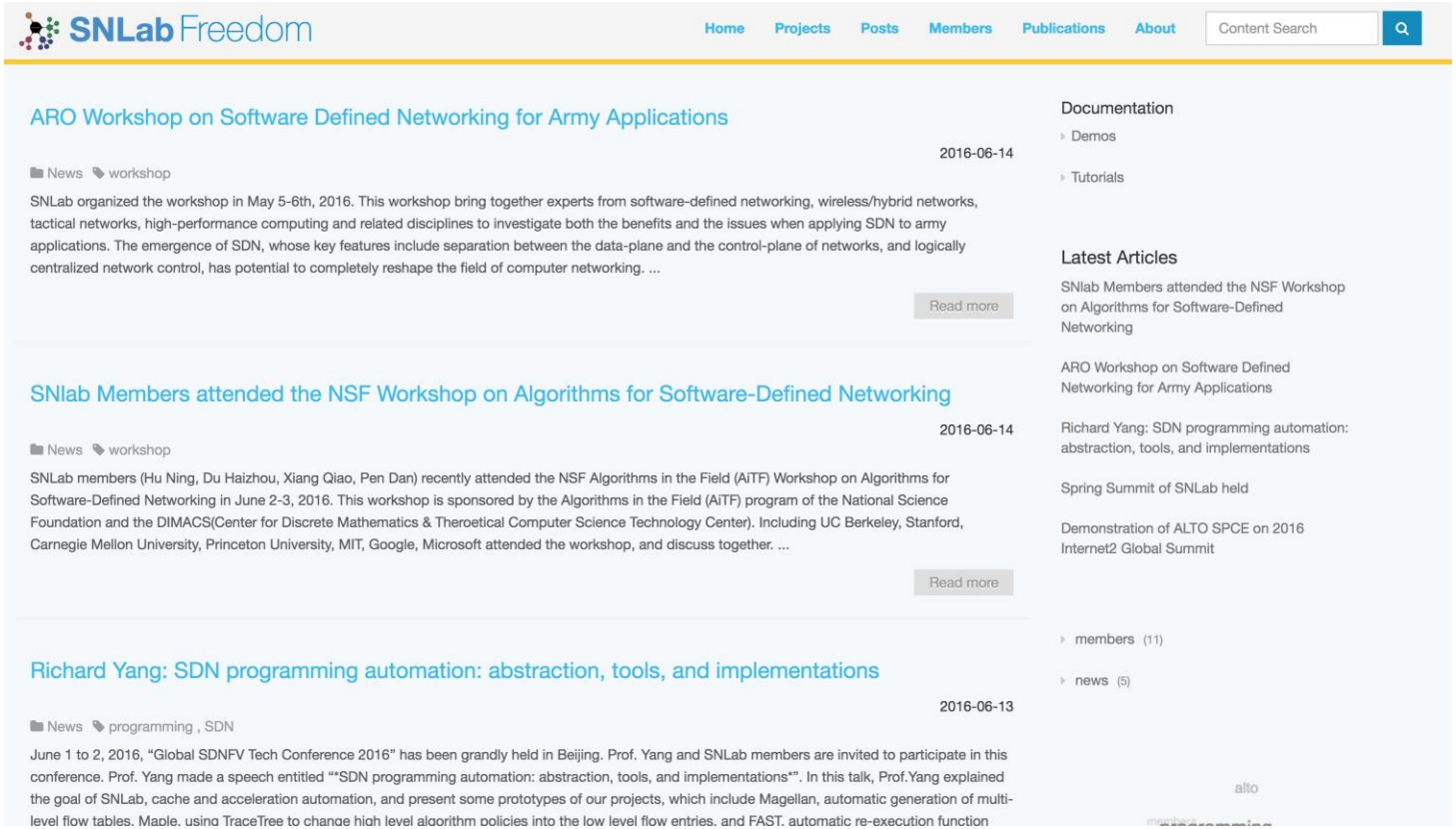
1. Sept 26, OpenDaylight Summit 2016 Tutorial

2. Oct 20, OpenDaylight Advisor Group Online Demo

3. Nov 15-17, Exhibition in SuperComputing 2016

More information

- Details: snlab-freedom.github.io
- Email: snlab.org@gmail.com



Thank You

Backup slides

What does SDN bring to us?

- Software Defined Networking (SDN) is enabling organizations to accelerate application deployment and delivery, dramatically reducing IT costs through policy-enabled workflow automation.

--Cisco

- Software defined networking controllers and applications make it possible for you to address your data center needs with an open architecture that enables programmable network control and an abstracted underlying infrastructure.

--Brocade

Agile!

The way to be agile

- Introduce extremely simple abstractions, and powerful tools, to substantially advance SDN/NFV programming
  - Why abstractions
    - "Modularity based on abstraction is the way things get done." -- *Barbara Liskov*
  - Why powerful tools
    - [control plane 要给的应该]"是猎枪，而不是猎物."

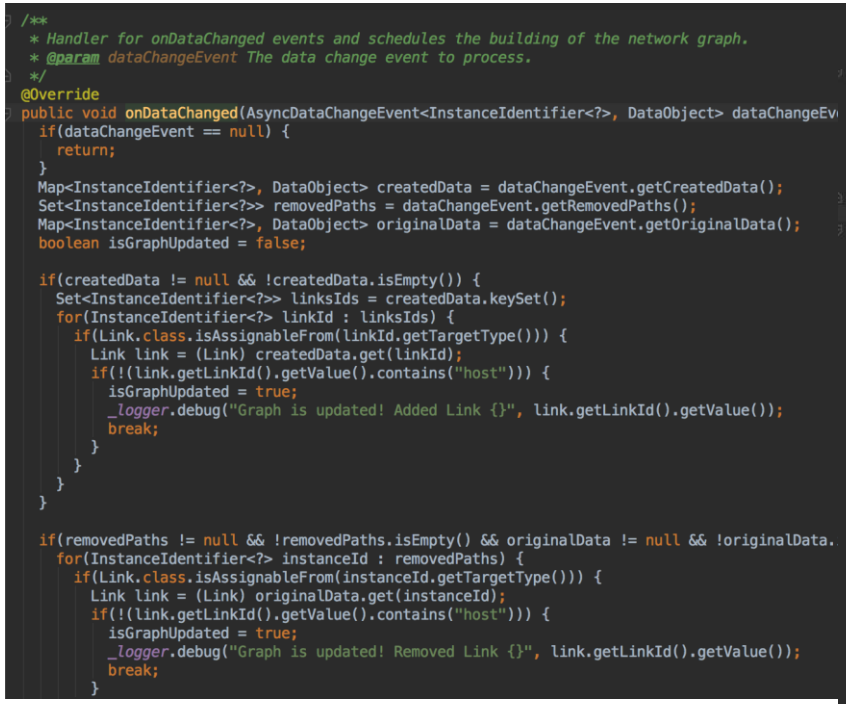
Source: *Y.Richard Yang, 2015 SDNFV Global Summit, Beijing*

Summary

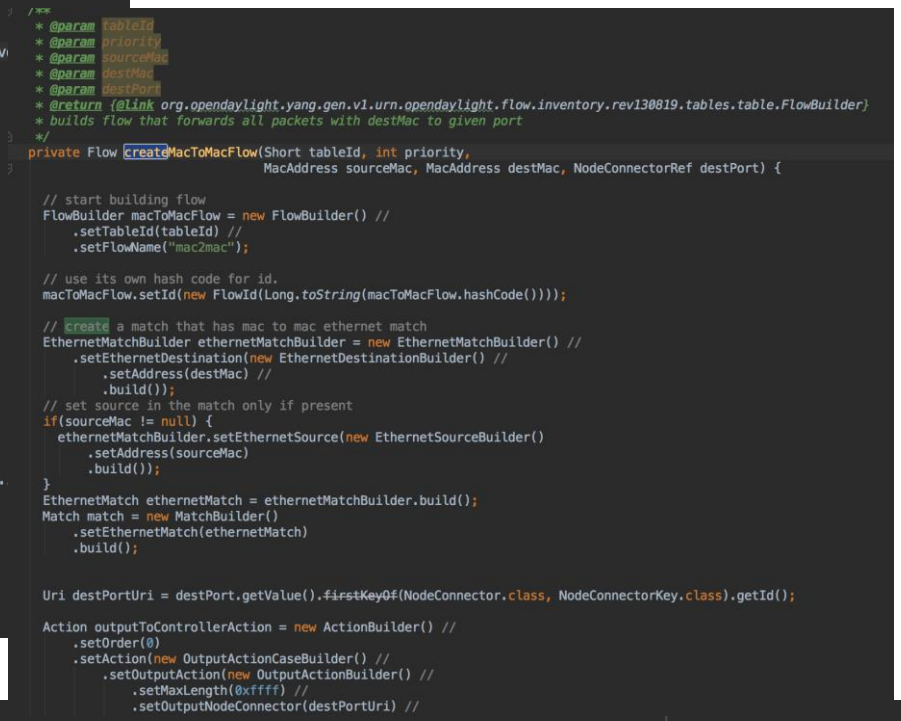
- Complex, manual programming
  - DevOpen simplifies SDN programming, automatically deploying the programs, integrating different views (programs, topology, routes)
- Low level, limited programming model
  - Maple provides high-level, per-packet, south-bound agnostic, cross-layer (L2-L7) programming
- Complex data consistency management
  - FAST automatically tracks execution dependencies, re-executing functions when dependencies change

For more information, please contact us: supersdnprogramming@gmail.com

Huge numbers of code!



```java
/**
 * Handler for onDataChanged events and schedules the building of the network graph.
 * @param dataChangeEvent The data change event to process.
 */
@Override
public void onDataChanged(AsyncDataChangeEvent<InstanceIdentifier<?>, DataObject> dataChangeEv
  if(dataChangeEvent == null) {
    return;
  }
  Map<InstanceIdentifier<?>, DataObject> createdData = dataChangeEvent.getCreatedData();
  Set<InstanceIdentifier<?>> removedPaths = dataChangeEvent.getRemovedPaths();
  Map<InstanceIdentifier<?>, DataObject> originalData = dataChangeEvent.getOriginalData();
  boolean isGraphUpdated = false;

  if(createdData != null && !createdData.isEmpty()) {
    Set<InstanceIdentifier<?>> linksIds = createdData.keySet();
    for(InstanceIdentifier<?> linkId : linksIds) {
      if(Link.class.isAssignableFrom(linkId.getTargetType())) {
        Link link = (Link) createdData.get(linkId);
        if(!(link.getLinkId().getValue().contains("host"))) {
          isGraphUpdated = true;
          _logger.debug("Graph is updated! Added Link {}", link.getLinkId().getValue());
          break;
        }
      }
    }
  }

  if(removedPaths != null && !removedPaths.isEmpty() && originalData != null && !originalData.
    for(InstanceIdentifier<?> instanceId : removedPaths) {
      if(Link.class.isAssignableFrom(instanceId.getTargetType())) {
        Link link = (Link) originalData.get(instanceId);
        if(!(link.getLinkId().getValue().contains("host"))) {
          isGraphUpdated = true;
          _logger.debug("Graph is updated! Removed Link {}", link.getLinkId().getValue());
          break;
        }
```



```java
/**
 * @param tableId
 * @param priority
 * @param sourceMac
 * @param destMac
 * @param destPort
 * @return {@link org.opendaylight.yang.gen.v1.urn.opendaylight.flow.inventory.rev130819.tables.table.FlowBuilder}
 * builds flow that forwards all packets with destMac to given port
 */
private Flow createMacToMacFlow(Short tableId, int priority,
                                MacAddress sourceMac, MacAddress destMac, NodeConnectorRef destPort) {

  // start building flow
  FlowBuilder macToMacFlow = new FlowBuilder() //
      .setTableId(tableId) //
      .setFlowName("mac2mac");

  // use its own hash code for id.
  macToMacFlow.setId(new FlowId(Long.toString(macToMacFlow.hashCode())));

  // create a match that has mac to mac ethernet match
  EthernetMatchBuilder ethernetMatchBuilder = new EthernetMatchBuilder() //
      .setEthernetDestination(new EthernetDestinationBuilder() //
          .setAddress(destMac) //
          .build());
  // set source in the match only if present
  if(sourceMac != null) {
    ethernetMatchBuilder.setEthernetSource(new EthernetSourceBuilder()
        .setAddress(sourceMac)
        .build());
  }
  EthernetMatch ethernetMatch = ethernetMatchBuilder.build();
  Match match = new MatchBuilder()
      .setEthernetMatch(ethernetMatch)
      .build();


  Uri destPortUri = destPort.getValue().firstKeyOf(NodeConnector.class, NodeConnectorKey.class).getId();

  Action outputToControllerAction = new ActionBuilder() //
      .setOrder(0)
      .setAction(new OutputActionCaseBuilder() //
          .setOutputAction(new OutputActionBuilder() //
              .setMaxLength(0xffff) //
              .setOutputNodeConnector(destPortUri) //
```

```java
/**
 * Registers as a data listener to receive changes done to
 * {@link org.opendaylight.yang.gen.v1.urn.tbd.params.xml.ns.yang.network.topology.rev131021.network.topology.topology.Link}
 * under {@link org.opendaylight.yang.gen.v1.urn.tbd.params.xml.ns.yang.network.topology.rev131021.NetworkTopology}
 * operation data root.
 */
public ListenerRegistration<DataChangeListener> registerAsDataChangeListener() {
  InstanceIdentifier<Link> linkInstance = InstanceIdentifier.builder(NetworkTopology.class)
      .child(Topology.class, new TopologyKey(new TopologyId(topologyId))).child(Link.class).build();
  return dataBroker.registerDataChangeListener(LogicalDatastoreType.OPERATIONAL, linkInstance, this, AsyncDataBroker.DataChangeScope.BASE);
}
```

For more information, please contact us: snlab.org@gmail.com

What does programmability bring to us?

- Network can quickly respond to applications
- Lower the total cost of infrastructures
- Network can be more secure

- Datacenters deploy programmable network to increase scale and utilization
- Clouds and service providers using programmable network to simply deployment, and decrease provisioning time.

DevOpen

Lightweight and portable

One-click app deployment

- Simplify the project deployment process

Multiple views for development and operations

- Developer's view (Project, Network simulator, Debugger)
- Operator's view (Topology, Routes)

For more information, please contact us: snlab.org@gmail.com