SystemX INSTITUT DE RECHERCHE TECHNOLOGIQUE

Accélérateur de la transformation numérique

INVESTISSEMENTS D'AVENIR

Securing Network Application Deployment in Software Defined Networking

Yuchia Tseng, Farid Naït-Abdesselam, and Ashfaq Khokhar

SystemX INSTITUT DE RECHERCHE TECHNOLOGIQUE

IOWA STATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCIENCE with PRACTICE

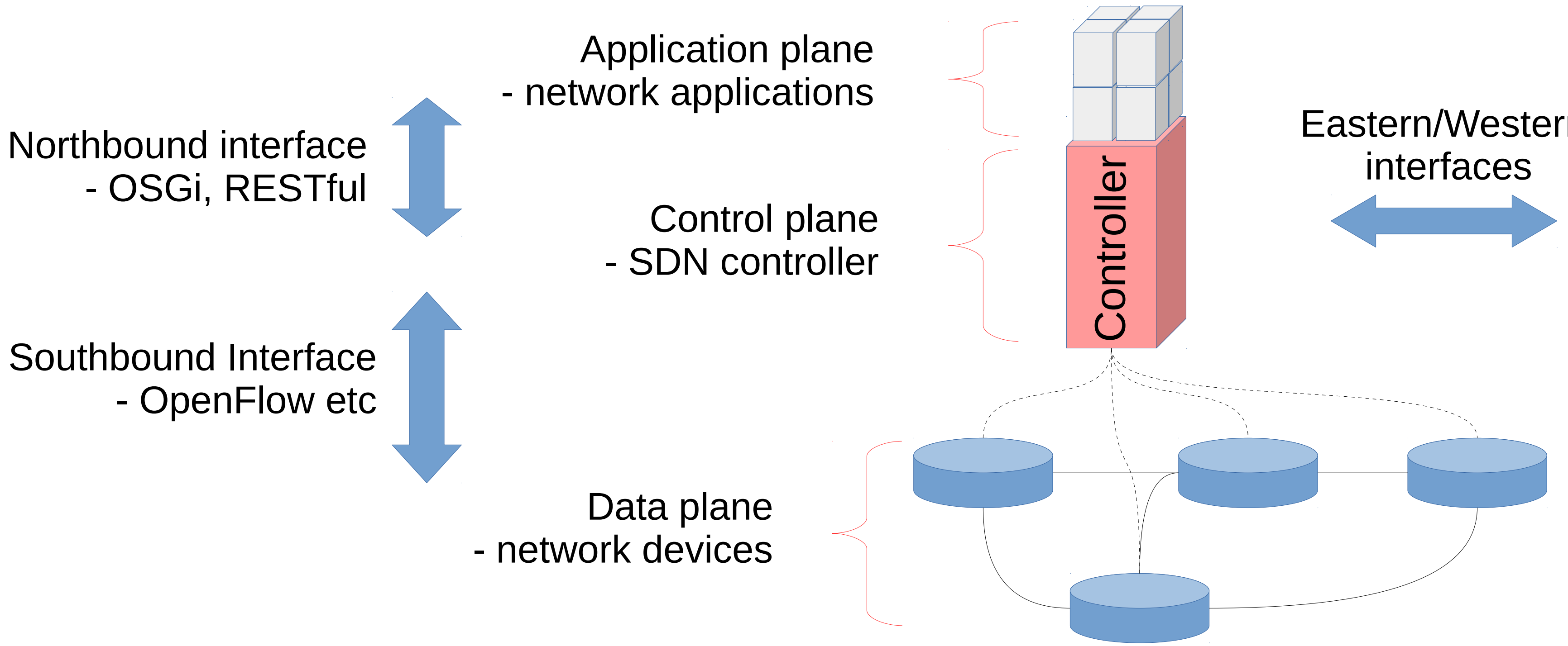
Outline

- Introduction to OpenFlow-based SDN
- Security issues of network application deployment in SDN
- Securing network application deployment in SDN
- Prototype evaluation
- Conclusion

Overview of OpenFlow-based SDN

Software-defined Networking(SDN)

- A new network paradigm decouples the control plane from the data plane
- Benefit : Provide centralized control and visibility over network
- Architecture
  - 3 planes: Data plane, Control plane, & Application plane
  - 3 interfaces: Southbound, Northbound, & Eastern/Western bound interfaces

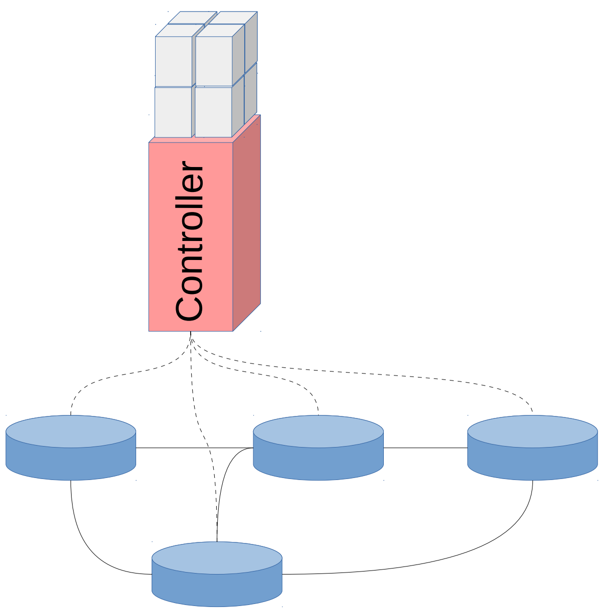


Overview of OpenFlow-based SDN

- **Data plane**
  - Networking devices for forwarding packets
  - Example: Open vSwitch, HP 2920/3500 Switch Series etc
- **Control plane**
  - Network OS, or called SDN controller, running on general purpose HW & OS
  - Example: OpenDaylight, ONOS, Floodlight, Ryu, NOX etc
- **Application plane**
  - Networking applications installed on SDN controller to enable network becoming intelligent
  - Example: Firewall, Load balancer, and IDS etc
- **Southbound interface**
  - Communication protocol between data plane & control plane
  - Example: OpenFlow, NETCONF, Border Gateway Protocol (BGP), Open vSwitch Database Management Protocol (OVSDB), MPLS Transport Profile (MPLS-TP) etc
- **Northbound interface**
  - Application programming interfaces (APIs) between control plane & application plane
  - Example: OSGi, native Java/Python APIs, RESTful APIs etc
- **Eastern/western interfaces**

Overview of OpenFlow-based SDN

- **Controller uses OpenFlow to keep listening what happens on the data plane, and then sends the network events to the application plane**
- **OpenFlow messages**
  - packet_in : send a captured packet to the controller (eg., a miss in the match tables)
  - packet_out : inject packets into the data plane
  - flow_mod : modify the state of an OpenFlow switch(eg., add a flow entry)
  - port_mod : modify the state of an OpenFlow port
  - port_status : indicate a change of port status



hello
packet_in
port_status
feature_res
error

hello
packet_out
flow_mod
port_mod
feature_req

- **If controller uses OpenFlow protocol to keep listening what happens on the data plane**
- **How can the network applications *KEEP LISTENING* the network events on the data plane through the SDN controller ?**
  - Internal APIs, such as OSGi, Java/Python APIs
  - External APIs, such as RESTful APIs(JSON-RPC)

Security Issues of Network Application Deployment in SDN

- **Security problem of using internal APIs**
  - Easier to be compromised by code injection

```
in] DEBUG n.f.core.internal.Controller - OFListeners for PACKET_IN: net.floodlightcontroller.attack
in] INFO  n.f.core.internal.Controller - Listening for switch connections on 0.0.0.0/0.0.0.0:6633
w I/O server worker #1-1] INFO  n.f.core.internal.Controller - New switch connection from /127.0.0.
w I/O server worker #1-2] INFO  n.f.core.internal.Controller - New switch connection from /127.0.0.
w I/O server worker #1-1] DEBUG n.f.core.internal.Controller - This controller's role is null, not
w I/O server worker #1-2] DEBUG n.f.core.internal.Controller - This controller's role is null, not
w I/O server worker #1-2] INFO  n.floodlightcontroller.attack.Crash - [ATTACK] Crash Application
~/floodlight-0.90#
```

App calls the System.exit function

Floodlight crash by calling `System.exit()`

```
19:52:44.229 [New I/O server worker #1-1] INFO  n.f.attack.MemoryLeak - [ATTACK] MemoryLeak Application
19:52:44.361 [New I/O server worker #1-1] ERROR n.f.core.internal.Controller - Error while processing me
java.lang.OutOfMemoryError: Java heap space
        at net.floodlightcontroller.attack.MemoryLeak.receive(MemoryLeak.java:59) ~[floodlight.jar:na]
        at net.floodlightcontroller.core.internal.Controller.handleMessage(Controller.java:1285) ~[flood
        at net.floodlightcontroller.core.internal.Controller$OFChannelHandler.processOFMessage(Controlle
        at net.floodlightcontroller.core.internal.Controller$OFChannelHandler.messageReceived(Controller
        at org.jboss.netty.handler.timeout.IdleStateAwareChannelUpstreamHandler.handleUpstream(IdleState
        at org.jboss.netty.handler.timeout.ReadTimeoutHandler.messageReceived(ReadTimeoutHandler.java:18
```

FloodLight - Out of Memory Error

Floodlight memory leakage by inserting data into list

```
2014-05-12 09:26:33.219 PDT [Statistics Collector] DEBUG o.o.c.p.o.i.InventoryServiceShim - Connection service
accepted the inventory notification for OF|00:00:00:00:00:00:00:02 CHANGED
2014-05-12 09:26:34.217 PDT [Statistics Collector] DEBUG o.o.c.c.internal.ConnectionManager - updateNode: OF|00
:00:00:00:00:00:00:03 type CHANGED props [Description[None]]
2014-05-12 09:26:34.217 PDT [Statistics Collector] DEBUG o.o.c.s.internal.SwitchManager - updateNode: OF|00:00:
00:00:00:00:00:03 type CHANGED props [Description[None]] for container default
2014-05-12 09:26:34.217 PDT [Statistics Collector] DEBUG o.o.c.p.o.i.InventoryServiceShim - Connection service
accepted the inventory notification for OF|00:00:00:00:00:00:00:03 CHANGED
2014-05-12 09:26:51.791 PDT [SwitchEvent Thread] DEBUG o.o.c.h.internal.HostTracker - Received for Host: IP 10.
0.0.1, MAC 000000000001, HostNodeConnector [nodeConnector=OF|1@OF|00:00:00:00:00:00:00:01, vlan=0, staticHost=f
alse, arpSendCountDown=0]
2014-05-12 09:26:51.794 PDT [Thread-37] DEBUG o.o.c.h.internal.HostTracker - New Host Learned: MAC: 00000000000
1  IP: 10.0.0.1
2014-05-12 09:26:51.794 PDT [Thread-37] DEBUG o.o.c.h.internal.HostTracker - Notifying Applications for Host 10
.0.0.1 Being Added
2014-05-12 09:26:51.795 PDT [Thread-37] DEBUG o.o.c.h.internal.HostTracker - Notifying Topology Manager for Hos
t 10.0.0.1 Being Added
2014-05-12 09:26:51.796 PDT [SwitchEvent Thread] INFO  o.o.controller.attack.crash.Crash - [ATTACK.CRASH] Packe
t Received
2014-05-12 09:26:51.796 PDT [SwitchEvent Thread] INFO  o.o.controller.attack.crash.Crash - [ATTACK.CRASH] Syste
m.exit() called
2014-05-12 09:26:51.798 PDT [Listener:59957] DEBUG com.arjuna.ats.arjuna - Recovery listener existing com.arjun
a.ats.arjuna.recovery.ActionStatusService
2014-05-12 09:26:51.798 PDT [Thread-11] DEBUG org.jgroups.stack.GossipRouter - ConnectionHandler[peer: /127.0.0
.1, logical_addrs: localhost-12306] is being closed
2014-05-12 09:26:51.805 PDT [Thread-11] DEBUG org.jgroups.stack.GossipRouter - router stopped
$
```

Call a System Exit Function

OpenDayLight has been crashed

OpenDaylight crash by calling `System.exit()`

More problems: run infinite loop, create numerous threads, listen to the network traffic, or insert code through JNI etc

Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., Yegneswaran, V., Noh, J., and Kang, B. B. (2014). Rosemary: A robust, secure, and high-performance network operating system, CCS '14.

Security Issues of Network Application Deployment in SDN

- **Security problem of using external APIs**
  - Can be compromised by API abuse
  - READ permission : Saturating the bandwidth of the northbound interface by using infinite loop to request the northbound APIs
  - ADD permission : Flow tables is limited. Eg., High-performance chips EZchip NP-4 stores 125 000 – 1 000 000 flow entries. The attacker can flush out the high priority flow rules by the low priority ones
  - UPDATE & REMOVE permission : The APIs can be used to compromise the higher priority flow rule due to the coarse-grained access control

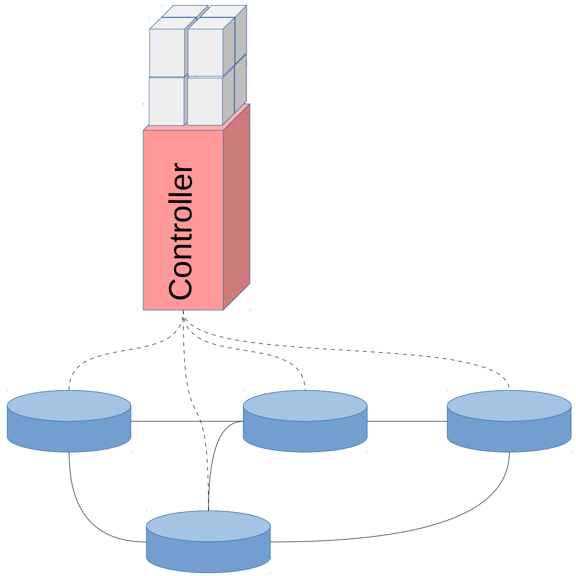
	FLOODLIGHT	ODL	ONOS	RYU
RAED	Packet dropping	Responding slower	Packet dropping	Responding slower
ADD	Flow entry limitation: 148223	Flow entry limitation: 140000	Flow entry limitation: 45000	
UPDATE		.../sal-flow:update-flow	.../devices/<deviceId>	.../flowentry/modify
REMOVE		.../sal-flow:remove-flow	../flows/<deviceId>/<flowId>	.../flowentry/clear/<dpid>

***Tseng, Y.**, Pattaranantakul, M., He, R., Zhang, Z., and Naït-Abdesselam, F. (2017), Controller DAC: Securing SDN Controller with Dynamic Access Control, ICC '17.*

Security Issues of Network Application Deployment in SDN

So, how to deploy network application securely for SDN controller?

- Internal APIs(OSGi, Java/Python APIs etc)?
  - ➔ Malicious code injection…
- External APIs (JSON-RPC)?
  - ➔ API abuse
  - ➔ Hard to get the network events in real time
- IPC?
  - ➔ System-level command injection, eg:
    - ➔ `Runtime.getRuntime().exec("shutdown -s -t 0");`
  - ➔ High complexity to deploy on the current existing controllers

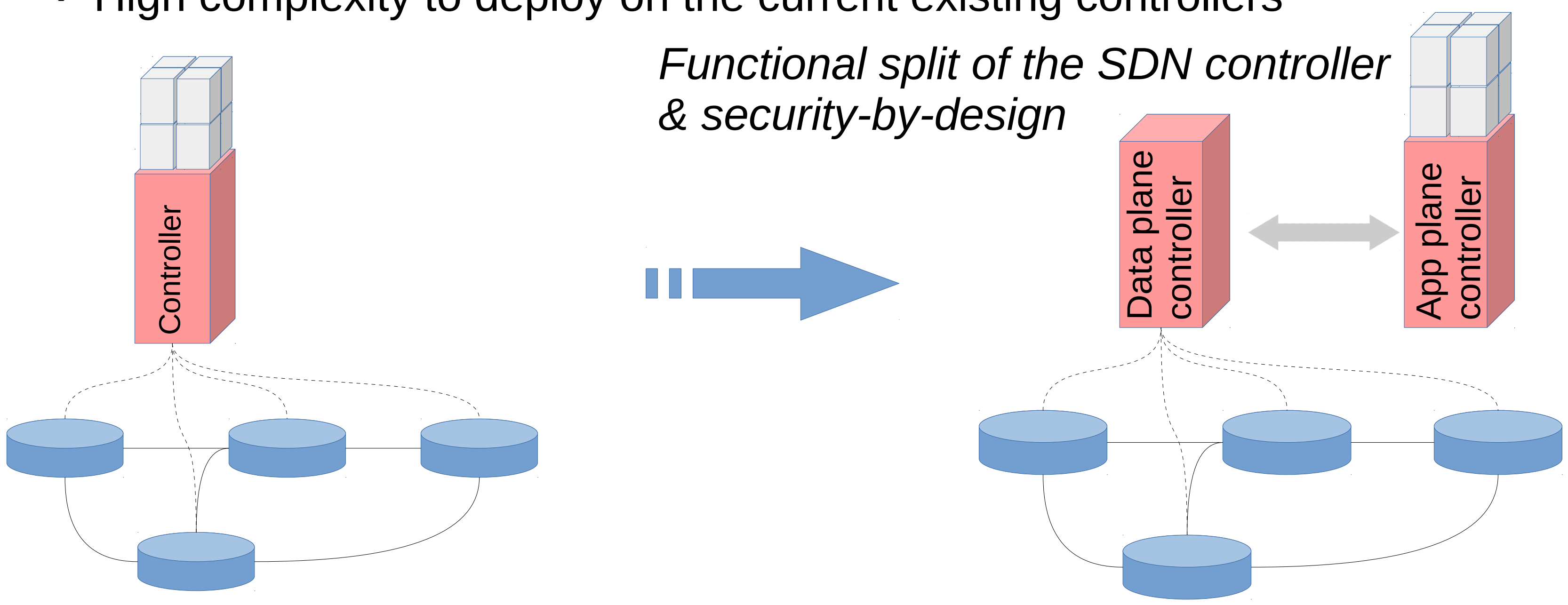


Securing Network Application Deployment in SDN

So, how to deploy network application securely for SDN controller?

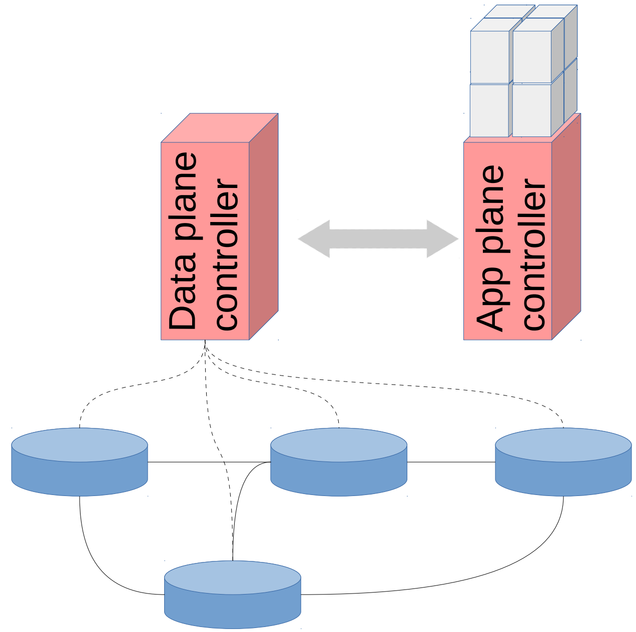
- Internal APIs(OSGi, Java/Python APIs etc)?
  - ➔ Malicious code injection…
- External APIs (JSON-RPC)?
  - ➔ API abuse
  - ➔ Hard to get the network events in real time
- IPC?
  - ➔ System-level command injection, eg:
    - ➔ `Runtime.getRuntime().exec("shutdown -s -t 0");`
  - ➔ High complexity to deploy on the current existing controllers

Functional split of the SDN controller & security-by-design



Securing Network Application Deployment in SDN

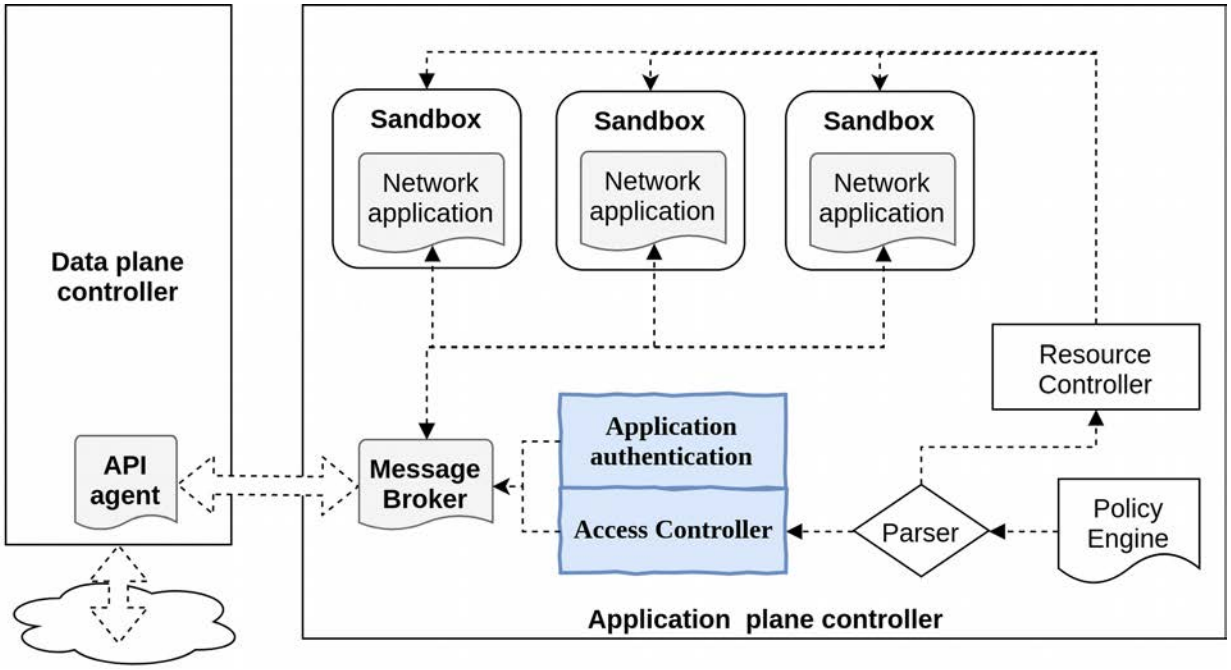
Application plane controller
- Dedicate all the requests from the application plane
- Application authentication
- Application authorization (access control)
- Application resource isolation, control, & monitoring
- Message-driven service (instead of server-client mode)



Data plane controller
- Interpret the network rules ↔ OF entries
- Communicate with the data plane

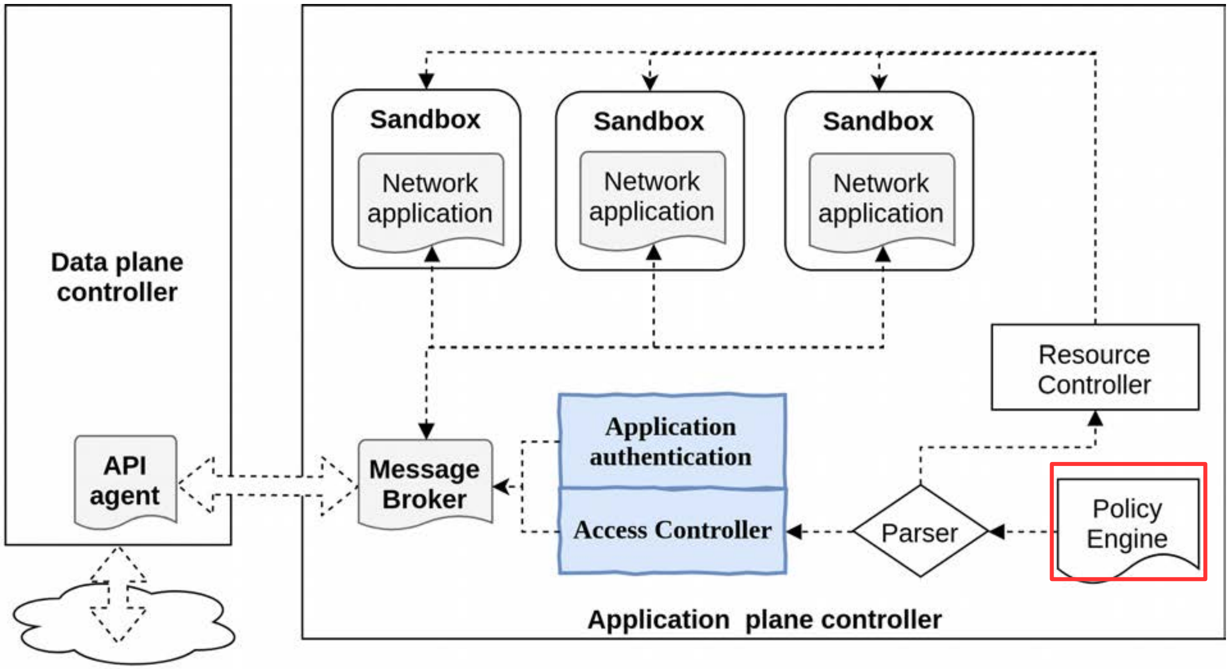
Securing Network Application Deployment in SDN

- **Policy Engine**: define high-level policy
- **Parser**: translate high-level policies to Access Controller & Resource Controller
- **Resource Controller**: control and monitor resource usage of network application
- **Application Sandbox**: isolate the resource usage of the network application
- **Authentication module**: authenticate the network application
- **Access Controller**: authorize the requests of network applications
- **Message Broker**: provide message-driven service to network applications



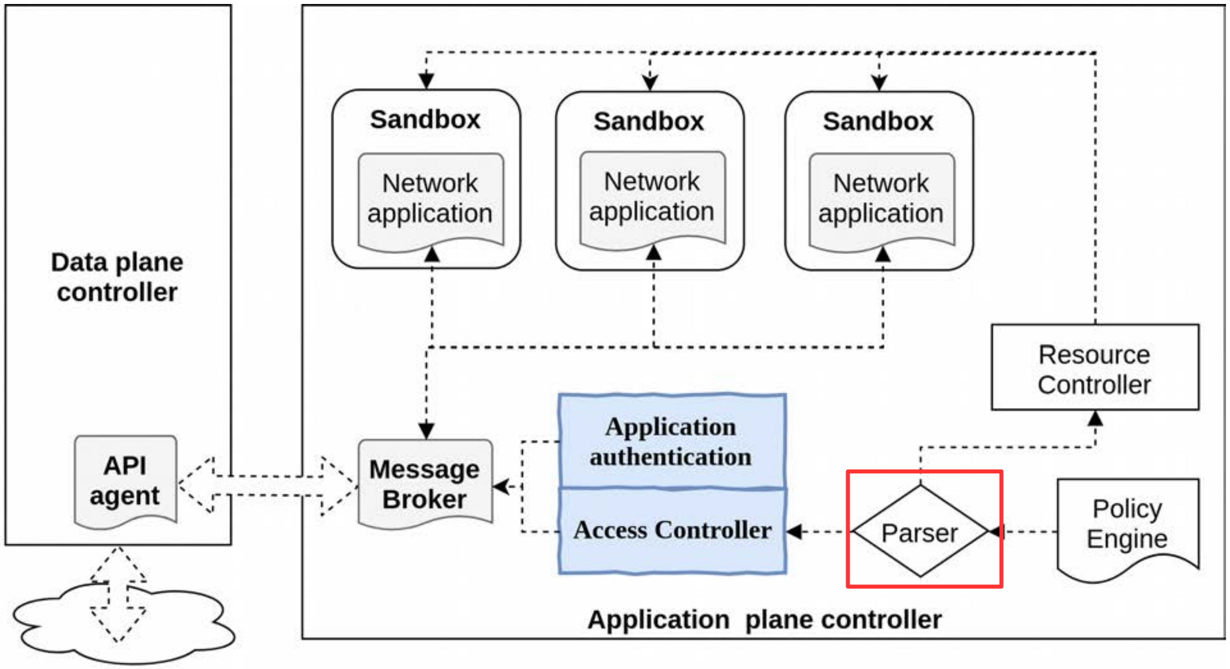
Securing Network Application Deployment in SDN

- **Policy Engine**: define high-level policy
- **Parser**: translate high-level policies to Access Controller & Resource Controller
- **Resource Controller**: control and monitor resource usage of network application
- **Application Sandbox**: isolate the resource usage of the network application
- **Authentication module**: authenticate the network application
- **Access Controller**: authorize the requests of network applications
- **Message Broker**: provide message-driven service to network applications



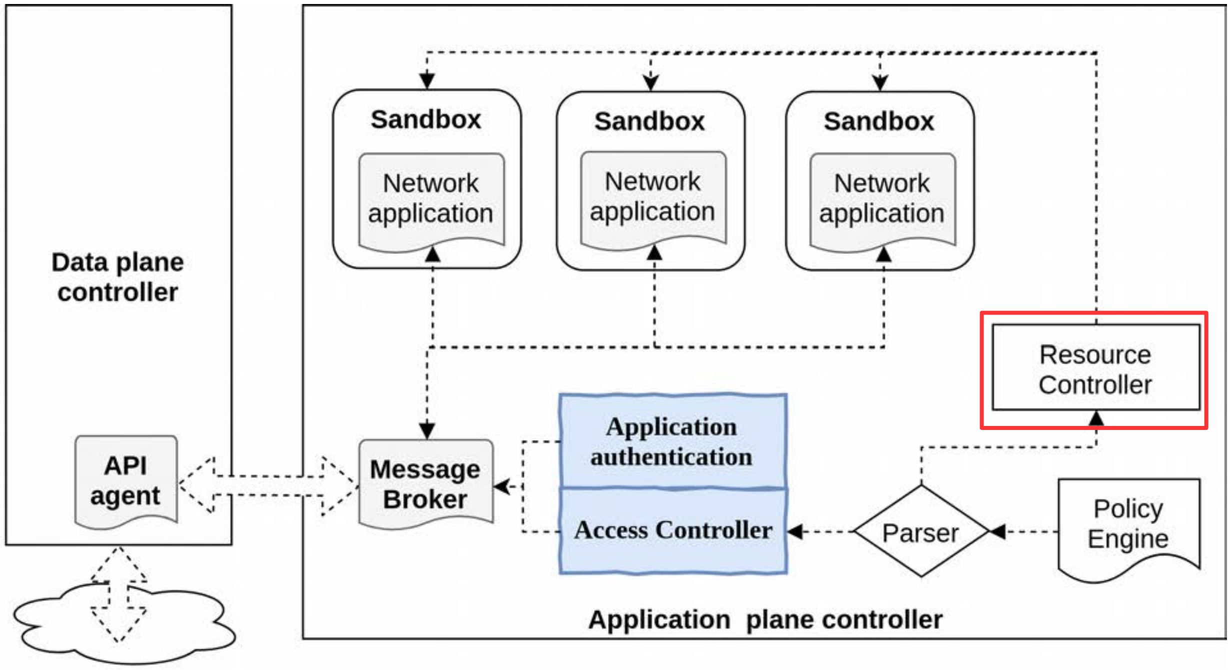
Securing Network Application Deployment in SDN

- **Policy Engine**: define high-level policy
- **Parser**: translate high-level policies to Access Controller & Resource Controller
- **Resource Controller**: control and monitor resource usage of network application
- **Application Sandbox**: isolate the resource usage of the network application
- **Authentication module**: authenticate the network application
- **Access Controller**: authorize the requests of network applications
- **Message Broker**: provide message-driven service to network applications



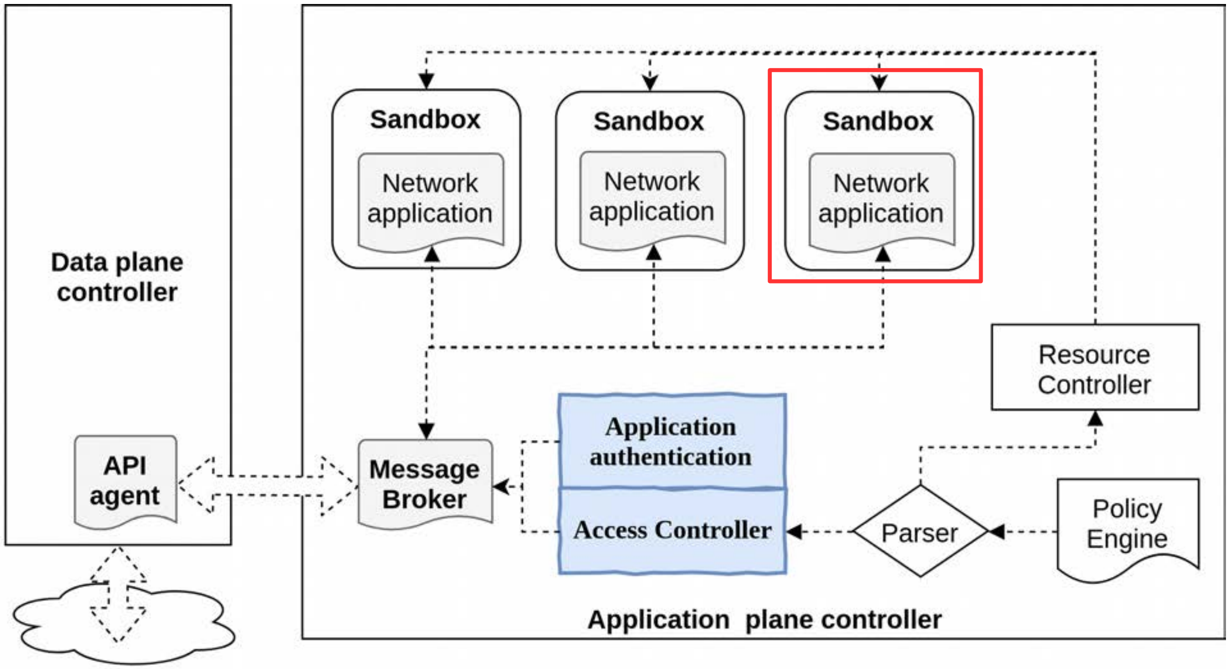
Securing Network Application Deployment in SDN

- **Policy Engine**: define high-level policy
- **Parser**: translate high-level policies to Access Controller & Resource Controller
- **Resource Controller**: control and monitor resource usage of network application
- **Application Sandbox**: isolate the resource usage of the network application
- **Authentication module**: authenticate the network application
- **Access Controller**: authorize the requests of network applications
- **Message Broker**: provide message-driven service to network applications



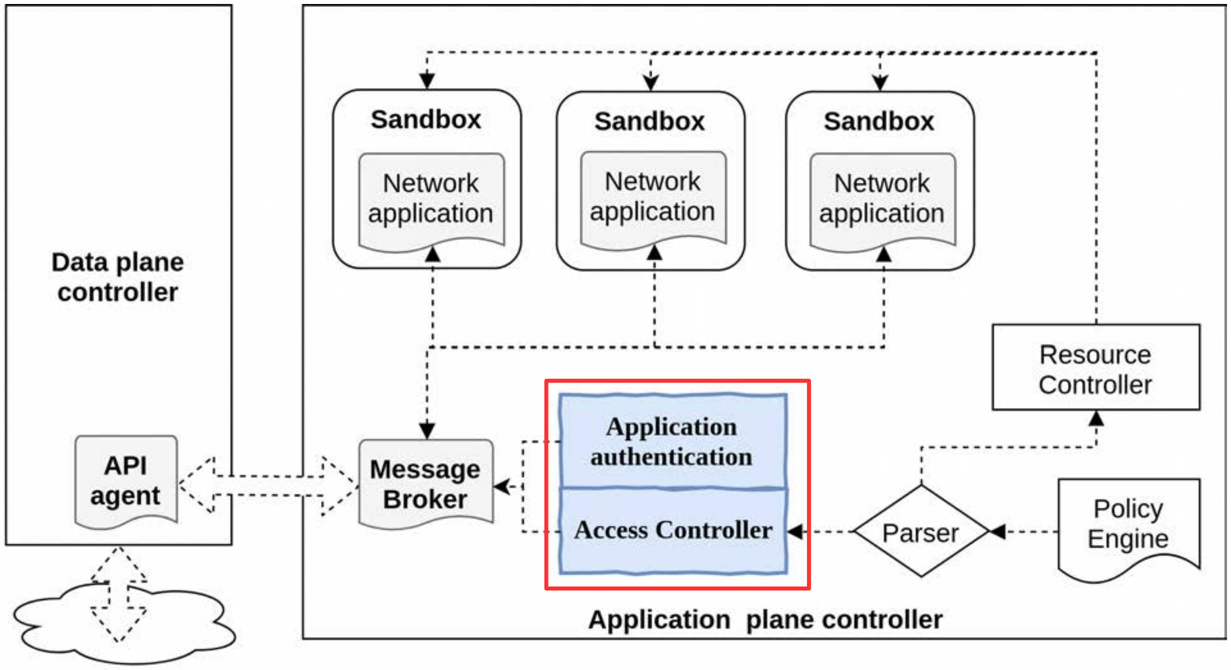
Securing Network Application Deployment in SDN

- **Policy Engine**: define high-level policy
- **Parser**: translate high-level policies to Access Controller & Resource Controller
- **Resource Controller**: control and monitor resource usage of network application
- **Application Sandbox**: isolate the resource usage of the network application
- **Authentication module**: authenticate the network application
- **Access Controller**: authorize the requests of network applications
- **Message Broker**: provide message-driven service to network applications



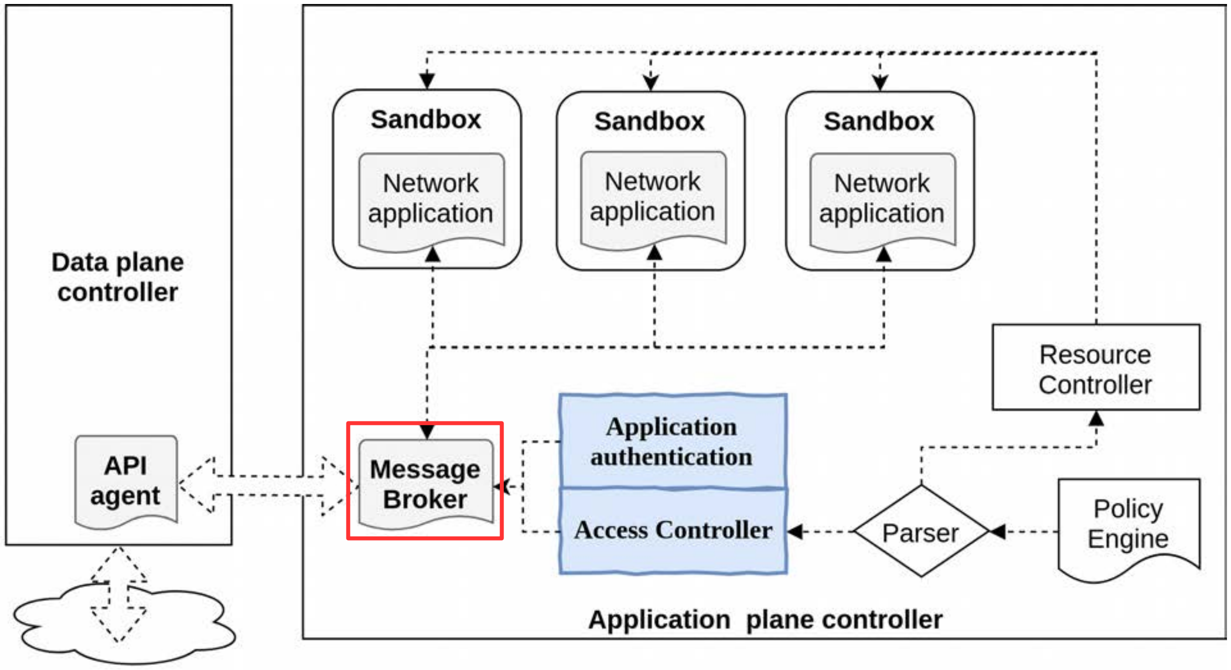
Securing Network Application Deployment in SDN

- **Policy Engine**: define high-level policy
- **Parser**: translate high-level policies to Access Controller & Resource Controller
- **Resource Controller**: control and monitor resource usage of network application
- **Application Sandbox**: isolate the resource usage of the network application
- **Authentication module**: authenticate the network application
- **Access Controller**: authorize the requests of network applications
- **Message Broker**: provide message-driven service to network applications



Securing Network Application Deployment in SDN

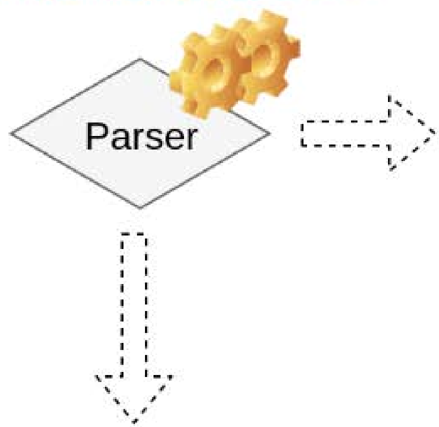
- **Policy Engine**: define high-level policy
- **Parser**: translate high-level policies to Access Controller & Resource Controller
- **Resource Controller**: control and monitor resource usage of network application
- **Application Sandbox**: isolate the resource usage of the network application
- **Authentication module**: authenticate the network application
- **Access Controller**: authorize the requests of network applications
- **Message Broker**: provide message-driven service to network applications



Prototype Evaluation

High-level policy in YAML

```yaml
- appname: app1
  cpus: 0.2
  memory: 128 MB
  storage: 100 MB
  networking: false
  readableServices:
    - DeviceInfo
    - LinkInfo
    - Statistics
    - ProactiveFlowEntry
  writableServices:
    - ProactiveFlowEntry
```



Json for Marathon framework

```json
{
  "id": "app1",
  "cpus": 0.2,
  "mem": 128,
  "disk": 100,
  "instances": 1,
  "container": {
    "docker": {
      "image": "sdn/netapp:v2",
      "network": "BRIDGE"
    },
    "type": "DOCKER"
  }
}
```

```
--allow-principalUser:app1 --consumer --topic DeviceInfo  --group *
--allow-principalUser:app1 --consumer --topic LinkInfo  --group *
--allow-principalUser:app1 --consumer --topic Statistics  --group *
--allow-principalUser:app1 --consumer --topic ProactiveFlowEntry  --group *
--allow-principalUser:app1 --producer --topic ProactiveFlowEntry  --group *
```

ACLs for Kafka server

Prototype Evaluation

Access Controller
- App 1 can read DeviceInfo messages
- App 1 is rejected to read Configuration messages

```
App read pack-in through message bus  DeviceInfo  {"srcIpv4:""10.0.0.3","dstIpv4:""10.0.0.4""srcMac":"96:38:6f:50:df:b5""dstMac":"f6:8a:6f:b2:e6:8b"}
App read pack-in through message bus              {"srcIpv4:""10.0.0.4","dstIpv4:""10.0.0.3""srcMac":"f6:8a:6f:b2:e6:8b""dstMac":"96:38:6f:50:df:b5"}
App read pack-in through message bus northbound: {"srcIpv4:""10.0.0.3","dstIpv4:""10.0.0.4""srcMac":"96:38:6f:50:df:b5""dstMac":"f6:8a:6f:b2:e6:8b"}
2017-08-31 13:06:31.235 WARN  [o.a.k.c.NetworkClient] Error while fetching metadata with correla  Configuration  {Configuration=UNKNOWN_TOPIC_OR_PARTITION}
2017-08-31 13:06:31.434 WARN  [o.a.k.c.NetworkClient] Error while fetching metadata with correla              {Configuration=UNKNOWN_TOPIC_OR_PARTITION}
2017-08-31 13:06:31.636 WARN  [o.a.k.c.NetworkClient] Error while fetching metadata with correlation id 79 : {Configuration=UNKNOWN_TOPIC_OR_PARTITION}
2017-08-31 13:06:31.837 WARN  [o.a.k.c.NetworkClient] Error while fetching metadata with correlation id 80 : {Configuration=UNKNOWN_TOPIC_OR_PARTITION}
```

Resource Controller
- An application contains a malicious loop to keep inserting data in a HashMap

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 83% dropped (2/12 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
mininet>
```

When the code runs on the runtime of controller (JVM for Floodlight), after 1 minute, the controller can no longer serve the data plane.

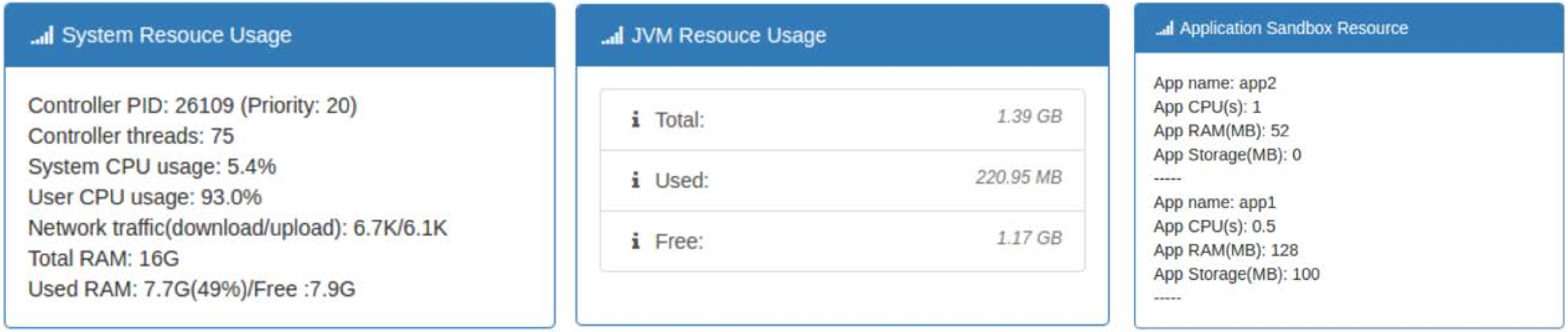
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

When the code runs in the sandbox (Docker with 0.2 CPU & 128 RAM), the controller keeps serving the data plane.

Prototype Evaluation

Resource usage monitoring dashboard (implemented on Floodlight)

- System-level resource usage
- JVM-level resource usage
- Application sandbox resource usage



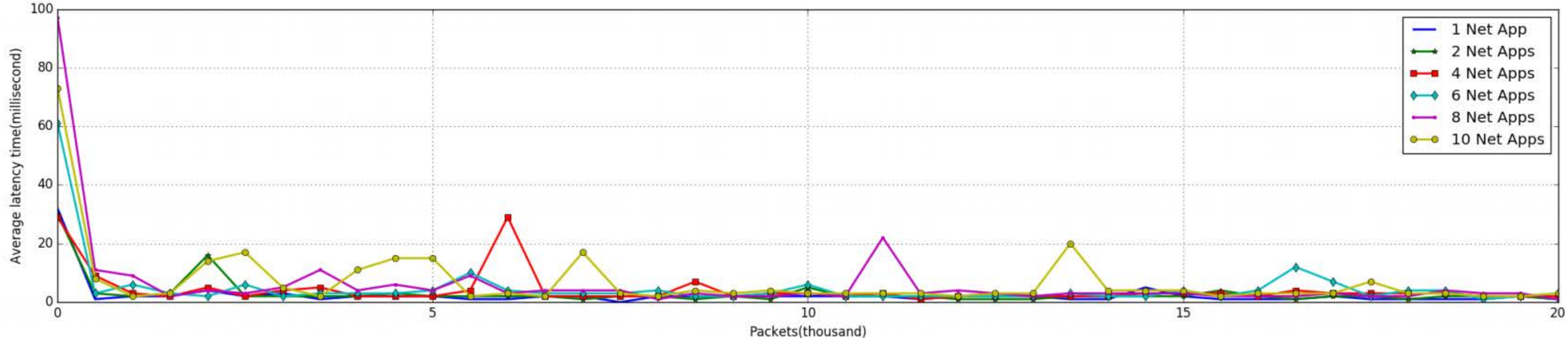
Prototype Evaluation

Testbed configuration

Component	Tool
Service broker	Apache Kafka
Resource controller	Apache Mesos API and Sigar
Application deployer	Marathon framework
Application sandbox	Docker
Policy	YAML
Parser	Java application
SDN controller	Floodlight master(v1.2)
Network simulator	mininet
OS	Ubuntu 16.04.2
CPU	Intel i7
Memory	16G

Sandbox number	Processing time(ms)
1	2.5946
2	2.9099
3	3.3204
4	4.4183
5	6.8173
6	7.3950
7	9.8372
8	8.9402
9	9.5253
10	13.2072

The average processing time for delivering 10 thousands *packet_in* messages from the data plane controller to 1-10 network application(s)



Processing time for delivering 20 thousands *packet_in* messages to 1-10 network applications

Conclusion

Problems

- How to deploy network applications securely on the SDN controller?

Solution

- Functional split of the SDN controller & security-by-design
- Application controller:
  - Resource Controller: resource isolation, control, and monitoring → against code injection and command injection
  - Message Broker + Access Controller → against API abuse

Preliminary results

- This architecture can implement in the existing SDN controller by adding a new module (agent)
- The latencies keeps around 5 milliseconds in long term for delivering to 10 network applications

Systemx INSTITUT DE RECHERCHE TECHNOLOGIQUE

Accélérateur de la transformation numérique

Thank you!
Questions?

www.irt-systemx.fr