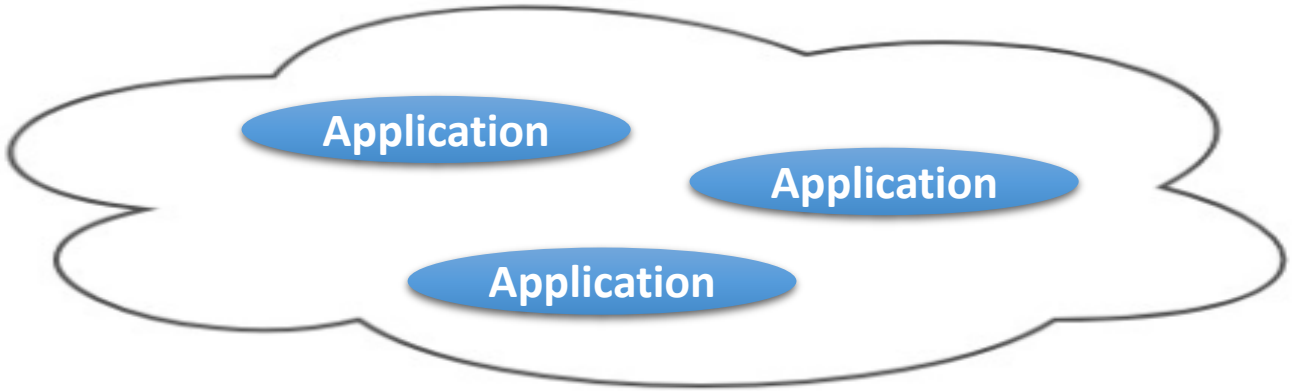
SDNKeeper: Lightweight Resource Protection and Management System for SDN-based Cloud

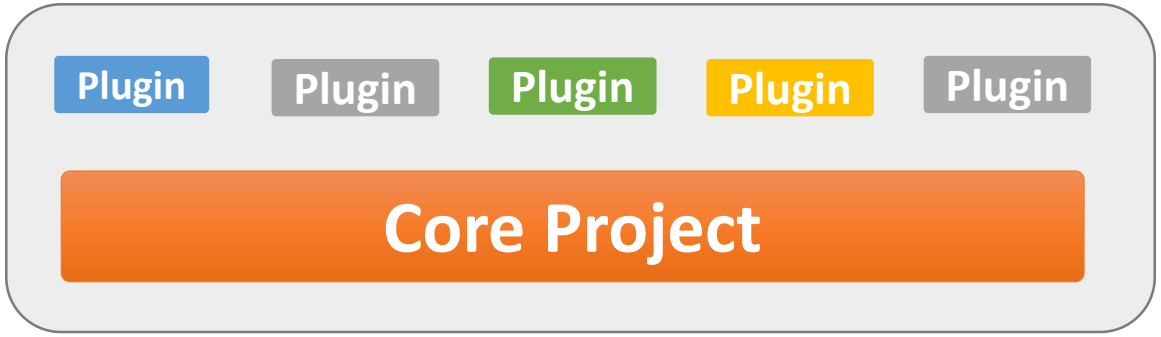
Xue Leng*

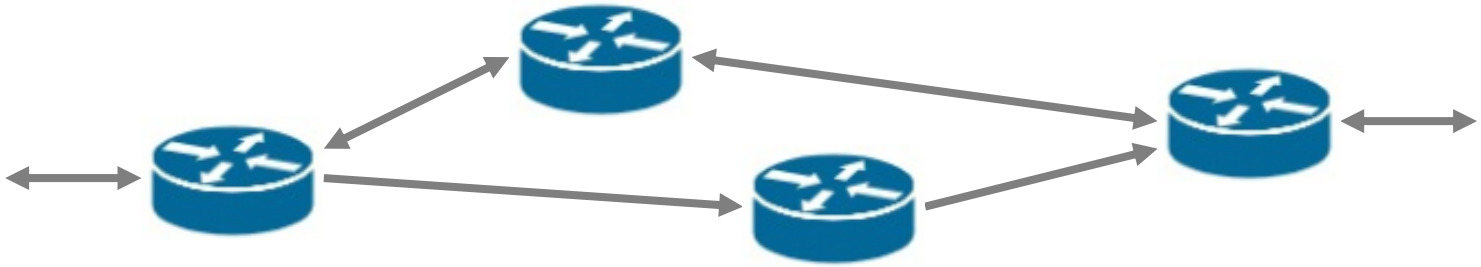
Kaiyu Hou#, Yan Chen*#, Kai Bu*, Libin Song#

*Zhejiang University** *Northwestern University#*

Background

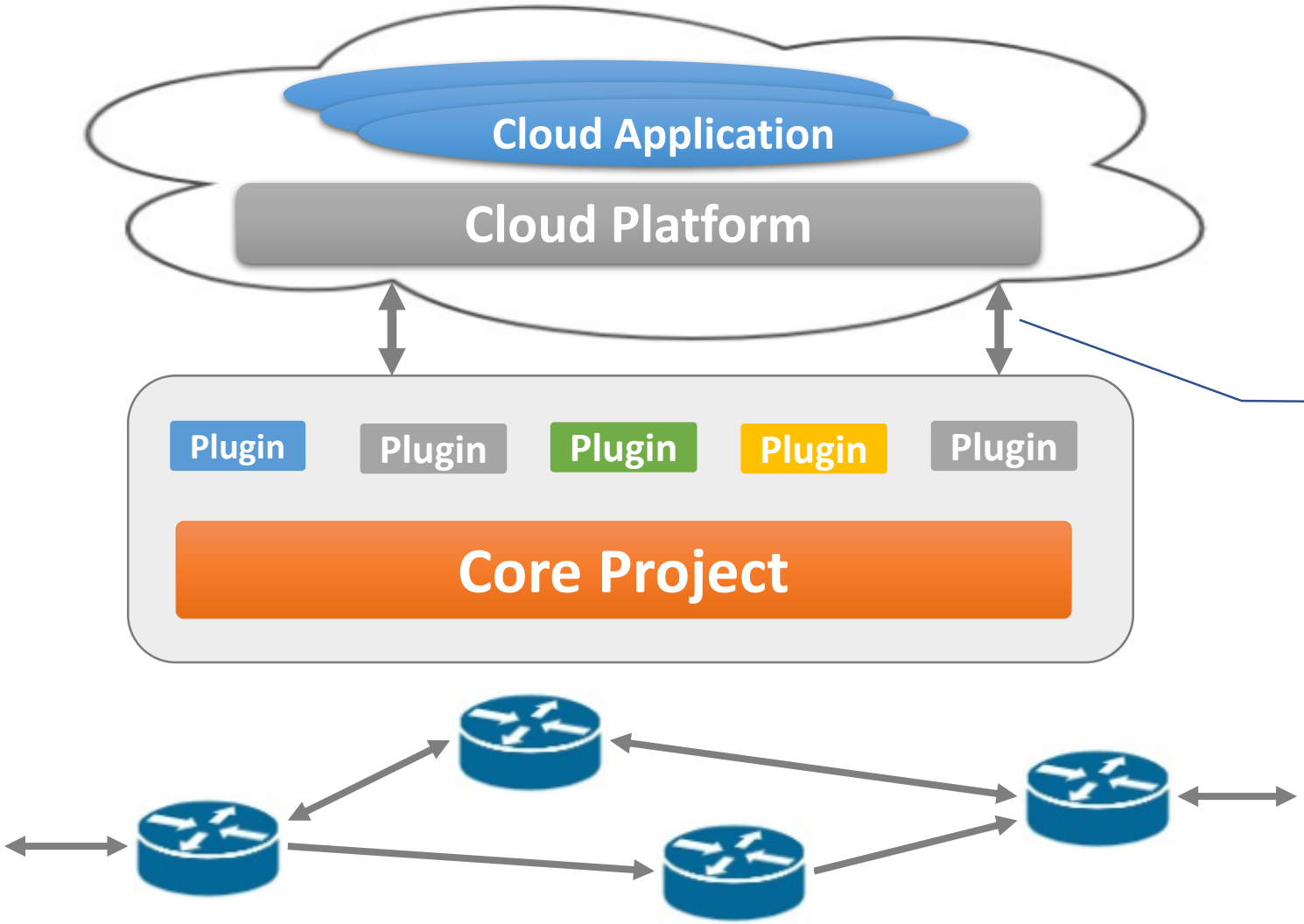






What is SDN ?
SDN-based Cloud ?

Background

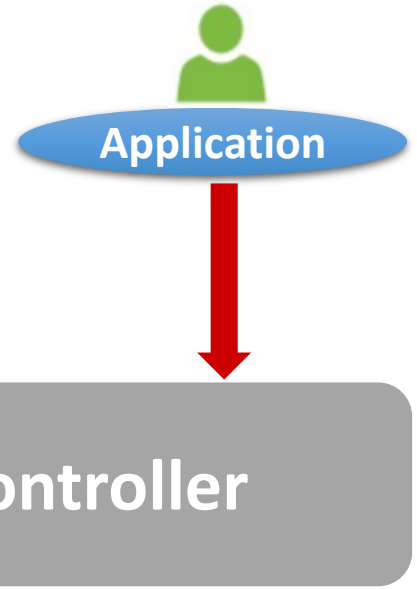


What is
SDN-based Cloud ?

Northbound Interface (NBI)

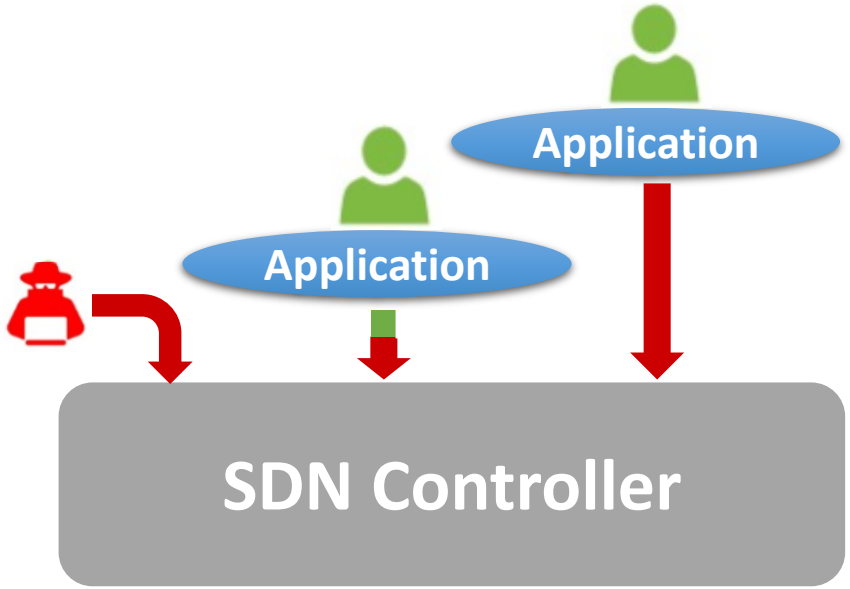
Problem 1: Absence of Effective Access Control

- Inaccurate requests from applications
- Requests are tampered with in transit
- Malicious requests sent through NBI directly

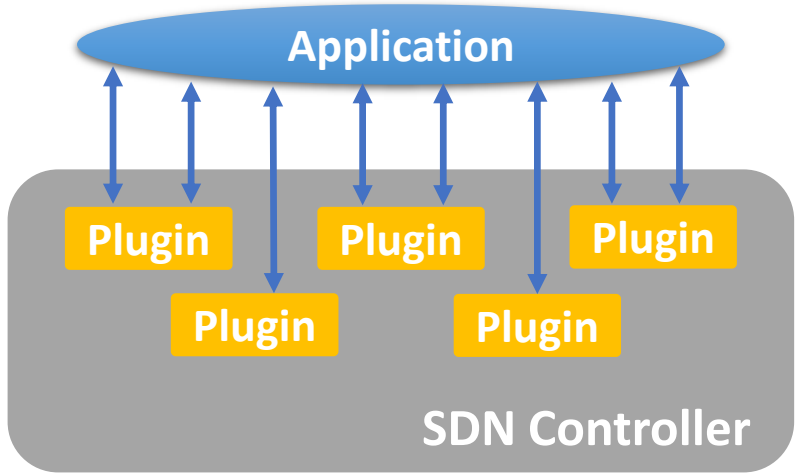


Problem 1: Absence of Effective Access Control

- Inaccurate requests from applications
- Requests are tampered with in transit
- Malicious requests sent through NBI directly

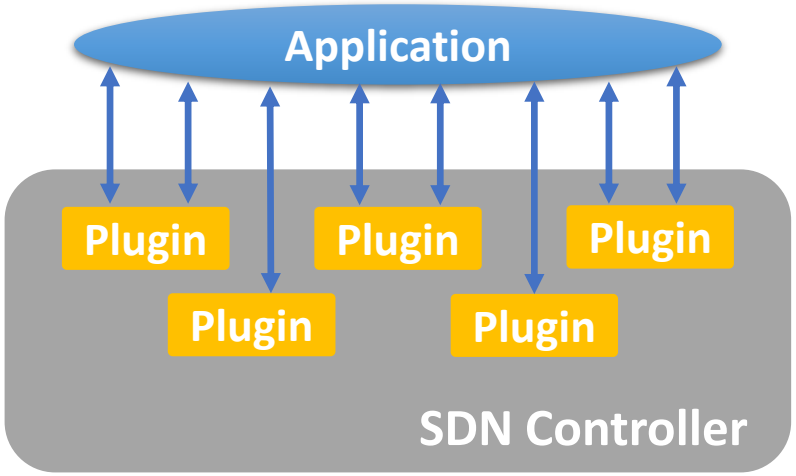


Problem 2: Absence of Unified Management



Problem 2: Absence of Unified Management

- **Inflexible control of resources[1]**
- **Error prone during network configuration**



[1] Resource is anything that can be utilized to provide services in response to client requests.

Current solutions

- **Access control on requests** **[JNSM'18], AAA Project in ODL**
  - Verify the legitimacy of user's identity

  Omit the legitimacy of user's operation, Coarse-grained
- **Reconciliating inside the plugin**
- **Redesigning API and controller architecture**

Current solutions

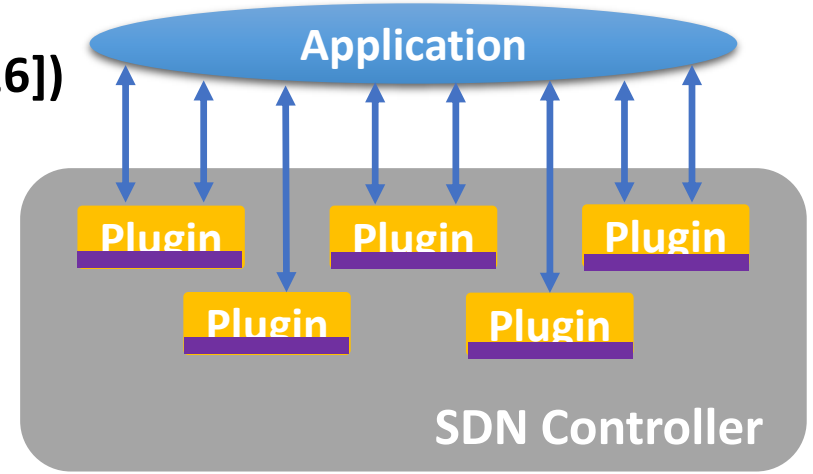
- **Access control on requests** [JNSM'18], AAA Project in ODL
  - Verify the legitimacy of user's identity

  Omit the legitimacy of user's operation, Coarse-grained

- **Reconciliating inside the plugin** (SDNShield[DSN'16])

  Code modification, Inflexible

- Redesigning API and controller architecture



Current solutions

- **Access control on requests** [JNSM'18], AAA Project in ODL
  - Verify the legitimacy of user's identity

  Omit the legitimacy of user's operation, Coarse-grained

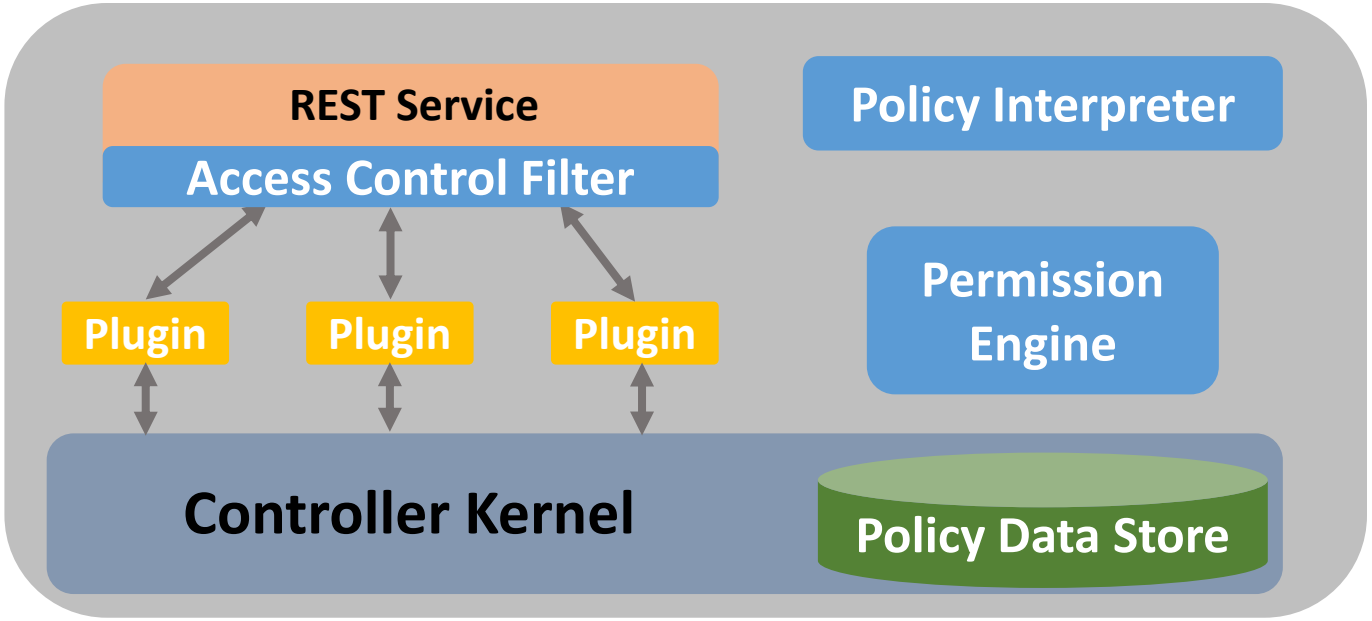
- **Reconciliating inside the plugin** (SDNShield[DSN'16])

  Code modification, Inflexible

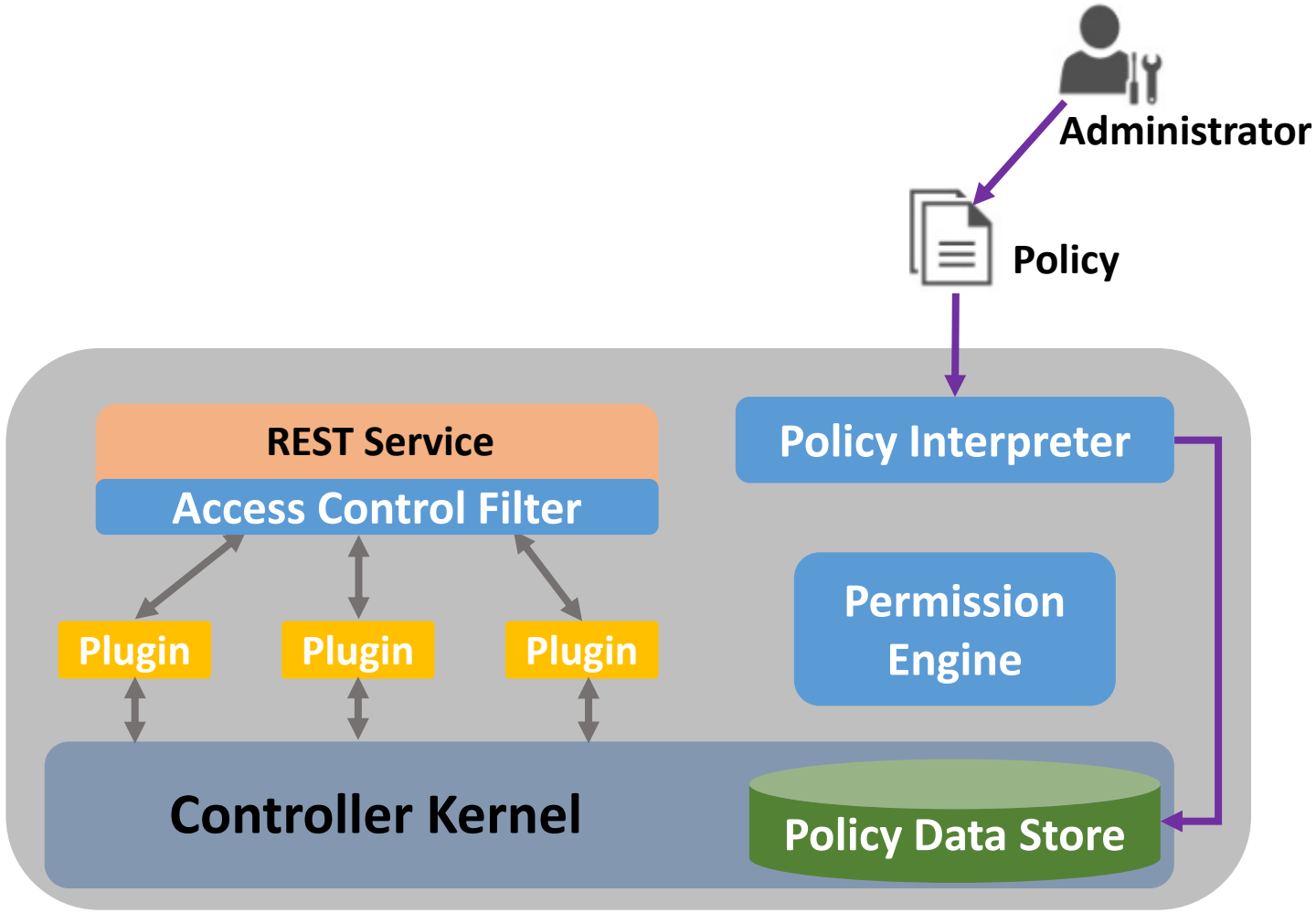
- **Redesigning API and controller architecture** [HotSDN'14, SIGCOMM CCR'13]

  Poor interoperability

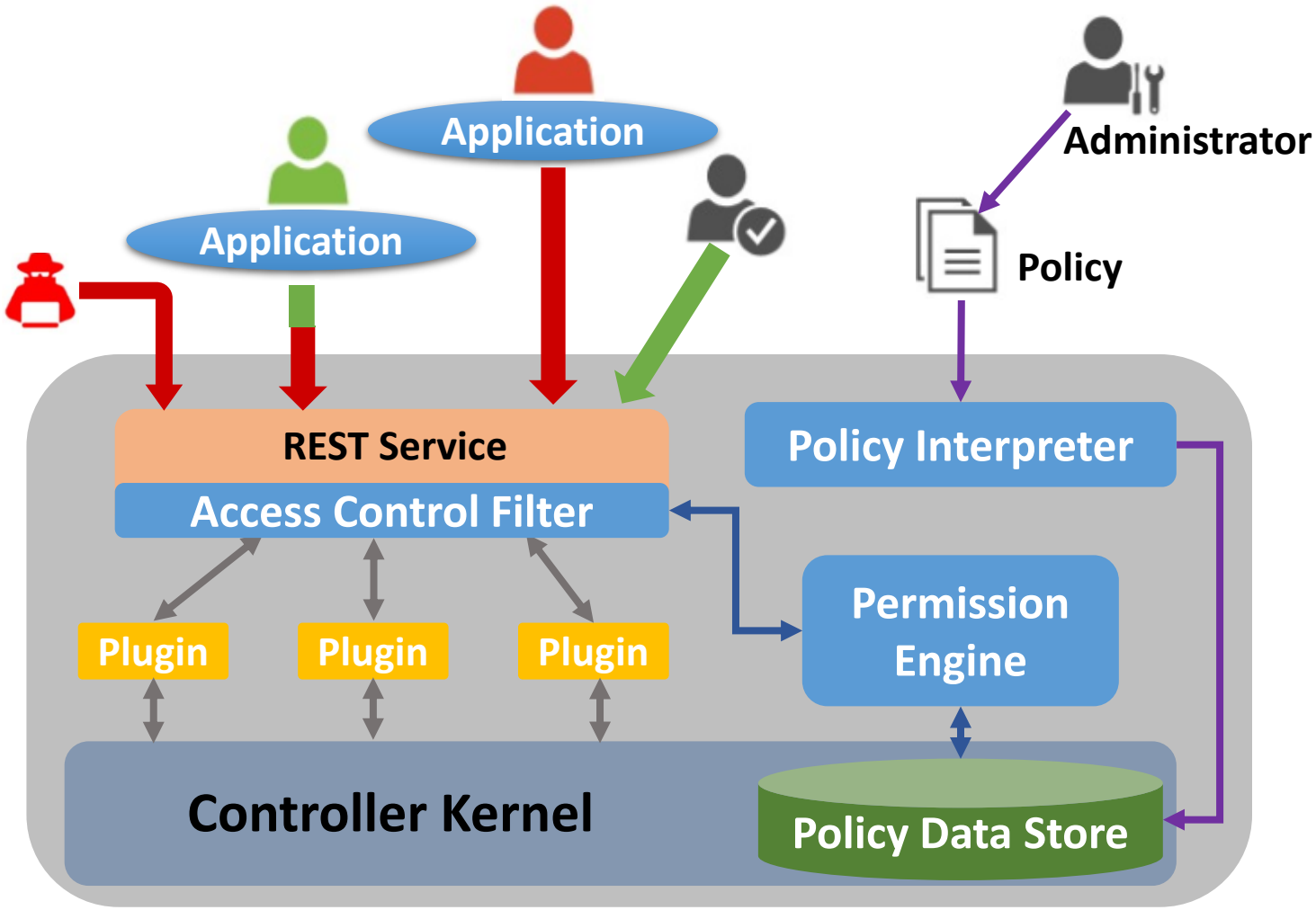
SDNKeeper


REST Service
Access Control Filter
Policy Interpreter
Plugin
Plugin
Plugin
Permission Engine
Controller Kernel
Policy Data Store

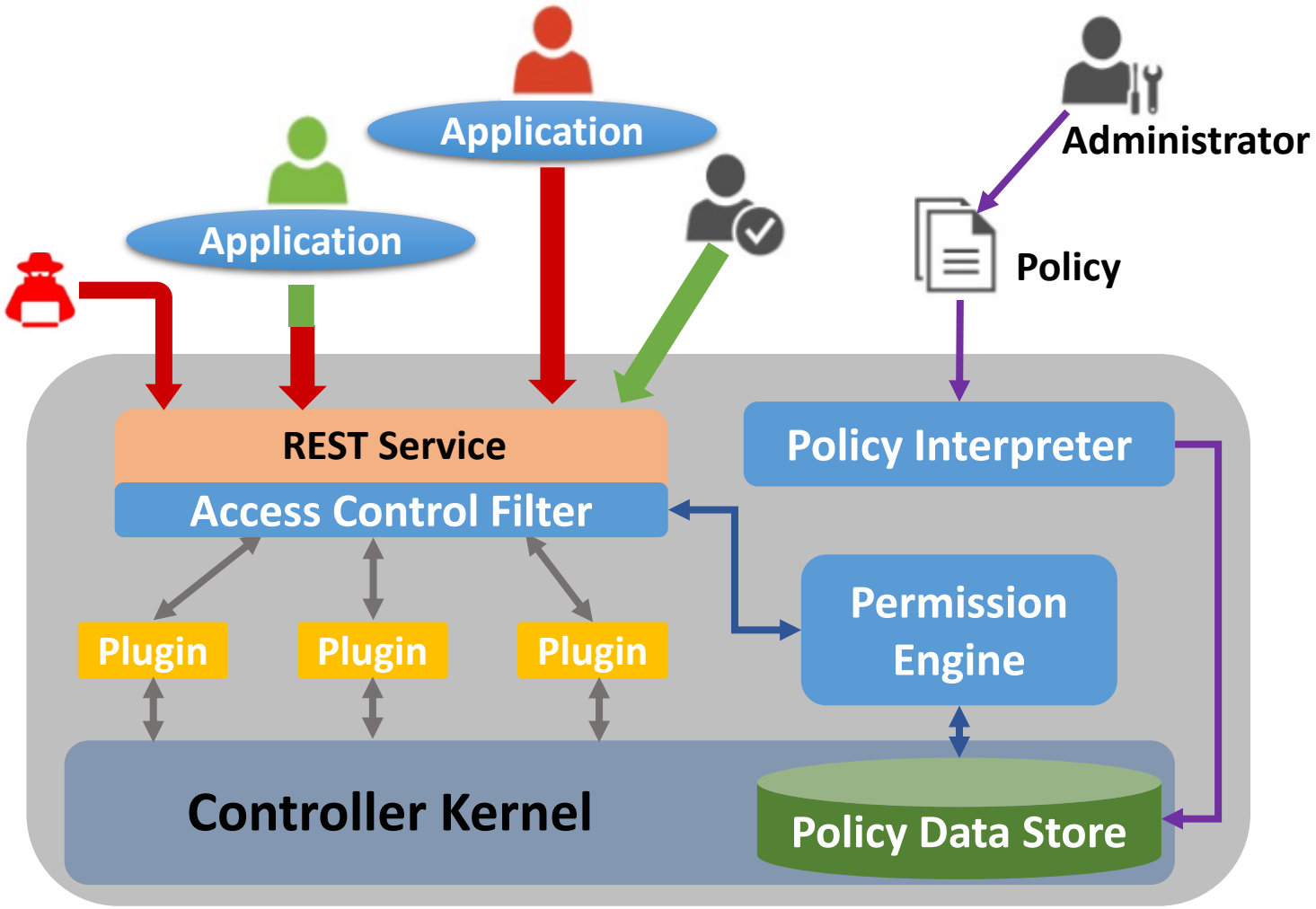
SDNKeeper


Administrator
Policy
REST Service
Access Control Filter
Policy Interpreter
Plugin
Plugin
Plugin
Permission Engine
Controller Kernel
Policy Data Store

SDNKeeper


Application
Application
Administrator
Policy
REST Service
Access Control Filter
Policy Interpreter
Plugin
Plugin
Plugin
Permission Engine
Controller Kernel
Policy Data Store

SDNKeeper



- Applicability
- Administrator Friendliness
- Centralized Management
- Hot Update

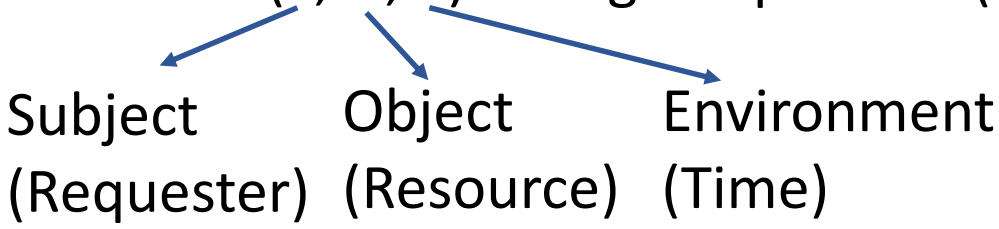
Detailed Designs

- **Policy Language – flexible permission abstractions**
- **Policy Interpreter – parsing semantic policies**
- **Permission Engine – performing access control on requests**

Policy Language

Attribute Based Access Control

P(S, O, E) <- Logic Expression(ATTR(S), ATTR(O), ATTR(E))



REST Request

```
1) Method: POST  (POST/GET/PUT/DELETE)
2) URI: https://<controller-ip>:<port>/networks/
3) Headers: {
     Content-Type : application/json,
     Authorization : {
        Username : Alice, Password : *** },
     ... }
4) Body (optional): {
     network : {
        name : alice-network,
        tenant_id : 9bacb3c5d39d41a7951...,
        subnets : [],
        network_type : vlan,
        ... }}
```

Policy Language

Attribute Based Access Control

P(S, O, E) <- Logic Expression(ATTR(S), ATTR(O), ATTR(E))

Policy: A set of assertion expressions
Composition: Iteration of *if-statements* and logical operators
Return: ACCEPT / REJECT

Policy

```
GLOBAL_POLICY {
  system_update {
    if (environment.time > 1am &&
        environment.time <  6am ) {
      REJECT }}}
LOCAL_POLICY {
  user {
    user_can_get_on_monday {
      if (action.method == 'GET') {
        if (environment.weekday == 'mon') {
          ACCEPT }}}}
  user.Alice {
    alice_cannot_delete_firewall {
      if (action.uri REG '/firewalls/') {
        if (action.method == 'DELETE') {
          REJECT }
        else {
          ACCEPT }}}
      ... }}
```

Global Policy

for ↳ all requests

Local Policy

for ↳ individual user group and user

Policy

```
GLOBAL_POLICY {
  system_update {
    if (environment.time > 1am &&
        environment.time <  6am ) {
      REJECT }}}
LOCAL_POLICY {
  user {
    user_can_get_on_monday {
      if (action.method == 'GET') {
        if (environment.weekday == 'mon') {
          ACCEPT }}}}
  user.Alice {
    alice_cannot_delete_firewall {
      if (action.uri REG '/firewalls/') {
        if (action.method == 'DELETE') {
          REJECT }
        else {
          ACCEPT }}}
      ... }}
```

Global Policy

for → all requests

Local Policy

for → individual user group and user

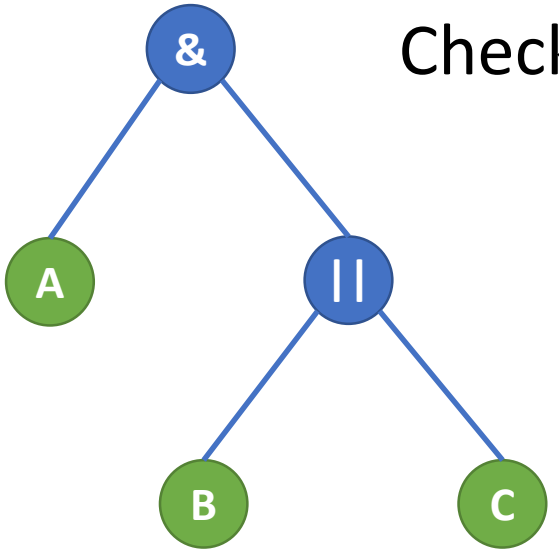
Performance

Expressiveness and simplicity

Detailed Designs

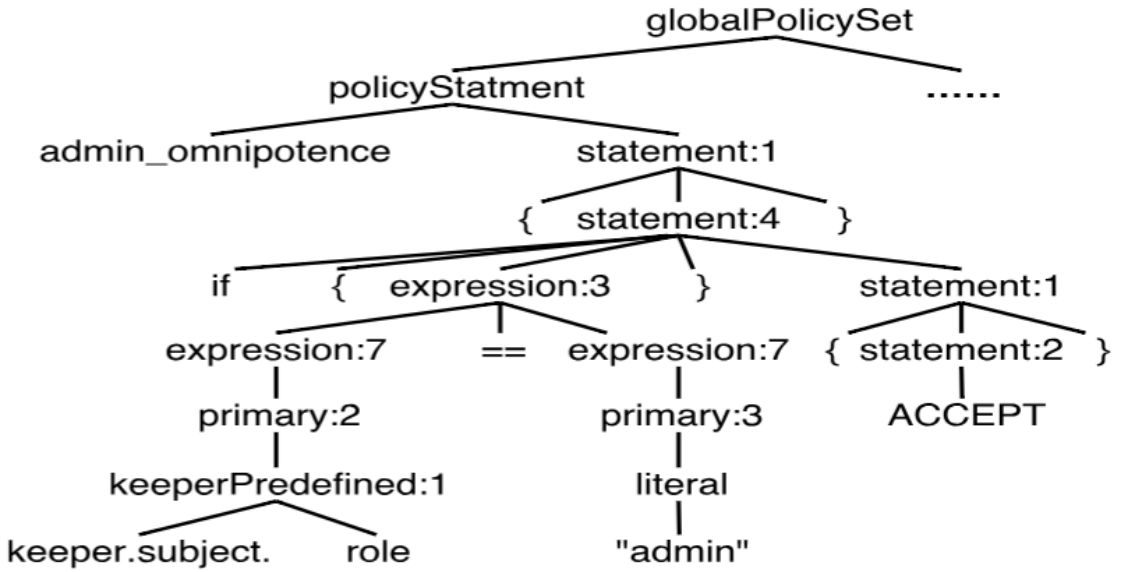
- **Policy Language – flexible permission abstractions**
- **Policy Interpreter – parsing semantic policies**
- **Permission Engine – performing access control on requests**

Policy Interpreter



Checking result

Logical operators



An attribute or a comparing value

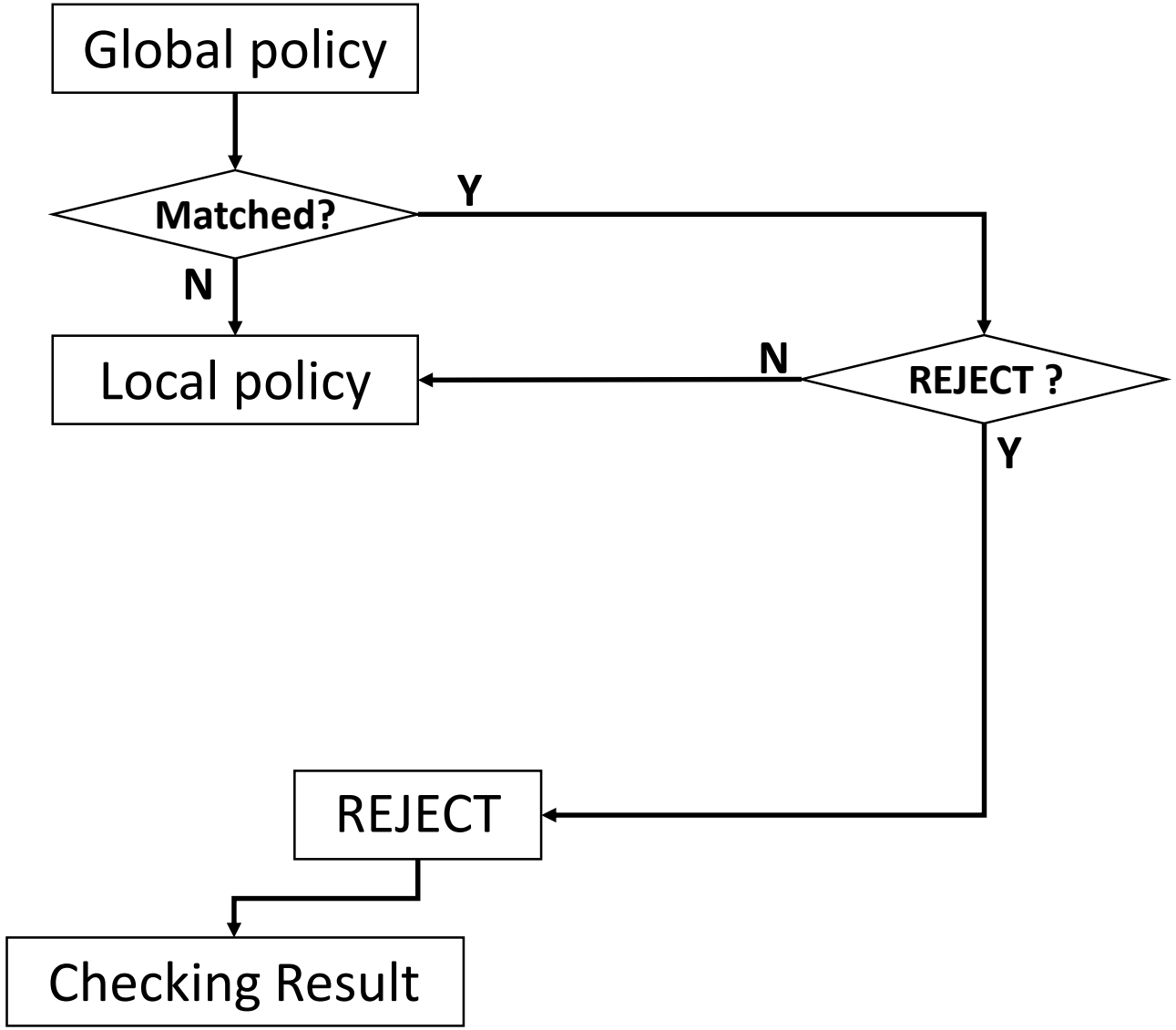
(A && (B || C))

Detailed Designs

- **Policy Language – flexible permission abstractions**
- **Policy Interpreter – parsing semantic policies**
- **Permission Engine – performing access control on requests**

Permission Engine

Step 1: Checking with Global Policies

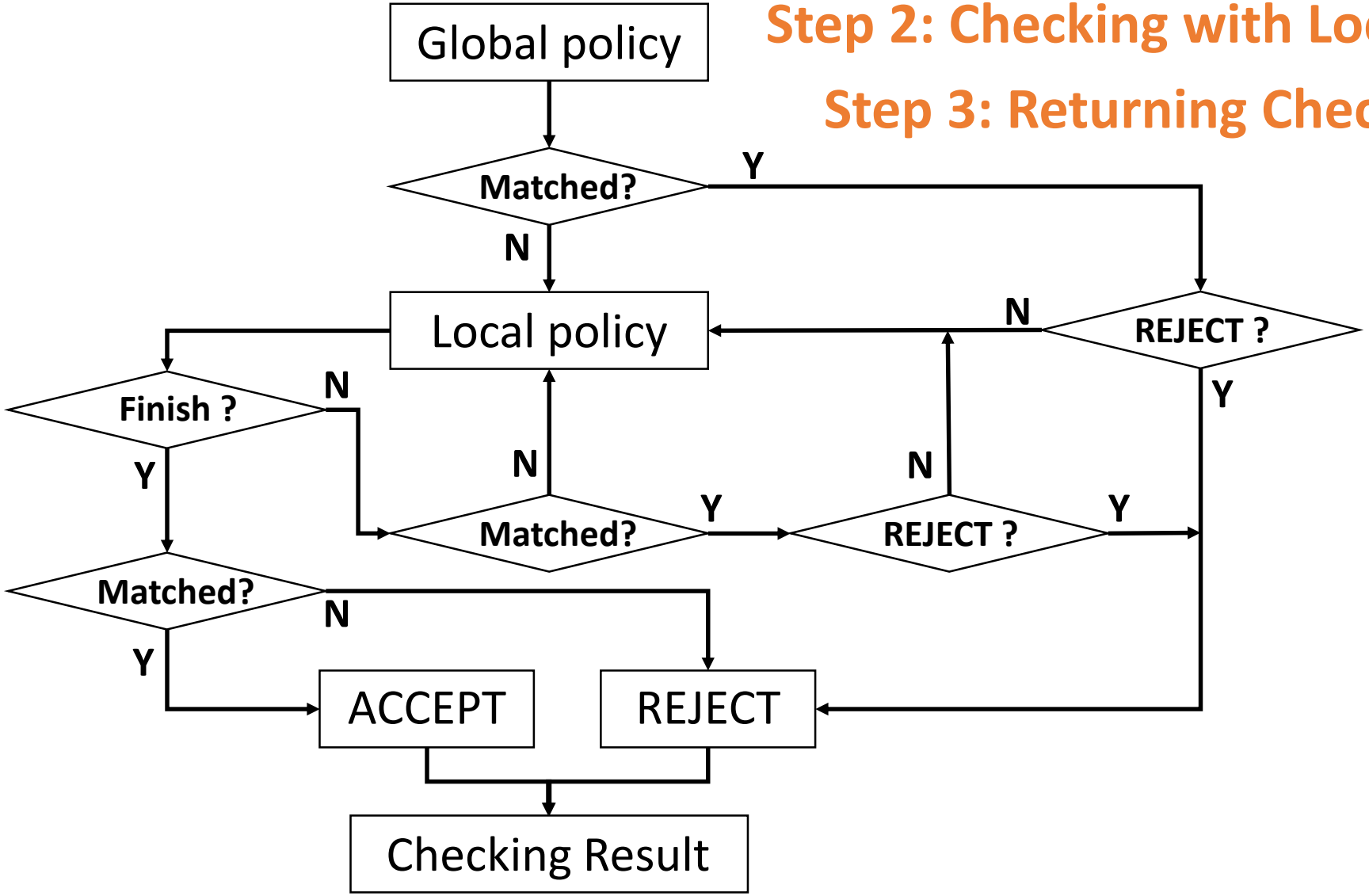


Permission Engine

Step 1: Checking with Global Policies

Step 2: Checking with Local Policies

Step 3: Returning Checking Result



Implementation

- **Filter-based, independent bundle**
  - Realizing the system on OpenDaylight controller
  - No modification is required to the controller and applications
- **Support for dynamic management**
  - CLI command: ***SDNKeeper: load/cache***

Effectiveness

Type	# API	# Attribute	Type	# API	# Attribute
Networking	6	220	Meter	2	13
Firewall	3	83	QoS	2	31
Security	2	24	Load Balance	2	81
VPN	4	104	BGP VPN	1	22
SFC	4	60	L2 Gateway	2	26

2789 policies

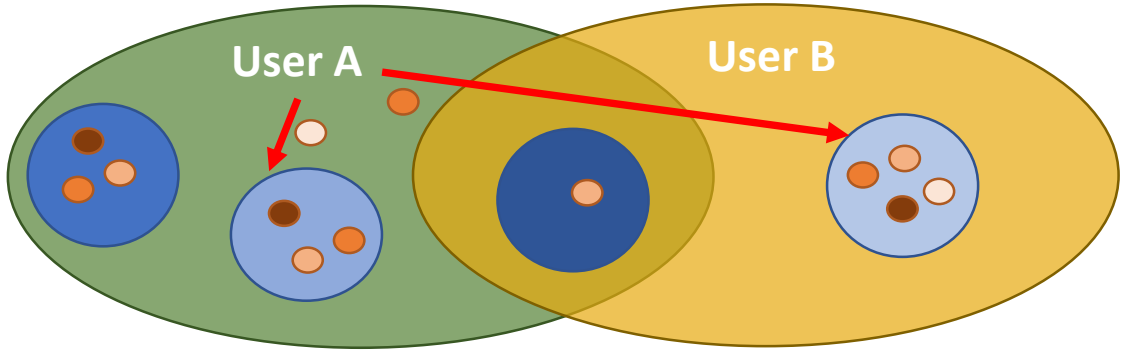
- **30 policies – all kinds of APIs**
- **185 policies – all kinds of actions in API**
- **664 policies – all kinds of attributes**
- **1910 policies – all possible combinations of two attributes**

2789 illegal requests

Effectiveness

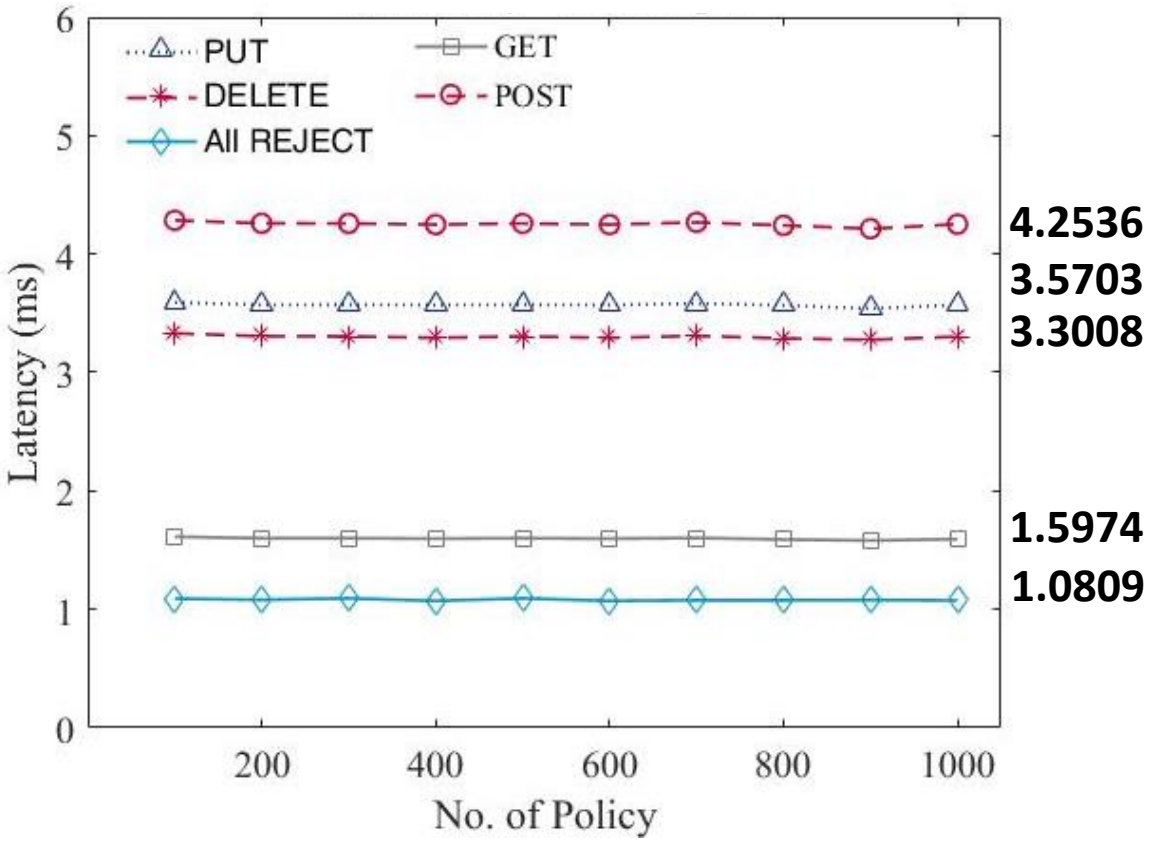
Type	# API	# Attribute	Type	# API	# Attribute
Networking	6	220	Meter	2	13
Firewall	3	83	QoS	2	31
Security	2	24	Load Balance	2	81
VPN	4	104	BGP VPN	1	22
SFC	4	60	L2 Gateway	2	26

2789 illegal requests

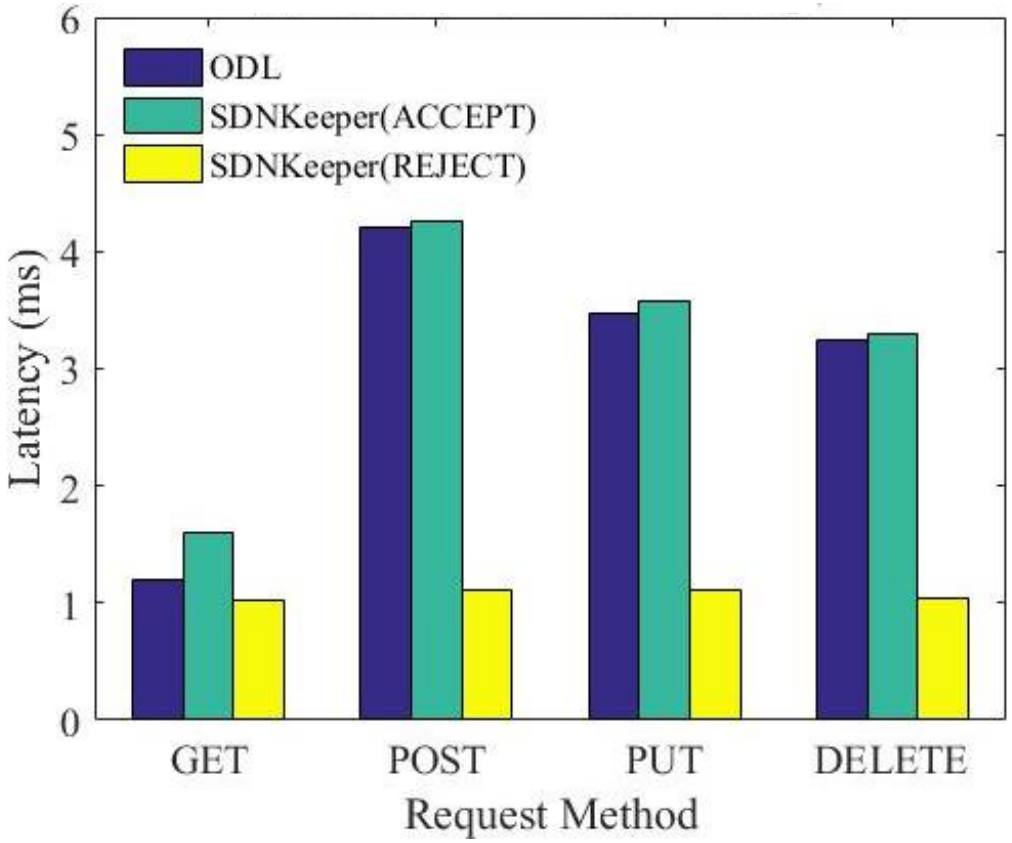


Processing Delay

Latency - SDNKeeper



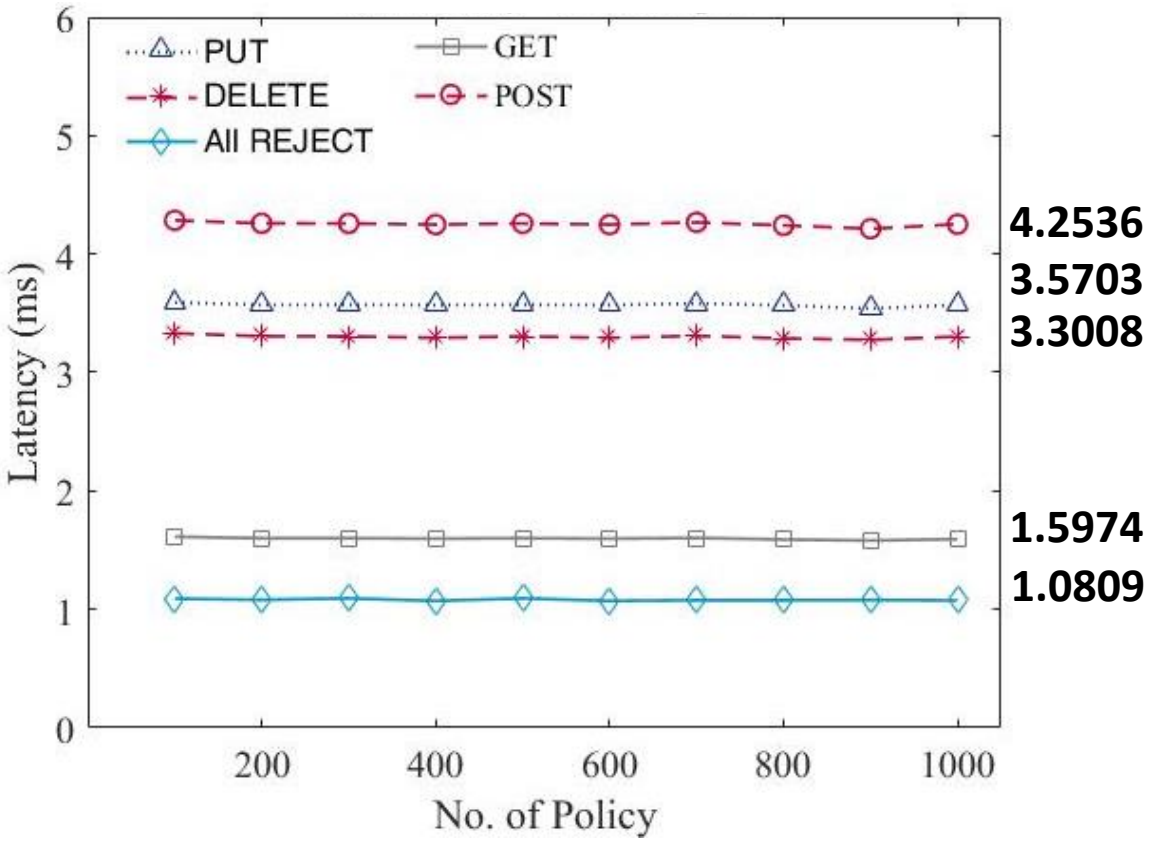
Latency – SDNKeeper VS OpenDaylight



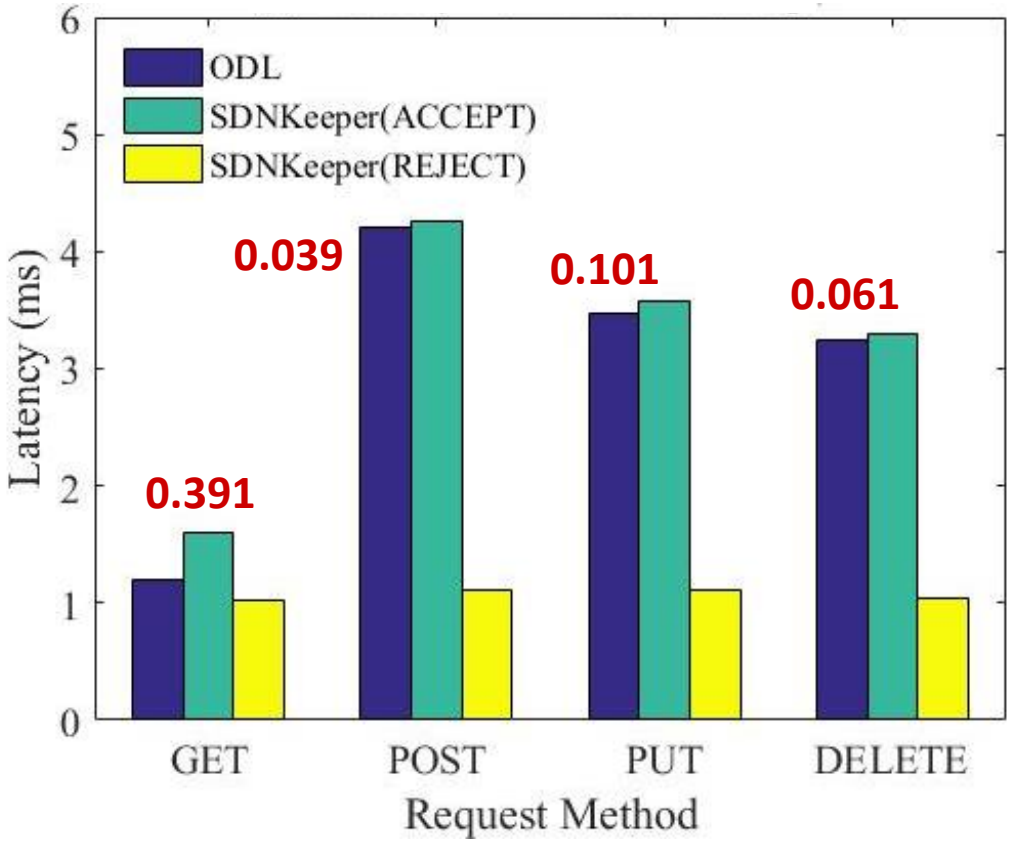
No significant increase in latency

Processing Delay

Latency - SDNKeeper



Latency – SDNKeeper VS OpenDaylight

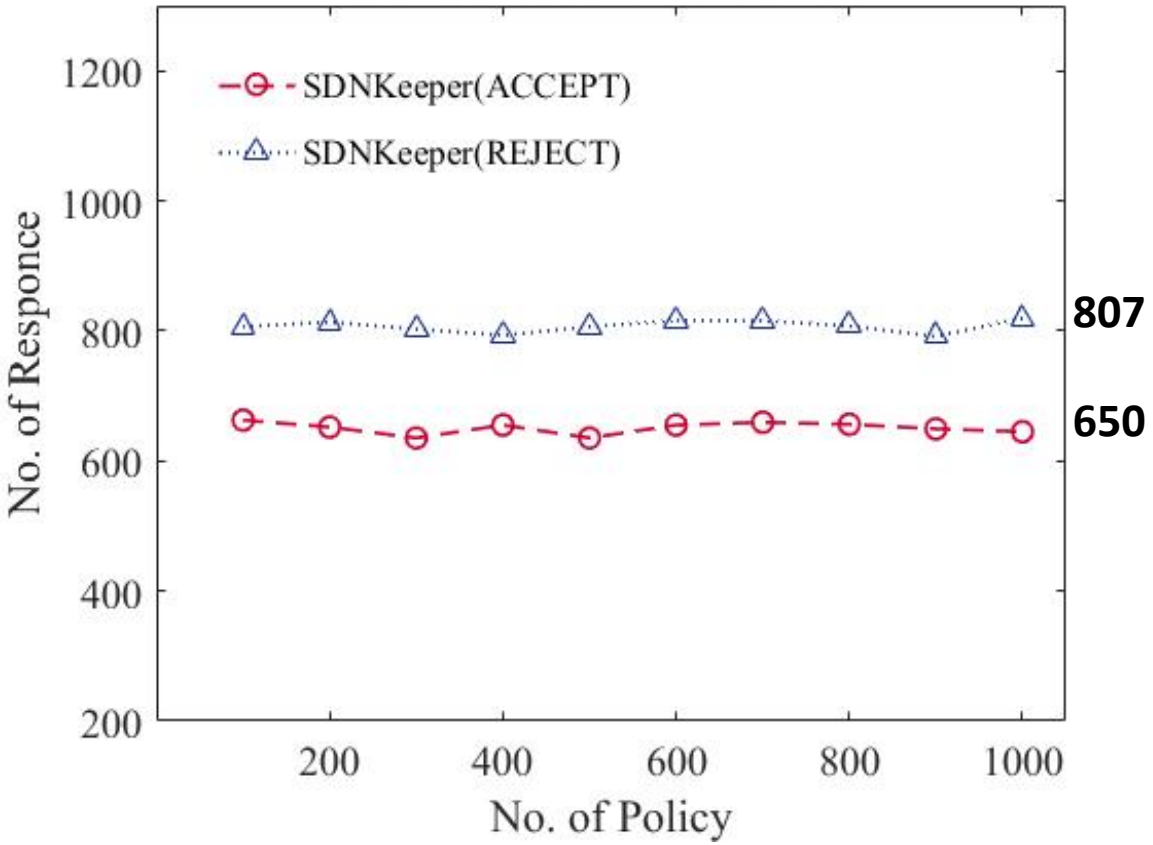


No significant increase in latency

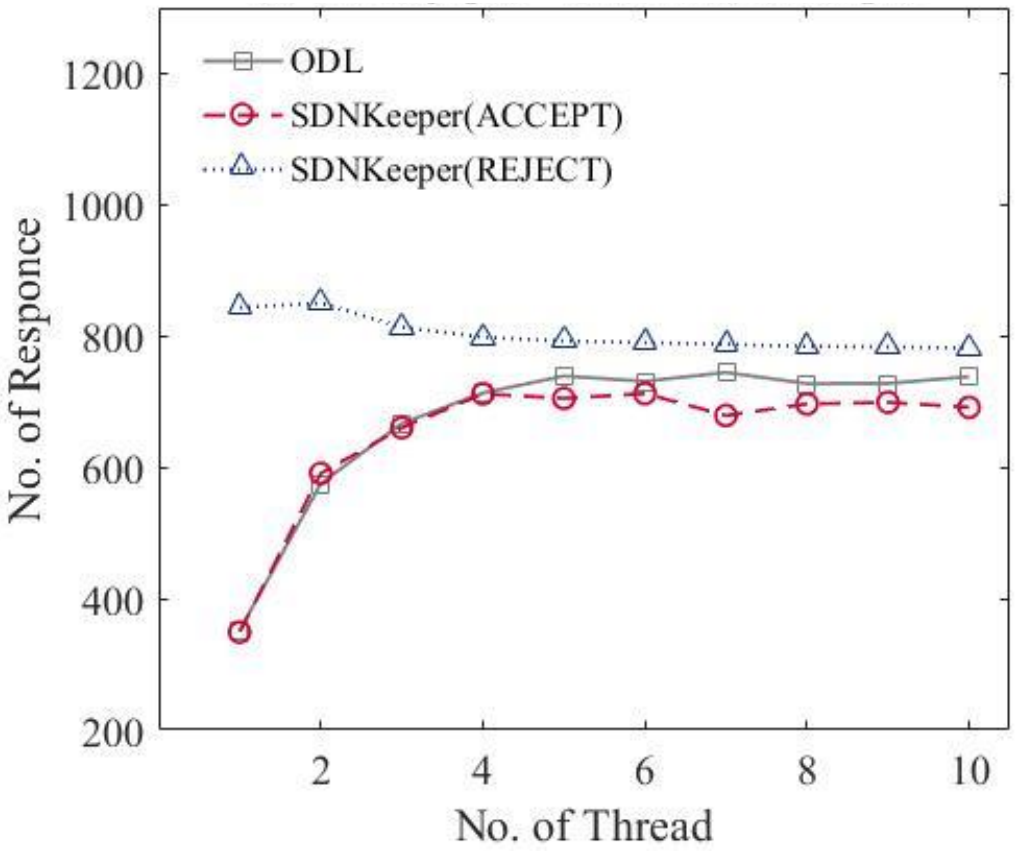
An average delay of 0.15ms

Throughput

Throughput - SDNKeeper



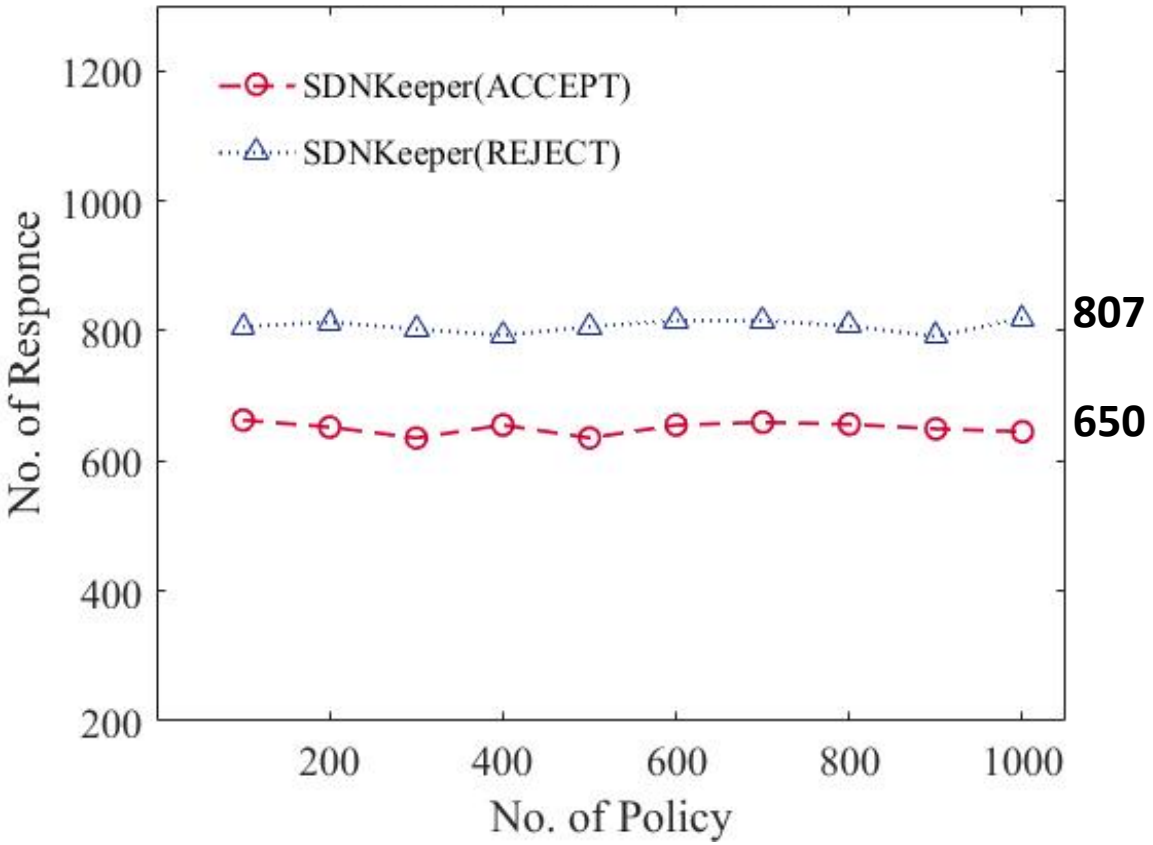
Throughput – SDNKeeper VS OpenDaylight



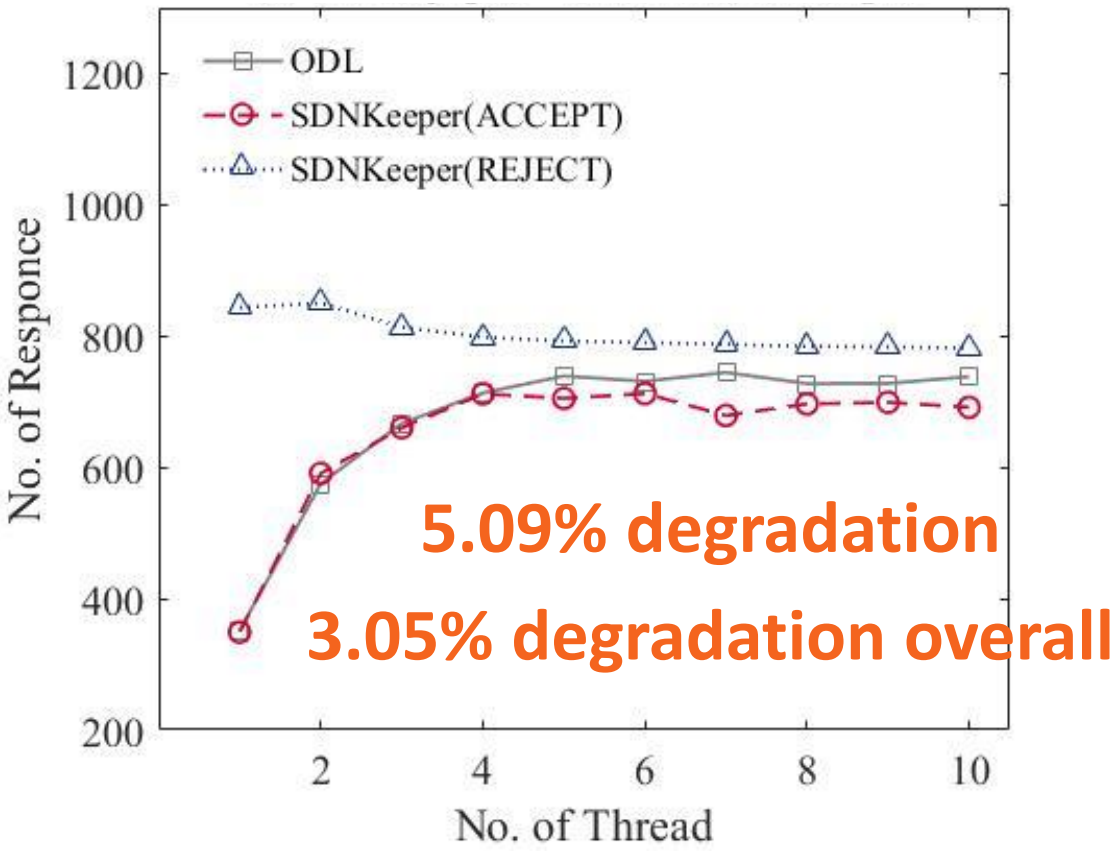
No significant effect in throughput

Throughput

Throughput - SDNKeeper



Throughput – SDNKeeper VS OpenDaylight



No significant effect in throughput

Conclusions

- **SDNKeeper: a lightweight access control system**
  - Defending against malicious requests
  - Assisting in managing resources
  - Real-time protection and policy hot-update
- **Reliable enforcement with good performance**

Thank you

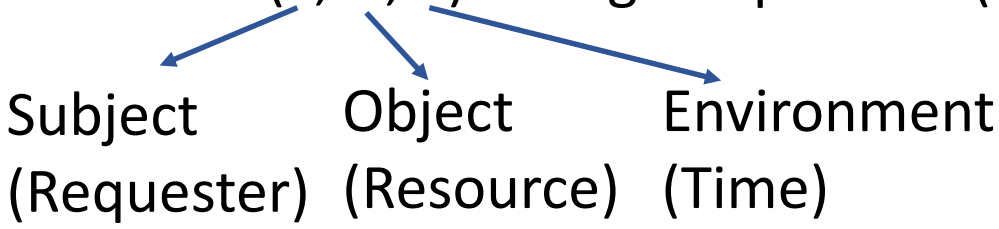
lengxue_2015@outlook.com

Back Up Page

Policy Language

Attribute Based Access Control

P(S, O, E) <- Logic Expression(ATTR(S), ATTR(O), ATTR(E))



Predefined Data Structure

REST Request

```
1) Method: POST   (POST/GET/PUT/DELETE)
2) URI: https://<controller-ip>:<port>/networks/
3) Headers: {
     Content-Type : application/json,
     Authorization : {
        Username : Alice, Password : *** },
     ... }
4) Body (optional): {
     network : {
        name : alice-network,
        tenant_id : 9bacb3c5d39d41a7951...,
        subnets : [],
        network_type : vlan,
        ... }}
```

Policy Language

Attribute Based Access Control

P(S, O, E) <- Logic Expression(ATTR(S), ATTR(O), ATTR(E))

Subject (Requester)

Object (Resource)

Environment (Time)

Predefined Data Structure

subject.role *subject.user*

action.uri *action.method*

$.{object_name}.attribute

$.network.network_type

REST Request

```
1) Method: POST  (POST/GET/PUT/DELETE)
2) URI: https://<controller-ip>:<port>/networks/
3) Headers: {
     Content-Type : application/json,
     Authorization : {
        Username : Alice, Password : *** },
     ... }
4) Body (optional): {
     network : {
        name : alice-network,
        tenant_id : 9bacb3c5d39d41a7951...,
        subnets : [],
        network_type : vlan,
        ... }}
```

Policy

```
policy    : policy_name '{' statement '}'
statement : 'ACCEPT' |  'REJECT' | if_state
if_state  : 'if (' expr ')' statement
            ('else' statement)?
```

Policy: A set of assertion expressions

Composition: Iteration of *if-statements* and logical operators

Return: ACCEPT / REJECT

```
alice_cannot_delete_firewall {
  if (action.uri REG '/firewalls/') {
    if (action.method == 'DELETE') {
      REJECT }
    else {
      ACCEPT }}}
```

```
system_update {
  if (environment.time > 1am &&
      environment.time <  6am ) {
    REJECT }}}
```