大连理工大学
DALIAN UNIVERSITY OF TECHNOLOGY

# SDNS

## Centrally Coordinated Replica Selection Architecture in Multi-controller SDN

**SDNS**
**集中协调的SDN多控制器副本选择架构**

# Agenda

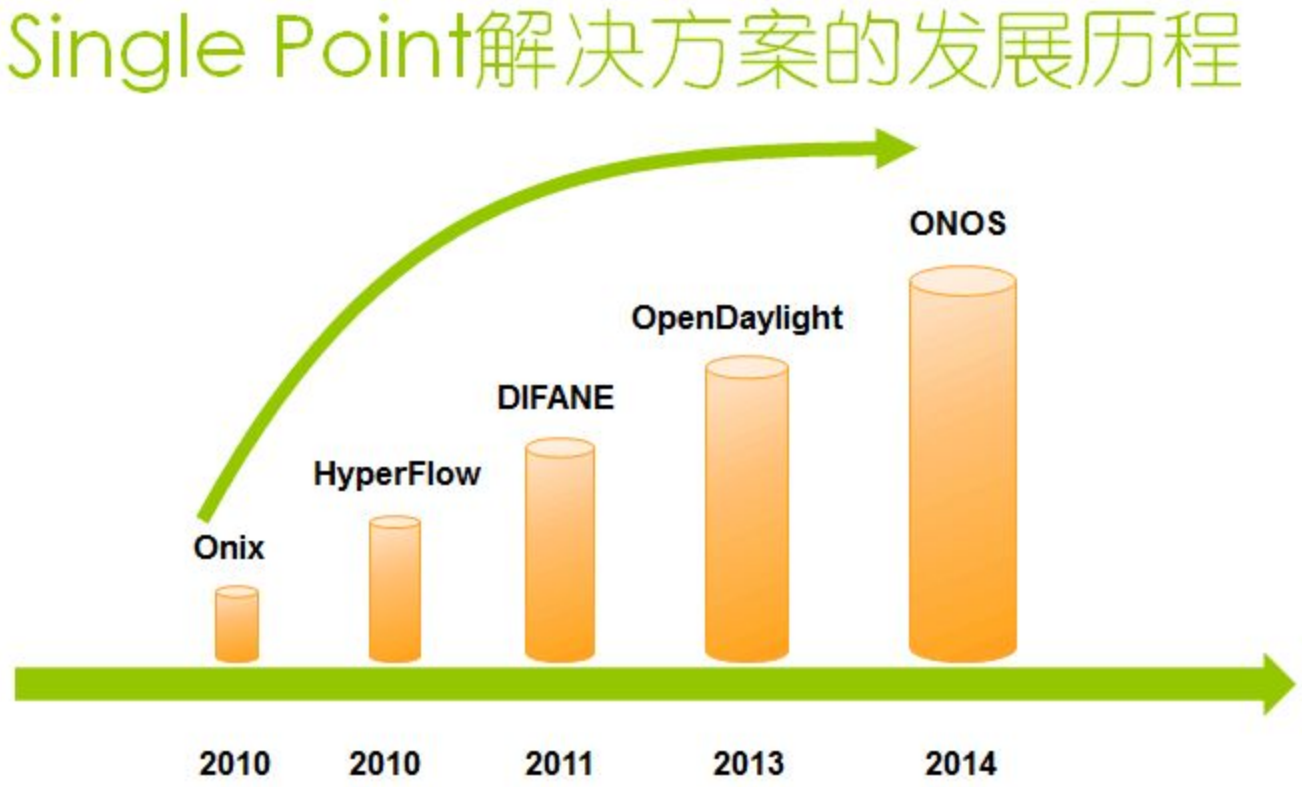
1. 在SDN管控分布式服务上的思考
2. Single Point解决方案的发展历程
3. SDNS架构的简介
4. SDNS架构的特性
5. SDNS架构的工作机制
6. Pentacore调度算法的简介
7. Coordinator调度器的实现
8. SDNS架构的性能评价
9. SDNS展望

# 在SDN管控分布式服务上的思考

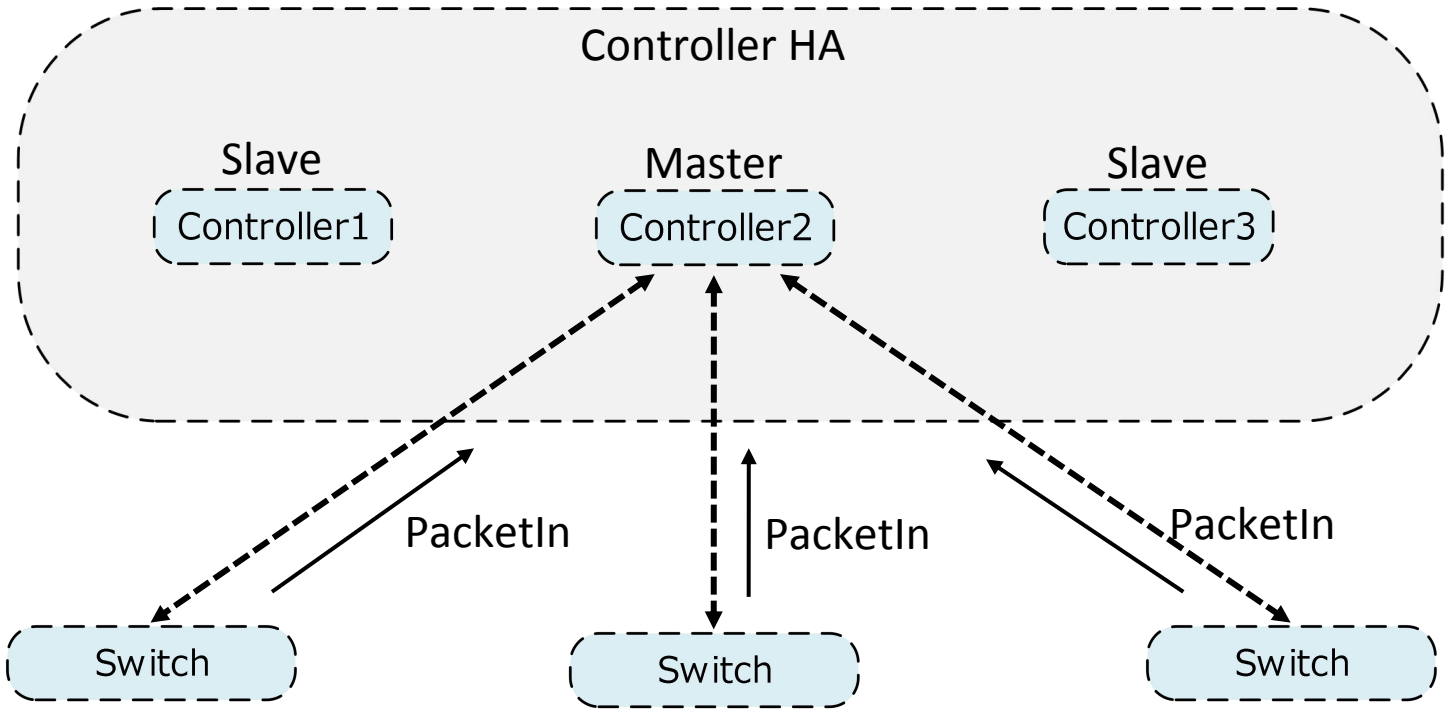
- SDN ---- 数据控制分离，灵活的管控
- 单Controller处理PacketIN ---- Single Point
  - 缺乏Process Load Balance
  - Fault Tolerance成为挑战

- 若所有Task交由单Controller处理，当Controller的处理处于Delay或Congestion时，有可能完全中断所有Task的处理，Switch将无法正常工作。

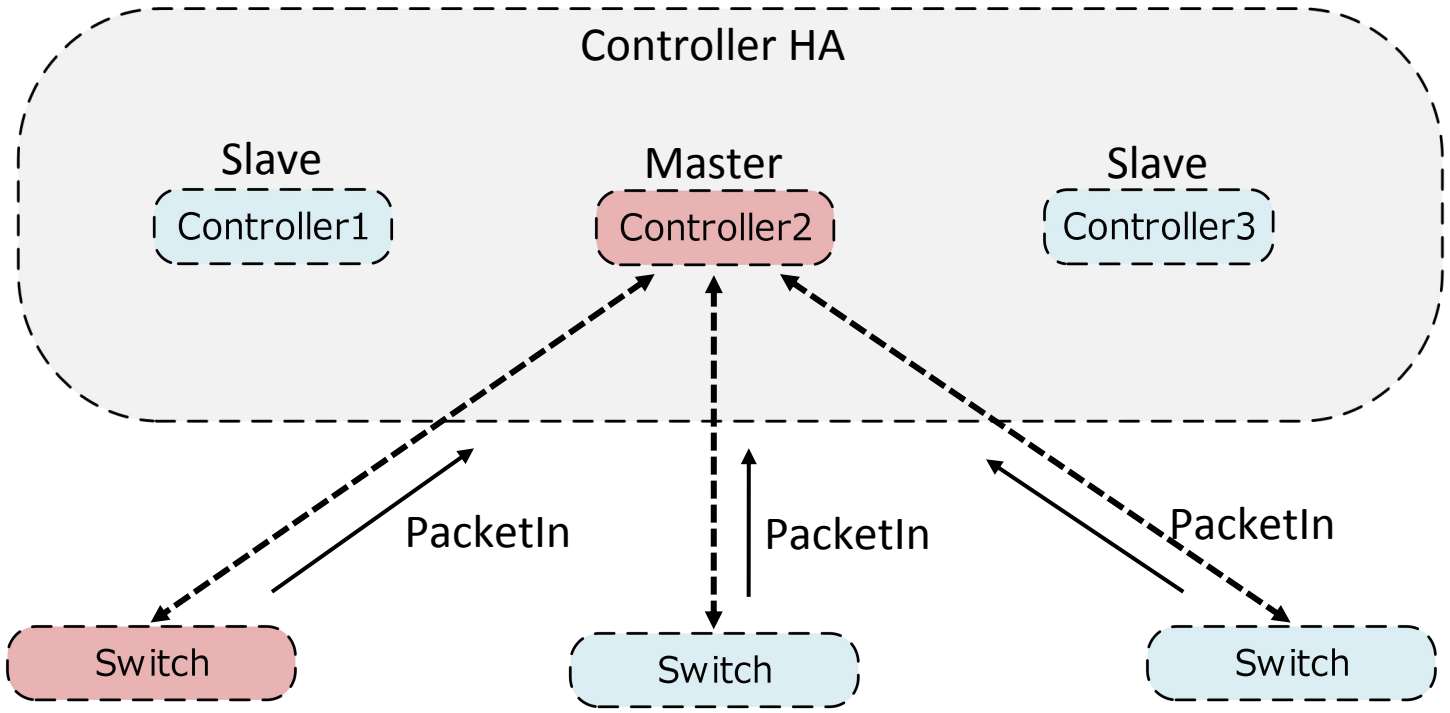
# Single Point解决方案的发展历程



# Onos--WideArea LoadBalance

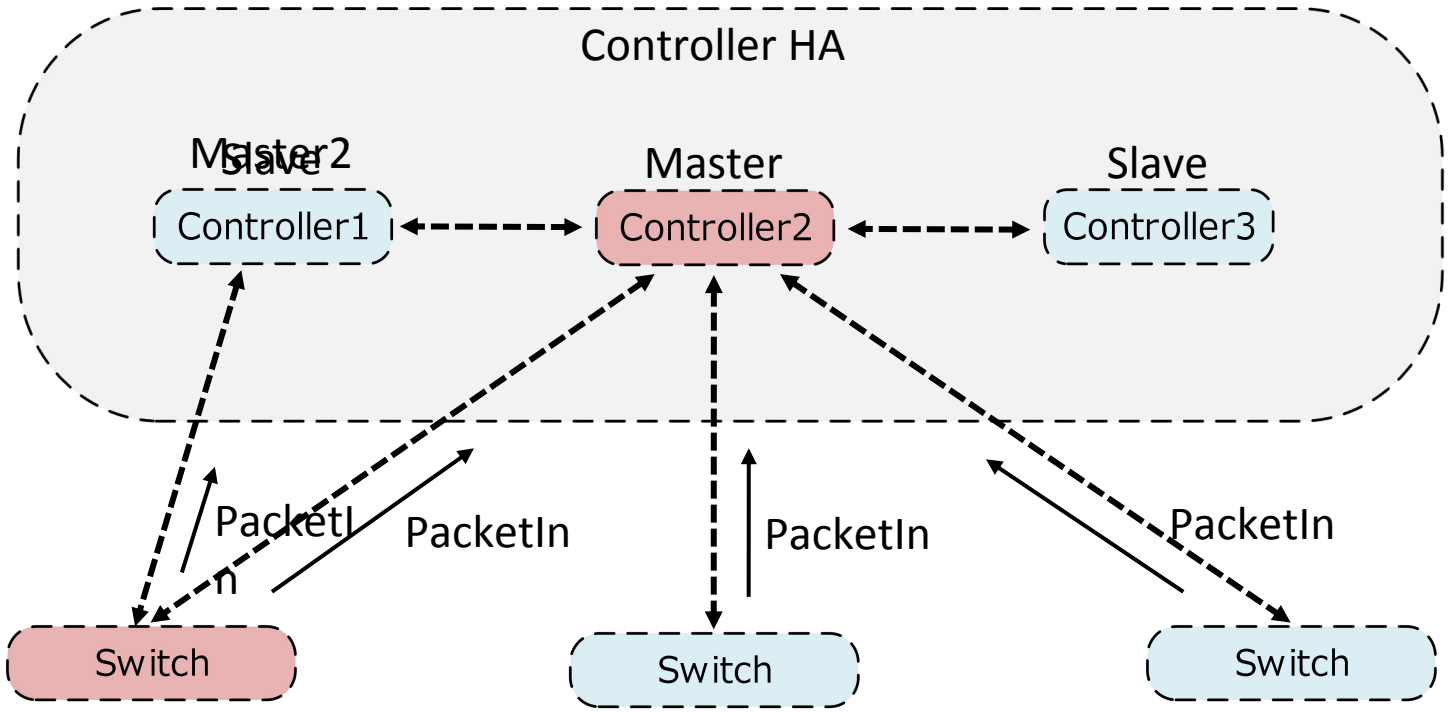


# Onos--WideArea LoadBalance



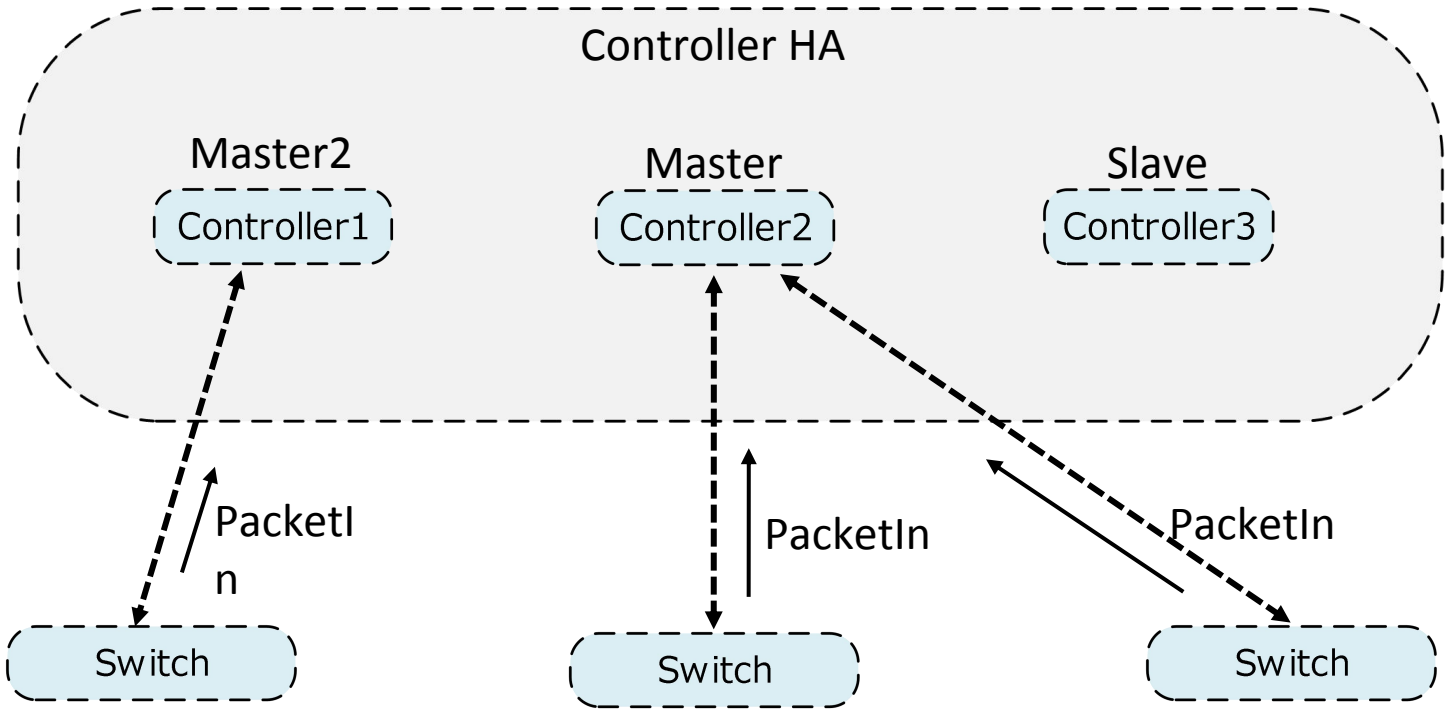

负载加重

# Onos--WideArea LoadBalance

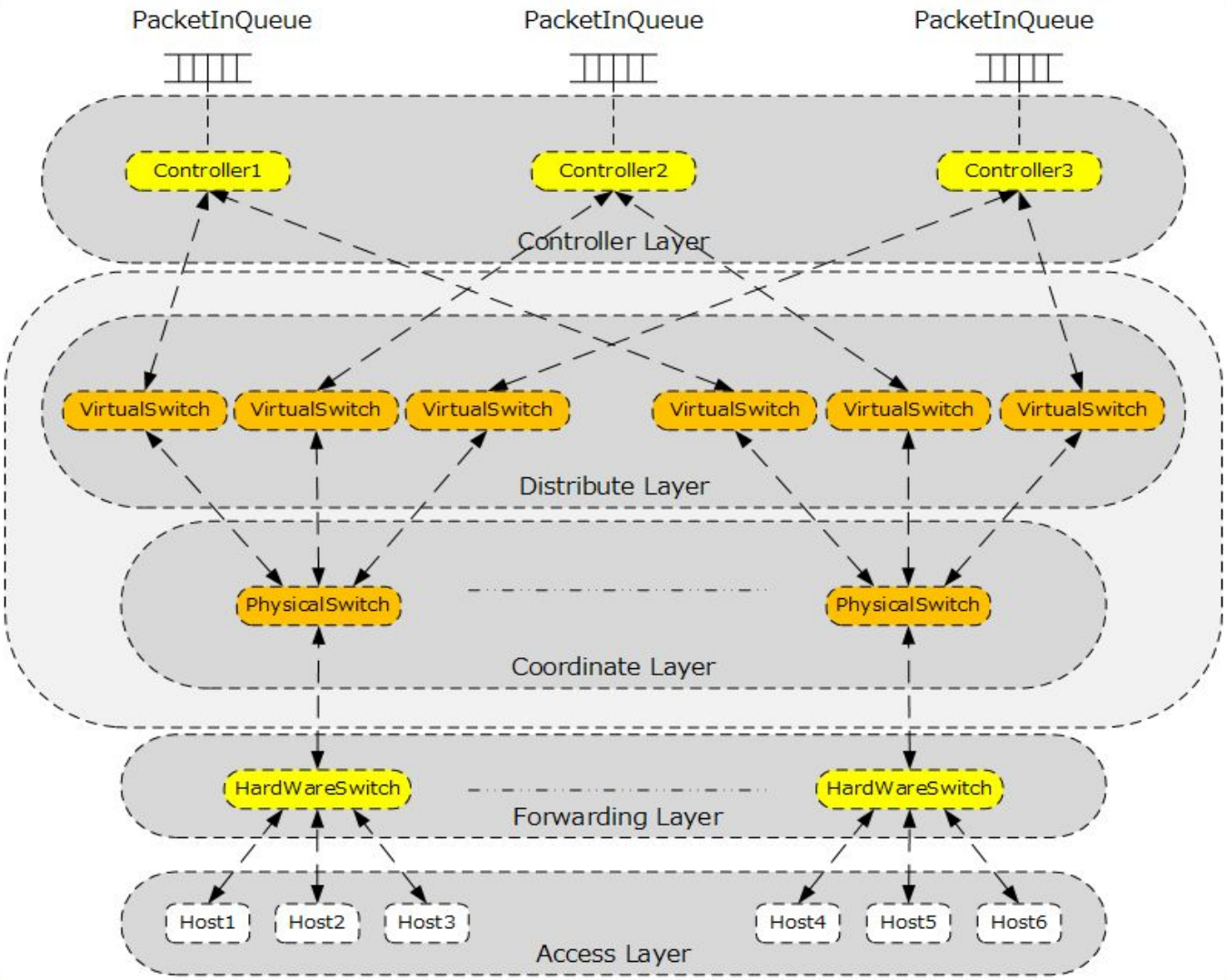



运行分布式选举算法 ……

# Onos--WideArea LoadBalance



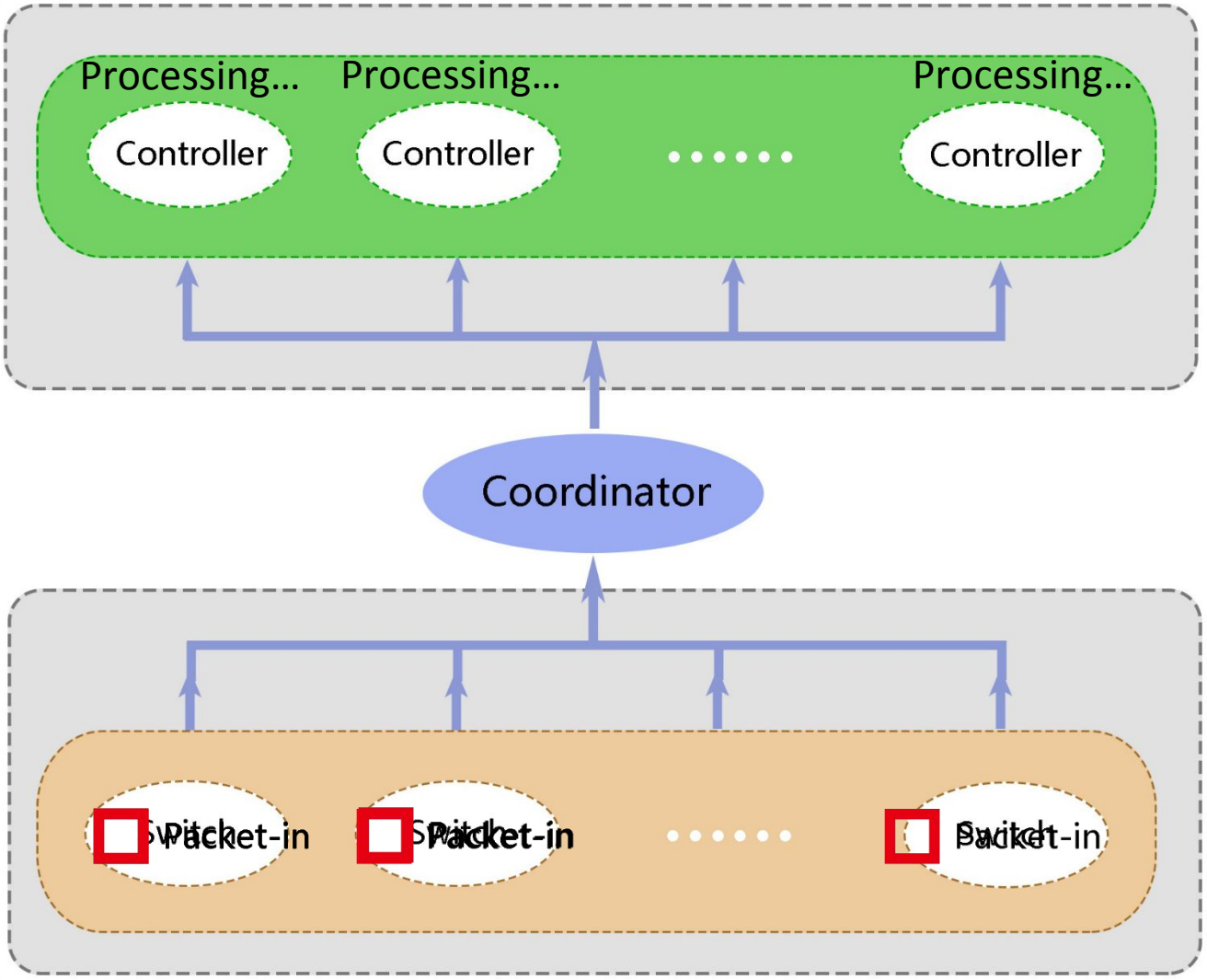
# SDNS架构的简介


PacketInQueue
PacketInQueue
PacketInQueue
Controller1
Controller2
Controller3
Controller Layer
VirtualSwitch
VirtualSwitch
VirtualSwitch
VirtualSwitch
VirtualSwitch
VirtualSwitch
Distribute Layer
PhysicalSwitch
PhysicalSwitch
Coordinate Layer
HardWareSwitch
HardWareSwitch
Forwarding Layer
Host1
Host2
Host3
Host4
Host5
Host6
Access Layer

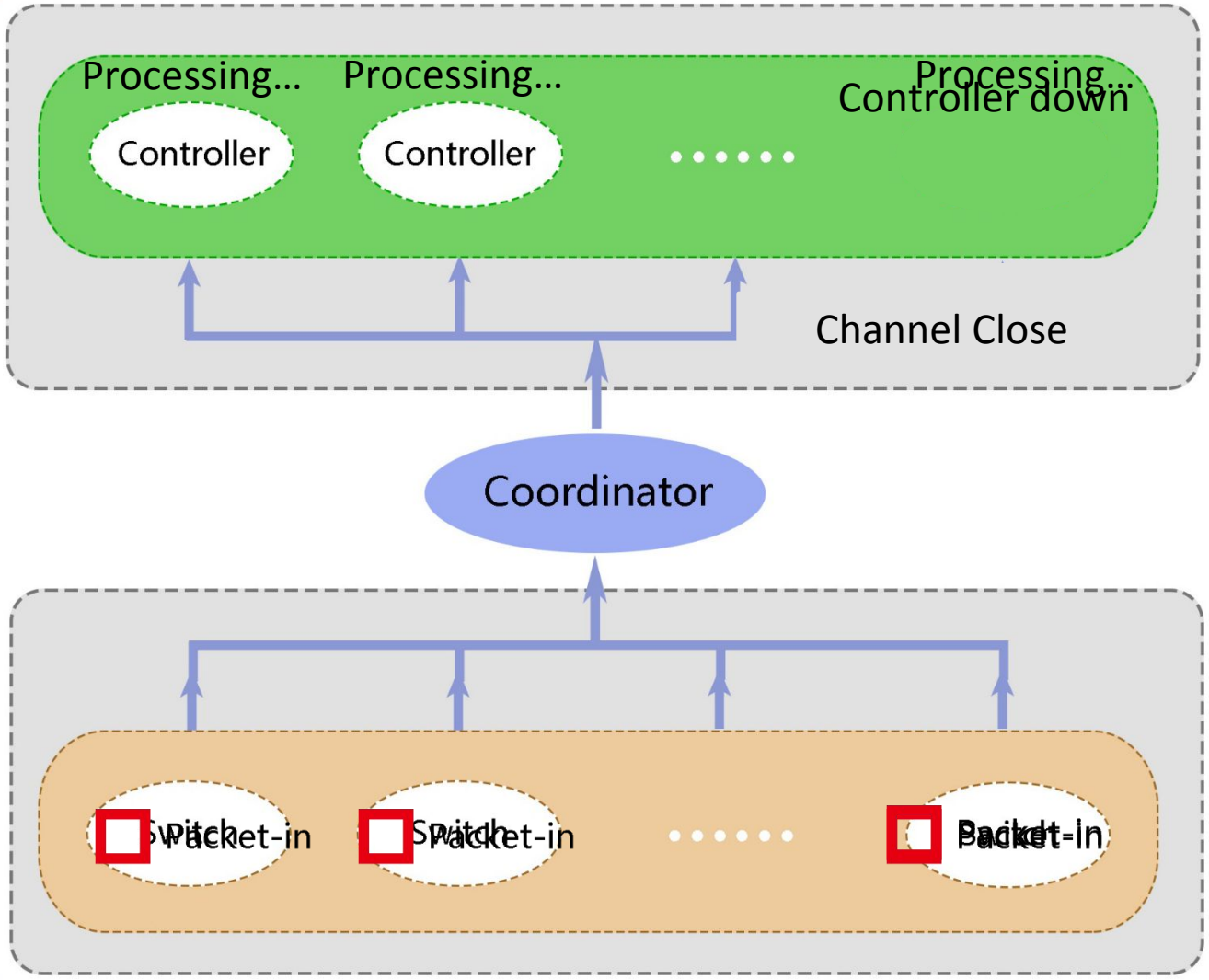
# SDNS架构的特性

- Application Domain
  - Controller Load Balancing
  - Maximum Reliability
  - Minimum Response Time
  - Service Chain
- Scalability

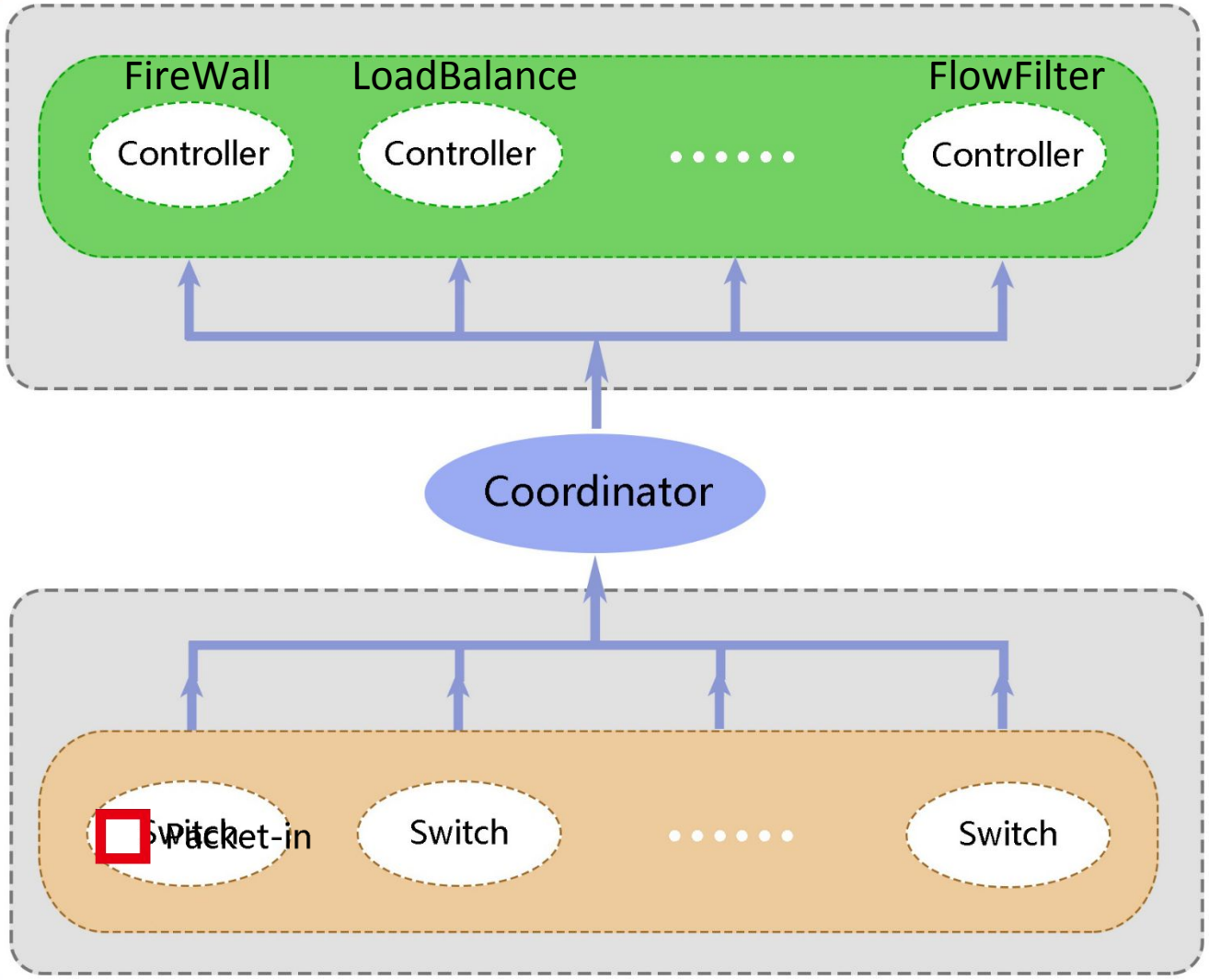
# Load Balancing



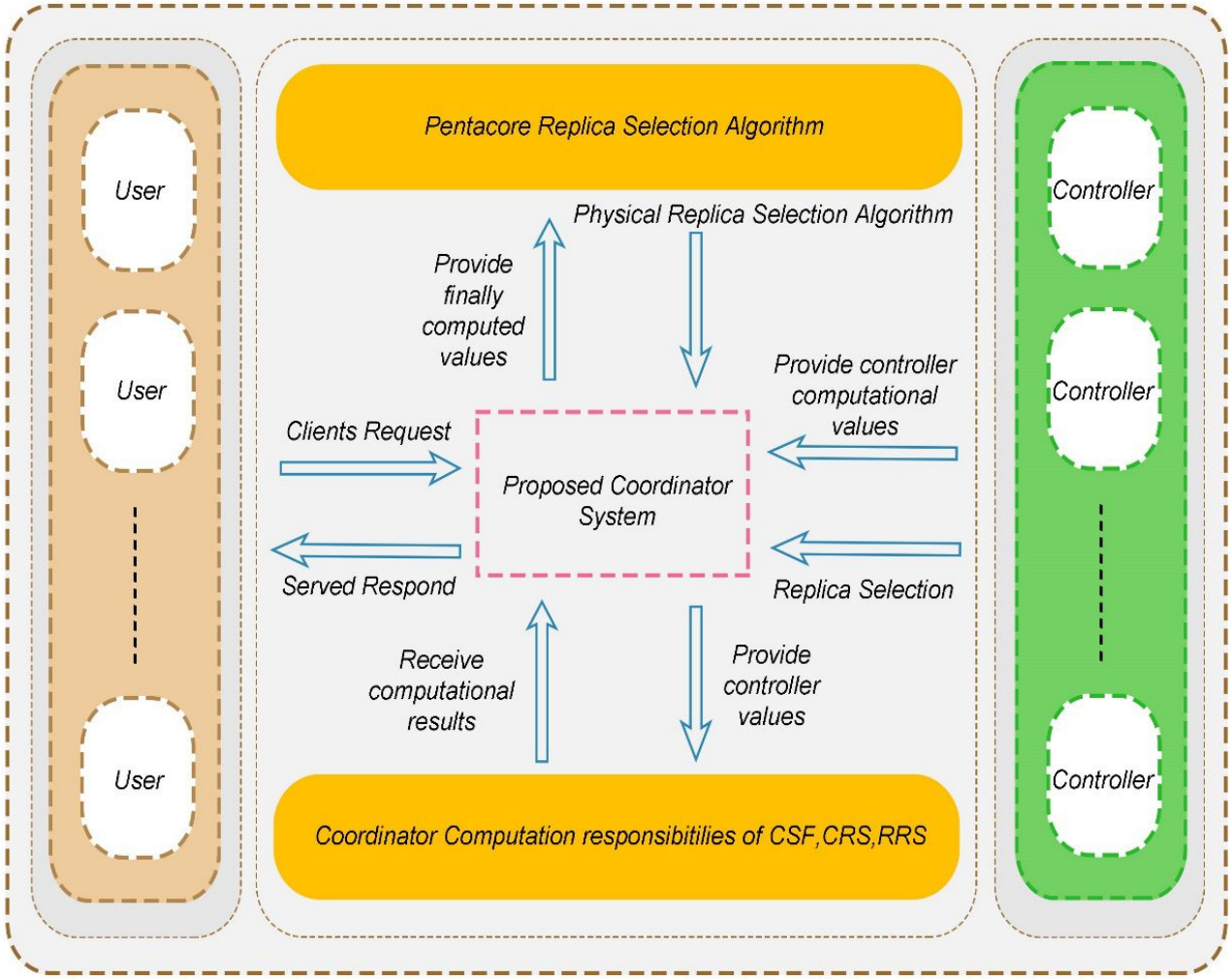
# Maximum Reliability



# Service Chain



# SDNS架构的工作机制



# Pentacore调度算法的简介

- Step 1 Random Selection
- Step 2 Power of Two choice
- Step 3 Batch-Sampling
- Step 4 Batch-Filling
- Step 5 Replica Ranking

**REQUIRE:** v=<*i*; *Server Load* > //set of replicas and their corresponding ServerLoad, ServerLoad is a set which includes response time *rt* and packetin queue length *pq*

**REQUIRE:** Client Inputs:

$Req_m$//request list, *m* is the size of the list

*sendList* ← ∅

*m* ← sizeof(*V*)

*selectedList* ← 2**m* randomly selected member from *V*

*sortedList* ← sort *selectedList* in decreasing order of ServerLoad

**while** (*m* > 0) **do**

K ← $[first(sortedList)]$

L ← $[second(sortedList)]$

$sortedList \leftarrow sortedList - \mathrm{K}$

$prod \leftarrow (rt_l * pq_l - rt_k * pq_k)/rt_l$

$sendList \leftarrow Req[0, (prod - 1)]$

$leftList \leftarrow Req[prod, (m - 1)]$

$m \leftarrow (m - prod)$

send($sendList$, K)

**end while**

# Coordinator调度器的实现

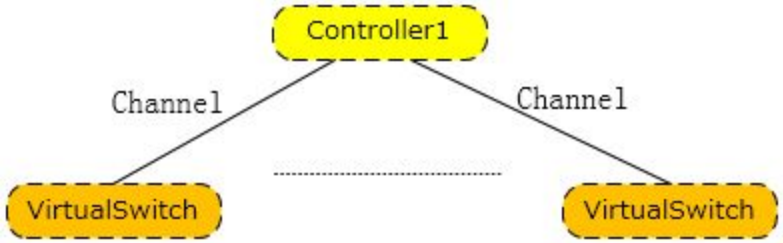
## Based on

–OpenVirtex : a network virtualization platform.

## Three Main parts

### –Contact

**1. True South(Controller Layer)**: 每个Controller都和真实交换机个数的VirtualSwitch对象通过多线程形式建立各自Channel通讯，所有线程由ClientBootStrap管理。

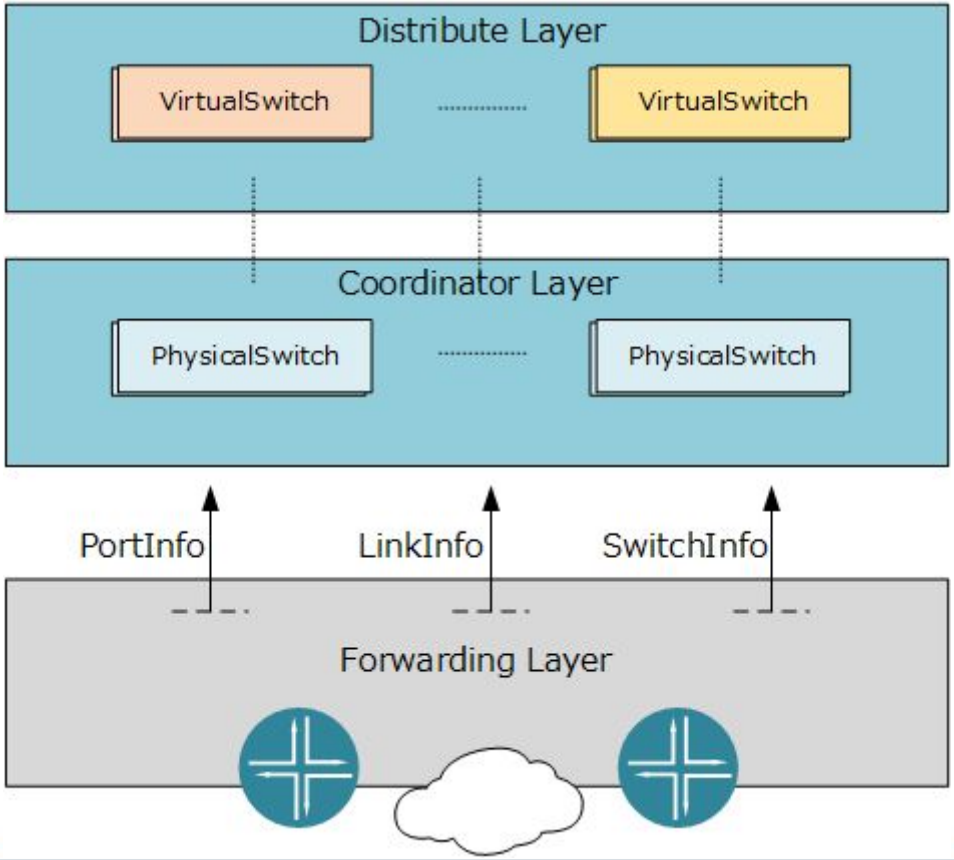



**2. True North(Forwarding Layer)**: 每个HardWareSwitch都和一个PhysicalSwitch对象通过多线程的形式建立各自的channel通讯，所有线程由SwitchBootStrap管理。

# Coordinator调度器的实现

## –Contact

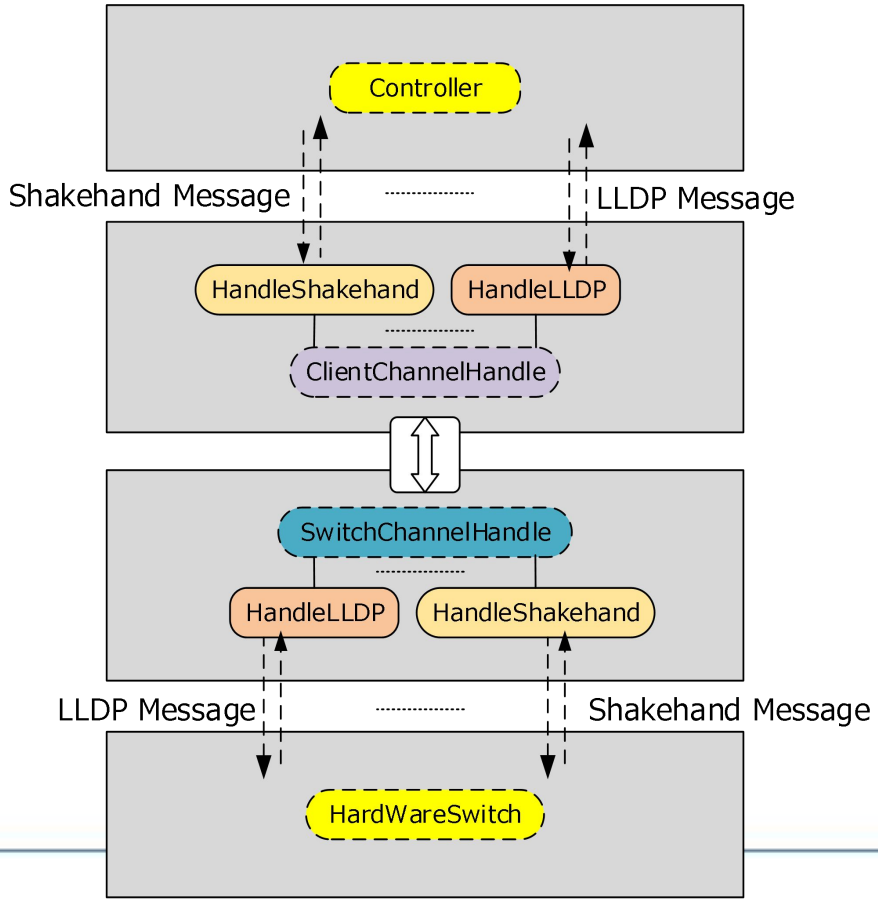
**3. Internal(Coordinator Layer)**: 每个VirtualSwitch都和它对应的PhysicalSwitch一一对应起来。同时Coordinator内部保存了从下层交换机获取来的PortInfo、LinkInfo、SwitchInfo等信息以及他们之间的联系并存储在内存中。



# Coordinator调度器的实现

## –Message Handling

**1. Pretender Handling :** 南向伪装成Controller，北向伪装成Switch处理包括Shackhand (HELLO/FEATURES_REQUEST/FEATURES_REPLY/SET_CONFIG/BARRIER_REQUEST/GET_CONFIG_REQUEST/BARRIER_REPLY/GET_CONFIG_REPLY/STATS_REQUEST/STATS_REPLY)，LLDP等信息。

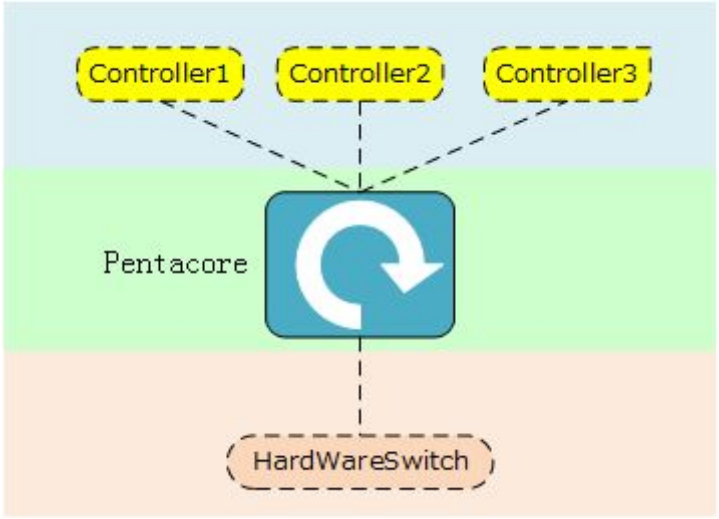


# Coordinator调度器的实现

## –Message Handling

**2. Core Message :**Packet_in信息通过可扩展的调度算法处理后进行最佳分配并上传给选择出的Controller；

Packet_out， FlowMod信息对应下发给PhysicalSwitch进而下发至HardWareSwitch。



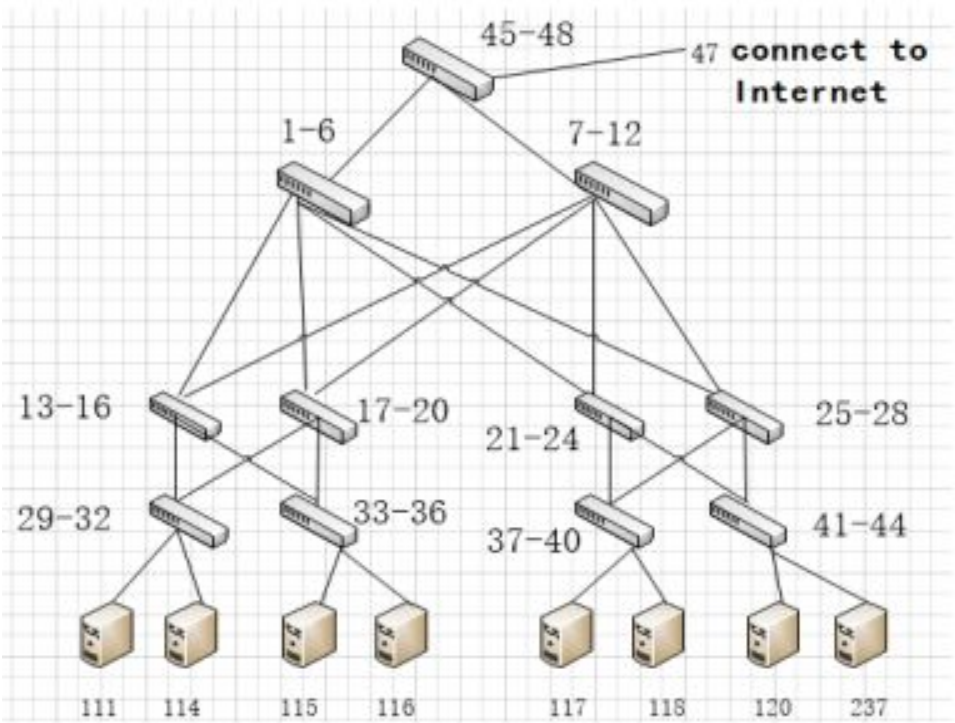
# Coordinator调度器的实现

## –Information Access

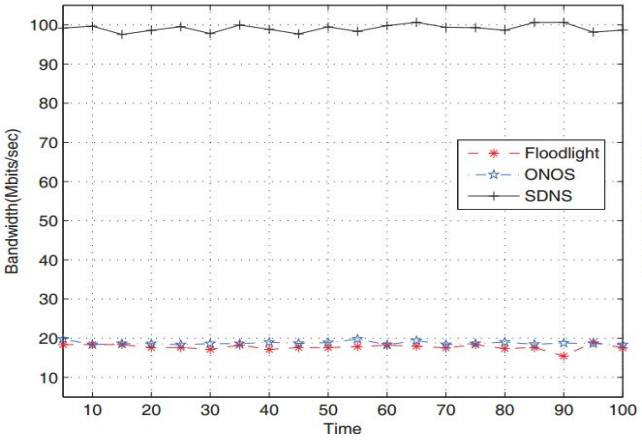
基于JettyServer，通过远程方法调用RPC向外部提供API接口，用于增加、删除Controller，选取当前的调度算法、获取网络拓扑信息等功能。

# SDNS架构的性能评价

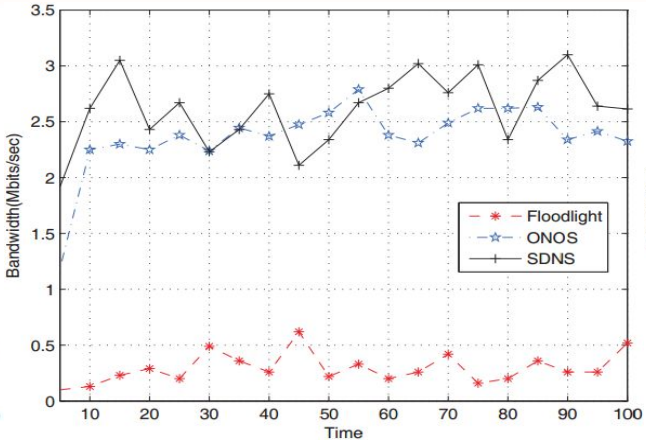
- Response Time
- Throughput
- Bandwidth
- Initializing time



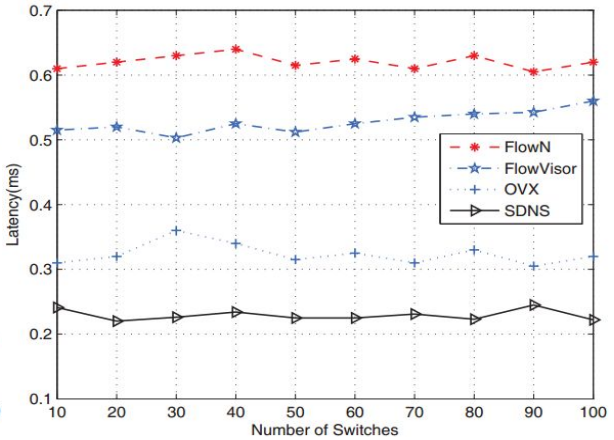
# SDNS架构的性能评价



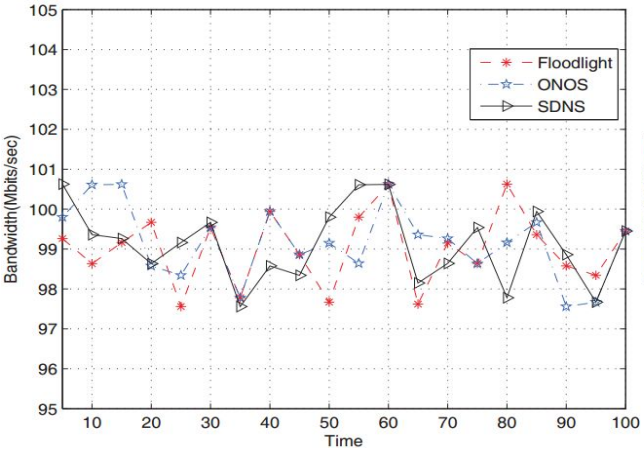

UDP Bandwidth without flow-mod



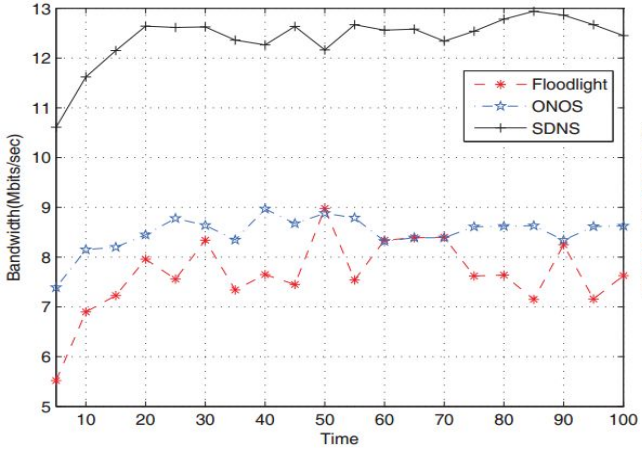

TCP Bandwidth without flow-mod



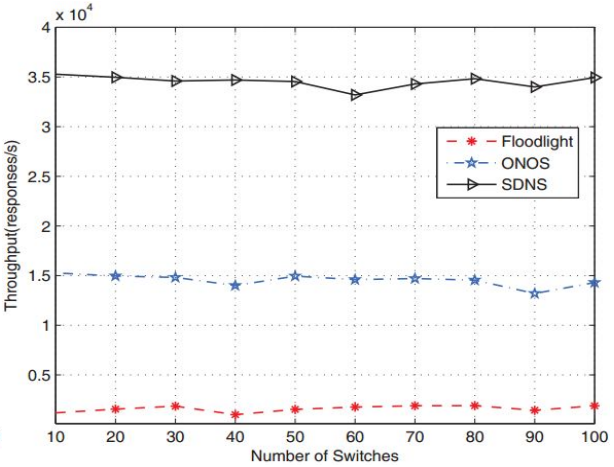

Latency of Coordinator




UDP Bandwidth with flow-mod




TCP Bandwidth with flow-mod




Throughput of Coordinator

# What ’s next

- –Multi-Coordinator
- –Coordinator HA
- –Multi-Core Big Controller

# Thanks