Exercise 4 – SDN Controllers

1. Controllers Overview (30P)

a) (20P) What are the two main approaches to implement controllers? What are their advantages and disadvantages? Give two examples for each approach.

Centralized controllers:

Single entity that manages all forwarding devices of the network.

Advantages:

Disadvantages:

Single point of failure and may have scaling limitations.

May not be enough to manage a network with a large number of data plane elements.

Example:

NOX/POX, Beacon, Floodlight

Distributed controllers:

Maybe a distributed controller can be a centralized cluster of nodes or a physically distributed set of elements or clusters of controllers inside each data center and distributed controller nodes in the different sites.

Advantages:

Some controllers tolerate crash failures.

Scalable.

Disadvantages:

Controllers do not tolerate arbitrary failures.

Consistency semantics problem.

Switch migration problem.

Example:

Onix, HyperFlow, HP VAN SDN, ONOS, DISCO,

b) (10P) Please list at least three criteria that can help in choosing the right controller for a certain task. Their advantages and disadvantages, such as performance, documents.

Learning Curve.

Programming language and model.

Developing community.

Type of target network.

2. Distributed Controllers (30P)

a) (10P) Please name and explain the main challenges of a distributed controller.

Synchronization/concurrency problems.

We should consider who controls which switch, and who reacts to which events.

b) (20P) What is weak consistency and what is strong consistency in terms of controller policies? Why and where is strong consistency needed?

Weak consistency:

A weak consistency called eventual consistency is slight relaxation –allowing readers to be behind for a short period of time.

When Physical network state changes asynchronously, we can use weak consistency, and control apps know how to deal with eventual consistency.

Strong consistency:

Ensures that only consistent state can be seen.

Upon an update to the network state by an instance, all subsequent reads by any instance returns the last updated value. It adds complexity and latency to distributed data management.

If you do not want to have multiple masters for a switch, you can use strong consistency.