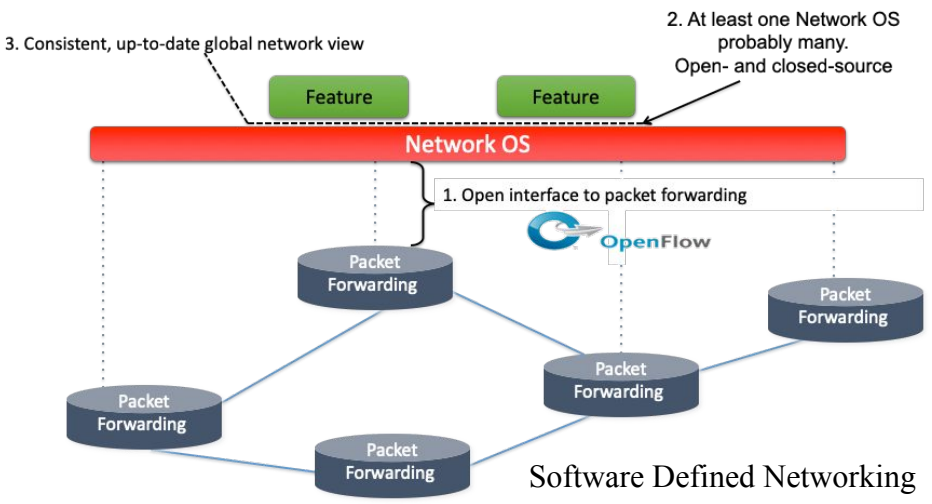
ACMC: An Architecture for Consistent Multi-Controller Communication in SDN

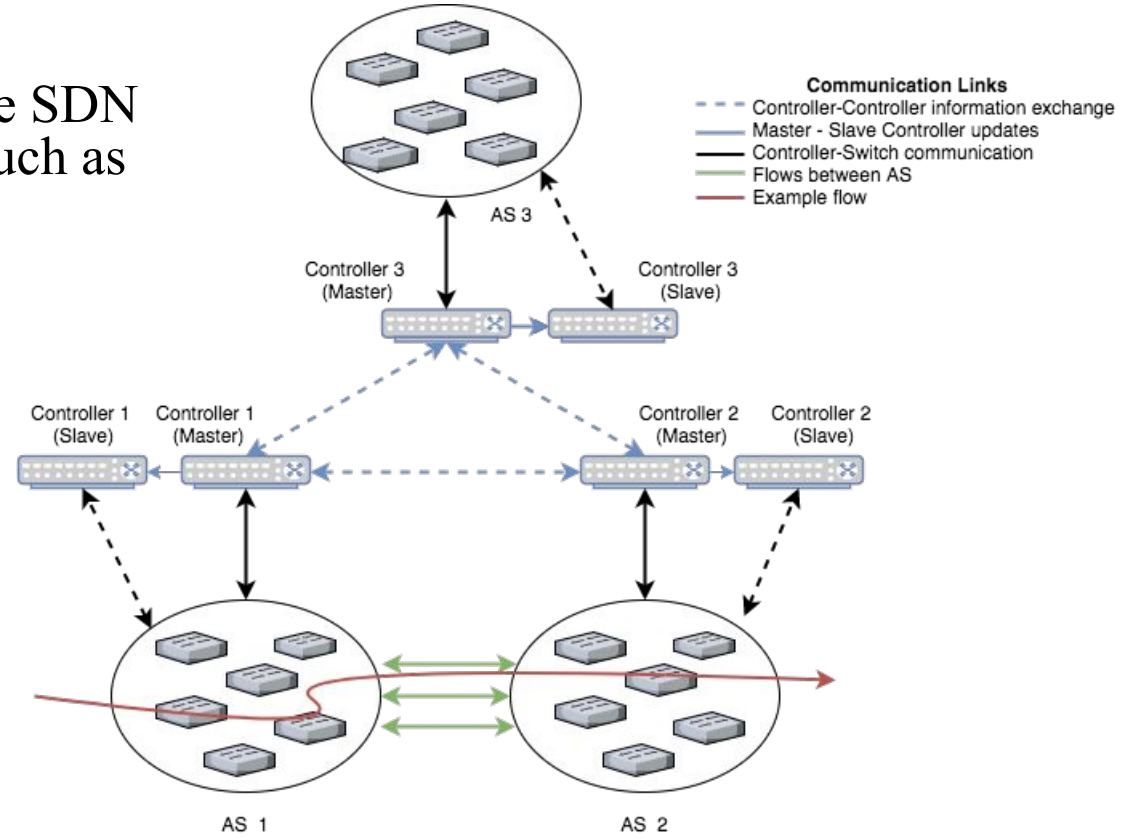
Sourav Panda, Aditya Dhakal, Elizabeth Liri, Risul Islam, Omar Faruk Rokon, Shohidul Islam, KK Ramakrishnan

Problem : Can we suggest a new architecture of SDN controller for what we can achieve consistent multi-controller communication?

- **Software defined networking (SDN)** enables efficient management of computer network by decoupling control plane (routing process of network packets) from data plane (forwarding process of network packets)
- **Autonomous systems (AS)** can benefit from SDN by simplifying the management of their control plane. E.g. by having a global flow table, route computation etc.
- In scenarios where multiple AS are connected to each other, the SDN can benefit the control plane by sharing relevant information such as
  - links connecting two AS
  - flows that are destined to another AS,
  - load balancing and traffic engineering over multiple links
- However, there are challenges associated with sharing control plane information such as
  - What information needs to be shared?
  - How do we share the information with different controllers?
  - How do we make sure the shared information is not stale?
  - How could foreign SDN controller benefit from information obtained from another SDN controller?

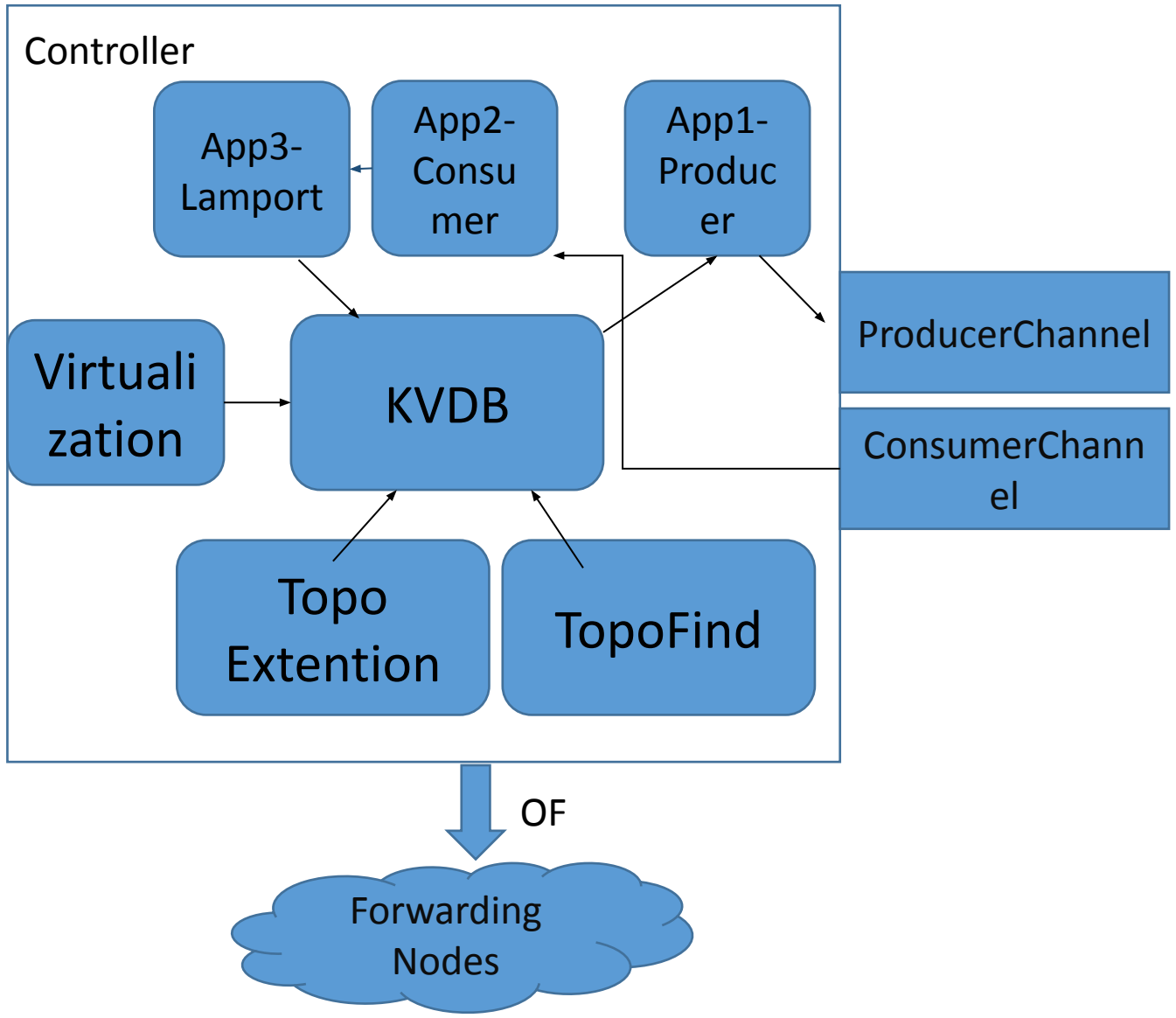


Software Defined Networking



Multiple AS SDN implementation

Architecture

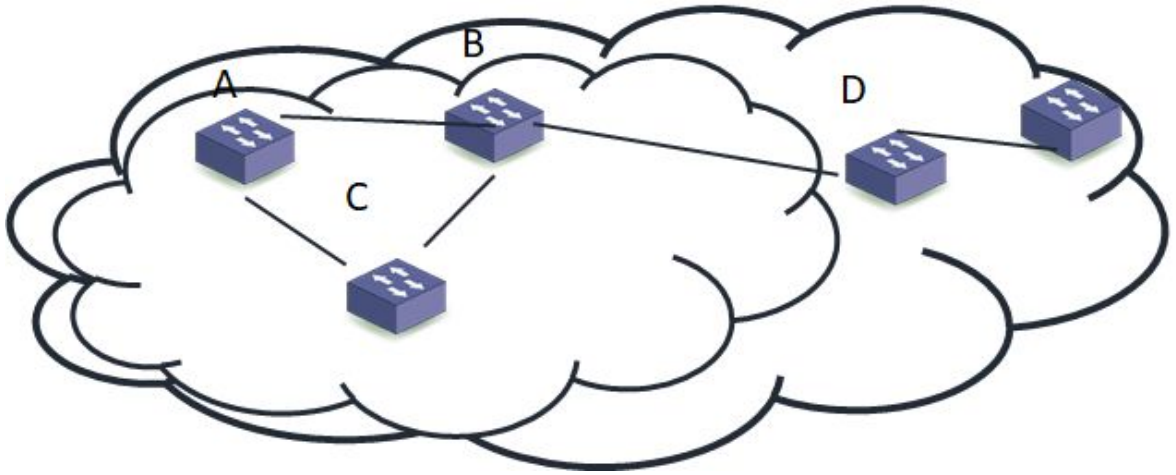


Key	Value	
Node ID	Is Virtual	IP,OF Version,Port numbers
Link ID	Is Virtual	Node ID src, dest, Port ID src, dst, BW, is Interdomain
Port ID	Is Virtual	Node ID, Port MAC, is active, is edge_post, VLAN ID, Throughput
Reachability	IP prefixes, length	
Node Table ID(Flow Entry)	OpenFlow specification columns	
Link Utilities	Link ID, Link utilities	
Flow Path	Port ID(in), Node ID src, Port ID(out).....	

SDN Virtualization

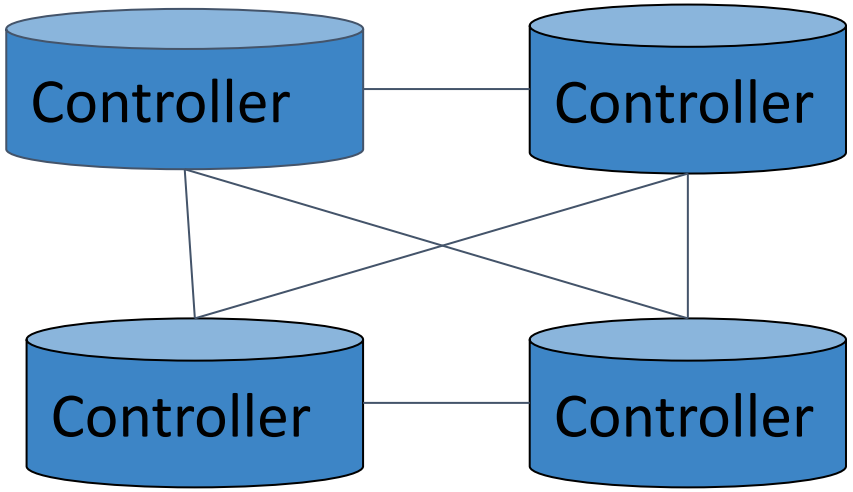
Physical Entity [edgelist->(NodeIP,Port)]	Virtual Entity	IP prefixes of hosts
AB->(Aip,0000,Bip,0001) BC->(Bip,0002,Cip,0003) CA->(Cip,0004,Aip,0005)	Xip,0056	Ypref, Zpref

- Which part of SDN is virtualized?
  - Each autonomous system (AS)
- How is it virtualized?
  - Each AS is fully visible only to its controller
  - Each AS is virtualized as single big switch to other AS
  - As shown, controller of AS1 sees AS2 as a single switch; only shared links are visible in both sides.



- Why virtualization?
  - Protects individual AS from security threat, while being connected with other AS

Communication Between Controllers (TCP/SSL Based Full Mesh)



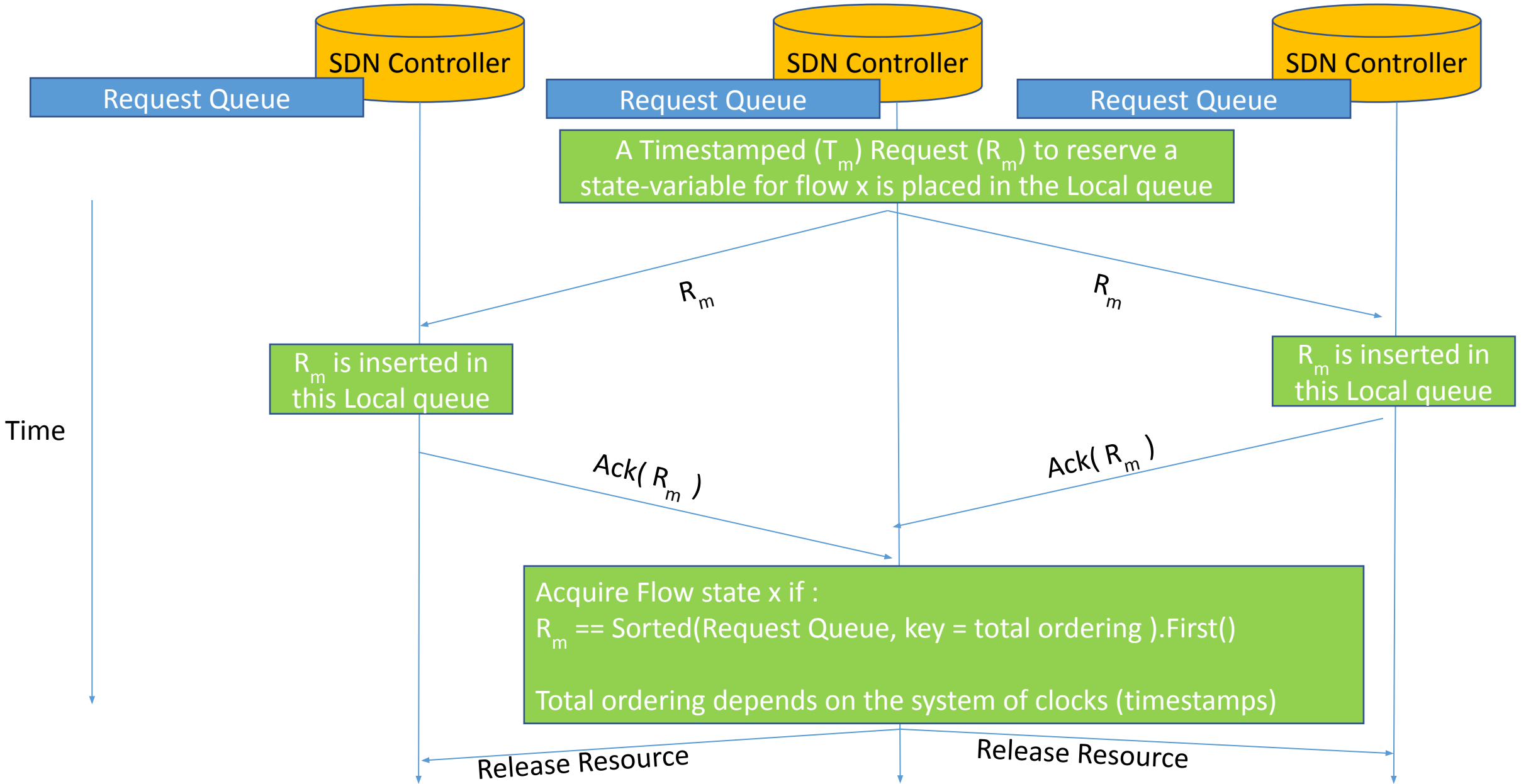
Connection: Set up a TCP/SSL connection with each peer. Via that connection, all messages are exchanged.

Messages: 4 types of messages like BGP

1. OPEN: After establishing TCP connection, the first message sent is the *Open*. Timers should be negotiated here. After that, *KEEP_ALIVE* will be kept sending.
2. UPDATE: Send the KVDB update entries whenever necessary in JSON file format
3. NOTIFICATION: If any error occurs, send notification to all and it will be in idle state
4. KEEP_ALIVE: To let others know that it is alive. If one controller won't receive KEEP_ALIVE message. It will release the connection thinking the link has failed.

Except UPDATE, all messages are sent in packet format.

Consistency (Lamport Clocks Per-State-Variable)



Implementation

- Goals of implementation
  - Determine if design is feasible
  - Evaluate performance
- Evaluation criteria
  - Overhead due to extra information
  - Time to synchronize
- Implementation:
  - Mininet with ONOS
- Information shared between controllers via E-W interface
  - Topology primitives
  - Flow primitives (5 tuple and port used to identify flows)