Controller HA using Active-Standby Model

ODL Magnesium DDF, Antwerp. 26-27 September 2019

Ajay Lele, Lumina Networks
alele@luminanetworks.com

Agenda

- OpenDaylight HA features
- Limitations of existing HA features
- HA using Active-Standby model
- Usage considerations

High Availability is **not** optional

- High Availability (HA) is the ability of a system to be continuously available over an extended period of time
- Telcom Service Providers have to abide by strict SLAs and hence place a very high importance on HA of components in their architecture
- Being responsible for managing network elements, SDN Controller is a critical component of any solution it is part of, and thus needs to be HA capable

HA in OpenDaylight

- At node level
  - Clustering implementation allows more than one Controller node to work in tandem with each other
  - Data is logically segmented into shards, with one node getting elected as leader of every shard
  - Election of leader as well as replication of data across majority of followers happens using RAFT distributed consensus algorithm
  - Cluster is tolerant to failure of [n - (n/2) - 1] nodes
  - Geo-clustering feature extends HA to another geographical location. Ex: 6-node geo-cluster with 3-node (voting) at primary site and 3-node (non-voting) at DR site
  - In a geo-cluster, switching of voting behavior is not automatic

- At service level
  - When only one instance of service across the cluster is desired
  - Clustering Singleton Service provides ability to deploy a service on multiple cluster nodes but to have it running actively on only one node
  - If node hosting the active service instance fails, another candidate node is automatically selected to host the service

Limitations with existing HA features

- Scalability
  - OpenDaylight clustered data-store (CDS) uses strict consistency model i.e. Tx can complete only when majority of cluster nodes confirm that they have successfully persisted the change locally
  - This adds considerable overhead and results in performance degradation when Tx rate is very high and/or size of data involved is large

- Supportability
  - CDS implementation (including that of underlying RAFT algorithm) is home-grown and barrier to entry in understanding and debugging it is high
  - CDS is built on top of akka (remoting, clustering, persistence) and is affected by its bugs and limitations. Ex: poor handling of large message sizes

Controller HA Requirements

1. Ability to detect when a controller instance fails
2. Ability to create or switch to a new controller instance
3. Ability to initialize new instance with last known controller state
4. Ability to switch NB and SB connections from failed to new instance
5. Ability to perform steps 1-4 while meeting required SLAs (ex. downtime, data loss)
6. Runtime and operational overheads (performance impact, ease-of-use) due to HA mechanism should be reasonable

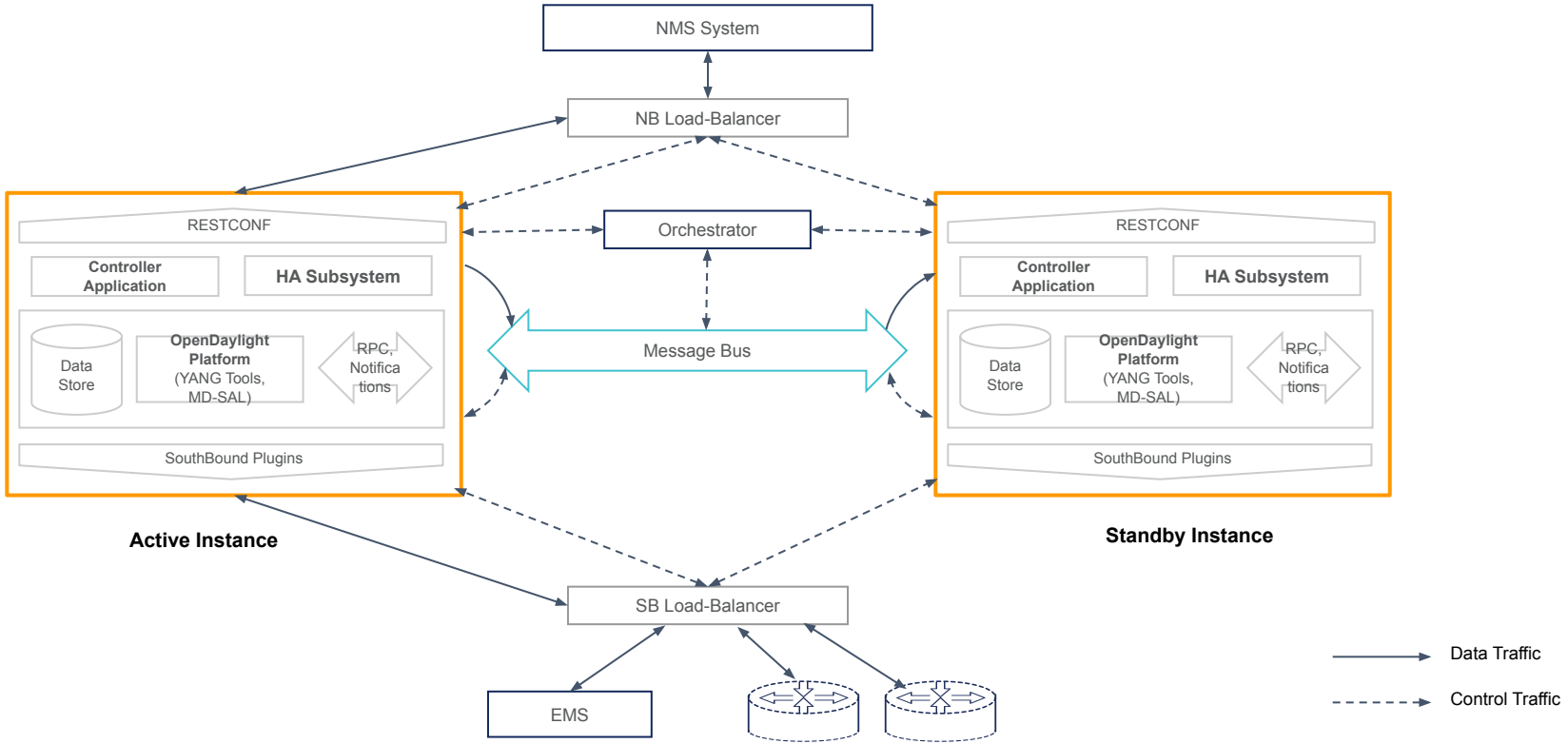
Active-Standby Model for HA

- Two Controller instances working in tandem
  - One in active mode, other in standby mode
  - Application is deployed on both instances
  - Active instance runs the business logic and handles NB/SB interactions

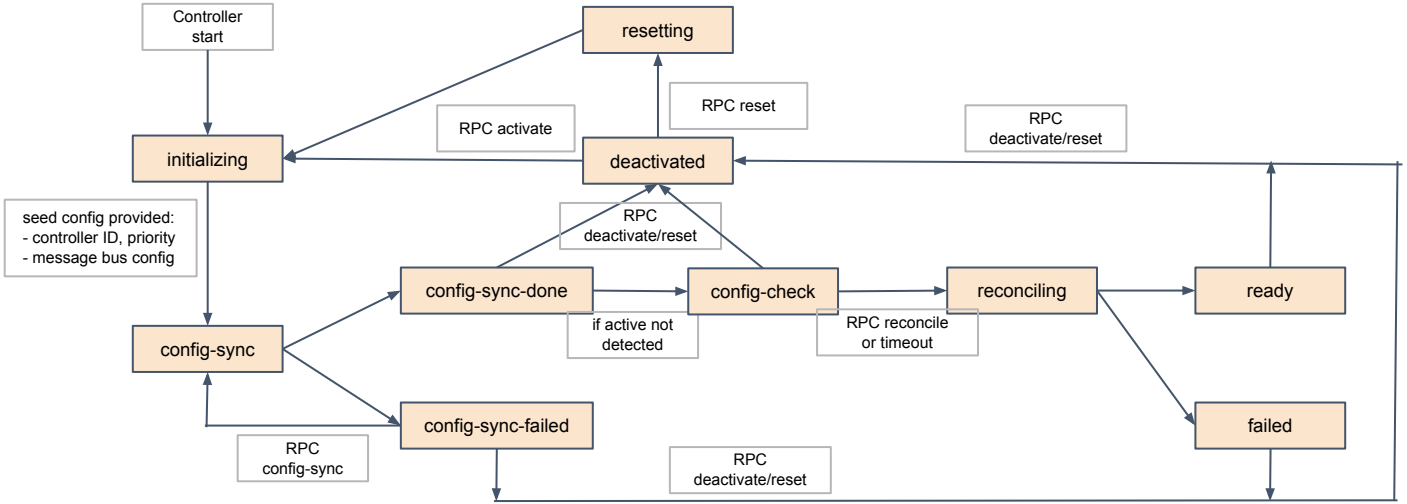
- Instances communicate with each other via an external message bus
  - Separate topics for different types of data
  - Periodic heartbeats used for liveness check
  - Config data changes propagated from active to standby instance (eventually consistent)

- In the event of failure of active instance
  - Standby instance detects failure and transitions to become the new active
  - NB/SB load-balancer (external) is re-configured to forward traffic to the new active instance
  - Business logic on new active instance reconciles with the network state

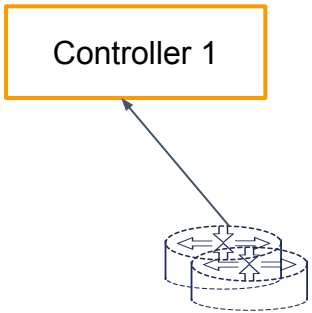
System Components



System States

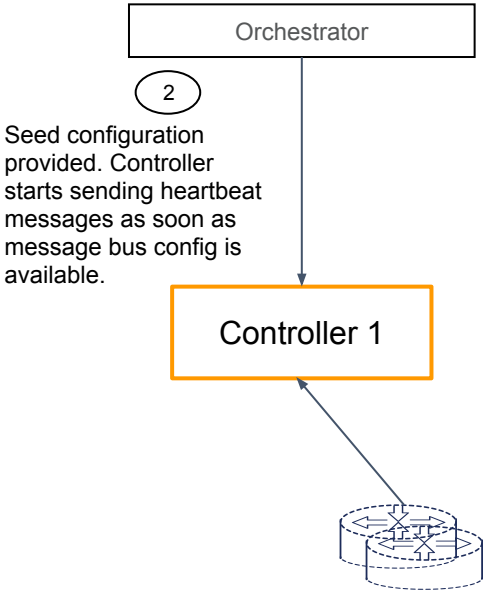


Initial Installation (1/3)



1

initializing state (role=other): Controller boots and ends in initializing state because there is no seed configuration.



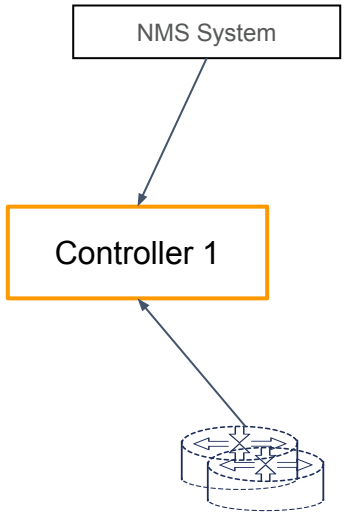
2

Seed configuration provided. Controller starts sending heartbeat messages as soon as message bus config is available.

3

ready state (role=active): Controller transitions to config-sync-done state. It detects that active instance is not available, changes its role to active and moves through config-check and reconciling to ready state.

Initial Installation (2/3)

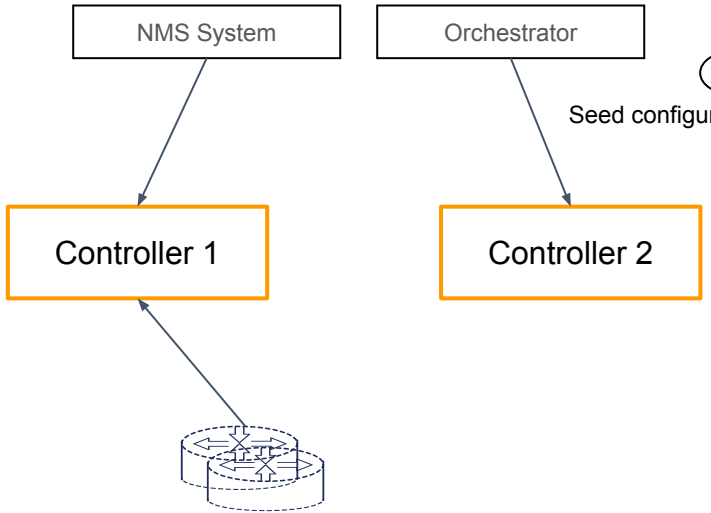


ready state (role=active): Controller sends periodic heartbeats and config change updates to message bus. NB/SB connections are routed to this instance.

4

initializing state (role=other): Controller boots and ends in initializing state because there is no seed configuration.

Initial Installation (3/3)

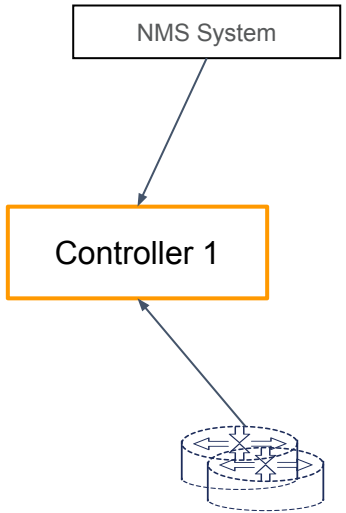


ready state (role=active): Controller sends periodic heartbeats and config change updates to message bus. NB/SB connections are routed to this instance.

config-sync-done state (role=standby): Controller reads config data from the message bus and moves to config-sync-done state. Since active instance is present, it changes role to standby and remains in this state. It monitors the bus for periodic heartbeat messages from active instance as well as any configuration updates.

Active Failover (1/3)

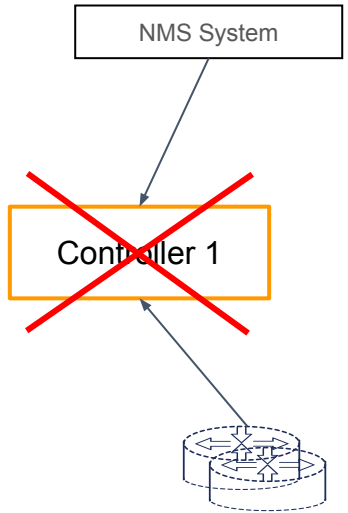
ready state (role=active): Controller sends periodic heartbeats and config change updates to message bus. NB/SB connections are routed to this instance.



Controller 2

config-sync-done state (role=standby): Controller monitors the bus for periodic heartbeat messages from active instance as well as any configuration updates.

Active Failover (2/3)



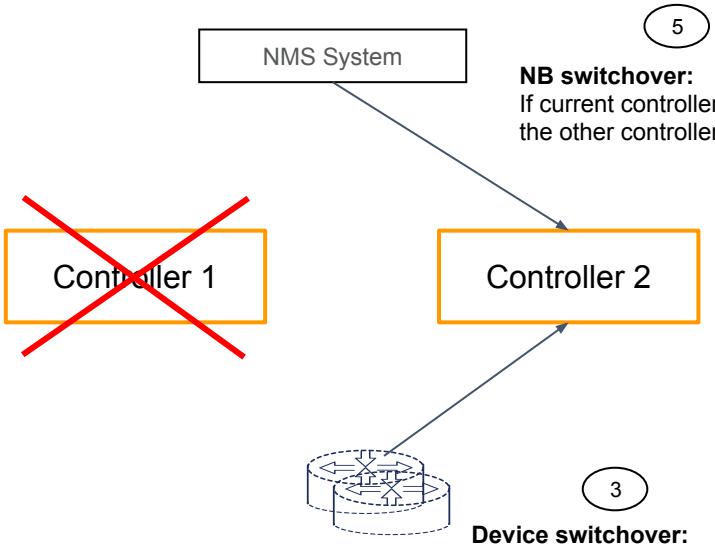
1

Controller becomes unavailable or moves to failed state.

2

config-check/reconciling state (role=active): Controller detects that active instance is down/failed. It changes its role to active and transitions to config-check and then reconciling state.

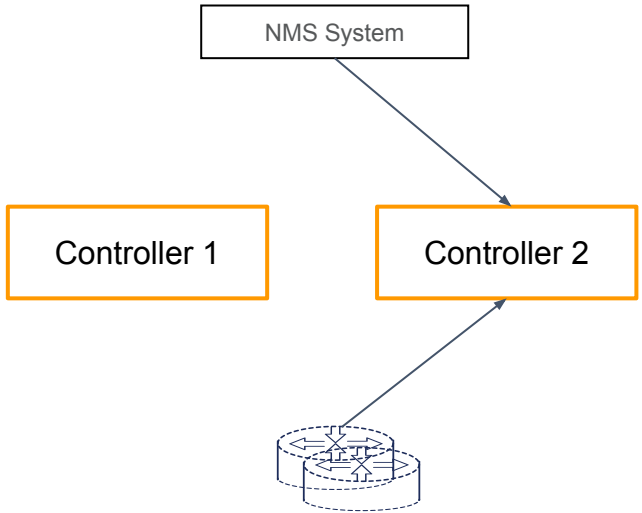
Active Failover (3/3)



Old Active Recovery

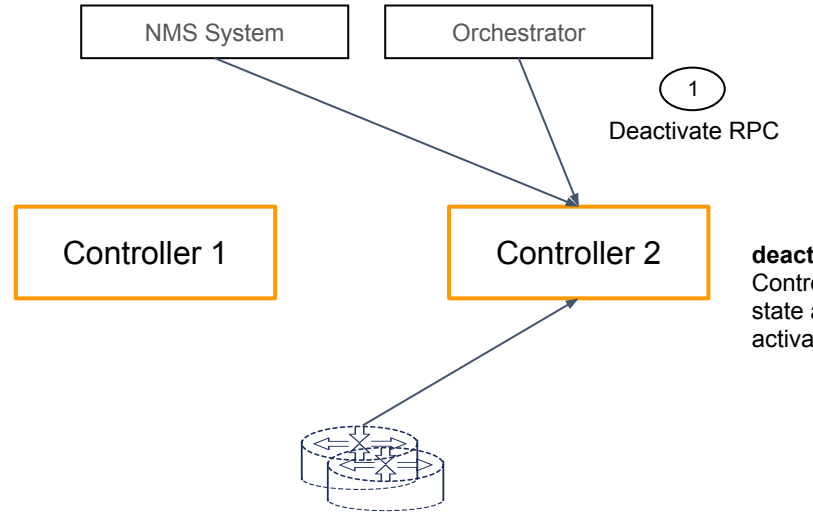
1

config-sync-done state (role=standby):
After Controller restarts or resumes operation, it moves to config-sync-done state. Since active instance is present, it changes role to standby and remains in this state. It monitors the bus for periodic heartbeat messages from active instance as well as any configuration updates.

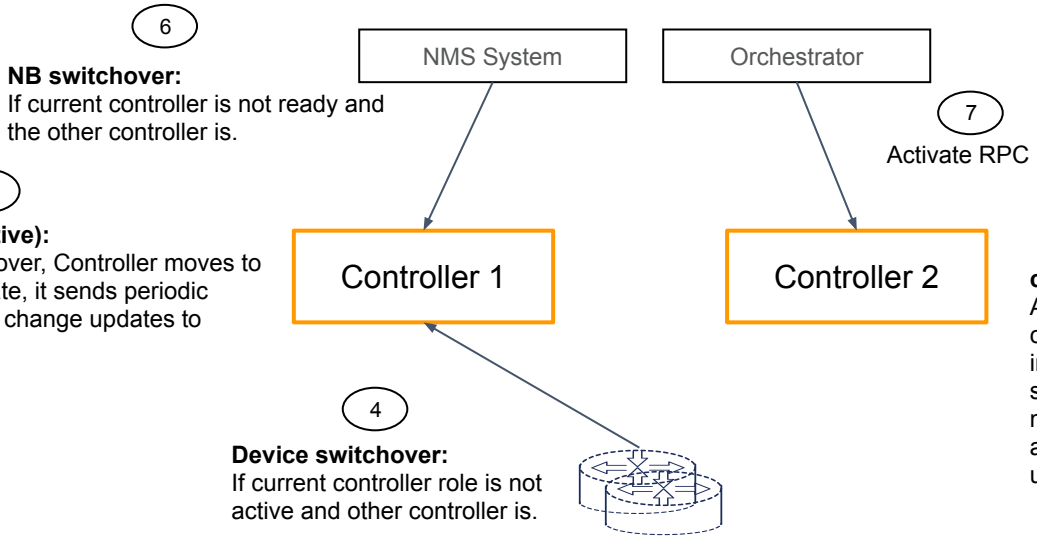


ready state (role=active):
Controller sends periodic heartbeats and config change updates to message bus. NB/SB connections are routed to this instance.

Manual Switchover (1/2)



Manual Switchover (2/2)



NB/SB Connection Switchover

- When Controller switchover occurs, NB/SB connections should move to new active instance
- HA Subsystem exposes RESTCONF endpoint to provide HA status (role & state info)
- This endpoint is periodically polled by NB/SB load-balancers to determine when switchover needs to happen

- SB switchover criteria
  - If current Controller role is not active, and other Controller role is active
  - Switchover needs to finish before/during reconciliation stage

- NB switchover criteria
  - If current Controller state is not ready, and other Controller state is ready
  - Switchover happens after active instance has moved to ready state

Config State Replication

- Config state replication to standby instance happens using message bus
  - Separate topic per module
  - Data retention strategy configured taking into account max. instance downtime

- Active instance publishes config change updates to the bus
  - Entire module data snapshot whenever any data in the module changes
    - Suitable if module data is relatively small, changes are infrequent
  - Periodic module data snapshot and individual module data change events
    - Suitable if module data is large, changes are frequent

- Standby instance reads config change updates from bus to keep its state updated
  - Last `n` messages for each module during config-sync to recreate current state
  - New messages in config-sync-done state to keep its state updated

- Since data replication provides eventual consistency, there is possibility of data loss
  - `config-check` state is used by Orchestrator to fetch/compare config state against expected state and fix any discrepancies found

Areas for Further Exploration

- Scalability (large data sizes, Tx rates)
- Need for device connection to not drop during failover (ex. route injection with BGP)
- Standard mechanism to publish data change notifications to external message bus

Thanks