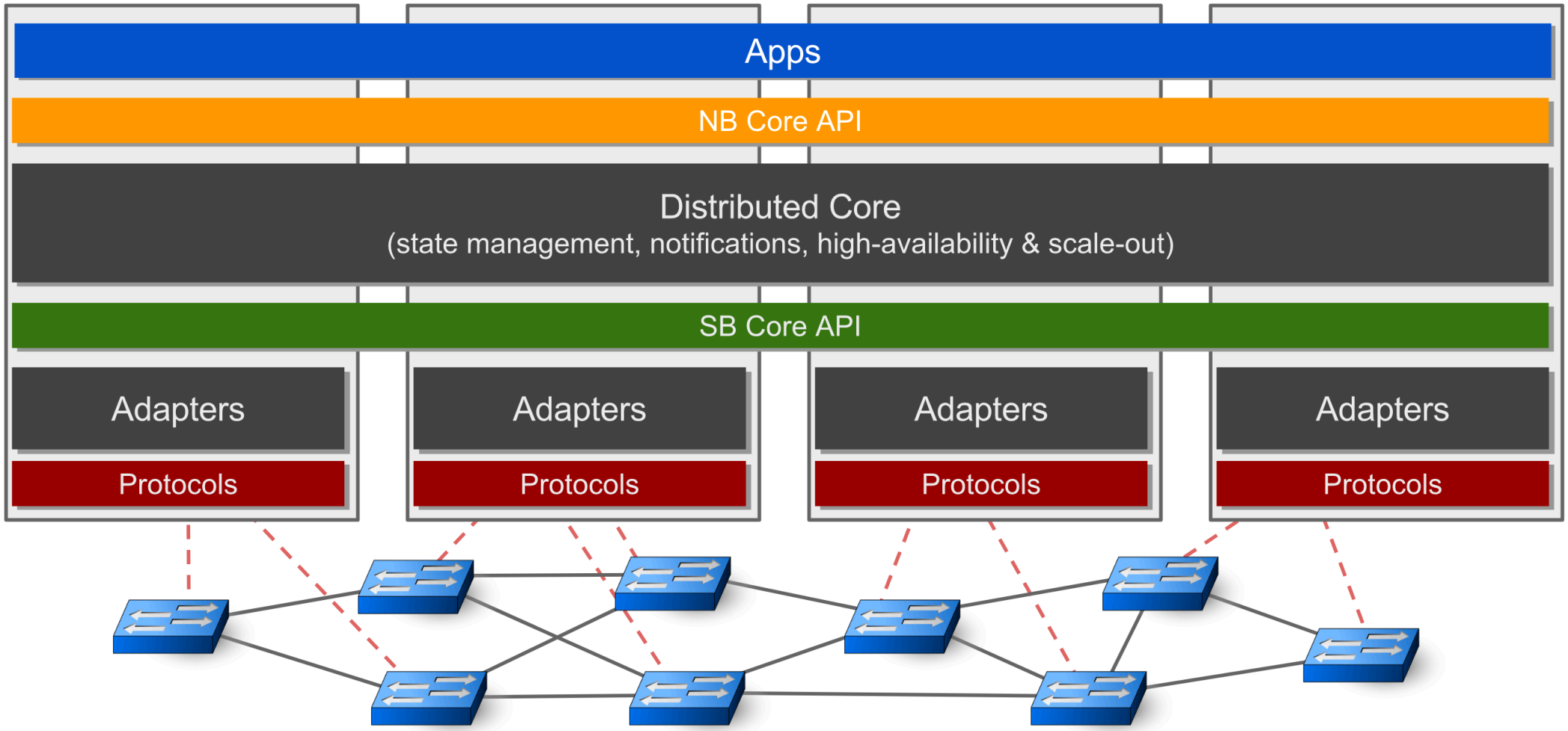
ONOS East/West Communication

ONOS architecture recap: the big picture



ONOS architecture recap: NB

- The ONOS Northbound exposes core services to apps that may need to use them
- It provides abstracted views of the Network state
- Apps can use these Northbound interfaces to modify network behavior

ONOS architecture recap: SB

- The ONOS Southbound provides simpler interaction between the ONOS core services and the physical hardware (through providers and drivers)
- The Southbound supports a variety of protocols including NETCONF, OpenFlow and other major standards
- Information about network state is also provided back to the ONOS core via the Southbound

ONOS architecture recap: core

- The CORE services provide means for apps to manipulate or discover network state
- These services are backed by the distributed stores in the ONOS core
- The ONOS core contains a number of distributed stores which are used to maintain both configured state and discovered network state

Subsections

- ONOS communication
- Copycat communication
- Gossip protocol

ONOS communication

- Heartbeats are used by the ONOS cluster to determine node up/down status
- This information is accrued and processed in DistributedClusterStore
- In order to provide adaptive failure detection ONOS employs an accrual failure detector

Accrual failure detector: general case

- Conventional failure detectors produce a binary status, useful if the detector can be customized for each use case but inflexible for different scenarios
- Accrual failure detectors build up state and give a level of certainty that a given node is up or down, by using this type of information different dependent processes can begin handling failures at different points based on requirements

Failure detection in ONOS

- ONOS uses this failure detector only to determine node up/down state, the threshold (phi) is configurable but the degree of certainty that a given node is up/down cannot be directly queried
- When ONOS detects a node is down it updates the state and this results in appropriate notifications being propagated and handling behavior being initiated

Copycat communication

What is Copycat

- Copycat is an open source implementation of the RAFT consensus protocol
- Copycat runs across Copycat Server implementations on each of the nodes on an ONOS cluster, these servers take commands and answer queries from clients including those hosted on the same machine as the servers themselves
- Copycat’s east/west communication includes a heartbeat mechanism used to determine the liveness of cluster memebers

Basics of Copycat

- Copycat maintains a single consistent log of commits across all participating servers
- Copycat nodes create heartbeats by appending empty commits to the log, these commits can then be used to determine the liveness of cluster members
- As entries are committed to the log they are applied to state machines in each server which update state and provide a consistent view of the most up to date state

Copycat customization in ONOS

- To improve performance ONOS uses partitions
- Partitions are subsets of the overall ONOS cluster that host a subset of the total number of state machines in the cluster
- These partitions maintain serial ordering within each partition but provide no guarantees of ordering among clusters

Copycat consistency levels

- Sequential consistency – the client guarantees that changes are seen in the order they occur this does not have a realtime guarantee and so is lighter weight for cluster communication
- Linearizable consistency – when a client places a query a majority of the cluster is contacted to retrieve the appropriate information to form a response
- Linearizable lease consistency – optimizes communication costs by allowing a read from the leader **without** contacting the cluster majority if a majority has been reached within the last timeout period

Copycat replication

- Replication among active replicas is done in a strongly consistent manner, this is relatively costly but provides good guarantees when the amount of data involved is small
- Gossip protocol attempts to keep passive replicas up to date however without the same priority given to timeliness, this incurs lower costs but may require some amount of catchup when a passive replica is moved to active state

Copycat failure detection

- Failure detection is handled by a binary failure detector based on heartbeats
- Failure is determined by the leader who pushes the heartbeats to cluster members
- In the case of communication failure where the leader cannot form a quorum the leader steps down and another node will initiate election procedures

Potential improvements to Copycat

- Improved parallelism in clients
- Replacement of failed nodes within a partition
- Balancing leadership responsibilities for partitions within a cluster
- Removal of serialization responsibilities from Copycat

Gossip Protocol for Eventual Consistency

Gossip Protocol

- For certain ONOS stores replication is provided via a gossip protocol
- Gossip protocols incur smaller overhead costs as compared to strongly consistent protocols since the required frequency of messaging is lower
- This is also known as optimistic replication

Gossip Protocol

- Gossip protocols are useful in a more specific case, particularly a case where changes are not too frequent
- Gossip protocols typically are only guaranteed to converge after some period of quiescence
- This makes gossip protocols a good match for the device and link subsystems and any other where changes are not frequent and there is ample time for state to converge

What is a Gossip Protocol

- In the ONOS implementation of the gossip protocol each participant will periodically select another peer and send that peer an advertisement representing its current state
- The recipient of this advertisement will compare the received advertisement to its internal state and attempt to reconcile the two
- The peer who sent the advertisement will be informed of the resolution
- The process repeats

How is it Different?

- The key difference is that there is no serialized log or other ordered representation of events over time
- There may be times throughout the process where state among nodes is inconsistent
- There are possible conditions where state will not converge (due to persistent frequent changes)

Q & A