

VARIOUS STUDIES ON SDN

Sedef Savas

July 21, 2017

Netlab Friday Group Meeting

THE ROLE OF INTER-CONTROLLER TRAFFIC IN THE PLACEMENT OF DISTRIBUTED SDN CONTROLLERS

TIANZHU ZHANG, ANDREA BIANCO, SAMUELE DE DOMENICO, PAOLO GIACCONE DEPT.
ELECTRONICS AND TELECOMMUNICATIONS, POLITECNICO DI TORINO, ITALY
2016, ARXIV

Motivation

Multiple controllers to improve network performance and reliability.

Openflow control traffic exchanged between **controllers and switches**, control traffic exchanged **among the controllers** in the cluster, needed to run **coordination** and **consensus algorithms** to keep the controllers synchronized.

They suggest a careful placement of controllers, that should take into account both the above kinds of control traffic.

They optimize the planning and the design of the network supporting the control plane, especially when the network is very **large** and **in-band** control plane is adopted.

Control Plane

Single controller:

limited reliability, due to the single point-of-failure.

Control traffic between the switches and the controller concentrates on a single server, whose processing capability is limited, creating scalability issues

Distributed:

Less processing per controller

More resiliency

Distributed Controllers

Distributed controllers adopt **coordination protocols** to synchronize their shared data structures which define the network state and to enable a centralized view of the network state.

They follow a **consensus-based** approach in which coordination information is exchanged among controllers to reach a common network state.

Delays

These delays affect the controller reactivity perceived at the switches.

Any read/write of a shared data structure at a controller is directed to a possibly different “data owner” controller.

Controller-to-controller delays must be added to the switch-to-controller delays when evaluating the controller’s reactivity perceived at the switches.

Thus, optimal placement of the controllers on the network topology must consider not only the delays between the switches and the controllers, but also the delays between controllers

SDWAN

SDWANs poses severe technical challenges.

- supporting a responsive controller-to-controller interaction
- in-band control

Data consistency models

Network state is stored in shared data structures (e.g., topology graph, the mapping between any switch to its master controller, the list of installed flow rules), whose consistency across the SDN controllers can be either **strong or eventual**.

Strong consistency implies that contemporary reads of some data occurring in different controllers always lead to the same result.

Eventual consistency implies that contemporary reads may eventually lead to different results, for a transient period.

Data consistency models (cont.)

Anytime a data structure is shared across the controllers, they must synchronize through **a consensus algorithm** that guarantees a consistent view of the data in case of updates.

Consensus algorithm is very complex, to deal with all possible failures and network partitions, and it is tailored to a specific level of data consistency.

Each controller is required to interact with the other controllers through the Ctr-Ctr plane, thus introducing some latency to synchronize their internal data structures.

Consensus Algorithm for ONOS and ODL

Raft consensus algorithm is based on a logically centralized approach, since any data update is always forwarded to the controller defined as leader of the data structure.

- Leader propagates the update (followers).
- Update is considered committed whenever the majority of the follower controllers acknowledges the update.

In ONOS data can be also synchronized according to an eventual consistent model, in parallel to strong-consistent data structures.

Eventual consistency is achieved through the so called “anti-entropy” algorithm, updates are local in the master controller and propagate periodically in the background with a simple gossip approach: each controller picks at random another controller, compares the replica and eventual differences are reconciled based on timestamps.

DATA-OWNERSHIP MODELS IN DISTRIBUTED CONTROLLERS

Controller reactivity as perceived by a switch depends on the local availability of the data necessary for the controller.

In a single data-ownership (SDO) model, a single controller(data owner) is responsible for update of data structure, and read/write operations must be forwarded to the data owner.

- Ctr-Ctr affects Sw-Ctr plane, as some Sw-Ctr request messages (e.g., packet-in) trigger transactions with the data owner on the Ctr-Ctr, perceived controller reactivity is affected by the Ctr-Ctr delay.
- data-ownership model in ODL and ONOS, strong-consistent data structures managed by Raft algorithm: a local copy of main data structures is stored at each controller, but any read/write operation is always forwarded to the leader.
- With this centralized approach, data consistency is easily managed and the distributed nature of the data structures is exploited only during failures.

DATA-OWNERSHIP MODELS IN DISTRIBUTED CONTROLLERS

In a multiple data-ownership (MDO) model, each controller has a local copy of data and can run locally read/write operations.

A consensus algorithm distributes local updates to all the other controllers. This model can decouple Sw-Ctr, Ctr-Ctr interaction, improves reactivity perceived by switch

Disadvantage is introduction of possible update conflicts that must be solved with ad-hoc solutions and of possible temporary data state inconsistencies leading to network anomalies (e.g. forwarding loops).

Thus, the model applies to generic eventual consistent data structures, as the ones managed by the anti-entropy algorithm in ONOS.

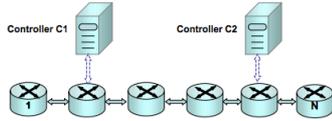


Fig. 1: Placement with minimum Sw-Ctr delay

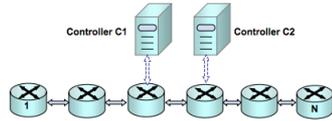


Fig. 2: Placement with minimum Ctr-Ctr delay

Small Sw-Ctr delays imply high reactivity of the controllers (i.e. small reaction time), whereas small Ctr-Ctr delays imply lower probability of network state inconsistency.

Reactivity of Data Ownership Models

They consider only the propagation delays of the physical links, and neglect all the processing times and the queueing delays due to network congestion.

A. Reactivity model for MDO model

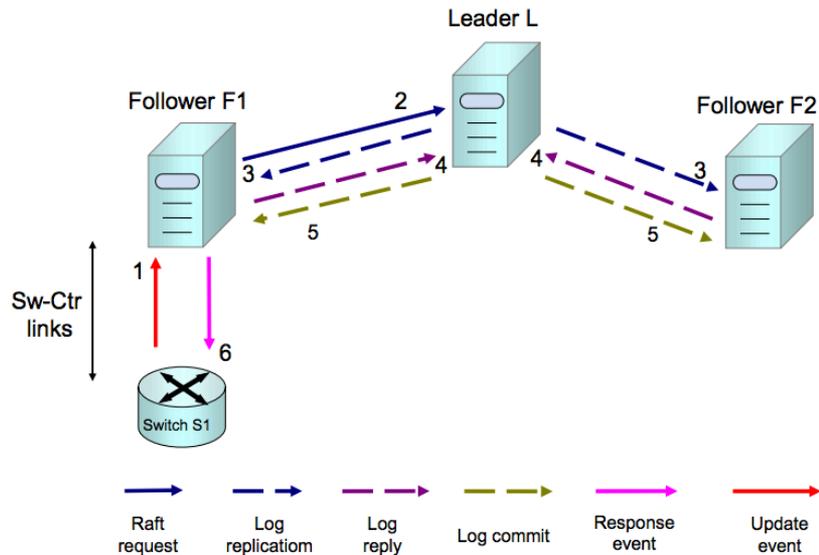
Property 1: In a MDO scenario for distributed SDN controllers, the reaction time perceived at the switch is:

$$T_R^{(m)} = 2d_{\text{sw-ctr}} \quad (1)$$

B. Reactivity model for S

According to Raft implementation in ODL, the controller can operate as either unique leader or as one of the followers, for a specific data store.

Leader sends a “log replication” message to its followers, waits for ack from majority, update is committed through a “log commit” sent to all the followers. After receiving the commit message, S1’s master controller can process the update, generate the response event to the switch.



Property 2: In a SDO scenario (e.g. adopting Raft consensus algorithm) for distributed SDN controllers, the reaction time $T_R^{(s)}$ perceived at the switch is:

$$T_R^{(s)} = 2d_{\text{sw-ctr}} + 2d_{\text{ctr-leader}} + 2d_{\text{ctr}^*-\text{leader}} \quad (2)$$

Thus, the reaction time is identical to the one for MDO model plus either 2 or 4 times the RTT between the controllers, when the master controller is either leader or follower of the shard, respectively.

PARETO-OPTIMAL RESILIENT CONTROLLER PLACEMENT IN SDN-BASED CORE NETWORKS

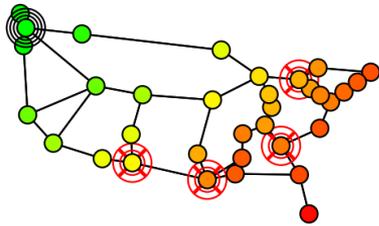
DAVID HOCK, MATTHIAS HARTMANN, STEFFEN GEBERT, MICHAEL JARSCHER, THOMAS
ZINNER, PHUOC TRAN-GIA UNIVERSITY OF WURZBURG, INSTITUTE OF COMPUTER
SCIENCE, WURZBURG, GERMANY

Evaluated on more than 140 topologies of the Topology Zoo.

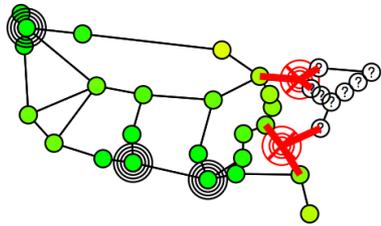
Their finding: for most of the topologies more than 20% of all nodes need to be controllers to assure a continuous connection of all nodes to one of the controllers in any arbitrary double link or node failure scenario.

In most topologies, where a single controller would be enough from a latency point-of-view, many more controllers are necessary to meet resilience requirements.

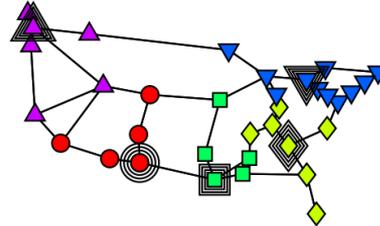
Illustration of different issues to be considered when judging the resilience of a controller placement.



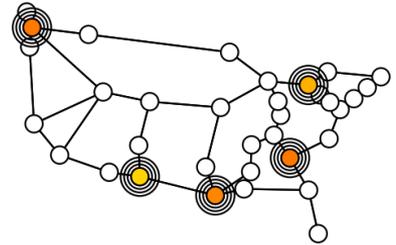
(a) Latency during controller failures.



(b) Controller-less nodes.



(c) Load imbalance.



(d) Inter-controller latency.

Controllers

 Nodes

 Broken controllers

 Broken nodes

 Controller-less nodes

Controller Failures

To increase resilience against controller failure, the controller placement optimization should not only consider the latencies during failure-free routing, but also worst case latencies during controller failures.

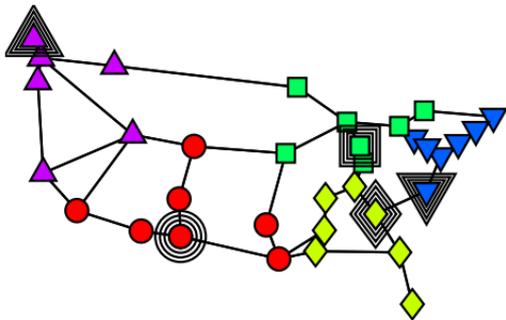
Network Disruption

the outage of network components, such as links and nodes, often has a much higher impact on the network stability, as it alters the topology itself. The shortest paths between some of the nodes change, leading to different latencies and possibly to the reassignment of nodes to other controllers. Even more severe is that entire parts of the network are in danger of being cut off by link or node outages. In the worst case, some nodes can no longer be connected to a controller as they are cut off from all controllers. These nodes are still working and able to forward traffic, but cannot request instructions anymore.

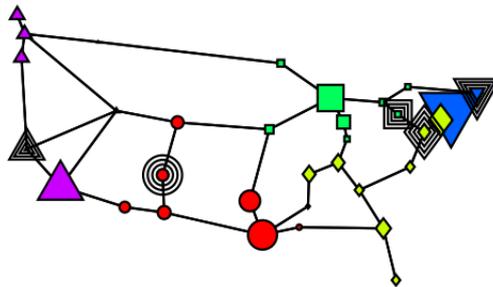
Load Imbalance

If the number of node-to-controller requests in the network increases, so does the chance of additional delays due to queuing at the controller system.

To be resilient against controller overload, the assignment of nodes to the different controllers should be well-balanced.



(a) Uniform node weights.



(b) City populations as node weights.

Inter-Controller Latency

If the control logic of the network is distributed over several controllers, these controllers need to synchronize to maintain a consistent global state. Depending on the frequency of the inter-controller synchronization, the latency between the individual controllers plays an important role.