Hierarchical Controller Architecture for SDN

Gourav Khaneja
Sweta Yamini Seethamraju
Supervisor: Prof. Philip Brighten Godfrey

November 5, 2013

Outline

Problem

- Single controller architecture is not scalable
- Distributed controller architectures provide global view to control application

Problem

- Single controller architecture is not scalable
- Distributed controller architectures provide global view to control application
- Problems are
  - Communication overhead
  - Network state inconsistencies
  - Management barriers across networks
  - Scalability of complete global view of the network

Problem

- Single controller architecture is not scalable
- Distributed controller architectures provide global view to control application
- Problems are
    - Communication overhead
    - Network state inconsistencies
    - Management barriers across networks
    - Scalability of complete global view of the network
- Trade-offs
    - Consistency vs. Responsiveness
    - Application design complexity (architecture aware vs. agnostic)

Problem

- Single controller architecture is not scalable
- Distributed controller architectures provide global view to control application
- Problems are
  - Communication overhead
  - Network state inconsistencies
  - Management barriers across networks
  - Scalability of complete global view of the network
- Trade-offs
  - Consistency vs. Responsiveness
  - Application design complexity (architecture aware vs. agnostic)
- Aim
  - Make control application agnostic to underlying distributed architecture
  - Aggregate network to simplify network view
  - Handle inconsistency in "SDN datapath"

Previous Work

- Hyperflow

Previous Work

- Hyperflow
  - Flat distributed controller architecture
  - Appliations are agnostic to underlying state distribution
  - Provides a logically centralized global view of the network
  - Results in performance degradation and transient inconsistencies

Previous Work

- Hyperflow
  - Flat distributed controller architecture
  - Appliations are agnostic to underlying state distribution
  - Provides a logically centralized global view of the network
  - Results in performance degradation and transient inconsistencies
- Onix

Previous Work

- Hyperflow
  - Flat distributed controller architecture
  - Appliations are agnostic to underlying state distribution
  - Provides a logically centralized global view of the network
  - Results in performance degradation and transient inconsistencies
- Onix
  - Provides flexible framework to handle controller topology
  - Provides generic distributed state management API
  - Provides a framework but not an approach
  - Design decisions left to control application

Previous Work

- Hyperflow
  - Flat distributed controller architecture
  - Appliations are agnostic to underlying state distribution
  - Provides a logically centralized global view of the network
  - Results in performance degradation and transient inconsistencies
- Onix
  - Provides flexible framework to handle controller topology
  - Provides generic distributed state management API
  - Provides a framework but not an approach
  - Design decisions left to control application
- Kandoo

Previous Work

- Hyperflow
  - Flat distributed controller architecture
  - Appliations are agnostic to underlying state distribution
  - Provides a logically centralized global view of the network
  - Results in performance degradation and transient inconsistencies
- Onix
  - Provides flexible framework to handle controller topology
  - Provides generic distributed state management API
  - Provides a framework but not an approach
  - Design decisions left to control application
- Kandoo
  - Two-level controller architecture
  - Events are propagated only on subscription
  - Applications need to be aware of hierarchy
  - View of root controller varies with control application

Our Approach

- What is being done ?
  - It's a Replicated state machine (strong or lazy replication) architecture, which had been well studied in Distributed Systems community
  - Past work uses DHTs (Onix), Transactional DB (Onix), DFS (Hyperflow)
- We are trying to study
  - Instead of just using a general repilication mechanism, can we exploit the fact that network views are being replicated ?
  - Could SDN datapath (replication client) make use of replication timestamps ?

Hierarchical Architecture

- Hierarchical Architecture
  - Controllers' view do not span the whole network, but only the 'underlying network'
  - Each controller behaves as a giant openflow switch for parent controller (Network Aggregation)
  - Controller translates OFP commands and network updates between parent controller and underlying networks

Network Aggregation

- Definition - Create and publish one flow table (referred to as *nflow*) for a given network. (1) Support addition of flows to *nflow* efficiently. (2) Handle intra-link failure efficiently, in such a way that *nflow* does not change.
- *nflow* schema (*portin*, *flowin*, *portout*, *flowout*, *priority*, *attributes*)
- Data structures[1]:
  - Add *external nodes* to the network graph G
  - Flow graphs: For each external node, e, we create/maintain a directed flow graph, f_e as an overlay graph on G, rooted at e, which is the source. End host and other external nodes acts as sink. The flow in f_e are divided at nodes among the incident links according to the flow tables at corresponding routers.
  - Other data structures include: Path array, Link array

[1] code @
https://github.com/MugiwaraLuffy/ACN/tree/master/simulator/src/graph

Timestamp

- An example - Link Balancer Controller [4]
- On a flow arriving at an ingress port, a path with minimum maximum utilization is choosen
- Each domain i maintains a vector timestamp, $T_i[1...n]$. $T_i[j]$ contains the latest revision number of domain j link utilizations, seen by domain i
- Each router in a domain is updated with domain's timestamp
- When a flow path is being set up at domain j, it is augmented by the T_j
- When a router in a domain i receives a flow with timestamp T_j, it may change part of flow's path if $T_j[i] < T_i[i]$
- Flow's timestamp is updated as $max\{T_j[k], T_i[k]\}$, $k = 1...n$
- OK, can we do better ? Yes!, a domain can change the entire flow's path according to timestamps.

Preliminary Results

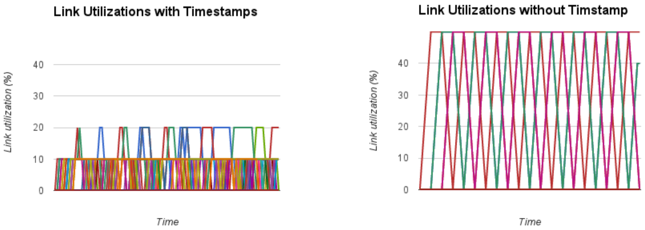

Figure: Comparision of Link utilizations with/without Timestamp [2]

'Long lived flows' flow between domain 1 and domain 3 through domain 2. A link balancer control application runs on each controller

[2]Simulator @ https://github.com/MugiwaraLuffy/ACN/tree/master/simulator/src/timestamp

Hitches

- Hierarchical Architecture
  - Suboptimal performance
- Network Aggregation
  - May backfire: single internal update may cause multiple updates in aggregated network
- Timestamps
  - Not a true datapath technique
  - Improve the performance of control applications, however, does not provide any guarantees

Future Work

- Understand the trade-off between sub-optimality and frequency of updates, in case of hierarchical architecture
- Revise Network Aggregation data structures so as not to explode network updates
- Implement and simulate each approach on real ISP topology with real traffic data. Analyze performance

Thank you!

References

1 T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker. Onix: a distributed control platform for large-scale production networks. In Proceedings of the 9th USENIX OSDI conference , pages 1 - 6, 2010.

2 A. Tootoonchian and Y. Ganjali. Hyperflow: a distributed control plane for openflow. In Proceedings of the 2010 INM conference, 2010.

3 Soheil Hassas Yeganeh and Yashar Ganjali. Kandoo: a framework for efficient and scalable offloading of control applications. In Proc. 1st Workshop on Hot Topics in Software Defined Networking (HotSDN 2012), pages 19-24, New York, 2012. ACM Press.

4 D. Levin, A. Wundsam, B.Heller, N. Handigol, and A. Feldmann, Logically Centralized State Distribution Trade-offs in Software Defined Networks in HotSDN, 2012.

References

5 Stefan Schmid and Jukka Suomela, Exploiting Locality in Distributed SDN Control, in HotSDN, 2013.

6 Advait Dixit, Fang Hao, Sarit Mukherjee, T. V. Lakshman and Ramana Kompella, Towards an Elastic Distributed SDN Controller, in HotSDN, 2013.

7 Nate Foster, Rob Harrison, Matthew L. Meola, Michael J. Freedman, Jennifer Rexford and David Walker, Frenetic: A High-Level Language for OpenFlow Networks. ACM PRESTO 2010.