Scalability and Resilience in Data Center Networks: Dynamic Flow Reroute as an Example

A use case of devolved controllers

Adrian S.-W. Tam Kang Xi H. Jonathan Chao

Department of Electrical & Computer Engineering
Polytechnic Institute of New York Univeristy

Globecom 2011

Use of controllers in routing

- Ethane (2007): Call set-up in OpenFlow network
- VL2 (2009): MAC address look-up prior to forwarding
- Hedera (2010): Dynamic flow reroute

M. Casado et al., “Ethane: Taking control of the enterprise,” in Proc. SIGCOMM, 2007.
A. Greenberg et al., “VL2: A scalable and flexible data center network,” in Proc. SIGCOMM, 2009.
M. Al-Fares et al., “Hedera: Dynamic flow scheduling for data center networks,” in Proc. NSDI, 2010.

Use of controllers in routing

- Ethane (2007): Call set-up in OpenFlow network
- VL2 (2009): MAC address look-up prior to forwarding
- Hedera (2010): Dynamic flow reroute

They're *omniscient* controllers

M. Casado et al., "Ethane: Taking control of the enterprise," in Proc. SIGCOMM, 2007.
A. Greenberg et al., "VL2: A scalable and flexible data center network," in Proc. SIGCOMM, 2009.
M. Al-Fares et al., "Hedera: Dynamic flow scheduling for data center networks," in Proc. NSDI, 2010.

Use of controllers in routing

- Ethane (2007): Call set-up in OpenFlow network
- VL2 (2009): MAC address look-up prior to forwarding
- Hedera (2010): Dynamic flow reroute

They're *omniscient* controllers

Full topology information

M. Casado et al., "Ethane: Taking control of the enterprise," in Proc. SIGCOMM, 2007.
A. Greenberg et al., "VL2: A scalable and flexible data center network," in Proc. SIGCOMM, 2009.
M. Al-Fares et al., "Hedera: Dynamic flow scheduling for data center networks," in Proc. NSDI, 2010.

Use of controllers in routing

- Ethane (2007): Call set-up in OpenFlow network
- VL2 (2009): MAC address look-up prior to forwarding
- Hedera (2010): Dynamic flow reroute

They're *omniscient* controllers

Full topology information

Scalable?

M. Casado et al., "Ethane: Taking control of the enterprise," in Proc. SIGCOMM, 2007.
A. Greenberg et al., "VL2: A scalable and flexible data center network," in Proc. SIGCOMM, 2009.
M. Al-Fares et al., "Hedera: Dynamic flow scheduling for data center networks," in Proc. NSDI, 2010.

Problems of omniscient controllers

- Full detail of network: Cost of operation
  - Memory, storage, probing bandwidth
- Slow
  - Dijkstra's algorithm is $O(V^2)$ or $O(E + V \log V)$
    i.e. larger the network, slower the response time

Problems of omniscient controllers

- Full detail of network: Cost of operation
  - Memory, storage, probing bandwidth
- Slow
  - Dijkstra's algorithm is $O(V^2)$ or $O(E + V \log V)$
    i.e. larger the network, slower the response time

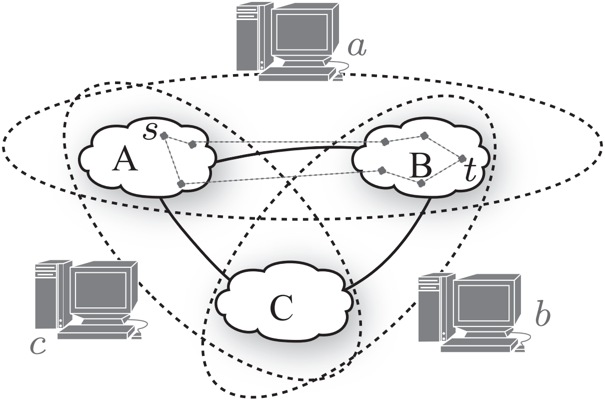
Omniscient controllers cannot scale with the network

Solution: Devolved controllers

devolved = not centralized

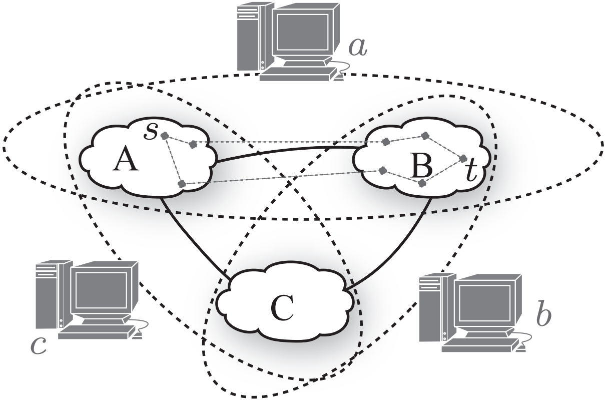
- Together is a centralized controller, with complete information
- Scalable
- Redundancy is almost free
- Favorible to those who needs real-time computation

Multipath network



- Each controller manages a partial topology
- Together covers the whole network

Example: Dynamic flow reroute

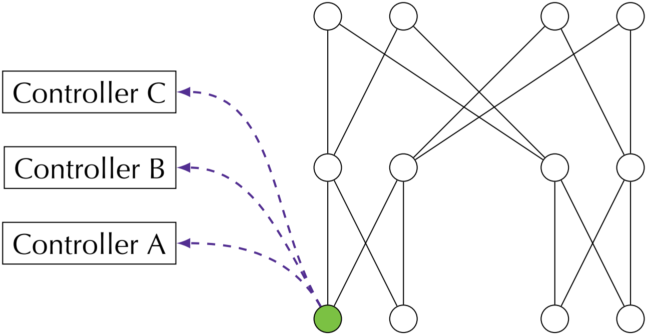


- Controller monitors link loads
- Move big flows off heavily loaded links dynamically

Operation

Reroute

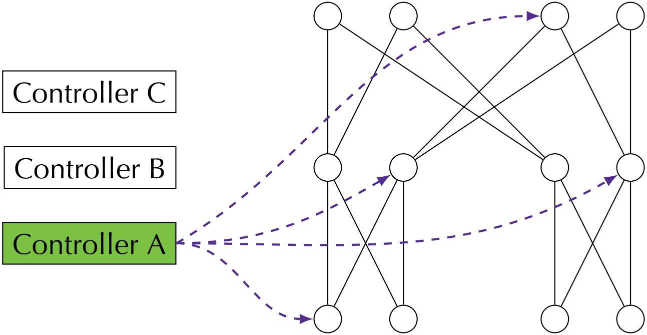
When a big flow is detected by an edge router...



Edge routers send flow info to controllers

Reroute

When a big flow is detected by an edge router...



Controller C

Controller B

Controller A

Controller install/remove flow-based routes at routers

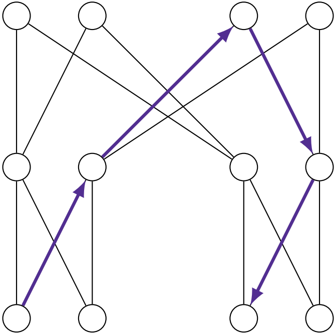
Reroute

When a big flow is detected by an edge router...

Controller C

Controller B

Controller A



Forwarding proceeds

Look-up tables

Look-up table at edge

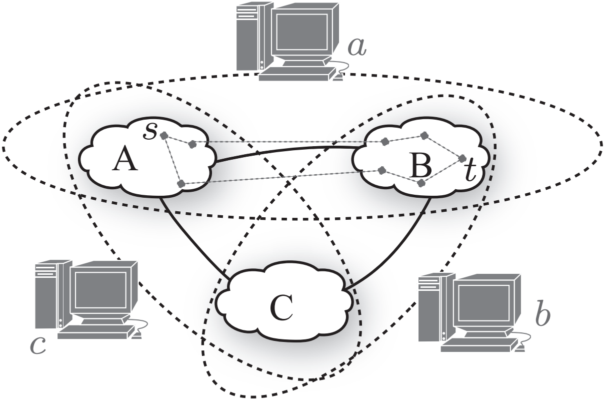
pair	controllers
(s, p)	$a,\ b,\ c$
(s, q)	$e,\ a,\ b$
(s, t)	$a,\ c,\ d$
$\vdots$	$\vdots$

Configuration of a devolved controller

pair	paths	controllers
(n_1, n_2)	p_1, p_2, p_3	a, b, c
(n_2, n_1)	p_4, p_5, p_6	b, a, c
(n_2, n_3)	p_7, p_8, p_9	a, c, d
$\vdots$	$\vdots$	$\vdots$

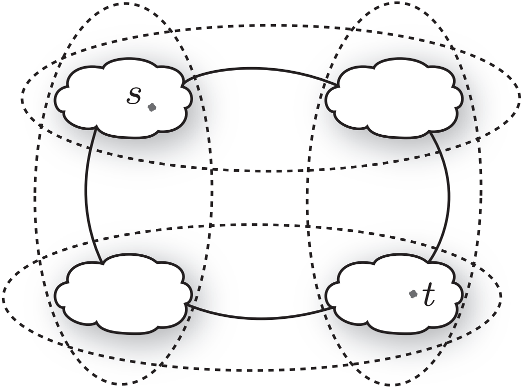
Configuration

Configuration of devolved controllers



Paths from *s* to *t* is known by controller *a*

Configuration of devolved controllers



No controller covers any path from s to t

How to configure?

Associate part of a network to a controller, so that

- All flows are reroutable by at least one controller
  - So that all requests can be fulfilled
- Minimize the number of links covered by any controller
  - So that monitoring cost can be minimized

How to configure?

Associate part of a network to a controller, so that

- All flows are reroutable by at least one controller
  - So that all requests can be fulfilled
- Minimize the number of links covered by any controller
  - So that monitoring cost can be minimized

Heuristic algorithm:

- Network of n nodes has $n(n-1)$ pairs
- A pair has k paths that a flow can use
- Iteratively allocate a pair into a controller
- Minimize num of links allocated to a controller

Path-partition heuristic algorithm

```
Data: Network $G = (V, E)$, $q$ =number of controllers
foreach $s, t \in V$ in random order do
    /* Retrieving a multipath from $s$ to $t$ */
    $M := k$ paths joining $s$ to $t$;
    /* Allocate into controllers */
    for $i := 1$ to $q$ do
        $c_i :=$ cost of adding multipath $M$ to controller $i$
    end
    $Q := \{1, \dots, q\}$;
    for $i := 1$ to $r$ do
        Allocate $M$ to controller $j = \arg\min_{j \in Q} c_j$;
        Remove $j$ from $Q$;
        Remove other controllers from $Q$ that violate the resilience constraints;
    end
end
```

Path-partition heuristic algorithm

- *k* paths are prepared for each pair
- Shuffle the pairs into random order
- Allocate each pair *M* (*k* paths) into the *r* best controllers
- Guided by a cost function:

$$c_i = \alpha \nu_i(M) + \mu_i$$

Weighting factor

# links in *M* that is not yet covered by *i* (Prefer a controller that already covers most of links in *M*)

# distinct links covered by *i* (Try to balance the number of links in each controller)

Partition-path heuristic algorithm

```
Data: Network $G = (V, E)$, $q$ =number of controllers
/* Partition links to controllers preliminarily */
foreach $i := 1$ to $q$ do
    Prepare set of links $\mathcal{E}_i \subset E$;
end
/* Enumerate multipaths and allocate into controllers */
foreach $s, t \in V$ in random order do
    foreach $i := 1$ to $q$ do
        $M_i :=$ Find $k$ paths for $(s, t)$ with priority to $\mathcal{E}_i$;
        $c_i :=$ cost of adding multipath $M_i$ to controller $i$
    end
    $Q := \{1, \ldots, q\}$;
    for $i := 1$ to $r$ do
        Allocate $M_j$ to controller $j = \arg\min_{j \in Q} c_j$;
        $\mathcal{E}_j := \mathcal{E}_j \cup \{e :$ for all links $e$ in $M_j\}$;
        Remove $j$ from $Q$;
        Remove other controllers from $Q$ that violate the resilience constraints;
    end
end
```

Partition-path heuristic algorithm

- Distribute links to controllers first (partition)
- Each controller finds k paths for a pair
- Select the best r according to the cost function

The two heuristic algorithms

Partition-path algorithm:
Fewer # links per controller

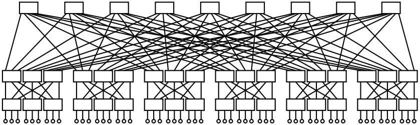
Path-partition algorithm:
Guarantees shortest-paths

The two heuristic algorithms

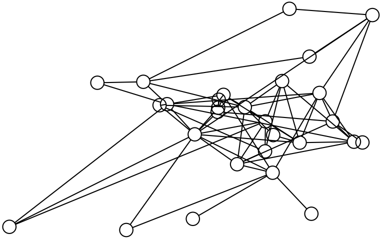
Partition-path algorithm:
Fewer # links per controller

Path-partition algorithm:
Guarantees shortest-paths

(Good for regular topologies)



(Good for irregular topologies)



Resilience

Redundancy

- Paths for every pair is known by *r* controllers
- At any moment, only *one* of the *r* controllers is *active*
- When the active one fails, another controller takes over
- Controllers talk to each other with *heartbeat protocol*

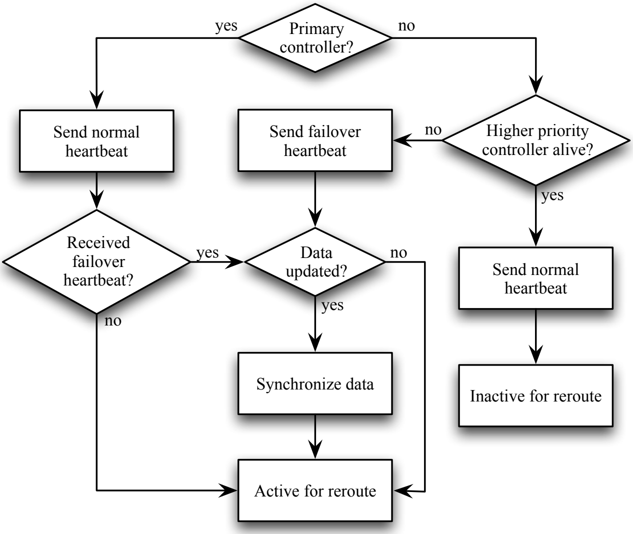
Redundancy

Configuration of a devolved controller

pair	paths	controllers
(n_1, n_2)	p_1, p_2, p_3	a, b, c
$\vdots$	$\vdots$	$\vdots$

- Controller priority: $a > b > c$
- Controller a is the primary controller for pair (n_1, n_2)
- Controllers b and c are the secondary controller
- If all controllers are healthy, a is the active controller

Flow chart of failover algorithm

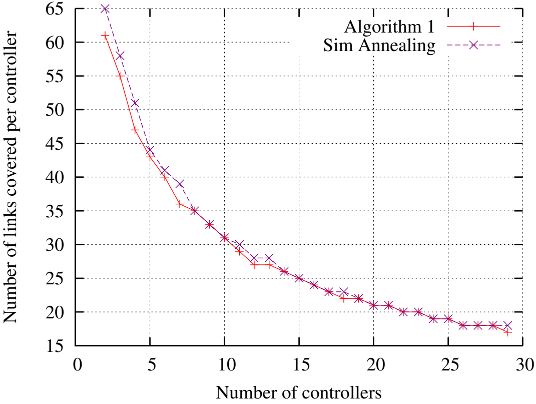


Heartbeat messages

Normal heartbeat: *"I am X. I am alive."*
Failover heartbeat: *"I am X. I am taking over controllers Y and Z"*

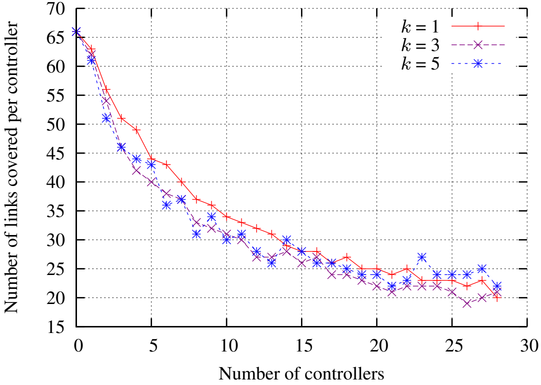
Evaluation

Optimality



Heuristic algorithm is as good as simulated annealing

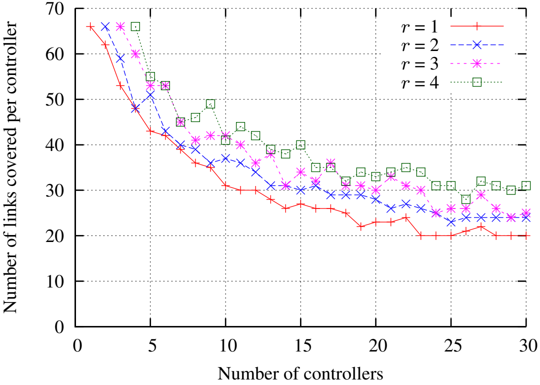
Number of paths per pair



k = multiplicity of paths

More paths per pair does not significantly increase coverage size

Redundancy



r = redundancy factor

Redundancy is almost free

Conclusion

Conclusion

- Devolved controllers is a viable concept
- Heuristic algorithms proposed to help configuration
- Protocol on dynamic flow reroute with resilience