Burst-tolerance in Jellyfish

A New Routing Algorithm

By: Liwei Cui, Mou Zhang, Yifeng Yin

Content

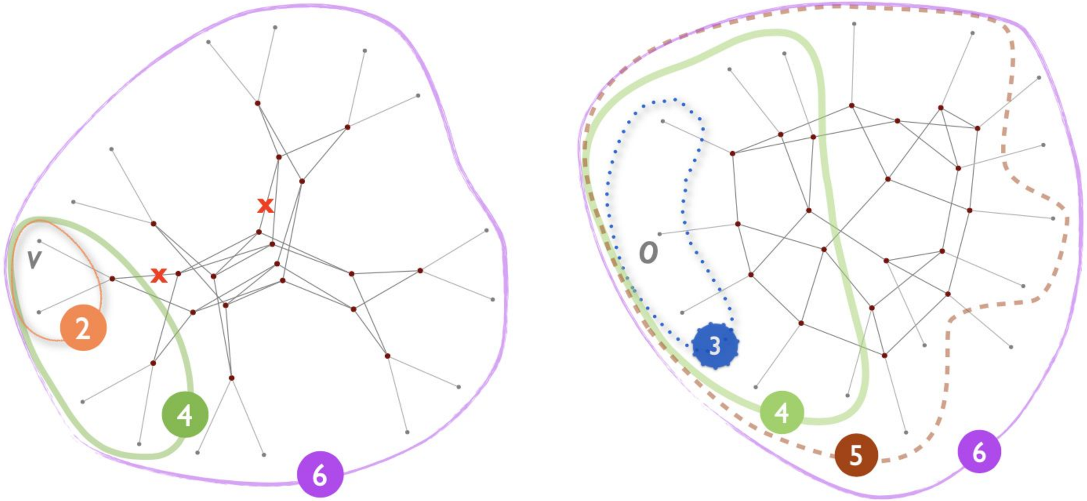
1. Problem: burst traffic
2. Recap: Jellyfish
3. Jellyfish’s Approach
4. Motivation
5. Our Progress
6. Next Steps
7. Acknowledgement

WHAT IS BURST TRAFFIC?

“Burst traffic”, is a sudden, uncertain, unpredictable traffic peak. Cause big trouble for network.

Recap: Jellyfish

Forget about structure – let’s have no structure at all!



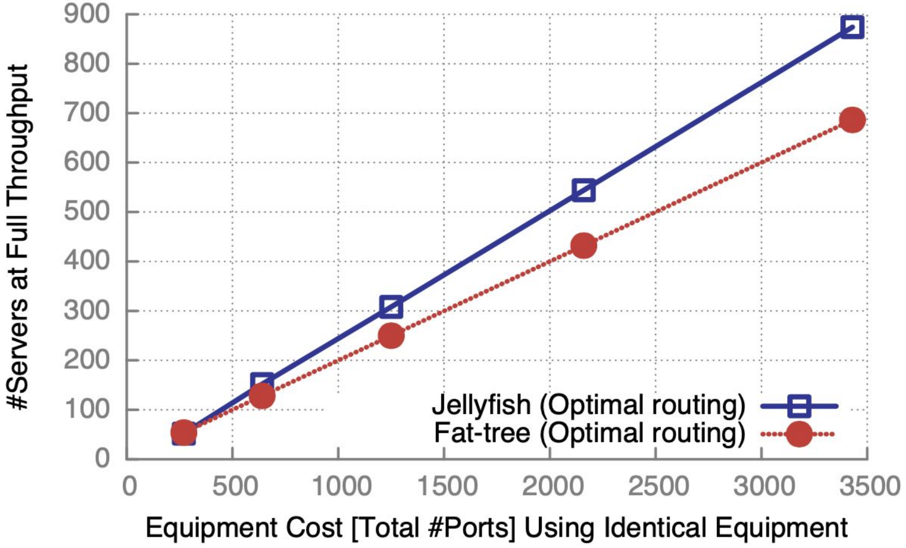
(a)

(b)

Networking Data Centers, Randomly. A Singla et al., NSDI 2012

Recap: Jellyfish

Identical Equipment, **25%** more throughput!

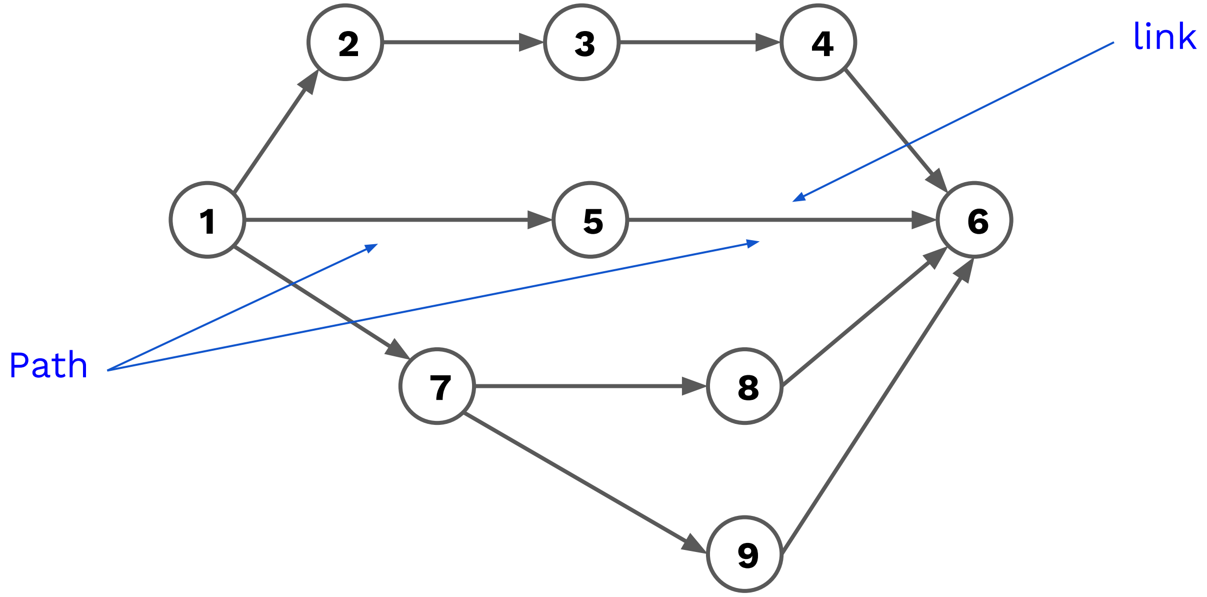


Networking Data Centers, Randomly. A Singla et al., NSDI 2012

> “While the above experiments establish that Jellyfish topologies have **high capacity**, it remains unclear whether this potential can be realized in real networks.”

Section 5 in Jellyfish Paper

ECMP Does Not Work

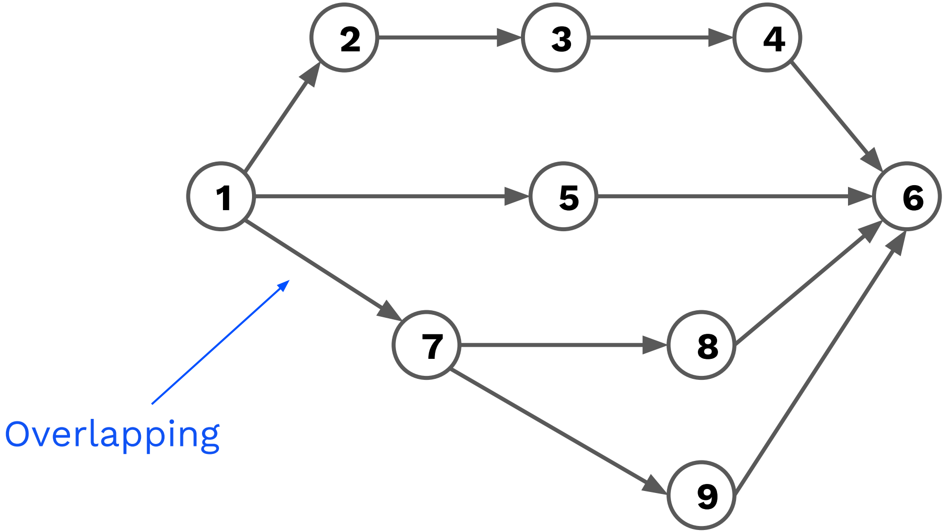


1 → 6: only one(1) path is available; it uses two(2) links (**1 → 5**, **5 → 6**)

WHAT IS JELLYFISH’S APPROACH?

“K-shortest path routing”. It is not only about a shortest path but also about next k - 1 shortest paths (longer, maybe).

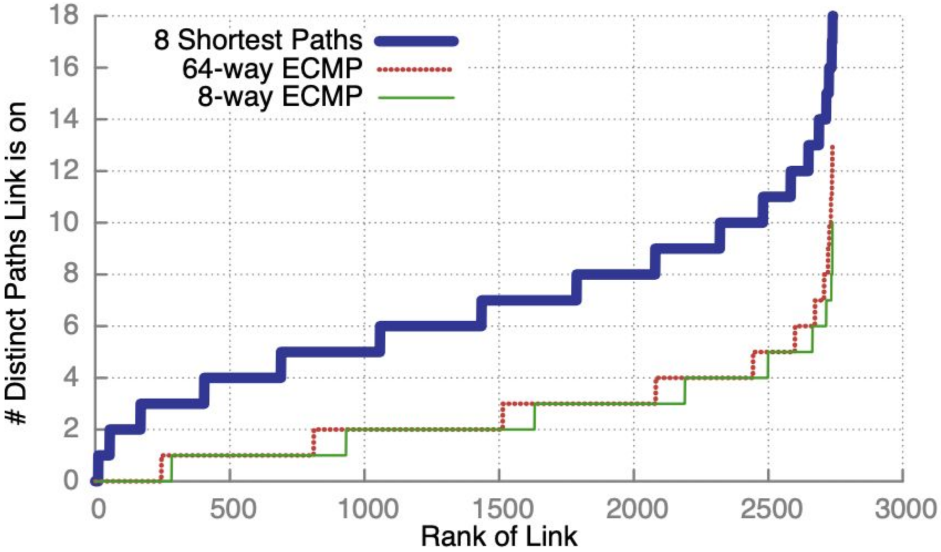
3-shortest Paths Works (Much) Better



1 → 6: three(3) paths are available:

1. **1 → 5 → 6**
2. **1 → 7 → 8 → 6**
3. **1 → 7 → 9 → 6**

Path Diversity



For each **link**, we count the number of **distinct paths** it is on.

The more the path diversity, the less path overlap may occur.

Networking Data Centers, Randomly. A Singla et al., NSDI 2012

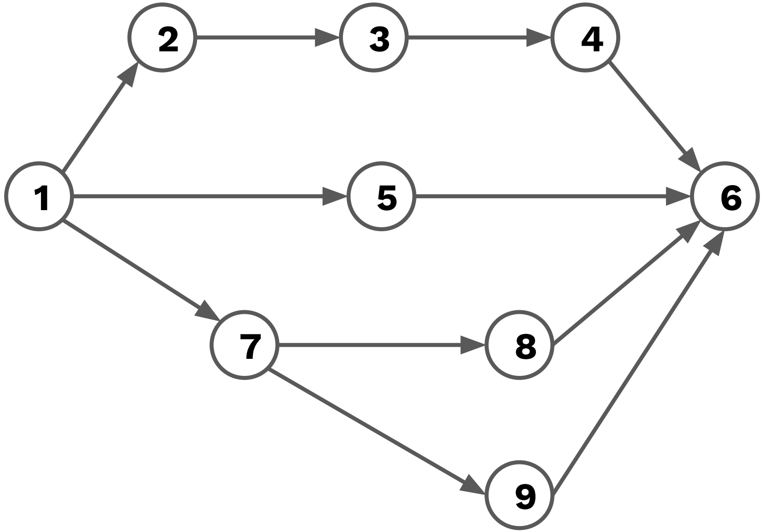
Motivation

What if we maximize the path diversity, to achieve higher throughput?

WHAT IS OUR ALGORITHM?

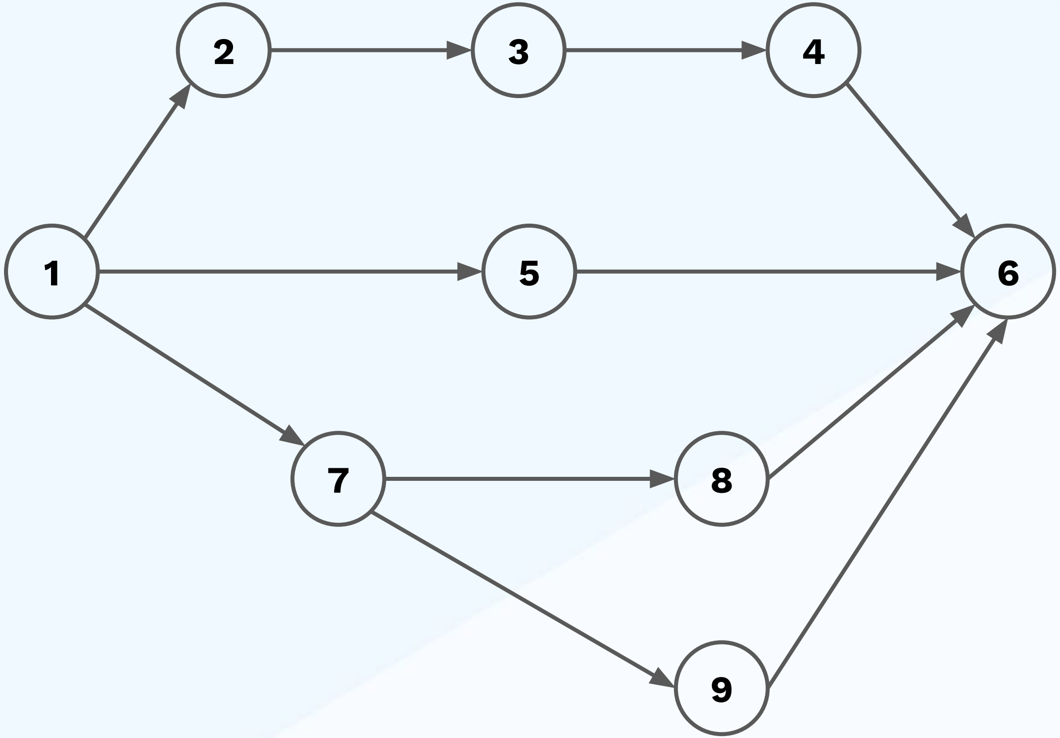
“K-non-overlapping path routing”. It is guarantees all links on paths from A to B have no overlapping (longer, likely).

3-non-overlapping Paths Works Best



1 → 6: three(3) paths are available:

1. **1 → 5 → 6**
2. **1 → 7 → 8 → 6**
3. **1 → 2 → 3 → 4 → 6**



ECMP (Shortest Paths)

1 → 6: One(1) path is available:

1. **1 → 5 → 6**

3-shortest Paths

1 → 6: three(3) paths are available:

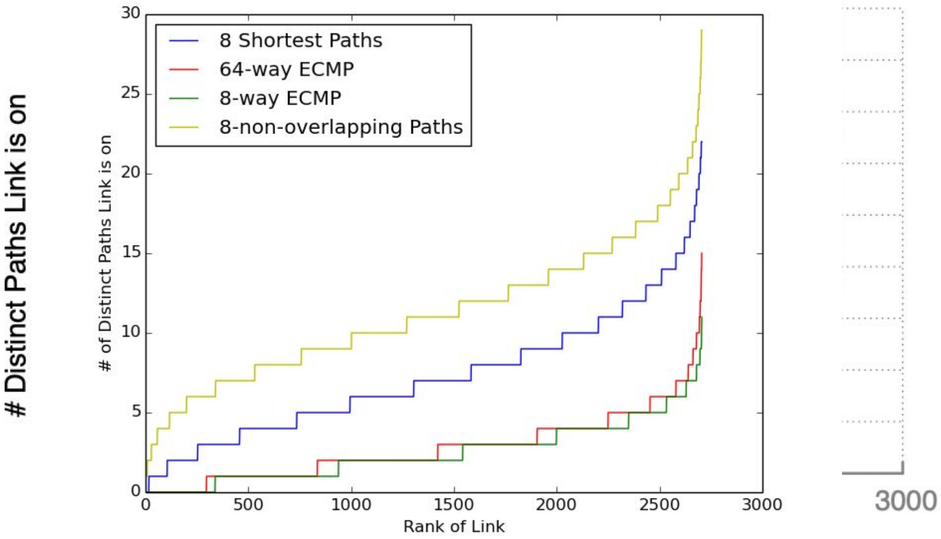
1. **1 → 5 → 6**
2. **1 → 7 → 8 → 6**
3. **1 → 7 → 9 → 6**

3-non-overlapping Paths

1 → 6: three(3) paths are available:

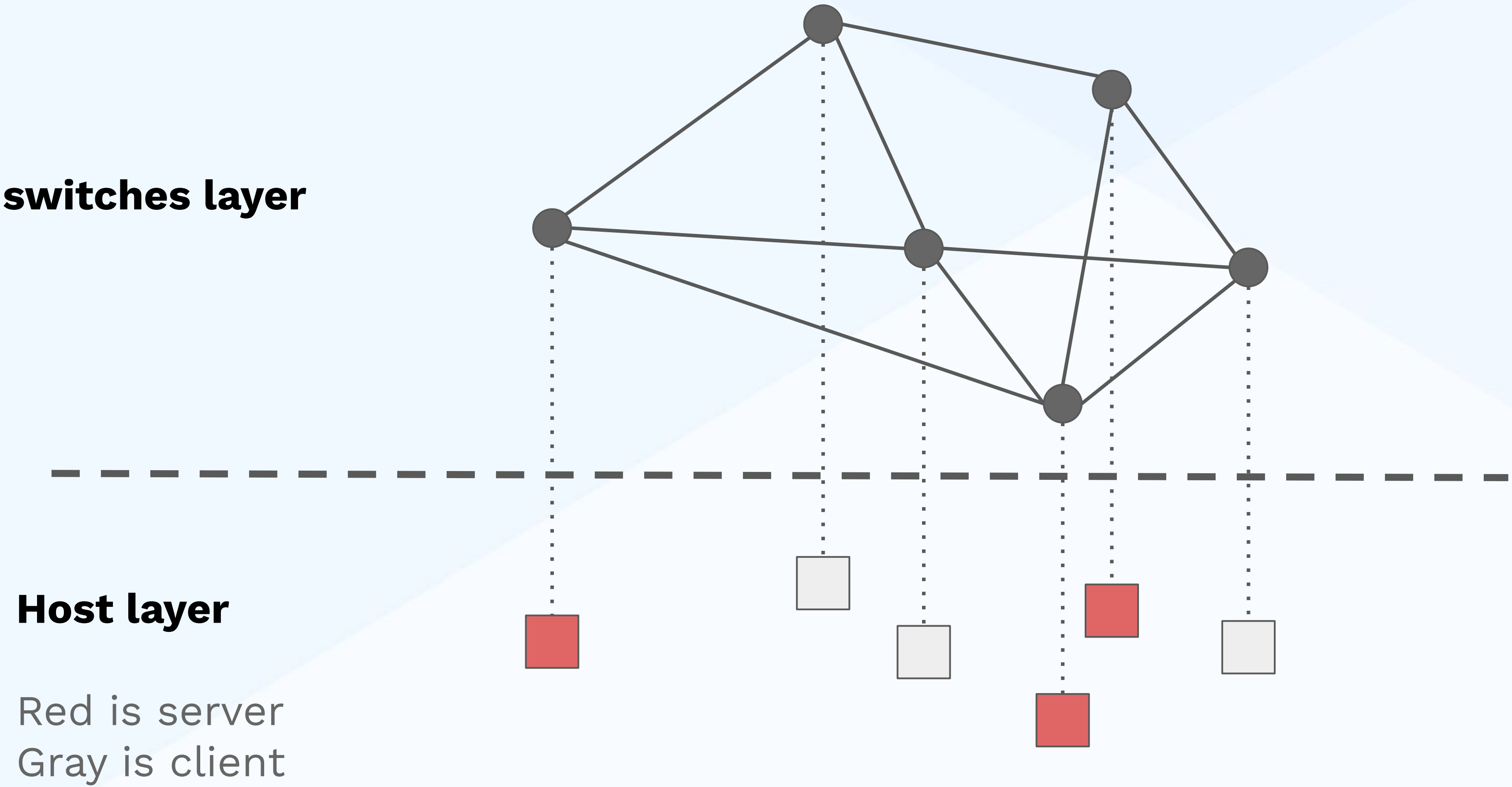
1. **1 → 5 → 6**
2. **1 → 7 → 8 → 6**
3. **1 → 2 → 3 → 4 → 6**

Path Diversity



For each **link**, we count the number of **distinct paths** it is on.

The more the path diversity, the less path overlap may occur.



switches layer

Host layer

Red is server
Gray is client

Experiment Configuration

50

switches

8

ports connecting peer switches

10 Mbps

bandwidth between switches

12

random hosts as servers

12

random hosts as clients

100 Mbps

bandwidth between switches and hosts

↑23%

8-Shortest-Paths achieving **27.98 Mbps** throughput
8-Non-overlapping achieving **34.58 Mbps** throughput

Our Progress

Reproduce Jellyfish

We leveraged several libraries (Mininet, Pox, RipL, RipL-POX) and open-source code to reproduce the Jellyfish network and k-shortest-paths routing

New Routing Algorithm

We put forward and implemented a new routing algorithm (*Non-overlapping Path Algorithm*), which guarantees all links on paths from A to B have no overlapping

Comparison and Test

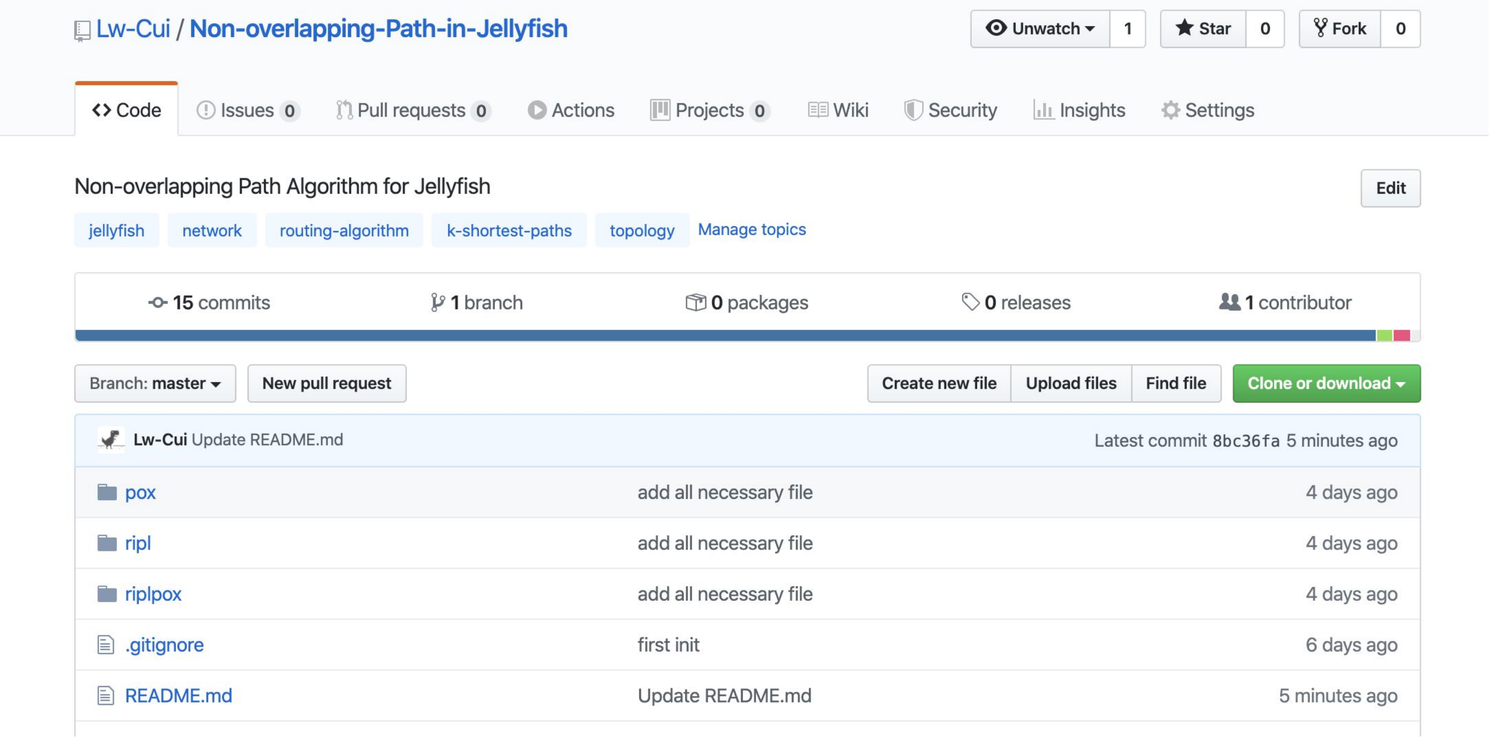
We compared the path diversity between non-overlapping routing with k-shortest-path & tested the average throughput per server, achieving 23.6% more throughput

Deployment on Cloud

We set up our experimental environment on the Google Cloud Platform, which makes it easier to conduct future experiments.

> **Talk is cheap. Show me the code.**
>
> Linus Torvalds

Non-overlapping-Path-in-Jellyfish @ Github



Lw-Cui / Non-overlapping-Path-in-Jellyfish

Non-overlapping Path Algorithm for Jellyfish

jellyfish network routing-algorithm k-shortest-paths topology

15 commits · 1 branch · 0 packages · 0 releases · 1 contributor

Lw-Cui Update README.md — Latest commit 8bc36fa 5 minutes ago

pox	add all necessary file	4 days ago
ripl	add all necessary file	4 days ago
riplpox	add all necessary file	4 days ago
.gitignore	first init	6 days ago
README.md	Update README.md	5 minutes ago

WHAT IS NEXT STEPS?

RELEASE RESTRAINT

Release strict non-overlapping restraint to balance the path length and path diversity;

MORE EXPERIMENT

Perform more tests under various topologies and circumstances.

Acknowledge

1. **Mininet** library for network emulation
2. **POX** library for OpenFlow controller
3. **RipL** library for simplifying data center code
4. **RipL-POX** library for controller built on RipL
5. **Austin Poore** and **Tommy Fan's** repo for inspiration to reproduce Jellyfish

Burst-tolerance in Jellyfish

Thank you for listening