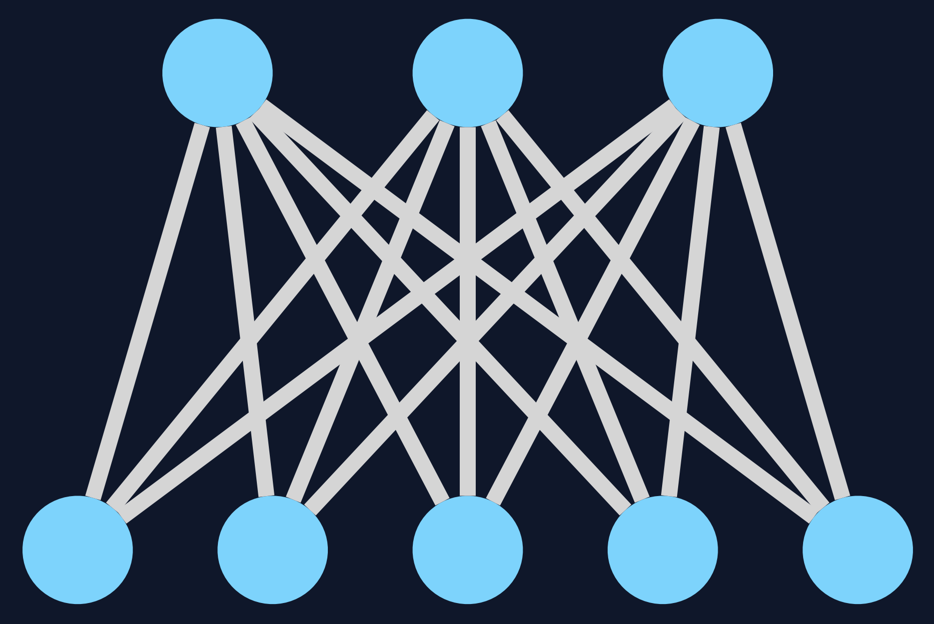
Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

Anchengcheng Zhou
Princeton University

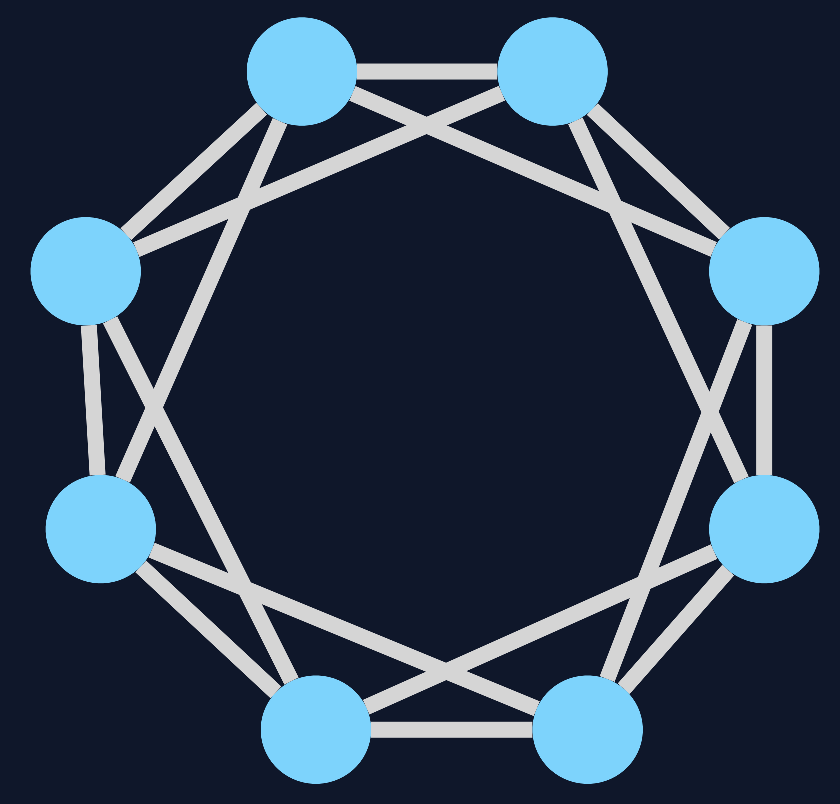
Vipul Harsh, Sangeetha Abdu Jyothi, Maria Apostolaki, Brighten Godfrey

Leaf-spine, the default for small-scale data centers today, is suboptimal



+56% throughput

No hardware upgrade

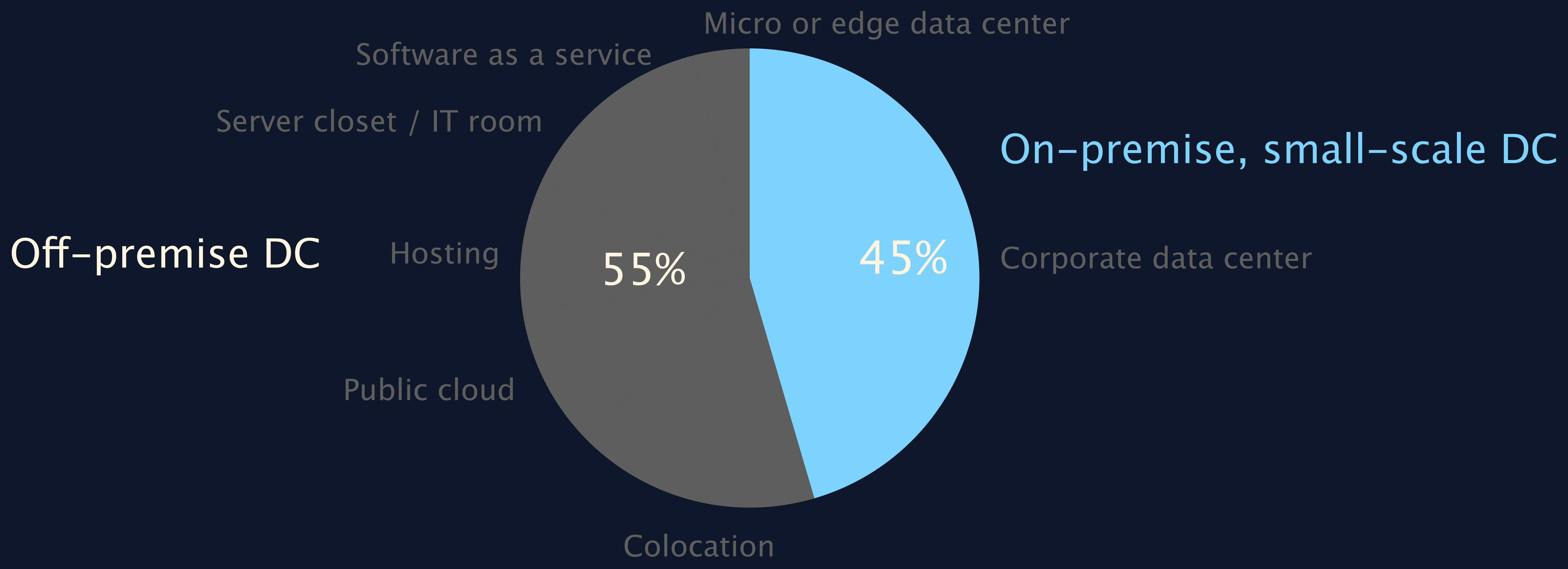


Status quo: Leaf-spine

Our work: Starfish

Uptime Institute

"At least 45% of all enterprise IT workloads are run in small-scale data centers."

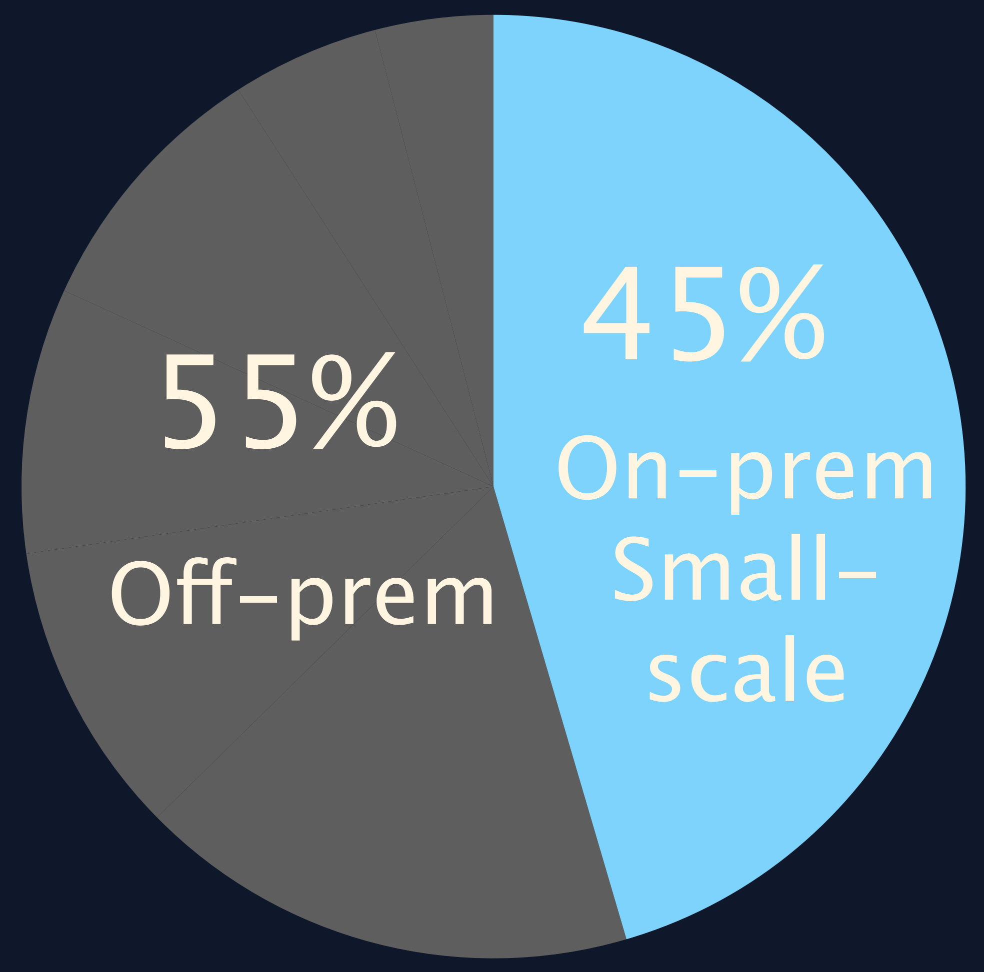


(Small-scale data centers) are prevalent

- 10x – 100x racks
- 100x – 1,000x servers

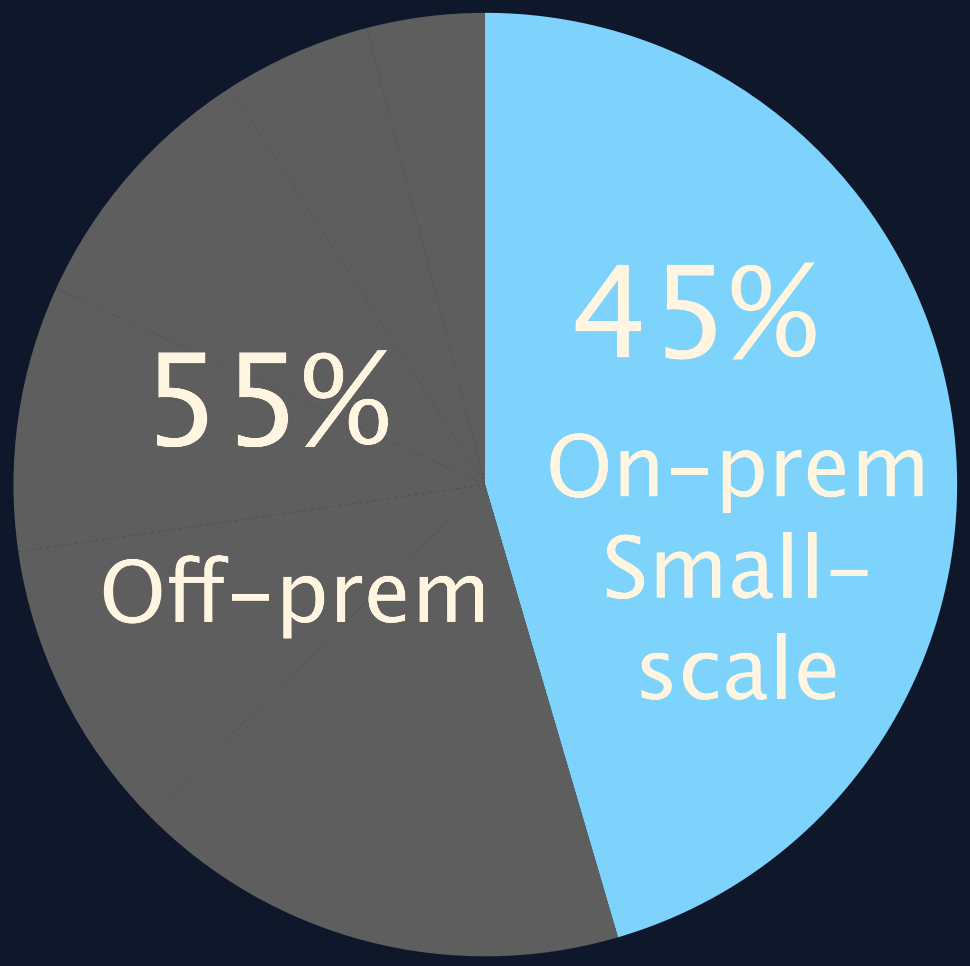
- Enterprises' private DCs VISA
- Edge clouds
- Internet exchange points

……



Small-scale data centers are prevalent **but overlooked**

None of the existing literature looks at these small-scale data centers



Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

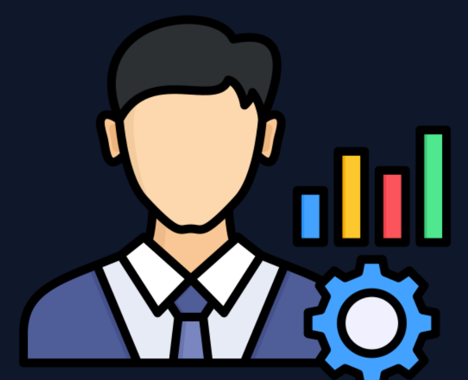
- **Motivation** Small-scale DCs: important but overlooked
- **Challenge** Trilemma: performance – fault tolerance – deployability
- **Starfish** Topology-routing co-design
- **Evaluation** Simulation results: high-performing & resilient to failures

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

- **Motivation** Small-scale DCs: important but overlooked
- **Challenge** Trilemma: performance - fault tolerance - deployability
- **Starfish** Topology-routing co-design
- **Evaluation** Simulation results: high-performing & resilient to failures

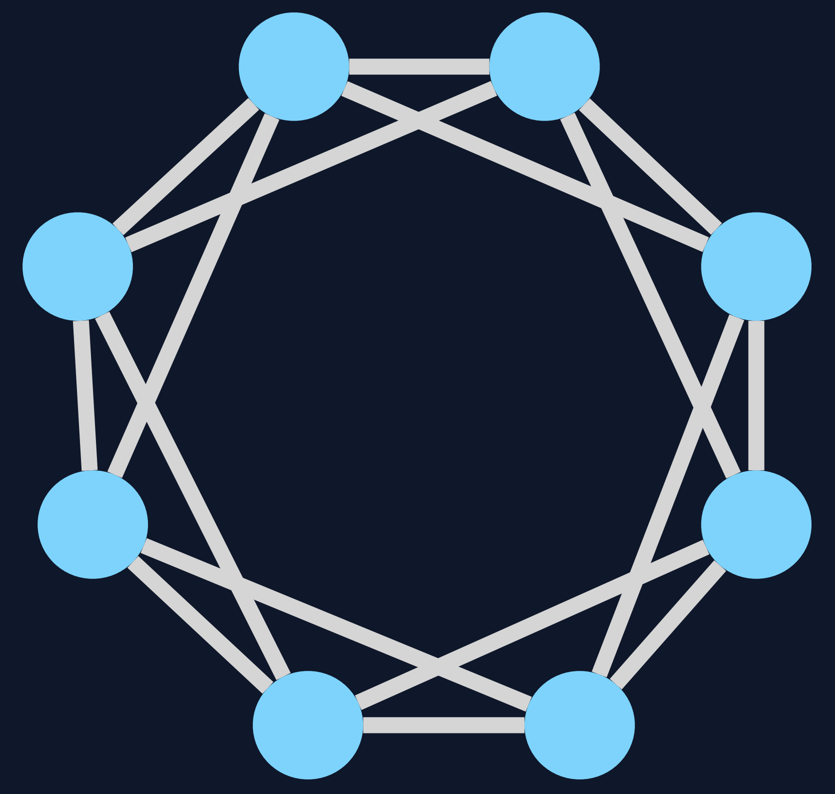
Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

- **Motivation** Small-scale DCs: important but overlooked
- **Challenge** Trilemma: performance - fault tolerance - deployability
- **Starfish** Topology-routing co-design
- **Evaluation** Simulation results: high-performing & resilient to failures

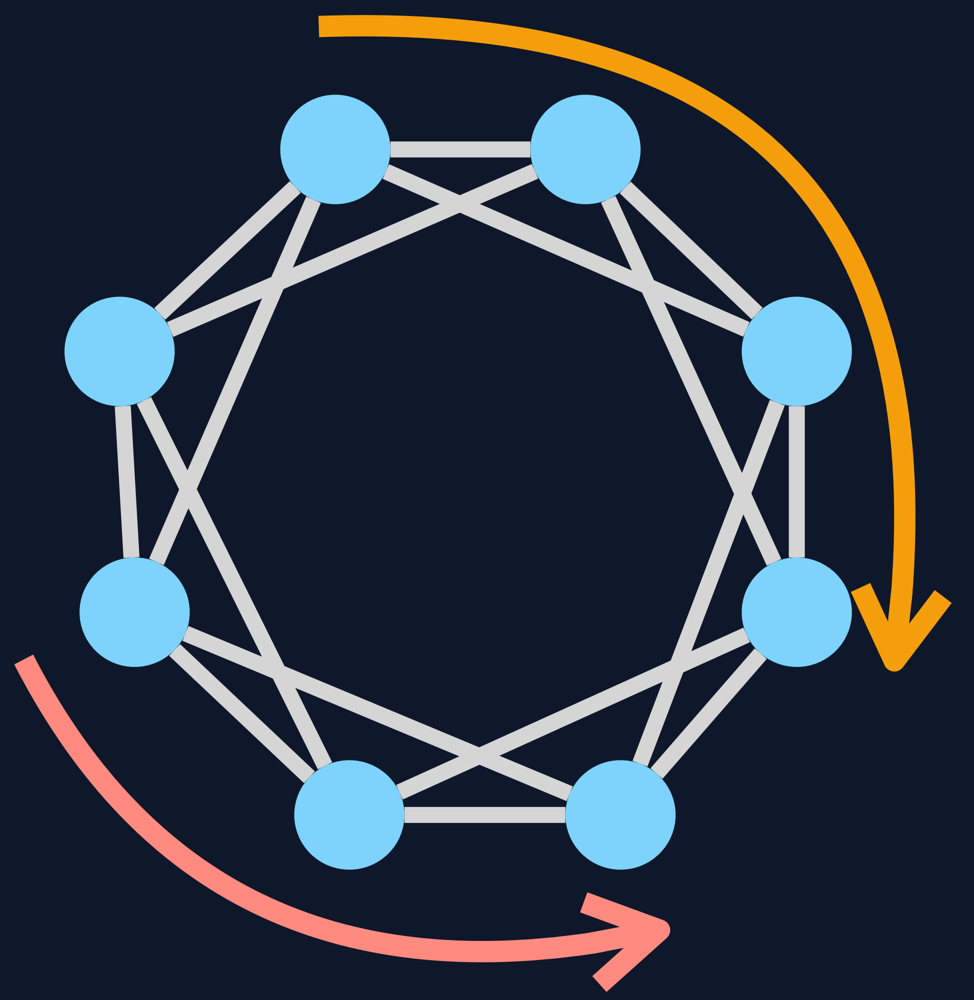


Small-DC
Network Operator

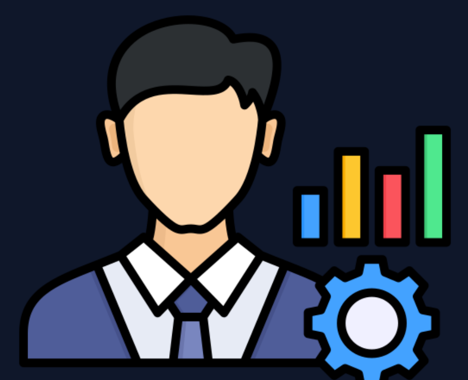
"I want to practically get the most performance out of my network while staying failure-resilient."



How to connect the servers & switches?

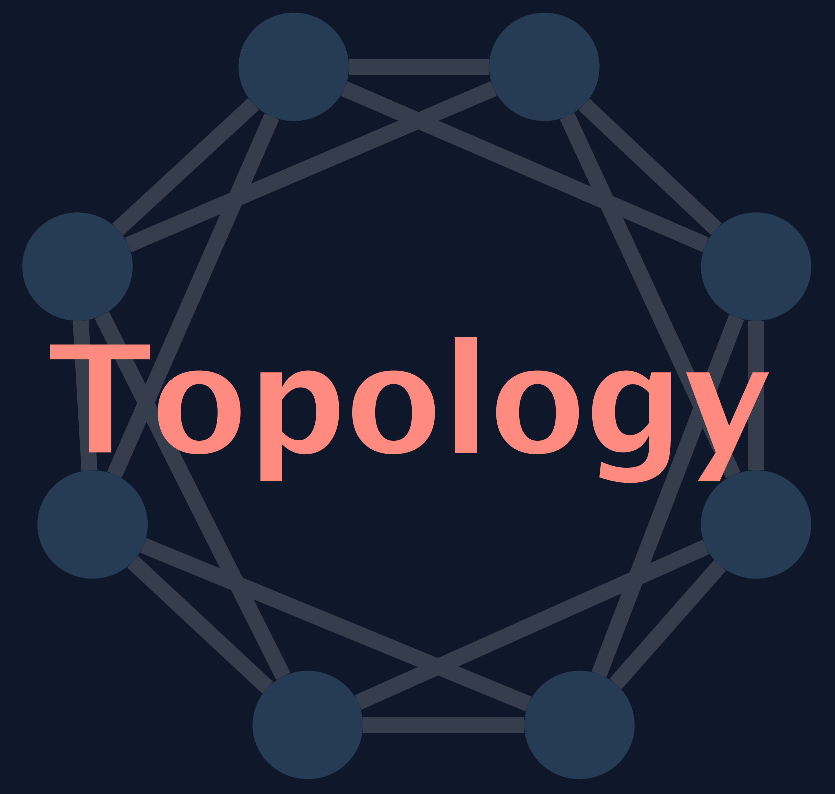


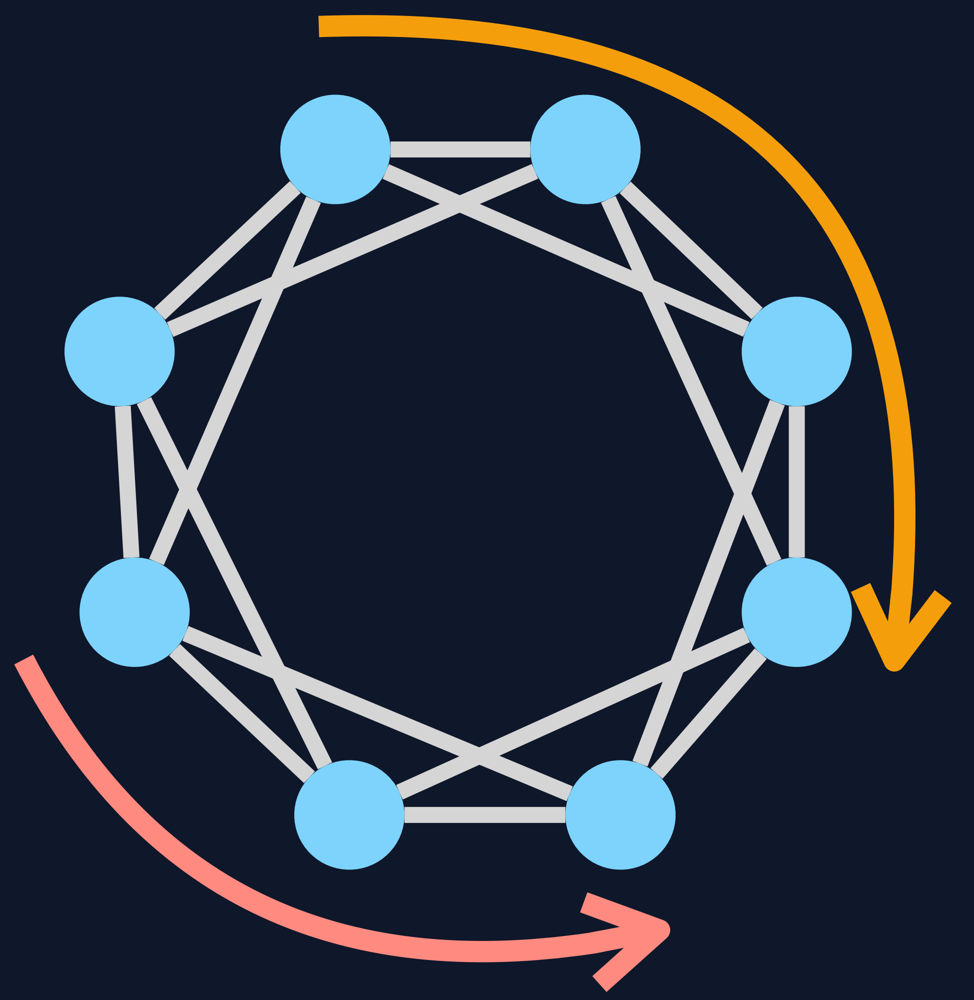
What entries to install in switches for each flow?



Small-DC
Network Operator

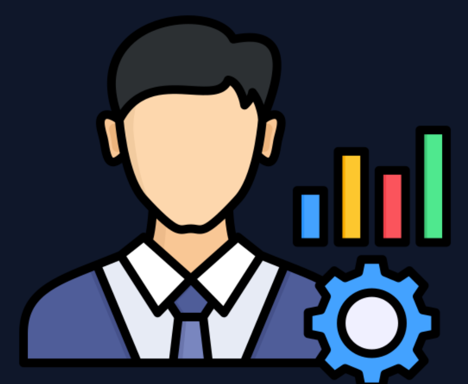
“I want to practically get the most performance out of my network while staying failure-resilient.”





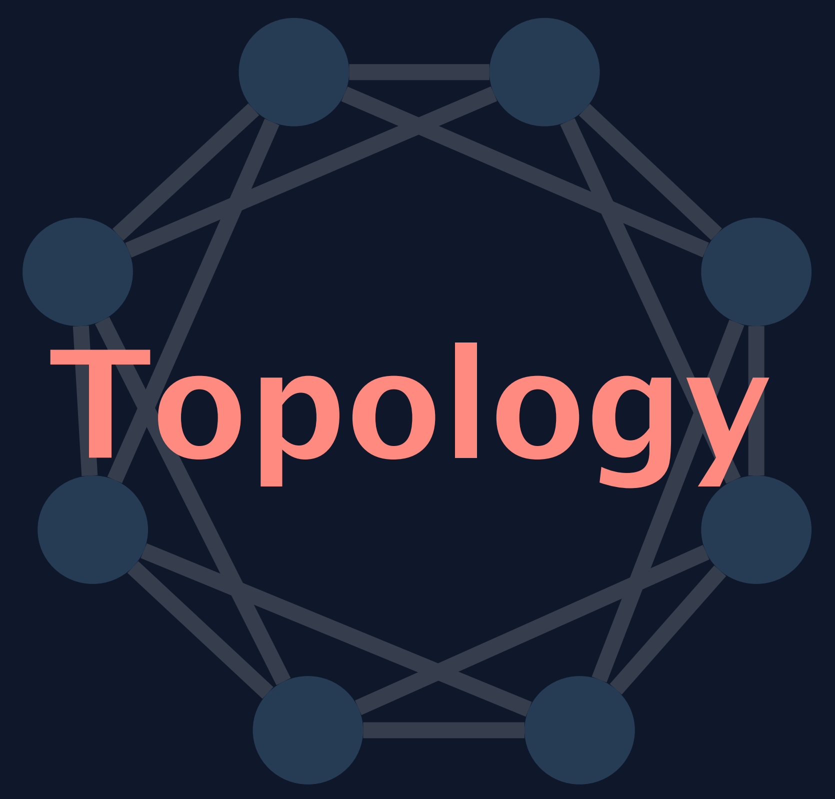
How to connect the servers & switches?

What entries to install in switches for each flow?

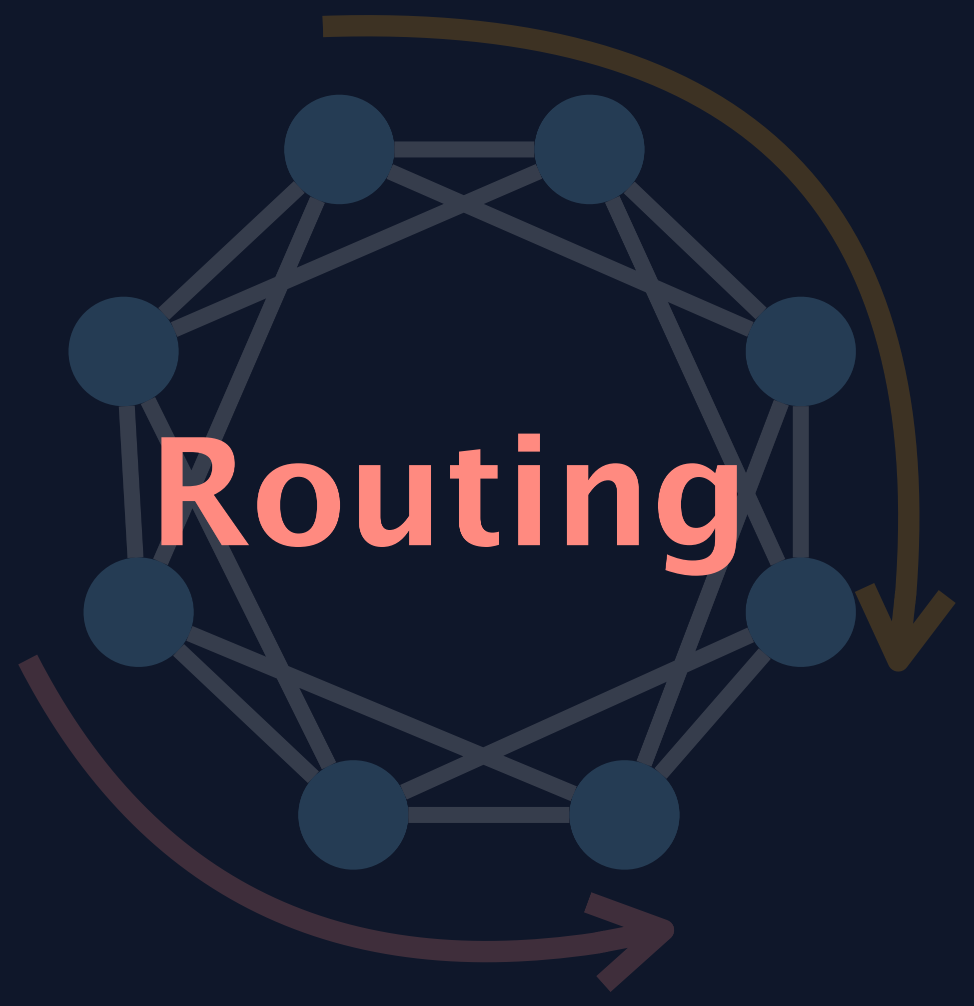


Small-DC
Network Operator

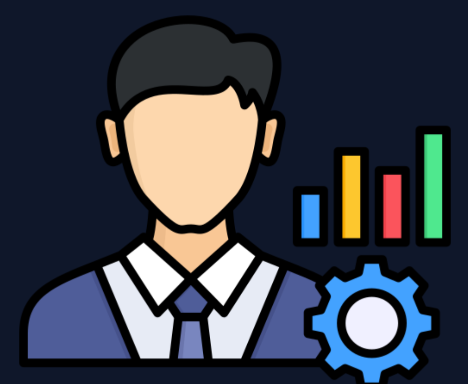
"I want to practically get the most performance out of my network while staying failure-resilient."



How to connect the servers & switches?



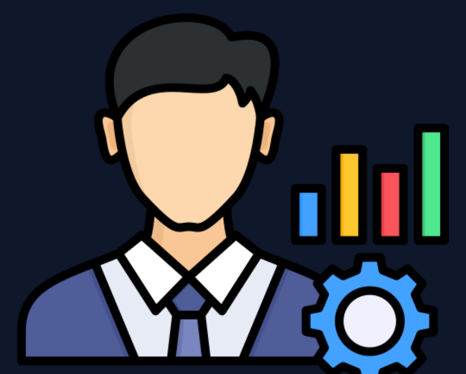
What entries to install in switches for each flow?



Small-DC
Network Operator

"I want to practically get the most performance out of my network while staying failure-resilient."

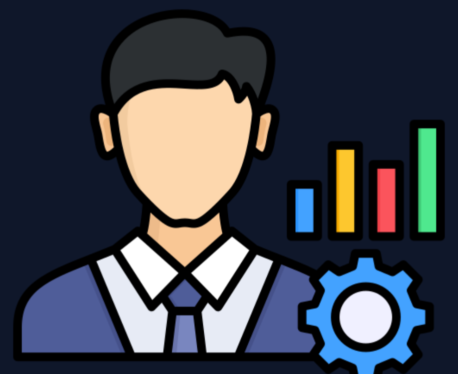
- **High performance** — Support high throughput & low latency under both uniform & skewed traffic



Small-DC
Network Operator

"I want to practically get the most performance out of my network while staying failure-resilient."

- **High performance** — Support high throughput & low latency under both uniform & skewed traffic
- **Fault tolerance** — Retain high performance under random link & switch failures



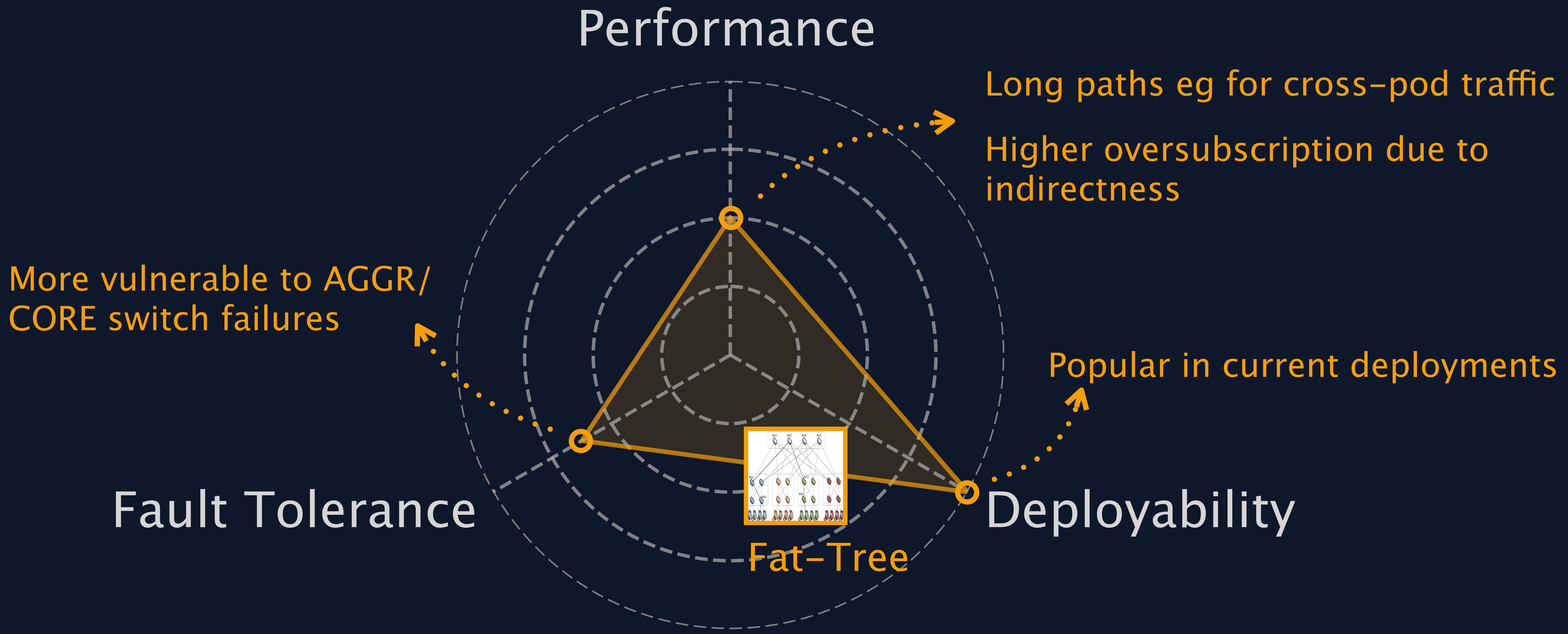
Small-DC
Network Operator

"I want to practically get the most performance out of my network while staying failure-resilient."

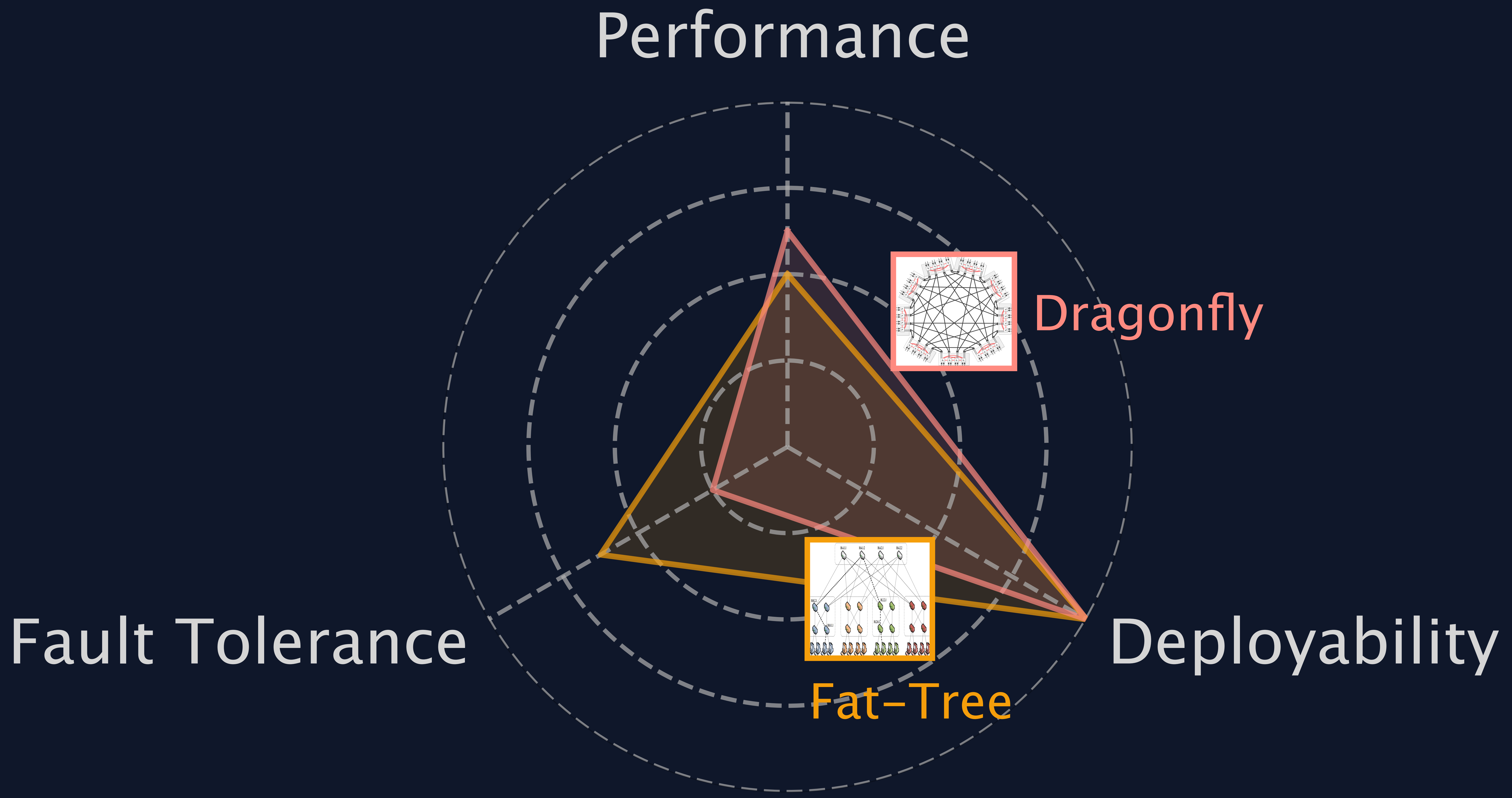
- **High performance** — Support high throughput & low latency under both uniform & skewed traffic
- **Fault tolerance** — Retain high performance under random link & switch failures
- **Deployability** — Enable practical routing & ease of management

Existing work fails to achieve the three simultaneously

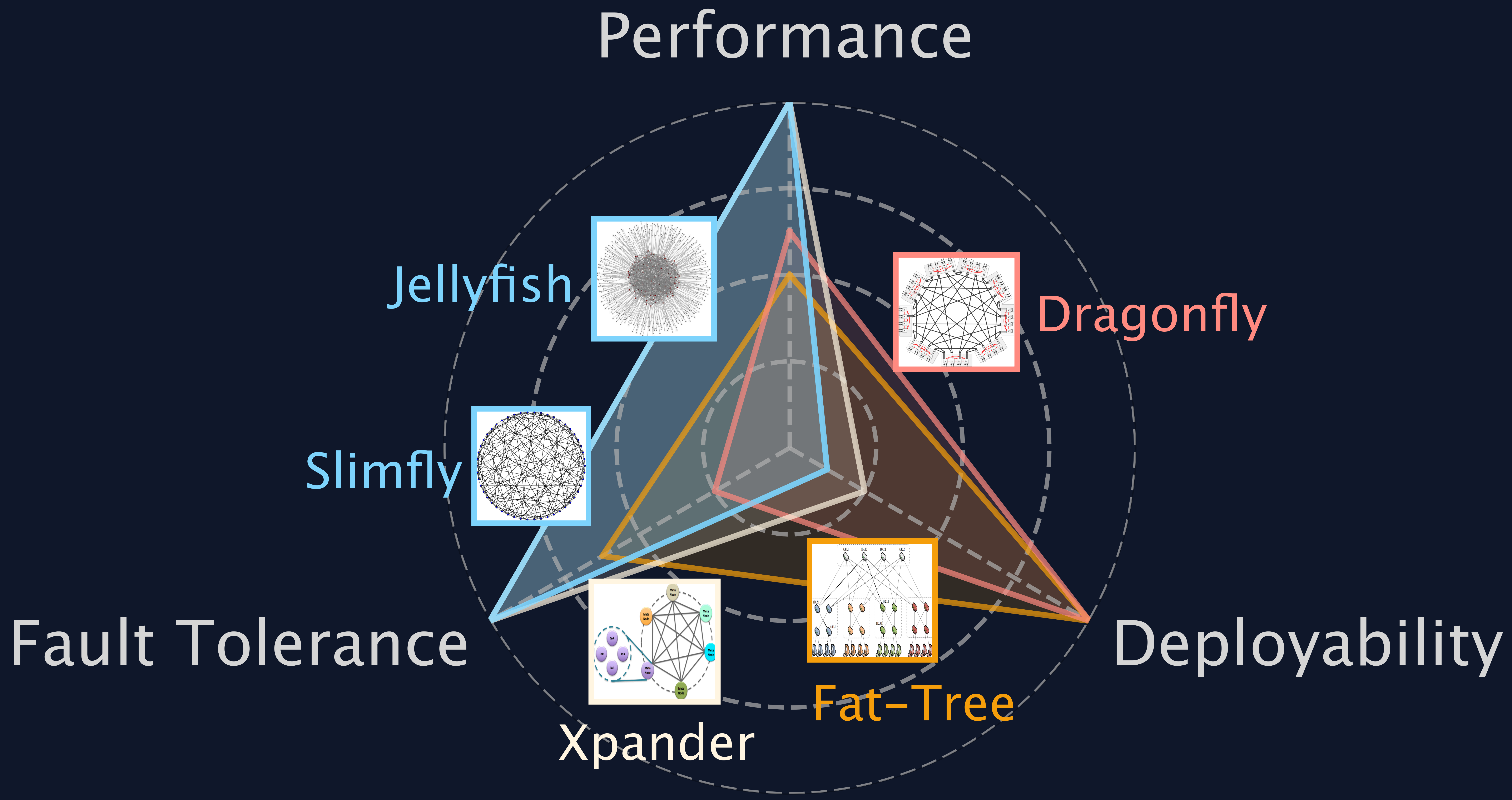
Existing work fails to achieve the three simultaneously



Existing work fails to achieve the three simultaneously



Existing work fails to achieve the three simultaneously

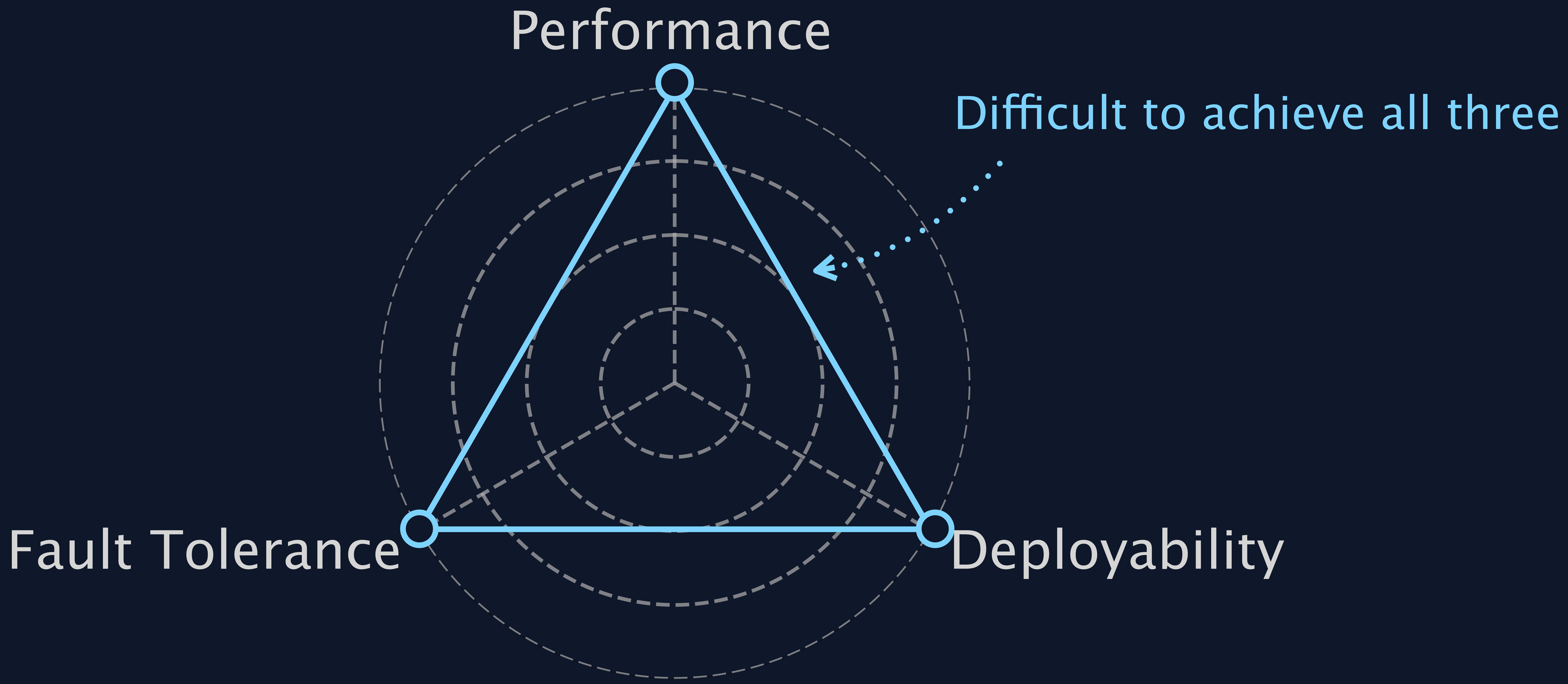


Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

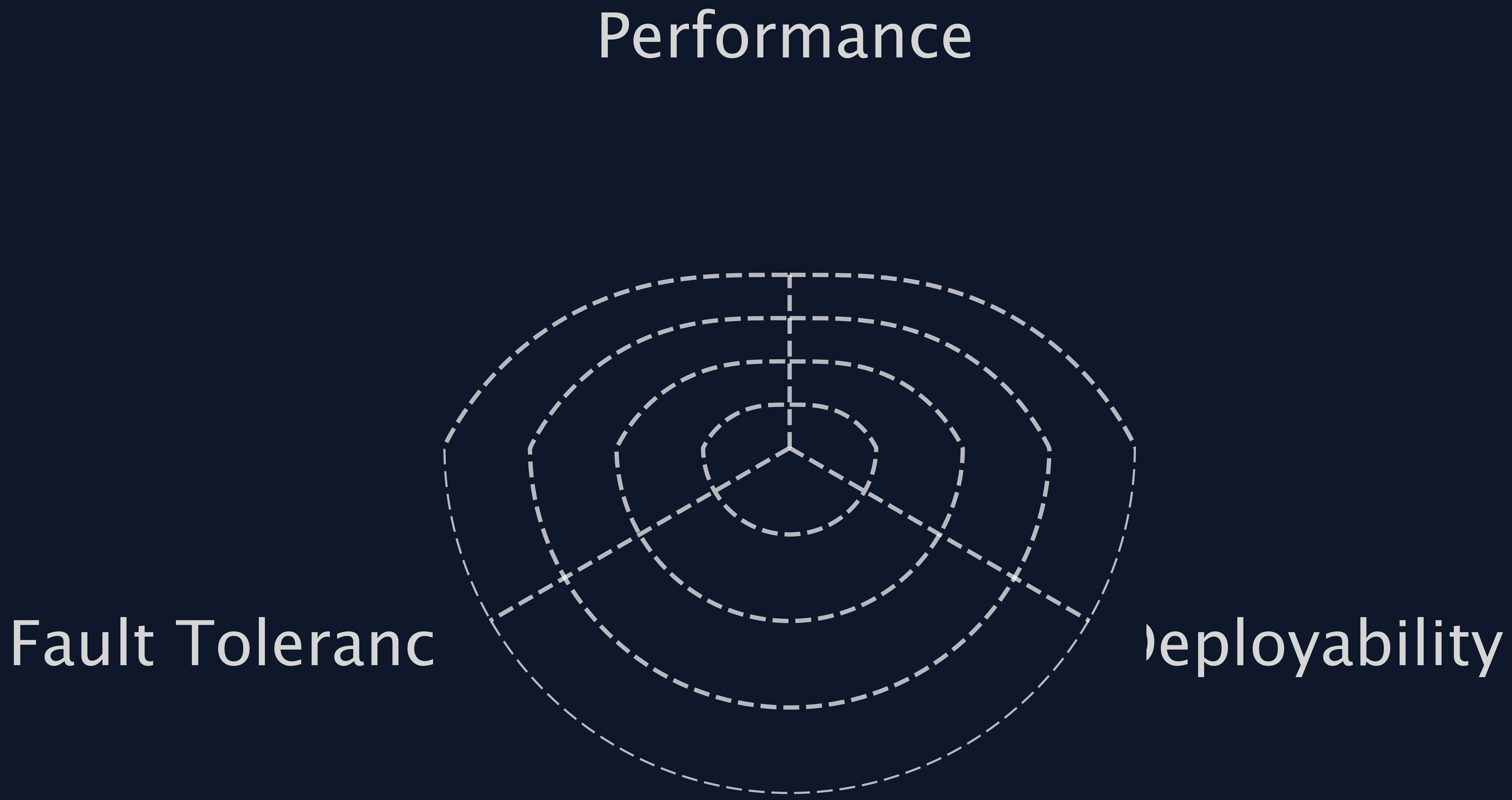
- **Motivation** Small-scale DCs: important but overlooked
- **Challenge** Trilemma: performance - fault tolerance - deployability
- **Starfish** Topology-routing co-design
- **Evaluation** Simulation results: high-performing & resilient to failures

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

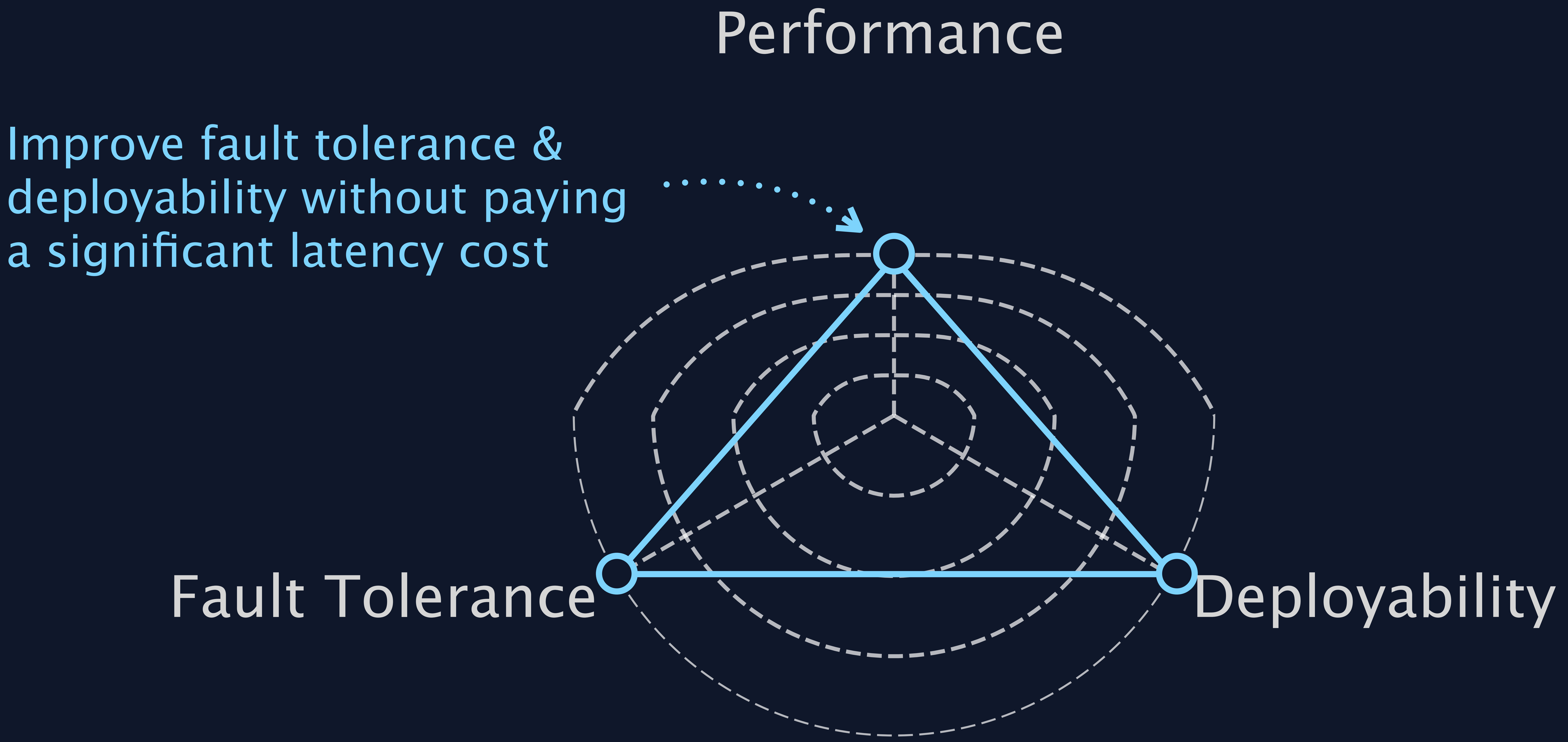
- **Motivation** Small-scale DCs: important but overlooked
- **Challenge** Trilemma: performance - fault tolerance - deployability
- **Starfish** Topology-routing co-design
- **Evaluation** Simulation results: high-performing & resilient to failures


Performance
Difficult to achieve all three
Fault Tolerance
Deployability

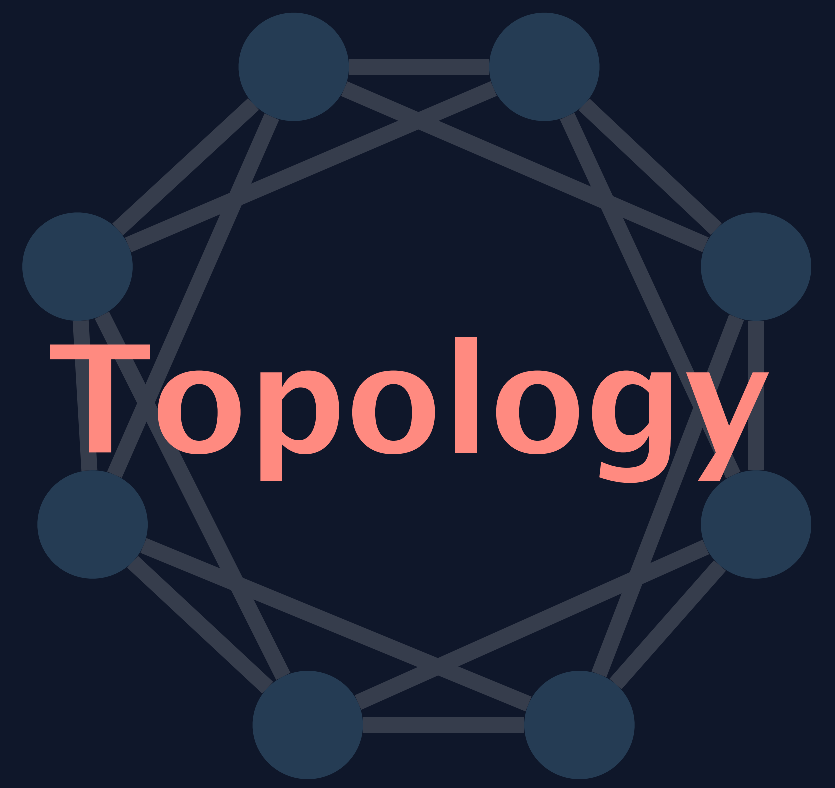
Opportunity at small scale: path length headroom



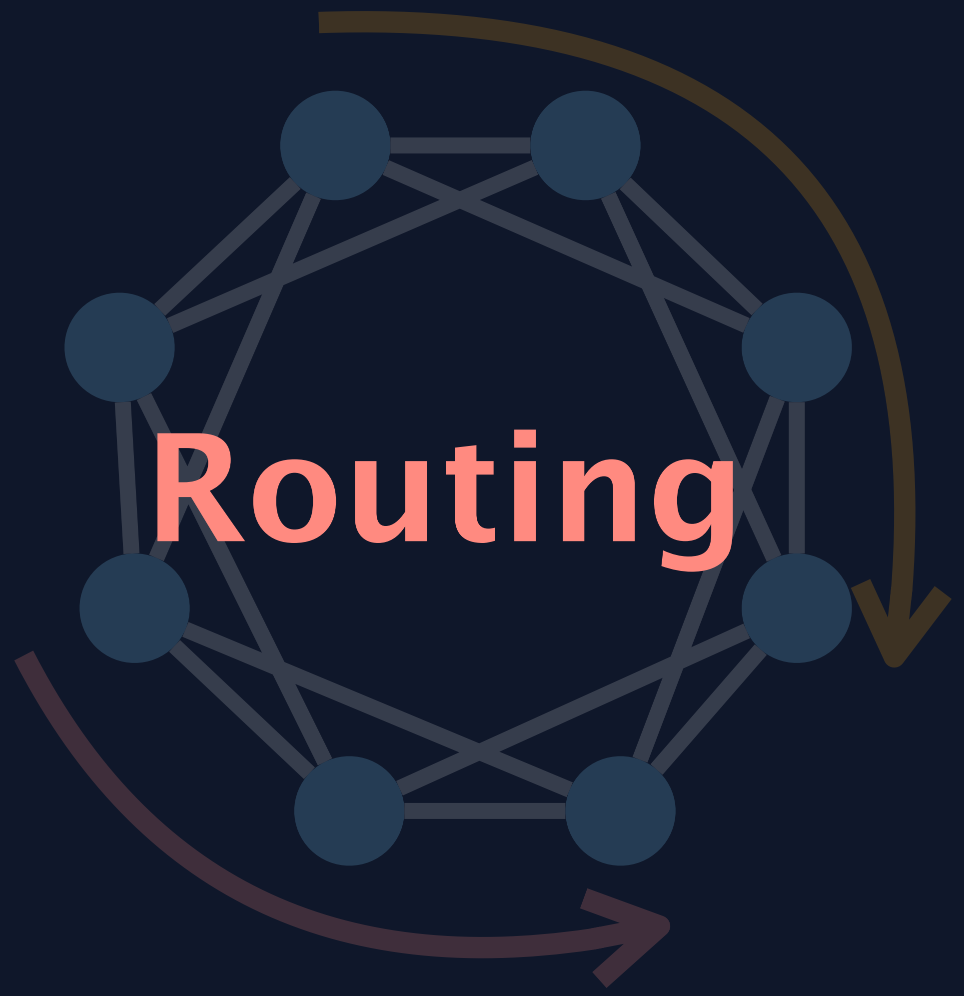
Opportunity at small scale: path length headroom



Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

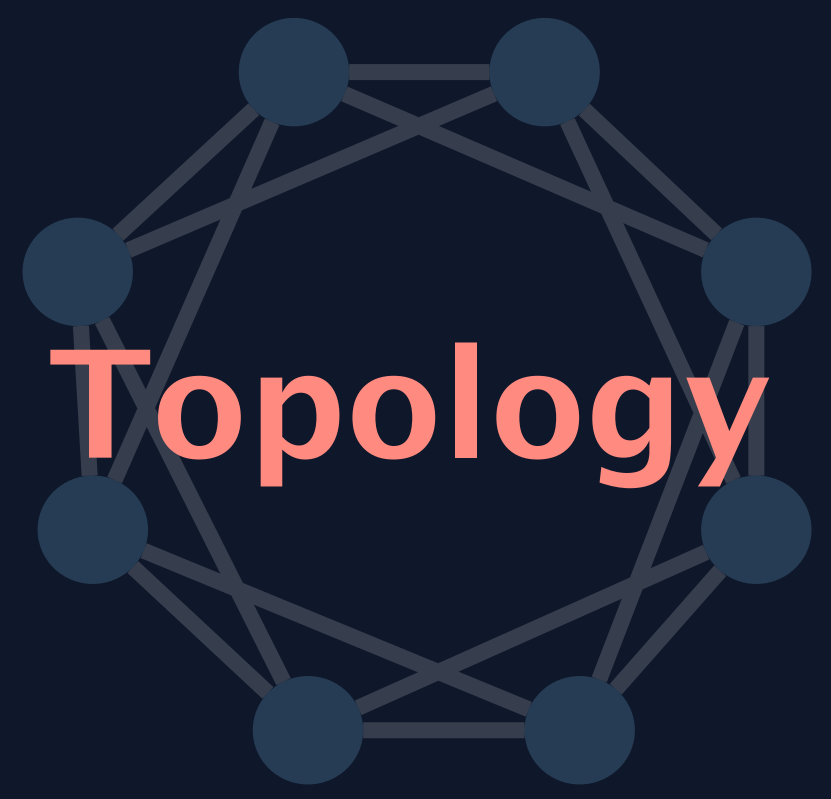


How to connect the servers & switches?

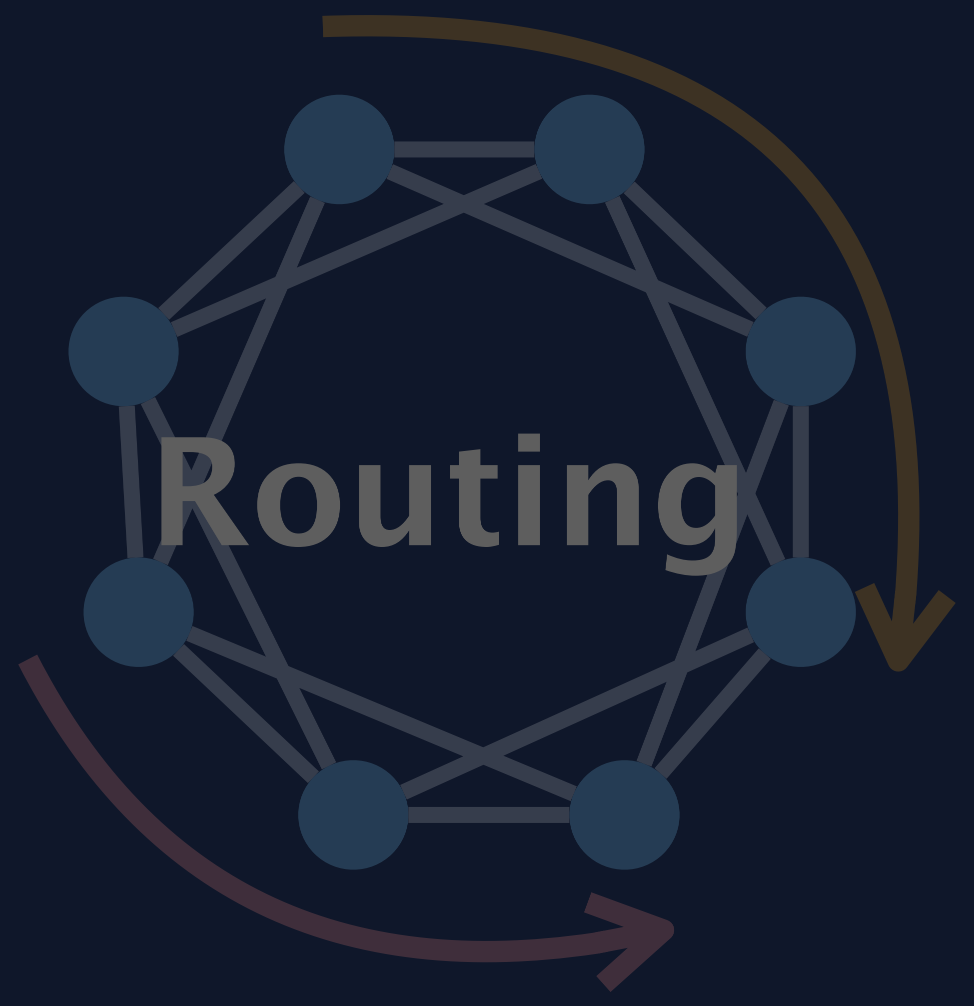


What entries to install in switches for each flow?

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers



How to connect the servers & switches?



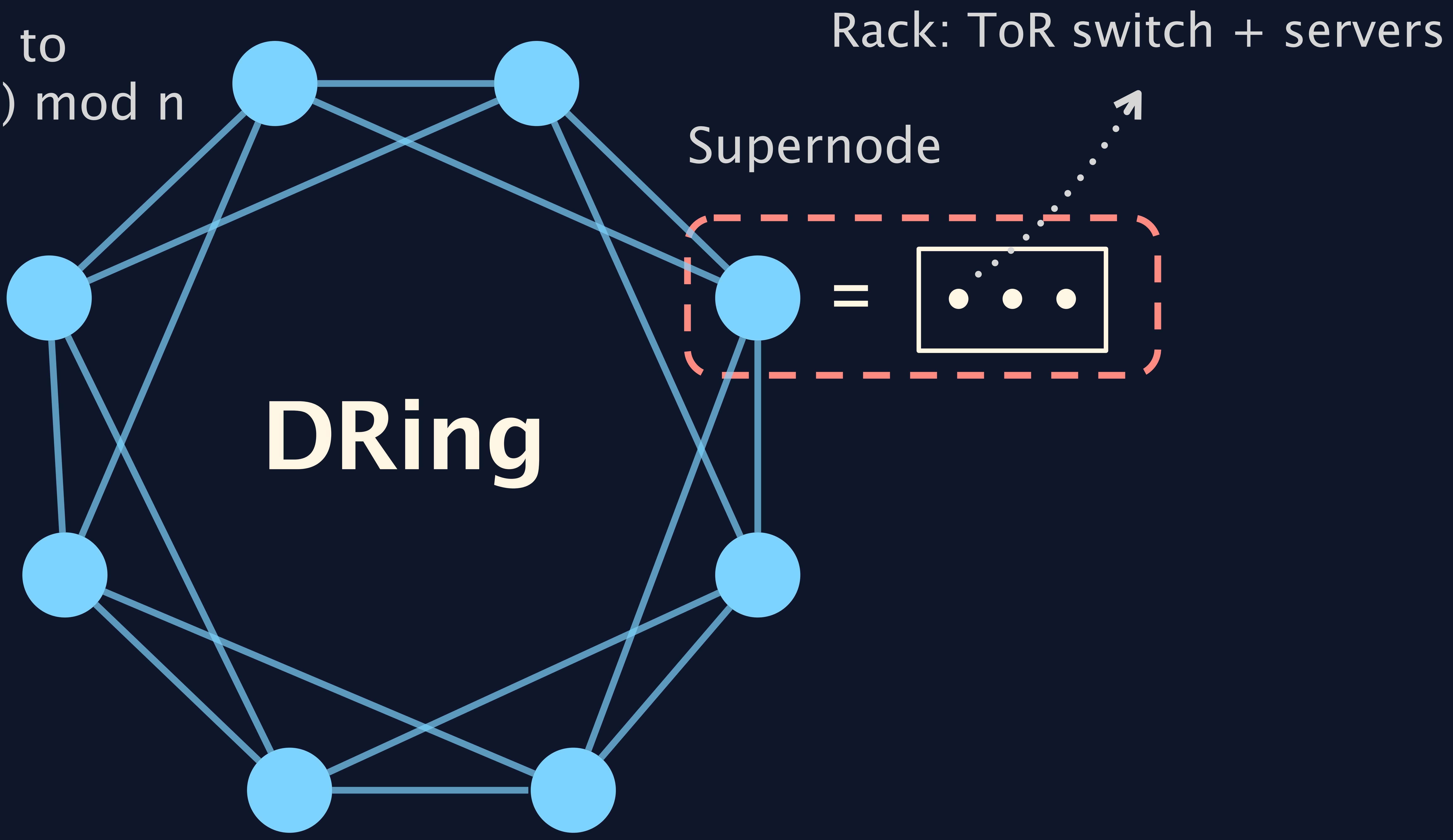
What entries to install in switches for each flow?

Supergraph:

Supernode (i) is connected to supernodes (i+1) and (i+2) mod n

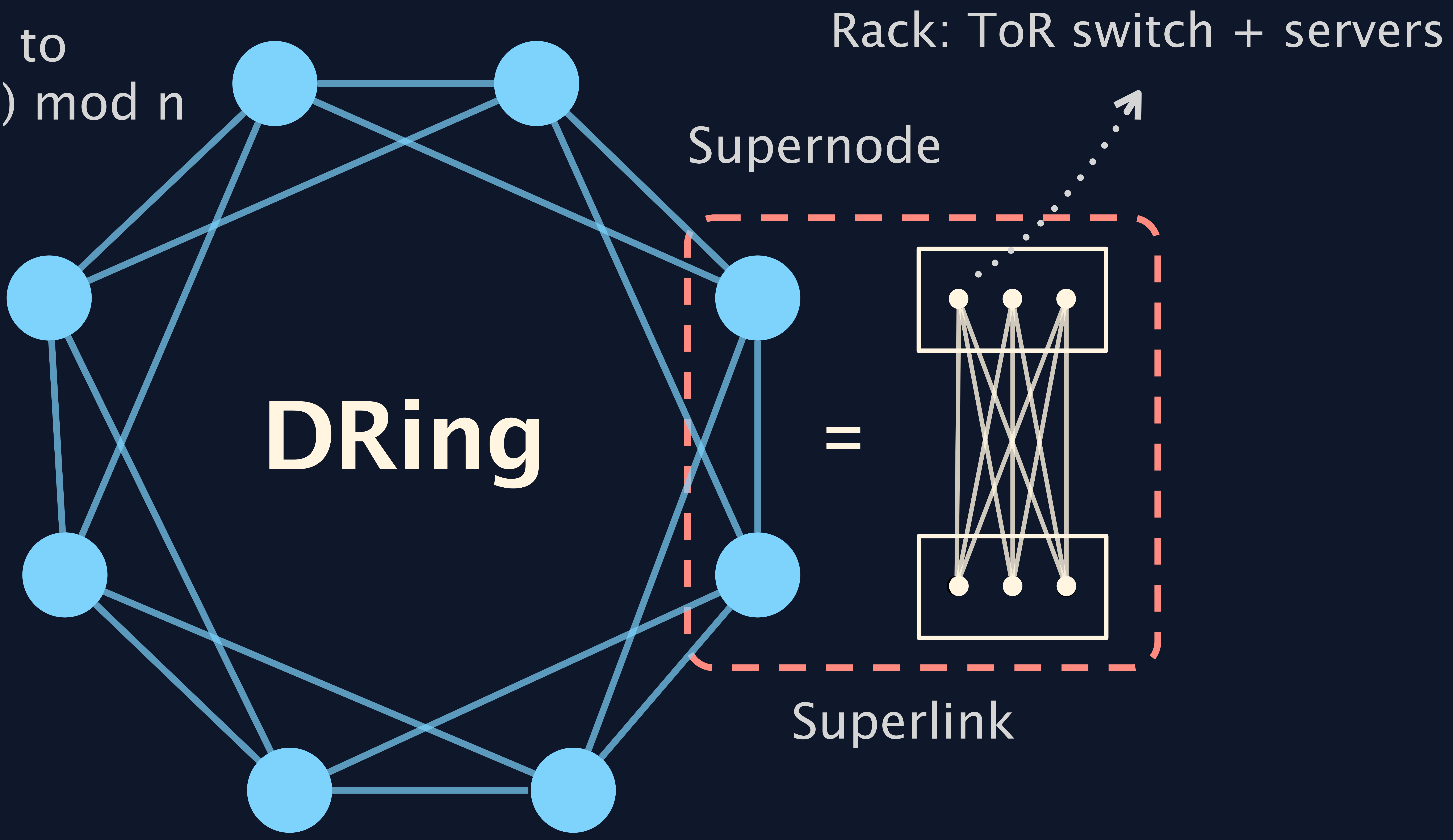
Supergraph:

Supernode (i) is connected to supernodes (i+1) and (i+2) mod n

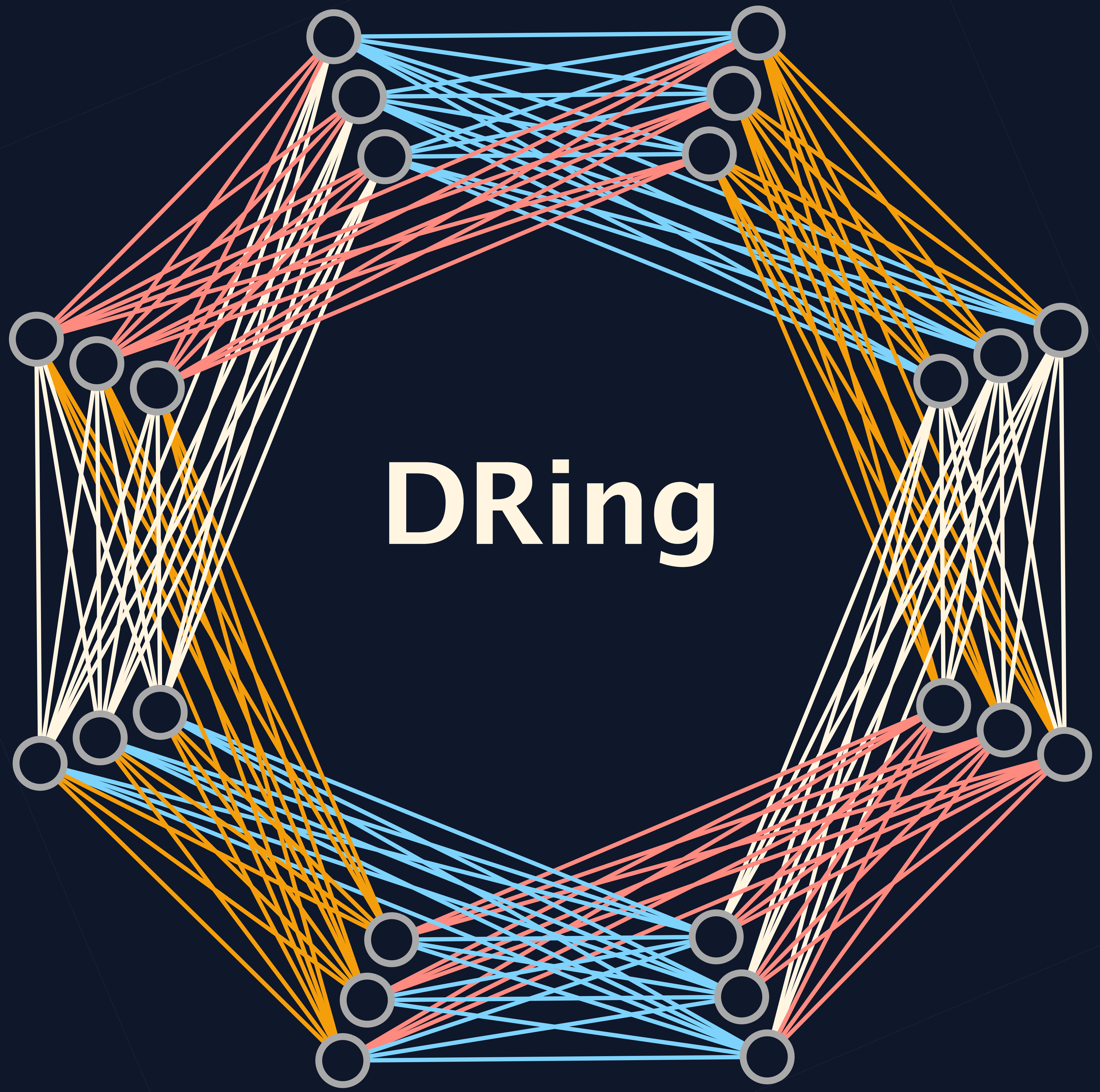


Supergraph:

Supernode (i) is connected to
supernodes (i+1) and (i+2) mod n



DRing achieves performance, fault tolerance & deployability

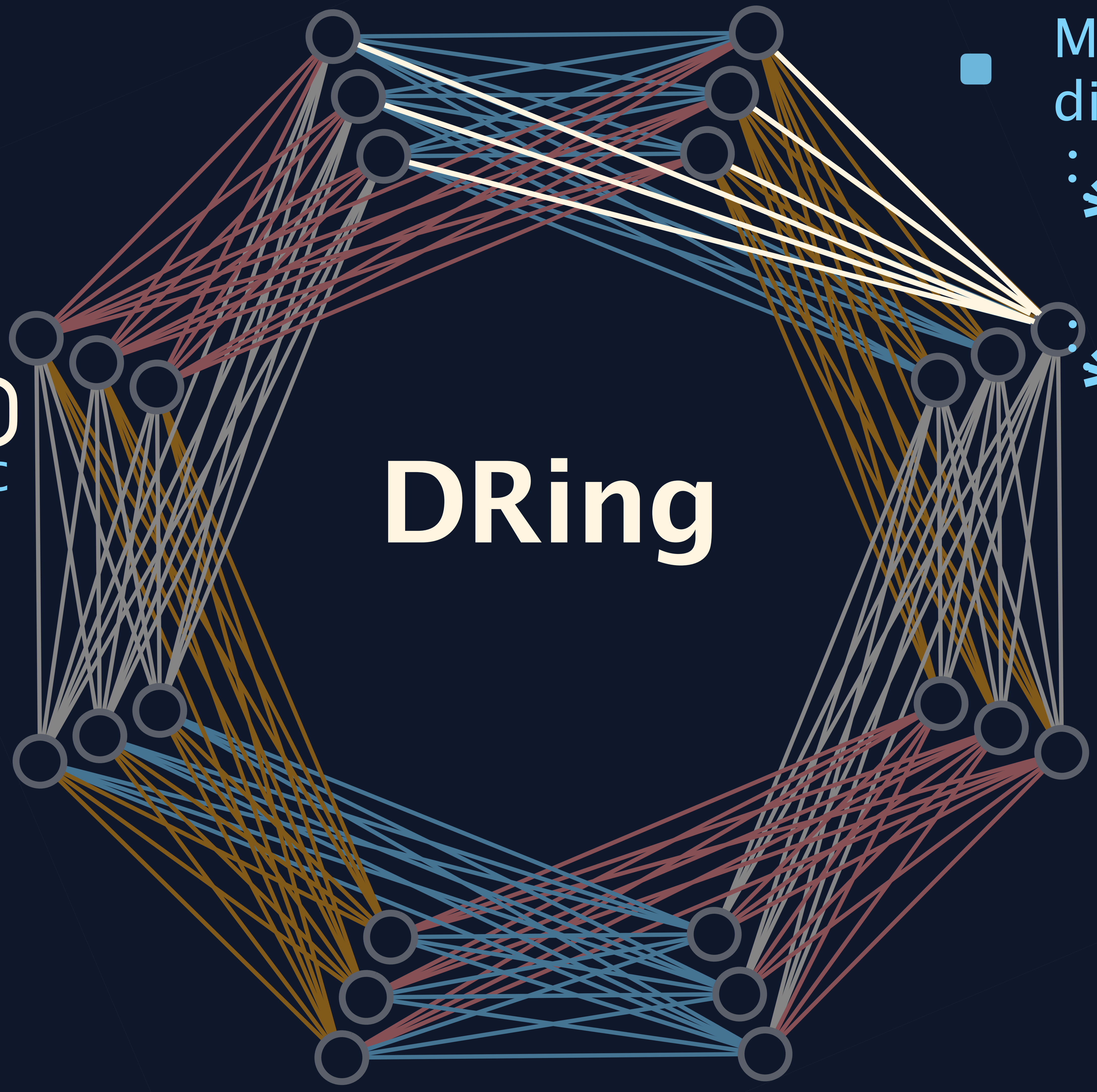


DRing achieves performance, fault tolerance & deployability

- Servers are distributed across all switches
  - Smaller rack oversubscription
  - Higher performance under skewed traffic

DRing achieves performance, fault tolerance & deployability

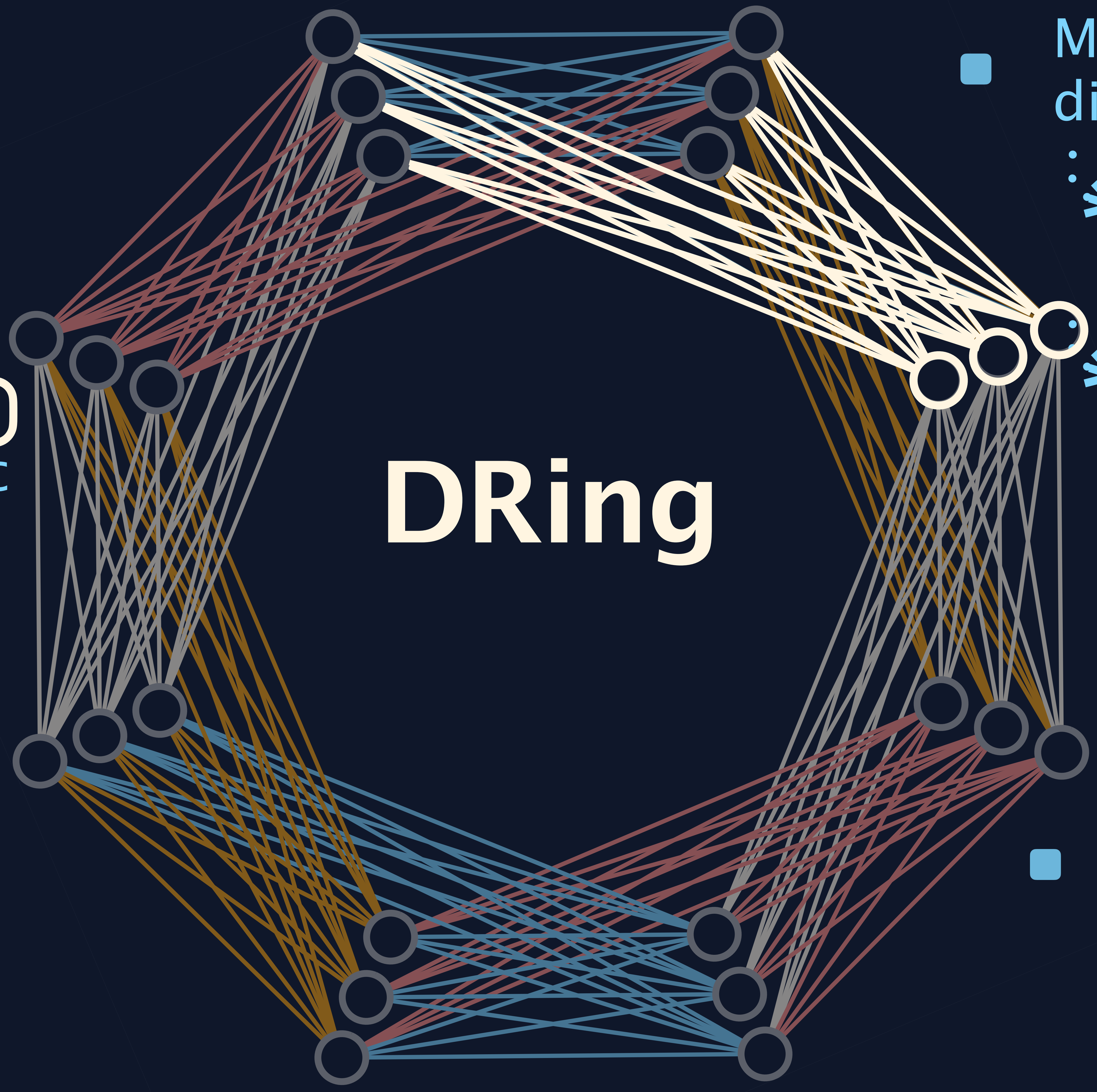
- Servers are distributed across all switches
  - Smaller rack oversubscription
  - Higher performance under skewed traffic



- Many disjoint paths + distance-2 connectivity
  - Many near-shortest paths
  - Higher performance under uniform traffic + better fault tolerance

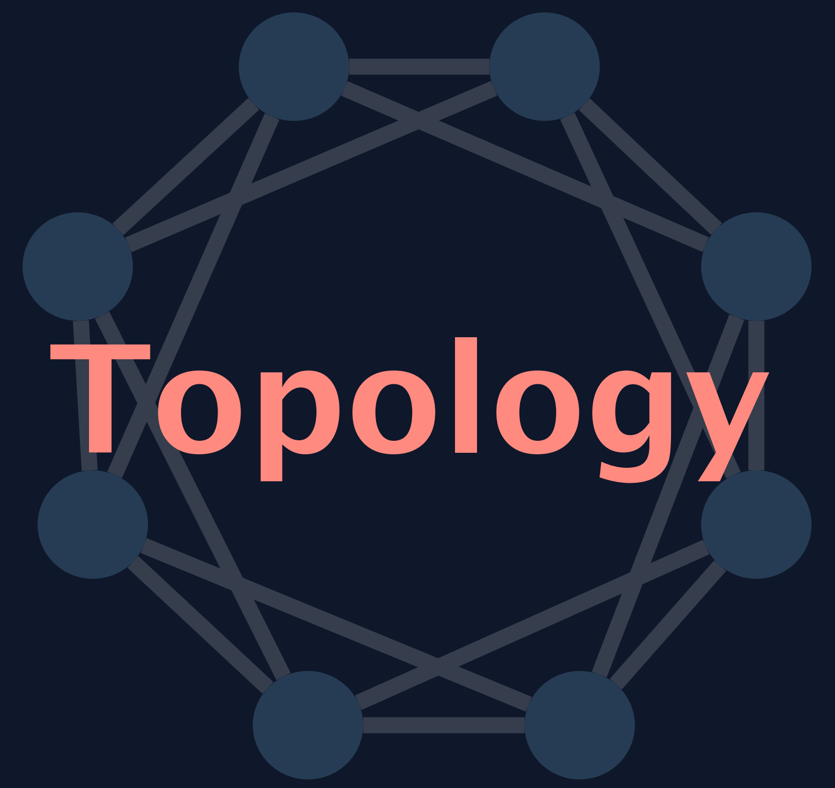
DRing achieves performance, fault tolerance & deployability

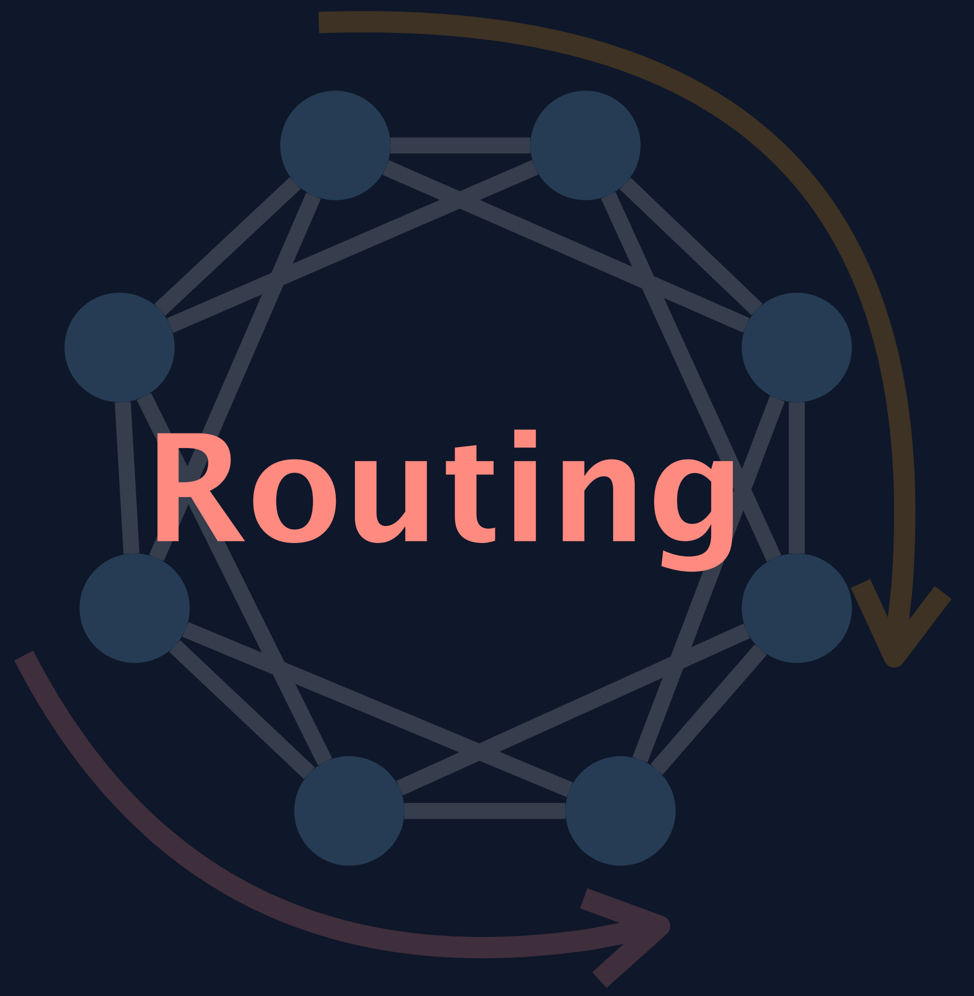
- Servers are distributed across all switches
  - Smaller rack oversubscription
  - Higher performance under skewed traffic



- Many disjoint paths + distance-2 connectivity
  - Many near-shortest paths
  - Higher performance under uniform traffic + better fault tolerance
- Natural units of co-placement & packaging
  - Easier management

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

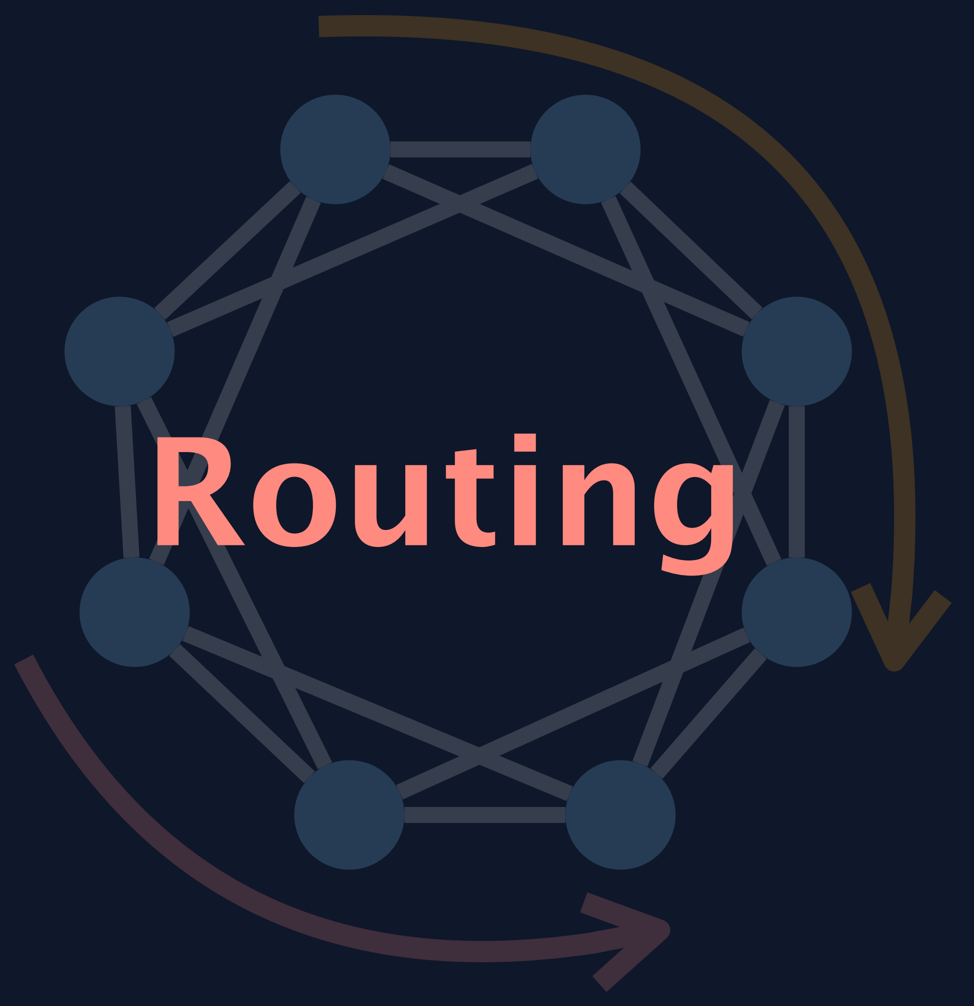




How to connect the servers & switches?

What entries to install in switches for each flow?

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

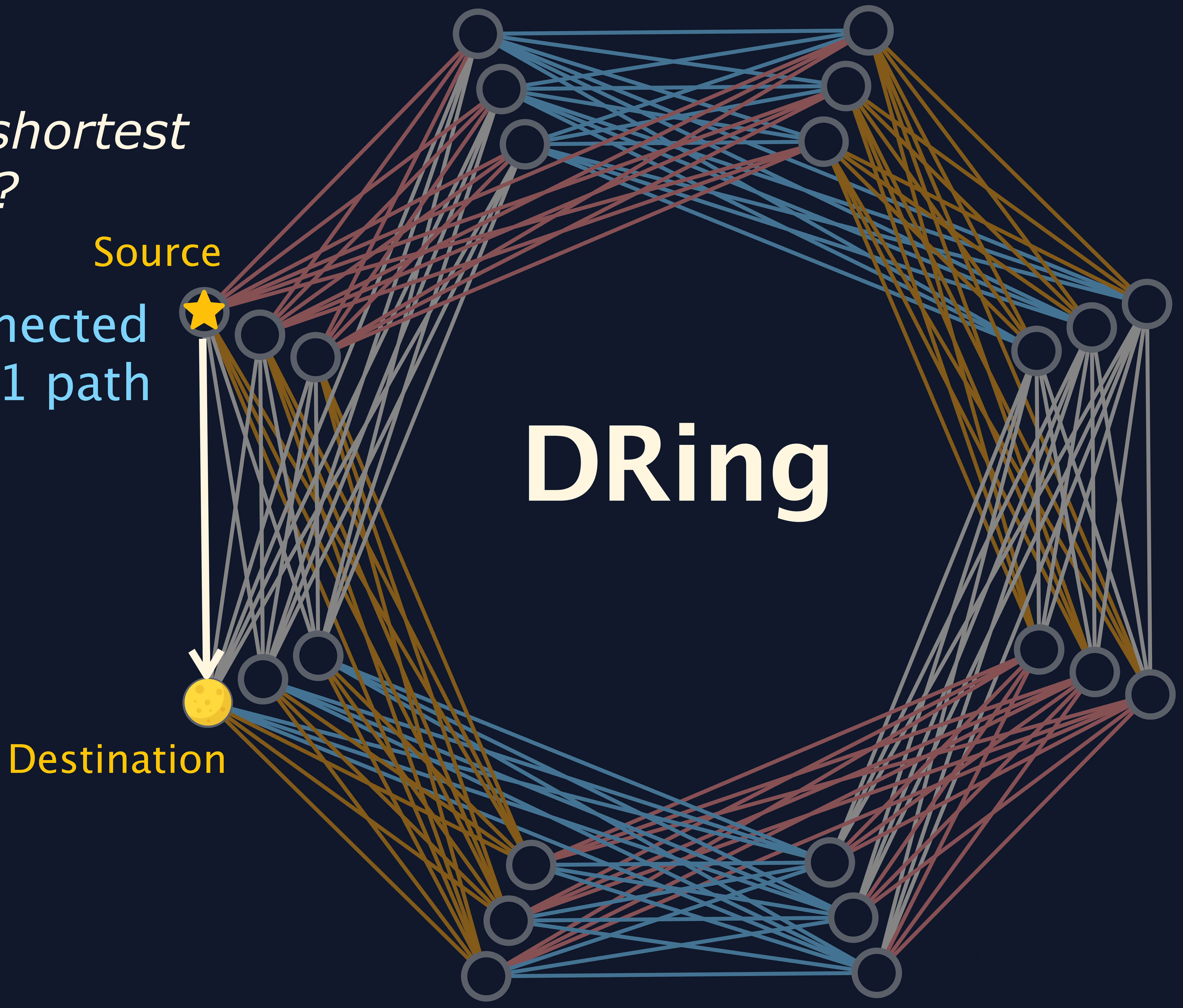


How to connect the servers & switches?

What entries to install in switches for each flow?

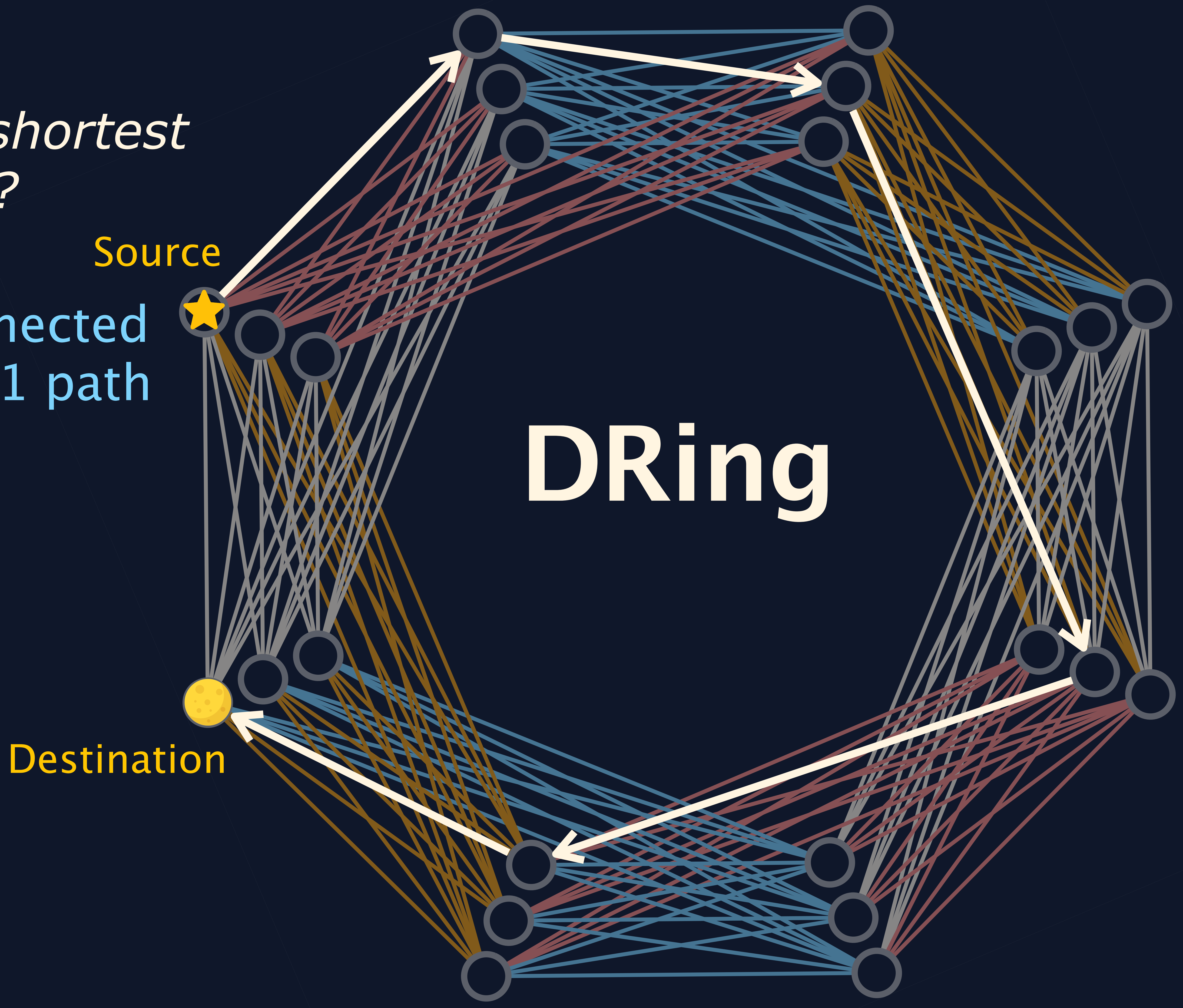
Using only the shortest path i.e., ECMP?

No, directly connected racks only have 1 path



Using only the shortest path i.e., ECMP?

No, directly connected racks only have 1 path



How about this path?

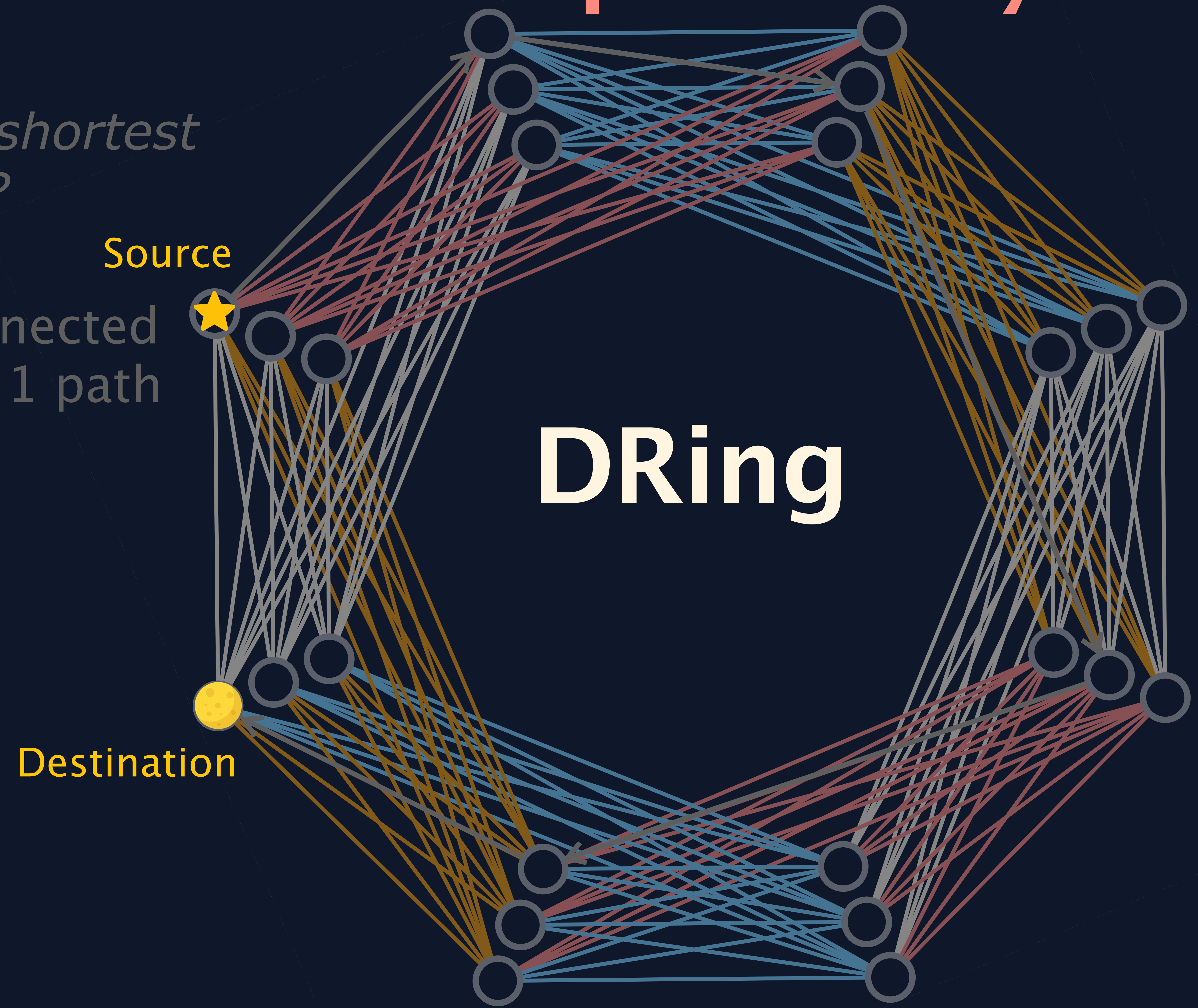
No, long paths incur higher bandwidth tax

Paths need to balance length & diversity

⇢ Use near-shortest paths only when needed

Using only the shortest path i.e. ECMP?

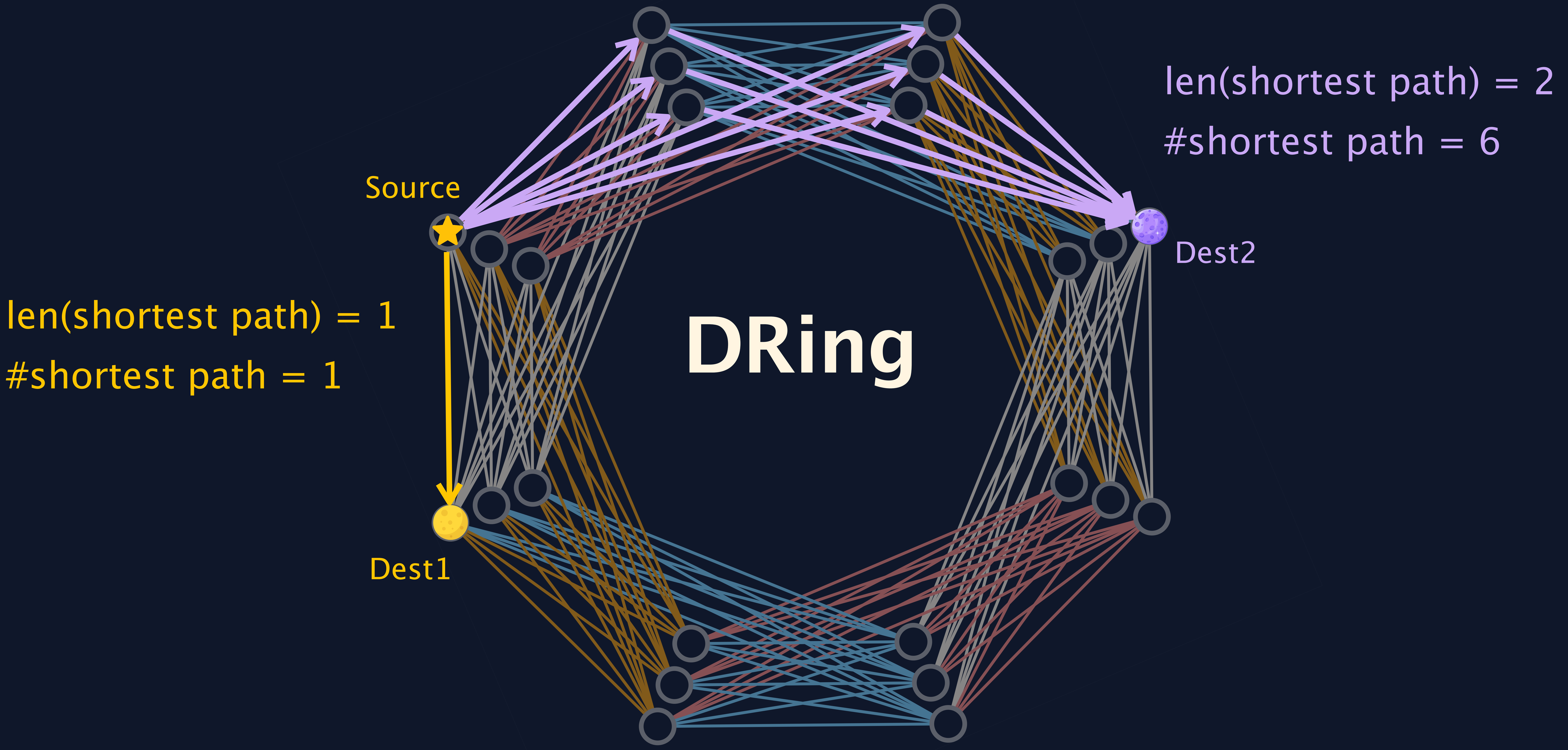
No, directly connected racks only have 1 path



How about this path?

No, long paths incur higher bandwidth tax

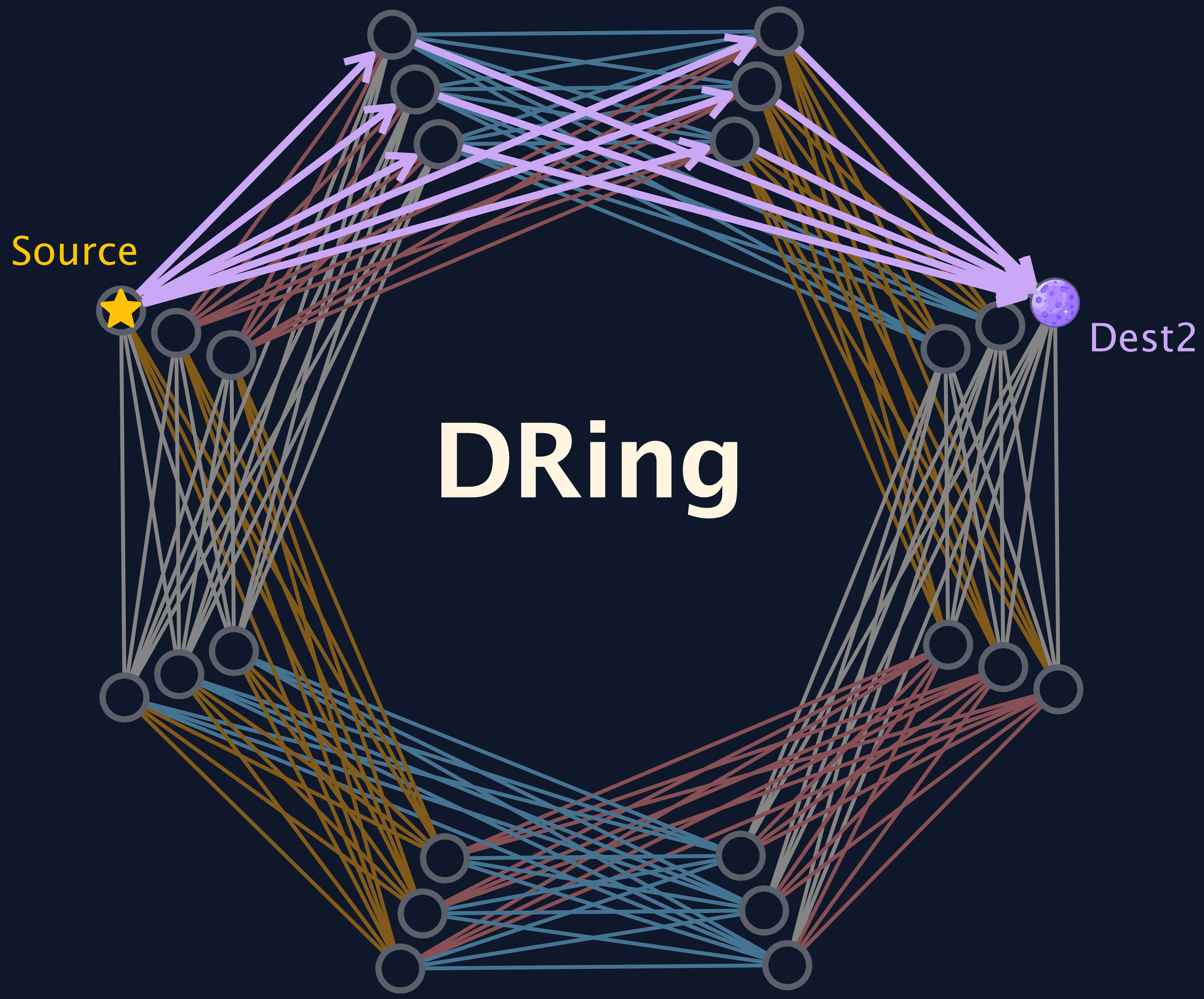
DRing: #shortest path ∝ shortest path len



Shortest-Union-K

Use all paths from source to destination which are either:
(1) shortest paths
(2) or len(path) <= K

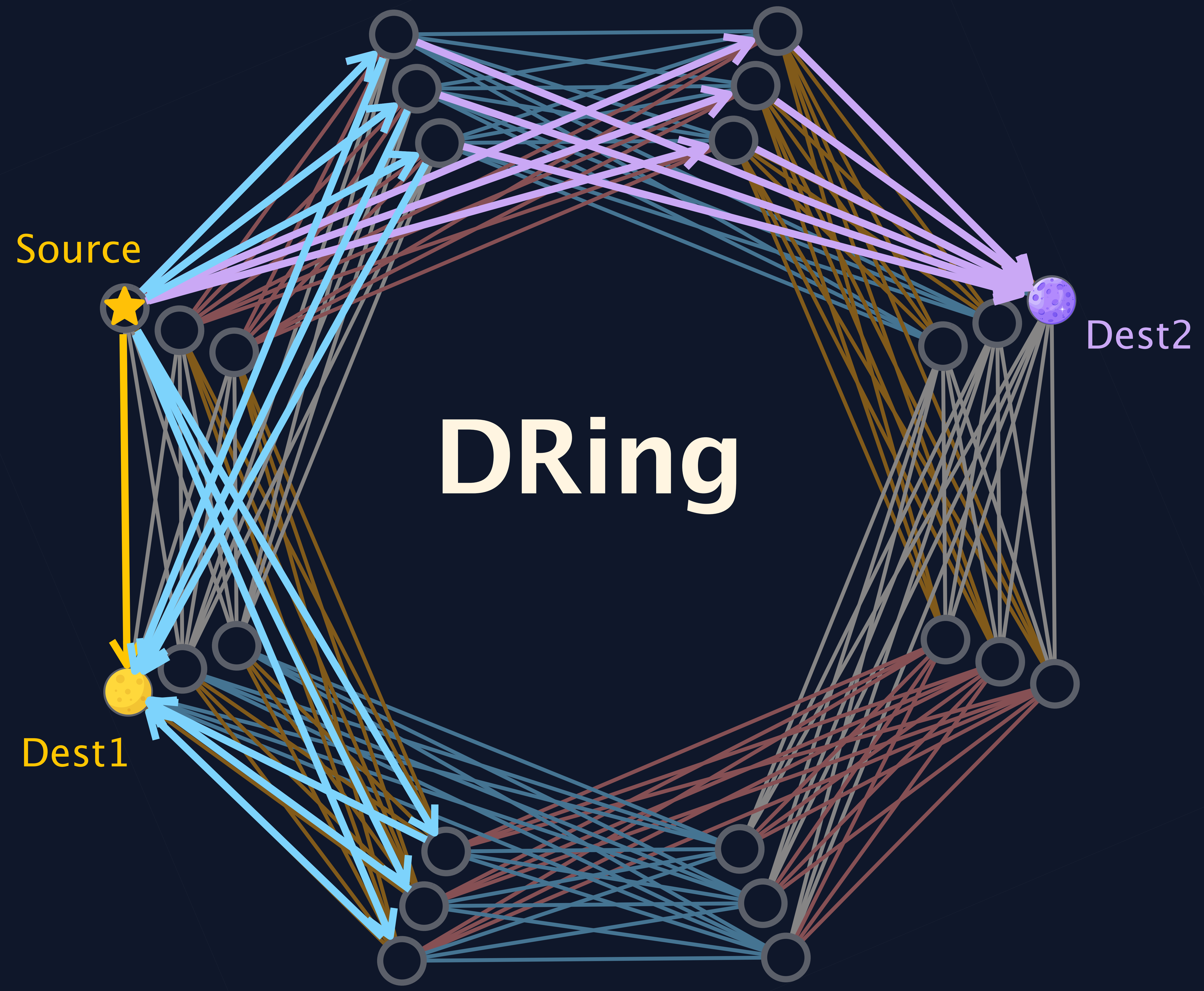
Eg, K = 2



Shortest-Union-K

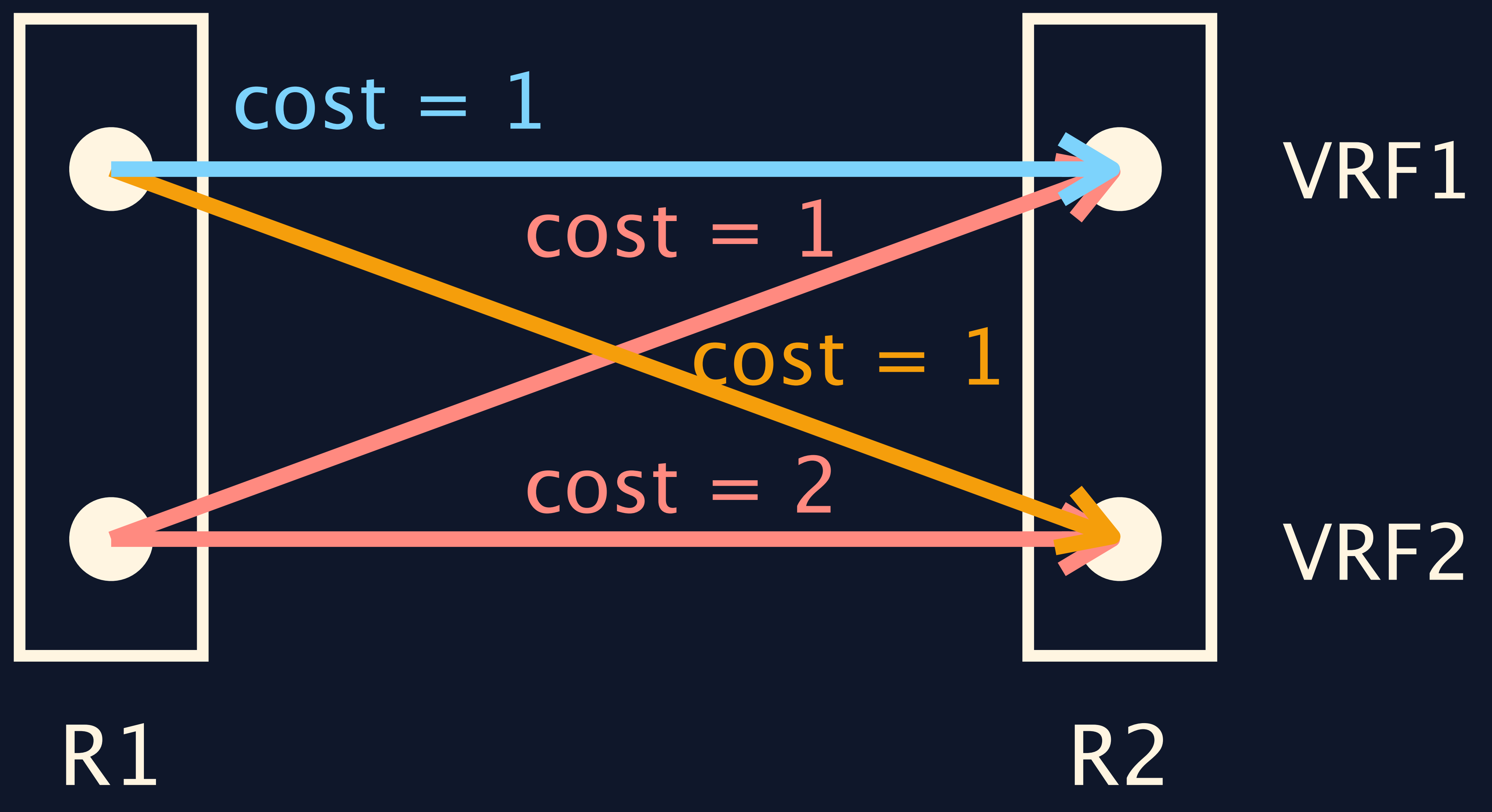
Eg, K = 2

Use all paths from source to destination which are either:
(1) shortest paths
(2) or len(path) <= K



SU-K runs on standard hardware via VRFs

Eg, K = 2



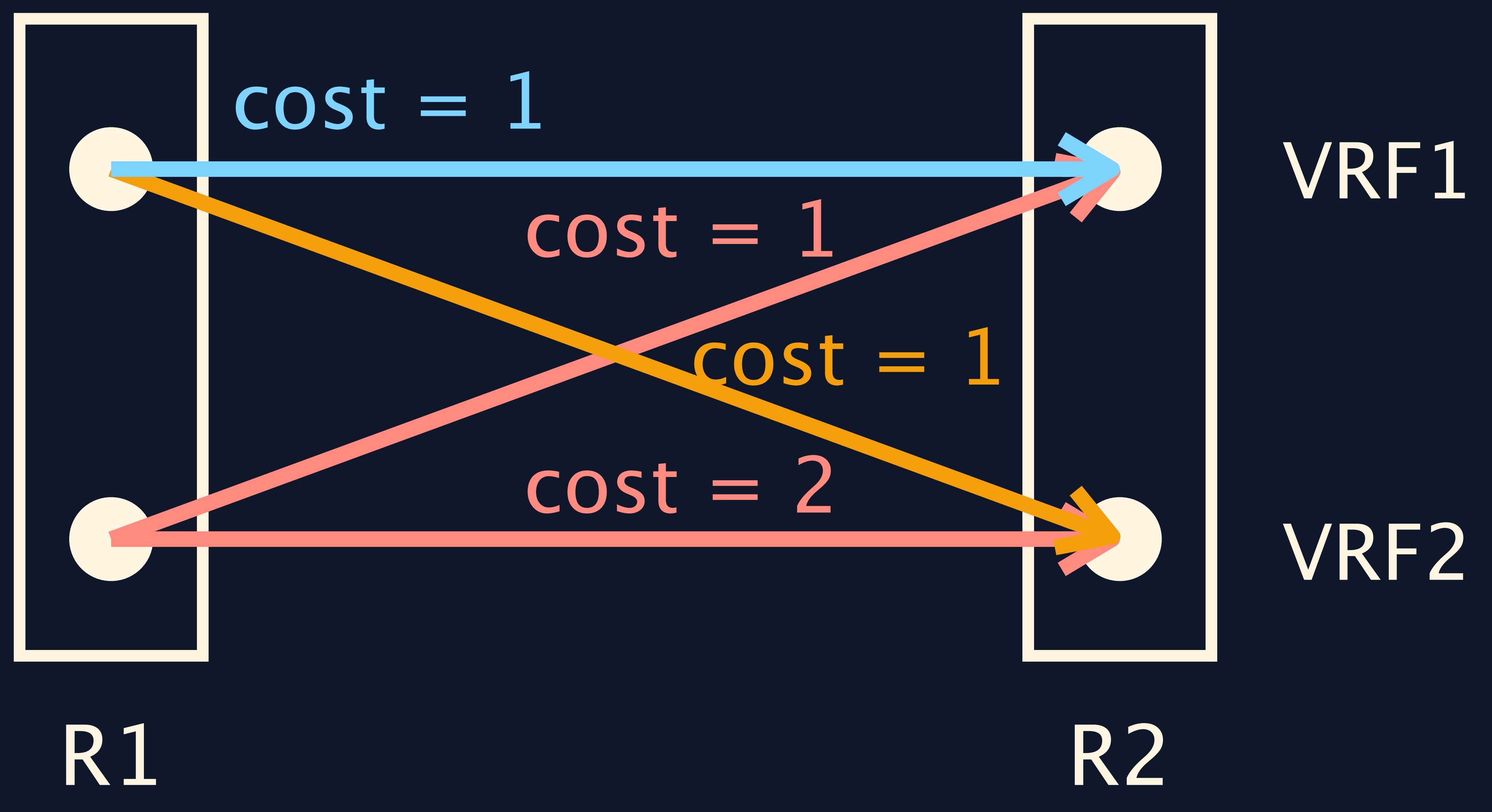
Partition each router into K virtual routers

Create the following virtual connections:

- (VRF K, R1) → (VRF i, R2) of cost i ∀ i
- (VRF (i–1), R1) → (VRF i, R2) of cost 1 ∀ i
- (VRF 1, R1) → (VRF 1, R2) of cost 1

SU-K runs on standard hardware via VRFs

Eg, K = 2



Just need shortest path routing in the VRF graph

- Prototype with BGP in GNS3
- Correctness proof in the paper

Oblivious & semi-oblivious weight assignment

- Path selection: shortest-union-K
- Oblivious weight assignment: equal-split
- Semi-oblivious weight assignment:
  Central controller collecting traffic, solving an MCF-based LP & distributing weights (every ~30 min)

  ↳ Good path selection facilitates further weight optimization

  ↳ Distributing traffic more widely accommodates traffic variations

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

- **Motivation** Small-scale DCs: important but overlooked
- **Challenge** Trilemma: performance - fault tolerance - deployability
- **Starfish** Topology-routing co-design
- **Evaluation** Simulation results: high-performing & resilient to failures

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

- **Motivation** Small-scale DCs: important but overlooked
- **Challenge** Trilemma: performance - fault tolerance - deployability
- **Starfish** Topology-routing co-design
- **Evaluation** Simulation results: high-performing & resilient to failures

Key evaluation takeaways

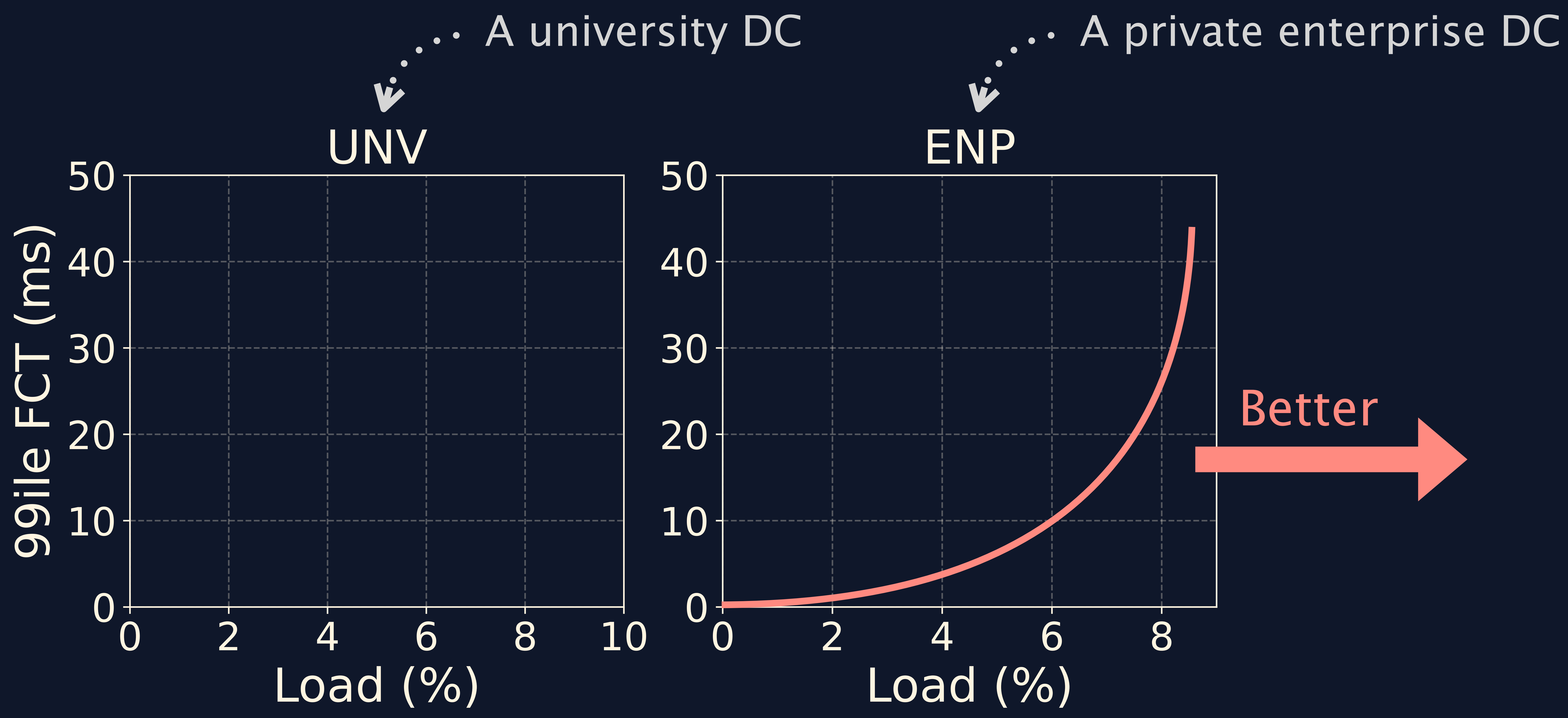
- Starfish effectively supports more capacity on real-world traces
- Starfish is resilient to random link and ToR failures
- Starfish's performance benefits are more pronounced at small scales and with larger oversubscription

Key evaluation takeaways

- Starfish effectively supports more capacity on real-world traces
- Starfish is resilient to random link and ToR failures
- Starfish's performance benefits are more pronounced at small scales and with larger oversubscription

htsim packet-level simulation

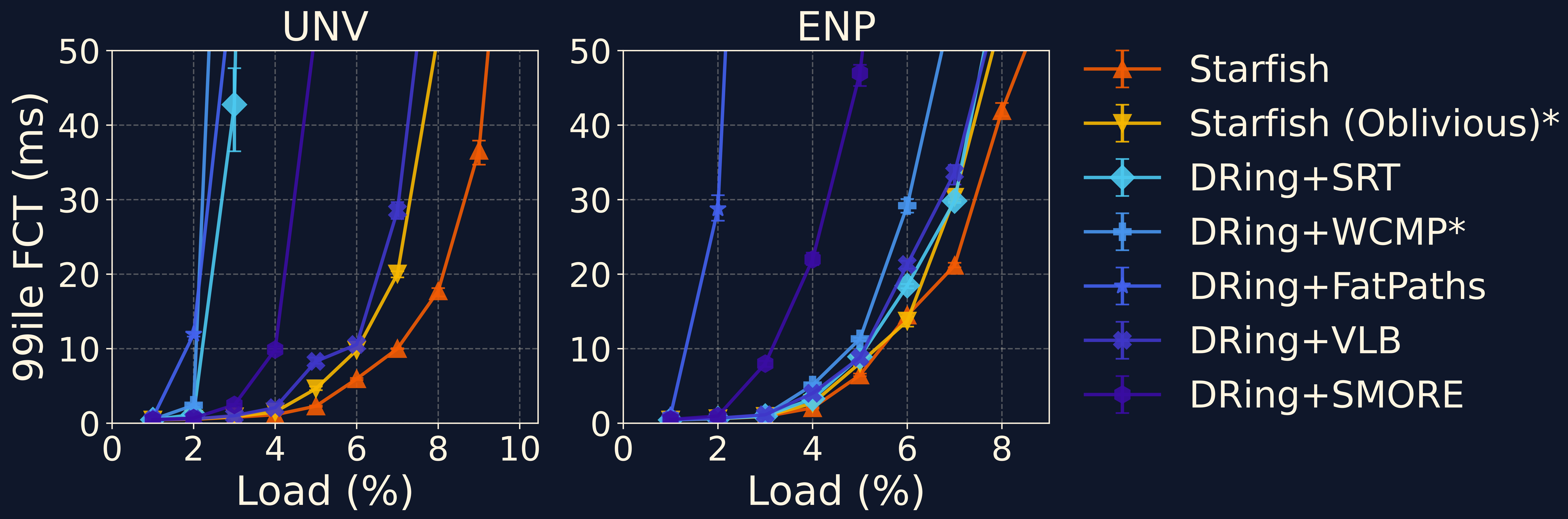
Starfish effectively supports more network capacity



Same #switches, #servers, #links

Starfish effectively supports more network capacity

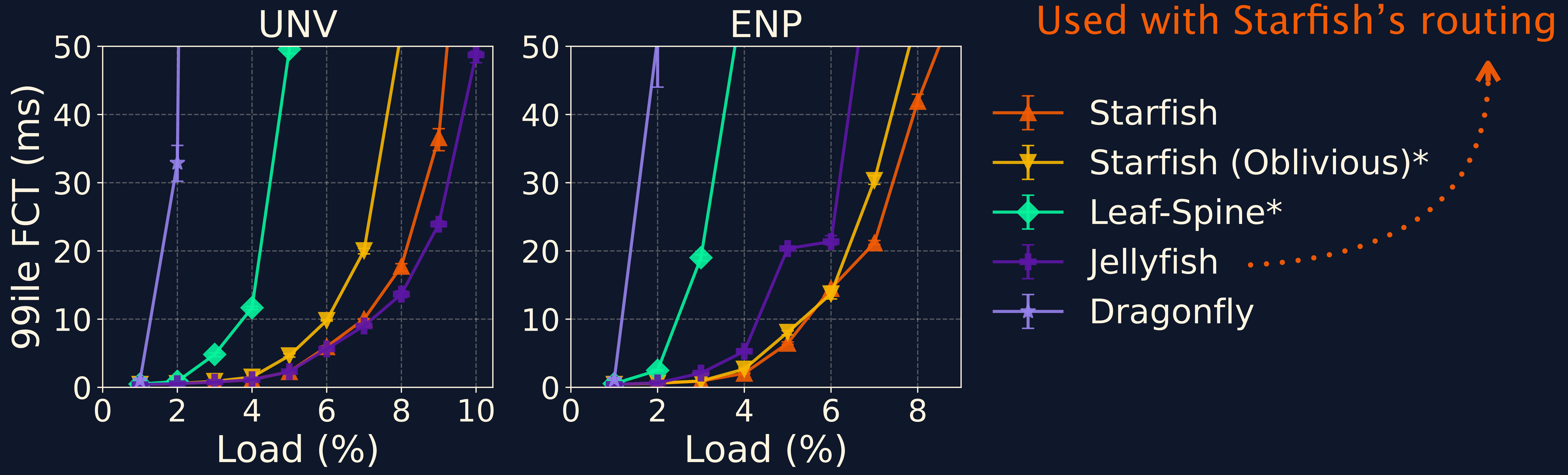
Starfish's routing: 26% more traffic on average than VLB (the strongest routing baseline)



Starfish effectively supports more network capacity

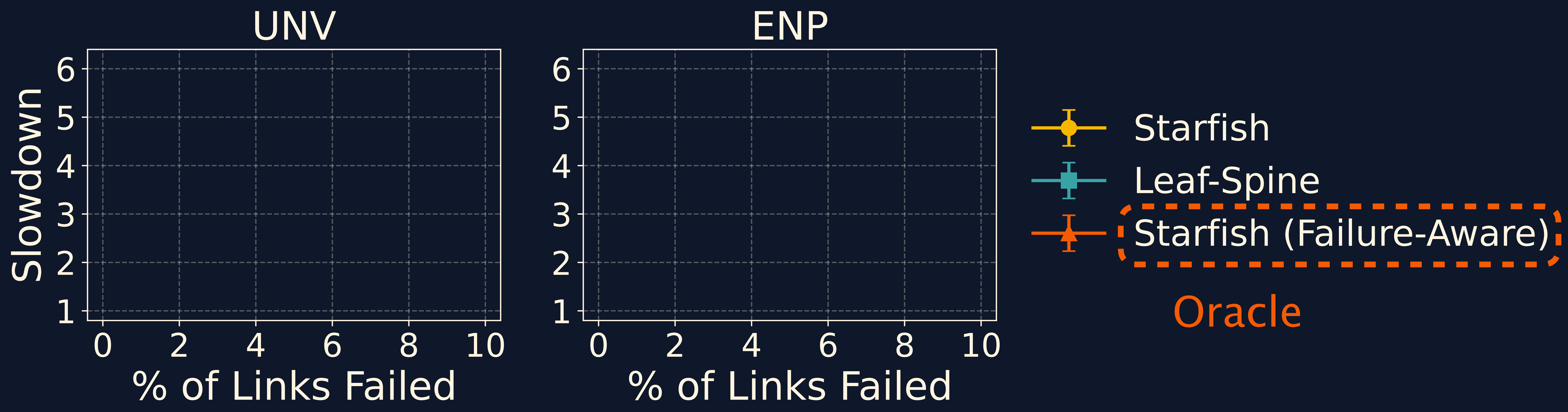
Starfish: 56% more traffic on average than leaf-spine

Starfish (Oblivious): 33% more traffic on average



Jellyfish (with Starfish's routing): 52% more traffic on average

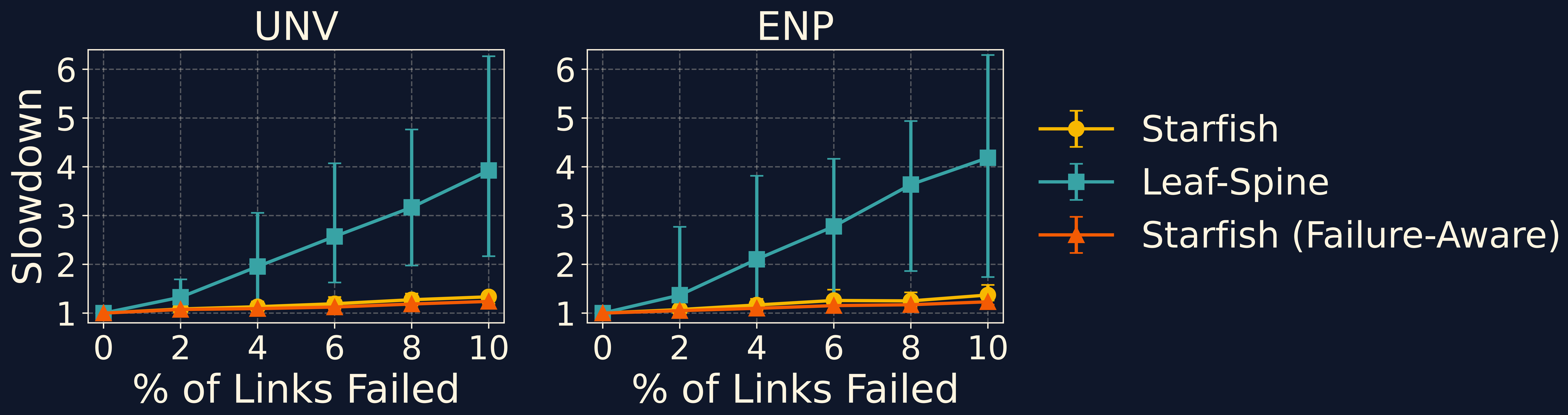
Starfish is resilient to random link failures



Starfish is resilient to random link failures

Starfish: <=1.37 & approach the oracle

Leaf-Spine: larger & more varied slowdowns



Conclusion

Starfish enables small-scale DCs to support 56% more traffic just by rearranging their existing networking equipments

- Starfish also enables Jellyfish topologies to support 52% more traffic than leaf-spine with a ready-to-deploy routing scheme
- We encourage future efforts from the community on small-scale DCs
- Find out more about Starfish:

Paper

Artifact

Image References

- Icon made by Freepik from www.flaticon.com
- Icon made by Dewi Sari from www.flaticon.com
- Icon made by Pixel perfect from www.flaticon.com
- Icon made by Vectors Market from www.flaticon.com
- Icon made by vectorsmarket15 from www.flaticon.com
- https://www.cs.cornell.edu/courses/cs5413/2014fa/lectures/08-fattree.pdf
- https://htor.inf.ethz.ch/publications/img/sf_sc_slides_website.pdf
- https://www.usenix.org/sites/default/files/conference/protected-files/jellyfish_nsditalk.pdf
- https://husant.github.io/Xpander/files/xpander-conxet-slides.pdf
- https://www.researchgate.net/figure/Sample-Dragonfly-topology-with-h2-p2-a4-36-routers-and-72-compute-nodes_fig2_261313973

Thank you!

Starfish: A Topology-Routing Co-Design for Small-Scale Data Centers

Anchengcheng Zhou
Princeton University