Routing in Dragonfly Topologies

IETF 116 Yokohama

Dmitry Afanasiev, Yandex
Roman Glebov, Yandex
Jeff Tantsura, Nvidia

Why New Topologies for Data Center?

- Network diameter
- Number of links, especially long links, and corresponding cost
- Scalability - larger network with the same number of switches and inter-switch links
- Path diversity and graceful degradation in presence of failures

- Many/most ideas originated in HPC interconnects world and now percolate into IP/Ethernet
  - but we don't have the same mechanisms (e.g. virtual channels, credits, proper adaptive routing)

Network topologies for large-scale compute centers: It's the diameter, stupid!

Advanced Topologies

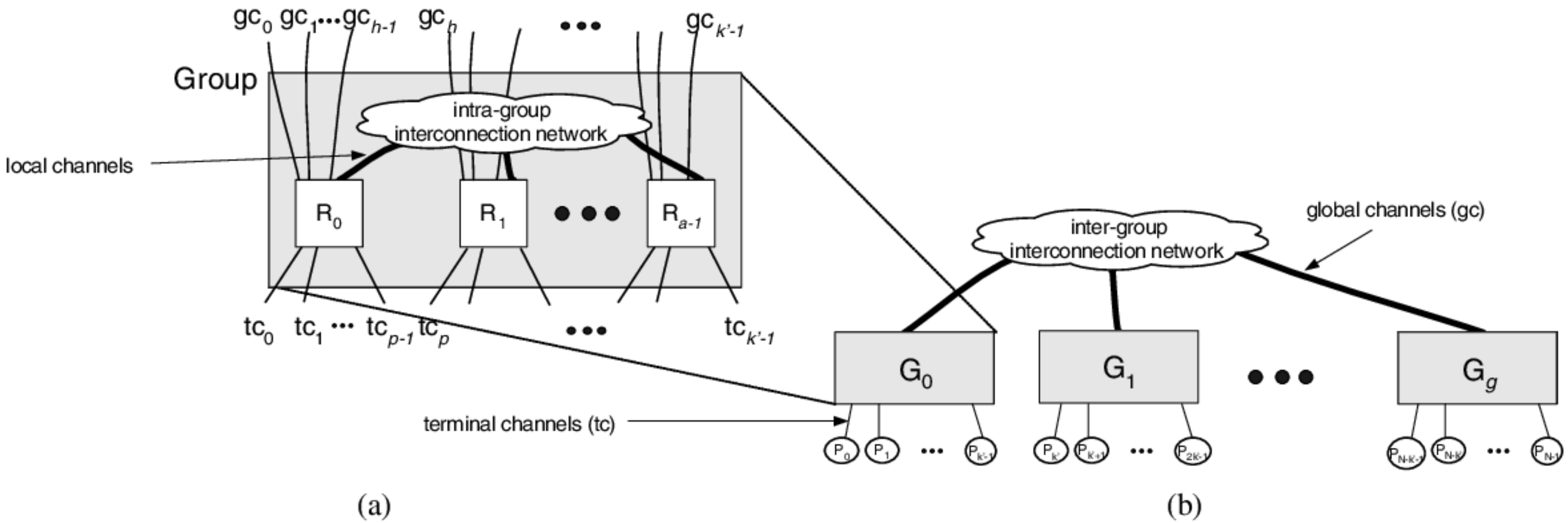
- A lot of academic research - something new almost every year @ NSDI and SIGCOMM
- Many are interesting, but not really deployable:
  - Difficult to expand or deploy incrementally
  - Complex wiring rules
  - Sometimes irregular (Jellyfish as an extreme example)
- Some are easier than others to make work with tools we have in IP networks
- All require more complex routing and forwarding
  - Non-minimal routing and forwarding
  - More forwarding state
  - Adaptive routing for efficiency

Dragonfly Topology

Dragonfly is a hierarchical topology with the following properties:

- Several groups / pods, full mesh between groups
- Any topology inside group
    - Different intra-group topologies produce different Dragonfly “flavors"
- Focus on reducing the number of long links and network diameter to reduce total cost of network
- Maximum 3 hops: one global, two local
- Requires Adaptive Routing to enable efficient operation

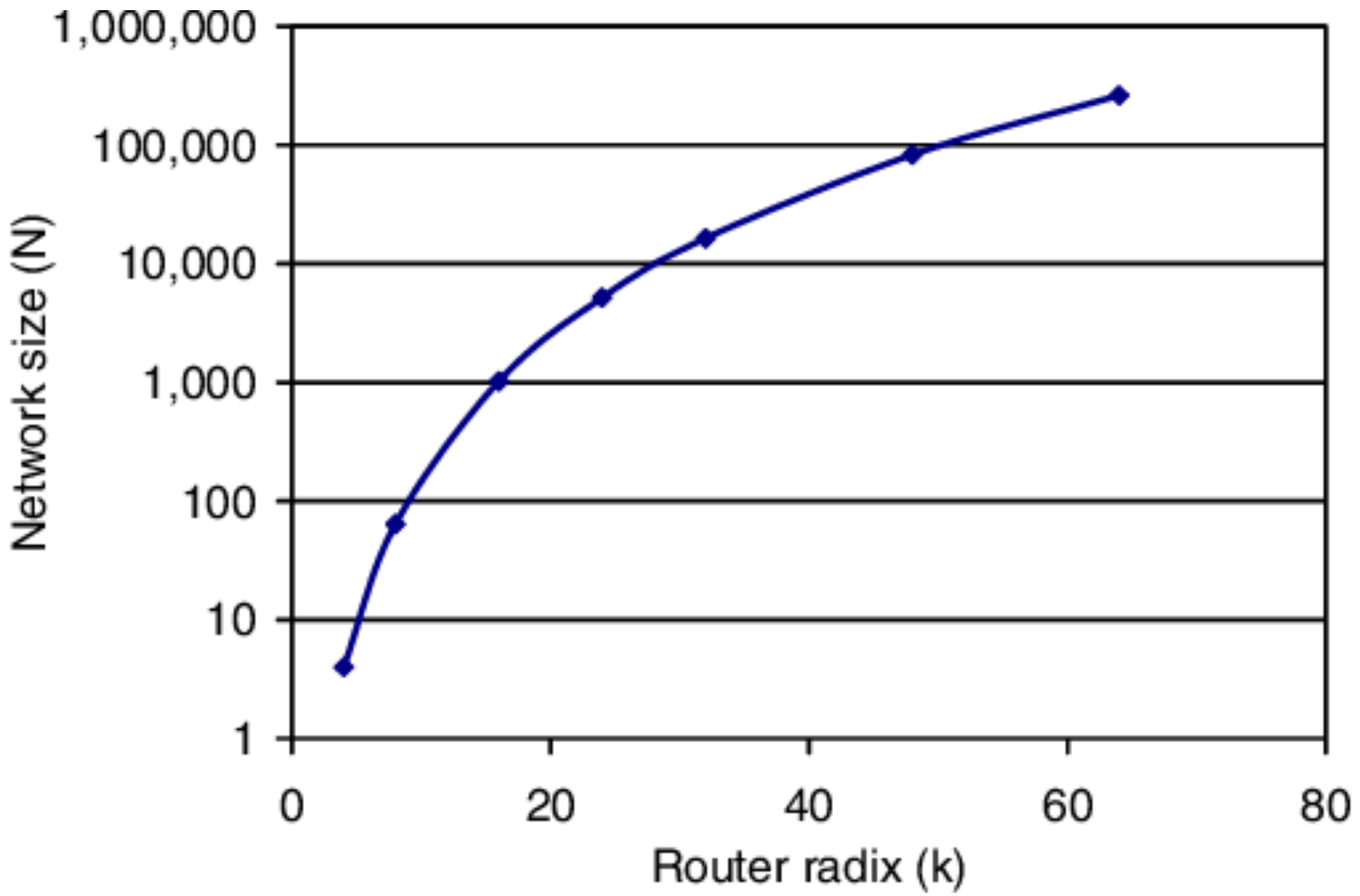
Dragonfly Topology



[Technology-Driven, Highly-Scalable Dragonfly Topology]()

Scalability

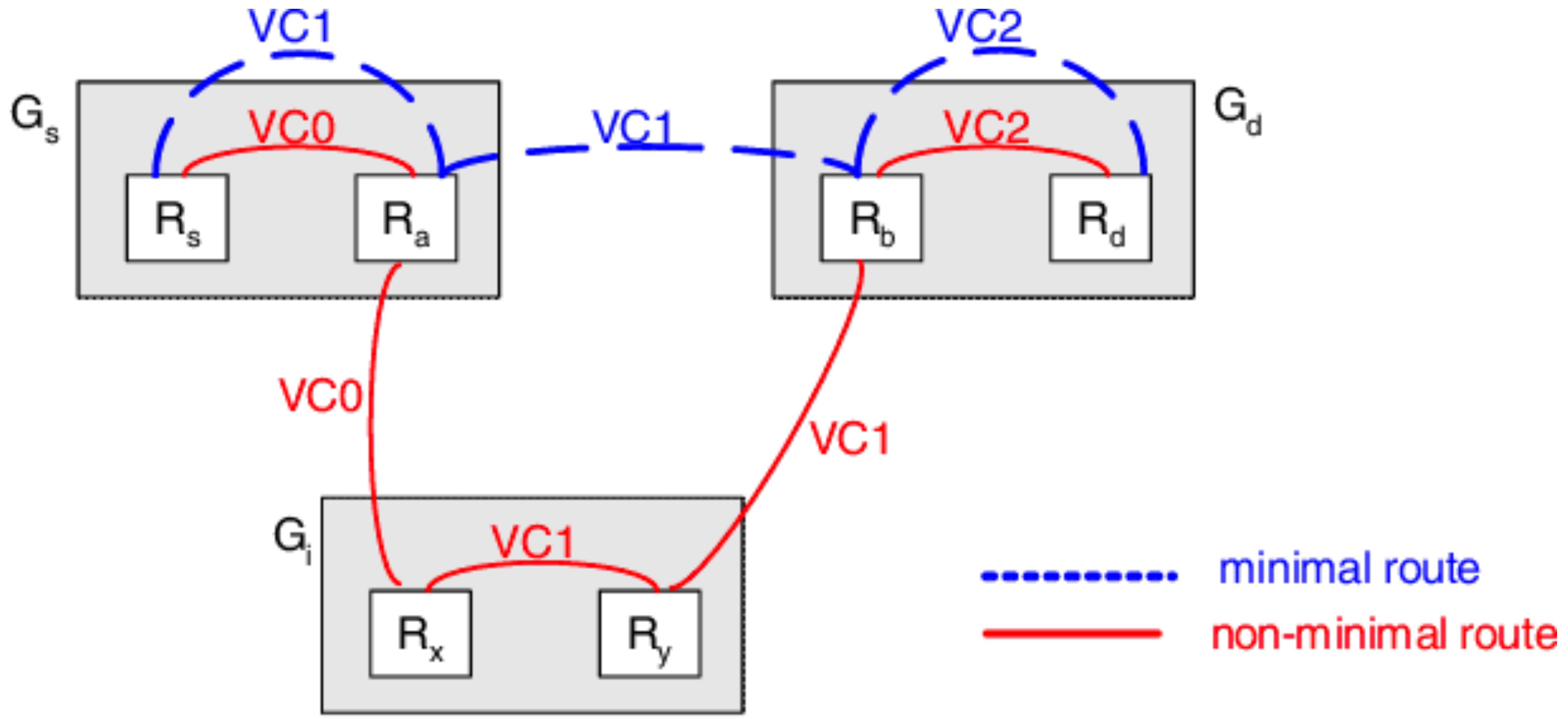
Network size vs router radix



•

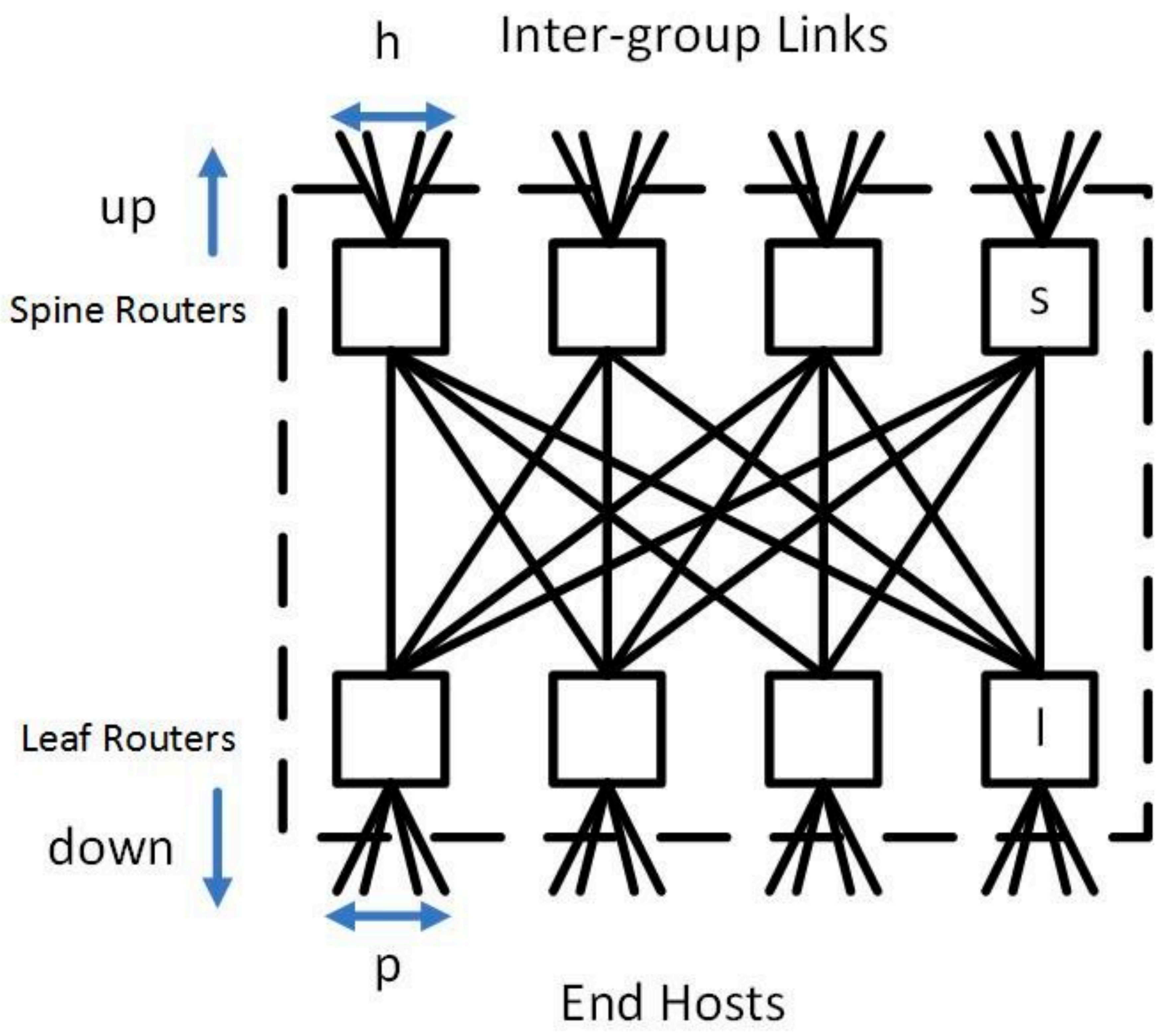
Routing in Dragonfly

- Maximum 3 hops: one global, two local
- Uses virtual channels



Dragonfly+

- Full bipartite / leaf-spine intra-group topology



https://www.researchgate.net/publication/313341364_Dragonfly_Low_Cost_Topology_for_Scaling_Datacenters

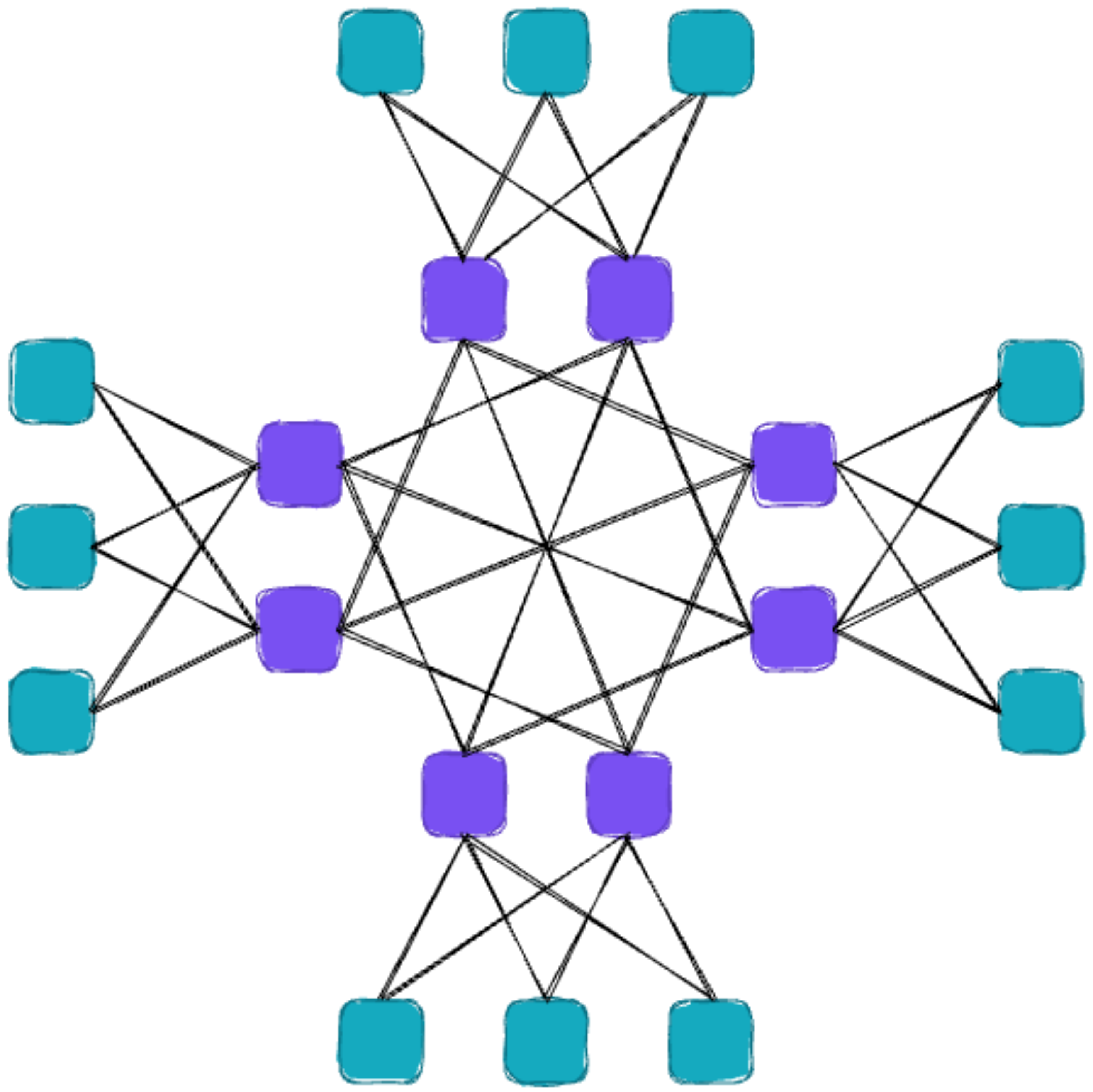
https://hipineb.i3a.info/hipineb2017/wp-content/uploads/sites/6/2017/05/slides_alex.pdf

http://www.hpcadvisorycouncil.com/events/2019/APAC-AI-HPC/uploads/2018/07/Exascale-HPC-Fabric-Topology.pdf

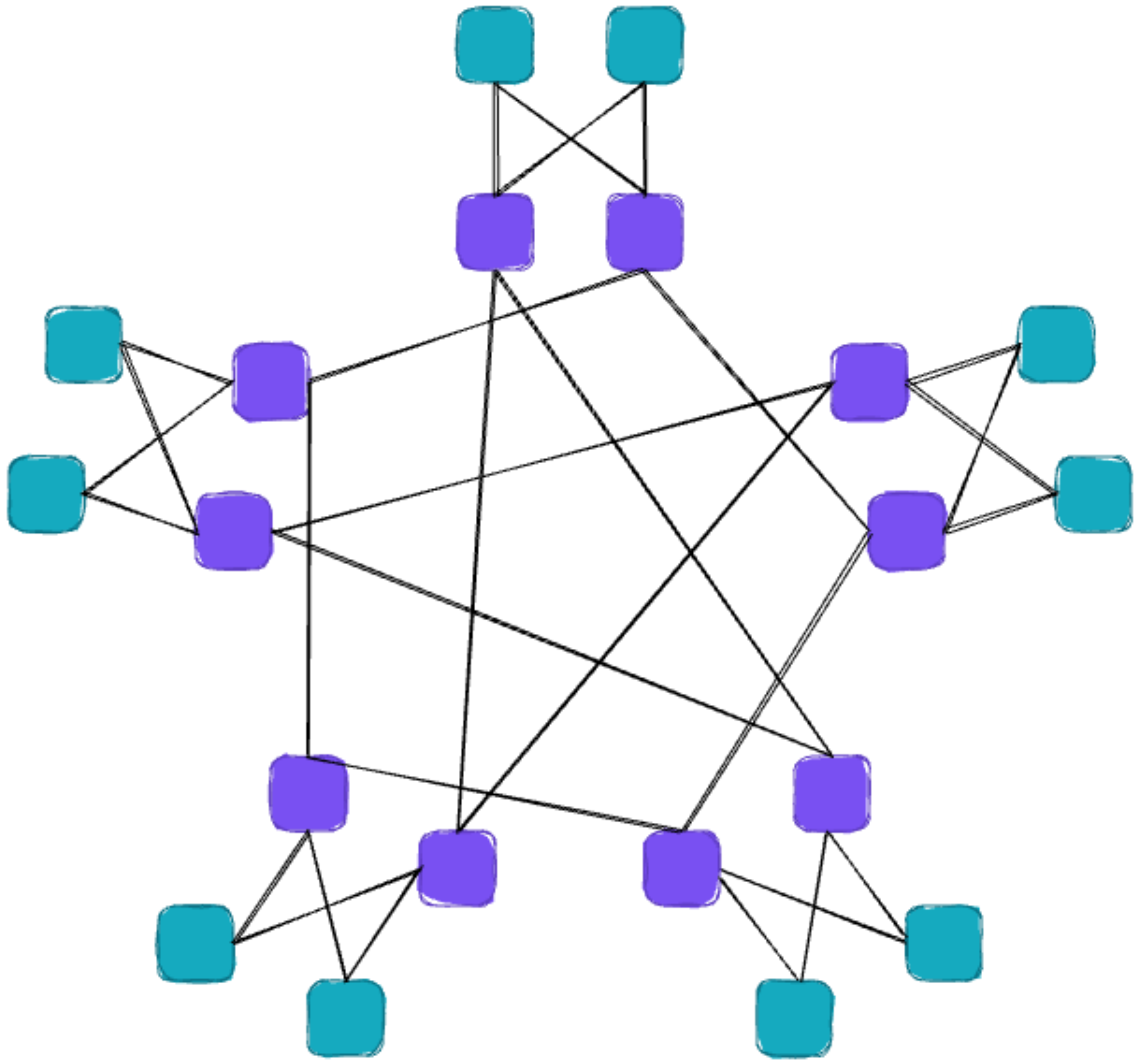
Paths in Dragonfly+

Is every spine in every group is connected to every other group?

Yes

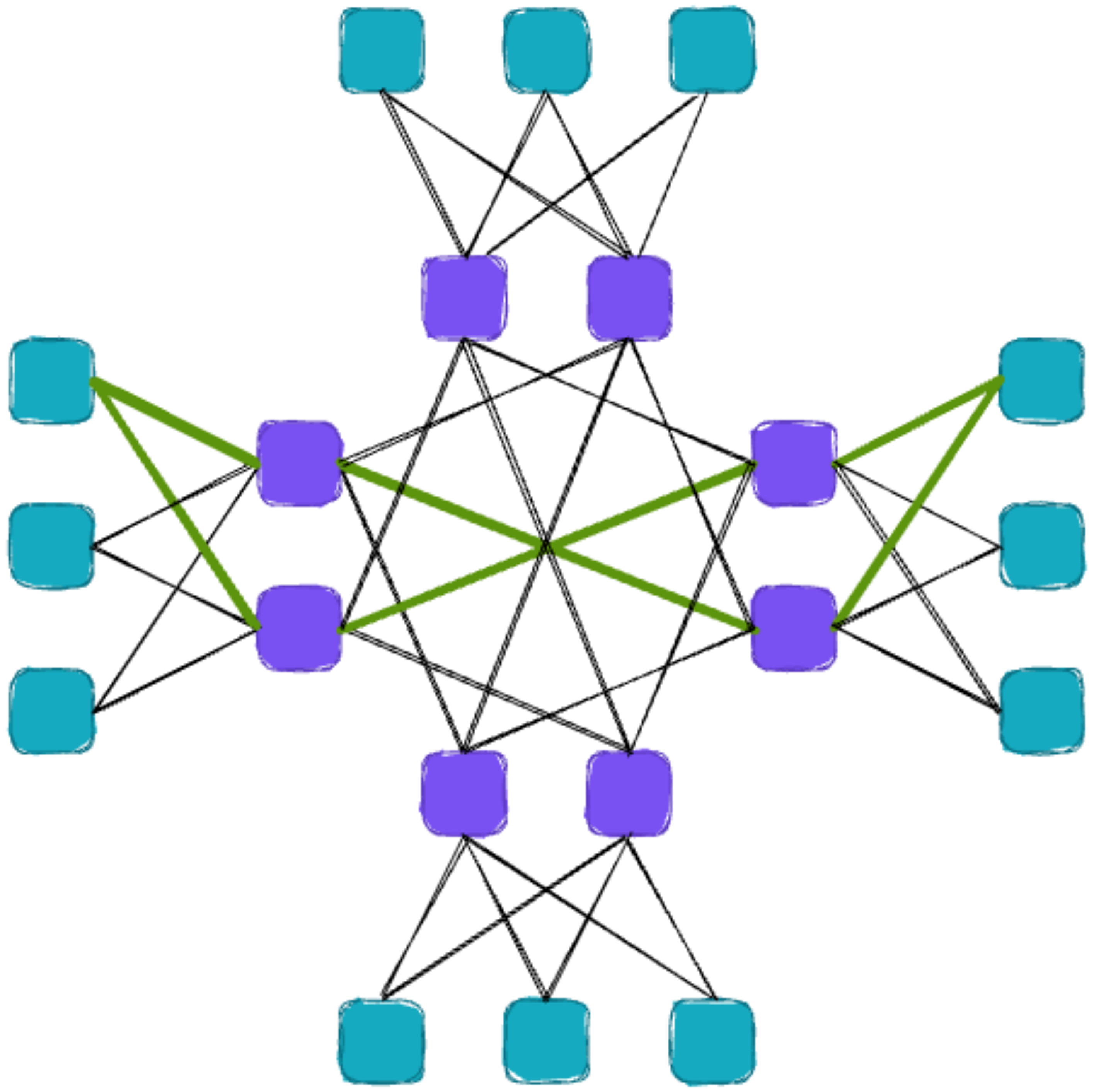


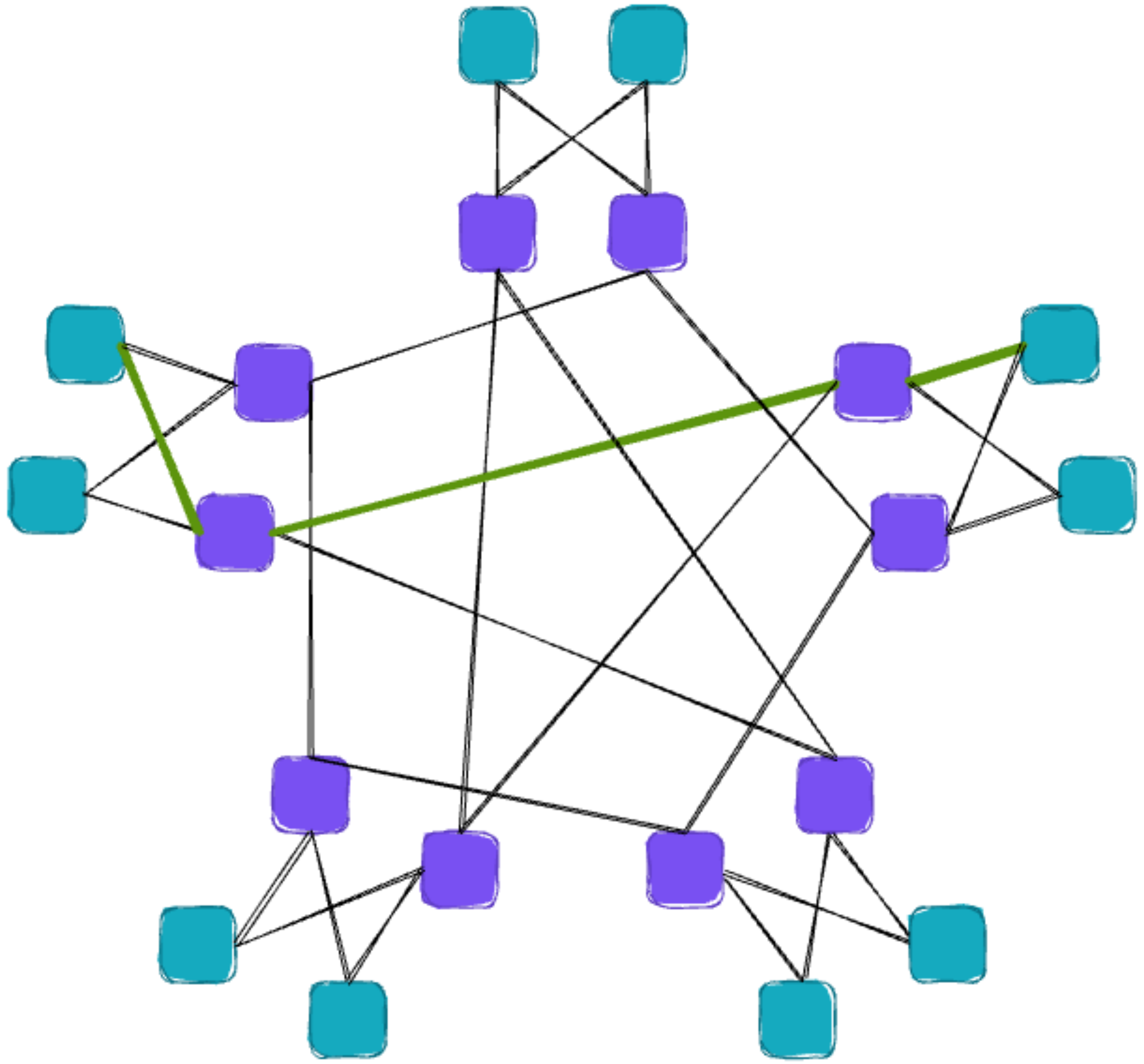
No



Paths in Dragonfly+

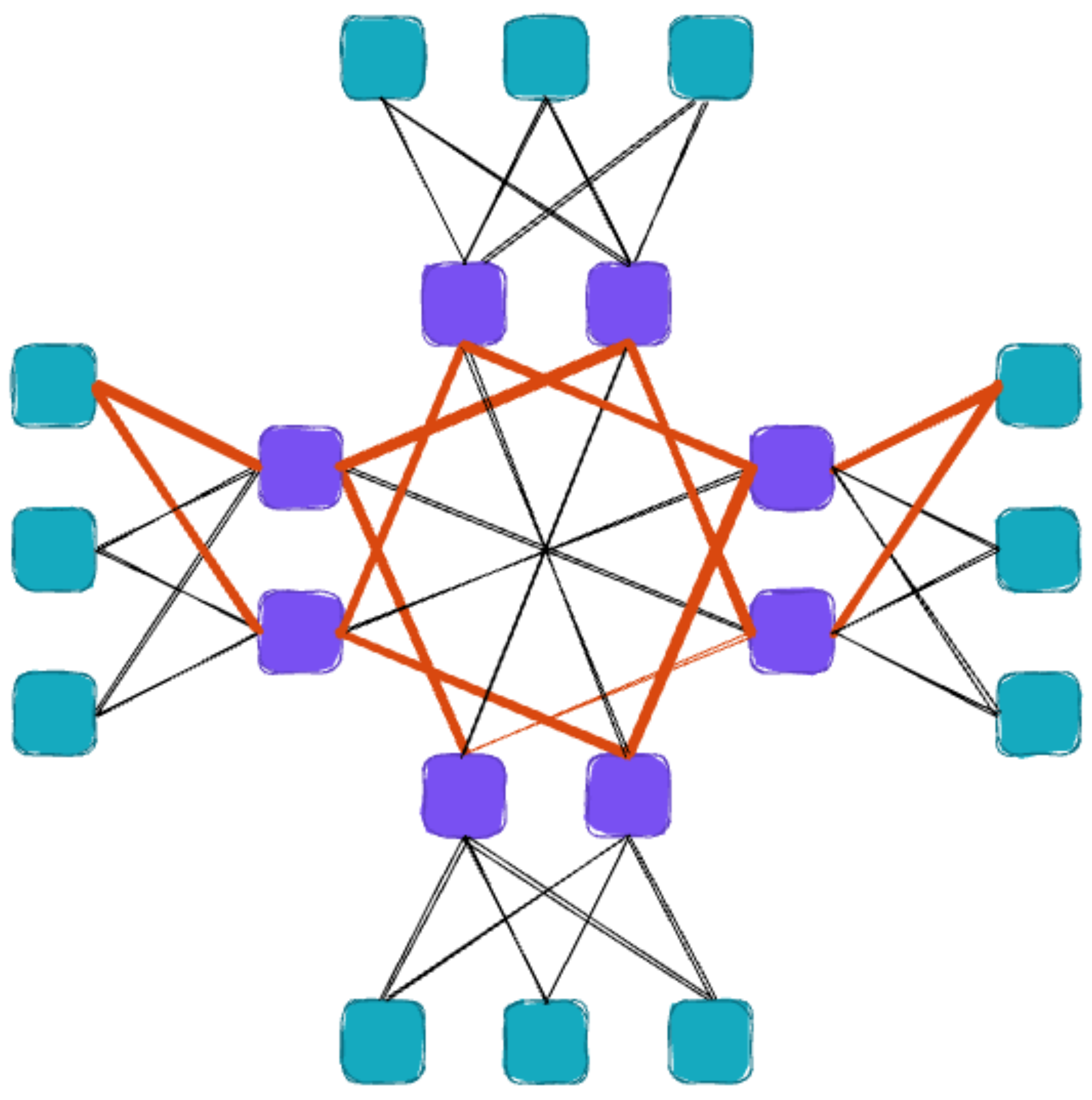
Minimal

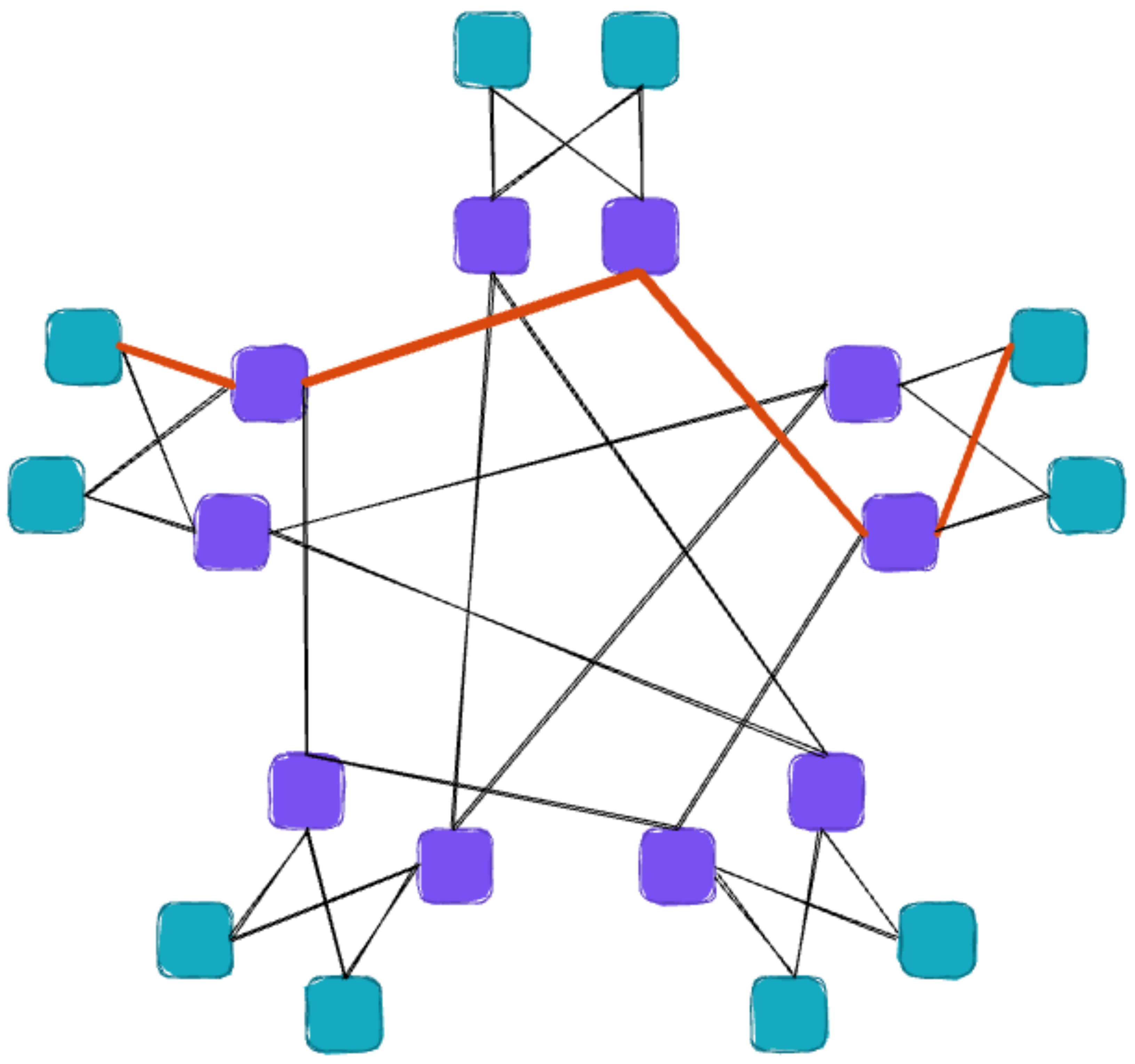




Paths in Dragonfly+

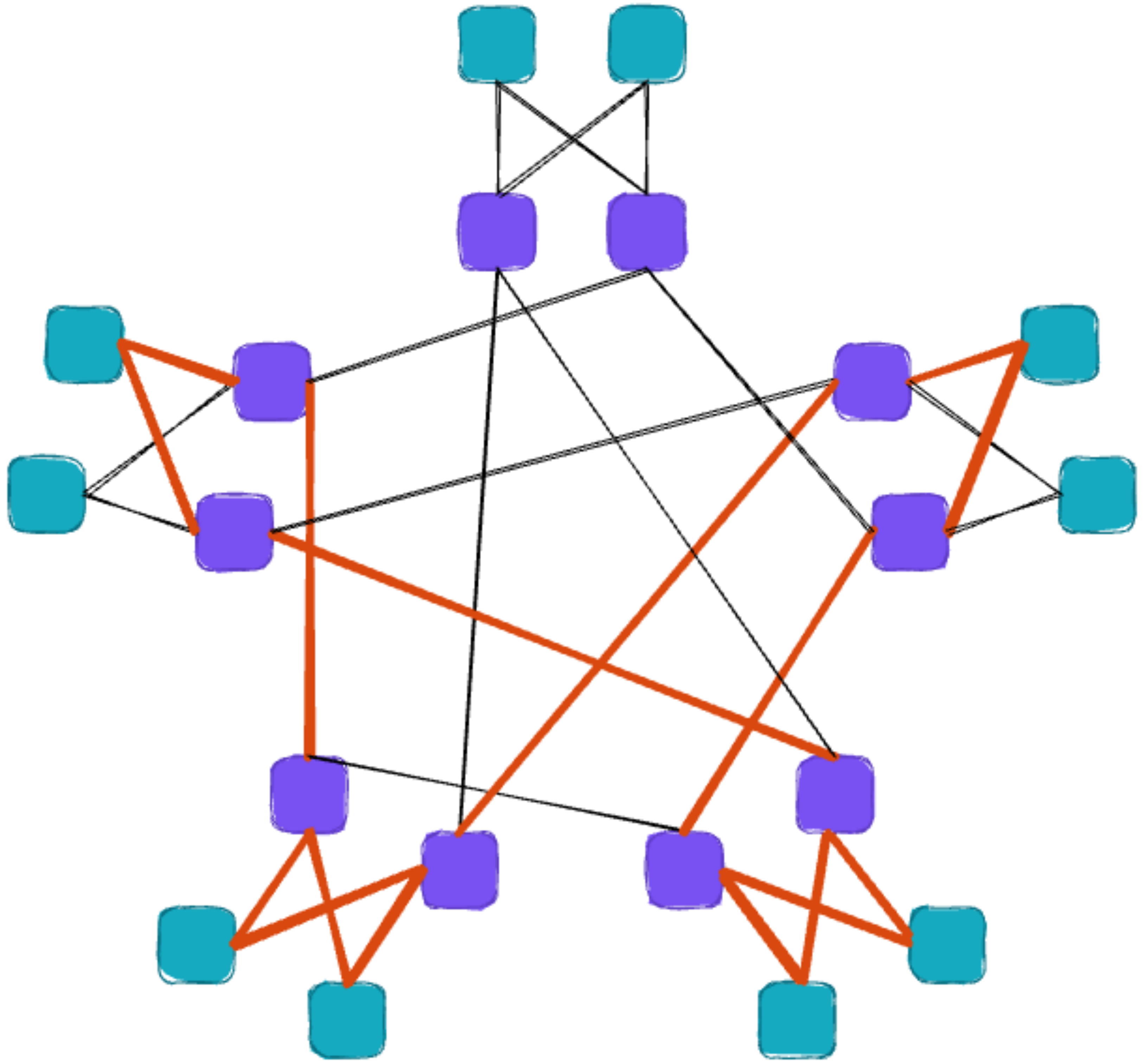
min+1





Paths in Dragonfly+

min+3

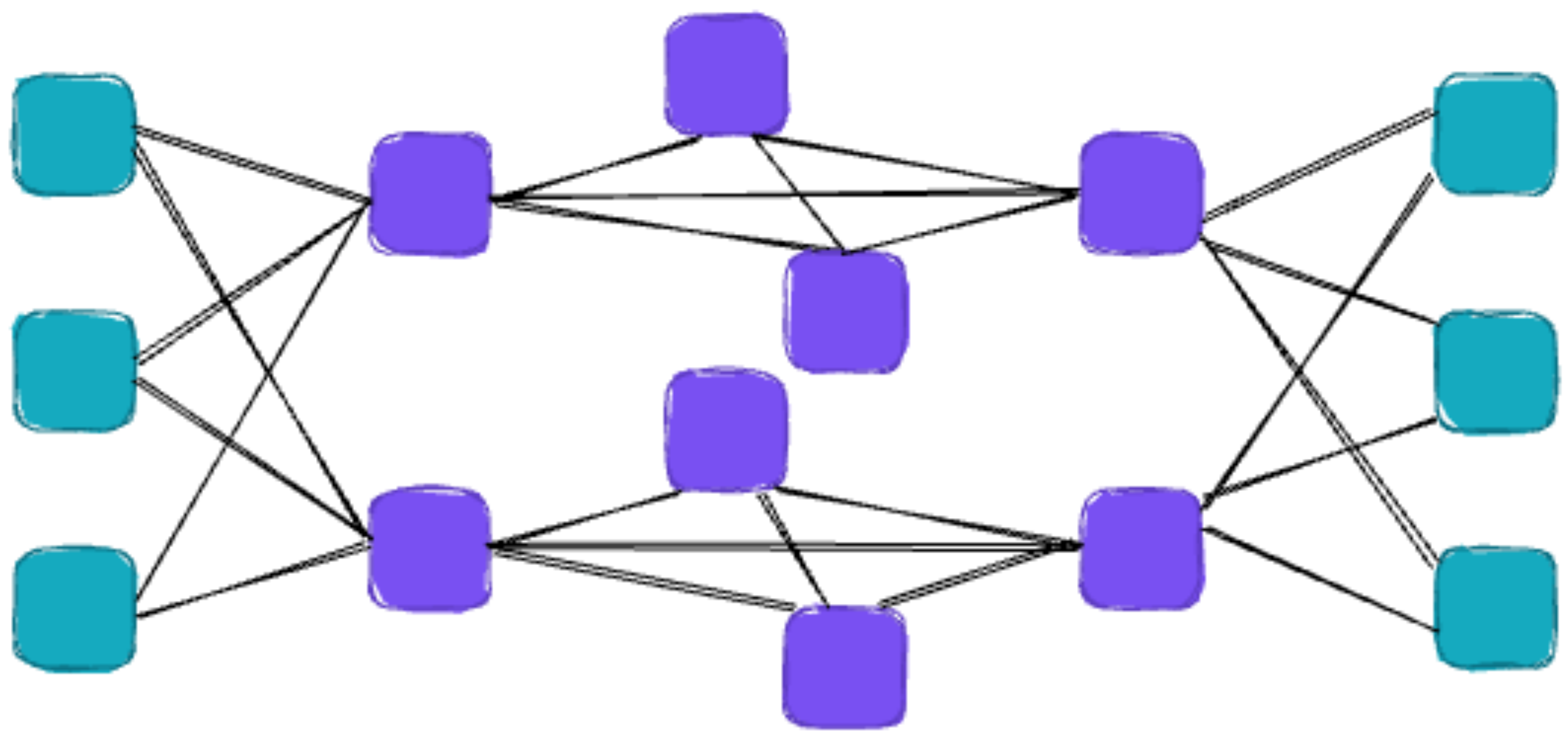


min+1 vs min+3 paths

- min+1
  - Distance to the destination does not increase along the path
    - and there only one hop where it stays constant
  - it's possible to make it work without source based routing
- min+3
  - Distance to the destination does increase along the path
  - requires proper source-based routing / all-at-once forwarding decision

Planes in Dragonfly+

Intra-group planes in Dragonfly+ with only min and min+1 paths



Non-minimal Forwarding with VRFs

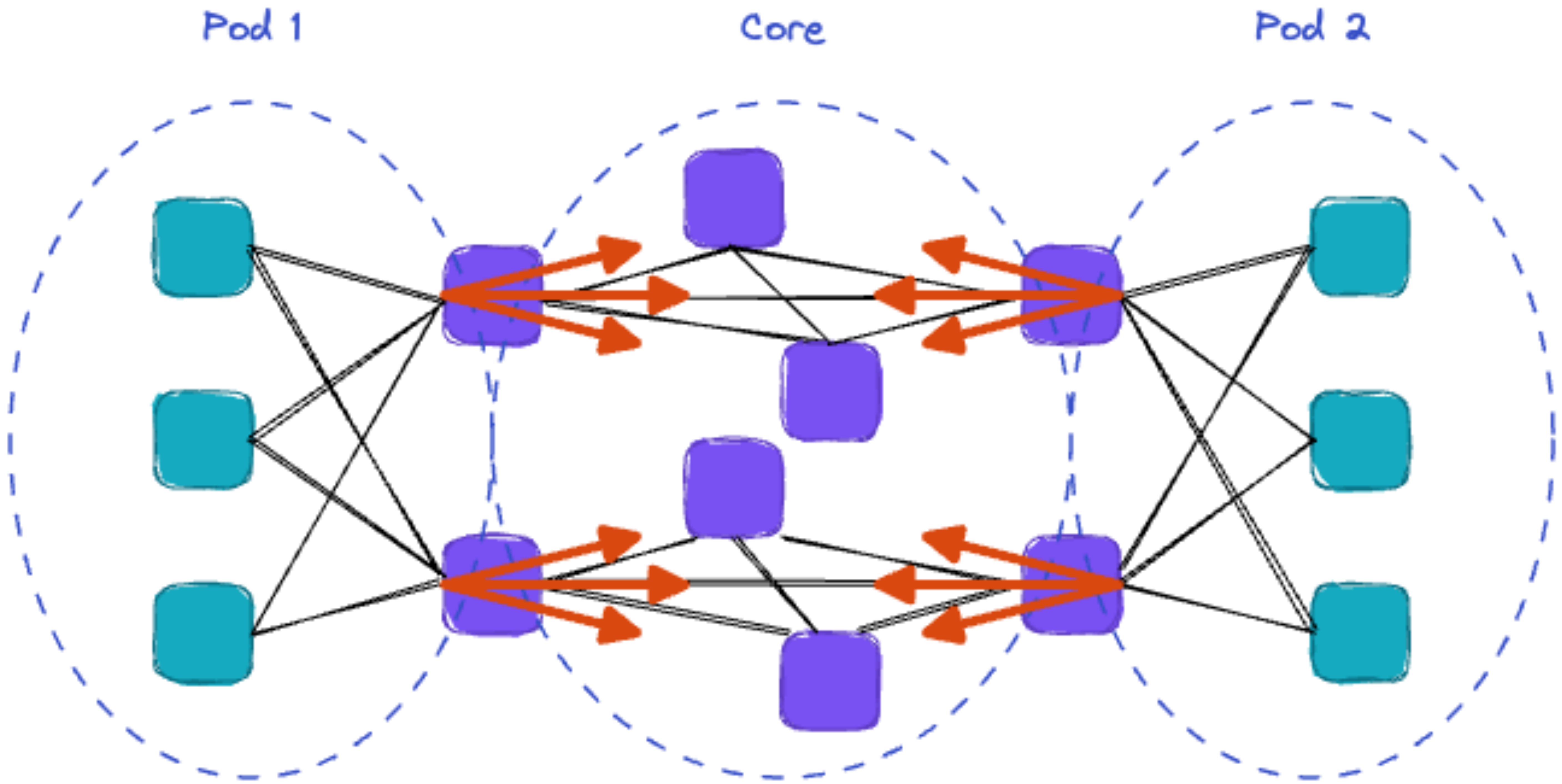
- Use separate VRFs for pods and core:
- only minimal paths in the core VRF
- in the pod VRF
    - minimal intra-pod paths
    - ECMP towards other pods

Google mentioned using VRFs to prevent loops in SIGCOMM paper describing Dragonfly-like topology:

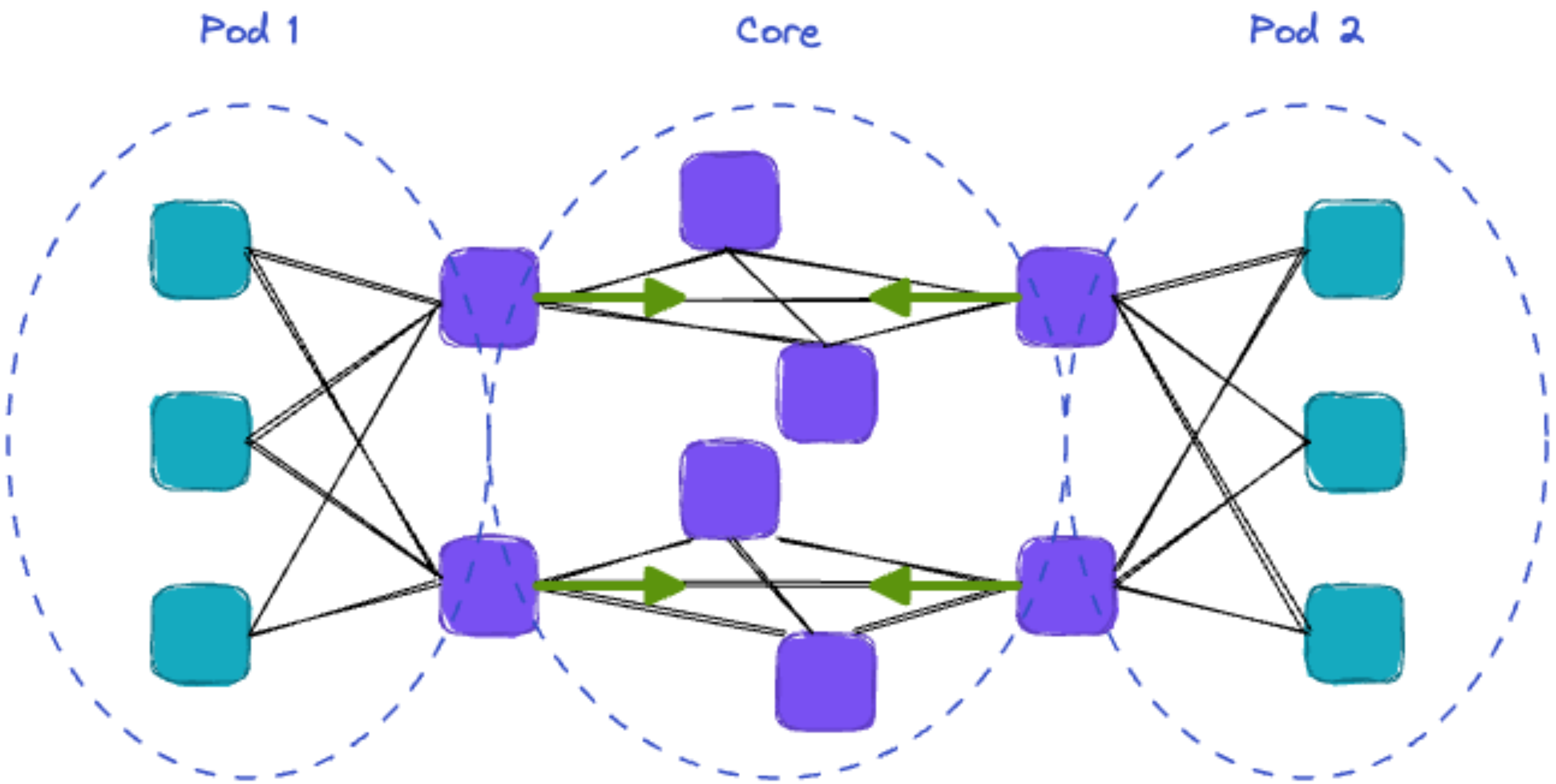
https://dl.acm.org/doi/10.1145/3544216.3544265

https://www.youtube.com/watch?v=Hfl-i56hZUg

Non-minimal Forwarding with VRFs: Pod



Non-minimal Forwarding with VRFs: Core



Non-minimal Routing with BGP

- BGP policies allow to implement additional logic taking into account network topology:
- Simple counting scheme to limit number of hops announce will travel in the core and to prevent path hunting
  - Add C1 when sending announce to the core (if neither C1 nor C2 are present)
  - Add C2 when propagating announce with C1
  - Don't propagate announces with C2
- Make min (C1) and min+1 (C1 & C2) routes eligible for ECMP or WCMP on import into the pod VRF:
  - prepend AS-PATH for routes with C1 only
  - or rewrite AS-PATH

Adaptive Routing

- ECMP or WCMP are not really adequate to express what we want to do:
  - use minimal paths first
  - use non-minimal paths when minimal ones are filled
  - try to dynamically shift existing traffic in case of congestion
- We need adaptive routing
  - requires some long lived artefacts (like flows)
    - otherwise there is nothing to move
  - spraying is easy to implement but can't adapt

Adaptive Routing

- Global
  - based on path properties
- Local
  - based on egress queue occupancy
  - supported on many new devices but of limited use
    - local state is not representative of path state

Global Adaptive Routing is Reactive

- proactive would need accurate current network state per queue, max few RTTs old - a lot of data to collect and distribute
  - RTT in DCN is ~ 10 us, state can change significantly over several RTTs
  - 10s to 100s of 1000 of links
  - multiple queues per link
  - need to distribute 100s of 1000s of parameters @ 10000+ Hz

Adaptive Routing with ECN and Flow Label

- There is no adaptive routing and ARNs (adaptive routing notifications) or similar mechanisms in IP
- But we have ECN and flow label
- ECN to detect congestion
  - works only with ECN capable transport
  - doesn't provide info about point of congesiton
  - still better than nothing
- flow label to influence flow to path mapping

Adaptive Routing with ECN and Flow Label

- Need to modify reaction to congestion:
    - adjust congestion control parameters (as usual)
    - or change flow label to pick some other path
- picking another path randomly is fine - we don't and can't know up to date global queue state anyway
    - statistically traffic will move away from more congested paths to less congested

Google described similar mechanism:

https://dl.acm.org/doi/10.1145/3544216.3544226

https://www.youtube.com/watch?v=j5pKdU2Lad0&list=PLU4C2_kotFP2rg92oGchLFN0Y7F3liFio&index=15

Adaptive Routing with ECN and Flow Label

- Open questions:
  - how to decide what to do - shift flow or adjust congestion control?
  - how adapt quickly and minimize reordering ?
  - shifting flow too many times in a short period probably not going to help - add some sort of dampening?
  - moving old vs new flows?
- Cross-group work

Thank You!