Routing in Dragonfly Topologies

IETF 117 San Francisco

Dmitry Afanasiev
Roman Glebov, Yandex
Jeff Tantsura, Nvidia

Why New Topologies for Data Center?

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

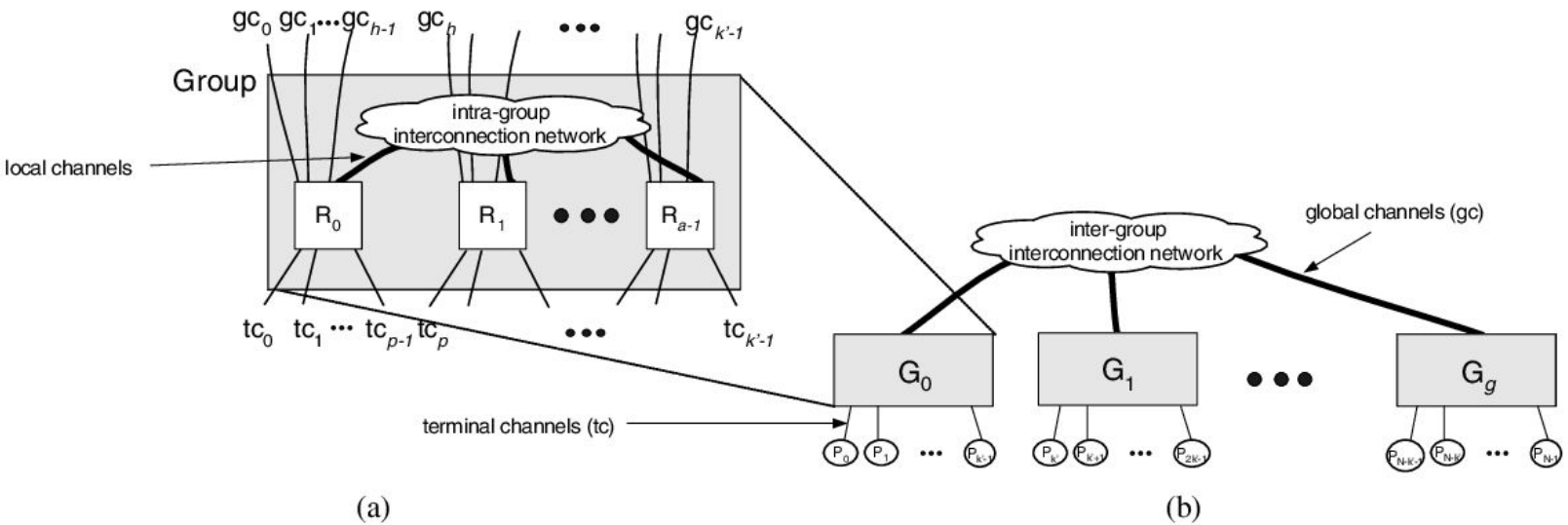
Dragonfly and Dragonfly+

Dragonfly Topology

- Dragonfly is a hierarchical topology:
  - groups - act like very high radix virtual routers
  - inter-group links - full mesh
- Groups:
  - Original Dragonfly is a direct topology with full mesh intra-group topology, also mentions flattened butterfly
  - Other intra-group topologies are possible resulting in different Dragonfly “flavors"
  - Dragonfly+/Megafly - leaf-spine
- Focus on
  - Reducing the number of long links and network diameter to reduce latency and cost of the network
  - Modularity
  - Path diversity
- Requires adaptive routing for efficient operation
- Widely deployed in HPC - default topology for default choice for the Cray XC series

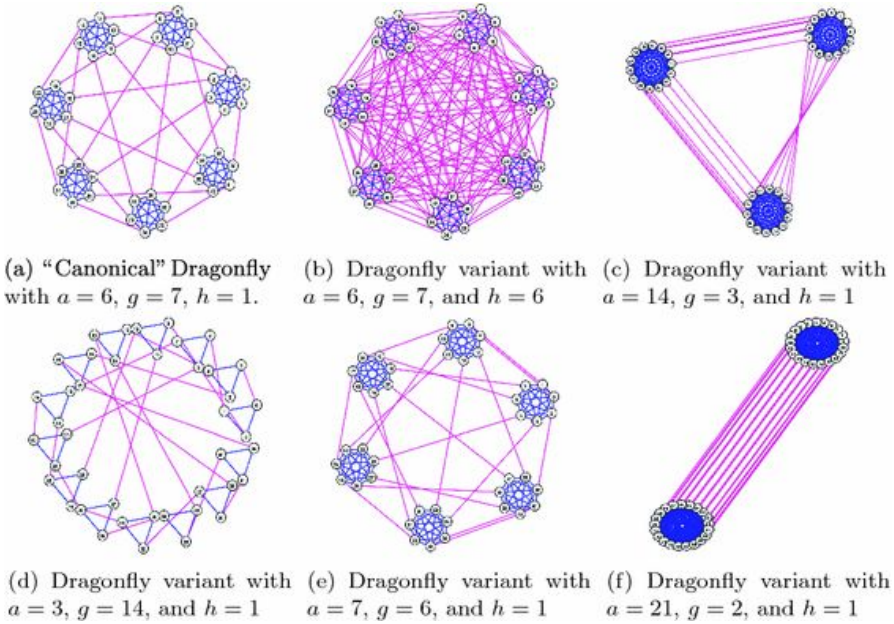
Technology-Driven, Highly-Scalable Dragonfly Topology

Dragonfly Topology (2)



ure 3. (a) Block diagram of a group (virtual router) and (b) high level block diagram of a dragonfly topology composed of multir

Dragonfly Topology (3)



(a) "Canonical" Dragonfly with $a = 6$, $g = 7$, $h = 1$.

(b) Dragonfly variant with $a = 6$, $g = 7$, and $h = 6$

(c) Dragonfly variant with $a = 14$, $g = 3$, and $h = 1$

(d) Dragonfly variant with $a = 3$, $g = 14$, and $h = 1$

(e) Dragonfly variant with $a = 7$, $g = 6$, and $h = 1$

(f) Dragonfly variant with $a = 21$, $g = 2$, and $h = 1$

https://www.researchgate.net/publication/320493515_Design_Space_Exploration_of_the_Dragonfly_Topology

Terminology

Dragonfly Scalability

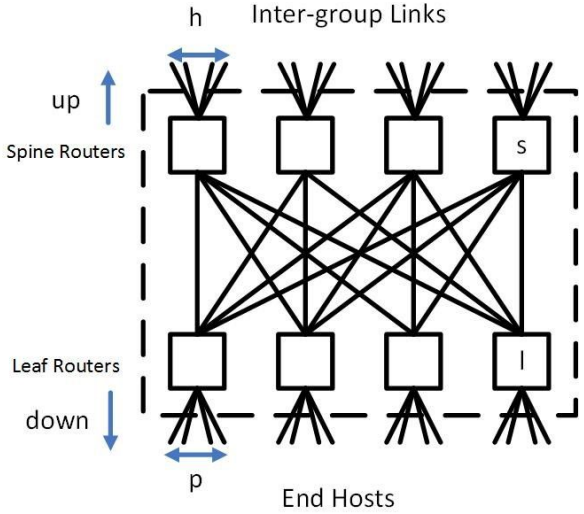
Dragonfly Scalability

In a maximum-size (N = ap(ah + 1)) dragonfly, there is exactly one connection between each pair of groups. In smaller dragonflies, there are more global connections out of each group than there are other groups. These excess global connections are distributed over the groups with each pair of groups connected by at least & ah+1

g ' channels

Dragonfly+ / Megafly

- Dragonfly+ http://dx.doi.org/10.1109/HiPINEB.2017.11
  - indirect hierarchical topology with high path diversity based on Dragonfly
    - spines are for transit only
  - full bipartite / leaf-spine intra-group topology
  - full mesh inter-group topology
- Also known as Megafly https://doi.org/10.1007/978-3-319-92040-5_15



See also

- 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

Sparse and Dense Dragonfly+

- In a maximum-size Dragonfly+, there is exactly one connection between each pair of groups.
- In smaller dragonflies, there are more global connections out of each group than there are other groups. These excess global connections are distributed over the groups

Dragonfly+ Scalability

Optim

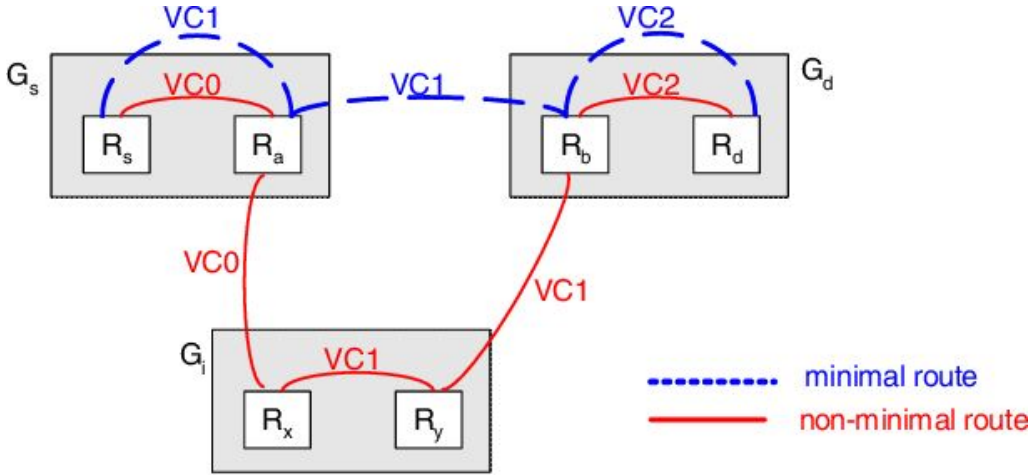
Paths and Routing in Dragonfly+

Some Notes on Routing

- Simplified view of routing is that it is just about connectivity
  - really it is about both connectivity and capacity
- Achieving capacity:
  - tricks/hacks that depend on graph symmetry - e.g. multipath (both ECMP and WCMP)
    - a lot of assumptions about symmetry
  - or explicitly dealing with many paths, including non-minimal
    - more forwarding state and more complex topology calculations
- Problems with relying on symmetry:
  - Does not stay perfect for long it the real world due to failures and changes
  - Many interesting topologies are not symmetric enough to use ECMP directly
  - Static mechanisms, traffic placement may be suboptimal
- It useful to have a way to deal with imperfect symmetry and non-minimal paths

Routing in (Original) Dragonfly

- Maximum 3 hops:
  one global, two local
  - Uses virtual channels
    - 1 hop may include several links
  - Global hop can go via intermediate group

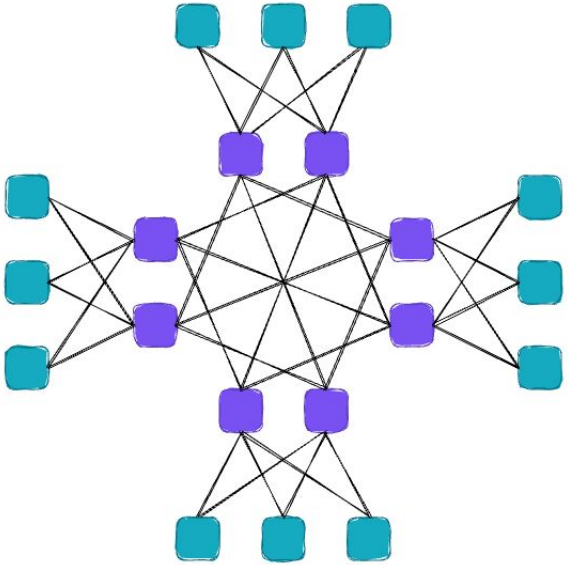


Routing in Dragonfly+

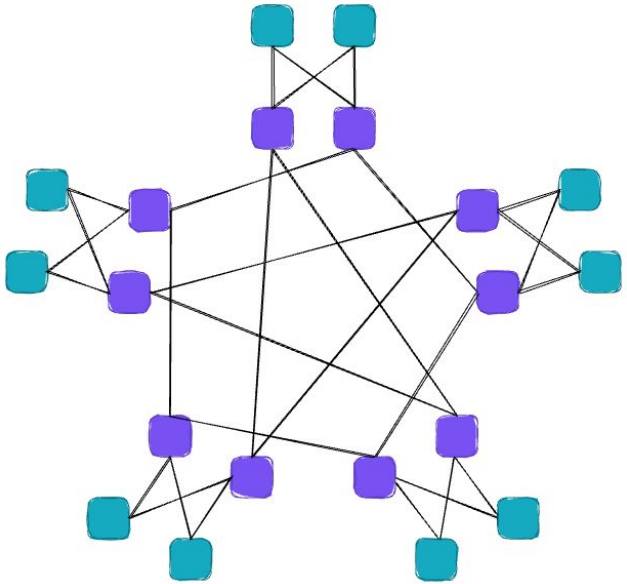
- Intra-group
  - Solved problem - it is just a leaf-spine
  - Minimal routing + ECMP is enough
- Inter-group
  - Same as in original Dragonfly
  - Non-minimal paths
  - Needs VLB or better adaptive routing

Paths in Dragonfly+ - Dense vs Sparse Topology

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

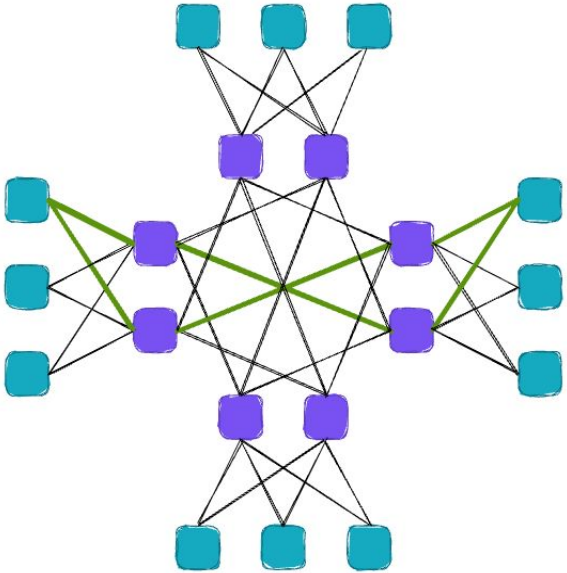


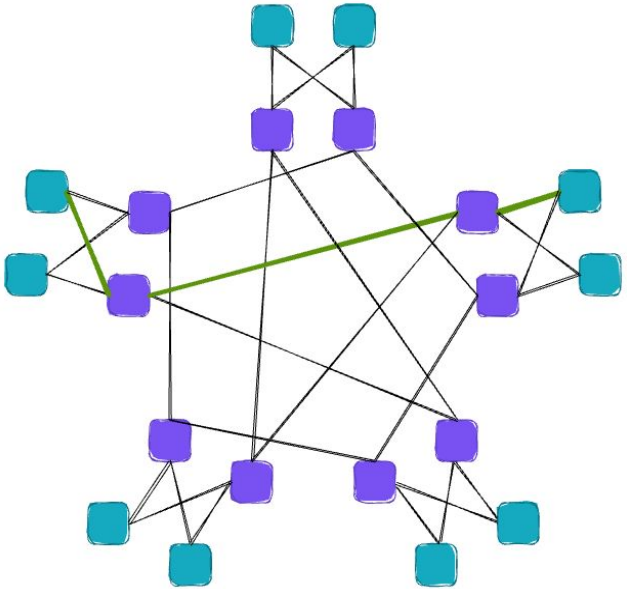
Yes - dense



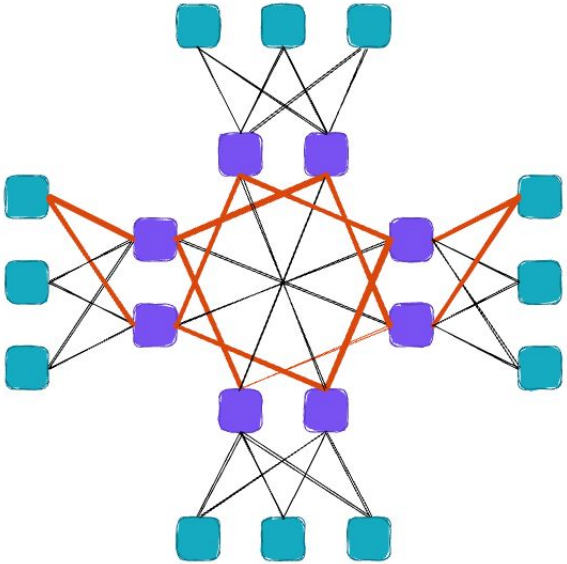
No - sparse

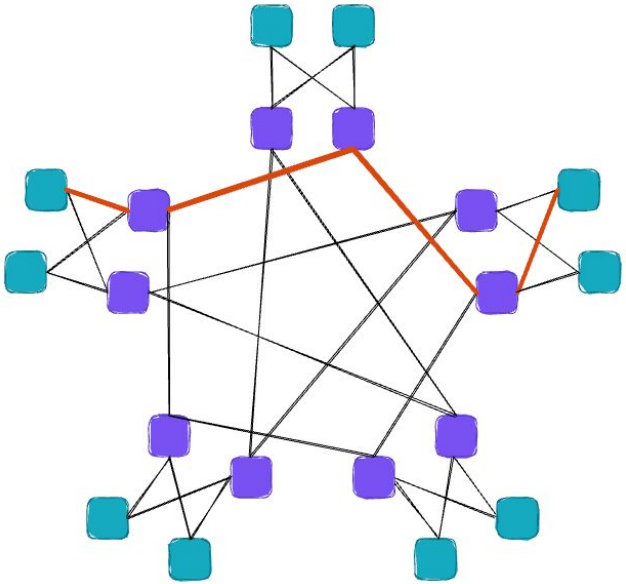
Paths in Dragonfly+: min/LGL





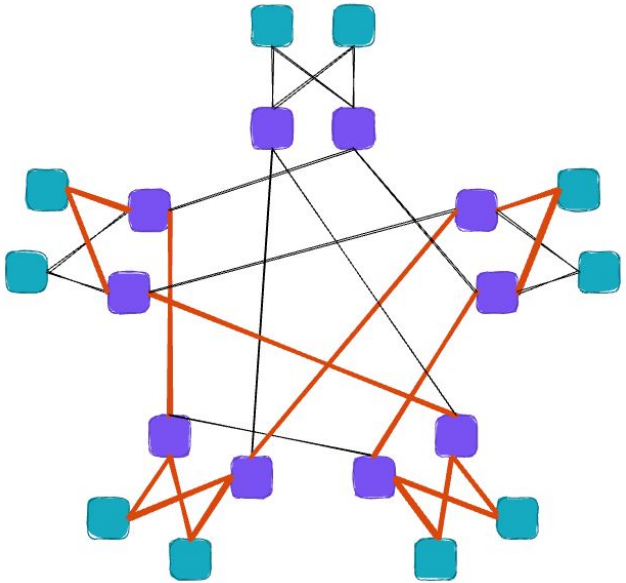
Paths in Dragonfly+: LGGL





Paths in Dragonfly+: LGLLGL

- Ingress spine != egress spine in transit group
  - Appear only in sparse topologies
  - Could be useful in dense Dragonfly+ with asymmetrical inter-group capacity but will be preempted by LGGL paths

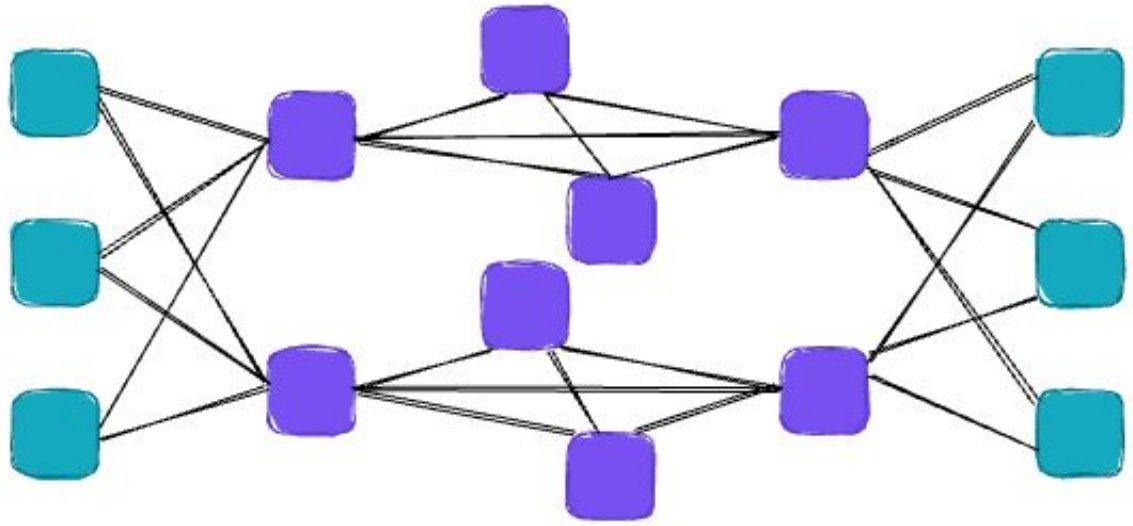


Non-minimal paths: LGGL vs LGLLGL

- min+1 / LGGL
  - Distance to the destination does not increase along the path
  - And there only one hop where it stays constant
  - Easy to make it work without source based routing
- min+3 / LGLLGL
  - Distance to the destination does increase along the path
  - Still can be implemented but with a lot of extra tricks
    - Additional transit VRF acting as a virtual link
    - More complex import/export policies

Planes in Dragonfly+

By grouping corresponding spines in each group inter-group connectivity can be represented as a set of planes:



Routing in Dragonfly+ with BGP and VRFs

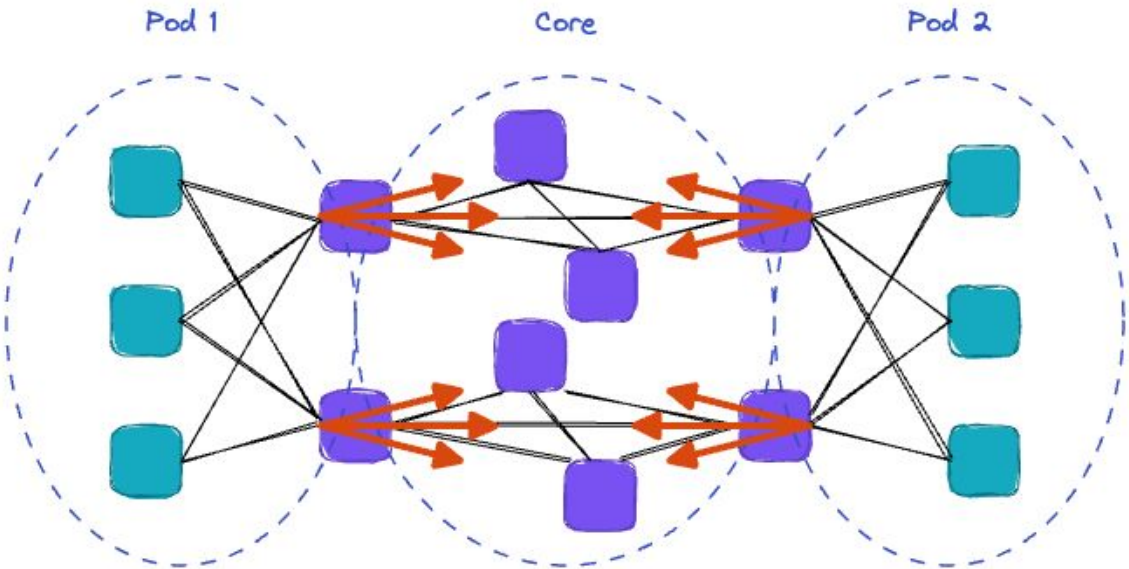
Non-minimal Forwarding with VRFs

- Use separate VRFs for intra-group and inter-group/core:
- Inter-group VRF
  - Only minimal paths in the core VRF
- Intra-group VRF
  - Minimal intra-group paths
  - ECMP towards other groups - route leaking, next hops in core VRF

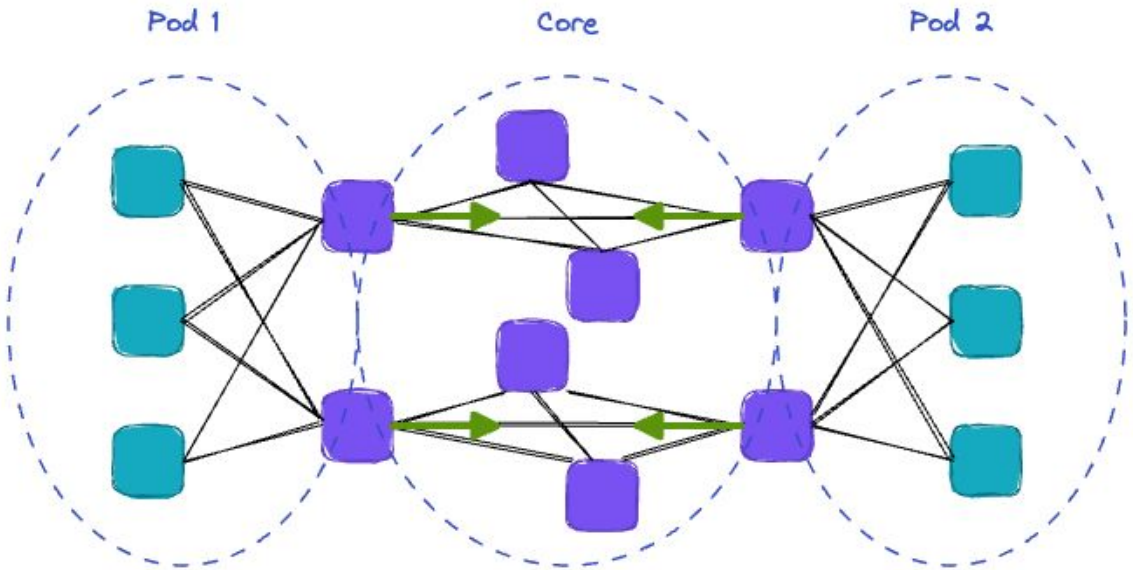
Google mentioned using VRFs to prevent loops in SIGCOMM paper describing Dragonfly-like topology:

- https://dl.acm.org/doi/10.1145/3544216.3544265
- https://www.youtube.com/watch?v=Hfl-i56hZUg

Non-minimal Forwarding with VRFs: Group



Non-minimal Forwarding with VRFs: Core



Non-minimal Forwarding with VRFs: LGLLGL Paths

- Paths going through intermediate group where ingress and egress spines are different
  - more difficult to implement
  - required only in sparse Dragongly+
- Require additional transit VRF (~ virtual link) in each group
  - Used only for transit
  - Imports only shortest paths from the core
  - Reflects announces via all leaves to other spines - breaks valley-free routing
  - Announces reflected via leaves are only propagated further in the core if spine does not have shorter paths
  - Used for load balancing when imported in other groups
- Can't use intra-group VRF because it imports non-minimal paths and doing ECMP - will cause loops
- Require some form of multiplexing since transit and intra-group VRFs are using the same leaf-spine links

Non-minimal Routing with BGP - LGGL

- 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

Non-minimal Routing with BGP - LGLLGL

- TODO - how import and leaking works
-

Adaptive Routing (Forwarding)

Note on Terminology

What is traditionally called adaptive routing in HPC interconnect world

- is not about topology and path discovery
- traffic to paths mapping mechanism
- corresponds to forwarding and load balancing in IP networks

Need for Adaptive Routing (Forwarding)

- ECMP or WCMP or any static mechanism is 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
- Dynamic adaptive forwarding is required to efficiently utilize capacity in networks with non-minimal paths
  - requires some identifiable long lived artefacts - flows
    - otherwise there is nothing to move

Spraying is not a Solution

- Spraying (per-packet LB) works well when:
  - Topology is highly symmetrical - e.g. multi-stage Clos
  - Everybody is doing spraying
  - Still can cause minor reordering
- Suboptimal in topologies with non-minimal paths
- Easy to implement but can't adapt
  - per-hop decisions while we need in e2e/per-path
  - no long-lived traffic artefacts to map to paths and to shift during congestion

Adaptive Routing (Forwarding)

- Global
  - Based on global state - e.g e2e path properties and congestion
- Local
  - Based on local state - e.g. egress queue occupancy
  - supported on many new devices but of limited help
    - local state is not representative of path state
  - Can improve initial traffic placement

Global Adaptive Routing in Data Center is Reactive

- Proactive adaptive routing would need accurate current network state info
  - 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

Tools we don't have

Traffic management tools usually available in HPC interconnects but not in IP networks:

- ARNs (adaptive routing notifications) or similar mechanisms in IP
- Virtual channels
- Credit-based flow control

Tools we have

- ECN
  - Can detect congestion
  - But has to go full round trip, no signaling directly from congestion point
  - doesn't provide info about point of congesiton
  - Works only with ECN capable transport
  - Return signaling and reaction are transport-dependent
- Flow label
  - Can influence flow to path mapping
- Better than nothing :)

Adaptive Routing (Forwarding) with ECN and Flow Label

- 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
  - works as long as statistically traffic will move away from more congested paths to less congested
- Can be generalized to other entropy headers

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

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

Where in IETF?

- Adaptive routing is difficult to place in IETF
  - Congestion control, entropy headers, midpoint signaling etc. are all in different groups
  - Often with strong opinions how things should/should not be done :)

Thanks!