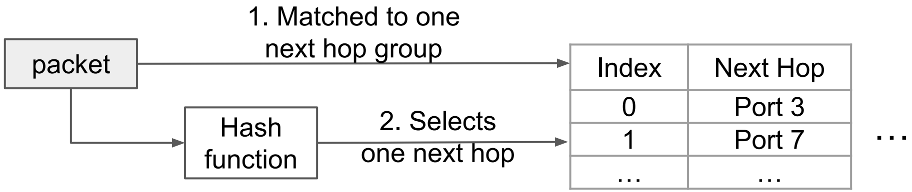
Hashing Design in Modern Networks: Challenges and Mitigation Techniques

Yunhong Xu, Keqiang He, **Rui Wang**, Minlan Yu, Nick Duffield,
Hassan Wassel, Shidong Zhang, Leon Poutievski, Junlan Zhou, Amin Vahdat

TEXAS A&M UNIVERSITY
HARVARD UNIVERSITY
Google

Multipath forwarding in modern networks

- Datacenter and wide-area networks employ multipath forwarding: equal-cost multipath (ECMP) and its weighted variant (WCMP).



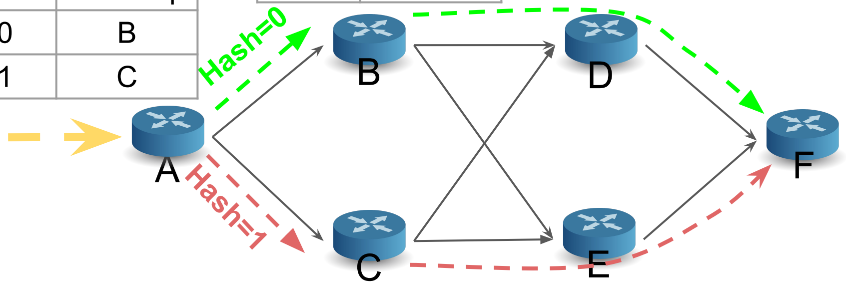
- A good hash function should hash flows equally across the next hops.

Hash polarization

- Hash function reuse at different switches can lead to poor load balancing.

Hash	Next hop
0	B
1	C

Hash	Next hop
0	D
1	E



4 equal-cost paths:
A → B → D → F
A → B → E → F
A → C → D → F
A → C → E → F

Only 2 out of 4 paths are used!

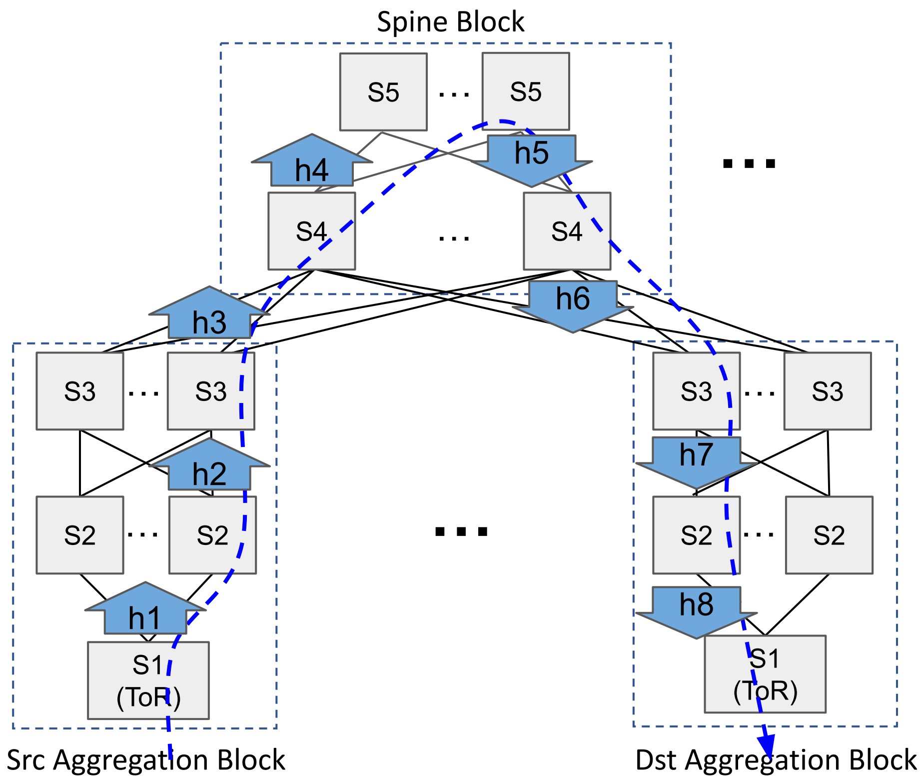
- Poor hashing leads to
  - High tail latency detrimental to application performance, e.g., ML collective communication.
  - Usable bandwidth << provisioned bandwidth.

Our contributions

- Uncovered the hashing problem
    - Lack of uncorrelated hash functions in commodity switch (random seeds are ineffective!)
    - Shortage becomes more acute with increasing adoption of direct-connect topologies [Sigcomm'22]
- Identified a common forwarding structure that enables hash function reuse
- An SDN-based approach to decorrelate switches using the same hash function
- Our solutions are close to optimal for DCN, reclaiming 33% of network capacity otherwise lost to poor hashing.

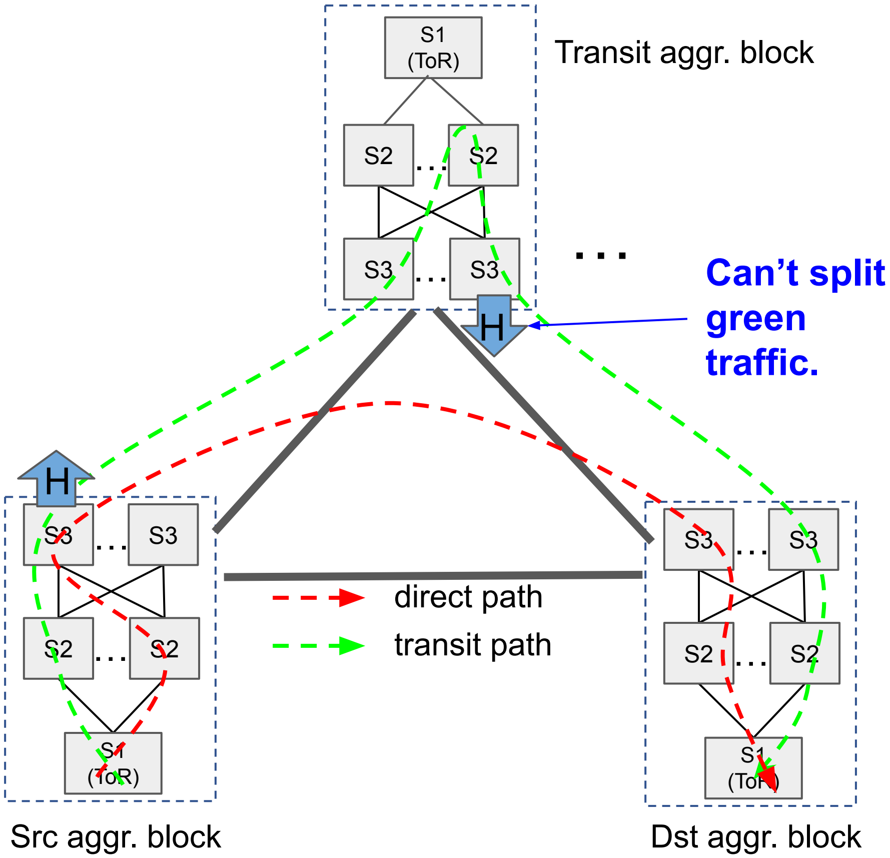
Hashing requirements of Google’s datacenter networks

- **Multi-stage Clos** comprising 5 stages of switches [*Jupiter Rising, Sigcomm’15*]
- Every inter-aggregation-block path has 8 hops and goes through the same stage sequence.
- Ideally 8 uncorrelated hash functions are needed. S2, S3, and S4 each has to have two hash functions for northbound and southbound traffic, respectively.
- We call this *per-stage allocation*.



Hashing requirements of Google's datacenter networks

- **Direct-connect topology** meshes aggregation blocks without spineblocks. [*Jupiter Evolving, Sigcomm'22*]
- Inter-aggregation-block traffic goes over direct links and transits other aggregation blocks.
- Per-stage allocation lead to hash correlation between src->transit and transit->dst.
- Ideally, each aggregation block should be assigned a unique hash function, which means dozens of hash functions.

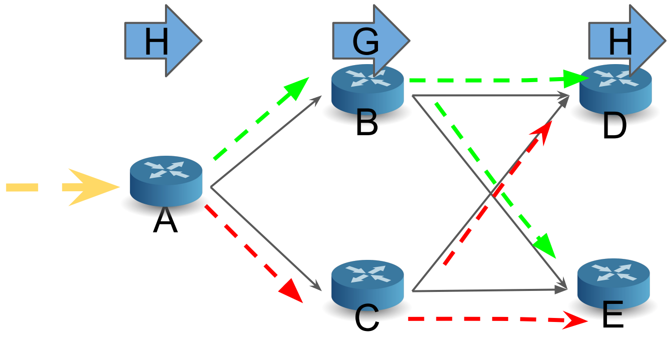


However…

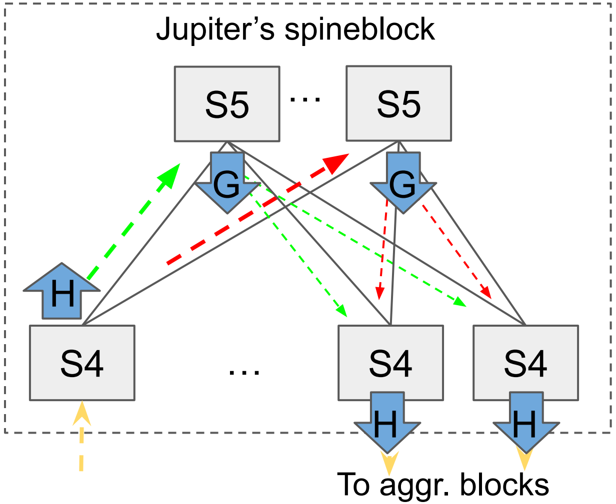
- Commodity switches offer limited set of hash functions.
- Common implementation of switch-unique seed is not effective in decorrelating two switches using the same hash function
  - The seed is concatenated the input bit vector to the hash function.
  - The effect of random seeding is similar to scrambling the next hops.
  - Seed’s inability to decorrelate hash functions is shown by both theoretical proof as well as simulation. (subsection 2.3.2 of the paper)

Hash function reuse – color recombination

- B and C have to use an orthogonal (uncorrelated) hash function to A's to avoid polarization.
- D and E can reuse A's hash function because its input traffic is no longer polarized.



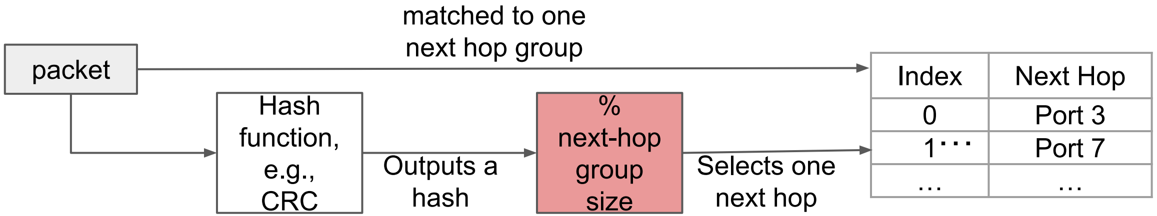
- Color recombination applied



2-stage folded-Clos. Color recombination enables S4 to use the same hash function for northbound and southbound traffic.

Hash function reuse – Copriming

- There are really *two* hash operations involved in selecting the next hop of a packet.



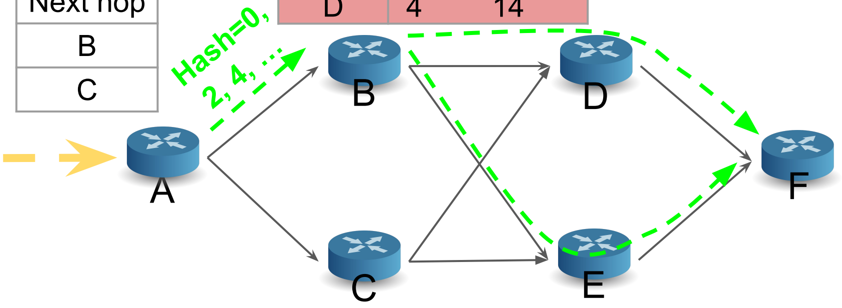
- If the (ECMP/WCMP) group sizes across two switches are *coprime*, the two switches are uncorrelated even if they use the same hash function. (Theorem 2 in the paper).
- Group sizes are configured by the SDN controller.

Copriming illustration

- A weight skew may be introduced when a group is expanded to coprime size by entry replication.
- A trade-off exists between weight precision and switch table usage.
- Copriming a WCMP group is a little more tricky (see the paper).

Next hop	Hash
D	0 2 10 …
E	6 16
D	2 12
E	8 18
D	4 14

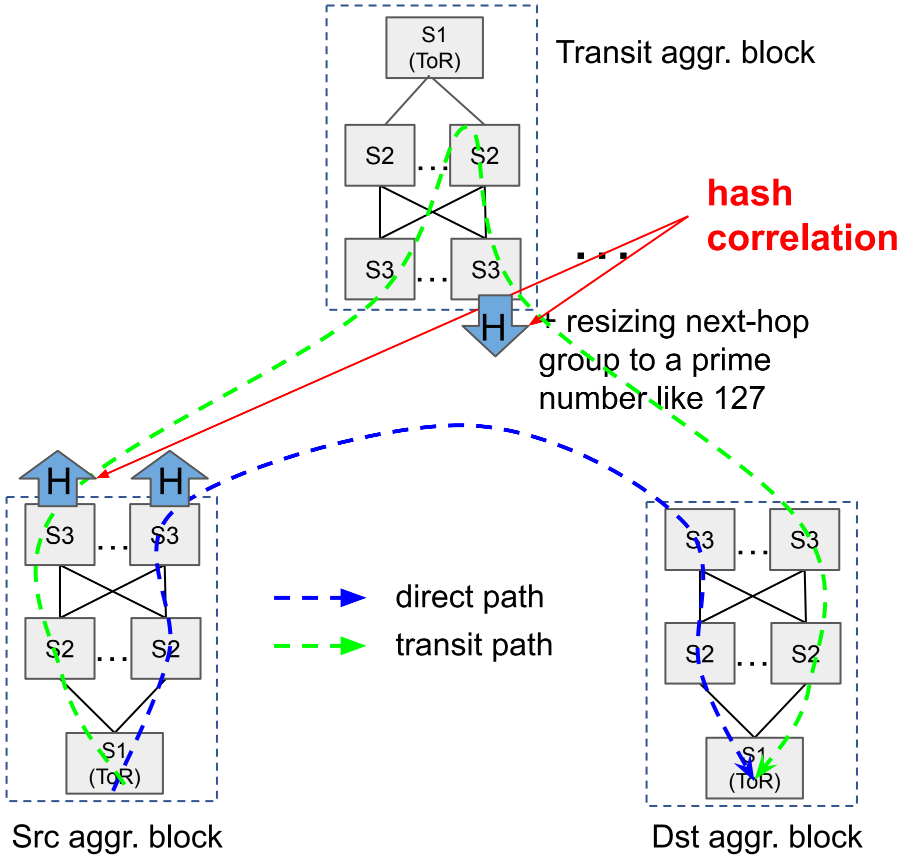
Next hop
B
C



A → B → E → F is now used along with A → B → D → F, with a traffic split of 2:3 (vs. intended 1:1).

Copriming applied

- Need to decorrelate transit->dst from src->transit.
- Resize transit->dst group to a prime number, which should be coprime with src->transit group size.
- transit->dst tends to be ECMP whereas src tends to do WCMP for traffic engineering. Resizing ECMP group produces less weight skew.



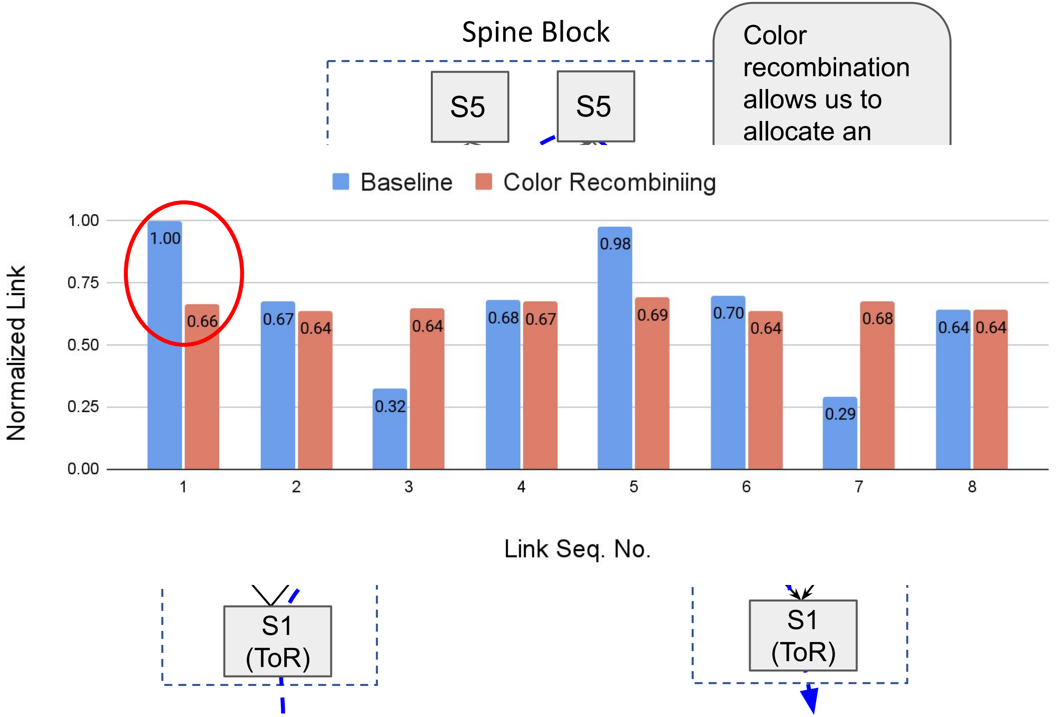
Simulation

- Network topologies
- Traffic traces
  - CAIDA trace dataset [caida.org]
- Hash functions
  - RTAG7
    - Widely deployed
    - Includes seven hash functions
- Metrics
  - Normalized Link utilization
  - CV (Coefficient of Variance)
- Seed
  - Each switch is configured with a random seed.

Topology	#Nodes	#Links
Multi-stage Clos DCN	2 aggr. blocks 1 spine block	128
Direct Connect DCN	8 aggr. blocks	512

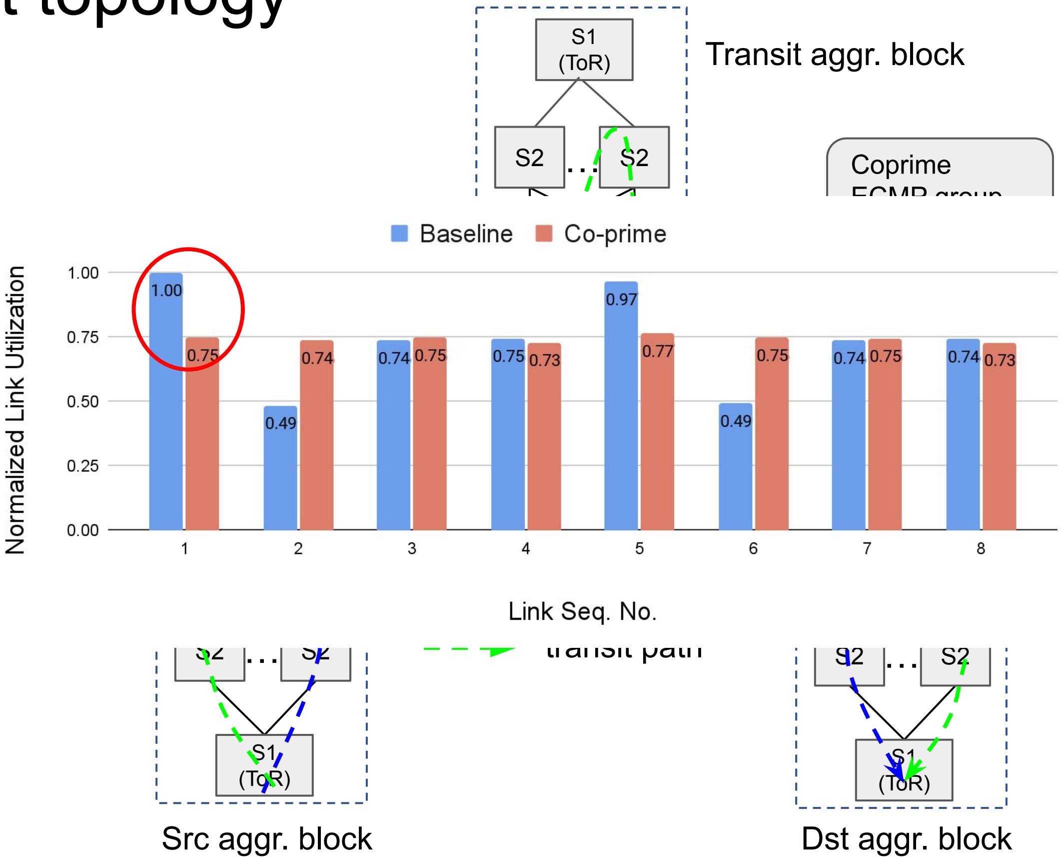
Simulation: Clos topology

- Configurations
  - Baseline: Same hash function for north and south bound traffic at S3.
  - Color Recombining: different hash functions for north and south bound traffic.
- Load balance for a representative S3
  - CV: 0.6 vs. 0.05
- Takeaway
  - Color recombining increases effective capacity by **33%**.



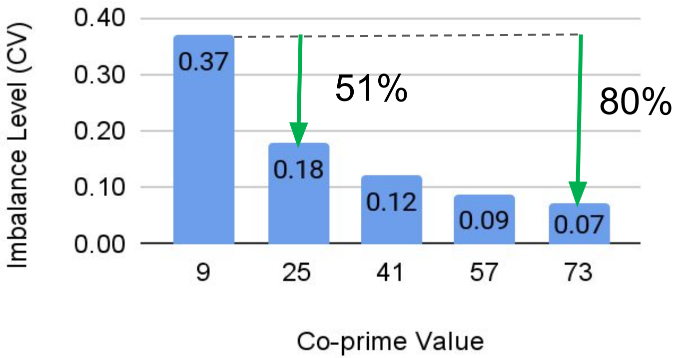
Simulation: direct connect topology

- Baseline: original group size without Copriming
- Load balance for a representative S3
  - CVs
    - Baseline: as large as 0.5
    - co-prime: CV < 0.1
- Takeaway
  - Copriming improves effective capacity improved by **23%**



Trade-off between weight skew and switch table usage

- Takeaways
  - By increasing the coprime value from 9 to 73,
    - Load imbalance is reduced by 80%.
    - ECMP group size is increased from 9 to 73, using more table space on the switch.
    - The largest reduction of imbalance (51%) is when coprime value is increased from 9 to 25, with diminished return afterwards.



Production Deployment

- Color recombination and Copriming have been widely deployed in Google's data center networks for many years.
  - Color recombination is employed in Jupiter [Jupiter Rising, Sigcomm'15]
  - Color recombination and Copriming are employed in Direct Connect Jupiter [Jupiter Evolving, Sigcomm'22]
  - Excellent load balancing performance with CV of 0.03

Future works

- Hash polarization detection in live networks.
    - Attributing load imbalance to hash polarization in the face of multiple confounders like, WCMP, traffic engineering, elephant flow (or traffic entropy issue) can be challenging.
    - Intermittent hashing issue is even harder. E.g., Good color mixing (most of the time) that hides the hashing issue.
- Better switch hashing support.
    - Truly achieve “unlimited” uncorrelated hash functions, generated by seeding the same function randomly.
- Auto detection and discovery of color recombination patterns.

Conclusion

- Hash polarization is a real yet underestimated problem in large-scale production networks, reducing usable bandwidth by as much as 33%.
- Two proven approaches to achieve near-optimal hashing.
    - A color recombining pattern for Clos networks
    - A SDN-based Copriming technique for non-hierarchical mesh networks
- Collaboration with vendor on better switch hashing implementation.