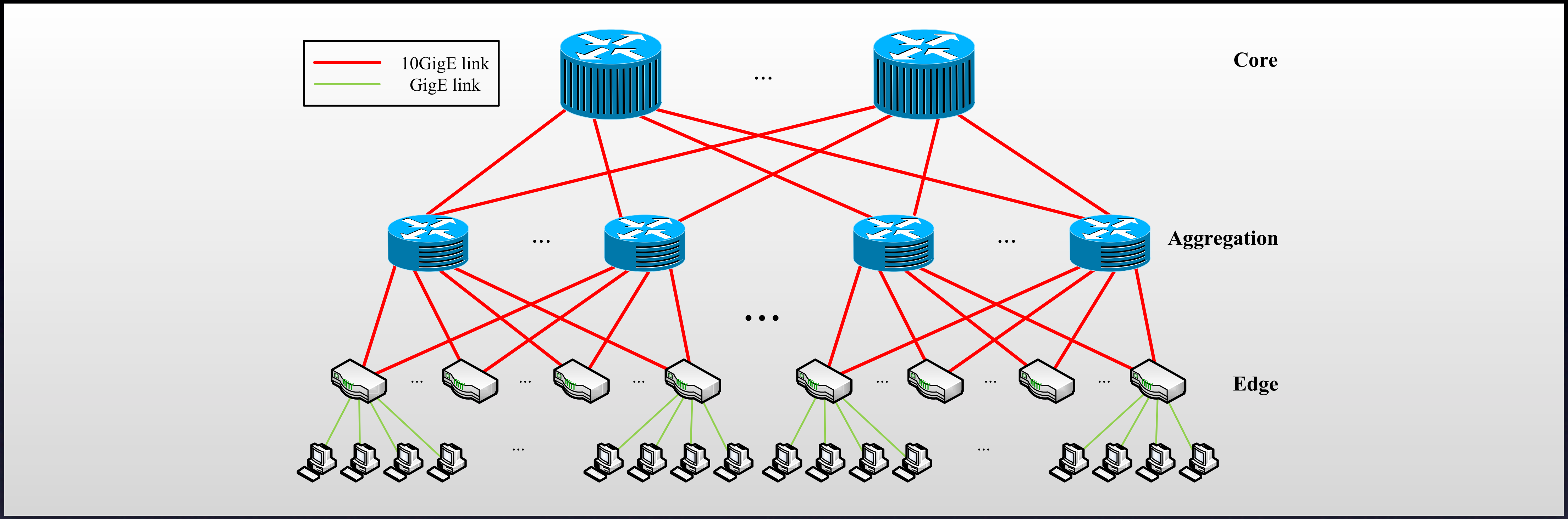
Hedera: Dynamic Flow Scheduling for Data Center Networks

Mohammad Al-Fares Sivasankar Radhakrishnan
Barath Raghavan[*] Nelson Huang Amin Vahdat

UC San Diego [*] Williams College

- USENIX NSDI 2010 -

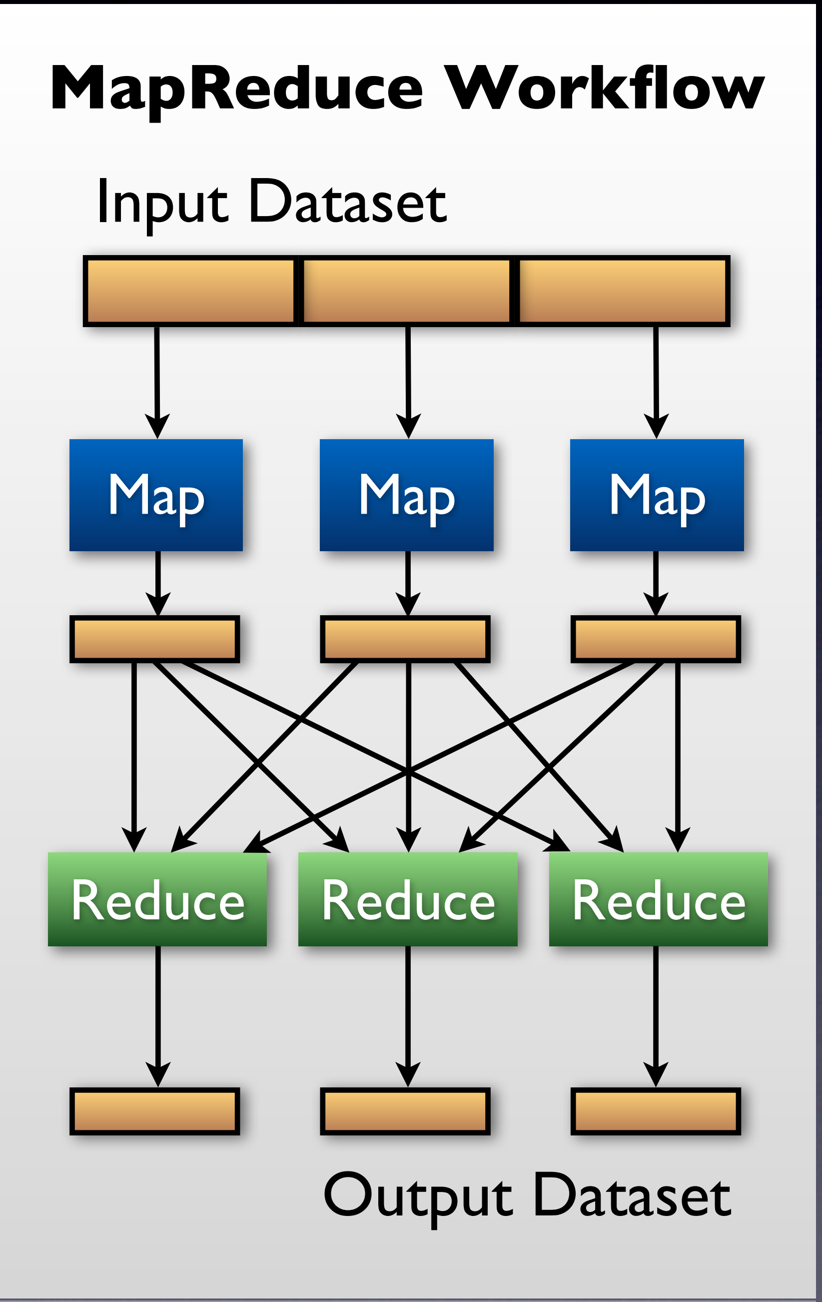
Motivation



- Current data center networks support tens of thousands of machines
  - Limited port-densities at the core routers ➔ Horizontal expansion = increasingly relying on multipathing

Motivation

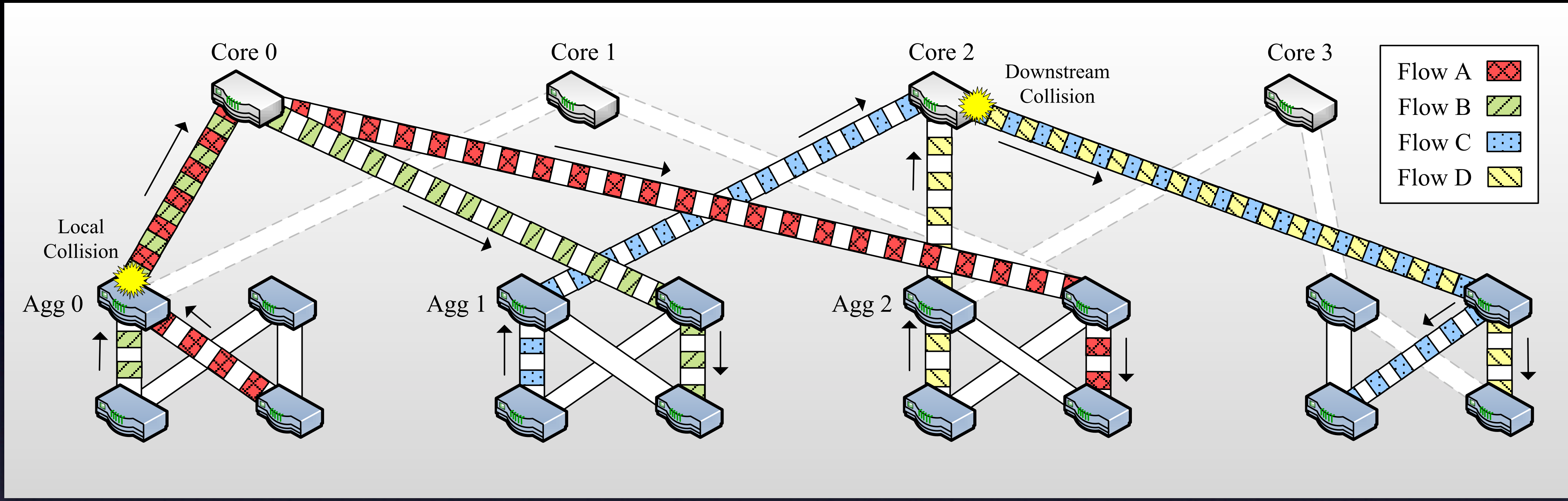
- MapReduce / Hadoop -style workloads have substantial BW requirements
  - Shuffle phase stresses network interconnect
  - Oversubscription / Bad forwarding → Jobs often bottlenecked by network



Contributions

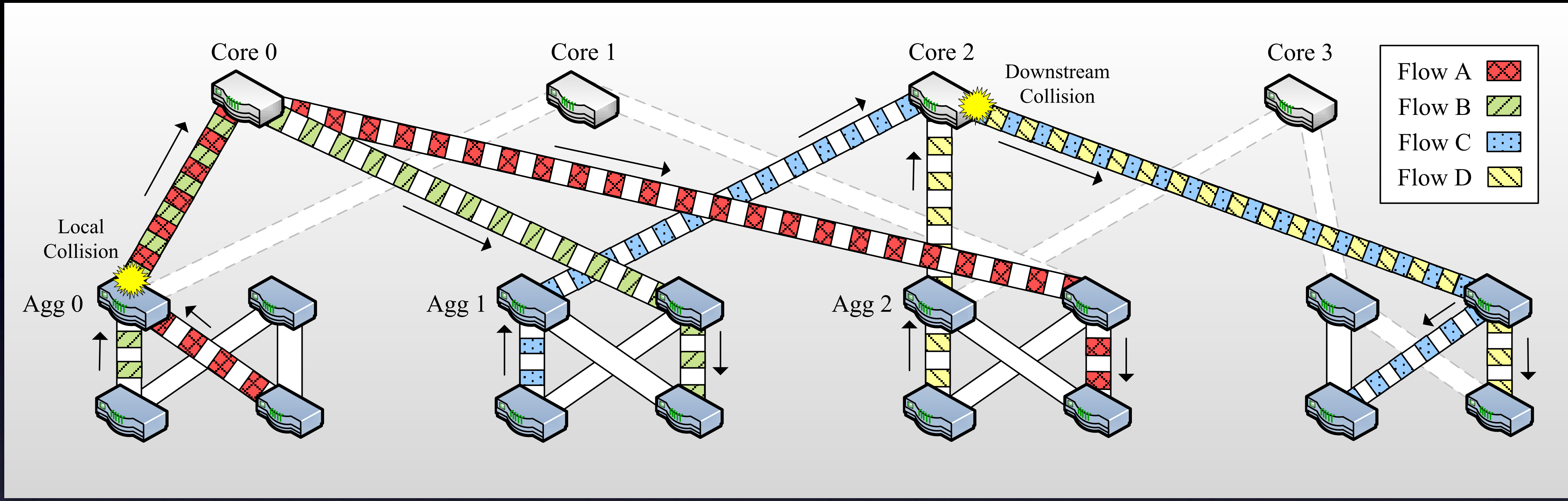
- Integration + working implementation of:
  1. Centralized Data Center routing control
  2. Flow Demand Estimation
  3. Efficient + Fast Scheduling Heuristics
- Enables more efficient utilization of network infrastructure
  - Upto 96% of optimal bisection bandwidth, > 2X better than standard techniques

Background



- Current industry standard: Equal-Cost Multi-Path (ECMP)
  - Given a packet to a subnet with multiple paths, forward packet based on a hash of packet's headers
  - Originally developed as a wide-area / backbone TE tool
  - Implemented in: Cisco / Juniper / HP ... etc.

Background



- ECMP drawback: Static + Oblivious to link-utilization!
  - Causes long-term local/downstream flow collisions
  - On 27K-host fat-tree and a randomized matrix, ECMP wastes <u>*average*</u> of 61% of bisection bandwidth!

Problem Statement

Problem:

Given a dynamic traffic matrix of flow demands, how do you find paths that maximize network bisection bandwidth?

Constraint:

Commodity Ethernet switches + No end-host mods

Problem Statement

1. Capacity Constraint	2. Flow Conservation	3. Demand Satisfaction
$\sum_{i=1}^{k} f_i(u,v) \leq c(u,v)$	$\sum_{w \in V} f_i(u,w) = 0 \quad (u \neq s_i, t_i)$ $\forall v,u : f_i(u,v) = -f_i(v,u)$	$\sum_{w \in V} f_i(s_i,w) = d_i$ $\sum_{w \in V} f_i(w,t_i) = d_i$

Multi-Commodity Flow problem:

- Single path forwarding (no flow splitting)
  - Expressed as Binary Integer Programming (BIP)
  - Combinatorial, NP-complete
- Exact solvers CPLEX/GLPK impractical for realistic networks

Problem Statement

1. Capacity Constraint	2. Flow Conservation	3. Demand Satisfaction
$\sum_{i=1}^{k} f_i(u,v) \leq c(u,v)$	$\sum_{w \in V} f_i(u,w) = 0 \quad (u \neq s_i, t_i)$	$\sum_{w \in V} f_i(s_i,w) = d_i$
	$\forall v,u : f_i(u,v) = -f_i(v,u)$	$\sum_{w \in V} f_i(w,t_i) = d_i$

- Polynomial-time algorithms known for 3-stage Clos Networks (based on bipartite edge-coloring)
  - None for 5-stage Clos (3-tier fat-trees)
- Need to target arbitrary/general DC topologies!

Architecture

- Hedera : Dynamic Flow Scheduling
    - Optimize achievable bisection bandwidth by assigning flows non-conflicting paths
    - Uses flow demand estimation + placement heuristics to find good flow-to-core mappings

Architecture

- Scheduler operates a tight control-loop:
  1. Detect large flows
  2. Estimate their bandwidth demands
  3. Compute good paths and insert flow entries into switches

Elephant Detection

Elephant Detection

- Scheduler continually polls edge switches for flow byte-counts
  - Flows exceeding B/s threshold are "large"
    - > %10 of hosts' link capacity in our implementation (i.e. > 100Mbps)
- What if only "small" flows ?
  - Default ECMP load-balancing efficient

Elephant Detection

- Hedera *complements* ECMP!
    - Default forwarding uses ECMP
    - Hedera schedules large flows that cause bisection bandwidth problems

Demand Estimation

Demand Estimation

Motivation:

- Empirical measurement of flow rates are not suitable / sufficient for flow scheduling
  - Current TCP flow-rates may be constrained to inefficient forwarding
- Need to find the flows' overall fair bandwidth allocation, to better inform placement algorithms

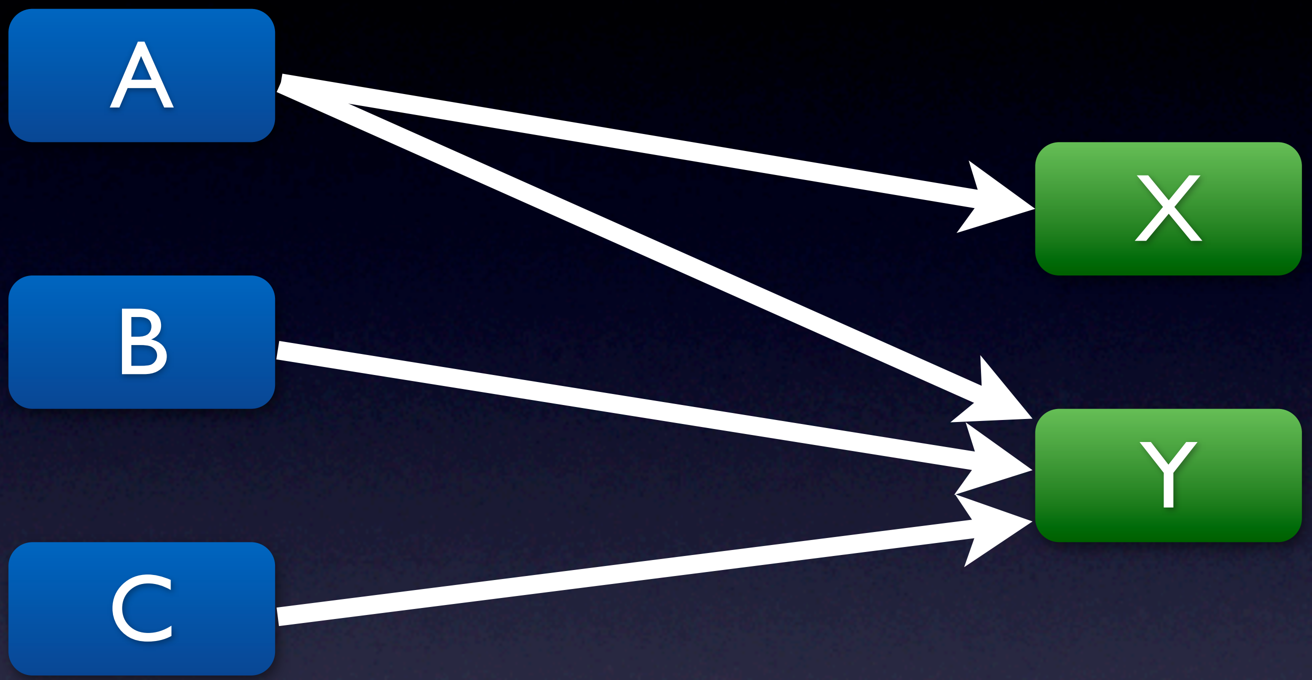
Demand Estimation

- TCP's AIMD + Fair Queueing try to achieve max-min fairness in steady state
  - When routing is a degree of freedom, establishing max-min fair demands is hard
  - Ideal case: find max-min fair bandwidth allocation as if constrained by host-NIC

Demand Estimation

- Given traffic matrix of large flows, modify each flow's size at **Src + Dst** iteratively:
  1. Sender equally distributes unconverged bandwidth among outgoing flows
  2. NIC-limited receivers decrease exceeded capacity equally between incoming flows
  3. Repeat until all flows converge
- Guaranteed to converge in O(|F|) time

Demand Estimation



Flow	Estimate	Conv. ?
A → X		
A → Y		
B → Y		
C → Y		

Senders

Sender	Available Unconv. BW	Flows	Share
A	1	2	1/2
B	1	1	1
C	1	1	1

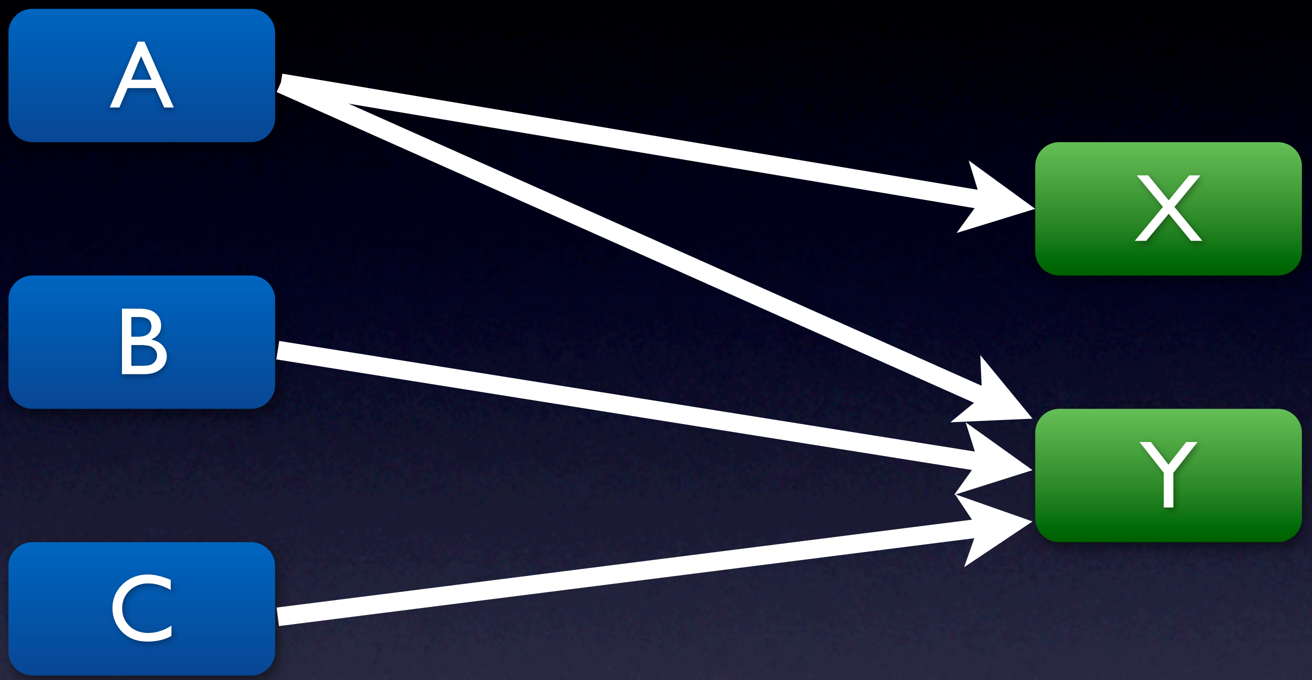
Demand Estimation

Flow	Estimate	Conv. ?
A → X	1/2	
A → Y	1/2	
B → Y	1	
C → Y	1	

Receivers

Recv	RL?	Non-SL Flows	Share
X	No	-	-
Y	Yes	3	1/3

Demand Estimation



Flow	Estimate	Conv. ?
A → X	1/2	
A → Y	1/3	Yes
B → Y	1/3	Yes
C → Y	1/3	Yes

Senders

Sender	Available Unconv. BW	Flows	Share
A	2/3	1	2/3
B	0	0	0
C	0	0	0

Demand Estimation

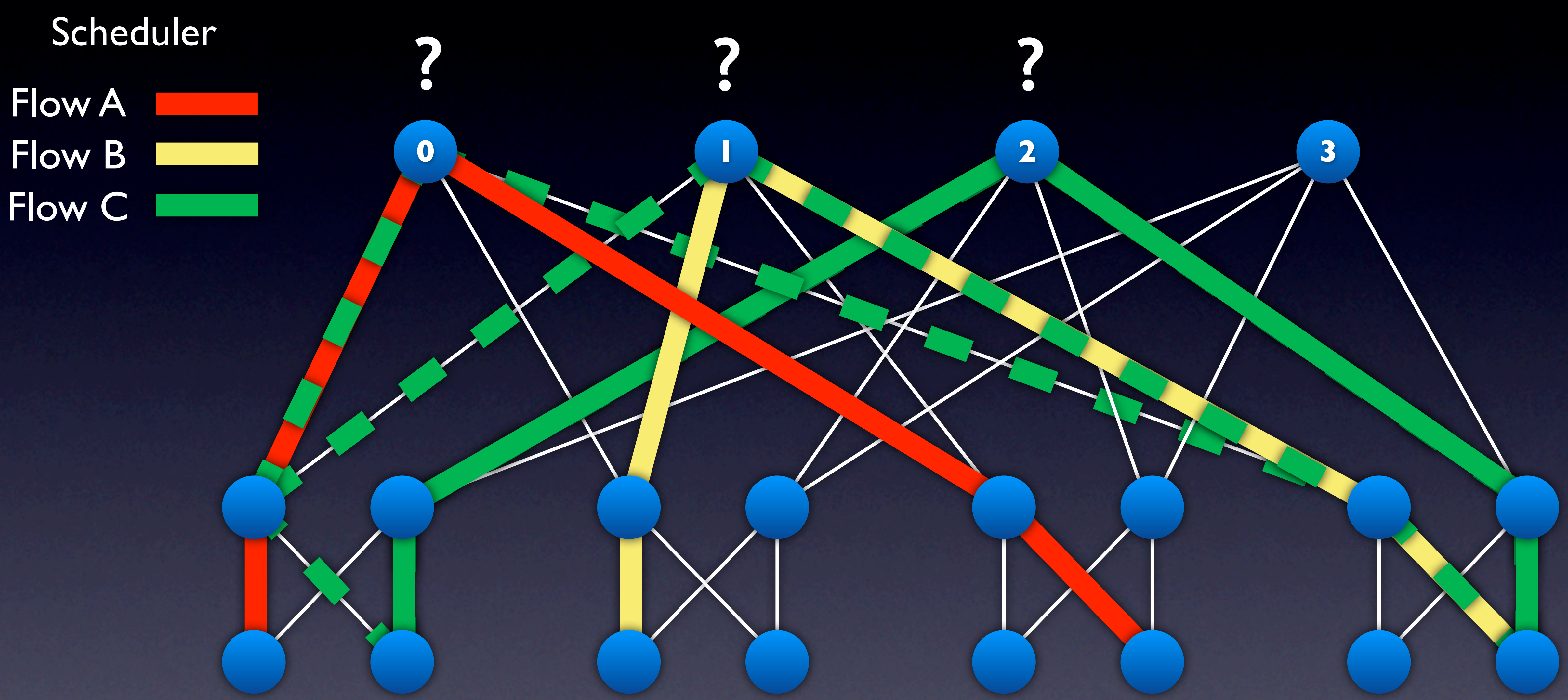
Receivers

Flow	Estimate	Conv. ?
A → X	2/3	Yes
A → Y	1/3	Yes
B → Y	1/3	Yes
C → Y	1/3	Yes

Recv	RL?	Non-SL Flows	Share
X	No	-	-
Y	No	-	-

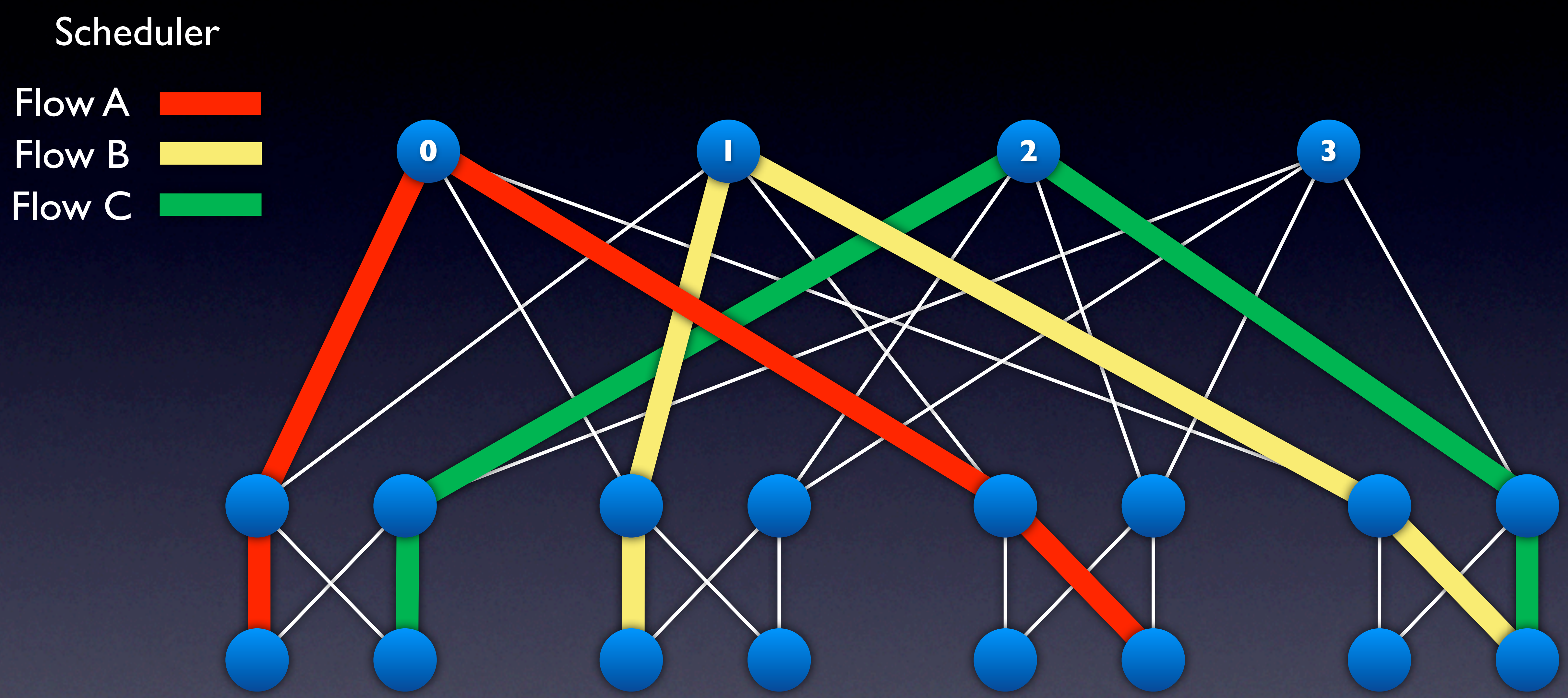
Placement Heuristics

Global First-Fit



- New flow detected, linearly search all possible paths from S➔D
- Place flow on first path whose component links can fit that flow

Global First-Fit



- Flows placed upon detection, are not moved
- Once flow ends, entries + reservations time out

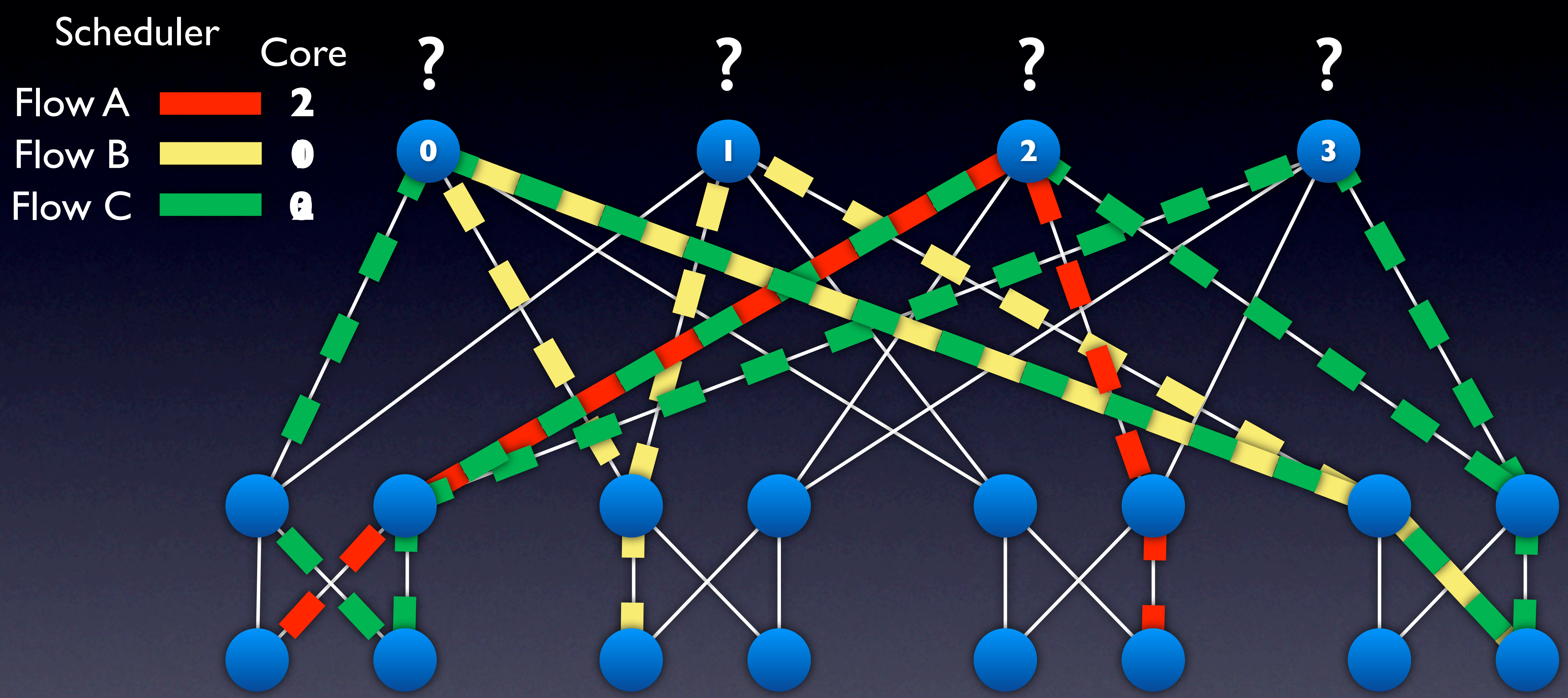
Simulated Annealing

- Probabilistic search for good flow-to-core mappings
  - Goal: Maximize achievable bisection bandwidth
- Current flow-to-core mapping generates neighbor state
  - Calculate total exceeded bandwidth capacity
  - Accept move to neighbor state if bisection BW gain
- Few thousand iterations for each scheduling round
  - Avoid local-minima; non-zero prob. to <u>worse</u> state

Simulated Annealing

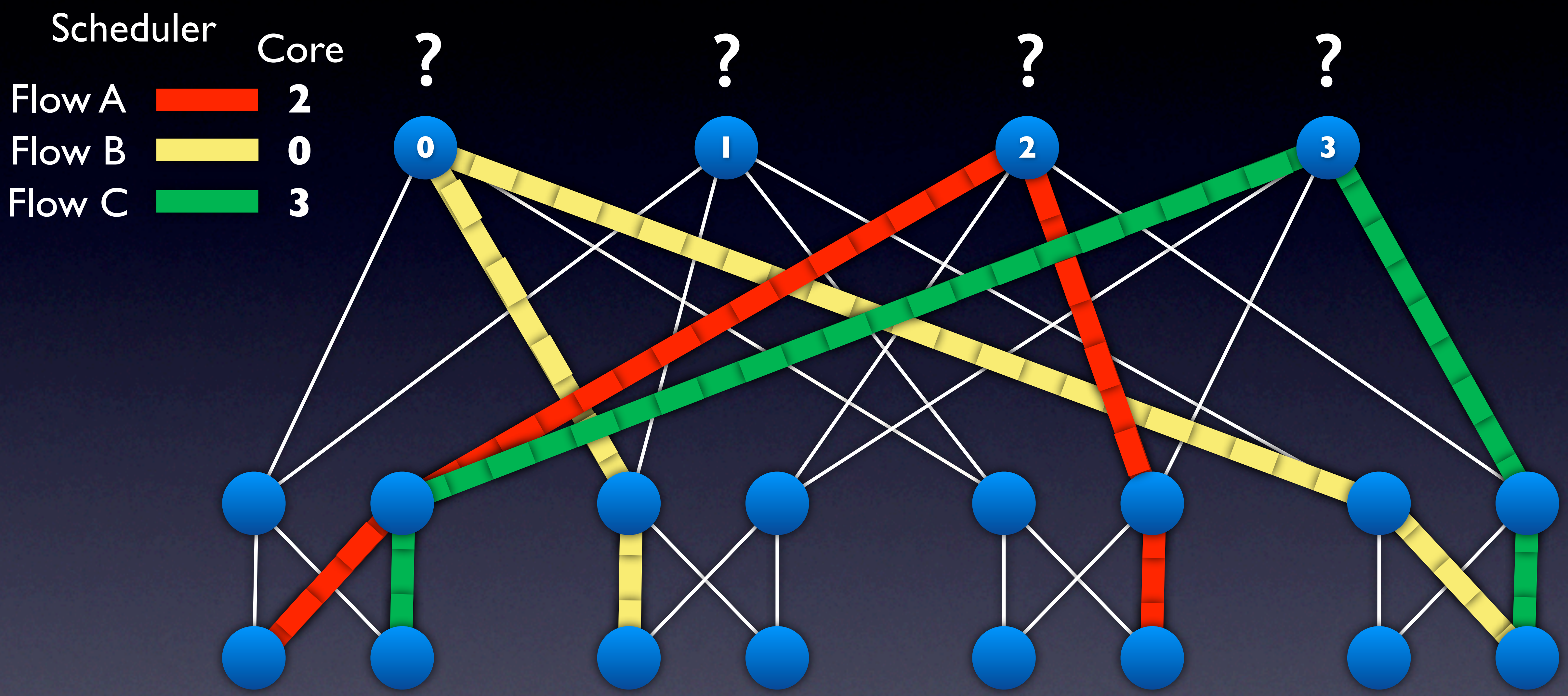
- Implemented several optimizations that reduce the search-space significantly:
  - Assign a single core switch to each destination host
  - Incremental calculation of exceeded capacity
  - .. among others

Simulated Annealing



- Example run: 3 flows, 3 iterations

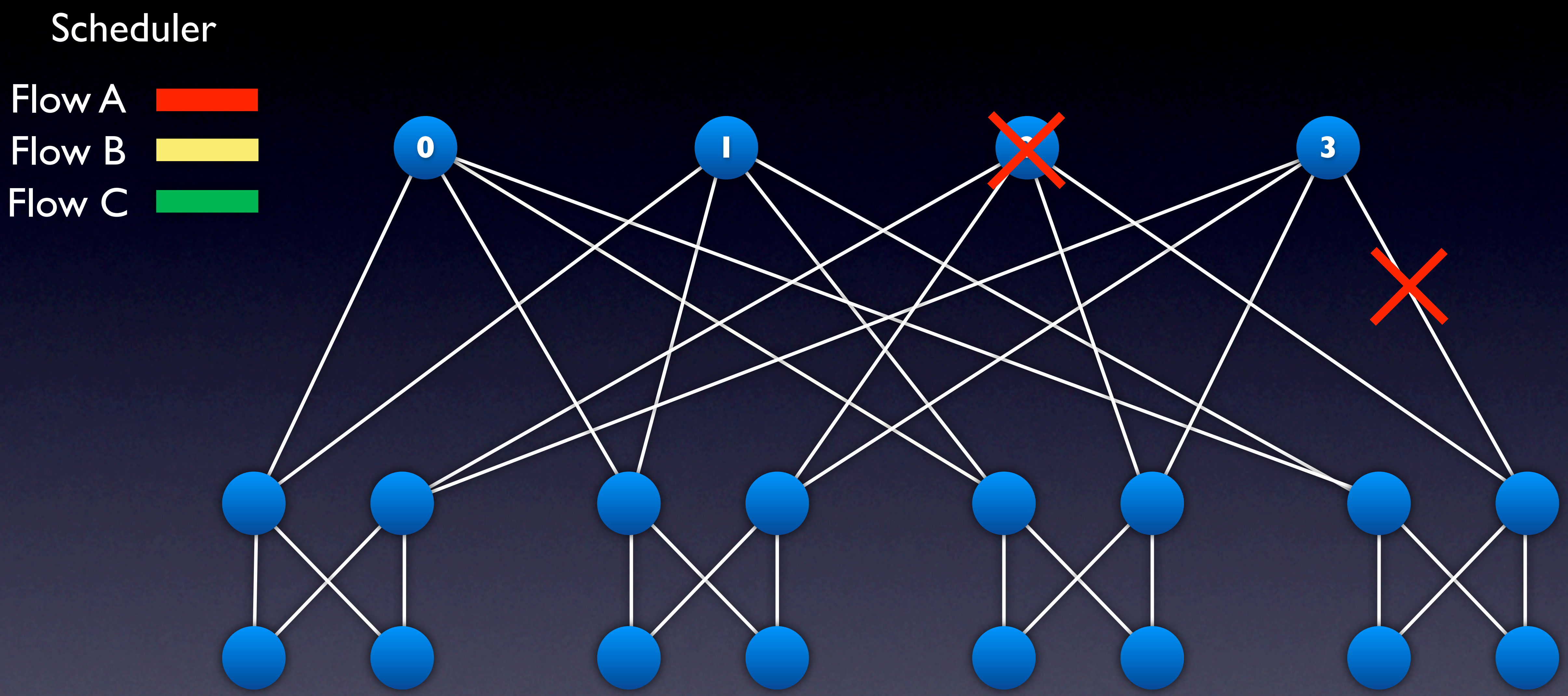
Simulated Annealing



- Final state is published to the switches and used as the initial state for next round

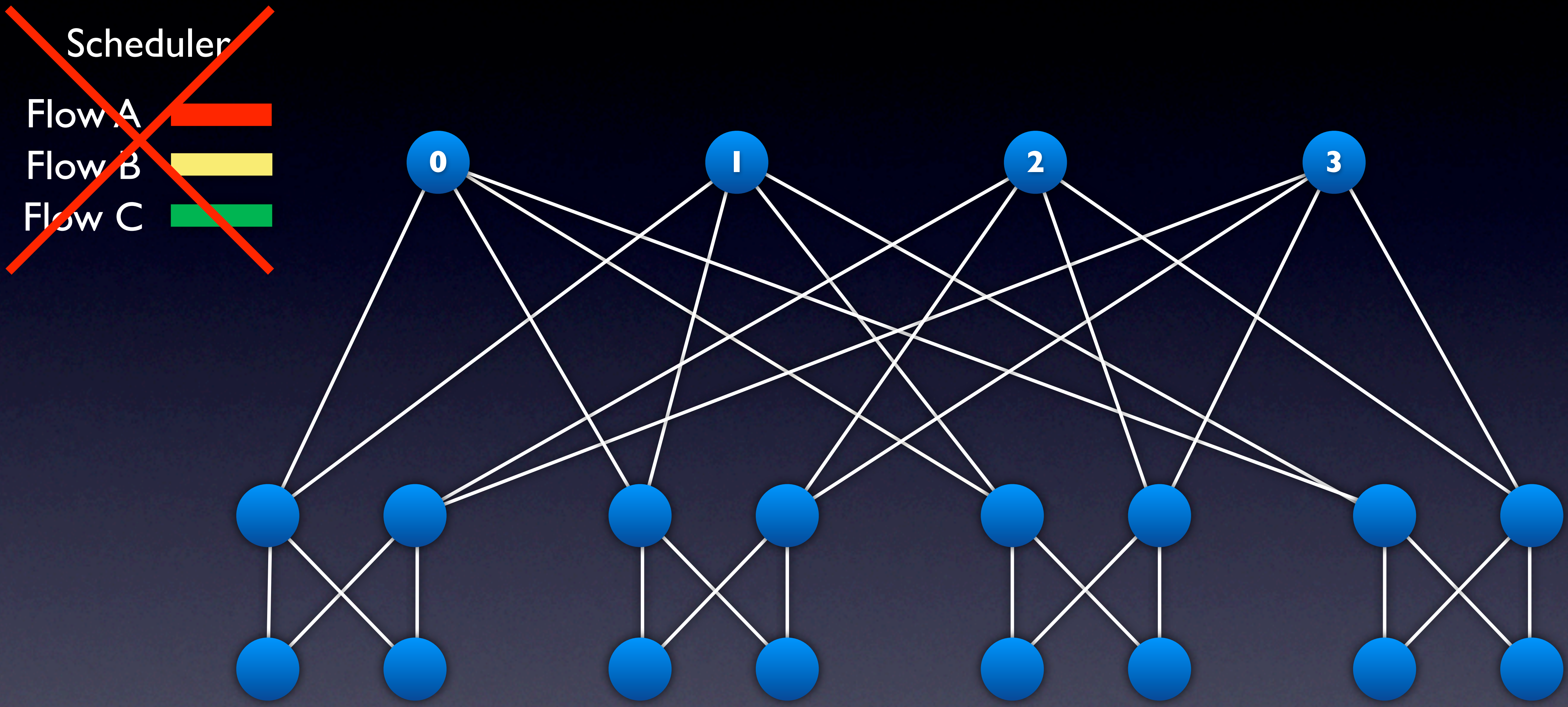
Fault-Tolerance

Fault-Tolerance



- Link / Switch failure: Use PortLand's fault notification protocol
  - Hedera routes around failed components

Fault-Tolerance



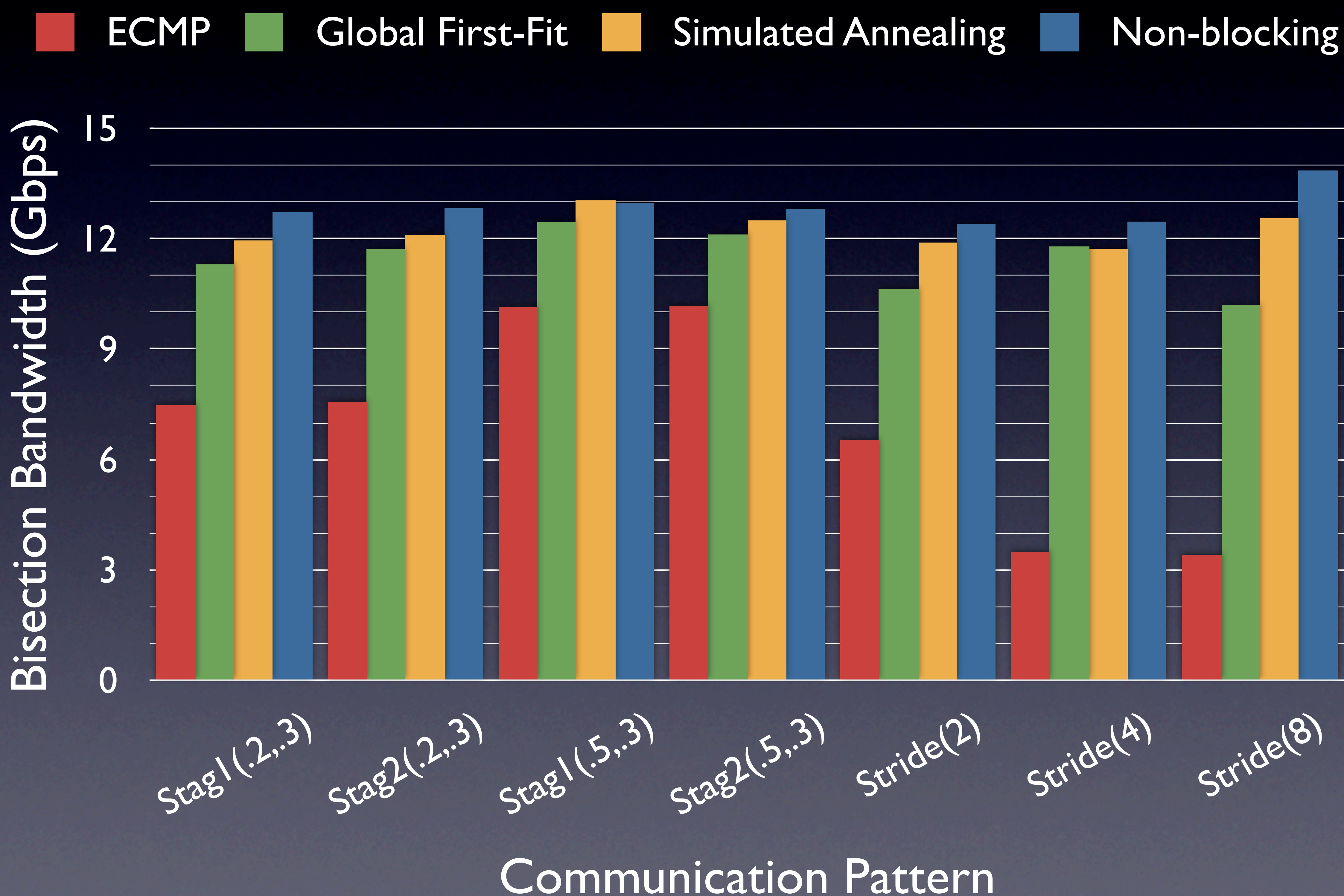
- Scheduler failure:
  - Soft-state, not required for correctness (connectivity)
  - Switches fall back to ECMP

Implementation

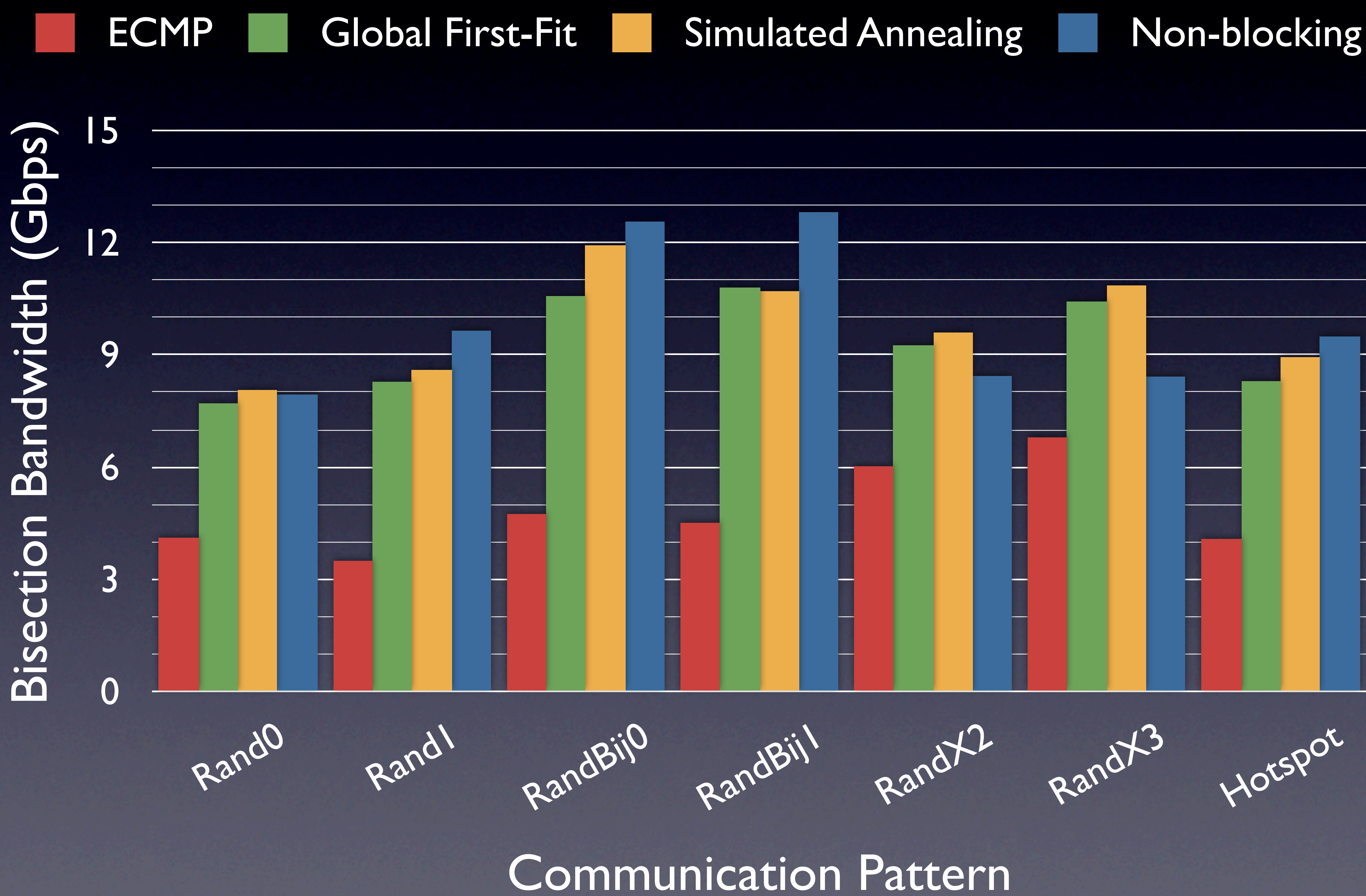
Implementation

- 16-host testbed
  - *k*=4 fat-tree data-plane
  - 20 machines; 4-port NetFGPAs / OpenFlow
  - Parallel 48-port non-blocking Quanta switch
- 1 Scheduler machine
  - Dynamic traffic monitoring
  - OpenFlow routing control

Evaluation - Testbed



Evaluation - Testbed



Data Shuffle

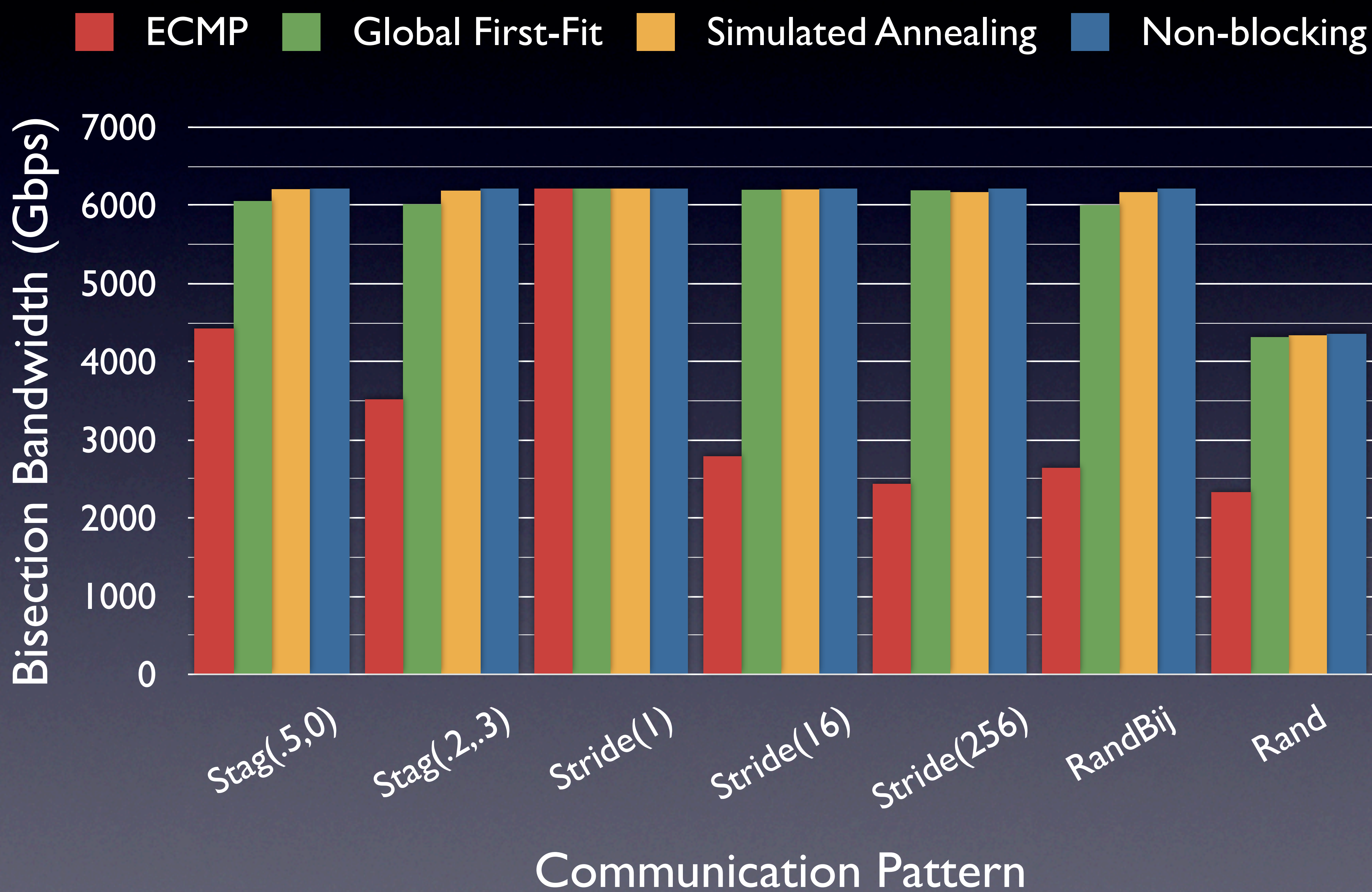
	ECMP	GFF	SA	Control
Total Shuffle Time (s)	438.4	335.5	336.0	306.4
Avg. Completion Time (s)	358.1	258.7	262.0	226.6
Avg. Bisection BW (Gbps)	2.81	3.89	3.84	4.44
Avg. host goodput (MB/s)	20.9	29.0	28.6	33.1

- 16-hosts: 120 GB all-to-all in-memory shuffle
  - Hedera achieves 39% better bisection BW over ECMP, 88% of ideal non-blocking switch

Evaluation - Simulator

- For larger topologies:
  - Models TCP's AIMD behavior when constrained by the topology
  - Stochastic flow arrival times / Bytes
  - Calibrated its performance against testbed
- What about *ns2* / *OMNeT++* ?
  - Packet-level simulators impractical at these network scales

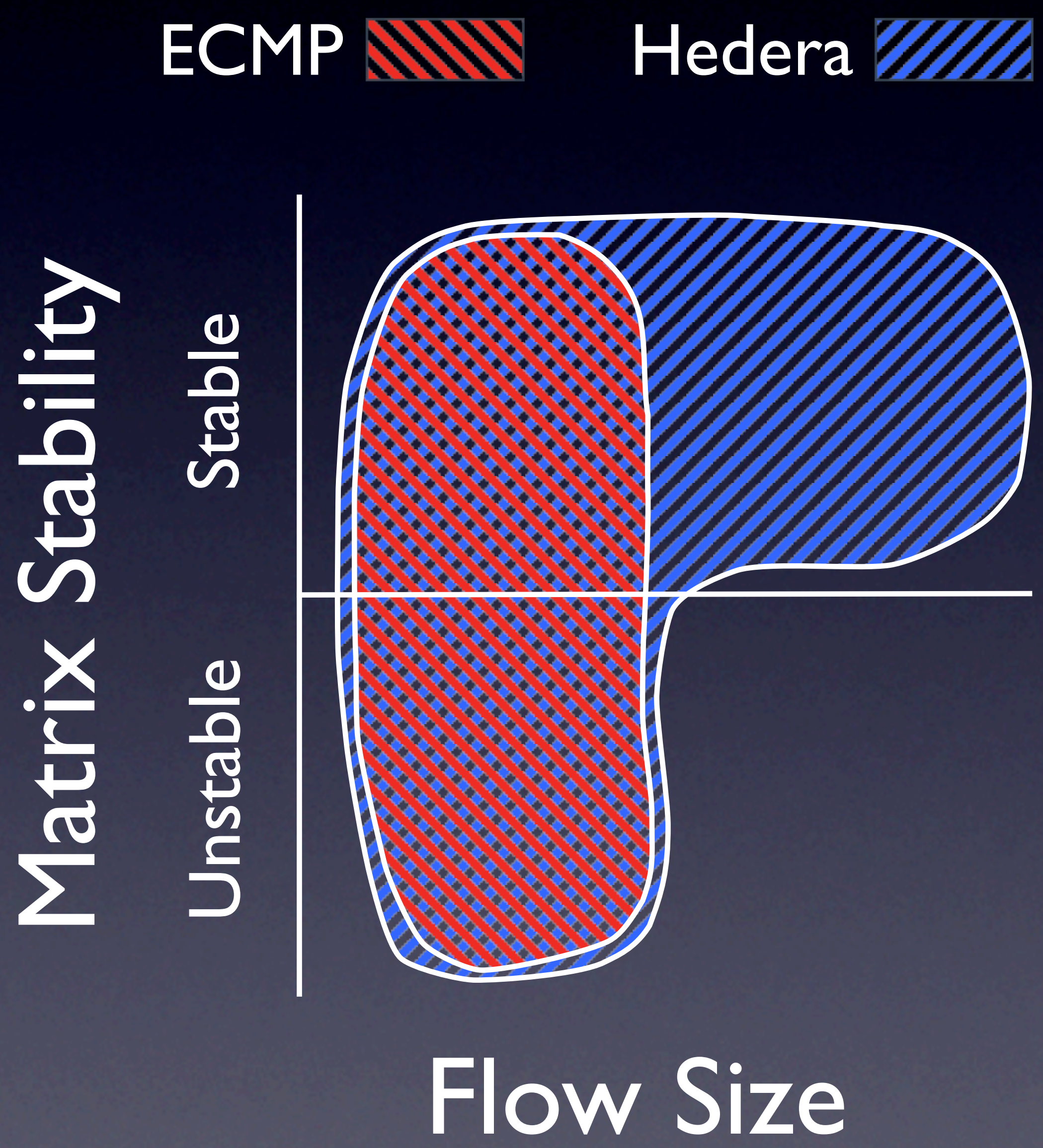
Simulator - 8,192 hosts (*k*=32)



Reactiveness

- Demand Estimation:
  - 27K hosts, 250K flows, converges < 200ms
- Simulated Annealing:
  - Asymptotically dependent on # of flows + # iter:
    - 50K flows and 10K iter: 11ms
  - Most of final bisection BW: first few hundred iter
- Scheduler control loop:
  - Polling + Estimation + SA = 145ms for 27K hosts

Limitations



- Dynamic workloads, large flow turnover faster than control loop
  - Scheduler will be continually chasing the traffic matrix
- Need to include penalty term for unnecessary SA flow re-assignments

Future Work

- Improve utility function of Simulated Annealing
  - SA movement penalties (TCP)
  - Add flow priorities (QoS)
  - Incorporate other metrics: e.g. Power
- Release combined system: PortLand + Hedera (6/1)
- Perfect, non-centralized, per-*packet* Valiant Load Balancing

Conclusions

- Simulated Annealing delivers significant bisection BW gains over standard ECMP
- Hedera *complements* ECMP
  - RPC-like traffic is fine with ECMP
- If you're running MapReduce/Hadoop jobs on your network, you stand to benefit greatly from Hedera; tiny investment!

Questions?

http://cseweb.ucsd.edu/~malfares/

Traffic Overhead

- 27K host network:
  - Polling: 72B / flow * 5 flows/host * 27K hosts / 0.1 sec = < 100MB/s for DC
  - Could also use data-plane