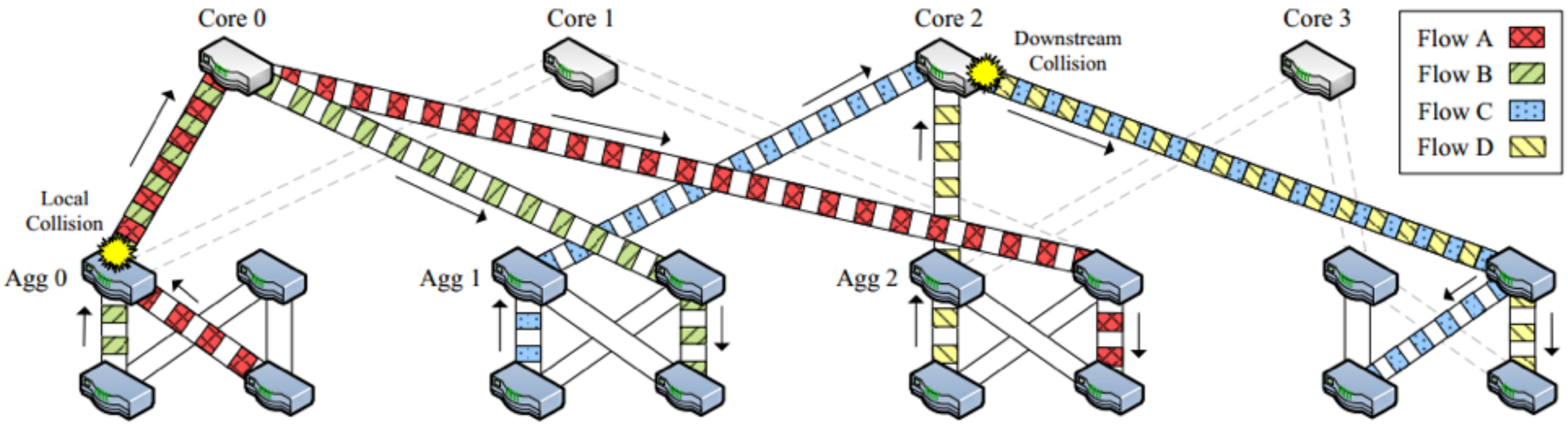
Final Project: Enhanced Multipath Switching for Data Centers

Under what circumstances ECMP fails?

- Does a great job on small flows but fails on large flows
- Static mapping of flows to paths does not account for either current network utilization or flow size


Core 0
Core 1
Core 2
Core 3
Downstream
Collision
Flow A
Flow B
Flow C
Flow D
Local
Collision
Agg 0
Agg 1
Agg 2

Stages

1. Elephant detection: detect large flows
2. Demand estimation
3. Schedule Flows

Elephant Detection

Idea: Poll edge switches for flow sizes.

1. Each new flow path calculated with basic ECMP.
2. Each flow has a counter that measures how many bytes send within the flow.
3. When counter's value is more than some threshold (10% of the capacity), the flow determined as a big one and forwarded to the controller.

Demand Estimation

Idea: Controller holds a matrix of all big flows: source and destination, and based on this matrix calculates bandwidth constrains on the flow.

1. When new flow added:
    1. Until convergence, modify flow sizes of each flow:
        1. Sender equally distributes free bandwidth among outgoing flows.
        2. Receiver (NIC) decrease exceeded capacity equally between incoming flows.
2. Existed flows aren't part of this calculation and their BW won't change.

Schedule Flows

Idea: in paper there are couple of implementation, the best one according the graphs is Simulated Annealing. However it's running time is bigger, so it's requires some optimisations.

Optimisations we'll use:

- Each destination has predefined core switch

Schedule Flows

1. init: s = current state, e = current bandwidth constrain
2. loop: until we reach destination
    1. loop for some number of interactions (max degree)
        1. choose neighbour of s, find free capacity of the link s -> neighbour of s*
        2. ns - best neighbour, ne - best bandwidth allowed on the link toward the neighbour.
    2. Install the flow on the switch (s -> ns)*
    3. s = ns, e = ne

Schedule Flows

Explanation:

1. Find free capacity of the link s -> neighbour of s*
    1. Controller contains the table of all free capacities for each link.
    2. Probably there is a command to check and update link's capacity in SDN - (we didn't look up for yet)
2. Install the flow on the switch: after the flow is idle for some predefined amount of time, it will be removed and capacity of the link will be updated.

Testing on Mininet

- Of course, we can emulate and test on common data center networks (like Fat-Tree or Clos-Network)
- We can make each host send to another random host large amount of data (say 500MB)
- We then measure the time which takes for all the stations send and receive all of the data.