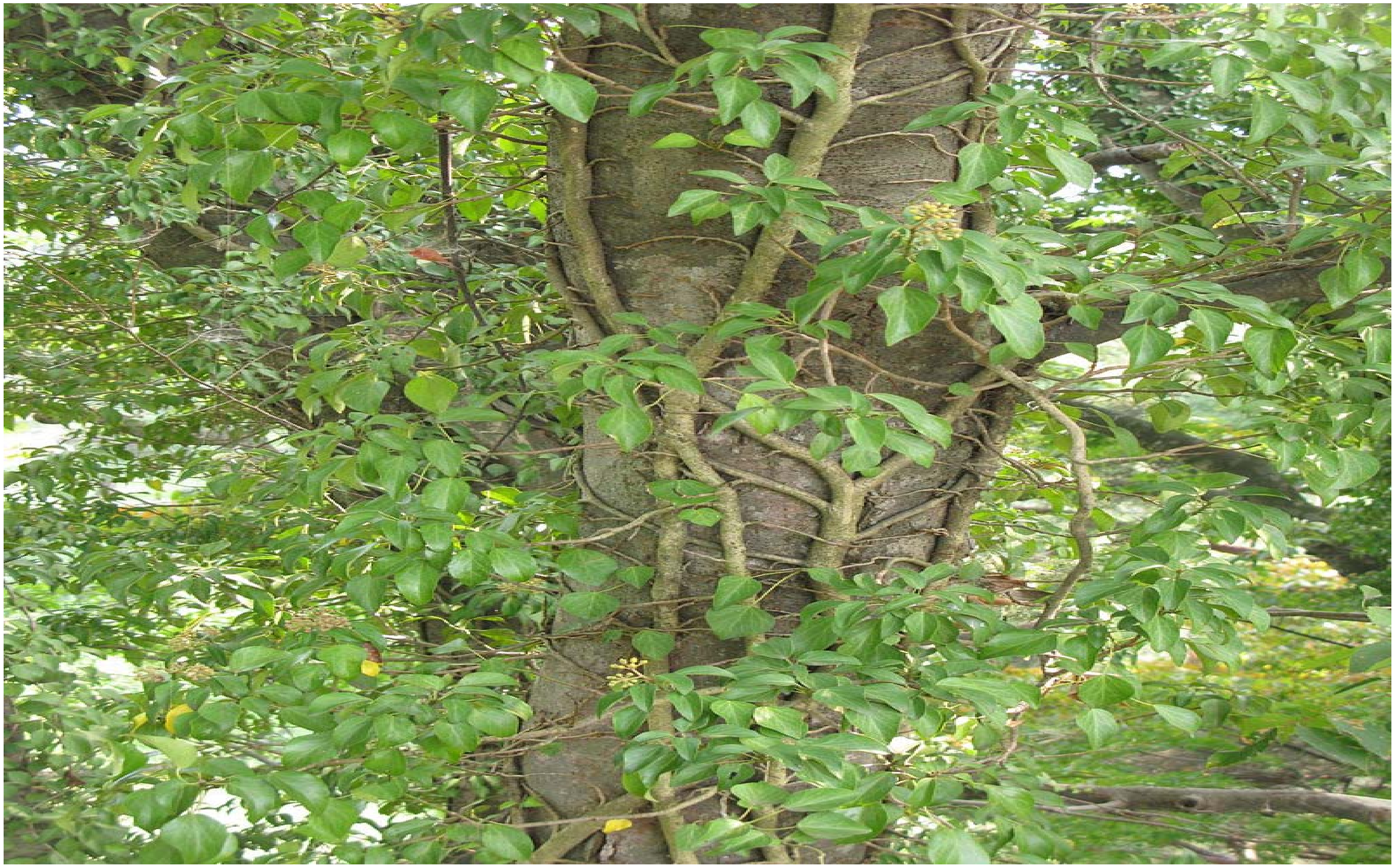
“Hedera” - An Analysis of Dynamic Flow Scheduling for Data Center Networks

Based on the paper “Hedera: Dynamic Flow Scheduling for Data Centers”
Presented by Richard Kramer – Oregon State University

What is Hedera?

What is Hedera? A multi-rooted tree! (actually a vine)!



Agenda

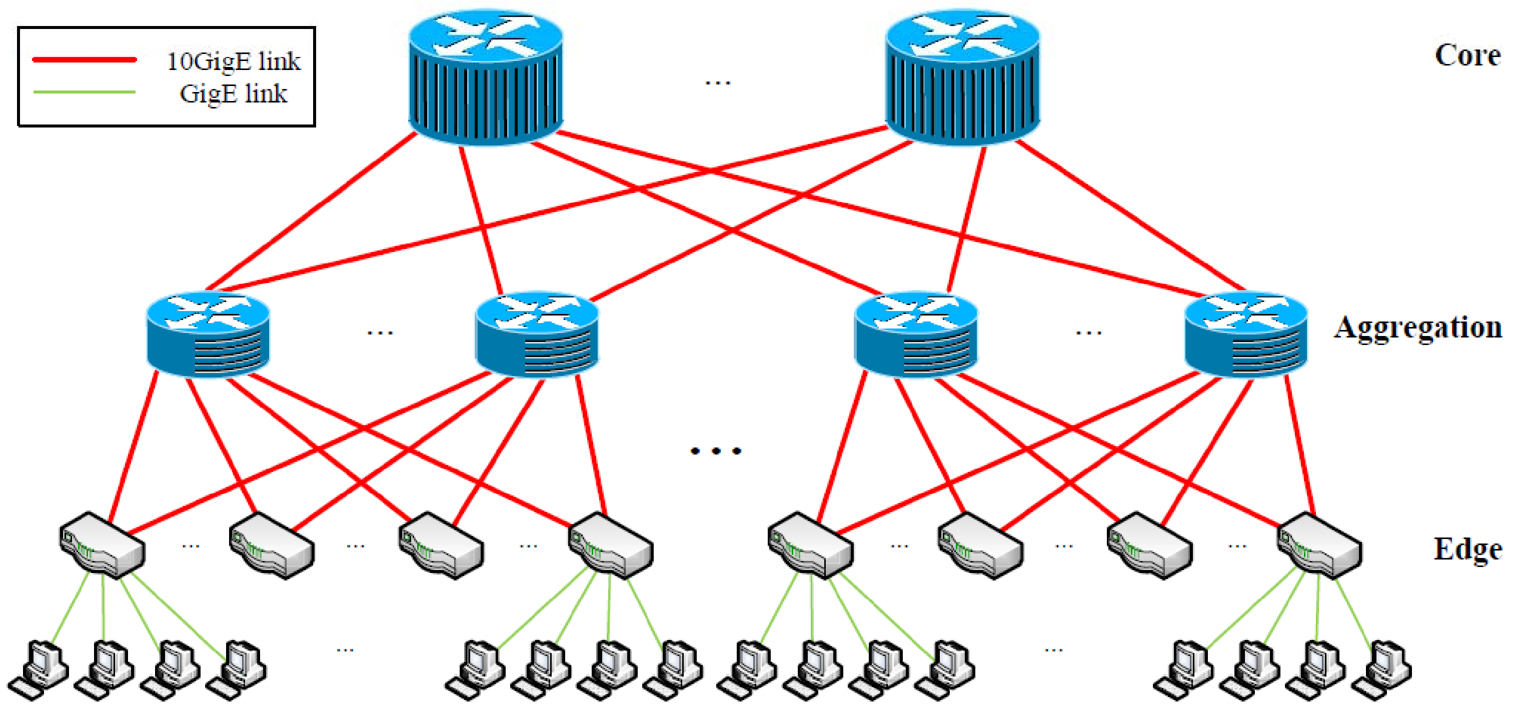
What is Hedera?

Besides being a vine, Hedera [1] is also …

- A scalable, dynamic flow scheduling system *for data centers*
- that adaptively schedules a multi-stage switching fabric
- to efficiently utilize aggregate network resources

[1] Mohammad Al-Fares, et al. Hedera: Dynamic Flow Scheduling for Data Center Networks. NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation, 2010.

Hedera is based on “common multi-rooted tree”



The Problems Hedera Seeks to Solve:

1. Data center designers have no way of knowing how data center network demand and workloads will vary over time,
   - Thus designers need a dynamic solution that can adapt over time.

2. The data center network system must operate using commercially available commodity system components
   - Without requiring protocols and/or software changes.

3. Inter-rack network bottlenecks make it difficult to ensure the virtualization instances will run on the same physical rack.

The Problems Hedera Seeks to Solve:

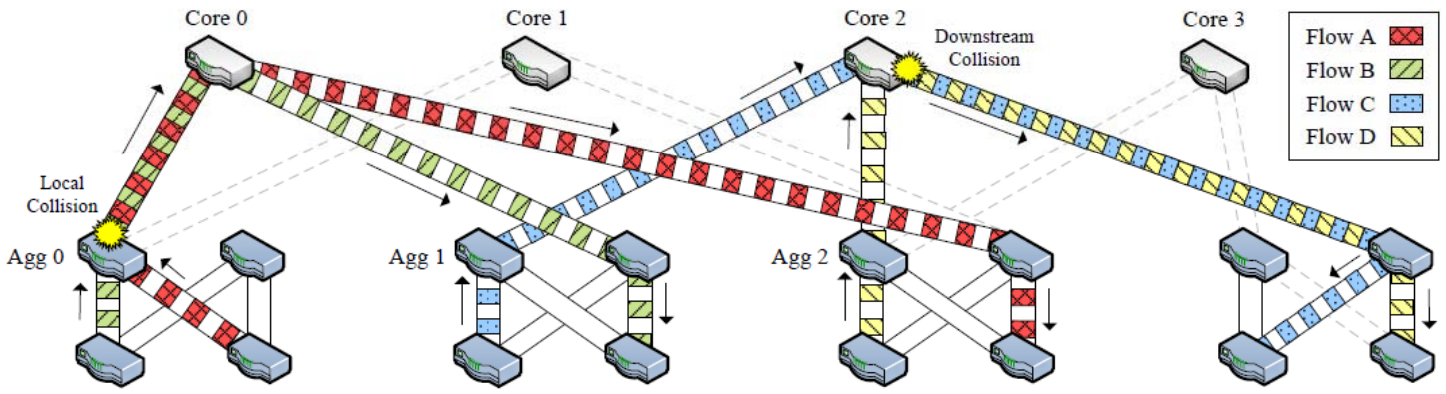
1. Data center designers have no way of knowing how data center network demand and workloads will vary over time,
   - Thus designers need a dynamic solution that can adapt over time.

2. The data center network system must operate using commercially available commodity system components
   - Without requiring protocols and/or software changes.

3. Inter-rack network bottlenecks make it difficult to ensure the virtualization instances will run on the same physical rack.

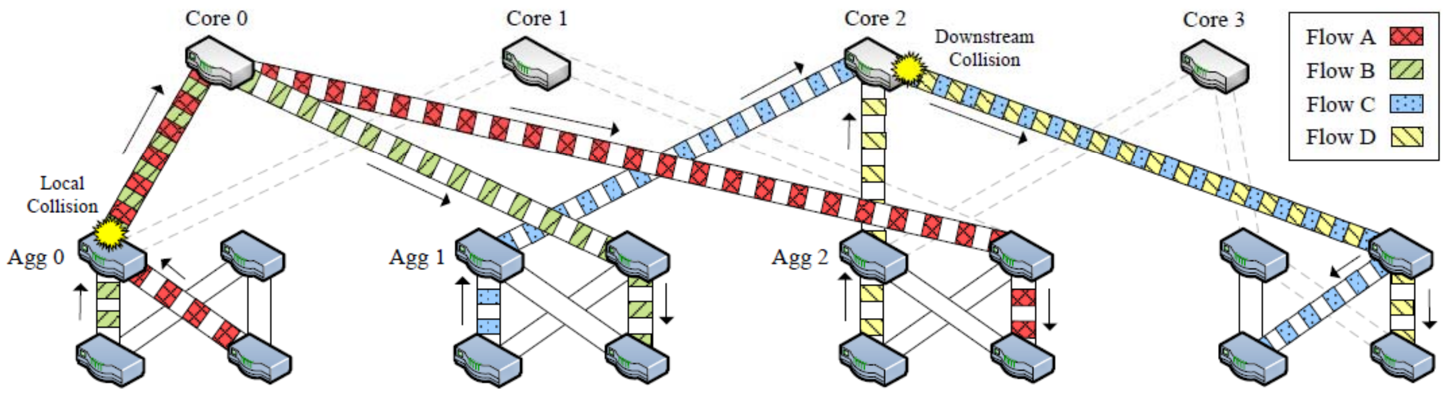
Hedera addresses the problems noted above by collecting flow information, dynamically computing non-conflicting paths for the data flows, and then programming commodity switches to reroute the traffic according to the newly computed non-conflicting paths.

The Problems Hedera Seeks to Solve:

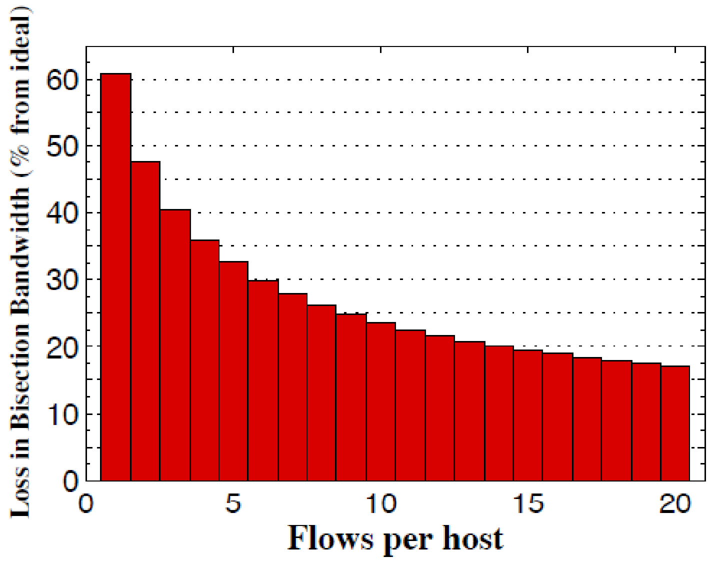


Examples of ECMP (Equal-Cost Multi-Path) Collisions.

The Problems Hedera Seeks to Solve:



Examples of ECMP (Equal-Cost Multi-Path) Collisions.



Example ECMP Bisection Bandwidth Loss

Hedera Proposes Two Alterative Algorithms as an Improvement over ECMP

- Hedera" proposes and evaluates the effectiveness of two different algorithms to provide dynamic flow scheduling improvements:
  - Global First Fit ("GFF").
  - Simulated Annealing ("SA").

an·neal: *verb*, **annealing**
heat (metal or glass) and allow it to cool slowly, in order to remove internal stresses and toughen it.

The proposed solutions are targeted for implementation on commodity switches and unmodified hosts (e.g. off-the-shelf components).

Hedera Proposes Two Alterative Algorithms, Continued

More specifically, Hedera performs the following tasks:

1. Detects large flows / Estimate the natural demand for large flows within the system.
2. Computes "good" non-conflicting paths for the large flows.
3. Installs the new computed "good" paths to accommodate the large flows within the switch fabric / instructs the switches to reroute.

Example of tracking demand reservations from T_0 to T_3:

$$\begin{bmatrix} & (\frac{1}{3})_1 & (\frac{1}{3})_1 & (\frac{1}{3})_1 \\ (\frac{1}{3})_2 & & (\frac{1}{3})_1 & 0_0 \\ (\frac{1}{2})_1 & 0_0 & & (\frac{1}{2})_1 \\ 0_0 & (\frac{1}{2})_2 & 0_0 & \end{bmatrix} \Rightarrow \begin{bmatrix} & [\frac{1}{3}]_1 & (\frac{1}{3})_1 & (\frac{1}{3})_1 \\ [\frac{1}{3}]_2 & & (\frac{1}{3})_1 & 0_0 \\ [\frac{1}{3}]_1 & 0_0 & & (\frac{1}{2})_1 \\ 0_0 & [\frac{1}{3}]_2 & 0_0 & \end{bmatrix} \Rightarrow \begin{bmatrix} & [\frac{1}{3}]_1 & (\frac{1}{3})_1 & (\frac{1}{3})_1 \\ [\frac{1}{3}]_2 & & (\frac{1}{3})_1 & 0_0 \\ [\frac{1}{3}]_1 & 0_0 & & (\frac{2}{3})_1 \\ 0_0 & [\frac{1}{3}]_2 & 0_0 & \end{bmatrix} \Rightarrow \begin{bmatrix} & [\frac{1}{3}]_1 & (\frac{1}{3})_1 & [\frac{1}{3}]_1 \\ [\frac{1}{3}]_2 & & (\frac{1}{3})_1 & 0_0 \\ [\frac{1}{3}]_1 & 0_0 & & [\frac{2}{3}]_1 \\ 0_0 & [\frac{1}{3}]_2 & 0_0 & \end{bmatrix}$$

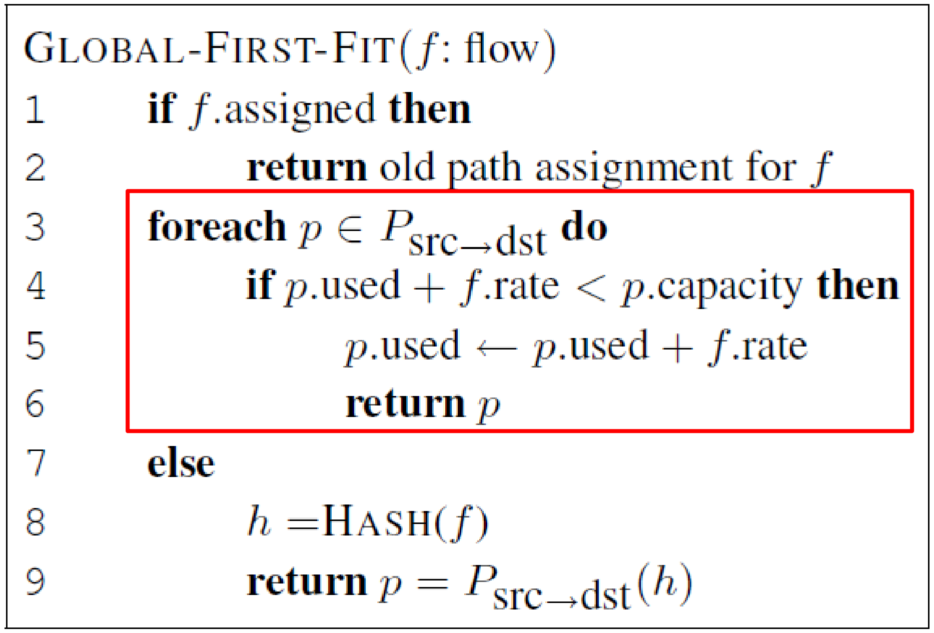
Hedera Proposes Two Alterative Algorithms, Continued

More specifically, Hedera performs the following tasks:

1. Detects large flows / Estimate the natural demand for large flows within the system.
2. Computes "good" non-conflicting paths for the large flows.
3. Installs the new computed "good" paths to accommodate the large flows within the switch fabric / instructs the switches to reroute.

Global First Fit ("GFF"):

- As the name indicates, GFF *globally* searches for the *first fitting* path that can accommodate the new flow and then reserves the capacity within the system to accommodate the new flow.

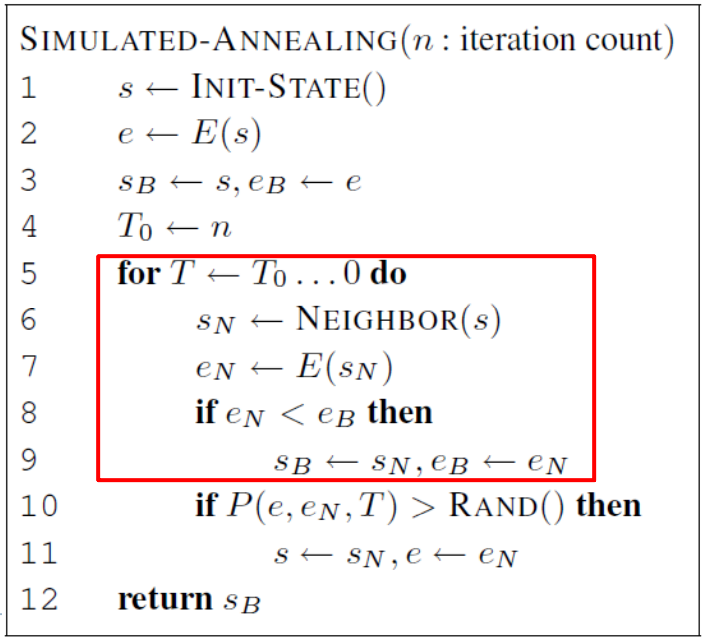


```
GLOBAL-FIRST-FIT($f$: flow)
    if $f$.assigned then
        return old path assignment for $f$
    foreach $p \in P_{\text{src}\rightarrow\text{dst}}$ do
        if $p$.used + $f$.rate < $p$.capacity then
            $p$.used $\leftarrow$ $p$.used + $f$.rate
            return $p$
    else
        $h$ = HASH($f$)
        return $p = P_{\text{src}\rightarrow\text{dst}}(h)$
```

- As a result, the system must maintain a record of the reserved capacity of every link within the network and release the reserved capacity when the flow expires.

Simulated Annealing ("SA"):

- The analogous initial annealing heating "energy" ("E") is equated *to the total exceeded network capacity over all links.*
- The analogous annealing decrementing / decreasing of "temperature" ("T") is equated *to the number of iterations that the SA algorithm "for loop" is executed.*



```
SIMULATED-ANNEALING($n$ : iteration count)
    $s \leftarrow$ INIT-STATE()
    $e \leftarrow E(s)$
    $s_B \leftarrow s, e_B \leftarrow e$
    $T_0 \leftarrow n$
    for $T \leftarrow T_0 \ldots 0$ do
        $s_N \leftarrow$ NEIGHBOR($s$)
        $e_N \leftarrow E(s_N)$
        if $e_N < e_B$ then
            $s_B \leftarrow s_N, e_B \leftarrow e_N$
        if $P(e, e_N, T) >$ RAND() then
            $s \leftarrow s_N, e \leftarrow e_N$
    return $s_B$
```

Simulated Annealing ("SA"):

- For each iteration of the SA "for loop" (e.g. decrease in temperature), the neighboring "state" ("s", mappings of destination hosts to core switches)
  - Available capacity is compared to the current selected state, seeking the lowest "energy".
  - When a lower neighboring energy value and state ("e_N" and "S_N") is found, the algorithm stores the better neighboring energy value and state as the "best" lowest energy and state ("e_B" and "s_B").
- Whereas for the next iteration and assignment of the state "s" to neighboring state "s_n" is further determined by a probabilistic based function "P" and a randomizer, seeking a "reasonable" best case.

GFF to SA Tradeoffs:

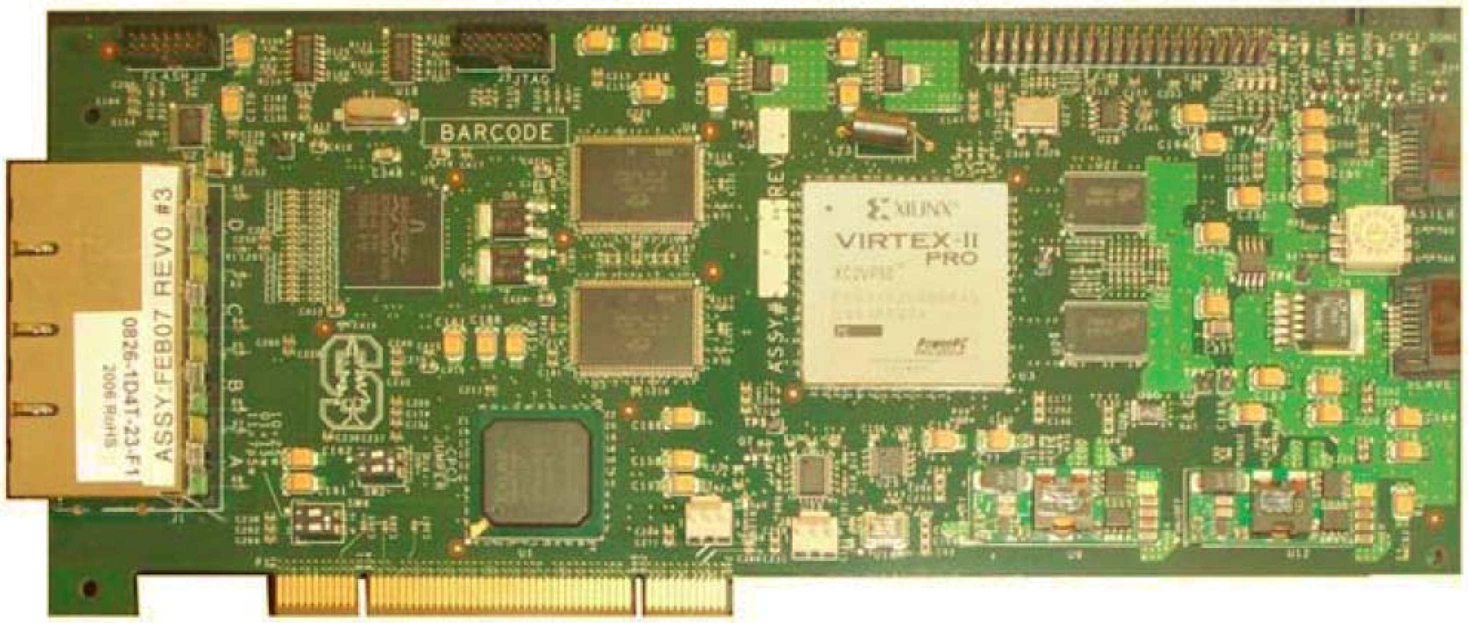
- **Processing Time:** With GFF, flows can be rerouted quicker when the following equation is true:
  - **Process_Time$_{(GFF)}$ [a function of $(k/2)^2$] < Process_Time$_{(SA)}$ [a function of f_{ave}]**

Where f_{ave} = average flows and k = the number of switch ports.

- **Overall Performance:** SA finds the <u>reasonably best</u> suited path versus GFF's first found path
  - *See Testbed and Simulation Results that follow*

Hedera was evaluated via a testbed using the NetFPGA programmable platform

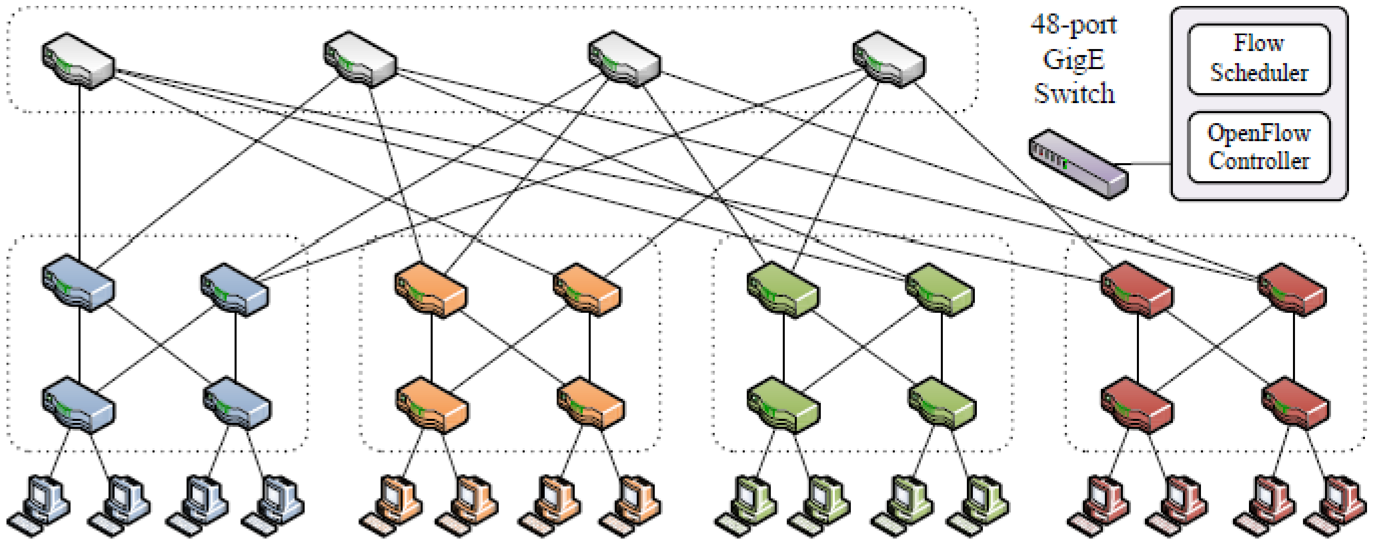
NetFPGA Technical Specifications



The NetFPGA is the low-cost reconfigurable hardware platform optimized for high-speed networking. The NetFPGA includes the all fo the logic resources, memory, and Gigabit Ethernet interfaces necessary to build a complete switch, router, and/or security device. Because the entire datapath is implemented in hardware, the system can support back-to-back packets at full Gigabit line rates and has a processing latency measured in only a few clock cycles.

Testbed Results

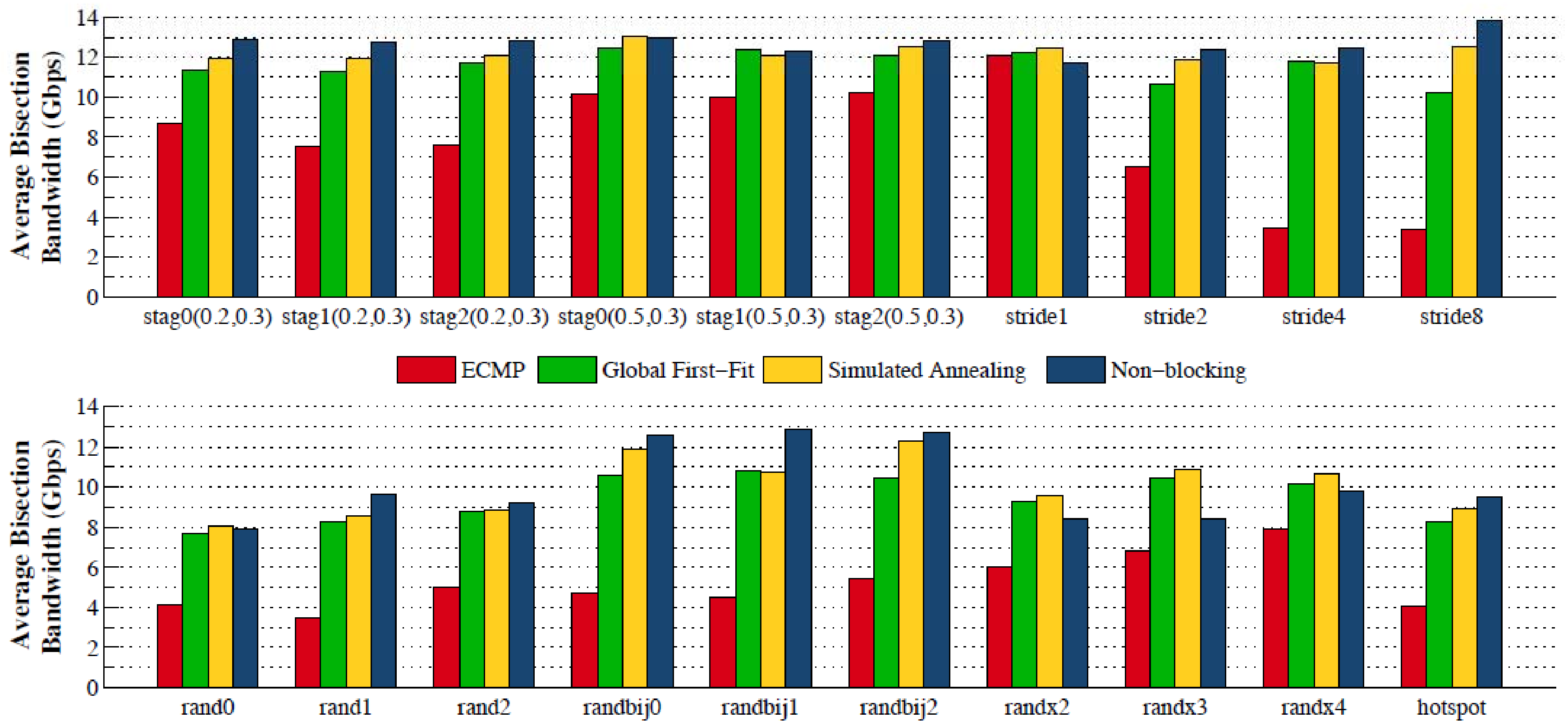
System Configuration:



OpenFlow: OpenFlow is an open standard that enables researchers to run experimental protocols in the campus networks. OpenFlow is added as a feature to commercial Ethernet switches, routers and wireless access points – and provides a standardized hook to allow researchers to run experiments, without requiring vendors to expose the internal workings of their network devices. OpenFlow is currently being implemented by major vendors, with OpenFlow-enabled switches now commercially available [5].

Testbed Results

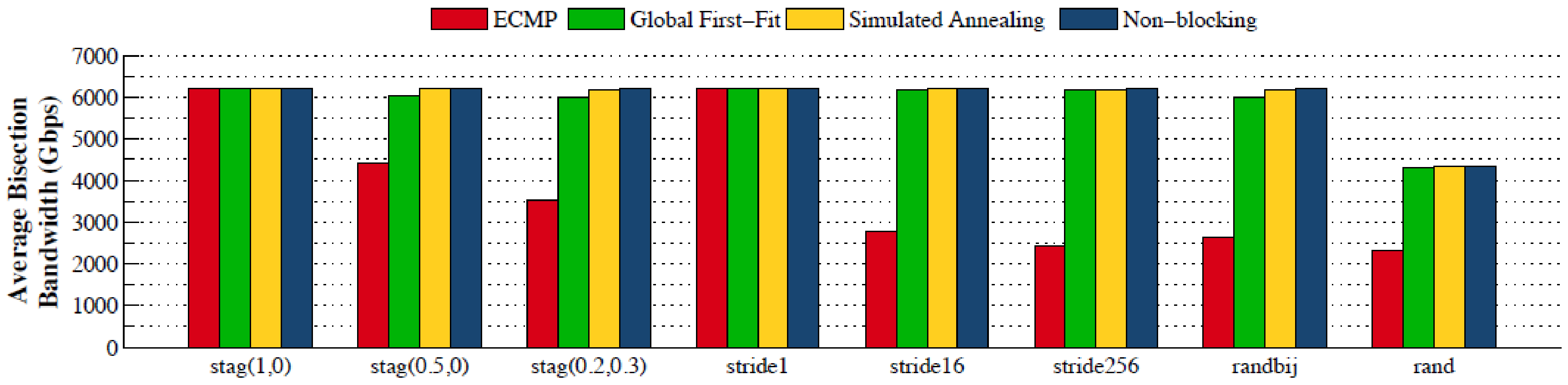
- A wide variety of traffic flows were tested.
- In all cases SA outperformed both ECMP and GFF



Simulation Results

Simulation results for 8,192 host data center:

- A wide variety of traffic flows were tested.
- 96% of optimal performance and
- 113% improvement over static load balancing methods such as ECMP static hashing [2].



Potential Improvements to Hedera

1. While the characterization of future applications is know, the characterization of present applications IS KNOWN.
   - One possible improvement to Hedera would be to "predict" the next T_n state loading, and assign a flow schedule/reservation table based on an optimal prediction.
2. Further, Hedera only mapped "large flows" using an arbitrary 10% threshold of a link's maximum capacity.
   - This invites optimization of SA and/or GFF based on the additional variable f(flow.threshold).
3. Lastly, Hedera did not seem to consider possible inter-system flows between servers.
   - Thus, optimization of inter-system flow dynamics exists.

Conclusion

- "Hedera" to be very insightful and compressive paper related to the subject of *dynamic flow scheduling for data center networks*.
- "Hedera's" analysis covers the spectrum of:
  1. Defining the problem,
  2. Identifying potential solutions,
  3. Then qualifying the potential solutions through simulation and on a testbed.
- In all, SA outperformed ECMP and GFF

Backup

Some of the Important Terms

- **AIMD:** Additive-Increase / Multiplicative-Decrease
- **ECMP:** Equal-Cost Multi-Path
- **Bijective:** Bijective function or one-to-one correspondence is a function between the elements of two sets, where every element of one set is paired with exactly one element of the other set, and every element of the other set is paired with exactly one element of the first set.
- **MapReduce / HADOOP:** MapReduce is a programming model and an associated implementation for processing and generating large data sets [6].
- **NetFPGA:** The NetFPGA is the low-cost reconfigurable hardware platform optimized for high-speed networking. The NetFPGA includes all logic resources, memory, and Gigabit Ethernet interfaces necessary to build a complete switch, router, and/or security device. Because the entire data path is implemented in hardware, the system can support back-to-back packets at full Gigabit line rates and has a processing latency measured in only a few clock cycles [4].
- **New Reno:** A TCP/IP congestion control and avoidance mechanism. New Reno improves upon TCP Reno (*see* "TCP Reno" below) by adding the ability to detect multiple packet losses and thus it is much more efficient in the event of multiple packet losses. [7]
- **OpenFlow:** OpenFlow is an open standard that enables researchers to run experimental protocols in the campus networks. OpenFlow is added as a feature to commercial Ethernet switches, routers and wireless access points – and provides a standardized hook to allow researchers to run experiments, without requiring vendors to expose the internal workings of their network devices. OpenFlow is currently being implemented by major vendors, with OpenFlow-enabled switches now commercially available [5].
- **RTT:** Round Trip Time
- **Static Hashing (ECMP):** A scheme of hashing the IP destination modulo the outgoing links "N" expresses as: H (Destination IP Address) = Destination IP Address mod N [2].
- **TCP Reno:** A TCP/IP congestion control and avoidance mechanism that uses the basic principle of slow starts and a coarse grain re-transmit time and adds additional intelligence so that lost packets are detected early and that the pipeline is not emptied every time a packet is lost [8][9].