Paper Presentation on "HULA: Scalable Load Balancing using Programmable Data-Planes"

Naga Ganesh Kurapati

EECE.7290 Selected Topics on Software Defined Networking

University of Massachusetts Lowell

Instructor: Prof.Yan Luo

March 20, 2017

Overview

1. Background
   - Traditionally, How routing is done?
   - Software-Defined Networking, What it is promising?
   - Understanding the parameters
   - Challenges faced by SDN
2. Efficiency - Better Load Balancing
   - Equal-cost multi-path routing (ECMP)
   - SDN, Central controller to decide traffic flow
   - CONGA: Congestion-Aware Load Balancing (Cisco)
   - Introduction to HULA
   - Design Challenges for HULA
   - HULAOverview: Scalable, Proactive, Adaptive, and Programmable
   - HULA Design: Probes and Flowlets
   - Programming HULA in P4
   - Evaluation of HULA

Traditionally, How routing is done?

- Distributed protocols like OSPF, RIP etc. are used to discover routes
- They run periodically to gather the information about link states and then processed to decide the route
- When one of link fails, this protocols can update the information with in seconds. And new route is calculated.
- Over the few decades it is widely accepted by different vendors
- Network operators have no access to modify the behavior to do things like traffic engineering

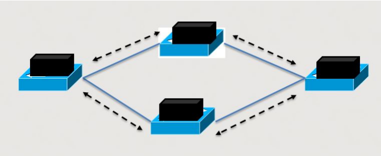


Figure: Traditional Network

RELIABLE, MOSTLY ADOPTED. BUT NOT FLEXIBLE.

Software-Defined Networking, What it is promising?

Decoupled Data plane and control plane.

1. Centralized view of the network
2. Simpler data plane(header matching). Functionality of the switch is abstracted
3. Unified control interface like OpenFlow

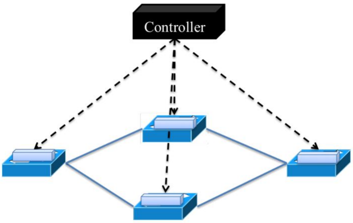


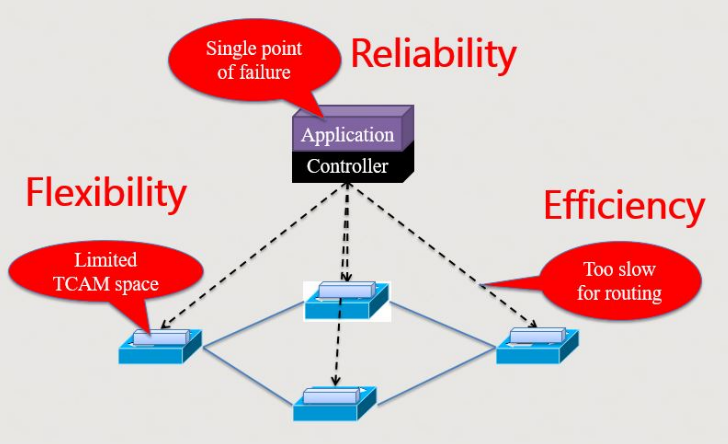
Figure: Software-Defined Networking

RELIABLE, FLEXIBLE AND EFFICIENT!!! HOW?

Understanding the parameters

1. Efficiency: Multiple paths for connecting network end points(Useful for diverting traffic). Avoid congestion when operating at high utilization.Efficient use of Underlying network bandwidth.Better Load Balancing
2. Flexibility: Flexible enough to accommodate different control plane rules. Sometimes larger set of rules and also lookup should be fast enough
3. Reliability: What if single central controller fails? Fault Tolerance

Challenges faced by SDN



Efficiency - Better Load Balancing

Traditionally, Equal-cost multi-path routing (ECMP)

– Data centers networks uses multi-rooted topologies like Leaf-Spine, Fat-Tree etc. to provide large bisection bandwidth
– Spreads traffic by randomly assigning each flow to one of several paths

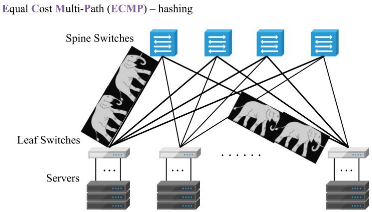


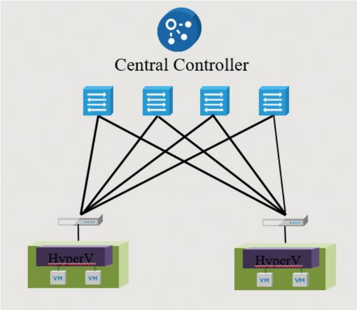
Figure: ECMP

- Degraded performance when two long-running flows are assigned to same path
- Not reactive to Link failures.Network is underutilized.Congested in asymmetric topologies

Efficiency - Better Load Balancing

SDN, Central controller to decide traffic flow

– Contoller has global visibility of congestion. Run traffic enginnering algorithms and push the commands to switches to manage traffic
– Hedera , SWAN(Microsoft) , and B4(Google)



- Control plane timescales are too slow to implement load balancing to efficiently use the available network capacity
- Takes minutes to react for changing network conditions. Slow down the volatile traffic

Efficiency - Better Load Balancing

CONGA: Congestion-Aware Load Balancing (Cisco)

– Data-plane load-balancing technique
– Load balancing decisions every few microseconds
– Determine the entire path. Maintains forwarding state for a large number of tunnels

Figure: CONGA

- Costly, use custom silicon chip.CONGA algorithm cannot be modified, once fabricated
- Maintains congestion state for every path at the leaf switches. Limited to topologies with a small number of paths like two-tier Leaf-Spine

Efficiency - Introduction to HULA

HULA - Hop-by-hop Utilization-aware Load balancing Architecture

1. Topology oblivious and Scalable: More Scalable than CONGA. Only picks next hop of globally best path to a destination.
2. Programmable data planes using P4: HULA algorithm can modified and inspected as for the needs of network operator.
3. Congestion-aware switches: Sends special probes periodically to gather link state information. Accumulate a table to determine next best hop towards any destination. Similar to distance vector protocol.
4. Fine-grained load balancing: Break down long-running flows into *flow-lets* to choose different paths to control congestion

Efficiency - HULA Design Challenges

1. Large path utilization matrix: Load balancing at Top of Rack switches (ToRs). Fat-Tree topology with radix k, then it needs to track k^2 paths for each destination ToR. For m leafs then $m * k^2$
   More memory.Expensive!!

Topology	# Paths between pair of ToRs	# Max forwarding entries per switch
Fat-Tree (8)	16	944
Fat-Tree (16)	64	15,808
Fat-Tree (32)	256	257,792
Fat-Tree (64)	1024	4,160,512

2. Large forwarding state: Addition to above matrix, need to store large forwarding tables in each switch to support a leaf-to-leaf tunnel for each path that it needs to route packets over.

Efficiency - HULA Design Challenges

3. Discovering uncongested paths: At high network utilization, more number of paths exits. It takes time to discover an uncongested path
4. Programmability: Tedious process for significant design and verification data-planeload-balancing schemes on hardware of switch. For modification, operator has to wait for the next product cycle. Solution for programmability:P4

Efficiency - HULAOverview: Scalable, Proactive, Adaptive, and Programmable

1. Maintaining compact path utilization: Instead of maintaining path utilization for all paths to a destination ToR, a HULA switch only maintains a table that maps the destination ToR to the best next hop as measured by path utilization. Effectively removing the pressure of path explosion on switch memory.
2. Scalable and adaptive routing: HULAs best hop table eliminates the need for separate source routing in order to exploit multiple network paths.Each switch independently chooses the best next hop to the destination
3. Automatic discovery of failures: If a switch does not receive a probe from a neighboring switch for more than a certain threshold of time

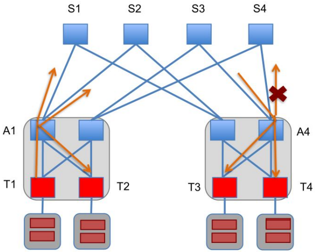
Efficiency - HULAOverview: Scalable, Proactive, Adaptive, and Programmable

4. Proactive path discovery: In HULA, probes are sent separately from data packets instead of piggybacking on them. This way, switches can instantaneously pick an uncongested path on the arrival of a new flowlet without having to first explore congested paths
5. Programmability:Processing a packet in a HULA switch involves switch state updates at line rate in the packet processing pipeline. In particular, processing a probe involves updating the best hop table and replicating the probe to neighboring switches. Processing a data packet involves reading the best hop table and updating a flowlet table if necessary.
6. Topology and transport oblivious: HULA is not designed for a specific topology.

Efficiency - HULA Design: Probes and Flowlets

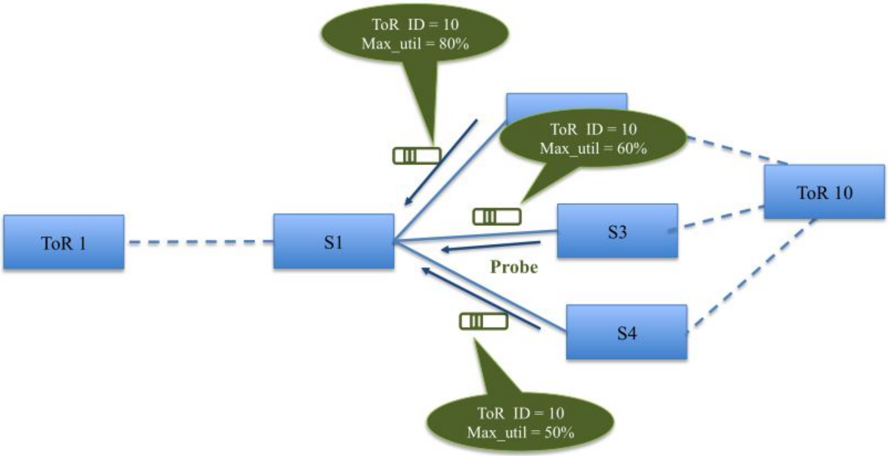
1. Origin and Replication of HULA Probes

– Every ToR sends HULA probes on all the uplinks that connect it to the data-center network
– Once the probes reach A1, it will forward the probe to all the other downstream ToRs (T2) and all the upstream spines (S1, S2).
– However, when the switch A4 receives a probe from S3, it replicates it to all its downstream ToRs but not to other upstream spines S4
– This makes sure that all paths in the network are covered by the probes
– Once a probe reaches another ToR, it ends its journey



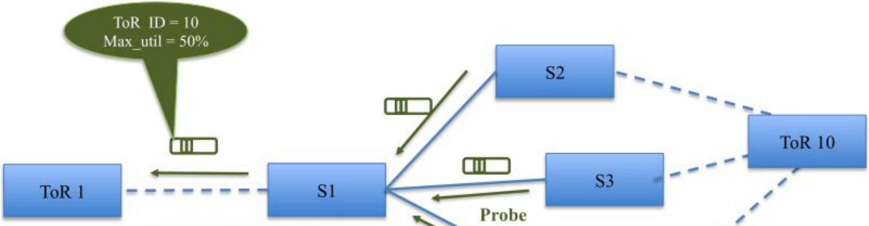
Efficiency - HULA Design: Probes and Flowlets

2. Processing Probes to Update Best Path



Efficiency - HULA Design: Probes and Flowlets

2. Processing Probes to Update Best Path

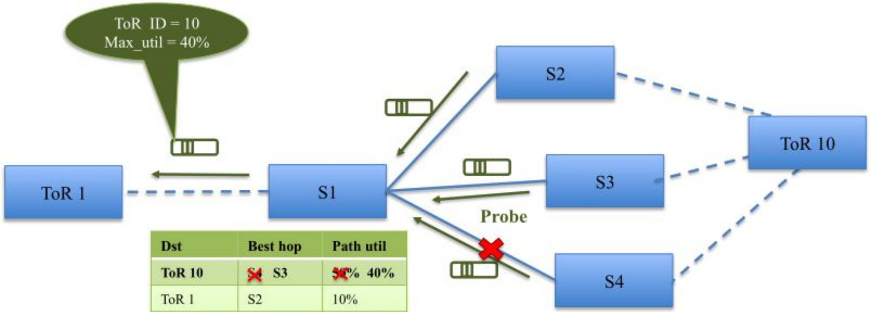


Dst	Best hop	Path util
ToR 10	**S4**	**50%**
ToR 1	S2	10%
...	...	

Best hop table

Efficiency - HULA Design: Probes and Flowlets

2. Processing Probes to Update Best Path

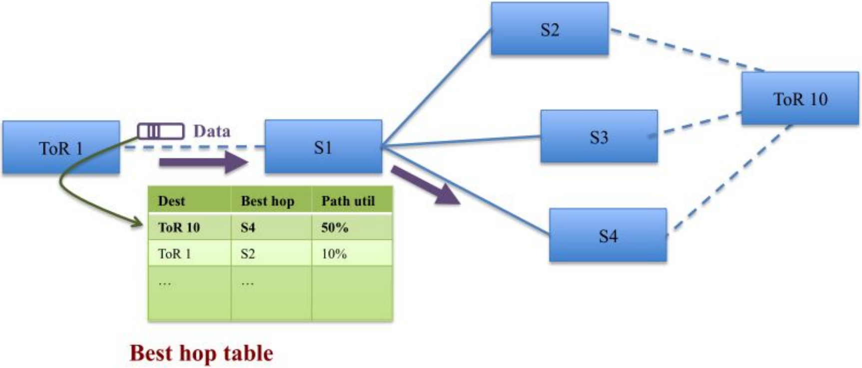


Dst	Best hop	Path util
ToR 10	~~S4~~ S3	~~[illegible]%~~ 40%
ToR 1	S2	10%
...	...	

Best hop table

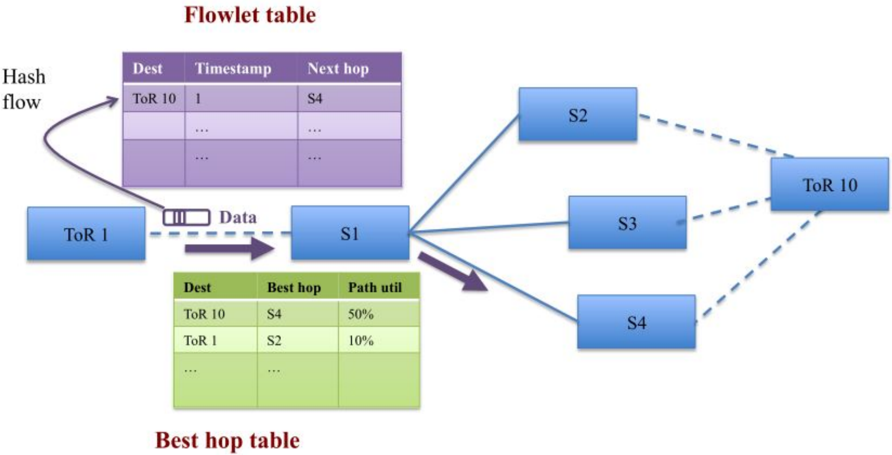
Efficiency - HULA Design: Probes and Flowlets

2. Processing Probes to Update Best Path



Efficiency - HULA Design: Probes and Flowlets

2. Processing Probes to Update Best Path



2. Processing Probes to Update Best Path

- Utilization Estimator. U $= D + U \times (1 - \frac{\Delta t}{\tau})$
  U is the link utilization estimator
  D is the size of the outgoing packet that triggered the update for the estimator
  Δt is the amount of time passed since the last update to the estimator
  $\tau$is a time constant

Efficiency - HULA Design: Probes and Flowlets

3. Flowlet Forwarding on Best Paths

- HULA load balances at the granularity of flowlets in order to avoid packet reordering in TCP
- A flowlet is detected by a switch whenever the inter-packet gap (time interval between the arrival of two consecutive packets) in a flow is greater than a flowlet threshold Tf
- Use flow table to keep track of all flowlets.

Efficiency - HULA Design: Probes and Flowlets

4. Data-Plane Adaptation to Failures

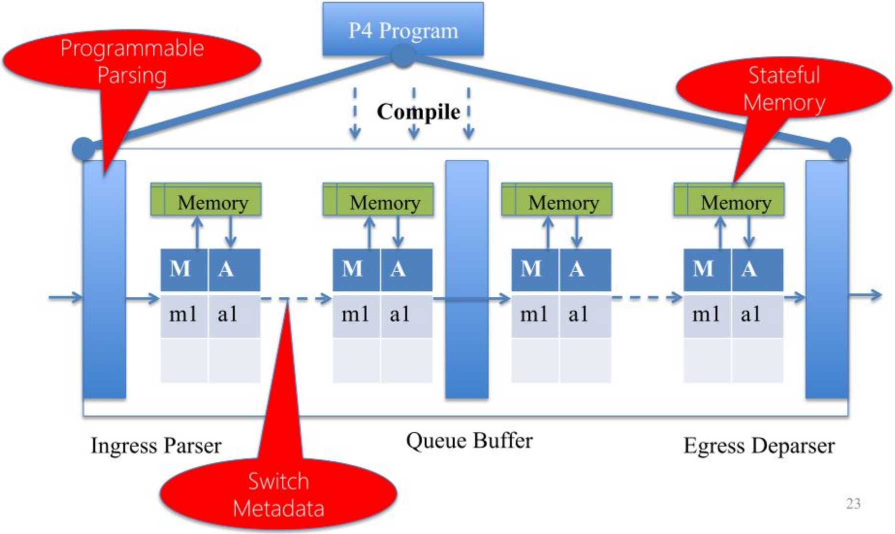
- HULA also learns about link failures from the absence of probes
- The data plane implements an aging mechanism for the entries in best Hop table.
- HULA tracks the last time bestHop was updated using an updateTime table. If a bestHop entry for a destination ToR is not refreshed within the last Tfail (a threshold for detecting failures), then any other probe that carries information about this ToR (from a different hop) will simply replace the bestHop and pathUtil entries for the ToR
- HULA does not need to rely on the control plane to detect and adapt to failures. And it is faster

Efficiency - HULA Design: Probes and Flowlets

5. Probe Overhead and Optimization

- Setting probe frequency:The network switches only use the congestion information to make load balancing decisions when a new flowlet arrives at the switch(CONGA).
- Optimization for probe replication:HULA maintains a lastSent table indexed by ToR IDs to avoid probe redundancy.Only one probe is sent by A to B within a time window of Tp
- Overhead: probe overhead on any given network link $= \frac{probeSize \times numToRs \times 100}{probeFreq \times linkBandwidth}$
  probeSize is 64 bytes, numTors is the total number of leaf ToRs supported in the network and probeFreq is the HULA probe frequency. Therefore, in a network with 40G links supporting a total of 1000 ToRs, with probe frequency of 1ms, the overhead comes to be 1.28%.

Efficiency - Programming HULA in P4



Efficiency - Programming HULA in P4

- HULA header format and control flow

```
header_type hula_header {
    fields{
        dst_tor : 24;
        path_util : 8;
    }
}

header_type metadata{
    fields{
        nxt_hop : 8;
        self_id : 32;
        dst_tor : 32;
    }
}
```

```
control ingress {
    apply(get_dst_tor)
    apply(hula_logic)
    if(ipv4.protocol == PROTO_HULA){
        apply(hula_mcast);
    }
    else if(metadata.dst_tor
              === metadata.self_id) {
        apply(send_to_host);
    }
}
```

Efficiency - Programming HULA in P4

- HULA stateful packet process in P4

```
action hula_logic{
  if(ipv4_header.protocol == IP_PROTOCOLS_HULA){
         /*HULA Probe Processing
     if(hula_hdr.path_util < tx_util)
         hula_hdr.path_util = tx_util;
     if(hula_hdr.path_util < min_path_util[hula_hdr.dst_tor] ||
         curr_time - update_time[dst_tor] > KEEP_ALIVE_THRESH)
     {
         min_path_util[dst_tor] = hula_hdr.path_util;
         best_hop[dst_tor] = metadata.in_port;
         update_time[dst_tor] = curr_time;
     }
     hula_header.path_util = min_path_util[hula_hdr.dst_tor];
  }
 else { /*Flowlet routing of data */
     if(curr_time – flowlet_time[flow_hash]> FLOWLET_TOUT) {
         flowlet_hop[flow_hash] = best_hop[metadata.dst_tor];
     }
     metadata.nxt_hop = flowlet_hop[flow_hash];
     flowlet_time[flow_hash] = curr_time;
  }
}
```

Efficiency - Evaluation of HULA

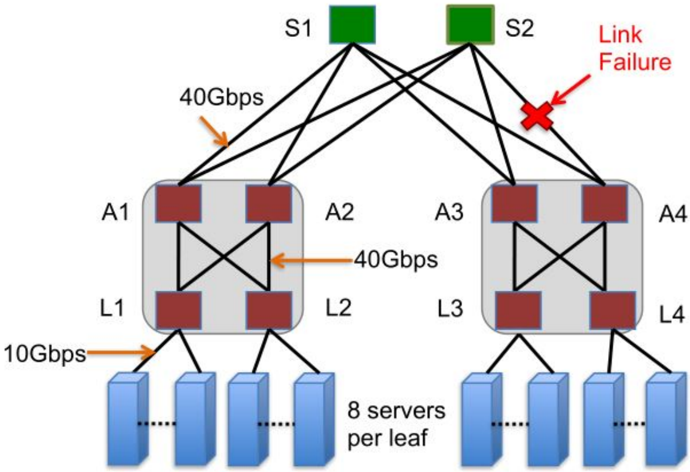
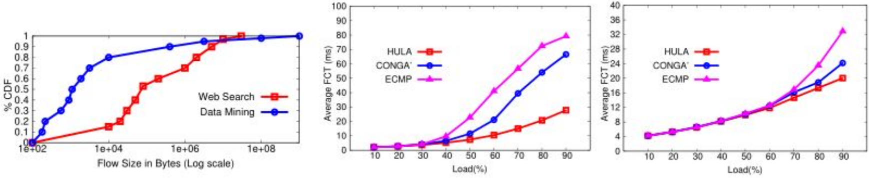


Figure 4: Topology used in evaluation



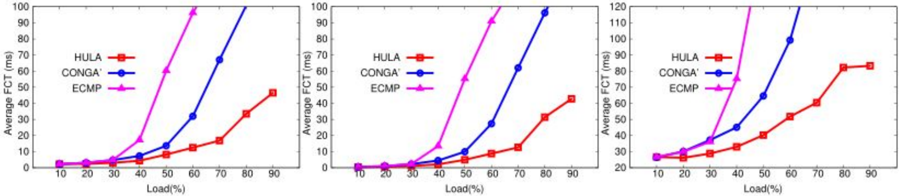
(a) Empirical traffic distribution used in evaluation

(b) Web-search overall avg FCT

(c) Data-mining overall avg FCT

Figure 5: Average FCT for the Web-search and data-mining workload on the *symmetric* topology.

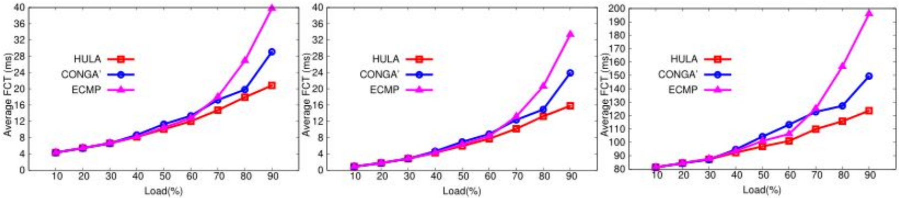
Efficiency - Evaluation of HULA



(a) Overall Average FCT (b) Small Flows (<100KB) (c) Large Flows (>10MB)

Figure 6: Average FCT for the Web-search workload on the *asymmetric* topology.

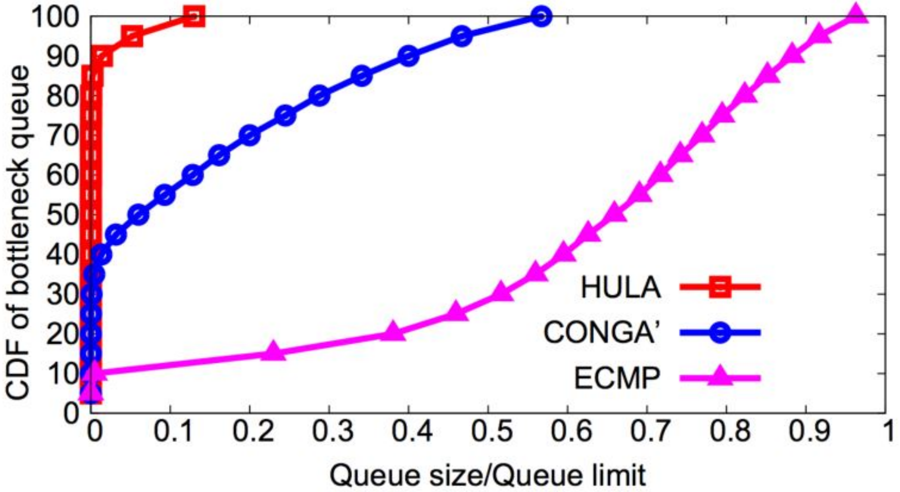
Efficiency - Evaluation of HULA



(a) Overall Average FCT (b) Small Flows (<100KB) (c) Large Flows (>10MB)

Figure 7: Average FCT for the data mining workload on the *asymmetric* topology.

Efficiency - Evaluation of HULA



Flexibility

Naga Katta, Omid Alipourfard,Jennifer Rexford and David Walker
CacheFlow: Dependency-Aware Rule-Caching for Software-Defined Networks
Proceedings of the Symposium on SDN Research,
Proceedings of the Symposium on SDN Research

Reliability

Naga Katta
Building Efficient and Reliable Software-Defined Networks
PhD. Dissertation,Princeton University

References

Naga Katta, Mukesh Hira, Changhoon Kim, Anirudh Sivaraman, Jennifer Rexford, March 14-15, 2016, Santa Clara, CA, USA

HULA: Scalable Load Balancing Using Programmable Data Planes Proceedings of the Symposium on SDN Research,

Proceedings of the Symposium on SDN Research

Naga Katta

Building Efficient and Reliable Software-Defined Networks

PhD. Dissertation,Princeton University

Questions?