Ward: Implementing Arbitrary Hierarchical Policies using Packet Resubmit in Programmable Switches

- **Mojtaba Malekpourshahraki**
- Brent Stephens
- Balajee Vamanan

BITS
Networked Systems Laboratory

THE UNIVERSITY OF ILLINOIS AT CHICAGO

mmalek3@uic.edu

Modern datacenter

- Datacenters host multiple applications with different requirements
  - Memcache
  - Spark
  - Backup and replication

- Datacenters host multiple competing tenants
  - Private datacenters
    - Facebook → Product and applications groups
  - Public datacenters
    - Microsoft Azure → Users renting virtual machines

Amazon EC2

Multiple tenants and diverse applications per tenant cause contention problem

Multi-tenancy problems

- Fairness
  - Large flows may saturate the whole bandwidth
    - Long delay for short flows

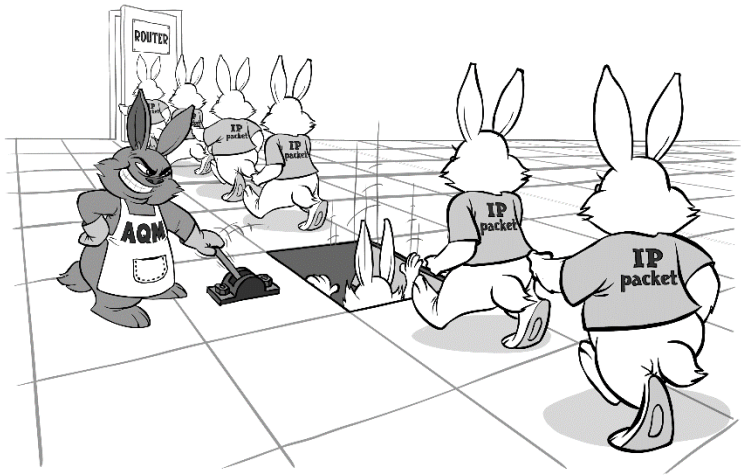
- Pricing and SLA
  - Better service for more money

How to handle these problems?

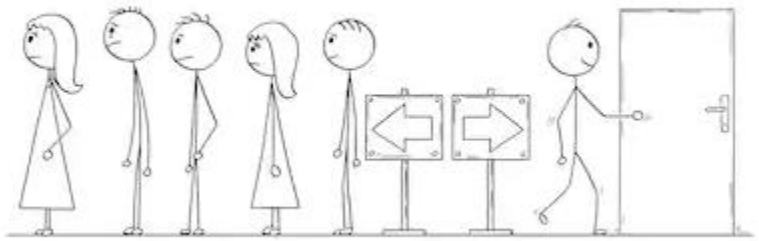
Datacenter needs to isolate the traffic based on a predefined policy

How to enforce desirable policies

- Enforce policies using:
  - Congestion control algorithms
    - Use congestion signals to change the source sending rate

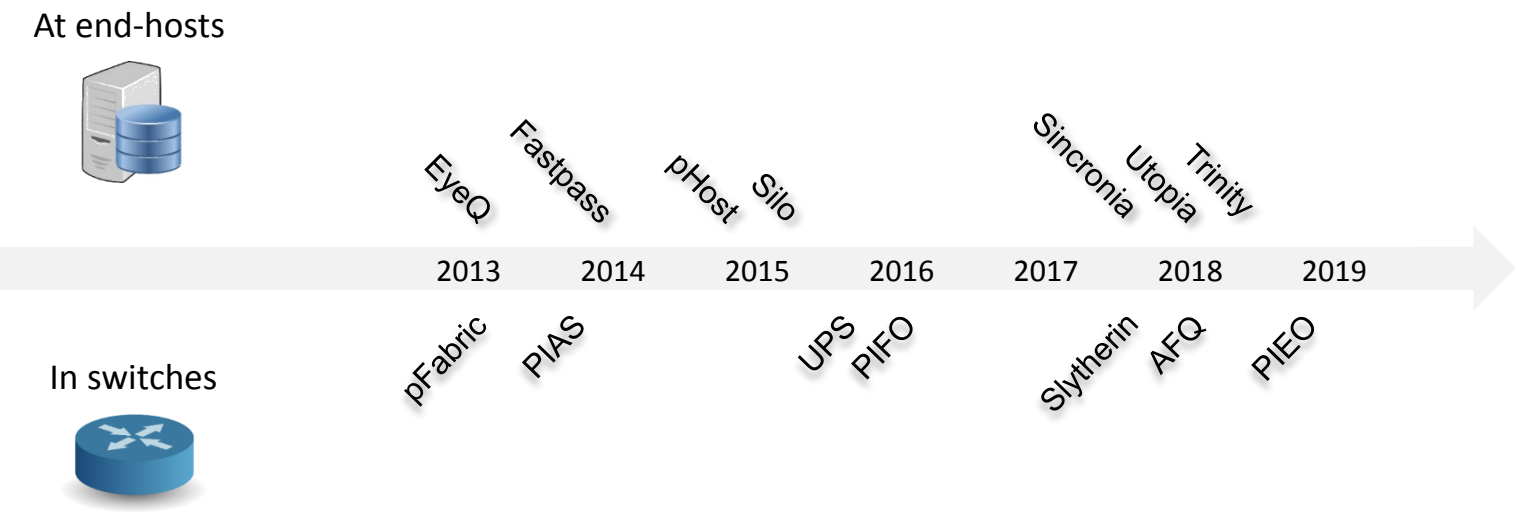


  - Schedulers
    - Change the dequeue to enforce desirable policy



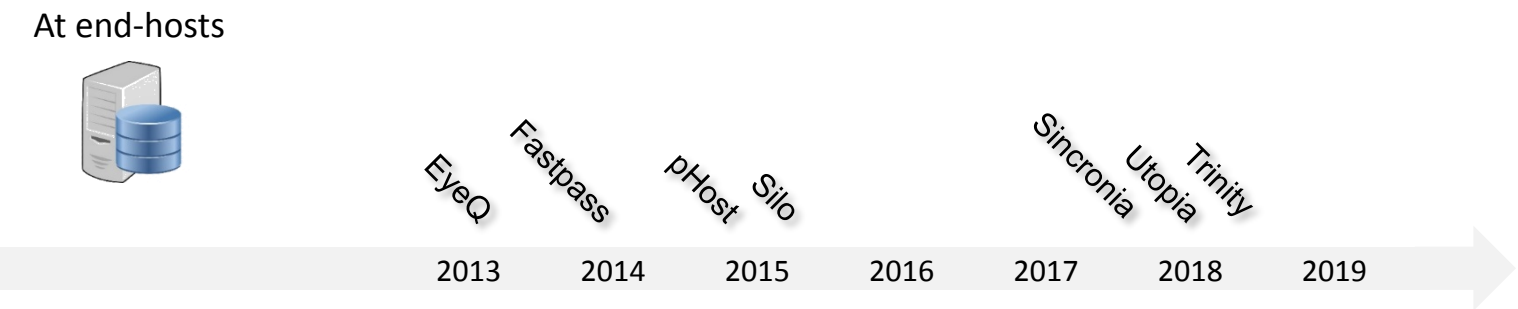
Approaches with policy considerations

- Two types of approaches
  - End-host based (end-to-end)
  - Switch based



End-host based approaches

- End-host
  - Has many queues → only at the end host doesn't need our requirement
- Shortcomings
  - Waste of resource
    - pHost (Sending RTS)
    - Silo (Limiting burst size)
  - High computational overhead
    - Fastpass (Centralized)

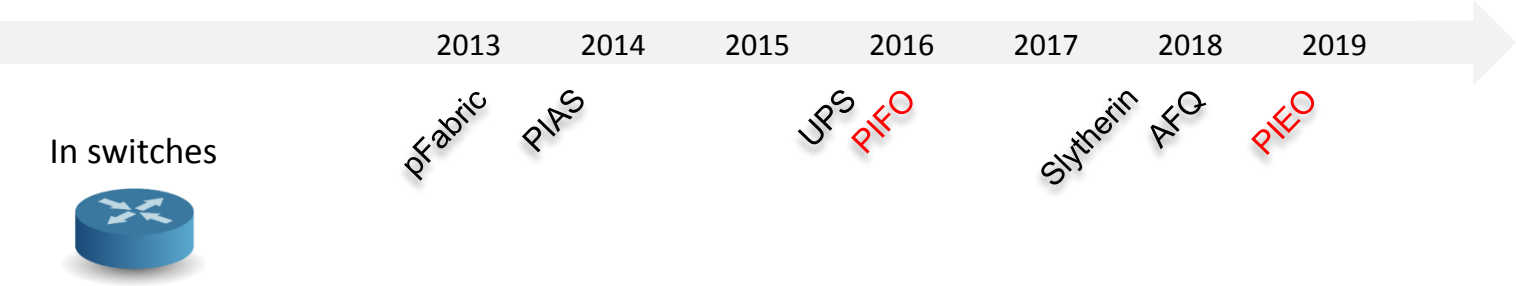


***In switch* approaches perform faster than end-to-end schedulers**

PIFO [SIGCOMM, 2016]/PIEO [SIGCOMM, 2019]

- PIFO/PIEO:
  - Can implement complex hierarchical programmable scheduling policies

- PIFO/PIEO resources on a switch are limited (less than 100 queues)
  - Cannot have all possible scheduler
  - The number of required queues increases with the number of traffic class



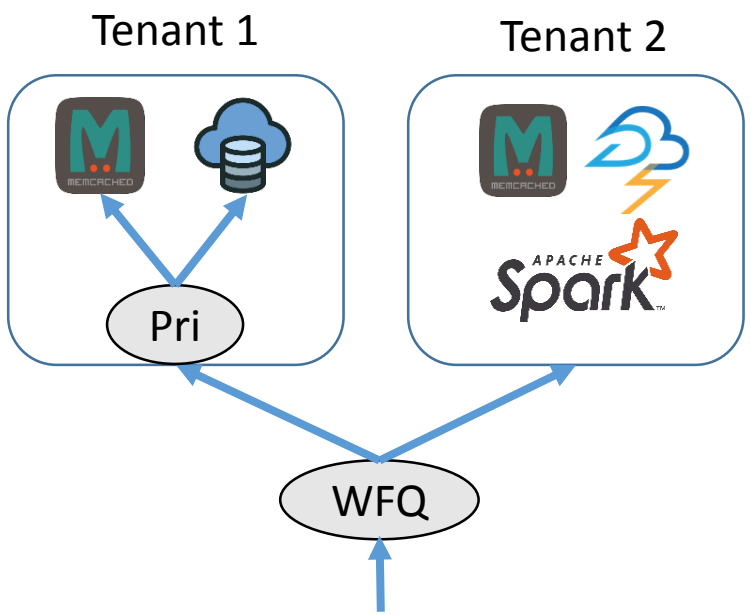
Key question!

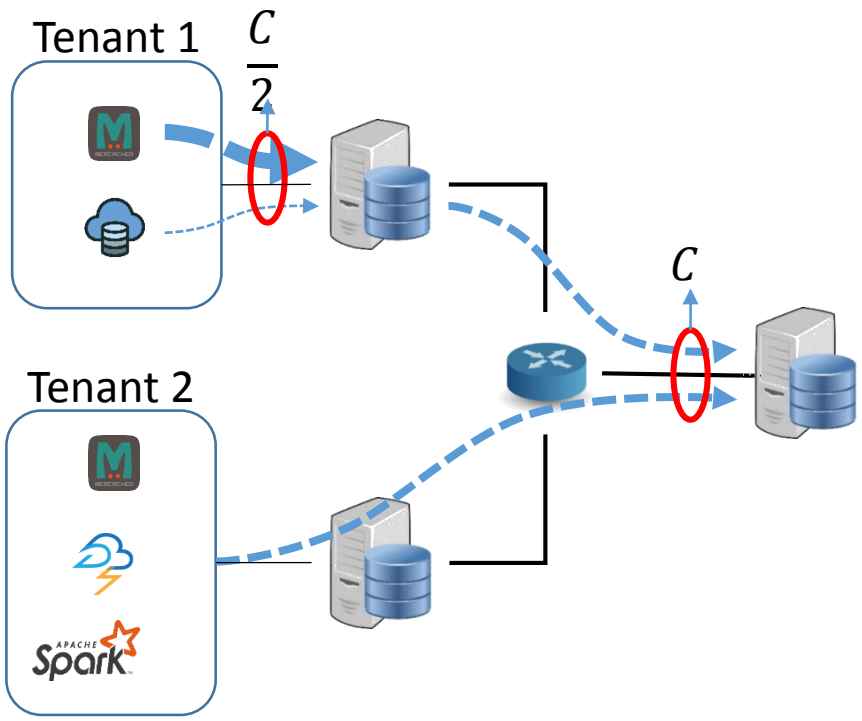
Can we implement any arbitrary hierarchical policy with limited switch resources in programmable switches?

Ward

Ward - Policy abstraction

- Different types of flows, applications, and tenants
  - Represent with hierarchical policy
  - Directed acyclic graph (DAG)



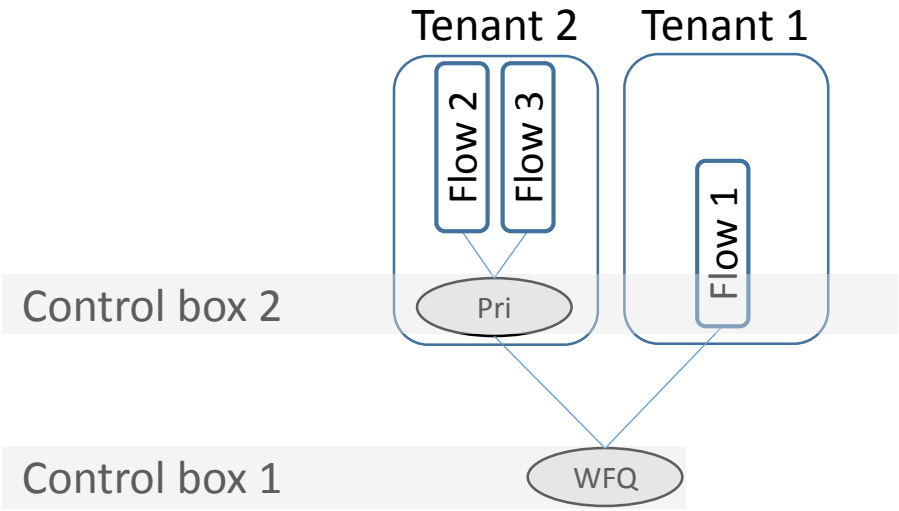


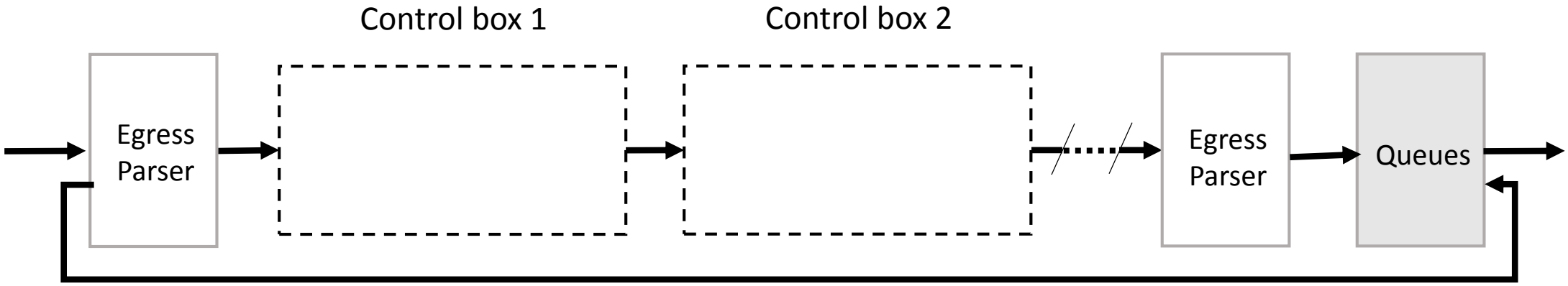
Strict priority for Memcashed over backup traffic

Equal share for tenants

Ward framework

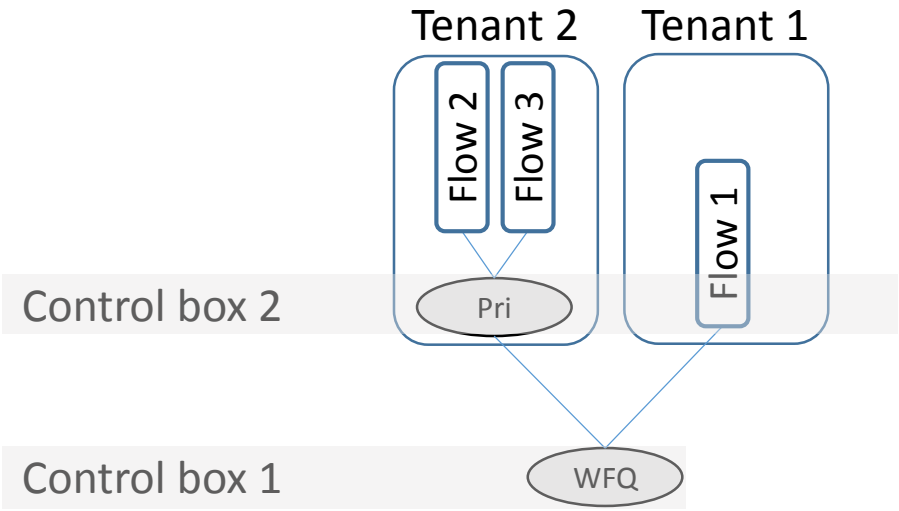
- Ward uses control boxes
  - One control box for each level of the tree
  - First box process the root of the tree

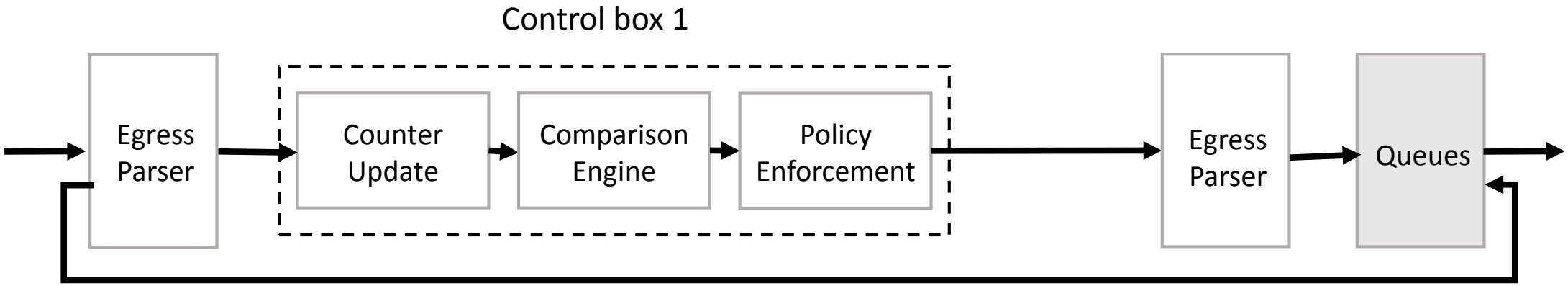




Ward framework

- Each control box has three steps:
  - Counter update
  - Comparison engine
  - Policy enforcement

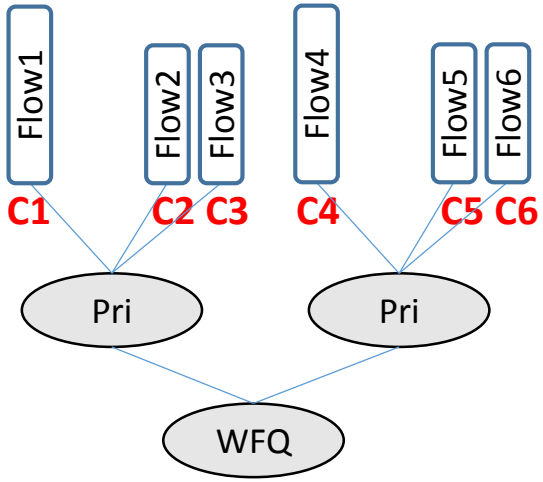




Ward framework – Counter update

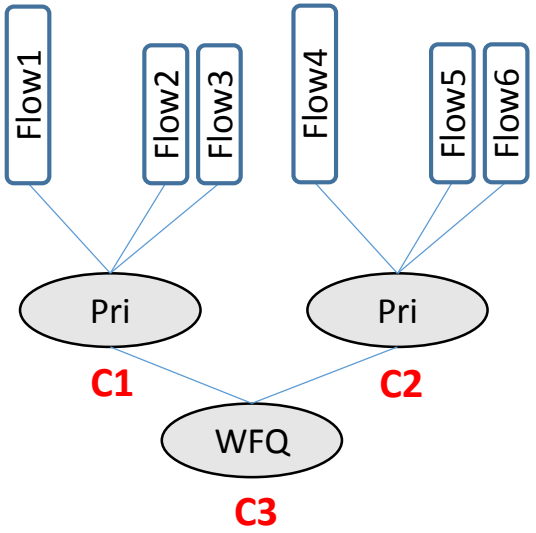
- Dual counter
  - Use a single counter to compare two streams of packets (e.g., flow, tenants)

One counter per flow

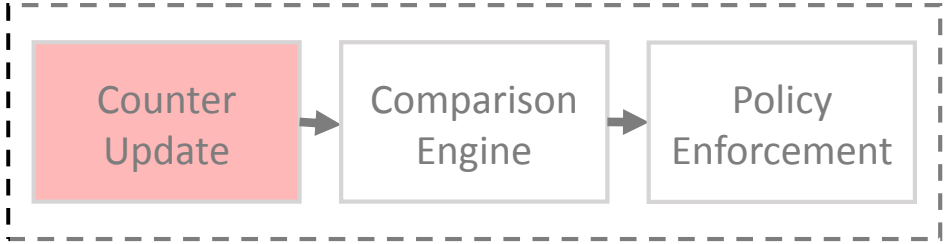


Six counters

One counter per policy node

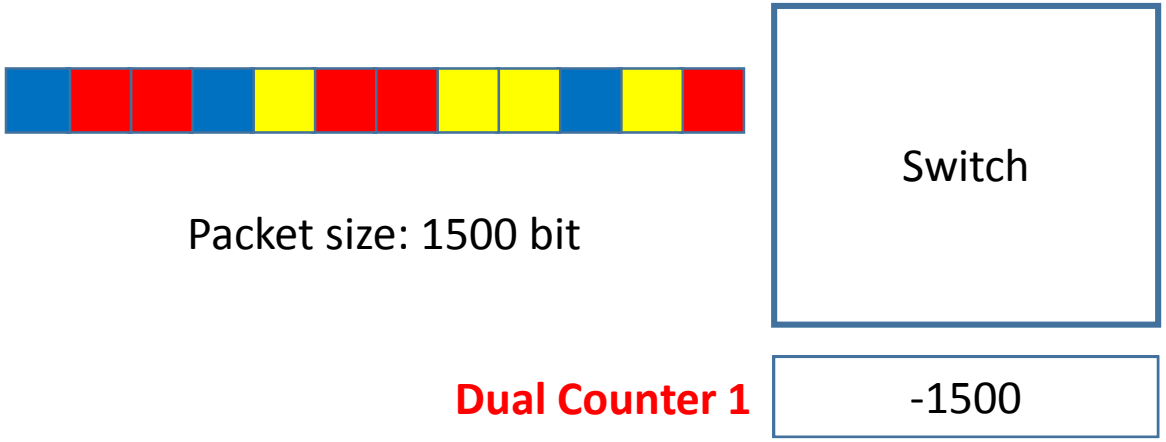


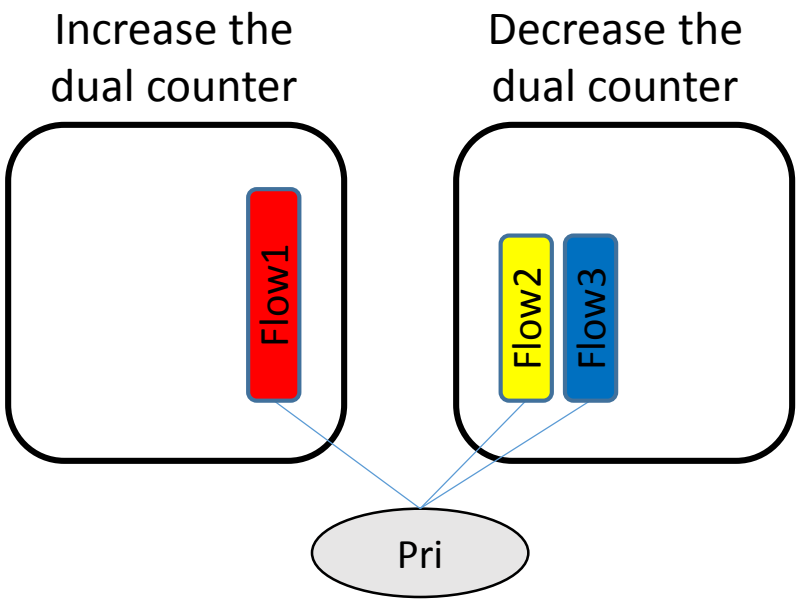
Three dual counters

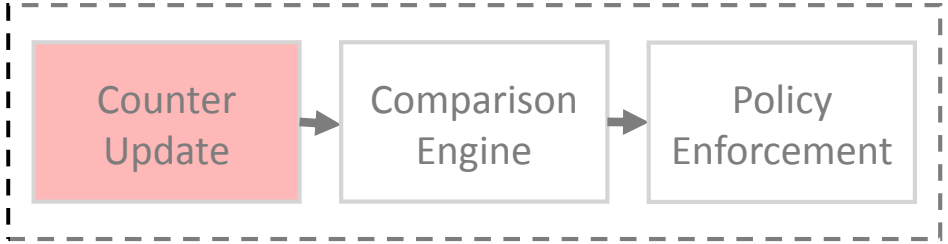


Ward framework – Counter update

- Dual Update counter:
  - Packet from left → increase the counter by a packet size unit
  - Packet from right → decrease the counter by a packet size unit

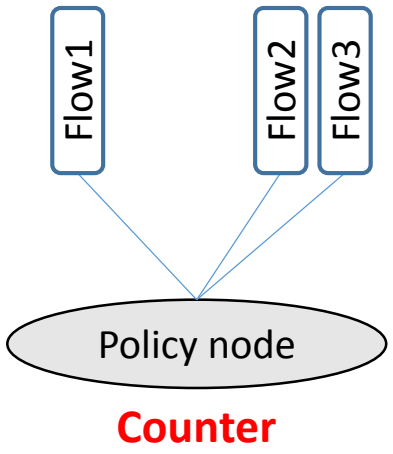






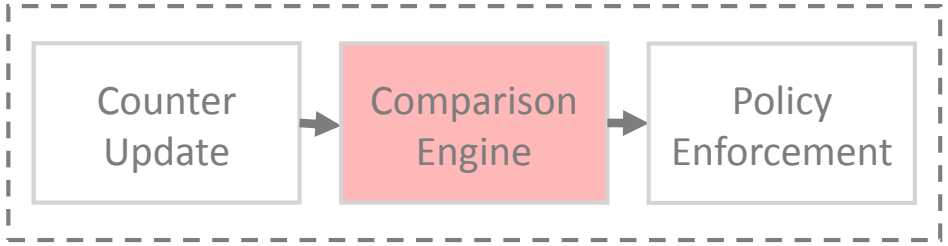
Ward framework – Comparison engine

- Check the value of the corresponding dual center
  - Check if the traffic violates the current policy or not
    - If yes: use a rate control policy




```
If (packet class is in left leaf and counter < 0):
    Choose a policy and add it as meta-data
If (packet class is in right leaf and counter > 0):
    Choose a policy and add it as meta-data
Otherwise
    Bypass the packet
```

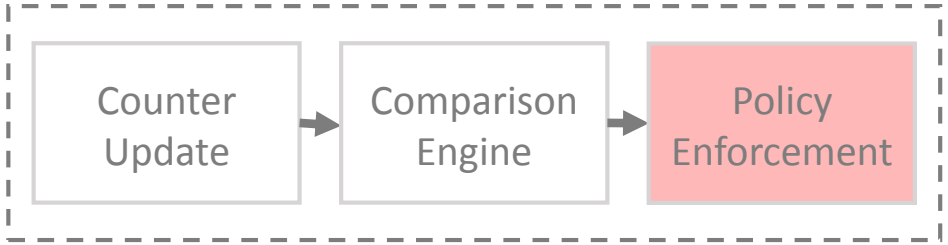
	Rate Control Policy
WFQ	ECN, Drop
Pri	Drop
SJF	ECN, Priority Queue,Drop



Ward framework – Policy enforcement

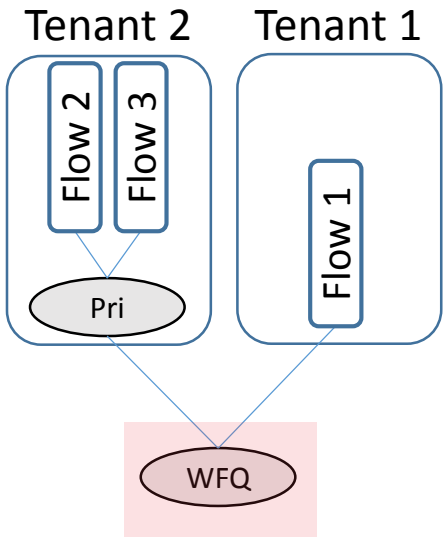
- Merge the result of the previous rate control decisions with the current one
  - Enforce the strongest rate control
  - Order of rate control decisions

Drop > Enqueue > ECN mark



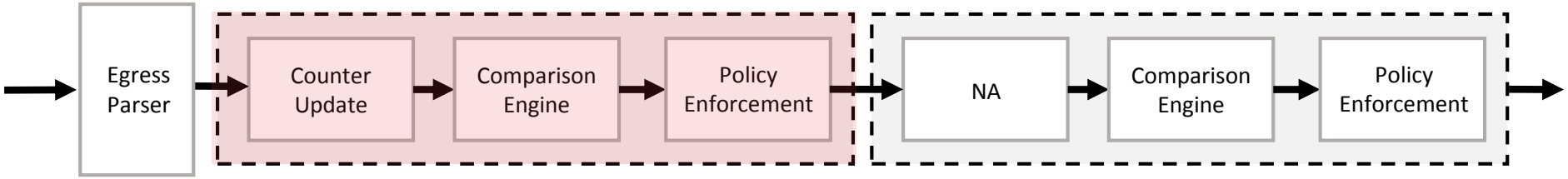
Example

- Two level of policy
  - Incoming packet belongs to Flow 2
  - Flow 2 has lower priority than Flow 3

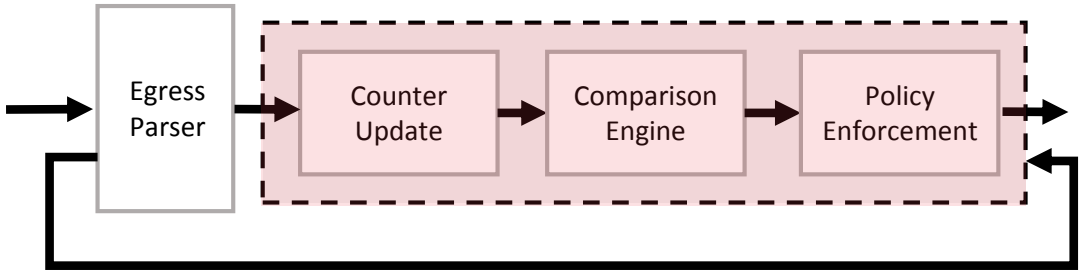


	Policy	Counter	Comparison	Enforce
First Box	WFQ	Update	ECN	$(ECN, \emptyset) = ECN$
Second Box	Pri	NA	Drop	$(ECN, Drop) = Drop$

Enough stages

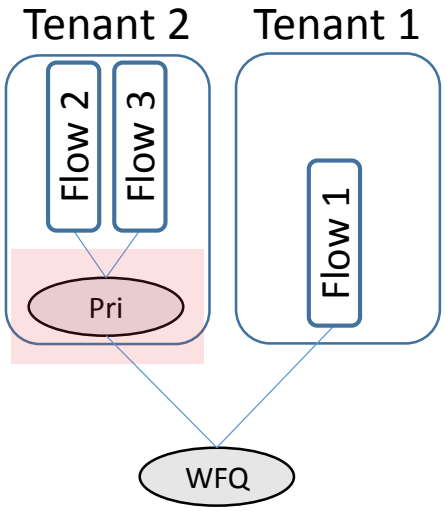


Using resubmit



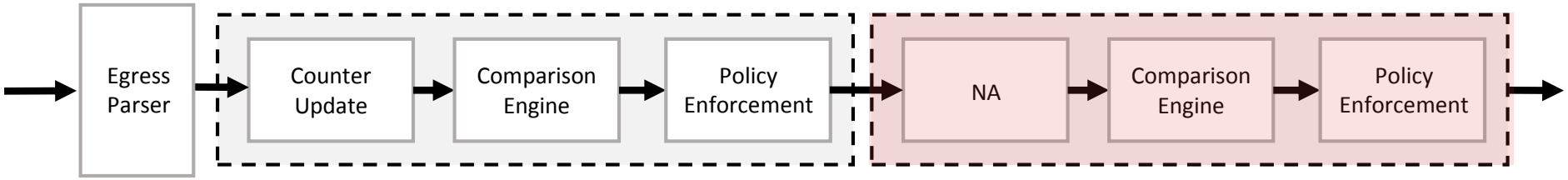
Example

- Two level of policy
  - Incoming packet belongs to Flow 2
  - Flow 2 has lower priority than Flow 3

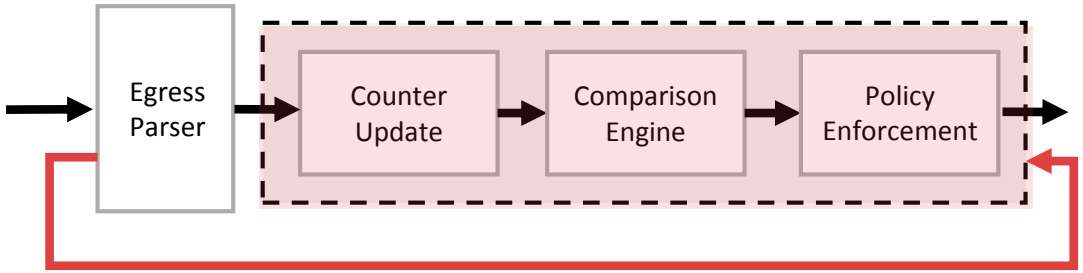


	Policy	Counter	Comparison	Enforce
First Box	WFQ	Update	ECN	$(ECN, \emptyset) = ECN$
Second Box	Pri	NA	Drop	$(ECN, Drop) = Drop$

Enough stages

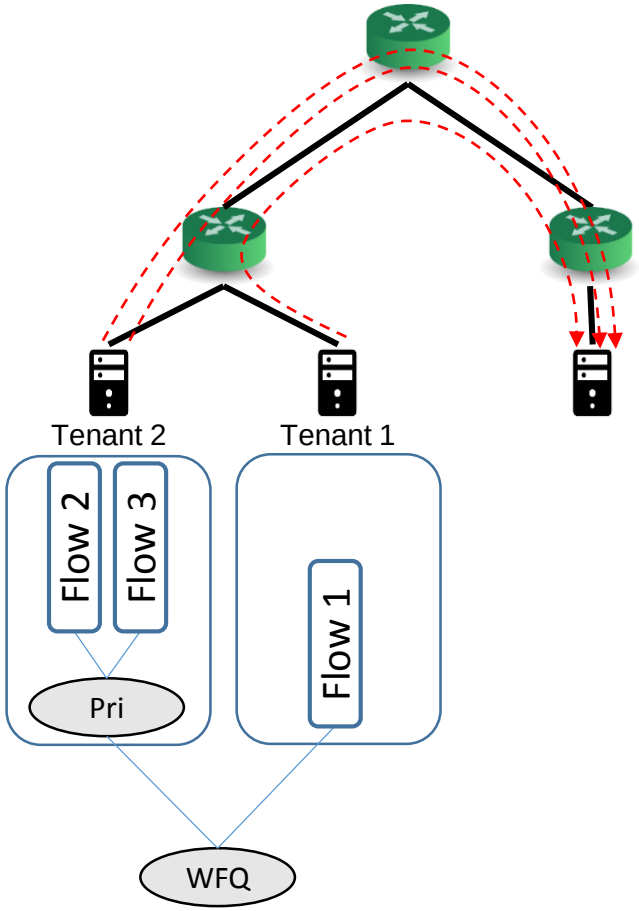


Using resubmit

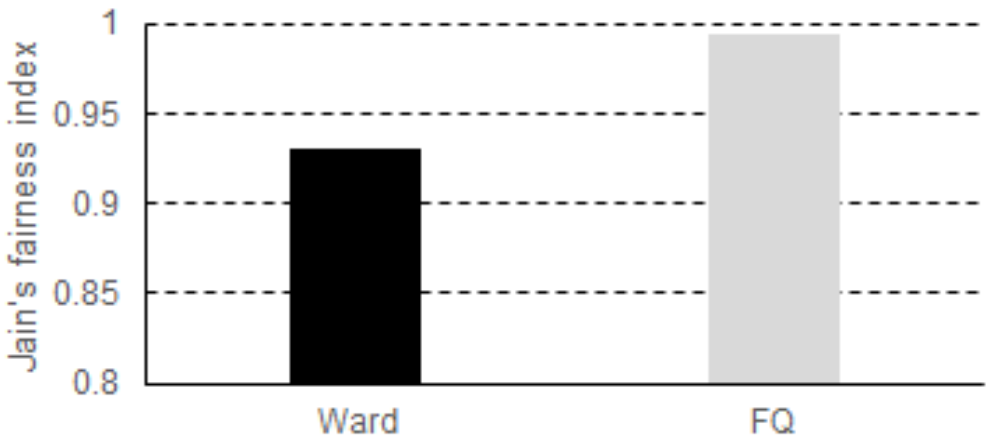


Evaluation

- Evaluation goals
  - Can Ward enforces a hierarchical policy in the network?
- Measured parameters
  - Jain's index fairness
  - 99 percentile tail flow completion time
- Evaluate Ward in both Mini-net/BMv2 and ns3
  - Leaf-spine topology
  - All links are 10 Gbps

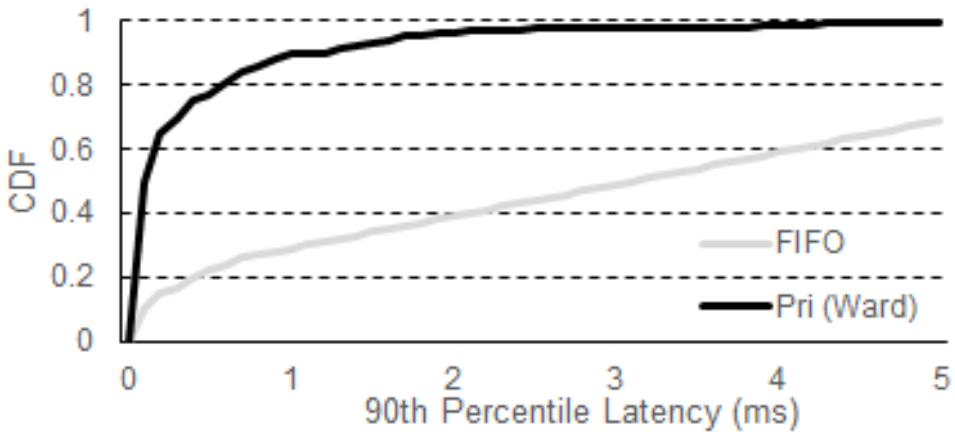


Fairness BMv2



Ward provides fairness with only 6% lower index compared to the ideal fair queuing

90th percentile for short flows



Ward can enforce strict priority compared to FIFO

Conclusion

- We proposed Ward
  - Implementing hierarchical policies
  - Use packet resubmit in programmable switches if needed

- We observed that Ward
  - Ward can enforce two-level policy

- Future work:
  - Implement Ward on programmable switches
  - Study the balance of performance loss and the depth of the policy tree
  - Optimize Ward architecture to reduce the number of required stages

Thanks for the attention

Mojtaba Malekpourshahraki

Email: mmalek3@uic.edu
Website: cs.uic.edu/~mmalekpo

UIC
UNIVERSITY OF ILLINOIS AT CHICAGO
Bits Networked Systems Laboratory
COLLEGE OF ENGINEERING

Reserved Slides