Programmable Calendar Queues for High-speed Packet Scheduling

Naveen Kr. Sharma[1], Chenxingyu Zhao[1], Ming Liu[1], Pravein G Kannan[2], Changhoon Kim[3], Arvind Krishnamurthy[1] and Anirudh Sivaraman[4]

UNIVERSITY of WASHINGTON

NUS National University of Singapore

BAREFOOT NETWORKS

NYU

PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING

Packet Scheduling

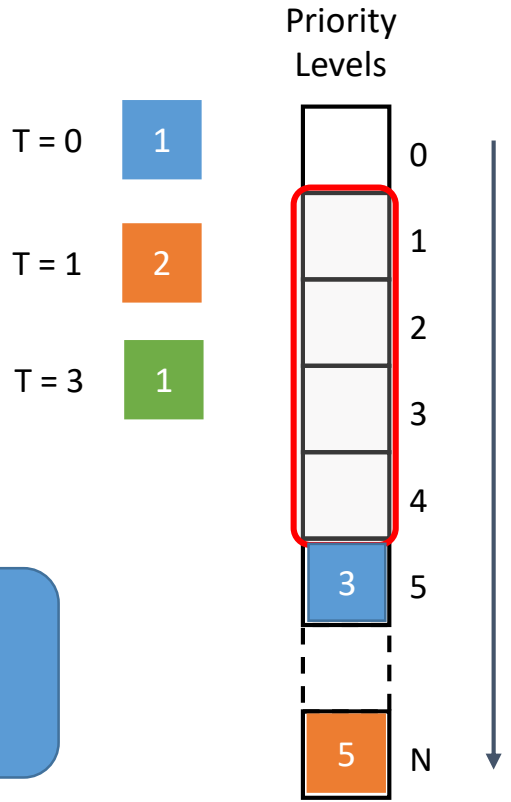
- Many scheduling algorithms require ordering packet at switches
- Enables rich application guarantees such as WFQ, EDF or SRPT
- Generally implemented using a priority queue with static priorities
  - Packet's priority (rank) is computed by the ingress pipeline
  - The priority does not change until the packet is transmitted
- However, static priorities are insufficient for several algorithms

Static Priority Limitations

Least Slack Time First

- Each packet has slack denoting time until delivery
- Enqueue packet with `rank = current_time + slack`
- Ranks increase over time, eventually exhausting priorities
- Other algorithms, WFQ, EDF, LBF have this property as well

- Need a mechanism that supports “dynamic priorities”
  - Implementable at high-speeds (preferably a bolt-on)



Calendar Queues (CQs)

- Proposed by Brown'88 for processing events in discrete event simulator
- Bucketed priority queue with O(1) insert and deletes
- Analogous to a desk calendar, consisting of multiple days
  - Events are scheduled by specifying a future day
  - Dequeued from the current day in sorted order
  - Once events are exhausted for a day, move onto next day – priority escalation
  - Make the previous day available to reuse at lowest priority – priority reuse

Our Contribution: Programmable Calendar Queues

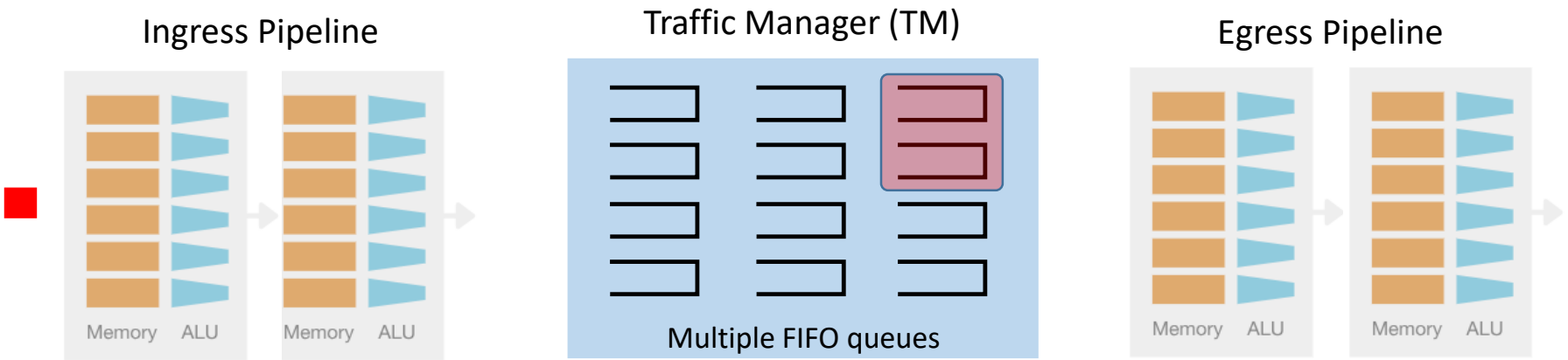
Combine calendar queues abstraction with programmable pipelines to realize scheduling algorithms at line-rate on today's hardware

- Calendar Queues provide dynamic priorities
- Programmable pipelines maintain scheduling algorithm state

Outline

- Background
- Programmable Calendar Queue (PCQs)
- Realizing scheduling algorithms on PCQs
- Implementing PCQs in hardware
- Case Study : Coflow Scheduling
- Case Study : Weighted Fair Queuing

Reconfigurable Switches

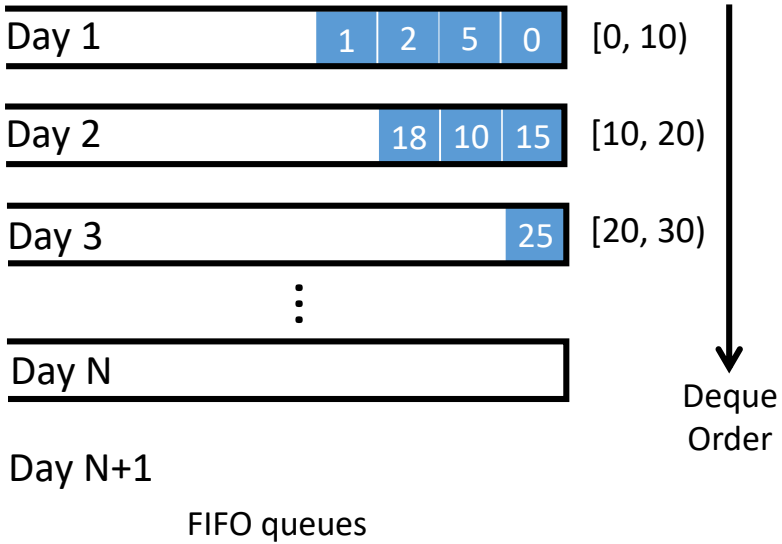


- Packets processed by ingress pipeline before being buffered in the TM
- Multiple queues attached to an egress port, configured using the switch CPU
- Queues scheduled using priority or round robin, with support for pausing

Programmable Calendar Queues (PCQs)

- Calendar Queue with programmable and stateful rank computation
- Customizable and configurable day duration and rotation policy

- Each day is mapped to a FIFO queue
- Packet ranks are bucketed into days
- Earliest day has highest deque priority
- Move to next day periodically
- Reuse the queue for future day



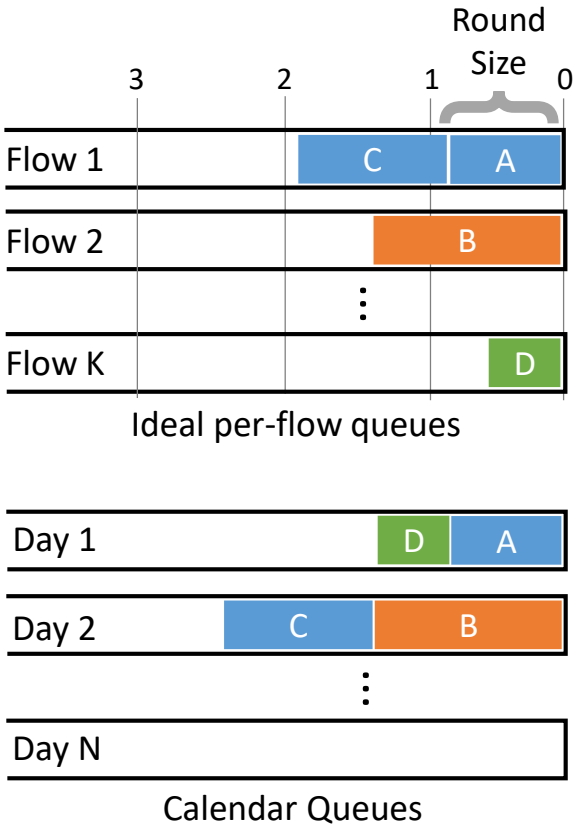
Realizing Algorithms using PCQs

- Calculate which day to enqueue arriving packets – Rank Compute
  - How far into the future to schedule the packet
- Decide when to move onto next day – Queue Rotation
  - When the current queue is empty – Logical Calendar Queue
  - Periodically based on wall clock time – Physical Calendar Queue
- Update algorithm state and enqueing behavior – State Update
  - Ensures algorithm invariants are maintained on rotation

Example using PCQs: Fair Queueing

- Emulate bit-by-bit round robin fair queueing
- Each round corresponds to a day in the CQ

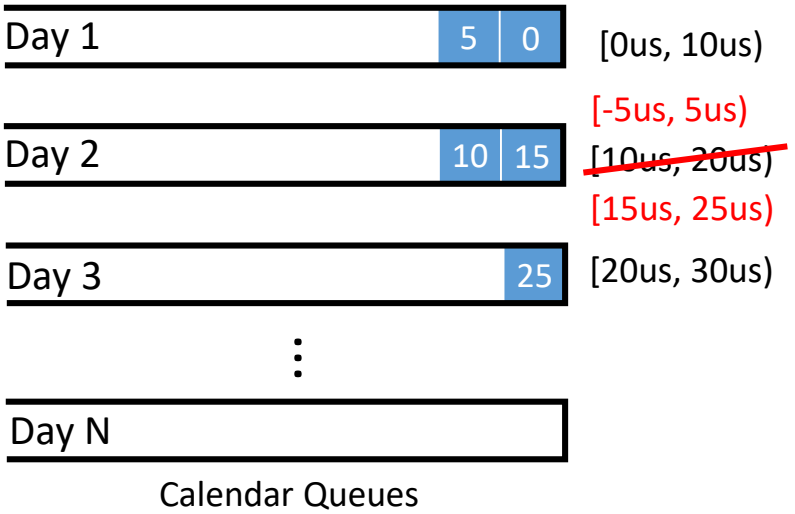
- Rank Computation
  - Rank = bytes sent by flow / round size
- Queue Rotation
  - Whenever the current queue is empty
- State Update
  - Increment round number by 1



Example using PCQs: Earliest Deadline First

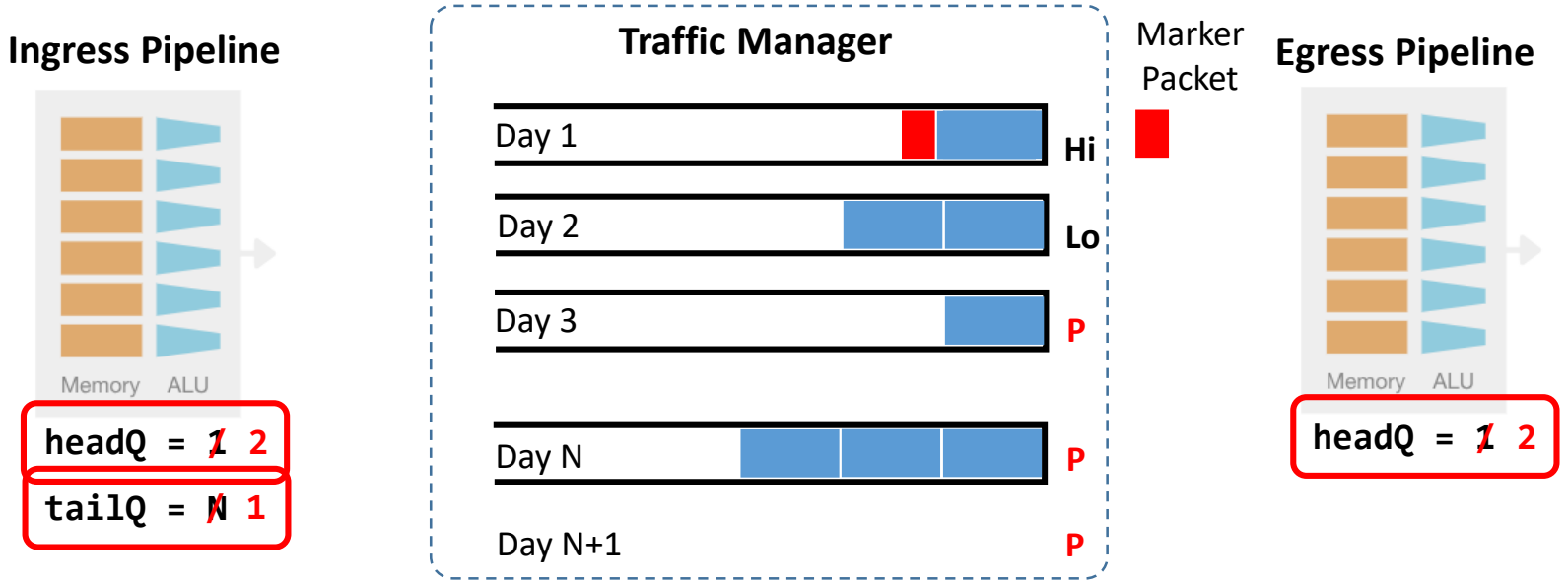
- Bucket packet deadlines into queues based on day duration
- Keep track of drift to maintain correct dequeue order

- Rank Computation
  - Rank = deadline + drift / bucket size
- Rotation
  - Current queue is empty
- State Update
  - Adjust drift based on time spent



Implementing PCQs in hardware

- Mutable switch state and recirculation of special packets
- Ability to change queue priority and status



Hardware Feasibility

- Most efficient implementation requires data plane support for modifying queue priority and status
  - Expected in next generation of programmable switches
  - Limited version already available for PFC mechanism
- Less responsive version can be realized using control plane
  - Our prototype uses switch CPU to update queue priorities

More details in the paper

- Approximations in PCQs
- Hierarchical Calendar Queues
- Expressiveness and Limitations of PCQs
- Hardware Prototype Results

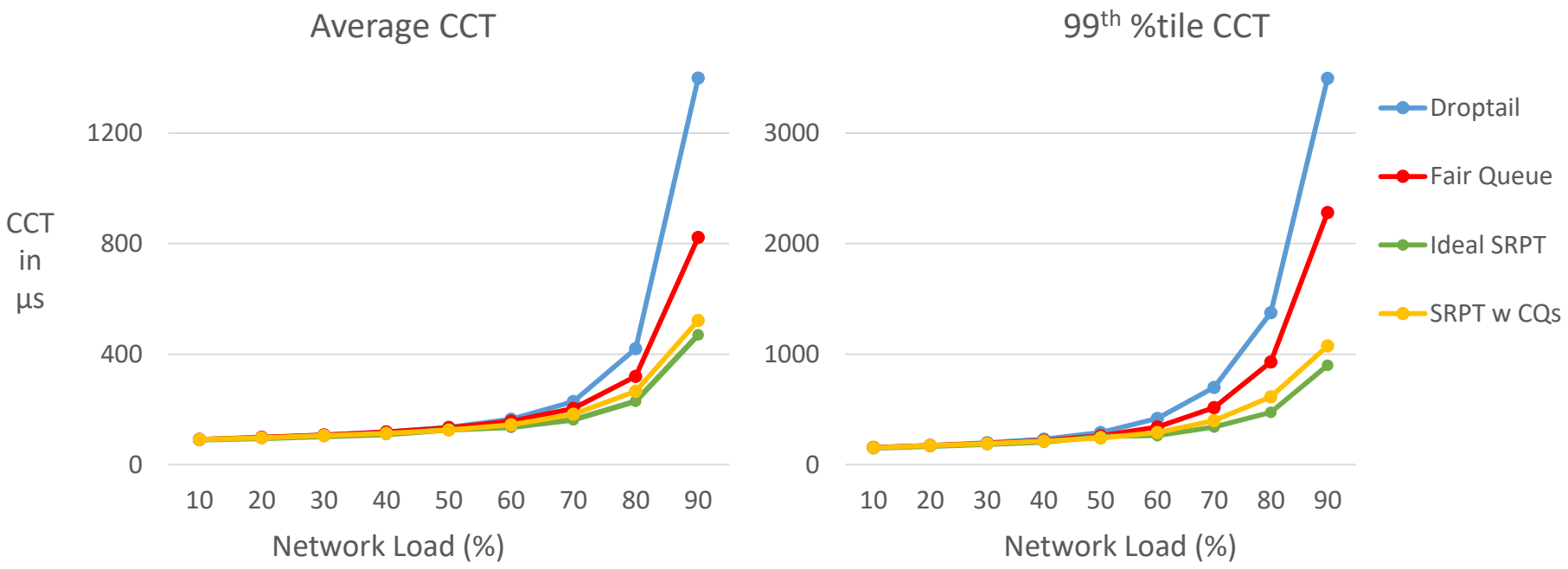
Case Study: Coflow Scheduling

- Many applications optimize the performance of collection of flows
- Ordering coflows smallest to largest gives close to optimal results
- We implement such a scheme using LSTF scheduling on PCQs
- Slack is set to the expected finish time of the largest sub-flow
- At any hop, packet with the shortest slack is sent out first

Coflow Testbed Setup

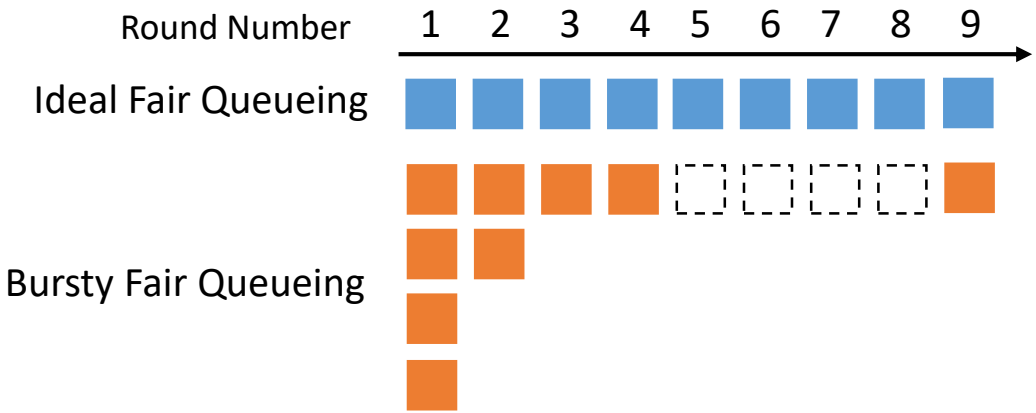
- 3-level fat-tree testbed with coflow and background traffic
- Each switch port implements a PCQ with 32 FIFO queues
- Compared with DCTCP over droptail and fair-queueing
- Measure and report the Coflow Completion Time (CCT)

Coflow Scheduling Evaluation

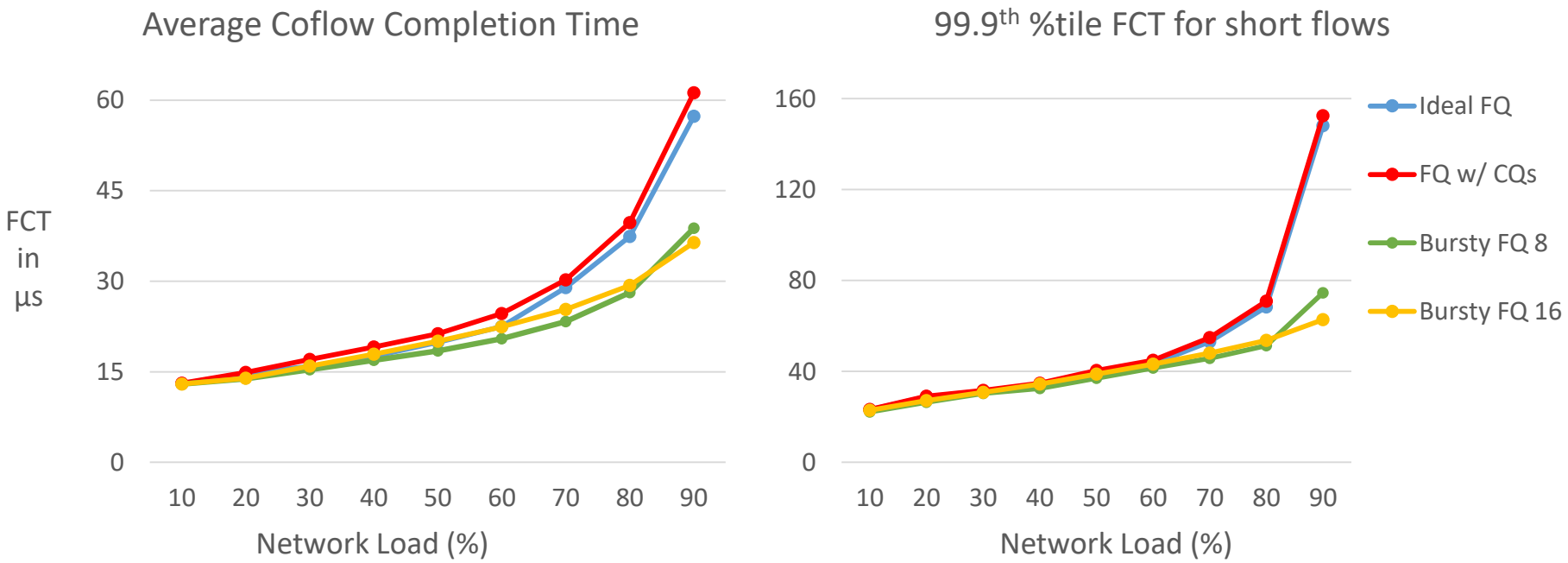


Case Study: Burst-friendly Fair Queueing

- Emulate a bit-by-bit round robin scheme at coarse granularity
- Desirable to permit a burst of packets for better tail latency
- Sacrifices fairness at short timescales but maintains it at long timescale



Burst-friendly Fair Queueing Evaluation



Summary

- Static priority mechanisms insufficient for class of scheduling algorithms
- Calendar Queue based approach is a better fit
- Can be implemented on today's multi-pipeline, high speed switches
- Inherently scalable to higher bandwidth and number of flows
- With a programmable pipeline, can implement a variety of algorithms