Universal Packet Scheduling

Radhika Mittal, Rachit Agarwal, Sylvia Ratnasamy, Scott Shenker

UC Berkeley
Many Scheduling Algorithms

• Many different algorithms
  – FIFO, FQ, virtual clocks, priorities…

• Many different goals
  – fairness, small packet delay, small FCT…

• Many different contexts
  – WAN, datacenters, cellular…
Many Scheduling Algorithms

- Implemented in *router hardware.*

- **How do we support different scheduling algorithms for different requirements?**
  - Option 1: Change router hardware for each new algorithm
  - Option 2: Implement *all* scheduling algorithms in hardware
  - Option 3: Programmable scheduling hardware*

*Towards Programmable Packet Scheduling, Sivaraman et. al., HotN
Many Scheduling Algorithms

- Implemented in *router hardware*.

- **How do we support different scheduling algorithms for different requirements?**
  - Option 1: Change router hardware for each new algorithm
  - Option 2: Implement *all* scheduling algorithms in hardware
  - Option 3: Programmable scheduling hardware*

*Towards Programmable Packet Scheduling, Sivaraman et. al., HotN
Many Scheduling Algorithms

- Implemented in *router hardware*.

- **How do we support different scheduling algorithms for different requirements?**
  - Option 1: Change router hardware for each new algorithm
  - Option 2: Implement *all* scheduling algorithms in hardware
  - Option 3: Programmable scheduling hardware*

*Towards Programmable Packet Scheduling, Sivaraman et. al., HotNets*
Many Scheduling Algorithms

• Implemented in *router hardware.*

• **How do we support different scheduling algorithms for different requirements?**
  – Option 1: Change router hardware for each new algorithm
  – Option 2: Implement *all* scheduling algorithms in hardware
  – Option 3: Programmable scheduling hardware*

*Towards Programmeable Packet Scheduling, Sivaraman et. al., HotN
We are asking a new question.....

How do we support different scheduling algorithms for different requirements?

Is there a universal packet scheduling algorithm?
UPS: Universal Packet Scheduling Algorithm

A single scheduling algorithm that can imitate the network-wide output produced by any other algorithm.
How can a single algorithm imitate all others?
Network Model

Input Traffic → CORE NETWORK

INGRESS
Network Model

Input Traffic  →  Scheduling Algorithm  →  CORE NETWORK

INGRESS
Network Model

Input Traffic

(Optional) Header Initialization

INGRESS

Scheduling Algorithm

CORE NETWORK

Output Traffic

EGRESS
Network Model

Input Traffic

(Optional) Header Initialization

INGRESS

Scheduling Algorithm

Output Traffic tied to Scheduling Algorithm

CORE NETWORK

Output Traffic

EGRESS
Network Model

Goal: Minimize Mean FCT

Input Traffic

INGRESS

Priority Value

Flow Size

CORE NETWORK

Priority Scheduling

Output Traffic

EGRESS
Network Model

Goal: Fairness

Input Traffic

INGRESS

CORE NETWORK

FQ

Output Traffic

EGRESS
Network Model

Goal: Weighted Fairness

Input Traffic → Flow Weights → WFQ → Output Traffic

INGRESS → CORE NETWORK → EGRESS
Network Model

* Uses packet header state to make scheduling decisions

Output Traffic tied to Header Initialization

Input Traffic → Header Initialization

INGRESS

CORE NETWORK

Scheduling Algorithm*

Output Traffic → EGRESS

* Uses packet header state to make scheduling decisions
Network Model

Greater processing capability in the edge than in the core.

As per on prior SDN-based architecture designs.
How do we formally define and evaluate a UPS?
Defining a UPS

Theoretical Viewpoint: Can it replay a given schedule?

Practical Viewpoint: Can it achieve a given objective?
Theoretical Viewpoint

Can it replay a given schedule?
Original Schedule

Only requirement from original schedule:
Output Times are viable

Input Traffic

(Optional) Header Initialization

INGRESS

Arbitrary Scheduling Algorithm

CORE NETWORK

Output Times o(p) for a packet p

EGRESS
Replaying the Schedule, given $o(p)$

For every packet $p$, $o'(p) \leq o(p)$
Pragmatic Constraints on a UPS

Input Traffic

Header Initialization

INGRESS

CORE NETWORK

Obliviousness: For initializing p’s header, use only o(p) and path(p)

UPS

Output Times o’(p) for a packet p

EGRESS
Pragmatic Constraints on a UPS

Header Initialization (using $o(p)$ for a packet $p$)

Obliviousness: For initializing $p$'s header, use only $o(p)$ and $\text{path}(p)$

INGRESS \rightarrow \text{CORE NETWORK} \rightarrow \text{UPS} \rightarrow \text{EGRESS}

Input Traffic

Output Times $o'(p)$ for a packet $p$
Pragmatic Constraints on a UPS

Obliviousness: For initializing p’s header, use only o(p) and path(p)
Pragmatic Constraints on a UPS

Obliviousness: For initializing p’s header, use only o(p) and path(p)
Pragmatic Constraints on a UPS

Obliviousness: For initializing p’s header, use only o(p) and path(p)

Limited State: Scheduling can use only header state and static information

Input Traffic

Header Initialization

INGRESS

CORE NETWORK

UPS

EGRESS

Output Times o’(p) for a packet p
Pragmatic Constraints on a UPS

Input Traffic

Header Initialization

Limited State: Scheduling can use only header state and static information

Obliviousness: For initializing p's header, use only o(p) and path(p)

Exclusive Access: For initializing p's header, use o(p) and path(p)

Output Times o'(p) for a packet p

INGRESS

CORE NETWORK

EGRESS
We call this Blackbox Initialization

Limited State: Scheduling can use only header state and static information

Obliviousness: For initializing p’s header, use only o(p) and path(p)

Input Traffic

Header Initialization

INGRESS

CORE NETWORK

Output Times o’(p) for a packet p

EGRESS
Basic Existence and Non-existence
Results

There exists a UPS under *Omniscient Initialization* when scheduling time at every hop is known.

No UPS exists under *Blackbox Initialization* when only the final output time is known.
How close can we get to a UPS?
Key Result: Depends on congestion points

<table>
<thead>
<tr>
<th>No. of Congestion Points per Packet</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
</tr>
</tbody>
</table>
Can we achieve this upper bound?
Can we achieve this upper bound? Yes, LSTF!
Least Slack Time First

• Packet header initialized with a slack value
  – slack = maximum tolerable queuing delay

• At the routers
  – Schedule packet with least slack time first
  – Update the slack by subtracting the wait time
## Key Results

<table>
<thead>
<tr>
<th>No. of Congestion Points per Packet</th>
<th>General</th>
<th>LSTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Not all algorithms achieve upper bound

<table>
<thead>
<tr>
<th>No. of Congestion Points per Packet</th>
<th>General</th>
<th>LSTF</th>
<th>Priorities</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
How well does LSTF perform empirically?
Empirically, LSTF is (almost) universal

- ns-2 simulation results on realistic network settings
  - Less than 3% packets missed their output times
  - Less than 0.1% packets are late by more than one transmission time
Practical Viewpoint
Can it achieve a given objective?
Achieving various network objectives

- Slack assignment based on heuristics
- Three objective functions
  - Tail packet delays
  - Mean Flow Completion Time
  - Fairness
- We also show how LSTF can facilitate AQM from the edge.
- See NSDI’16 paper for details!
Results Summary

- Theoretical results show that
  - There is no UPS under blackbox initialization
  - LSTF comes as close to a UPS as possible
  - Empirically, LSTF is very close

- LSTF can be used in practice to achieve a variety of network-wide objectives.
Implication

- Less need for many different scheduling and queue management algorithms.
- Can just use LSTF, with varying slack initializations.
There are still some interesting open questions!
Open Questions

• What is the least amount of information needed to achieve universality?
• Are there tractable bounds for the degree of lateness with LSTF?
• What is the class of objectives that can be achieved with LSTF \textit{in practice}?
Conclusion

- Theoretical results show that
  - There is no UPS under blackbox initialization.
  - LSTF comes as close to a UPS as possible.
  - Empirically, LSTF is very close.

- LSTF can be used in practice to achieve a variety of network-wide objectives.

Contact: radhika@eecs.berkeley.edu

Code: http://netsys.github.io/ups/

Thank You!