Universal Packet Scheduling

Radhika Mittal, Rachit Agarwal,
Sylvia Ratnasamy, Scott Shenker

UC Berkeley
Packet Scheduling

• Active research literature with many
  – Algorithms
    • FIFO, DRR, virtual clocks, priorities…
  – Goals
    • fairness, small packet delay, small FCT…
  – Contexts
    • WAN, datacenters, cellular…
We are asking a new question.....

Is there a 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-wide Model

Input Traffic

INGRESS

CORE NETWORK
Network-wide Model

Input Traffic → Scheduling Algorithm

INGRESS → CORE NETWORK
Network-wide Model

Input Traffic

(Optional) Header Initialization

INGRESS

Scheduling Algorithm

CORE NETWORK

Output Traffic

EGRESS
Network-wide Model

Input Traffic

(Optional) Header Initialization

INGRESS

Scheduling Algorithm

Output Traffic tied to Scheduling Algorithm

Output Traffic

CORE NETWORK

EGRESS
Network-wide Model

Goal: Minimize Mean FCT

Input Traffic

Priority Value
Flow Size

INGRESS

Core Network

Priority Scheduling

Output Traffic

EGRESS
Network-wide Model

Goal: Fairness

Input Traffic → FQ → Output Traffic

INGRESS → CORE NETWORK → EGRESS
Network-wide Model

Goal: Weighted Fairness

Input Traffic → Flow Weights → WFQ → Output Traffic

INGRESS → CORE NETWORK → EGRESS
Network-wide Model

* Uses packet header state to make scheduling decisions

**INGRESS**

Input Traffic

Header Initialization

**CORE NETWORK**

Scheduling Algorithm*

Output Traffic tied to Header Initialization

**EGRESS**

Output Traffic

* Uses packet header state to make scheduling decisions
Network-wide Model

Input Traffic

Header Initialization

INGRESS

CORE NETWORK

UPS?

Output Traffic

EGRESS
Universality vs Programmability
Programmability:

- Single hardware for multiple algorithms to meet multiple goals

Universality:

- Single algorithm with varying header initializations for multiple goals
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

Input Traffic

(Optional) Header Initialization

INGRESS

Arbitrary Scheduling Algorithm

CORE NETWORK

Output Timings o(p) for a packet p

EGRESS

State

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

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

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

Header Initialization

Input Traffic

-ingress

CORE NETWORK

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

Output Timings o'(p) for a packet p

EGRESS
Pragmatic Constraints on a UPS

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

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

Input Traffic

Header Initialization

INGRESS

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

Core Network

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

Output Timings

o’(p) for a packet p

EGRESS
Pragmatic Constraints on a UPS

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

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

Input Traffic

Header Initialization

INGRESS

CORE NETWORK

Output Timings

\( o'(p) \) for a packet p

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)
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

Alternate EDF-based implementation:
Static deadlines in packet headers with additional state in the routers.
## 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>×</td>
</tr>
<tr>
<td>3</td>
<td>×</td>
<td>×</td>
<td>×</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
Summarizing the theoretical viewpoint

• Evaluate the ability to replay a schedule, given its final output times

• Analytical Results:
  – No UPS exists
  – LSTF comes as close to a UPS as possible

• Empirical Results: LSTF is *almost* universal!
Practical Viewpoint

Can it achieve a given objective?
Achieving various network objectives

- Slack assignment based on heuristics
- Comparison with state-of-the-art
- Three objective functions
  - Tail packet delays
  - Mean Flow Completion Time
  - Fairness
Tail Packet Delays

**Slack Assignment:** Same slack for all packets

**State-of-the-art:** FIFO, FIFO+

Smaller Tail Packet Delays with LSTF (FIFO+)
Mean Flow Completion Time

Slack Assignment: Proportional to flow size

State-of-the-art: SJF, SRPT

Mean FCT with LSTF comparable to SRPT and SJF
Fairness

Slack Assignment: Based on Virtual Clocks

State-of-the-art: Fair Queuing

Result: Eventual convergence to fairness
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 algorithms.

• Can just use LSTF, with varying initializations.
There are still a bunch of 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 in practice?
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