Buffer Management Algorithms for Packets with Heterogeneous Processing

Sergey I. Nikolenko

December 19, 2015
Outline

1. One queue with heterogeneous processing
   - Introduction
   - Lazy algorithms and their analysis

2. Other settings
   - Multiple queues and ports
   - Multiple heterogeneous characteristics
Intro

Modern network processors have to perform increasingly heterogeneous tasks.

Existing infrastructure does not always support it.

We study various settings of heterogeneous packet characteristics and compare different online algorithms with competitive analysis.

First part:

Problem setting

- A buffer $B$ that handles a sequence of arriving packets.
- Each packet $p$ has several required processing cycles $r(p) \in \{1, \ldots, k\}$, denoted by $r(p)$.
- Discrete time, each time slot contains:
  1. **arrival**: new packets arrive, and the buffer management unit performs admission control and, possibly, push-out;
  2. **assignment and processing**: a single packet is selected for processing by the scheduling module;
  3. **transmission**: packets with zero required processing left are transmitted and leave the queue.
A sample time slot
Basic definitions

- **Notation:**
  - $k$ is the maximal number of required processing cycles;
  - $B$ is the buffer size;
  - $C$ is the number of processing cores ($C = 1$ for now).

- **Common properties:** an algorithm is
  - *greedy* if it accepts all arrivals whenever there is buffer space available;
  - *preemptive* if it allows packets to push out (preempt) currently stored packets.
Basic definitions

The goal is to transmit as many packets as possible (i.e., drop as little as possible).

**Definition**

An online algorithm $A$ is said to be $\alpha$-competitive (for some $\alpha \geq 1$) if for any arrival sequence $\sigma$ the number of packets successfully transmitted by $A$ is at least $1/\alpha$ times the number of packets successfully transmitted by an optimal solution (denoted $\text{OPT}$) obtained by an offline clairvoyant algorithm.
Simple algorithms

- **Non-preemptive greedy** NPO: for an incoming packet $p$, if buffer occupancy is less than $B$ then accept $p$ else drop $p$.

- **Preemptive greedy** PO: for an incoming packet $p$,
  - if buffer occupancy is less than $B$ then accept $p$;
  - else let $q$ be the first (from HOL) packet with maximal number of residual processing; if $r_t(p) < r_t(q)$ then drop $q$ and accept $p$ according to FIFO order, else drop $p$,

- What are their competitive ratios?
Simple algorithms

- Obvious upper bound: any reasonable greedy work-conserving algorithm (even NPO) is $k$-competitive.
- Lower bound for NPO is also $k$:
  - fill NPO buffer with $k$'s;
  - keep NPO buffer full with $k$'s by adding one more every $k$ time slots;
  - at the same time, feed OPT with $1$'s (OPT does not accept all $k$'s and leaves room for $1$'s).
- This concludes our theoretical analysis of NPO.
Lower bounds for PO

- Lower bound for PO is at least \(2 \left(1 - \frac{1}{B}\right)\) for \(k \geq B\):

<table>
<thead>
<tr>
<th>(t)</th>
<th>Arriving</th>
<th>(IB_t^{{PO, LPO}})</th>
<th>#PO</th>
<th>(IB_t^{OPT})</th>
<th># OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1 1 B</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1 1 B 1 B</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1 1 B 1 B B 2</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>... ...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(B - 2)</td>
<td>1</td>
<td>1 1 B 1 B 2</td>
<td>0</td>
<td>(B - 2)</td>
<td>(B - 2)</td>
</tr>
<tr>
<td>(B - 1)</td>
<td>1 (\times) B</td>
<td>1 1 B 1 B 2 1</td>
<td>1</td>
<td>(B - 1)</td>
<td>(B - 1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(2B - 1)</td>
<td></td>
<td></td>
<td>(B)</td>
<td></td>
<td>(2B - 2)</td>
</tr>
</tbody>
</table>

- And for large \(k\) we can iterate it and get a logarithmic bound.

Theorem

*The competitive ratio of PO is at least \([\log_B k] + 1 - O\left(\frac{1}{B}\right)\).*
Lazy policies

- So the lower bound that we can show for PO is $\approx \log_B k$ – much better than $k$.
- But can we show a matching upper bound? It is far from obvious how to analyze PO.
- We do the analysis by defining a new class of algorithms – lazy processing policies.
Lazy policies

- **Lazy push-out algorithm** LPO mimics the behaviour of PO with two important differences:
  - LPO does not transmit HOL 1 if it has at least one packet with \( r > 1 \), until the buffer contains only 1s;
  - then, LPO transmits all 1s one by one, accepting new packets in the end of the queue (they cannot push out 1s).
Lazy policies

- Intuitively, LPO is a weakened version of PO since PO tends to empty its buffer faster.
- However, in the worst case they are incomparable:
  - there exists a sequence of inputs on which PO processes $\geq \frac{3}{2}$ times more packets than LPO;
  - there exists a sequence of inputs on which LPO processes $\geq \frac{5}{4}$ times more packets than PO.
Lazy policies

- Lower bounds on LPO almost exactly match lower bounds on PO:
  - the competitive ratio of LPO is at least \(2 \left(1 - \frac{1}{B}\right)\) for \(k \geq B\) and at least \(\frac{2k-1}{k}\) for \(k < B\);
  - for large \(k\), the competitive ratio of LPO is at least \(\lceil \log_B k \rceil + 1 - O\left(\frac{1}{B}\right)\).
- The difference is that for LPO, we can prove an upper bound.

**Theorem**

LPO *is at most* \((\max\{1, \ln k\} + 2 + o(1))\)-competitive.
Anatomy of an iteration

- **t\text{end}**
  - **flush** → \( \leq B \)
  - **Lem.5** → \( \leq \ln(k(B-A)/B) \)
  - → **B**

- **t\text{end} - B**
  - **Bx1** → **0**

- **t\text{con} + A**
  - **A+1**

- **t\text{con}**
  - **WA**
  - **HOL**
  - **Lem.3** → \( \leq B - 1 \)
  - → **0**

- **t\text{beg}**
  - **OPT buffer**
  - **LPO buffer**

**Introduction**
Lazy algorithms and their analysis

- **Buffer management algorithms**
Simulations: variable $\lambda_{on}$ and $k$
Simulations: variable $B$ and $C$

- $k = 3, \lambda_{on} = 0.2, C = 1$
- $k = 5, \lambda_{on} = 0.2, C = 1$
- $k = 10, \lambda_{on} = 0.2, C = 1$
- $k = 5, B = 5, \lambda_{on} = 0.2$
- $k = 10, B = 10, \lambda_{on} = 0.2$
- $k = 25, B = 50, \lambda_{on} = 0.2$
Let us now generalize a bit.

**Priority queueing** (PQ): a packet with minimal residual work is processed first.

**Reversed priority queueing** (RevPQ): a packet with maximal residual work is processed first.

FIFO with recycles (RFIFO): non-fully processed packets are recycled to the back of the queue.

And so on; we can decouple processing order from transmission order.

Lazy: a definition

Definition

A buffer processing policy LA is called lazy if it satisfies the following conditions:

(i) LA greedily accepts packets if its buffer is not full;

(ii) LA pushes out the first packet with maximal number of processing cycles in case of congestion;

(iii) LA does not process and transmit packets with a single processing cycle if its buffer contains at least one packet with more than one processing cycle left;

(iv) once all packets in LA’s buffer (say $m$ packets) have a single processing cycle remaining, LA transmits them over the next $m$ time slots, even if additional packets arrive during that time.
A general upper bound on LA

- Ideas of the LPO upper bound can be extended to a general upper bound on all lazy policies.
- Same as above, we define an iteration and it comes to a logarithmic bound ($\approx \ln k$).

**Theorem**

LA is at most $(3 + \frac{1}{B} \log_{B/(B-1)} k)$-competitive.
Lower bounds

- This upper bound is tight for some processing orders.

**Theorem**

LRFIFO, LRevPQ, and RFIFO are at least

\[
(1 + \frac{1}{B} \log B/(B-1)k)\text{-competitive}.
\]
Lower bounds

- **Proof**: denote \( \gamma = \frac{B - 1}{B} \).
- **First burst**: \((B - 1) \times k\) packets arrive followed by \(\gamma k\); OPT drops all \(k\)’s and only leaves \(\gamma k\).
- After \(\gamma k\) steps, all three policies will have \(B \times \gamma k\) in the buffer, and then \(\gamma^2 k\) arrives.
- Repeat this sequence (\(\gamma^{i+1} k\) arrives after \(\gamma^i\) more steps) until IB\(^{ALG}\) consists of 1’s.
- We get that OPT has processed \(\log \frac{1}{\gamma} k = \log \frac{B}{B - 1} k\) packets while LRevPQ (LRFIFO) has processed none.
- Then we flush out with a new burst of \(B \times 1\).
Theorem

(i) LPQ is at most $2$-competitive.

(ii) LPQ is at least $\left(2 - \frac{1}{B} \left\lceil \frac{B}{k} \right\rceil\right)$-competitive.

(i) Since PQ is optimal, during an iteration OPT cannot transmit more packets than reside in the LPQ buffer at the end of an iteration. By a previous Lemma, LPQ is at most $2$-competitive.

(ii) For $k \geq B$, consider two bursts of packets: $B \times \lfloor k \rfloor$ and then, in $(k - 1)B$ steps, $(B - 1) \times 1$ each. After these two bursts, OPT has processed $2B - 1$ packets, and LPQ has processed $B$ packets, so we can repeat them to get the asymptotic bound. For $k < B$, in the same construction $\left\lceil \frac{B}{k} \right\rceil$ packets are left in OPT's queue after $(k - 1)B$ processing steps.
Other extensions

**Theorem**

Any greedy Semi-FIFO policy is at least \((1 + \frac{m-1}{B})\)-competitive for \(m = \min\{k,B\}\).

**Theorem**

Any lazy policy LA (including LRevPQ) is incomparable with either FIFO or RFIFO in the worst case for every \(k > 2\) and \(B > 2\).

**Theorem**

Any greedy non-push-out Semi-FIFO policy NPO is at least \(\frac{k+1}{2}\)-competitive. Any lazy greedy non-push-out policy NLPO is at least \((k - 1)\)-competitive.
Constraints on push-out

- In some situations, we’d like to impose constraints on push-out; e.g., there might be copying cost $\alpha$ for each admitted packet.
- We introduce an additional constraint $\beta$: a policy $\text{ALG}_\beta$ pushes out only if the new arrival has at least $\beta$ times less work than the maximal residual work in the buffer.
Upper bound with $\beta$-preemption

- The key lemma will now have $W_{te} \leq W_{t-1} - \frac{M_{t-1}}{\beta}$.

**Theorem**

$LA_\beta$ is at most 
\[
\left(3 + \frac{1}{B} \log \frac{\beta B}{\beta B - 1} \right) \frac{1 - \alpha}{1 - \alpha \log \beta k}
\]-competitive for copying cost $0 < \alpha < \frac{1}{\log \beta k}$.
# Results summary

<table>
<thead>
<tr>
<th>Algorithm/family</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-FIFO</td>
<td>$1 + \frac{\min{k,B} - 1}{B}$</td>
<td>open problem</td>
</tr>
<tr>
<td>Lazy</td>
<td>$1 + \frac{\min{k,B} - 1}{B}$</td>
<td>$3 + \frac{1}{B} \log \frac{B}{B - 1} k$</td>
</tr>
<tr>
<td>LRFIFO, LRevFIFO</td>
<td>$1 + \frac{1}{B} \log \frac{B}{B - 1} k$</td>
<td>$3 + \frac{1}{B} \log \frac{B}{B - 1} k$</td>
</tr>
<tr>
<td>LPO</td>
<td>$\left\lfloor \log_B k \right\rfloor + 1$</td>
<td>$\max{1, \ln k} + 2$</td>
</tr>
<tr>
<td>LPQ</td>
<td>$2 - \frac{1}{B} \left\lceil \frac{B}{k} \right\rceil$</td>
<td>$k$</td>
</tr>
<tr>
<td>2LFIFO</td>
<td>$k - 1 + \frac{1}{B} \left\lfloor \frac{B}{k} \right\rfloor$</td>
<td>$k$</td>
</tr>
<tr>
<td>Lazy (\beta)-push-out</td>
<td>$1 + \frac{\min{k,B} - 1}{B}$</td>
<td>$\left( 3 + \frac{1}{B} \log \frac{\beta B}{\beta B - 1} k \right) \frac{1-\alpha}{1-\alpha \log_\beta k}$</td>
</tr>
<tr>
<td>Non-push-out</td>
<td>$\frac{k + 1}{2}$</td>
<td>$k$</td>
</tr>
<tr>
<td>Lazy non-push-out</td>
<td>$k - 1$</td>
<td>$k$</td>
</tr>
</tbody>
</table>
Outline

1. One queue with heterogeneous processing
   - Introduction
   - Lazy algorithms and their analysis

2. Other settings
   - Multiple queues and ports
   - Multiple heterogeneous characteristics
Multiple shared queues

Now for a brief review of our other results in this direction.

First: multiple separate queues with a shared buffer.

Multiple shared queues

- Let’s divide the buffer into $k$ queues and send each packet to the corresponding queue.

- Fairness properties become trivial, and there’s no need to implement priority queueing, move packets around a queue.
Multiple shared queues

- Reasonable policies:
  - LQF (longest queue first);
  - SQF (shortest queue first);
  - MQF (minimal queue first);
  - fair policies: PRR (packet round robin), CRR (cycle round robin).
Multiple shared queues

- Lower bounds:
  - LQF is at least $\frac{m}{2}$-competitive, $m = \min\{k, B\}$;
  - SQF is at least $k$-competitive;
  - MQF is at least $(1 + \frac{k-1}{2k})$-competitive.

**Theorem**

*MQF is at most 2-competitive.*
Multiple shared queues

- Important special case: what if there are only two kinds of required processing, $a$ and $b$.

**Theorem**

1. The competitiveness of MQF with two queues is at least

$$\left(1 + \frac{1 + \left\lceil \frac{aB-1}{b} \right\rceil}{B + \left\lceil \frac{1}{a} \left( b \left\lfloor \frac{aB-1}{b} \right\rfloor + 1 \right) \right\rceil} \right).$$

2. The competitiveness of MQF with two queues is at most

$$\left(1 + \frac{1 + \left\lceil \frac{aB-1}{b} \right\rceil}{B + \left\lceil \frac{1}{a} \left( b \left\lfloor \frac{aB-1}{b} \right\rfloor + 1 \right) \right\rceil} \right).$$
Next setting: let us now have multiple output ports, each packet is labeled with an output port, and packets with the same output port share properties (like required processing).

Multiple output ports

- First setting: $l \times n$ memory switch with shared buffer of size $B$, each FIFO queue $Q_i$ has packets with the same processing requirement $r_i$ (max $k$).
Multiple output ports

- Lower bounds:
  - NHST is at least $kZ$-competitive, $Z = \sum \frac{1}{r_i}$;
  - NEST is at least $n$-competitive;
  - NHDT is at least $\frac{1}{2}\sqrt{k \ln k}$-competitive;
  - LQD is at least $\sqrt{k}$-competitive;
  - BPD is at least $(\ln k + \gamma)$-competitive;
  - LWD is at least $(\frac{4}{3} - \frac{6}{B})$-competitive.

**Theorem**

*LWD is at most 2-competitive.*
Multiple characteristics

- Next setting: what if the packets have multiple different characteristics?
- Say, required processing and value.
- We restrict ourselves to a single queue, it’s hard enough already.
Multiple characteristics

- This time, it makes sense to concentrate on priority queues since there are many different versions.

<table>
<thead>
<tr>
<th>arrivals</th>
<th>$PQ_{w,v}$</th>
<th>$PQ_{v,-w}$</th>
<th>$PQ_{v/w}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>52 43 11</td>
<td>11 52 43</td>
<td>52 43 11</td>
</tr>
<tr>
<td></td>
<td>52 43 11</td>
<td>52 63 43</td>
<td>63 43 11</td>
</tr>
<tr>
<td></td>
<td>52 43 01</td>
<td>52 63 33</td>
<td>63 43 01</td>
</tr>
<tr>
<td></td>
<td>52 43</td>
<td>52 63 33</td>
<td>63 43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Multiple characteristics

- Lower bounds:
  - \(\text{PQ}_{w,v}\) is at least \(V\)-competitive and at most \(V\)-competitive;
  - \(\text{PQ}_{v,-w}\) is at least \(\left(\frac{V-1}{V}\right)W - o(1)\)-competitive;
  - \(\text{PQ}_{v/w}\) is at least \(\min(V, W)\)-competitive.

- General lower bound:

**Theorem**

*Every online deterministic algorithm ALG is at least \((\frac{5}{4} - O(1/W))\)-competitive.*
Multiple characteristics

- Two-valued case: what if required processing can be arbitrary, but there are only two values, 1 and $V$? Lower bounds:
  - $\text{PQ}_{-w,v}$ is at least $V$-competitive;
  - if $W \geq V$ then $\text{PQ}_{v/w}$ is at least $V$-competitive;
  - $\text{PQ}_{v,-w}$ is at least \((\frac{W}{V} + o(1))\)-competitive.

- General lower bound:

Theorem

*Every online deterministic algorithm ALG is at least* \((1 + \frac{V - 1}{V^2} - O\left(\frac{1}{W}\right))\)-competitive.

- Upper bound:

Theorem

$\text{PQ}_{v,-w}$ is at most \((1 + \frac{W+2}{V})\)-competitive.
Summary of our results:

<table>
<thead>
<tr>
<th>Processing policy</th>
<th>General case</th>
<th>Two-valued case</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lower bound</td>
<td>Lower bound</td>
</tr>
<tr>
<td>Any</td>
<td>5/4</td>
<td>$1 + \frac{V-1}{V^2} - O\left(\frac{1}{W}\right)$</td>
</tr>
<tr>
<td>$PQ_{-w,v}, PQ_{-w,v}^\beta$</td>
<td>$V$</td>
<td>$V$</td>
</tr>
<tr>
<td>$PQ_{v,-w}, PQ_{v,-w}^\beta$</td>
<td>$\frac{(V-1)}{V} W - o(1)$</td>
<td>$\frac{W}{V} + o(1)$</td>
</tr>
<tr>
<td>$PQ_{v/w}, PQ_{v/w}^\beta W \geq V$</td>
<td>$V$</td>
<td>$V$</td>
</tr>
<tr>
<td>$PQ_{v/w}, W &lt; V$</td>
<td>$W$</td>
<td>$\frac{W}{V} + o(1)$</td>
</tr>
</tbody>
</table>

$PQ \beta v, w, \beta W \geq V$

$PQ \beta v, w, \beta W < V$

$Sergey I. Nikolenko$  

Buffer management algorithms
In modern networking, heterogeneous characteristics (required processing, value, size etc.) abound.

This leads to new challenges in buffer management.

Worst-case competitive upper bounds are important since traffic distributions can be uneven and unpredictable.

We consider buffer management policies in many different contexts, proving bounds on their competitiveness:

- a single queue, where we have introduced lazy algorithms to prove upper bounds;
- buffer with multiple shared queues;
- buffer with a separate queue for each output port;
- queue with packets with multiple heterogeneous characteristics.

There are many more important contexts and settings to come.
Thank you for your attention!
Lower bounds for PO

Lower bound for PO is at least $2 \left( 1 - \frac{1}{B} \right)$ for $k \geq B$:

<table>
<thead>
<tr>
<th>$t$</th>
<th>Arriving</th>
<th>$IB^{PO, LPO}_t$</th>
<th>#PO</th>
<th>$IB^{OPT}_t$</th>
<th># OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 $B$</td>
<td>1 1 $B$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1 1 $B - 1$</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1 1 1 $B - 2$</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$B - 2$</td>
<td>1</td>
<td>1 ... 1 2</td>
<td>0</td>
<td>1</td>
<td>$B - 2$</td>
</tr>
<tr>
<td>$B - 1$</td>
<td>1 $\times B$</td>
<td>1 ... 1 1</td>
<td>1</td>
<td>1 ... 1 1</td>
<td>$B - 1$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$2B - 1$</td>
<td></td>
<td></td>
<td>$B$</td>
<td></td>
<td>$2B - 2$</td>
</tr>
</tbody>
</table>
Lower bounds for PO

- Lower bound for PO is at least \( \frac{2k}{k+1} \) for \( k < B \):
  - on step 1, there arrive \((1 - \alpha)B \times k\) followed by \(\alpha B \times 1\); PO accepts all, OPT rejects \(k\) s.
  - on step \(\alpha B\), there arrive \(\frac{\alpha B}{k} \times 1\); on step \(\alpha B(1 + \frac{1}{k})\), \(\frac{\alpha B}{k^2} \times 1\) and so on;
  - when PO is out of packets with \(k\) processing cycles, its queue is full \(1\) s, and OPT’s queue is empty; now, there arrive \(B \times 1\), they are processed, and the sequence is repeated.

- In order for this sequence to work, we need to have \(\alpha B \left(1 + \frac{1}{k} + \frac{1}{k^2} + \ldots\right) = k \left(1 - \alpha\right) B\), so we get \(\alpha = 1 - \frac{1}{k}\).

- During the sequence, OPT has processed \(\alpha B \left(1 + \frac{1}{k} + \frac{1}{k^2} + \ldots\right) + B = 2B\) packets, while PO has processed \((1 - \alpha)B + B = (1 + \frac{1}{k})B\) packets, so the competitive ratio is \(\frac{2}{1 + \frac{1}{k}}\).
Lower bounds for PO

- For large values of $k$, we can have a logarithmic lower bound. First step: suppose $k \geq (B - 1)(B - 2)$. Then:
  - we begin with buffer state

  \[
  \begin{array}{cccc}
  1 & 2 & 3 & 4 \\
  & & B-1 & (B-1)(B-2)
  \end{array}
  \]

  - OPT drops first packet and processes the rest while PO keeps processing the first;
  - then, for $B$ steps one 1 per step arrives; PO keeps dropping its HOL;
  - then PO has a queue of $1$s, so we flush it out with $B \times 1$.

- At the end of this iteration, PO has processed $B + 1$ packets; OPT, $3B$ packets.
Lower bounds for PO

- We can iterate this construction for larger values of $k$: having proven for $S = \Omega(B^{n-1})$, on the next step we begin with

\[
1 + S \quad 2 + S \quad 3 + S \quad 4 + S \quad \ldots \quad B - 1 + S \quad (B - 1)(B - 2 + S).
\]

**Theorem**

The competitive ratio of PO is at least $\lceil \log_B k \rceil + 1 - O\left(\frac{1}{B}\right)$. 
Upper bound on the competitiveness of LPO

Idea – we define an iteration:
- the first iteration begins with the first arrival;
- an iteration ends when all packets in the LPO buffer have a single processing pass left;
- each subsequent iteration starts after the transmission of all LPO packets from the previous iteration.

The plan is to count how many packets LPO can lose to OPT on each iteration.
Upper bound on the competitiveness of LPO

- Wlog, OPT never pushes out packets and it is work-conserving.
- Further, we give OPT an additional property for free:
  1. at the start of each iteration, OPT flushes out all packets remaining in its buffer from the previous iteration (for free, with extra gain to its throughput).
- Notation:
  - $A$, number of non-HOL packets in OPT’s buffer at time $t_{\text{con}}$;
  - $W_A$, their total required processing;
  - $M_t$, maximal number of residual processing cycles among all packets in LPO’s buffer at time $t$ in current iteration;
  - $W_t$, total residual work for all packets in LPO’s buffer at time $t$. 
Consider an iteration \( I \) that begins at time \( t_{\text{beg}} \) and ends at time \( t_{\text{end}} \); \( t_{\text{con}} \) is the time when LPO buffer is first congested. The following statements hold:

1. during \( I \), the buffer occupancy of LPO is at least the buffer occupancy of OPT;
2. if during a time interval \([t, t']\), \( t_{\text{beg}} \leq t \leq t' \leq t_{\text{con}} \), there is no congestion in LPO’s buffer then during \([t, t']\) OPT transmits at most \(|IB_{t'}^{\text{LPO}}|\) packets and LPO does not transmit any packets.
Upper bound on the competitiveness of LPO

Lemma

1. During $[t_{\text{beg}}, t_{\text{con}}]$, OPT processes at most $B - 1$ packets.
2. For every packet $p$ in OPT’s buffer at time $t_{\text{con}}$ except perhaps the HOL packet, there is a corresponding packet $q$ in LPO’s buffer with $r(q) \leq r(p)$.

Proof.

1. During $[t_{\text{beg}}, t_{\text{con}}]$, there arrive exactly $B$ packets (because LPO does not transmit any packets and becomes congested at $t_{\text{con}}$). Moreover, OPT cannot process all $B$ packets because then LPO would also have time to process them, and the iteration would be uncongested.
2. Every packet in OPT’s buffer also resides in LPO’s buffer because LPO has not dropped anything yet at time $t_{\text{con}}$; $r(q) \leq r(p)$ because LPO may have processed some packets partially.
Upper bound on the competitiveness of LPO

- By prev. Lemma, LPO buffer at time $t_{con}$ contains $A$ corresponding packets, so $W_{t_{con}} \leq W_A + (B - A)k$.

- Moreover, over the next $W_A$ time slots OPT will be processing these $A$ packets and LPO, being congested, will also not be idle, so at time $t_{con} + A$ we will have $W_{t_{con}+A} \leq (B - A)k$ (we give OPT its HOL packet for free, so OPT processes $A + 1$ packets over $[t_{con}, t_{con} + A]$).
Upper bound on the competitiveness of LPO

**Lemma**

For every packet accepted by OPT at time $t \in [t_{\text{con}}, t_{\text{end}}]$ and processed by OPT during time interval $[t', t'']$, $t_{\text{con}} \leq t' \leq t'' \leq t_{\text{end}}$, $W_{t''} \leq W_{t-1} - M_t$.

**Proof.**

If LPO’s buffer is full then a packet $p$ accepted by OPT either pushes out a packet in LPO’s buffer or is rejected by LPO. If $p$ pushes a packet out, then the total work $W_{t-1}$ is immediately reduced by $M_t - r_t(p)$. Moreover, after processing $p$, $W_{t''} \leq W_{t-1} - (M_t - r_t(p)) - r_t(p) = W_{t-1} - M_t$. If, on the other hand, $p$ is rejected by LPO then $r_t(p) \geq M_t$, and thus $W_{t''} \leq W_{t-1} - r_t(p) \leq W_{t-1} - M_t$. 

Sergey I. Nikolenko

Buffer management algorithms
Upper bound on the competitiveness of LPO

We denote by $f(B,W)$ the maximal number of packets that OPT can accept and process during $[t,t']$, $t_{con} \leq t \leq t' \leq t_{end}$, where $W = W_{t-1}$. The next lemma is crucial for the proof.

**Lemma**

*For every $\epsilon > 0$, $f(B,W) \leq \frac{B-1}{1-\epsilon} \ln \frac{W}{B}$ for $B$ sufficiently large.*

- Proof: all packets LPO transmits it does at the end of an iteration, hence, if the buffer of LPO is full, it will remain full until $t_{end} - B$.
- At any time $t$, $M_t \geq \frac{W_t}{B}$: the maximal required processing is no less than the average.
Upper bound on the competitiveness of LPO

- We know that for every packet $p$ accepted by OPT at time $t$, the total work $W = W_{t-1}$ is reduced by $M_t$ after OPT has processed $p$.

- Therefore, after OPT processes a packet at time $t'$, $W_{t'}$ is at most $W \left( 1 - \frac{1}{B} \right)$.

- Now by induction on $W$; for $W = B$ the base is trivial.
Upper bound on the competitiveness of LPO

- The induction hypothesis is that after a packet is processed by OPT, there cannot be more than

\[ f(B, \frac{W}{B} \left(1 - \frac{1}{B}\right)) \leq \frac{B - 1}{1 - \epsilon} \ln \left[ \frac{W}{B} \left(1 - \frac{1}{B}\right) \right] \]

packets left, and for the induction step we have to prove that

\[ \frac{B - 1}{1 - \epsilon} \ln \left[ \frac{W}{B} \left(1 - \frac{1}{B}\right) \right] + 1 \leq \frac{B - 1}{1 - \epsilon} \ln \frac{W}{B}. \]

- This is equivalent to

\[ \ln \frac{W}{B} \geq \ln \left[ \frac{W}{B} \frac{B - 1}{B} e^{\frac{1 - \epsilon}{B - 1}} \right], \]

and this holds asymptotically because for every \( \epsilon > 0 \), we have \( e^{\frac{1 - \epsilon}{B - 1}} \leq \frac{B}{B - 1} \) for \( B \) sufficiently large.
Upper bound on the competitiveness of LPO

- Applying Lemma 16 to the time $t_{\text{con}} + A$, we get the following.

**Corollary**

For every $\epsilon > 0$, the total number of packets processed by OPT between $t_{\text{con}}$ and $t_{\text{end}}$ in a congested iteration does not exceed

$$A + 1 + (B + o(B)) \ln \frac{(B - A)k}{B}.$$

- And the final result is as follows.

**Theorem**

LPO is at most $(\max\{1, \ln k\} + 2 + o(1))$-competitive.
Anatomy of an iteration

One queue with heterogeneous processing
Multiple queues and ports
Other settings
Multiple heterogeneous characteristics

Time | OPT buffer | LPO buffer
--- | --- | ---
$t_{begin}$ | | |
$t_{con}$ | $W_A$ | $A$ |
$t_{con}+A$ | | $W \leq W_A + k(B-A)$ |
$t_{end}$ | $\leq B-1$ | $0$ |
$t_{end}-B$ | $\leq \ln(k(B-A)/B)$ | $B \times 1$ |
flush | $\leq B$ | $B$ |

Sergey I. Nikolenko
Buffer management algorithms
Upper bound on the competitiveness of LPO

- Consider an iteration $I$ over time $[t_{\text{beg}}, t_{\text{end}}]$.
- If $I$ is uncongested then OPT cannot transmit more than $|IB_t^{LPO}|$ packets during $I$.
- Consider an iteration $I$ first congested at time $t_{\text{con}}$:
  - by a lemma, during $[t_{\text{beg}}, t_{\text{con}})$ OPT can transmit at most $B - 1$ packets, leaving $A + 1$ packets in its buffer;
  - by the corollary, OPT processes at most $A + 1 + \frac{B-1}{1-\epsilon} \ln \frac{(B-A)k}{B} + o(B \ln \frac{(B-A)k}{B})$ packets during $[t_{\text{con}}, t_{\text{end}}]$ and flushes out $\leq B$ packets at time $t_{\text{end}}$;
  - thus, the total number of packets transmitted by OPT over a congested iteration is at most
    $$2B + A + (B + o(B)) \ln \frac{(B-A)k}{B}.$$  
- It is now easy to check that for every $1 \leq A \leq B - 1$ the theorem’s statement is satisfied.
Ideas of the LPO upper bound can be extended to a general upper bound on all lazy policies.

**Lemma**

Consider an iteration $I$ that has started at time $t'$ and ended at time $t$. The following statements hold.

1. During $I$, the buffer occupancy of LA is at least the buffer occupancy of OPT.

2. Between two consecutive iterations $I$ and $I'$, OPT transmits at most $|IB^L_{t'}|$ packets.

3. If during an interval of time $[t', t'']$, $t' \leq t'' \leq t$, there is no congestion, then during $[t', t'']$ OPT transmits at most $|IB^L_{t''}|$ packets.
A general upper bound on LA

- Same as above.

**Lemma**

For any packet accepted by OPT at time $t$ and processed by OPT during $[t_s, t_e]$, $t \leq t_s \leq t_e$, if $|IB_{t-1}^{LA}| = B$ and $|IB_{t-1}^{OPT}| = 0$ then $W_{t_e} \leq W_{t-1} - M_{t-1}$. 
A general upper bound on LA

- And this comes to a logarithmic bound ($\approx \ln k$).

**Theorem**

LA is at most $(3 + \frac{1}{B} \log_{B/(B-1)} k)$-competitive.
In a congested iteration, any packet processed by OPT decreases the total LA work by $M_t$, i.e., by at least $W/B$.

After $n$ transmission rounds, the residual number of processing cycles in LA buffer is $W(1 - 1/B)^n$.

Since initially $W \leq kB$, $n \leq \log_{B/(B-1)} k$. 
Lower bounds

This upper bound is tight for some processing orders.

Theorem

LRFIFO, LRevPQ, and RFIFO are at least $(1 + \frac{1}{B} \log_{B/(B-1)} k)$-competitive.
Lower bounds

- Proof: denote $\gamma = \frac{B-1}{B}$.

- First burst: $(B - 1) \times k$ packets arrive followed by $\gamma k$; OPT drops all $k$’s and only leaves $\gamma k$.

- After $\gamma k$ steps, all three policies will have $B \times \gamma k$ in the buffer, and then $\gamma^2 k$ arrives.

- Repeat this sequence ($\gamma^{i+1} k$ arrives after $\gamma^i$ more steps) until $1B^{\text{ALG}}$ consists of 1’s.

- We get that OPT has processed $\log_k \frac{1}{\gamma} = \log \frac{B}{B-1} k$ packets while LRevPQ (LRFIFO) has processed none.

- Then we flush out with a new burst of $B \times 1$. 

Sergey I. Nikolenko  Buffer management algorithms