FIFO Queueing Policies for Packets with Heterogeneous Processing

Kirill Kogan Alejandro López-Ortiz Sergey I. Nikolenko
Alexander V. Sirotkin Denis Tugaryov

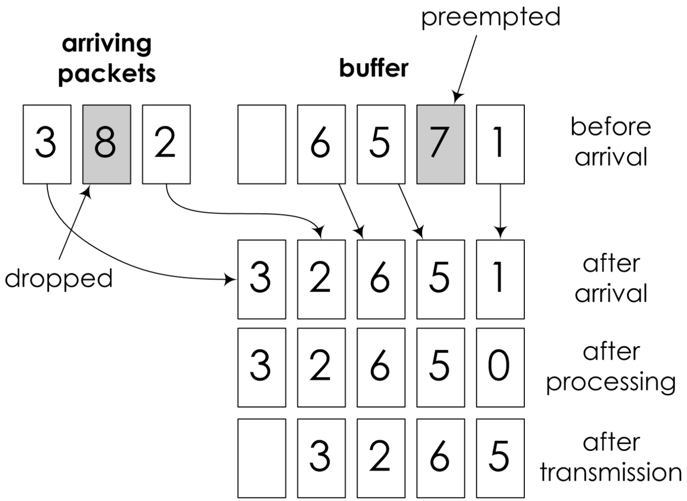
December 3, 2012

Outline

Problem setting

- A limited buffer B that handles a sequence of arriving packets (no deadlines/delays, but some packets get dropped).
- Each packet p has several required processing cycles $r(p) \in \{1, \ldots, k\}$, denoted by $\boxed{r(p)}$.
- 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 unit;
  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).
- Natural properties: an algorithm is
  - *greedy* if it accepts all arrivals whenever there is buffer space available;
  - *work-conserving* if it always processes whenever it has admitted packets that require processing in the queue;
  - *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 *α-competitive* (for some $\alpha \geq 1$) if for any arrival sequence σ 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 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 $\boxed{k}$s;
  - keep NPO buffer full with $\boxed{k}$s by adding one more every k time slots;
  - at the same time, feed OPT with $\boxed{1}$s (OPT does not accept all $\boxed{k}$s and leaves room for $\boxed{1}$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 $k \geq B$:

t	Arriving	$\mathrm{IB}_t^{\{\mathrm{PO},\,\mathrm{LPO}\}}$	#PO	$\mathrm{IB}_t^{\mathrm{OPT}}$	# OPT
1	$\boxed{1}\ \boxed{B}$	$\boxed{1}\ \boxed{B}$	0	$\boxed{1}$	1
2	$\boxed{1}$	$\boxed{1}\ \boxed{1}\ \boxed{B-1}$	0	$\boxed{1}$	2
3	$\boxed{1}$	$\boxed{1}\ \boxed{1}\ \boxed{1}\ \boxed{B-2}$	0	$\boxed{1}$	3
...		...		...	
$B-2$	$\boxed{1}$	$\boxed{1}\ \dots\ \boxed{1}\ \boxed{2}$	0	$\boxed{1}$	$B-2$
$B-1$	$\boxed{1} \times B$	$\boxed{1}\ \dots\ \boxed{1}\ \boxed{1}$	1	$\boxed{1}\ \dots\ \boxed{1}\ \boxed{1}$	$B-1$
...		...		...	
$2B-1$			B		$2B-2$

- Lower bound for PO is at least $\frac{2k}{k+1}$ for $k < B$ (a bit more technical).

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

$$\boxed{1}\ \boxed{2}\ \boxed{3}\ \boxed{4}\ \ldots\ \boxed{B-1}\ \boxed{(B-1)(B-2)}.$$

  - OPT drops first packet and processes the rest while PO keeps processing the first;
  - then, for B steps one $\boxed{1}$ per step arrives; PO keeps dropping its HOL (one of two $\boxed{B-1}$s, then one of two $\boxed{B-2}$s etc.);
  - then PO has a queue of $\boxed{1}$s, so we flush it out with $B \times \boxed{1}$.
- 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

$$\boxed{1+S}\,\boxed{2+S}\,\boxed{3+S}\,\boxed{4+S}\ \ldots\ \boxed{B-1+S}\,\boxed{(B-1)(B-2+S)}.$$

Theorem

The competitive ratio of PO *is at least* $\lfloor \log_B k \rfloor + 1 - O(\frac{1}{B})$.

Outline

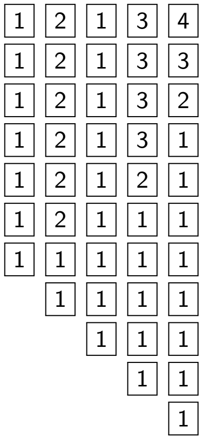
Lazy policies

- 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 $\boxed{1}$ if it has at least one packet with $r > 1$, until the buffer contains only $\boxed{1}$s;
  - then, LPO transmits all $\boxed{1}$s one by one, accepting new packets in the end of the queue (they cannot push out $\boxed{1}$s).

For example:



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.

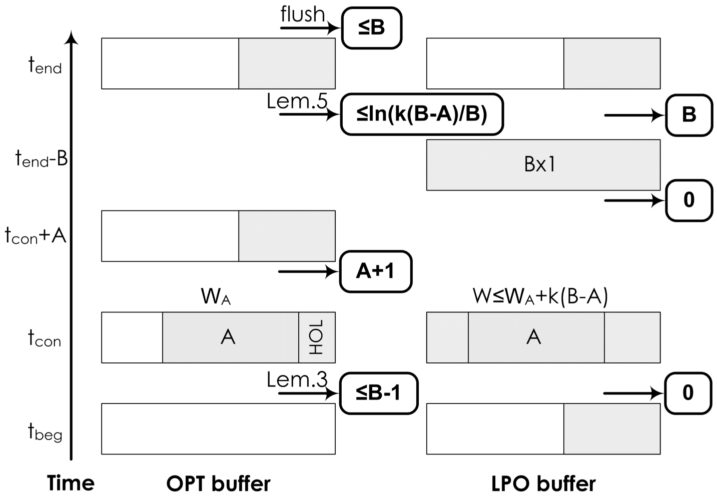
Lazy policies

- 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 $\lfloor \log_B k \rfloor + 1 - O(\frac{1}{B})$.
- The difference is that for LPO, we can prove an upper bound.

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.
- Wlog, OPT never pushes out packets and it is work-conserving.
- Further, we give OPT an additional property for free: 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).

Anatomy of an iteration



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 a time interval* $[t', t''] \subseteq [t_{\text{con}}, t_{\text{end}}]$, $W_{t''} \leq W_{t-1} - M_t$, *where* M_t *is the maximal required processing in* LPO *buffer.*

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 $W_{t''} \leq W_{t-1} - r_t(p) \leq W_{t-1} - M_t$. □

Final result

- And the final result is as follows.

Theorem

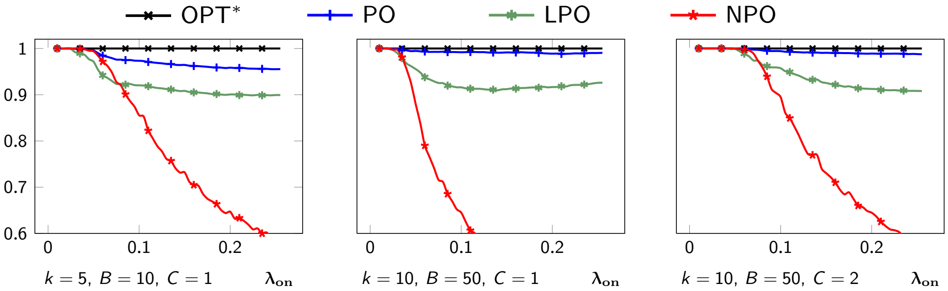
LPO *is at most* $(\max\{1, \ln k\} + 2 + o(1))$*-competitive.*

- Summary:

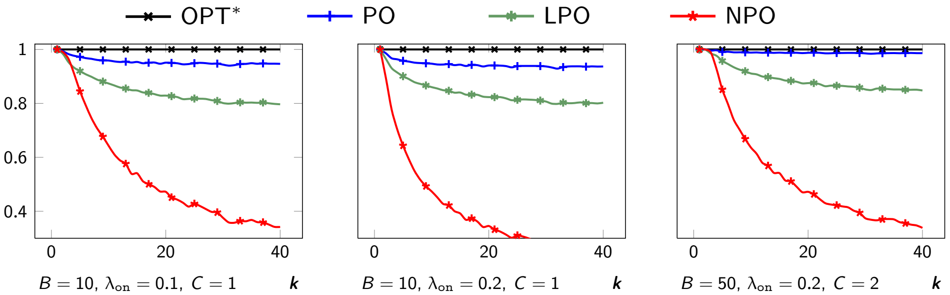
Algorithm	Lower bound	Upper bound
NPO	k	k
LPO	$\lfloor \log_B k \rfloor + 1$	$\max\{1, \ln k\} + 2$
PO	$\lfloor \log_B k \rfloor + 1$	Open problem

- Extensions: LPQ and semi-FIFO policies in general, copying cost.

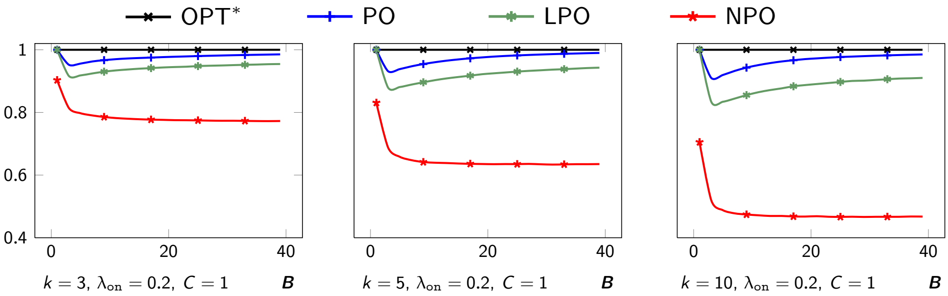
Simulations: variable λ_{on}



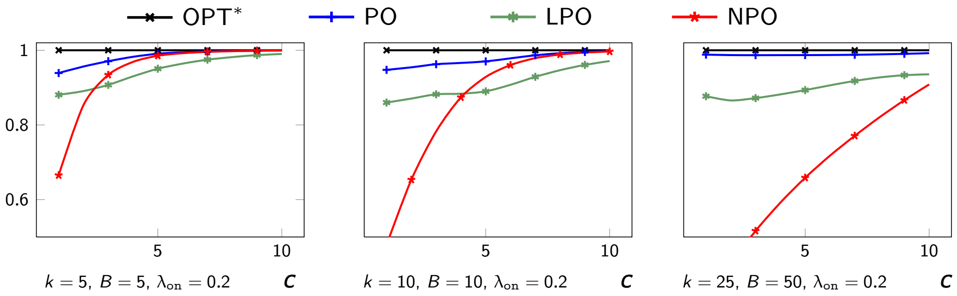
Simulations: variable max processing



Simulations: variable buffer



Simulations: variable # of cores



Thank you!

Thank you for your attention!

Secret slides

- Lower bound for PO is at least $\frac{2k}{k+1}$ for $k < B$:
  - on step 1, there arrive $(1-\alpha)B \times \boxed{k}$ followed by $\alpha B \times \boxed{1}$; PO accepts all, OPT rejects $\boxed{k}$s.
  - on step αB, there arrive $\frac{\alpha B}{k} \times \boxed{1}$; on step $\alpha B(1+\frac{1}{k})$, $\frac{\alpha B}{k^2} \times \boxed{1}$ and so on;
  - when PO is out of packets with k processing cycles, its queue is full $\boxed{1}$s, and OPT's queue is empty; now, there arrive $B \times \boxed{1}$, they are processed, and the sequence is repeated.
- In order for this sequence to work, we need to have $\alpha B\left(1+\frac{1}{k}+\frac{1}{k^2}+\ldots\right) = k(1-\alpha)B$, 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 $(1-\alpha)B+B = \left(1+\frac{1}{k}\right)B$ packets, so the competitive ratio is $\frac{2}{1+\frac{1}{k}}$.

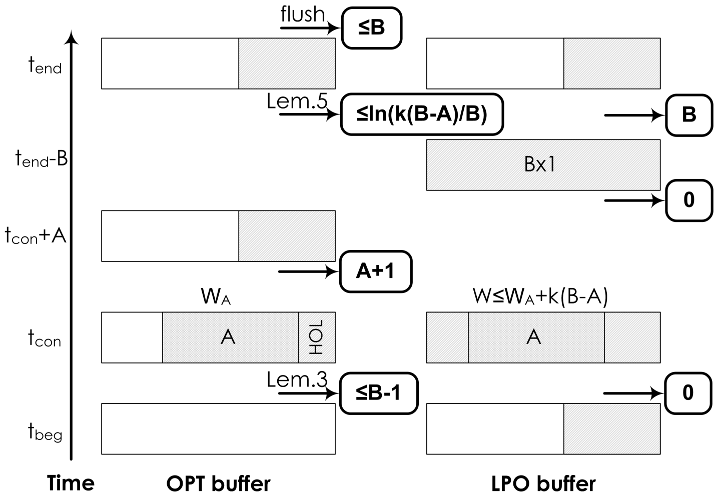
Upper bound on the competitiveness of LPO

- 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_{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.

Anatomy of an iteration



Upper bound on the competitiveness of LPO

- Consider an iteration I that begins at time t_{beg} and ends at time t_{end}; t_{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 $|\text{IB}_{t'}^{\text{LPO}}|$ 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_{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_{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_{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 $W_{t_{\text{con}}} \leq W_A + (B - A)k$.
- 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 $t_{\text{con}} + A$ we will have $W_{t_{\text{con}}+A} \leq (B - A)k$ (we give OPT its HOL packet for free, so OPT processes $A + 1$ packets over $[t_{\text{con}}, t_{\text{con}} + A]$).

Upper bound on the competitiveness of LPO

Lemma

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

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, $W_{t''} \le W_{t-1} - (M_t - r_t(p)) - r_t(p) = W_{t-1} - M_t$. If, on the other hand, p is rejected by LPO then $r_t(p) \ge M_t$, and thus $W_{t''} \le W_{t-1} - r_t(p) \le W_{t-1} - M_t$. □

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 $[t,t']$, $t_{\text{con}} \leq t \leq t' \leq t_{\text{end}}$, 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 $t_{\text{end}} - B$.
- 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 7 to the time $t_{\mathrm{con}} + A$, we get the following.

Corollary

For every $\epsilon > 0$, the total number of packets processed by OPT *between t_{con} and t_{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.*

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 $|\text{IB}_t^{\text{LPO}}|$ packets during I.
- Consider an iteration I first congested at time t_{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_{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.