Online Packet Scheduling with Bounded Delay and Lookahead

Martin Böhm[1], Marek Chrobak[2], Łukasz Jeż[3], Fei Li[4], Jiří Sgall[1], **Pavel Veselý**[1]

[1]Charles University, Prague, Czech Republic.

[2]University of California, Riverside, USA.

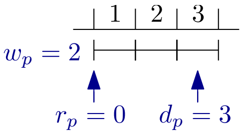
[3]University of Wrocław, Poland.

[4]George Mason University, USA.

MAPSP 2017, June 12

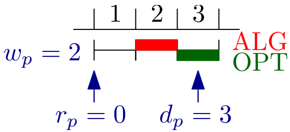
Online Packet Scheduling

- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots



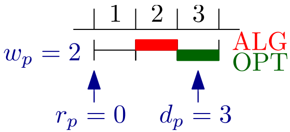
Online Packet Scheduling

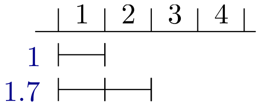
- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots

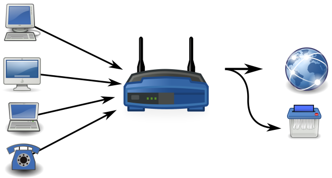


ONLINE PACKET SCHEDULING

- Also called BUFFER MANAGEMENT IN QUALITY OF SERVICE SWITCHES
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets

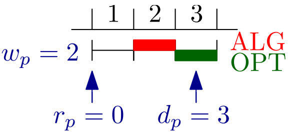


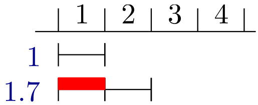


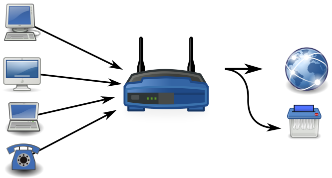


Online Packet Scheduling

- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets

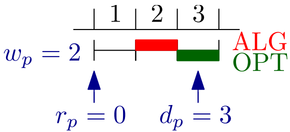


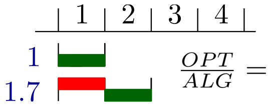




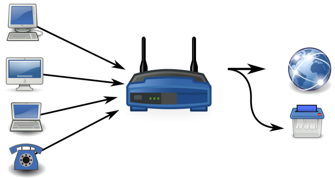
Online Packet Scheduling

- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets



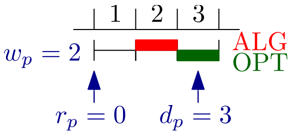


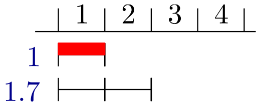
$$\frac{OPT}{ALG} = \frac{1.7+1}{1.7} \approx 1.59$$

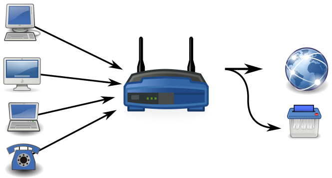


ONLINE PACKET SCHEDULING

- Also called BUFFER MANAGEMENT IN QUALITY OF SERVICE SWITCHES
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets

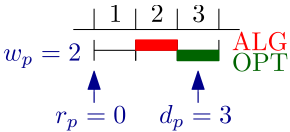


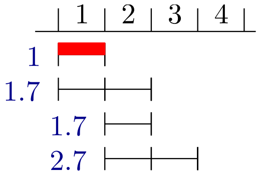


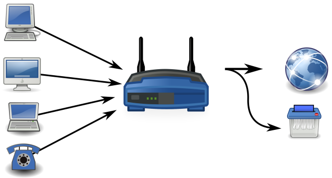


Online Packet Scheduling

- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets

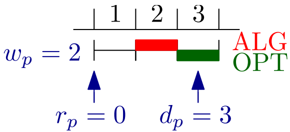


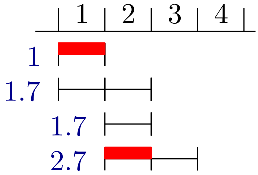


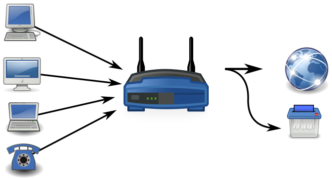


Online Packet Scheduling

- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets

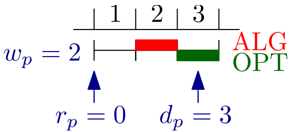


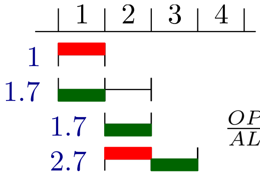




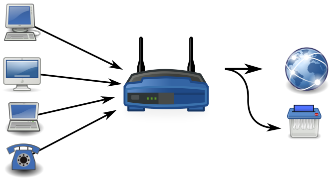
Online Packet Scheduling

- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets



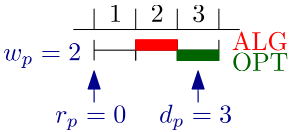


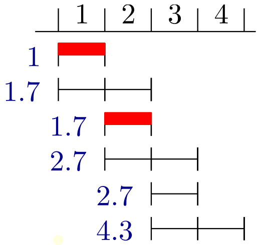
$$\frac{OPT}{ALG} = \frac{2\cdot 1.7+2.7}{1.7+2.7} \approx 1.39$$

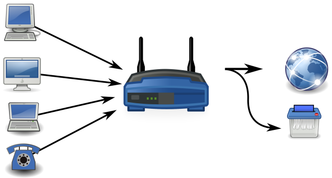


Online Packet Scheduling

- Also called Buffer Management in Quality of Service Switches
- Unit-length packets arrive over time
- Each packet has a deadline and a weight
- Time discretized to slots
- Goal: maximize total weight of scheduled packets







Competitive ratio of online algorithms

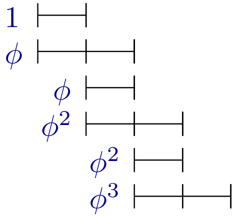
- ALG is R-competitive if for any instance I

$$ALG(I) \geq \frac{1}{R} OPT(I)$$

Previous work

- We focus on deterministic algorithms
- Lower bound of the golden ratio $\phi = \frac{1}{2}(\sqrt{5}+1) \approx 1.618$

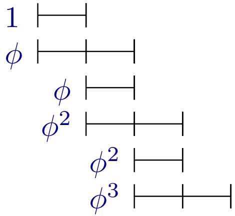
$$1 + \frac{1}{\phi} = \phi$$

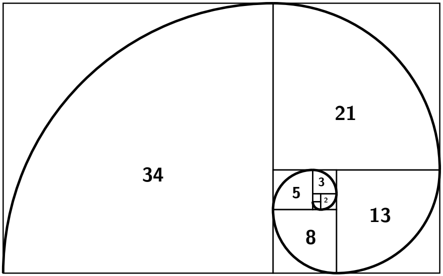


Previous work

- We focus on deterministic algorithms
- Lower bound of the golden ratio $\phi = \frac{1}{2}(\sqrt{5} + 1) \approx 1.618$

$$1 + \frac{1}{\phi} = \phi$$



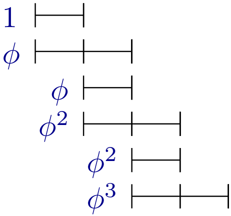


Previous work

- We focus on deterministic algorithms
- Lower bound of the golden ratio $\phi = \frac{1}{2}(\sqrt{5}+1) \approx 1.618$

$$1 + \frac{1}{\phi} = \phi$$

- $2\sqrt{2} - 1 \approx 1.828$-competitive algorithm by Englert and Westermann



Previous work: Algorithm EDF_α

- $h =$ the heaviest packet available in the current slot

Previous work: Algorithm EDF_α

- $h =$ the heaviest packet available in the current slot
- EDF_ϕ: *Earliest Deadline First*
  - Schedule the earliest-deadline packet f with $w_f \geq \frac{1}{\phi} w_h$

Previous work: Algorithm EDF_α

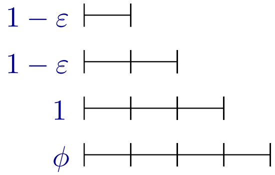
- $h =$ the heaviest packet available in the current slot
- EDF_ϕ: *Earliest Deadline First*
  - Schedule the earliest-deadline packet f with $w_f \geq \frac{1}{\phi} w_h$
- s-bounded instances
  - Each packet can be scheduled in at most s consecutive slots

Previous work: Algorithm EDF_α

- $h =$ the heaviest packet available in the current slot
- EDF_ϕ: *Earliest Deadline First*
  - Schedule the earliest-deadline packet f with $w_f \geq \frac{1}{\phi} w_h$
- s-bounded instances
  - Each packet can be scheduled in at most s consecutive slots
- ϕ-competitive for:
  - 2-bounded instances [Kesselman et al. '04]
  - 3-bounded instances [Chin et al. '06]

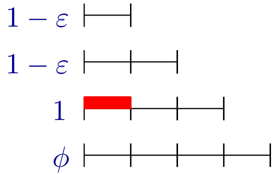
Previous work: Algorithm EDF_α

- $h =$ the heaviest packet available in the current slot
- EDF_ϕ: *Earliest Deadline First*
  - Schedule the earliest-deadline packet f with $w_f \geq \frac{1}{\phi} w_h$
- s-bounded instances
  - Each packet can be scheduled in at most s consecutive slots
- ϕ-competitive for:
  - 2-bounded instances [Kesselman et al. '04]
  - 3-bounded instances [Chin et al. '06]
  - but *not* for 4-bounded instances



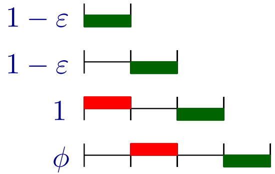
Previous work: Algorithm EDF_α

- $h =$ the heaviest packet available in the current slot
- EDF_ϕ: *Earliest Deadline First*
  - Schedule the earliest-deadline packet f with $w_f \geq \frac{1}{\phi} w_h$
- s-bounded instances
  - Each packet can be scheduled in at most s consecutive slots
- ϕ-competitive for:
  - 2-bounded instances [Kesselman et al. '04]
  - 3-bounded instances [Chin et al. '06]
  - but *not* for 4-bounded instances



Previous work: Algorithm EDF_α

- $h =$ the heaviest packet available in the current slot
- EDF_ϕ: *Earliest Deadline First*
  - Schedule the earliest-deadline packet f with $w_f \geq \frac{1}{\phi} w_h$
- s-bounded instances
  - Each packet can be scheduled in at most s consecutive slots
- ϕ-competitive for:
  - 2-bounded instances [Kesselman et al. '04]
  - 3-bounded instances [Chin et al. '06]
  - but *not* for 4-bounded instances



Our results

- ϕ-competitive algorithm for 4-bounded instances

Our results

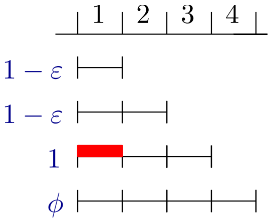
- ϕ-competitive algorithm for 4-bounded instances
- New model with *lookahead*
  - ℓ-lookahead $=$ at time t algorithm sees packets arriving by time $t+\ell$

Our results

- ϕ-competitive algorithm for 4-bounded instances
- New model with *lookahead*
  - ℓ-lookahead = at time t algorithm sees packets arriving by time $t + \ell$
  - Deterministic algorithms for 2-bounded instances
  - 1.303-competitive algorithm with 1-lookahead
  - Lower bound for ℓ-lookahead

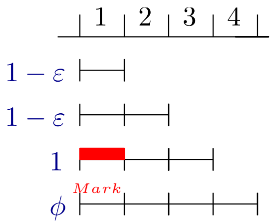
ToggleH: algorithm for 4-bounded instances

- Modification of EDF_ϕ



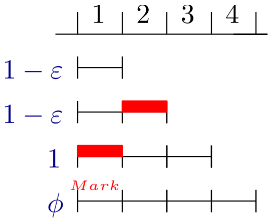
ToggleH: algorithm for 4-bounded instances

- Modification of EDF_{ϕ}



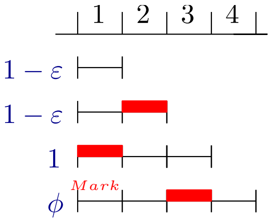
ToggleH: algorithm for 4-bounded instances

- Modification of EDF_ϕ



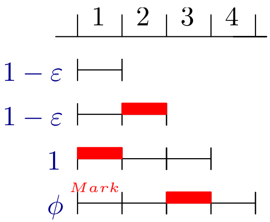
ToggleH: algorithm for 4-bounded instances

- Modification of EDF_ϕ



ToggleH: algorithm for 4-bounded instances

- Modification of EDF_ϕ
- $h =$ the heaviest packet
- $f =$ the earliest-deadline packet with $w_f \geq \frac{1}{\phi} w_h$

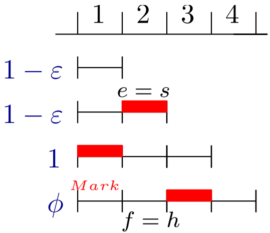


ToggleH: algorithm for 4-bounded instances

- Modification of EDF_ϕ

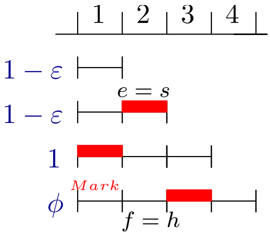
- h = the heaviest packet
- f = the earliest-deadline packet with $w_f \geq \frac{1}{\phi} w_h$

- s = the second-heaviest packet
- e = the earliest-deadline packet with $w_e \geq \frac{1}{\phi^2} w_h$



ToggleH: algorithm for 4-bounded instances

- Modification of EDF_ϕ



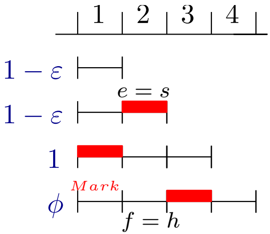
- h = the heaviest packet
- f = the earliest-deadline packet with $w_f \geq \frac{1}{\phi} w_h$
- s = the second-heaviest packet
- e = the earliest-deadline packet with $w_e \geq \frac{1}{\phi^2} w_h$

ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
  schedule e

ToggleH: algorithm for 4-bounded instances

- Modification of EDF_ϕ



- h = the heaviest packet
- f = the earliest-deadline packet with $w_f \geq \frac{1}{\phi} w_h$
- s = the second-heaviest packet
- e = the earliest-deadline packet with $w_e \geq \frac{1}{\phi^2} w_h$

ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
    schedule e
else
    schedule f
    if ($d_h = t+3$) $\wedge$ ($d_f = t+2$) **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.

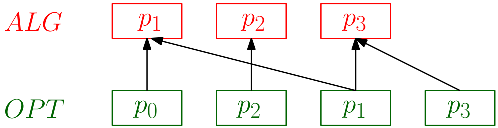
ALG	p_1	p_2	p_3	
OPT	p_0	p_2	p_1	p_3

ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
    schedule e (e**-step)**
else
    schedule f (f**-step)**
    if ($d_h = t + 3$) $\wedge$ ($d_f = t + 2$) **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.

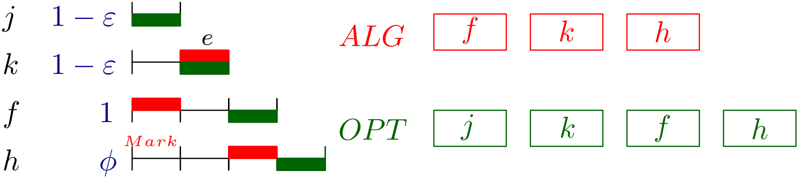


ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
  schedule e (*e*-step)
else
  schedule f (*f*-step)
  if ($d_h = t + 3$) $\wedge$ ($d_f = t + 2$) **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.

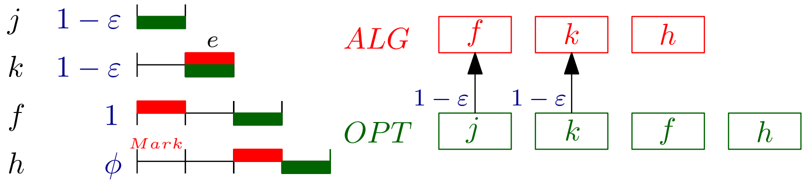


ToggleH

if $(h$ marked in the previous step$) \wedge (w_s < w_h/\phi) \wedge (d_e = t)$
  schedule e (e**-step)**
else
  schedule f (f**-step)**
  if $(d_h = t+3) \wedge (d_f = t+2)$ **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.

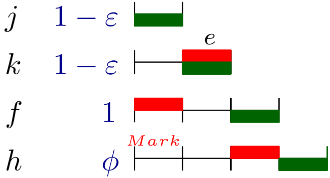


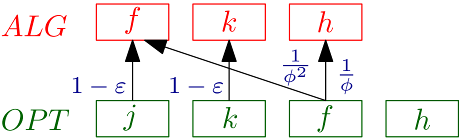
ToggleH

> **if** (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
>   **schedule e (*e*-step)**
> **else**
>   **schedule f (*f*-step)**
>   **if** ($d_h = t + 3$) $\wedge$ ($d_f = t + 2$) **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.



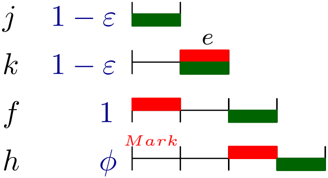


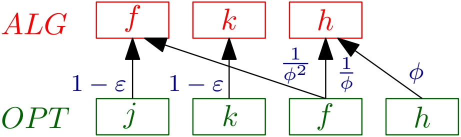
ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
  schedule e (*e*-step)
else
  schedule f (*f*-step)
  if ($d_h = t+3$) $\wedge$ ($d_f = t+2$) **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.



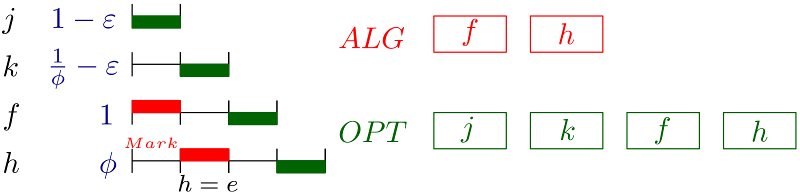


ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
  schedule e (*e*-step)
else
  schedule f (*f*-step)
  if ($d_h = t + 3$) $\wedge$ ($d_f = t + 2$) **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.

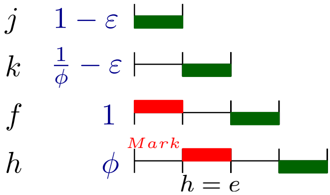


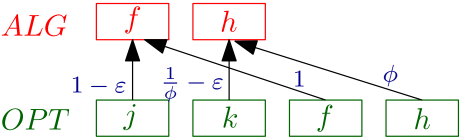
ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
    schedule e (*e*-step)
else
    schedule f (*f*-step)
    if ($d_h = t+3$) $\wedge$ ($d_f = t+2$) **then** mark h

Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.



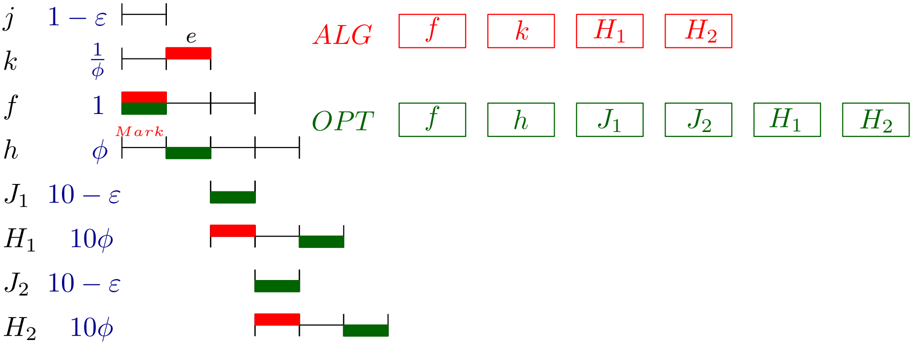


ToggleH

if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
    schedule e (*e*-step)
else
    schedule f (*f*-step)
    if ($d_h = t + 3$) $\wedge$ ($d_f = t + 2$) **then** mark h

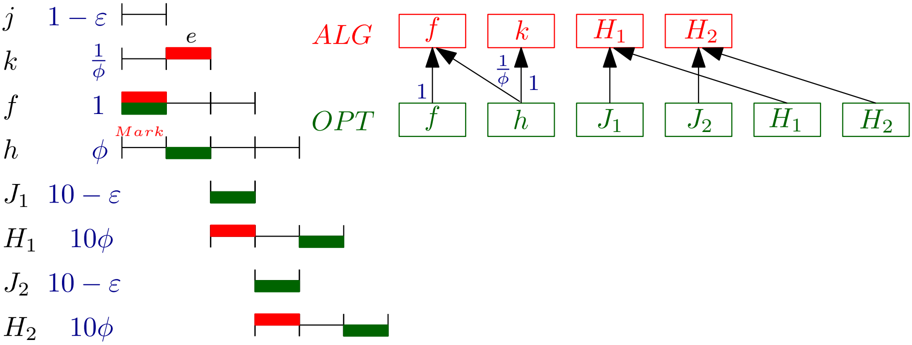
Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.



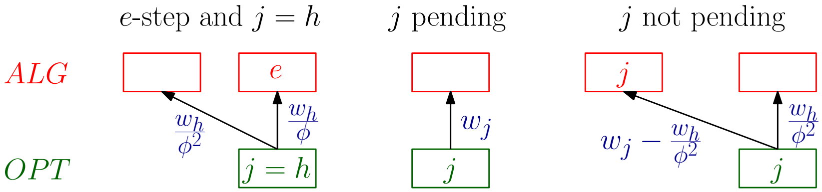
Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.



Charging scheme

- Idea: assign the weight of each packet in an optimal schedule to slots in algorithm's schedule.
- Goal: no slot t receives more that ϕw_{p_t}
  - p_t is the packet scheduled at t in ALG.



ToggleH

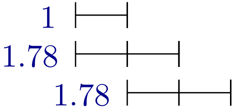
if (h marked in the previous step) $\wedge$ ($w_s < w_h/\phi$) $\wedge$ ($d_e = t$)
    schedule e (*e*-step)
else
    schedule f (*f*-step)
    if ($d_h = t + 3$) $\wedge$ ($d_f = t + 2$) **then** mark h

Algorithms with lookahead for 2-bounded instances

- ℓ-lookahead = at t algorithm sees all packets arriving by time $t + \ell$

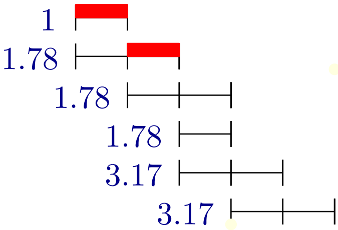
Algorithms with lookahead for 2-bounded instances

- ℓ-lookahead = at t algorithm sees all packets arriving by time $t + \ell$
- Lower bound of $\frac{1}{2(\ell+1)}\left(\sqrt{4\ell^2 + 8\ell + 5} + 1\right)$
  - $= \phi$ for $\ell = 0$,
  - $= \frac{1}{4}(\sqrt{17} + 1) \approx 1.28$ for $\ell = 1$:



Algorithms with lookahead for 2-bounded instances

- ℓ-lookahead = at t algorithm sees all packets arriving by time $t+\ell$
- Lower bound of $\frac{1}{2(\ell+1)}\left(\sqrt{4\ell^2+8\ell+5}+1\right)$
  - $=\phi$ for $\ell=0$,
  - $=\frac{1}{4}(\sqrt{17}+1)\approx 1.28$ for $\ell=1$:

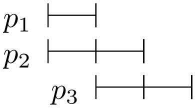


Algorithms with lookahead for 2-bounded instances

- ℓ-lookahead = at t algorithm sees all packets arriving by time $t + \ell$
- Lower bound of $\frac{1}{2(\ell+1)}\left(\sqrt{4\ell^2 + 8\ell + 5} + 1\right)$
  - $= \phi$ for $\ell = 0$,
  - $= \frac{1}{4}(\sqrt{17} + 1) \approx 1.28$ for $\ell = 1$:
- $\frac{1}{2}(\sqrt{13} - 1) \approx 1.303$-competitive algorithm with 1-lookahead
- Based on *plan*
  - Optimal schedule of pending or lookahead packets
  - Computed under assumption that no packet will arrive

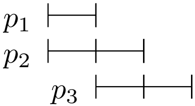
Algorithms with lookahead for 2-bounded instances

- ℓ-lookahead = at t algorithm sees all packets arriving by time $t + \ell$
- Lower bound of $\frac{1}{2(\ell+1)}\left(\sqrt{4\ell^2 + 8\ell + 5} + 1\right)$
  - $= \phi$ for $\ell = 0$,
  - $= \frac{1}{4}(\sqrt{17} + 1) \approx 1.28$ for $\ell = 1$:
- $\frac{1}{2}(\sqrt{13} - 1) \approx 1.303$-competitive algorithm with 1-lookahead
- Based on *plan*
  - Optimal schedule of pending or lookahead packets
  - Computed under assumption that no packet will arrive
  - This case: plan has 3 packets: p_1, p_2, p_3
  - We schedule p_1 or p_2.



Algorithms with lookahead for 2-bounded instances

- ℓ-lookahead = at t algorithm sees all packets arriving by time $t + \ell$
- Lower bound of $\frac{1}{2(\ell+1)}\left(\sqrt{4\ell^2 + 8\ell + 5} + 1\right)$
  - $= \phi$ for $\ell = 0$,
  - $= \frac{1}{4}(\sqrt{17} + 1) \approx 1.28$ for $\ell = 1$:
- $\frac{1}{2}(\sqrt{13} - 1) \approx 1.303$-competitive algorithm with 1-lookahead
- Based on *plan*
  - Optimal schedule of pending or lookahead packets
  - Computed under assumption that no packet will arrive
  - This case: plan has 3 packets: p_1, p_2, p_3
  - We schedule p_1 or p_2.



CompareWithBias(α)

if $r_{p_2} = t$ **and** $w_{p_1} < \min\left(w_{p_2}, w_{p_3}, \frac{1}{2\alpha}(w_{p_2} + w_{p_3})\right)$
  then schedule p_2
else schedule p_1

Open problems

- Design a ϕ-competitive algorithm

Open problems

- Design a ϕ-competitive algorithm
  - Or find a better lower bound with large span

Open problems

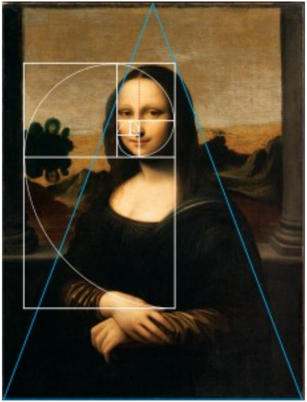
- Design a ϕ-competitive algorithm
  - Or find a better lower bound with large span
- Design a *better* than ϕ-competitive algorithm with lookahead

Open problems

- Design a ϕ-competitive algorithm
  - Or find a better lower bound with large span
- Design a *better* than ϕ-competitive algorithm with lookahead
- Randomization and lookahead

Open problems

- Design a ϕ-competitive algorithm
  - Or find a better lower bound with large span
- Design a *better* than ϕ-competitive algorithm with lookahead
- Randomization and lookahead



Thank you!

Randomized algorithms

Against oblivious adversary:

	General	2-bounded	s-bounded
Lower bound	1.25	1.25	1.25
Upper bound	$\frac{e}{e-1} \approx 1.582$	1.25	$\frac{1}{1-(1-\frac{1}{s})^s}$

Against adaptive adversary:

	General	2-bounded	s-bounded
Lower bound	1.333	1.333	1.333
Upper bound	$\frac{e}{e-1} \approx 1.582$	1.333	$\frac{1}{1-(1-\frac{1}{s})^s}$