An optimal online algorithm for packet scheduling with agreeable deadlines

Fei Li `lifei@cs.columbia.edu`

Joint work with

Prof. Jay Sethuraman   Prof. Clifford Stein

`jay@ieor.columbia.edu`   `cliff@ieor.columbia.edu`

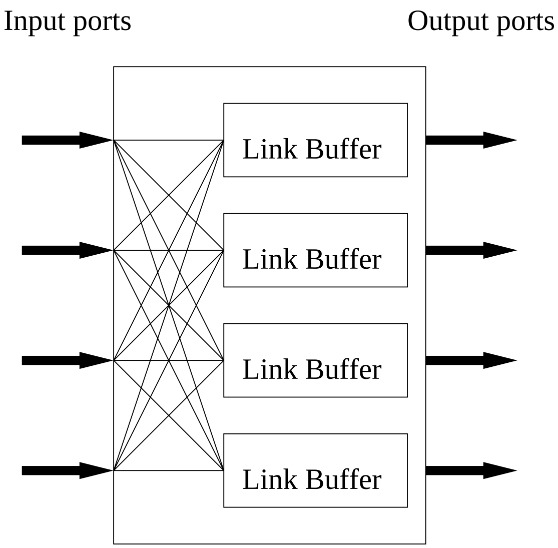
Outline

- Model & problem
- Previous work
- Our algorithm
- Analysis
- Open problems

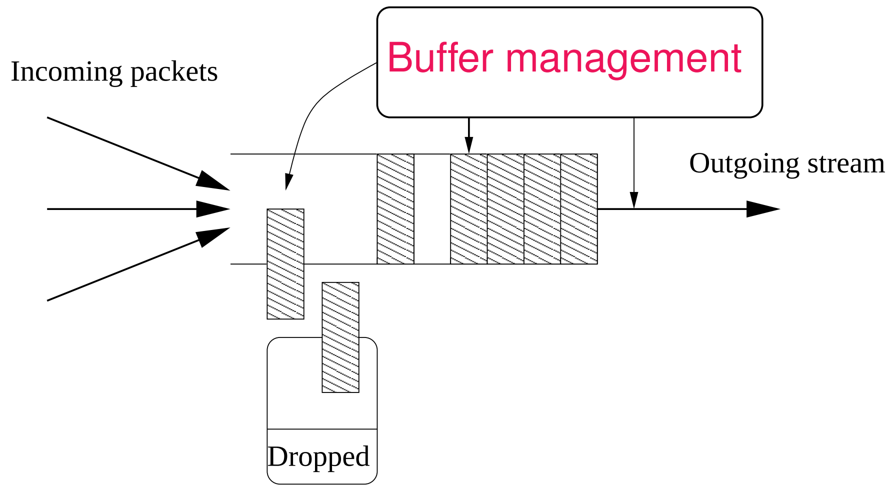
Model & problem

➢ Motivation: best-effort service provided by today's networks cannot support assured data transmission for real-time applications.

Goal: if the arriving packets cannot all be stored in a buffer, or if the packets have deadlines by which they must be delivered, the switch needs to identify the packets that should be dropped, without knowledge of future arrivals.



General switch structure



Output port structure

Model & problem

- **Model**: a buffer of size $B \in \mathbb{Z}$; packet j is released at release time $r_j \in \mathbb{Z}$, it has a weight $w_j > 0$, a deadline $d_j \in \mathbb{Z}$. At each integer time step, exactly one packet can be transmitted.

  Objective: $\max \sum w_j$ for transmitted packets j.

- **Bounded-delay buffer** `[Kesselman et al. STOC 01]`:

  Each packet must be transmitted within its *allowed delay time* (*slack time*: $s_j = d_j - r_j$) or else it is lost. The buffer management policy is allowed to re-order the packets. Given a known parameter s,

  - s-uniform: $\forall$ packet j, $s_j = d_j - r_j = s$
  - s-bounded: $\forall$ packet j, $s_j = d_j - r_j \leq s$
  - Agreeable deadlines: $\forall$ packets $i \neq j$, $r_i < r_j$ implies $d_i \leq d_j$ (including s-uniform, $\forall s$ and 2-bounded)

Model & problem

- **Competitive ratio**: for an online maximization problem, the input received is in an online manner, the output must be generated online. Let $OPT(I)$ denote the offline optimum, an online algorithm A is said to be c-competitive if for any finite input sequence I and an additive constant α,

$$c \cdot A(I) + \alpha \geq OPT(I)$$

- **Known result**
  - Lower bound $\phi := (\sqrt{5} + 1)/2$ applies to deterministic online algorithms in scheduling packets with arbitrary deadlines [Hajek CISS 01] [Chin & Fung Algorithmica 03] [Andelman et al. SODA 03].

- **Problem**: find a ϕ-competitive deterministic online algorithm.

Outline

- Model & problem
- Previous work
  - Lower bound ϕ for an instance with agreeable deadlines
  - A 2-competitive greedy algorithm for instances with arbitrary deadlines
  - An improved greedy algorithm
  - Randomized algorithms
- Our algorithm
- Analysis
- Open problems

Previous work

- Lower bound ϕ applies to instances with agreeable deadlines [Hajek CISS 01] [Chin & Fung Algorithmica 03].
- A simple greedy algorithm scheduling an available packet with the maximum weight is 2-competitive [Hajek CISS 01] [Kesselman et al. STOC 01].
- An improved greedy algorithm
  - $64/33 \approx 1.939$-competitive for instances with arbitrary deadlines; $5-\sqrt{10} \approx 1.838$-competitive for instances with agreeable deadlines [Chrobak et al. ESA 04].
    - A flag indicates alternatively scheduling the earliest packet and the maximum-weight packet; only 2 packets are considered.
- Randomized algorithms
  - Lower bound of 1.25.
  - A randomized algorithm is $e/(e-1) \approx 1.582$-competitive [Bartal et al. STACS 04].

Our result

- A memoryless, deterministic online algorithm with competitive ratio ϕ for instances with agreeable deadlines.

Outline

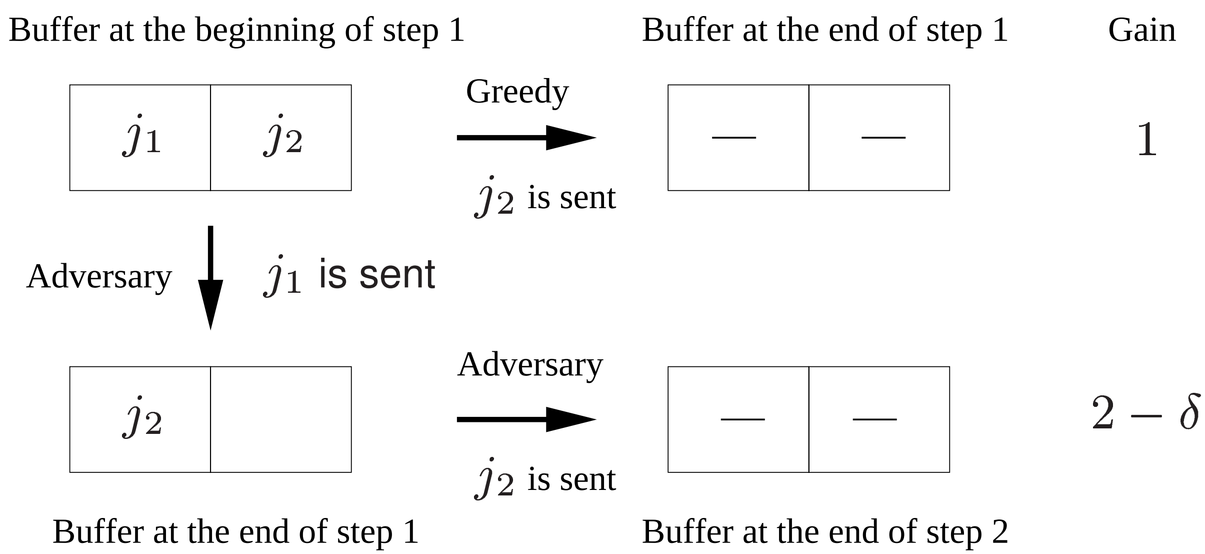
- Model & problem
- Previous work
- Our algorithm
- Analysis
- Open problems

Our algorithm

➢ Observation: why cannot the greedy algorithm be better than 2-competitive?

$j_1 : (w_1, d_1) = (1 - \delta, 1) \quad j_2 : (w_2, d_2) = (1, 2)$

- The greedy algorithm chooses j_2, though j_1 has only slightly lower value ➔ j_1 becomes invalid.



- Selecting the earliest packet among a set of sufficient large weight runs into a similar difficulty ➔ one packet with slightly lower value but earlier deadline becomes invalid.

Our algorithm

➢ Intuition: "balance" the first packet and the maximum-weight packet

- Identify a packet that has "sufficient large" weight compared to the maximum-weight packet, also "sufficient large" weight compared to an earliest-deadline packet.
- 2 important packets:
  - h: the maximum-weight packet with the earliest deadline in the buffer.
  - e: the earliest-deadline packet with the maximum weight in the buffer.

➡ ☼ We consider scheduling a packet "in-between" the earliest packet and the maximum-weight packet.

Our algorithm

- Technical issues
  - Early dropping: buffer contains a set of packets that can be scheduled. For example, if 3 packets arrive, each with slack time 2, the algorithm immediately drops a minimum-weight packet.
  - Canonical order: packets in the buffer are assigned in increasing order of deadline, with ties broken in order of decreasing weight.
- Our algorithm MG (modified greedy)
  1. At the beginning of each step t, MG identifies a set of packets from the packets in the buffer and newly released ones: maximum-valued feasible subset of at most B packets. All remaining packets are dropped.
  2. The selected packets are arranged in canonical order.
  3. If $w_e \geq w_h/\phi$, packet e is sent; otherwise, the earliest packet f satisfying $w_f \geq \max\{\phi w_e, w_h/\phi\}$ is sent – such packet exists because h is a candidate.

Our algorithm

➢ An example showing how MG runs

- Packet j is represented as (w_j, d_j).

Step	Buffer size B = 3	e−packet	h−packet	f−packet	MG gains
0	— | — | —	—	—	—	—
	↓ (3, 1) (2, 2) (3, 2) (1, 4)				
1	(3, 1) | (3, 2) | (1, 4)	(3, 1)	(3, 1)	—	3
Send e	↓ (6, 5)				
2	(3, 2) | (1, 4) | (6, 5)	(3, 2)	(6, 5)	(6, 5)	6
Send f	↓				
3	(1, 4) | — | —	(1, 4)	(1, 4)	—	1

Outline

- Model & problem
- Previous work
- Our algorithm
- Analysis
- Open problems

Analysis

- **Approach**: online computing is regarded as a game with its *adversary*. Our analysis method is original and it simplifies the analysis.

- **Basis**
  - MG and the adversary have identical buffers at the beginning of step t.
  - Both MG and the adversary process arriving packets and transmit a packet, at the end of the step, their buffer contents may be different.
  - Modify the adversary's buffer and make it identical to MG's buffer ➞ may let MG's adversary collect more weight.

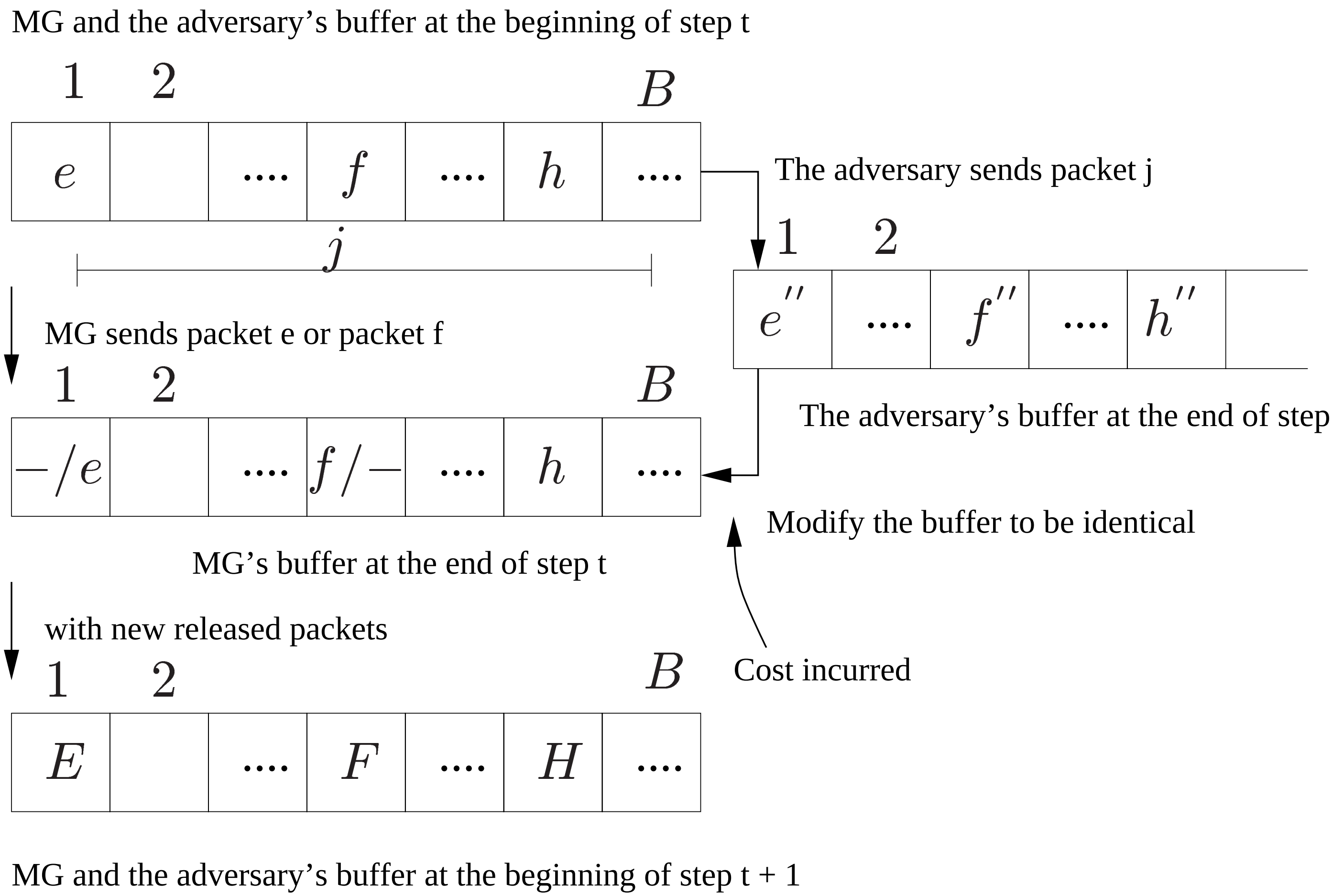
  For step t

  - v_{ADV}: value collected by the (modified) adversary (ADV).
  - v_{MG}: value collected by MG.

- **Crux**: $v_{MG} \geq v_{ADV}/\phi$.

Analysis

➢ Assumption: without loss of generality, ADV delivers packets in non-decreasing deadline order.



Analysis

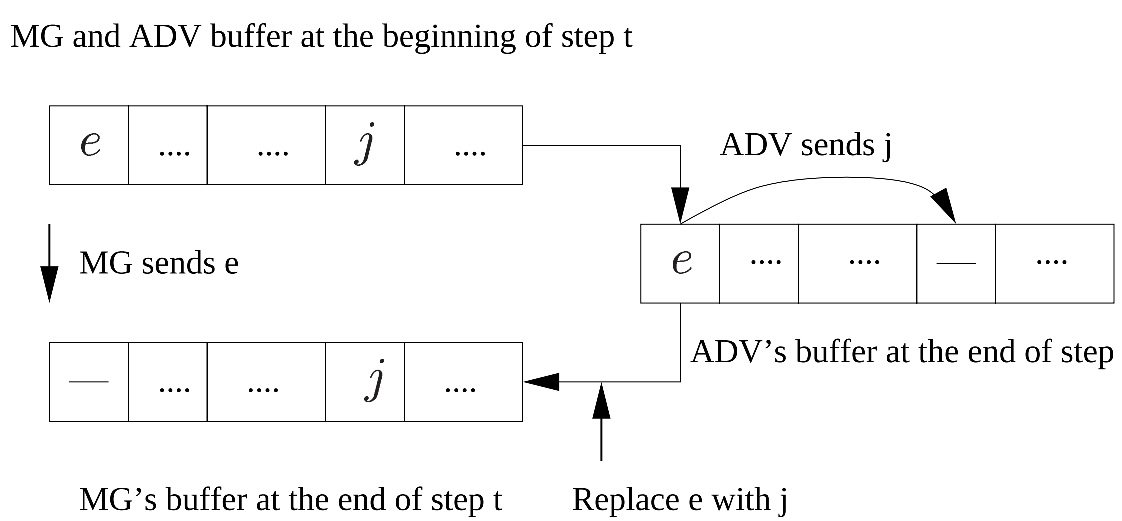
➢ Cases

1. If MG and ADV transmit the same packet: $v_{ADV} = v_{MG}$ and their buffers are identical.

2. MG transmits packet e; ADV transmits packet j:

   – $w_e \geq w_h/\phi \geq w_j/\phi \rightarrow v_{ADV} = w_j \leq \phi w_e = \phi v_{MG}$.

   – ADV's buffer does not contain j, but contains e; MG's buffer contains j, but not e. Replace e with j.



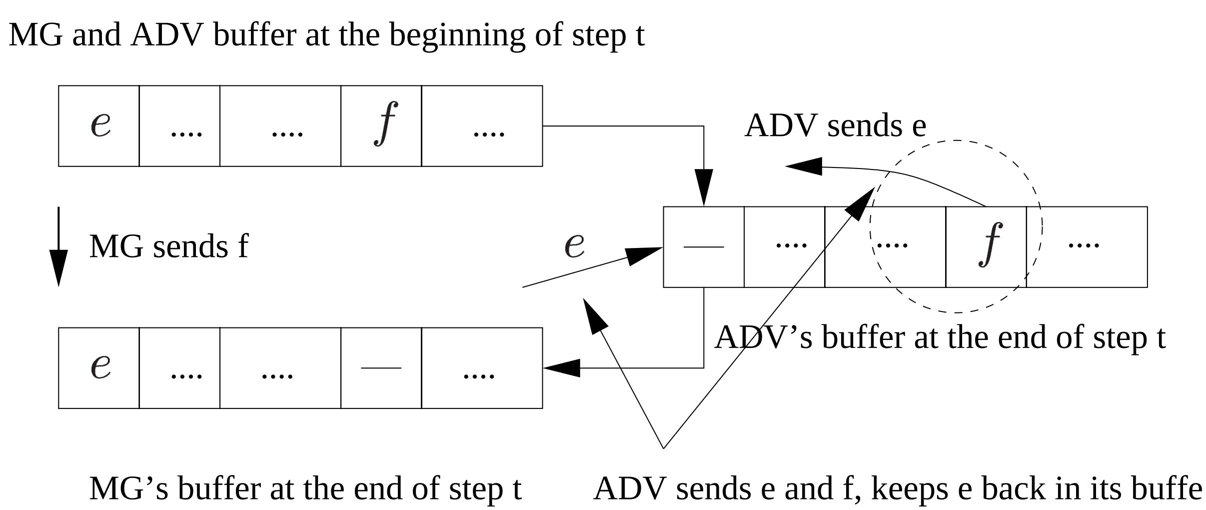
Analysis

➢ Cases

3. MG transmits packet $f \neq e$; ADV transmits packet e:

   – ADV must transmit f eventually. Let the modified adversary send e and f in this time step and keep e in its buffer.

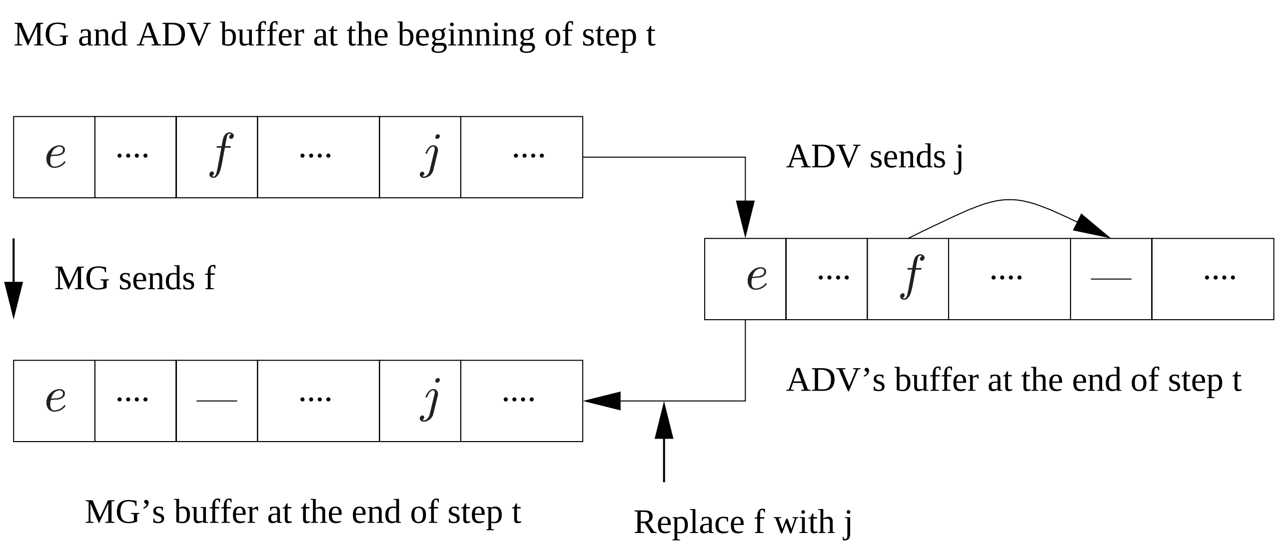
   – $v_{ADV} = w_e + w_f; v_{MG} = w_f$ ➞ $v_{ADV}/v_{MG} = 1 + w_e/w_f \leq 1 + 1/\phi = \phi$.



Analysis

➢ Cases

4. MG transmits packet $f \neq e, h$; ADV transmits packet j that is after packet f:
   - $w_f \geq w_h/\phi \geq w_j/\phi \rightarrow v_{ADV} = w_j \leq \phi w_f = \phi v_{MG}$.
   - ADV does not send $f \rightarrow w_j > w_f$.
   - ADV's buffer does not contain j, but contains f; MG's buffer contains j, but not f. Replace f with j.



Analysis

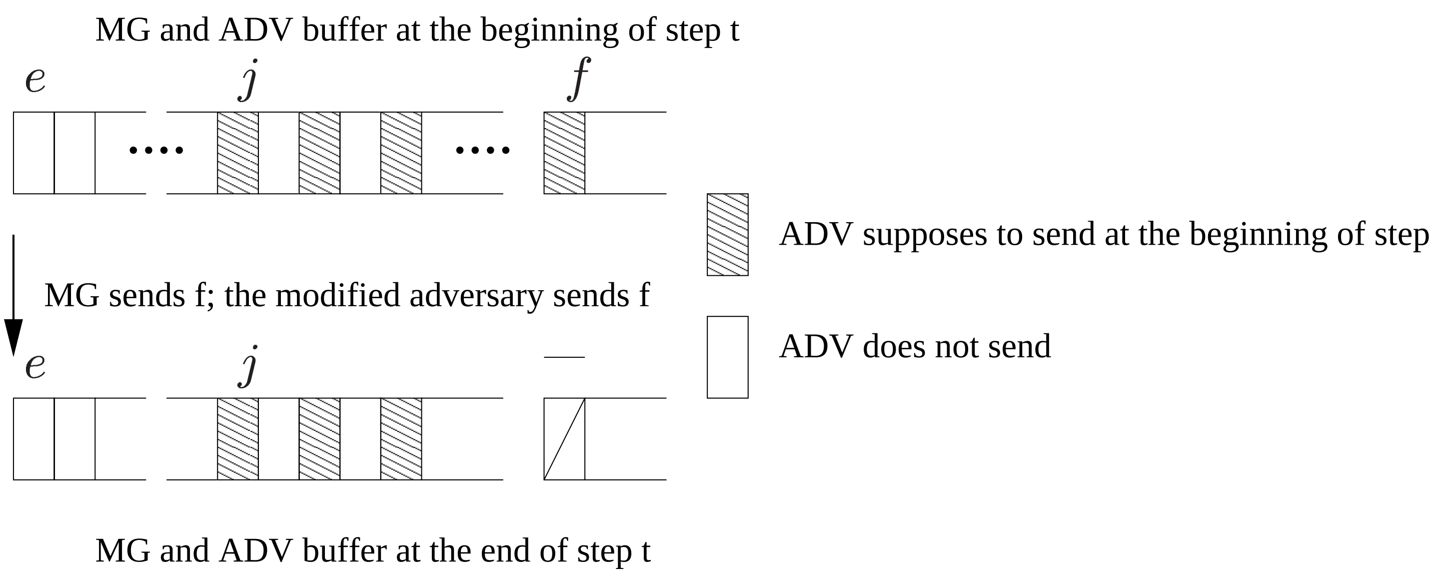
➢ Cases

5. MG transmits packet $f \neq e$; ADV transmits packet $j \neq e$ that is earlier than f:
   - f must be eventually transmitted by ADV.
   - ADV has a feasible schedule in which f is transmitted now, regardless of the future arrivals.
     * Let $p_1, p_2, \ldots,$ be the packets in the buffer: $f = p_l$, $j = p_k$. All packets are schedulable in the absence of future arrivals ➜ $d_{p_i} \geq t + i$. Packet p_i is *critical* if $d_{p_i} = t + i$.
     * Since $d_{p_i} \geq t + i$ and all future arrivals have deadlines no earlier than d_f, none of the packets $p_k, p_{k'}, \ldots, p_{l'}, p_l$ transmitted by the adversary is critical. So the sequence $p_l, p_k, p_{k'}, \ldots, p_{l'}$ is a valid transmission sequence for ADV.
   - f is sent ➜ MG and ADV transmit the same packet and gain identical weight in this step.

Analysis

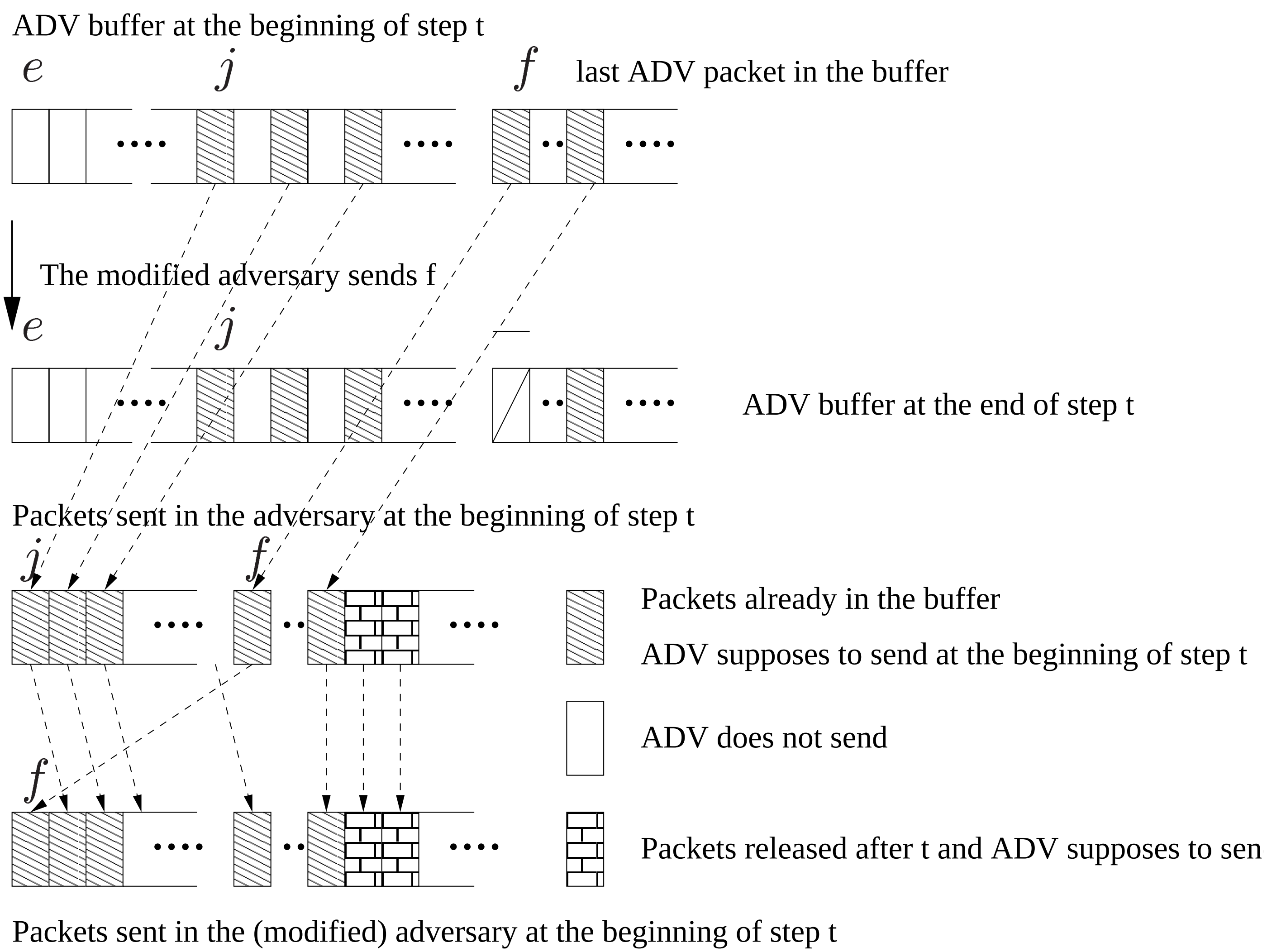
➢ Notice

- Case 5 is the only case that requires *agreeable deadlines* assumption.
  - Following is the buffer content comparison for MG and the modified ADV at the beginning and at the end of step t.



  - We investigate the packet sequence change with the modification on the adversary's buffer.

Analysis: case 5



Analysis (summary)

- Only case 5 requires agreeable deadline assumption.
- After each step, with corresponding modification on the adversary's buffer, the modified adversary and MG have identical buffers.
- For each step, the modified adversary gains at most ϕ times of what MG gains.
- MG is simple, both in design and analysis.

Case	v_{MG}	v_{ADV}	ratio (v_{ADV}/v_{MG})	modification
1	w_j	w_j	$w_j/w_j = 1$	–
2	w_e	w_j	$w_j/w_e \leq \phi$	replace e with j
3	w_f	$w_e + w_f$	$(w_e + w_f)/w_f < \phi$	send e and f, keep e
4	w_f	w_j $(w_j > w_f)$	$w_j/w_f \leq \phi$	replace f with j
5	w_f	w_f $(w_j < w_f)$	$w_f/w_f = 1$	send f, keep j

Outline

- Model & problem
- Previous work
- Our algorithm
- Analysis
- Open problems

Open problems

➢ Known

- MG is an optimal deterministic algorithm for instances with agreeable deadlines.
- A concrete instance with agreeable deadlines reaches lower bound ϕ.
- Lower bound of 1.25 for online randomized algorithms.
- A randomized algorithm with competitive ratio $e/(e-1) \approx 1.582$.

➢ Open problems

- MG's competitiveness for instances with arbitrary deadlines?
- Exists an instance with arbitrary deadlines, the lower bound $> \phi$?
- An optimal deterministic online algorithm for instances with arbitrary deadlines?
- Close the gap $[1.25, 1.582]$ for randomized online algorithms?