Online Time-Constrained Scheduling in Linear Networks

Gabriel Scalosub
Computer Science Deptartment
Technion

Joint work with Seffi Naor and Adi Rosén

Motivation

- Multimedia applications call for QoS guarantees.

Motivation

- Multimedia applications call for QoS guarantees.

$\Longrightarrow$ **Packets have deadlines**

Motivation

- Multimedia applications call for QoS guarantees.

$\Longrightarrow$ **Packets have deadlines**

- Different applications have different importance (DiffServ).

Motivation

- Multimedia applications call for QoS guarantees.

$\Longrightarrow$ **Packets have deadlines**

- Different applications have different importance (DiffServ).

$\Longrightarrow$ **Packets have weights**

Motivation

- Multimedia applications call for QoS guarantees.

$\Longrightarrow$ **Packets have deadlines**

- Different applications have different importance (DiffServ).

$\Longrightarrow$ **Packets have weights**

- Optical networks - opto-electronic conversion is costly (e.g. WDM).

Motivation

- Multimedia applications call for QoS guarantees.

$\Longrightarrow$ **Packets have deadlines**

- Different applications have different importance (DiffServ).

$\Longrightarrow$ **Packets have weights**

- Optical networks - opto-electronic conversion is costly (e.g. WDM).

$\Longrightarrow$ **Bufferless scheduling**

Motivation

- Multimedia applications call for QoS guarantees.

  $\Longrightarrow$ **Packets have deadlines**

- Different applications have different importance (DiffServ).

  $\Longrightarrow$ **Packets have weights**

- Optical networks - opto-electronic conversion is costly (e.g. WDM).

  $\Longrightarrow$ **Bufferless scheduling**

- Routing along linear networks serves as a building block in other topologies.

Motivation

- Multimedia applications call for QoS guarantees.

  $\Longrightarrow$ **Packets have deadlines**

- Different applications have different importance (DiffServ).

  $\Longrightarrow$ **Packets have weights**

- Optical networks - opto-electronic conversion is costly (e.g. WDM).

  $\Longrightarrow$ **Bufferless scheduling**

- Routing along linear networks serves as a building block in other topologies.

  $\Longrightarrow$ **Linear Networks**

Motivation

- Multimedia applications call for QoS guarantees.

  $\Longrightarrow$ **Packets have deadlines**

- Different applications have different importance (DiffServ).

  $\Longrightarrow$ **Packets have weights**

- Optical networks - opto-electronic conversion is costly (e.g. WDM).

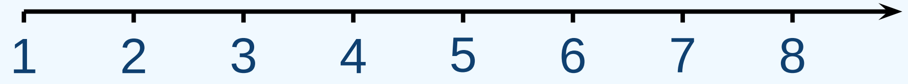
  $\Longrightarrow$ **Bufferless scheduling**

- Routing along linear networks serves as a building block in other topologies.

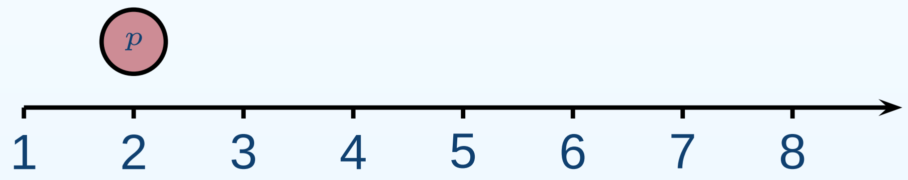
  $\Longrightarrow$ **Linear Networks**

- Traffic exceeds network capacities.

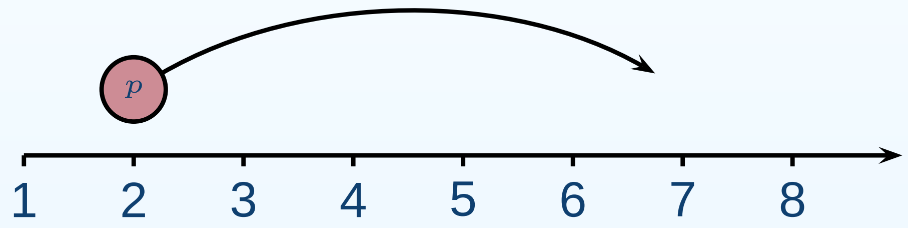
Network Model



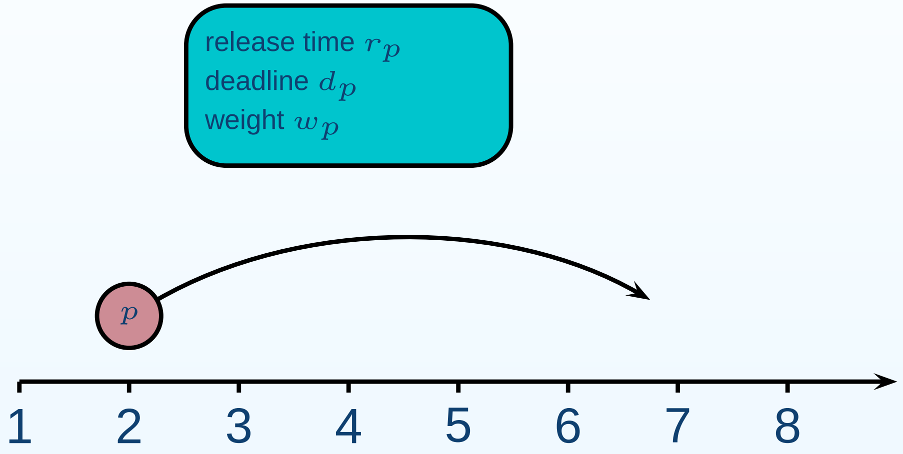
Network Model



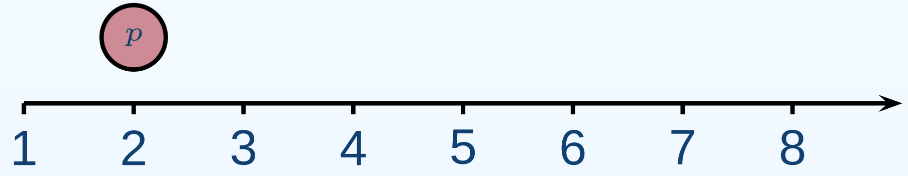
Network Model



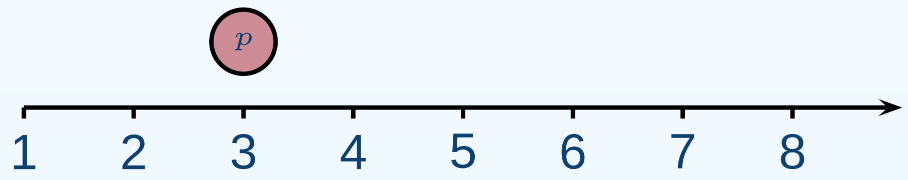
Network Model



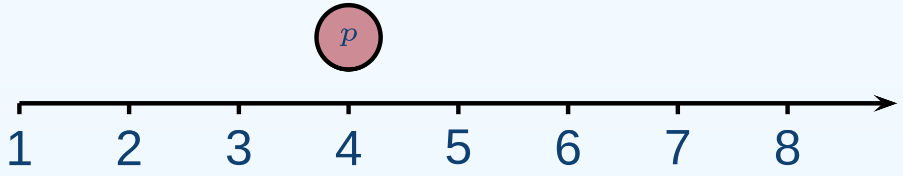
Network Model



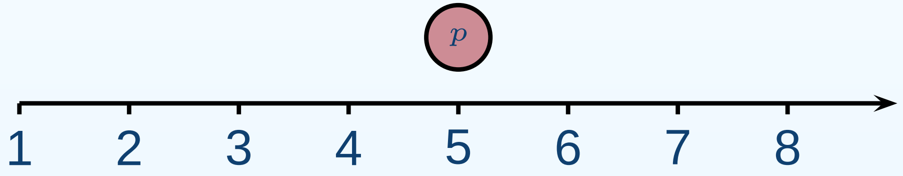
Network Model



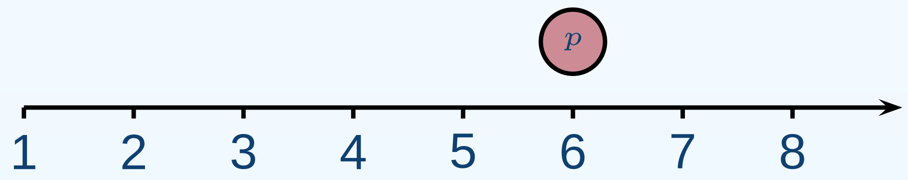
Network Model



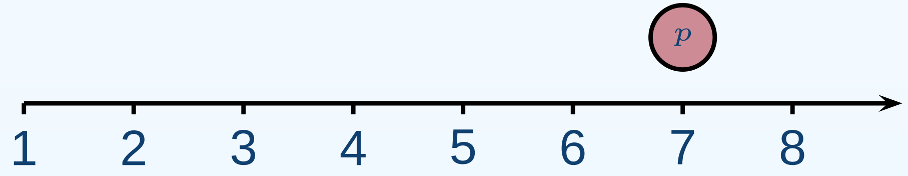
Network Model



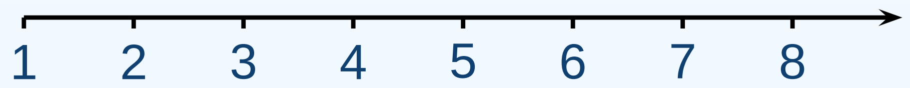
Network Model



Network Model



Network Model



Goal:

Find a maximum-weight subset of the packets S, and a bufferless schedule of S, such that every packet $p \in S$

- leaves its source after its release time, and
- arrives at its destination by its deadline.

Notation

- We consider several special weight functions
  - *Throughput Maximization*: all packet weights are equal.
  - *Maximum Network Utilization*: $\forall p, \quad w_p = |p|$.
- In what follows we use the following notation

M -	$\max_p |p|$	$\rho_{\min}$ -	$\min_p w_p/|p|$
m -	$\min_p |p|$	$\rho_{\max}$ -	$\max_p w_p/|p|$
α -	M/m	β -	$\rho_{\max}/\rho_{\min}$

  and R denotes the number of different packet lengths.

Competitive Analysis

- We study the *online* version of the problem, i.e.:
  At any time t, the algorithm makes decisions while only knowing of packets which arrived by time t.

- We use *competitive analysis* to evaluate the performance of our algorithms:
  - Compare the performance of our algorithm with an optimal (clairvoyant) schedule.
  - Analysis applicable to *every* input sequence (e.g., independent of probabilistic assumptions).

- An algorithm A is *δ-competitive* if for every input σ,

$$A(\sigma) \geq \frac{1}{\delta} OPT(\sigma).$$

Our Results

- Algorithm A_{MT} for Throughput Maximization
  - $O(\min\{\log\alpha, R\})$-competitive.
  - Experimental results comparing it to other algorithms.

Our Results

- Algorithm A_{MT} for Throughput Maximization
  - $O(\min\{\log \alpha, R\})$-competitive.
  - Experimental results comparing it to other algorithms.
- Algorithm A_{MNU} for Maximum Network Utilization
  - $(2\phi + 1)$-competitive.

Our Results

- Algorithm A_{MT} for Throughput Maximization
  - $O(\min\{\log \alpha, R\})$-competitive.
  - Experimental results comparing it to other algorithms.
- Algorithm A_{MNU} for Maximum Network Utilization
  - $(2\phi + 1)$-competitive.
- Arbitrary Weights:
  - A_{MNU} is actually $O(\beta)$-competitive for arbitrary weights.

Our Results

- Algorithm A_{MT} for Throughput Maximization
  - $O(\min\{\log\alpha, R\})$-competitive.
  - Experimental results comparing it to other algorithms.
- Algorithm A_{MNU} for Maximum Network Utilization
  - $(2\phi + 1)$-competitive.
- Arbitrary Weights:
  - A_{MNU} is actually $O(\beta)$-competitive for arbitrary weights.
- Lower bound of 2 for Throughput Maximization.

Our Results

- Algorithm A_{MT} for Throughput Maximization
  - $O(\min\{\log\alpha, R\})$-competitive.
  - Experimental results comparing it to other algorithms.
- Algorithm A_{MNU} for Maximum Network Utilization
  - $(2\phi + 1)$-competitive.
- Arbitrary Weights:
  - A_{MNU} is actually $O(\beta)$-competitive for arbitrary weights.
- Lower bound of 2 for Throughput Maximization.
- Lower bounds for Maximum Network Utilization and arbitrary weights follow from [Baruah *et al.* (1992)].

Our Results

- Algorithm A_{MT} for Throughput Maximization
  - $O(\min\{\log\alpha, R\})$-competitive.
  - Experimental results comparing it to other algorithms.
- Algorithm A_{MNU} for Maximum Network Utilization
  - $(2\phi + 1)$-competitive.
- Arbitrary Weights:
  - A_{MNU} is actually $O(\beta)$-competitive for arbitrary weights.
- Lower bound of 2 for Throughput Maximization.
- Lower bounds for Maximum Network Utilization and arbitrary weights follow from [Baruah *et al.* (1992)].
- Our algorithms extend well to the ring topology.

Previous Work

- Offline 2-approximation algorithm for Throughput Maximization in linear networks.
  [Adler *et al.* (1998)]
- Extension to arbitrary weights.
  [Adler *et al.* (1999)]
- Offline constant-approximations for trees/mesh networks.
  [Adler *et al.* (1999)]

Previous Work

- Offline 2-approximation algorithm for Throughput Maximization in linear networks.
  [Adler *et al.* (1998)]
- Extension to arbitrary weights.
  [Adler *et al.* (1999)]
- Offline constant-approximations for trees/mesh networks.
  [Adler *et al.* (1999)]

But,

Previous Work

- Offline 2-approximation algorithm for Throughput Maximization in linear networks.
  [Adler *et al.* (1998)]
- Extension to arbitrary weights.
  [Adler *et al.* (1999)]
- Offline constant-approximations for trees/mesh networks.
  [Adler *et al.* (1999)]

But,

None of these provide an online Algorithm!

Previous Work

- Offline 2-approximation algorithm for Throughput Maximization in linear networks.
  [Adler *et al.* (1998)]
- Extension to arbitrary weights.
  [Adler *et al.* (1999)]
- Offline constant-approximations for trees/mesh networks.
  [Adler *et al.* (1999)]

But,

None of these provide an online Algorithm!

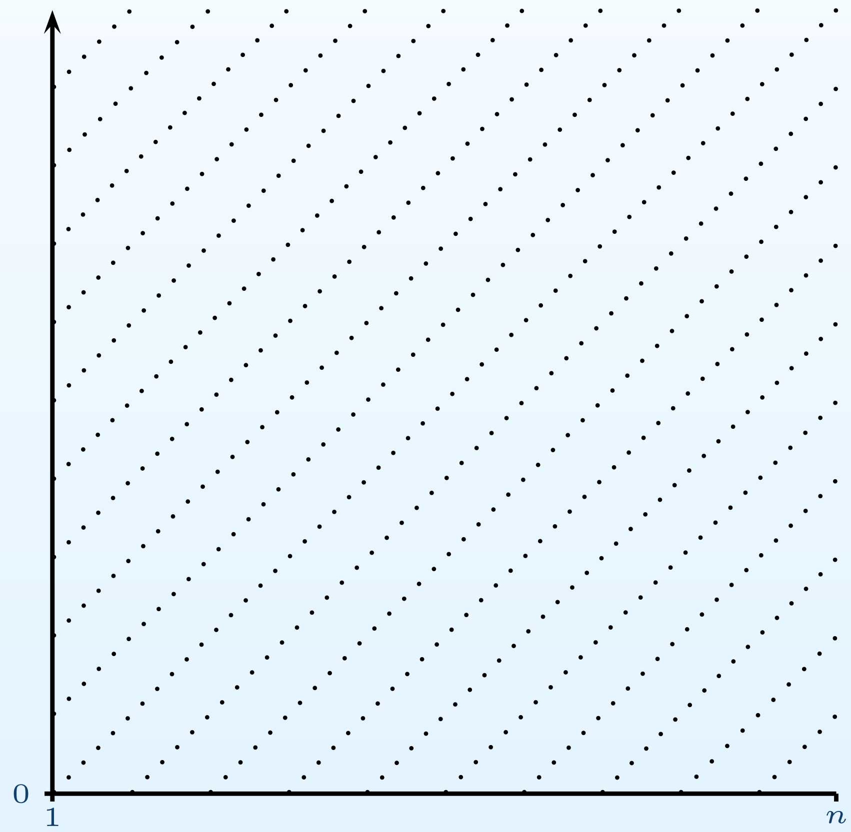
- $\Omega(\log n)$ online LB for trees. [Adler *et al.* (1999)]
- Closely related to interval scheduling and call control problems. [Garay *et al.* (1993)], [Lipton and Tomkins (1994)]
- Hard to approximate for general topologies. [Adler *et al.* (1999)]

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$
- Concept of *waves*: SW-NE lines on which we 'mount' packets.
- Every packet has a set of *eligible waves*.

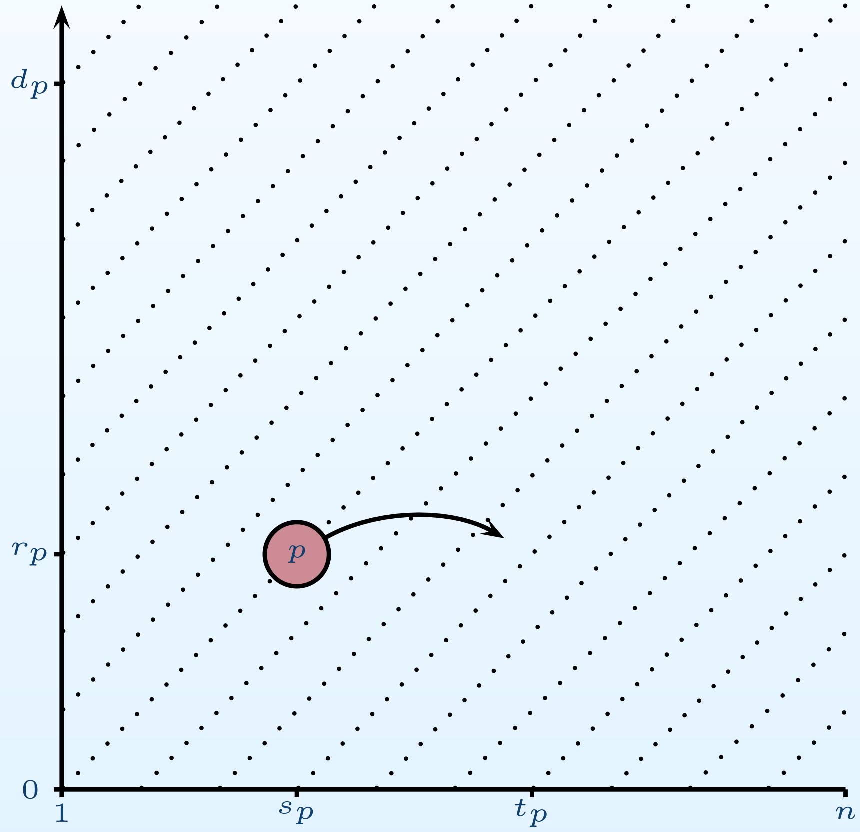


X-axis: the network

Y-axis: time

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$
- Concept of *waves*: SW-NE lines on which we 'mount' packets.
- Every packet has a set of *eligible waves*.

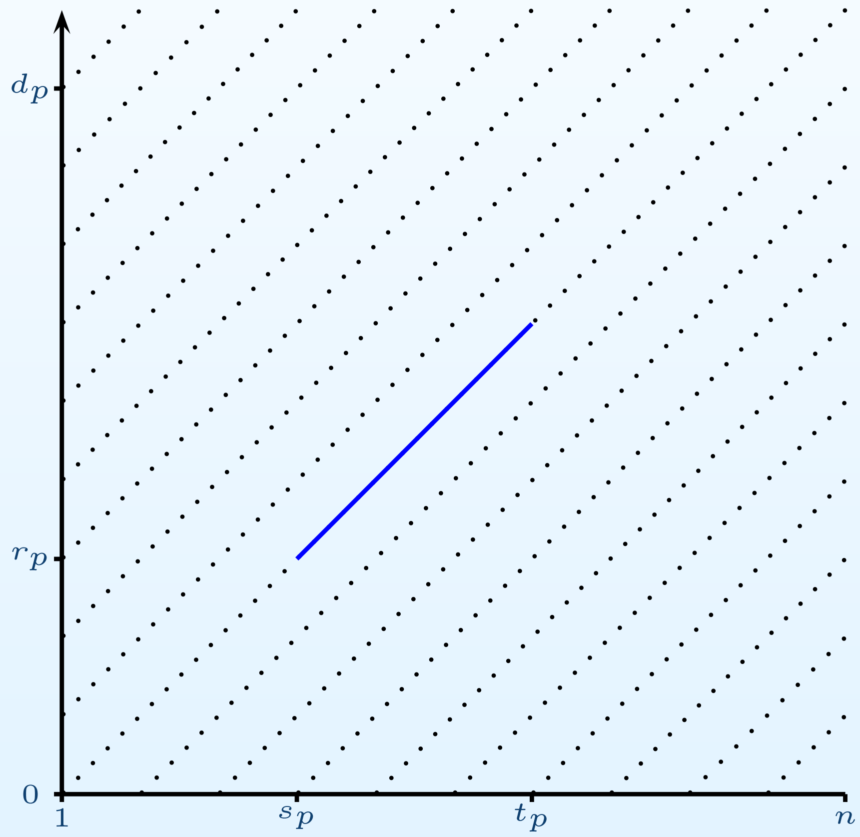


X-axis: the network

Y-axis: time

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$
- Concept of *waves*: SW-NE lines on which we 'mount' packets.
- Every packet has a set of *eligible waves*.

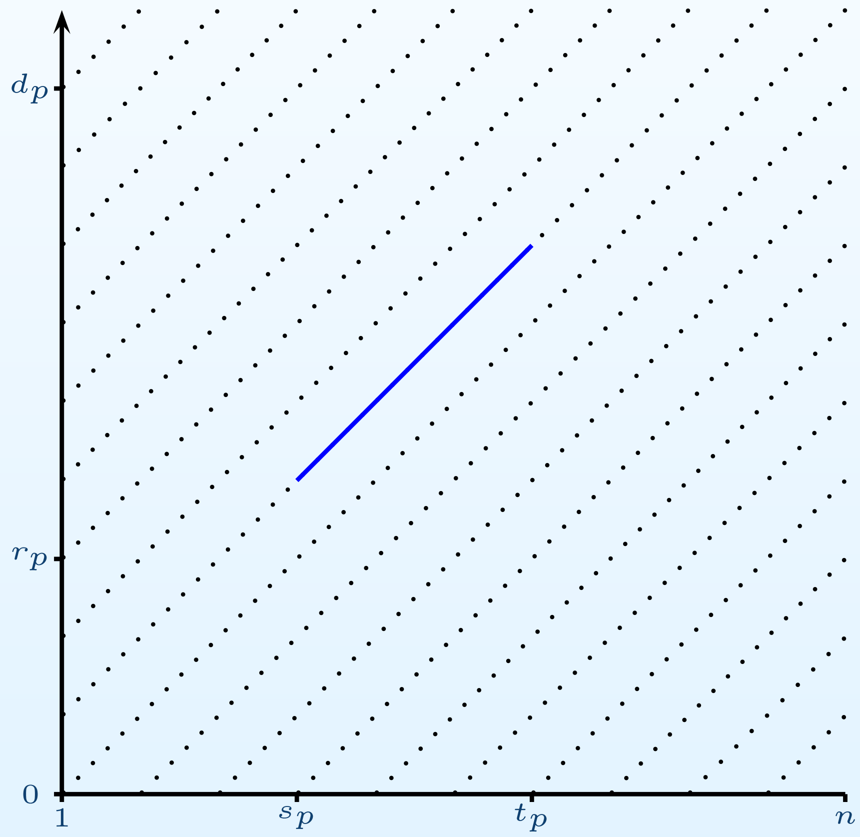


X-axis: the network

Y-axis: time

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$
- Concept of *waves*: SW-NE lines on which we 'mount' packets.
- Every packet has a set of *eligible waves*.

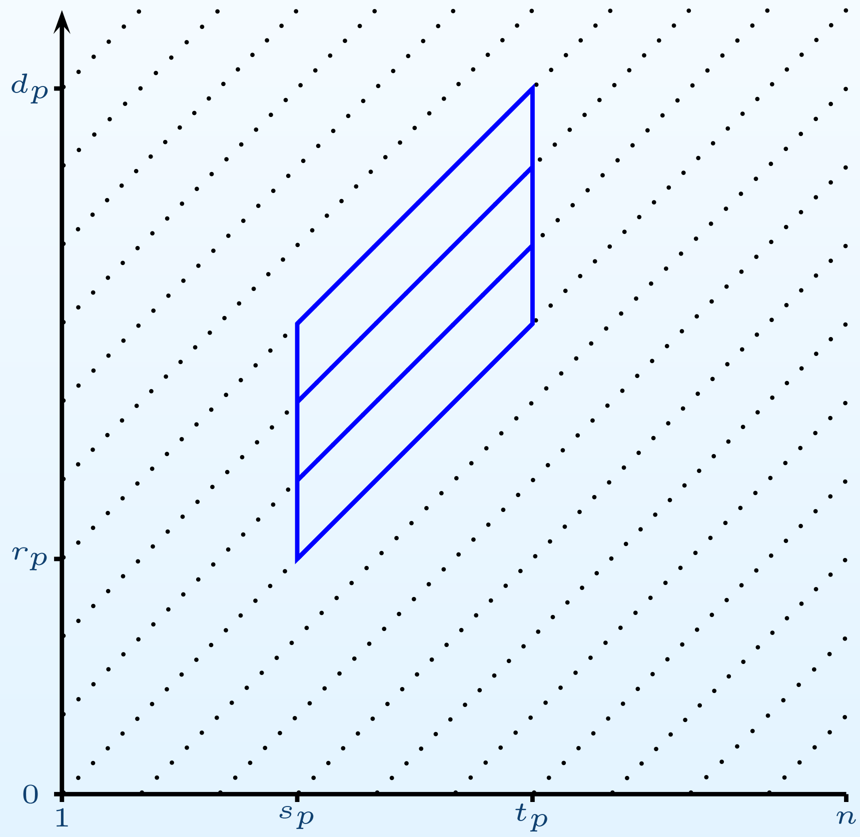


X-axis: the network

Y-axis: time

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$
- Concept of *waves*: SW-NE lines on which we 'mount' packets.
- Every packet has a set of *eligible waves*.

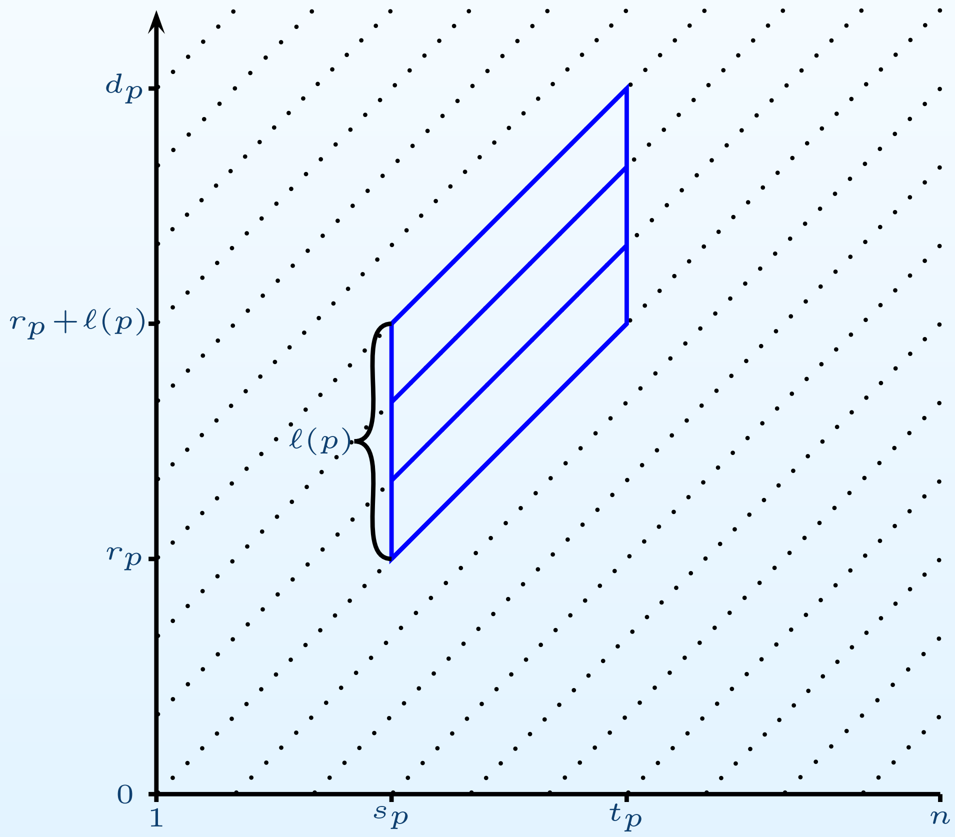


X-axis: the network

Y-axis: time

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$
- Concept of *waves*: SW-NE lines on which we 'mount' packets.
- Every packet has a set of *eligible waves*.

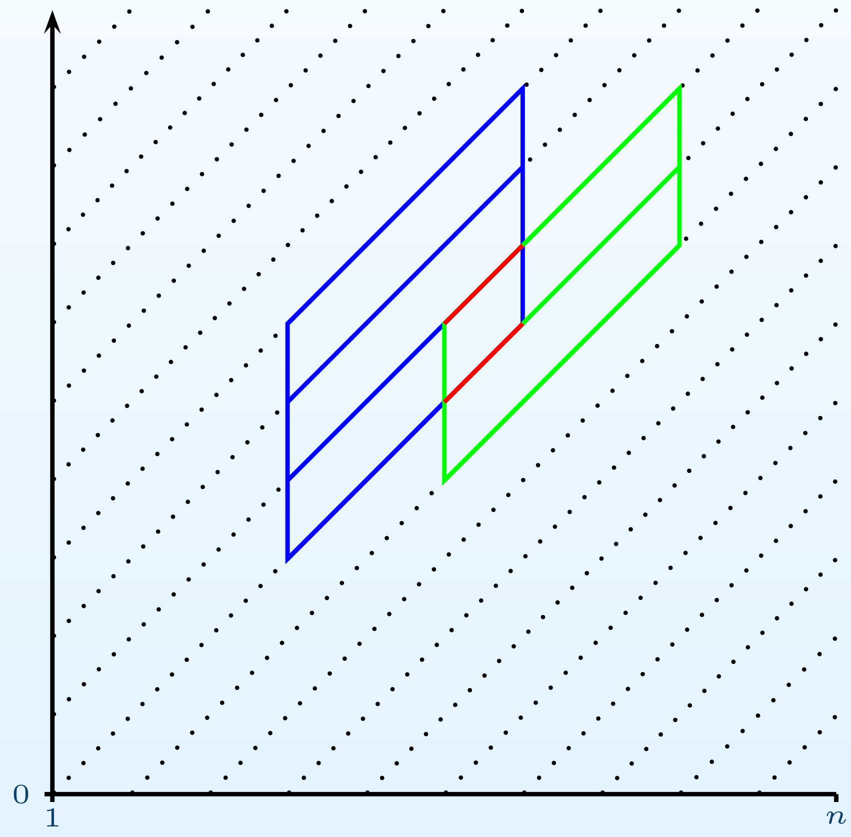


X-axis: the network

Y-axis: time

Slack and Geometric Interpretation

- The *slack* of packet p: $\ell(p) = d_p - r_p - |p|$
- Concept of *waves*: SW-NE lines on which we 'mount' packets.
- Every packet has a set of *eligible waves*.



X-axis: the network

Y-axis: time

Preliminaries and Lower bound

- Not allowing preemption might be costly

Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.

Preliminaries and Lower bound

- Not allowing preemption might be costly

  $\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*

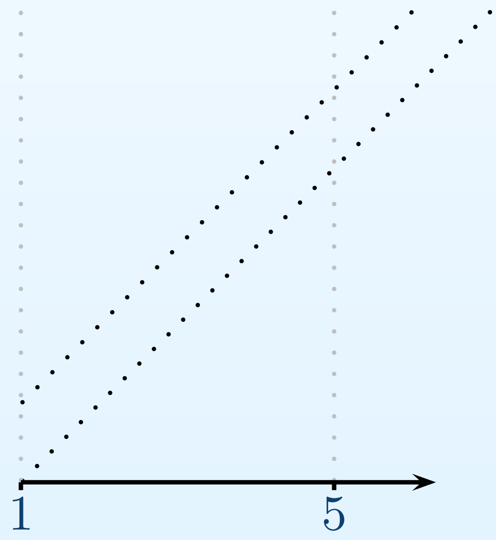
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



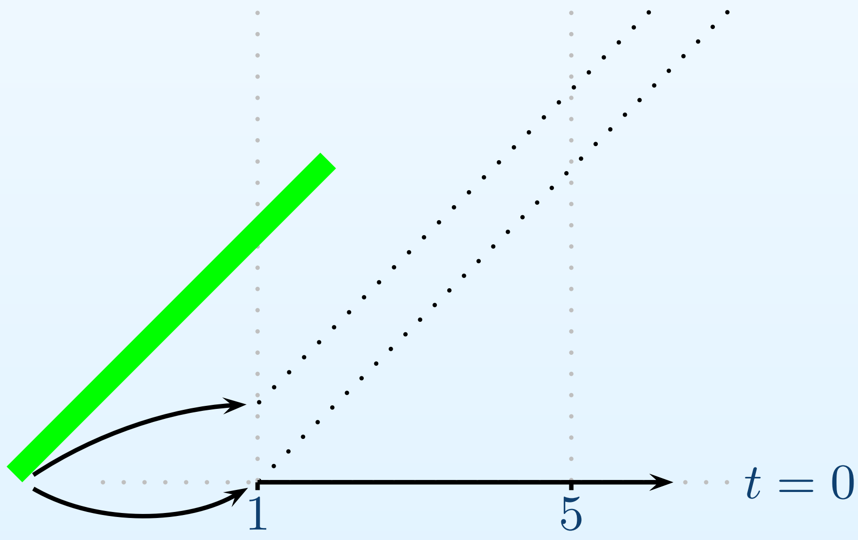
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



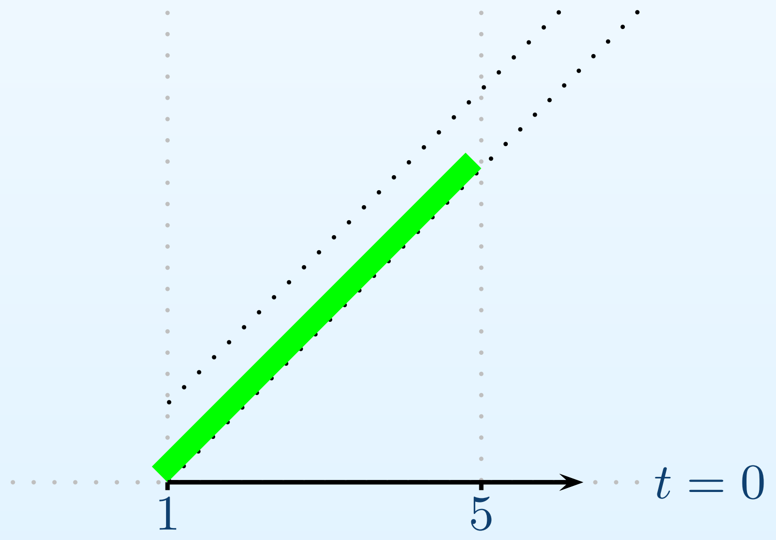
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



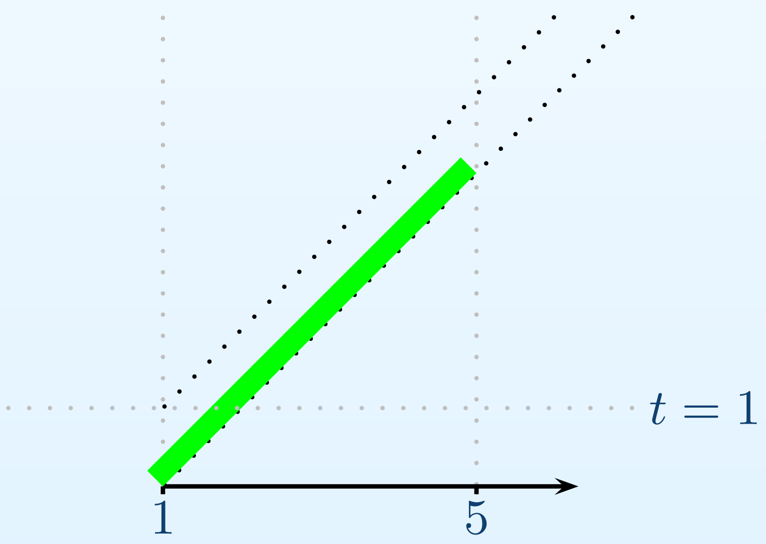
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



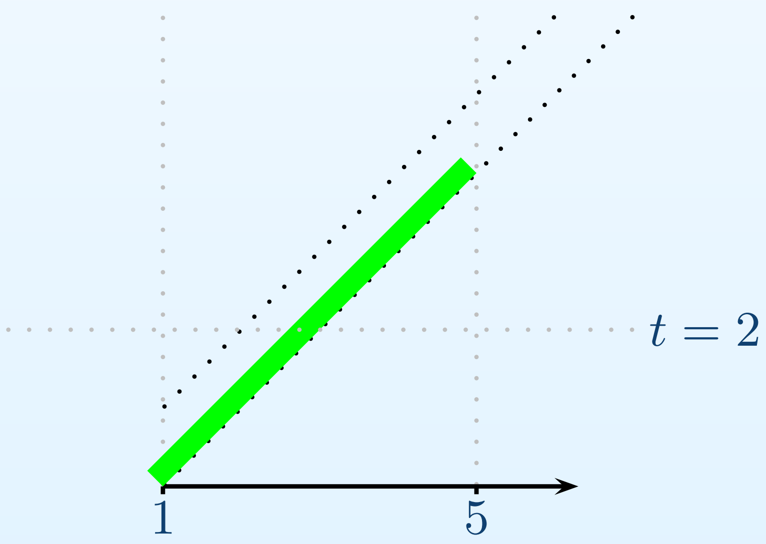
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



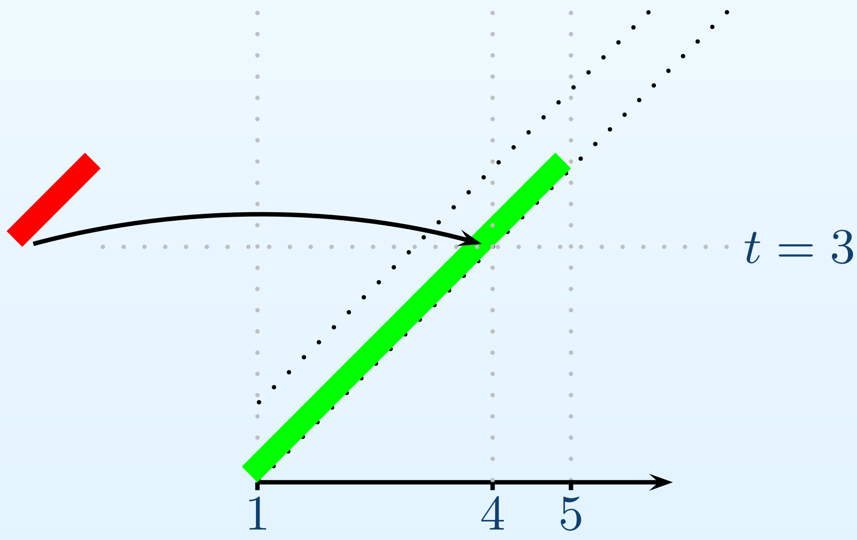
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



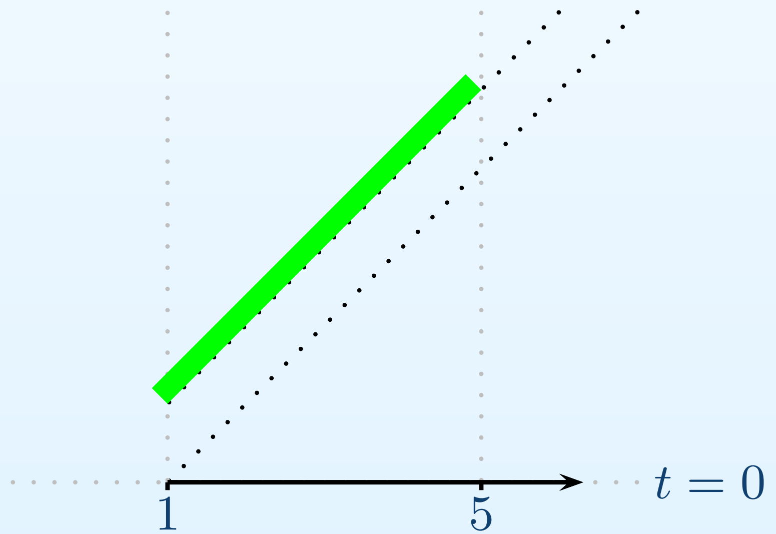
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



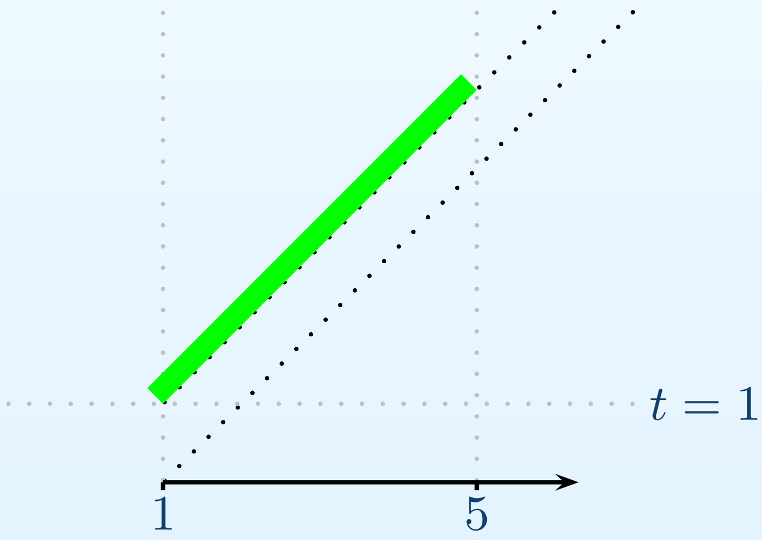
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



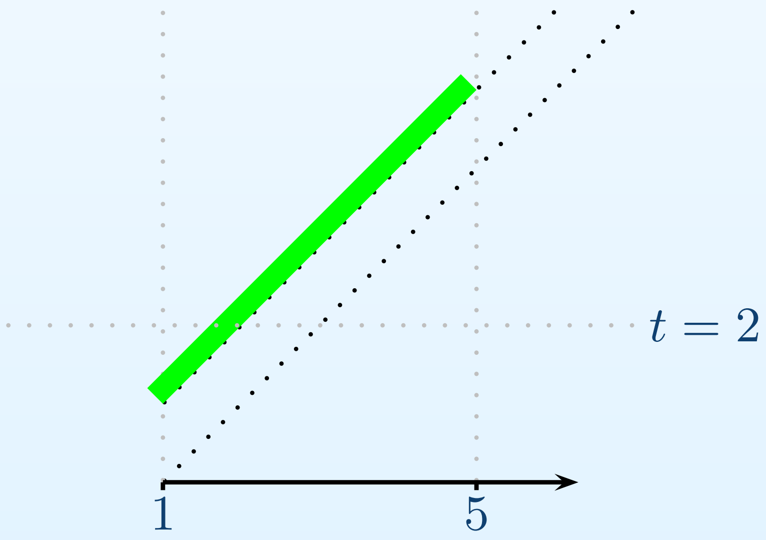
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



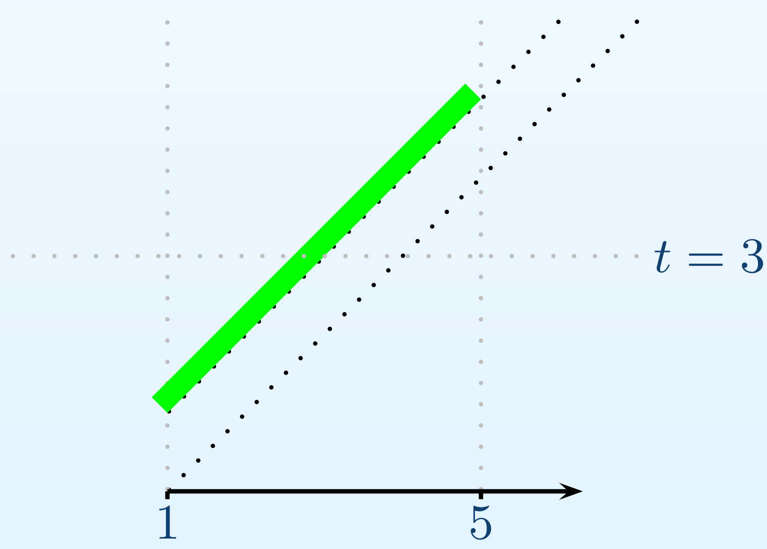
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



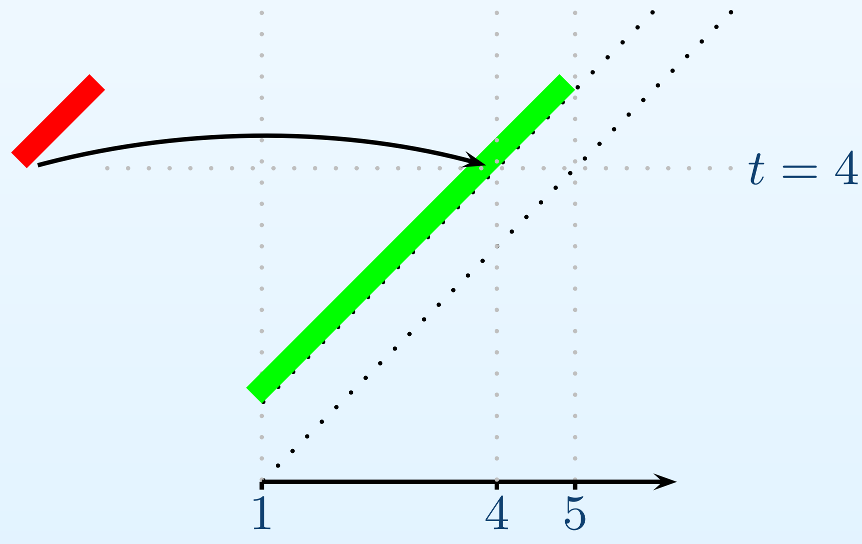
Preliminaries and Lower bound

- Not allowing preemption might be costly

$\Longrightarrow$ **Allow preemption**

- The case where all packets have zero-slack can be solved efficiently.
- What if we allow positive slack?

Theorem. *No deterministic algorithm can achieve a competitive ratio better than 2.*



Algorithm A_{MT}

- Intuition: prefer packets with short paths.

Algorithm A_{MT}

- Intuition: prefer packets with short paths.
- Overview:
  - The algorithm will assign packets to waves.
  - A packet's assignment turns *active* in due time.
  - De-assignment/preemption may occur at any time.

Algorithm A_{MT}

- Intuition: prefer packets with short paths.
- Overview:
  - The algorithm will assign packets to waves.
  - A packet's assignment turns *active* in due time.
  - De-assignment/preemption may occur at any time.
- Algorithm A_{MT}:

Given packet p newly arrived,

1. If p has a free eligible wave c,
   assign p to c.
2. Otherwise,
   Schedule p instead of an assigned q if
   - q is assigned to a wave eligible for p,
   - p and q intersect,
   - $|p| \leq |q|/2$, and
   - $t_p \leq t_q$.

Algorithm A_{MT}

- Intuition: prefer packets with short paths.
- Overview:
  - The algorithm will assign packets to waves.
  - A packet's assignment turns *active* in due time.
  - De-assignment/preemption may occur at any time.
- Algorithm A_{MT}:

Given packet p newly arrived,

1. If p has a free eligible wave c,
   assign p to c.
2. Otherwise,
   Schedule p instead of an assigned q if
   - q is assigned to a wave eligible for p,
   - p and q intersect,
   - $|p| \leq |q|/2$, and
   - $t_p \leq t_q$.

Theorem. A_{MT} *is* $O(\min\{\log \alpha, R\})$*-competitive.*

A_{MT} - Analysis

Let O be an optimal schedule.

- Every packet $q \in O \setminus A_{\mathrm{MT}}$ is mapped to a packet $p \in A_{\mathrm{MT}}$.
- $O(\min\{\log \alpha, R\})$ packets are mapped to any $p \in A_{\mathrm{MT}}$.

A_{MT} - Analysis

Let O be an optimal schedule.

- Every packet $q \in O \setminus A_{\mathrm{MT}}$ is mapped to a packet $p \in A_{\mathrm{MT}}$.
- $O(\min\{\log \alpha, R\})$ packets are mapped to any $p \in A_{\mathrm{MT}}$.

Consider a packet $q \in O \setminus A_{\mathrm{MT}}$.

A_{MT} - Analysis

Let O be an optimal schedule.

- Every packet $q \in O \setminus A_{\mathrm{MT}}$ is mapped to a packet $p \in A_{\mathrm{MT}}$.
- $O(\min\{\log \alpha, R\})$ packets are mapped to any $p \in A_{\mathrm{MT}}$.

Consider a packet $q \in O \setminus A_{\mathrm{MT}}$.

- **Case 1:** q is assigned to some wave c by A_{MT}.

A_{MT} - Analysis

Let O be an optimal schedule.

- Every packet $q \in O \setminus A_{\mathrm{MT}}$ is mapped to a packet $p \in A_{\mathrm{MT}}$.
- $O(\min\{\log \alpha, R\})$ packets are mapped to any $p \in A_{\mathrm{MT}}$.

Consider a packet $q \in O \setminus A_{\mathrm{MT}}$.

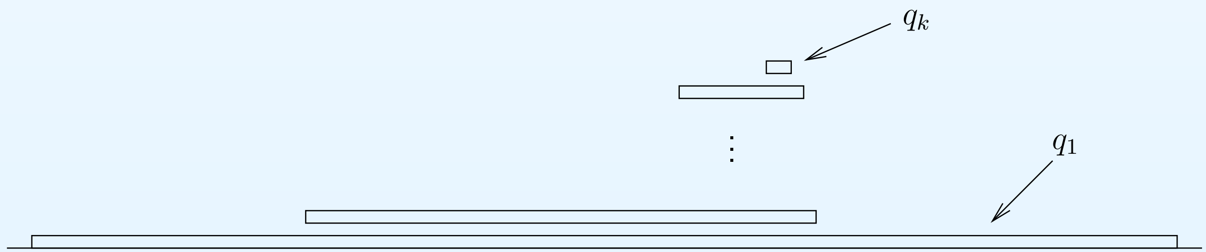
- **Case 1:** q is assigned to some wave c by A_{MT}.
  - q is later de-assigned by some q', de-assigned by some q''...

A_{MT} - Analysis

Let O be an optimal schedule.
- Every packet $q \in O \setminus A_{\mathrm{MT}}$ is mapped to a packet $p \in A_{\mathrm{MT}}$.
- $O(\min\{\log\alpha, R\})$ packets are mapped to any $p \in A_{\mathrm{MT}}$.

Consider a packet $q \in O \setminus A_{\mathrm{MT}}$.

- **Case 1:** q is assigned to some wave c by A_{MT}.
  - q is later de-assigned by some q', de-assigned by some q''...
  - Let $q_1, \ldots, q_k$ be the *preemption sequence* on c.



  - Notice that $|q_k| \leq 2^{-(k-1)}|q_1|$.

    $\Longrightarrow k \leq \log|q_1|/|q_k| + 1 \leq \log M/m + 1 = \log\alpha + 1$

A_{MT} - Analysis (Cont.)

- Clearly $k \leq R$.
- Hence, the length of any preemption sequence is $O(\min\{\log\alpha, R\})$.

A_{MT} - Analysis (Cont.)

- Clearly $k \leq R$.
- Hence, the length of any preemption sequence is $O(\min\{\log\alpha, R\})$.

q_k is sent by A_{MT} – accounts for $O(\min\{\log\alpha, R\})$ such packets

A_{MT} - Analysis (Cont.)

- Clearly $k \leq R$.
- Hence, the length of any preemption sequence is $O(\min\{\log \alpha, R\})$.

q_k is sent by A_{MT} – accounts for $O(\min\{\log \alpha, R\})$ such packets

- **Case 2:** q is never assigned by A_{MT}.

A_{MT} - Analysis (Cont.)

- Clearly $k \leq R$.
- Hence, the length of any preemption sequence is $O(\min\{\log\alpha, R\})$.

q_k is sent by A_{MT} – accounts for $O(\min\{\log\alpha, R\})$ such packets

- **Case 2:** q is never assigned by A_{MT}.
  - Consider the preemption sequence $q_1, \ldots, q_{k'}$ on wave c on which O schedules q.
  - Any q_i prevents an assignment of a packet p if:
    - $|p| > |q_i|/2$: at most 2 packets in O on this account, or
    - $d_p > d_{q_i}$: at most one packet from O on this account.

A_{MT} - Analysis (Cont.)

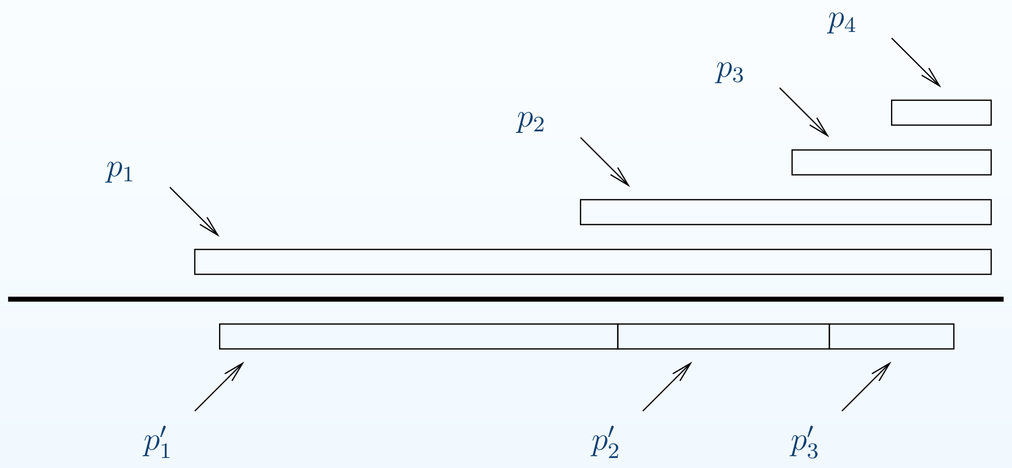
- Clearly $k \leq R$.
- Hence, the length of any preemption sequence is $O(\min\{\log \alpha, R\})$.

q_k is sent by A_{MT} – accounts for $O(\min\{\log \alpha, R\})$ such packets

- **Case 2:** q is never assigned by A_{MT}.
  - Consider the preemption sequence $q_1, \ldots, q_{k'}$ on wave c on which O schedules q.
  - Any q_i prevents an assignment of a packet p if:
    - $|p| > |q_i|/2$: at most 2 packets in O on this account, or
    - $d_p > d_{q_i}$: at most one packet from O on this account.

$q_{k'}$ is sent by A_{MT} – accounts for $O(\min\{\log \alpha, R\})$ such packets

A_{MT} - Tight Example



- All packets have zero slack.
- Non-tagged packets are decoys.
- p'_i is just a little too long to preempt p_i.
- A_{MT} ends up scheduling only the last non-tagged packet.
- There exists a schedule which schedules all tagged packets.

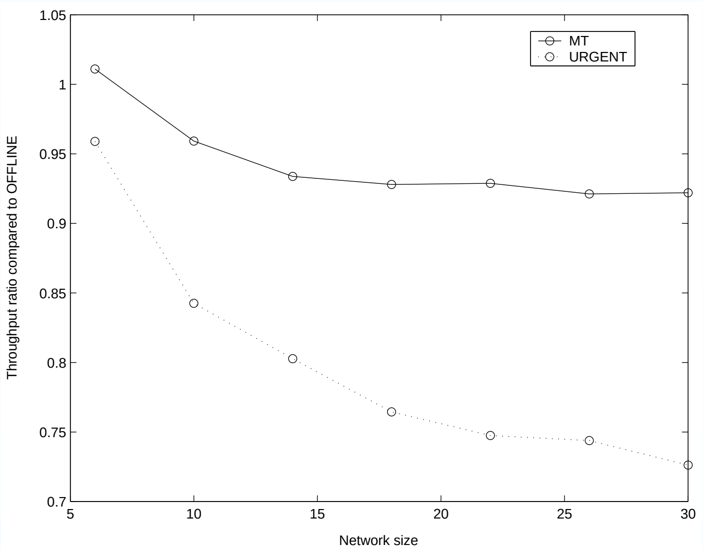
$\Longrightarrow$ A_{MT} is $\Omega(\log n)$-competitive.

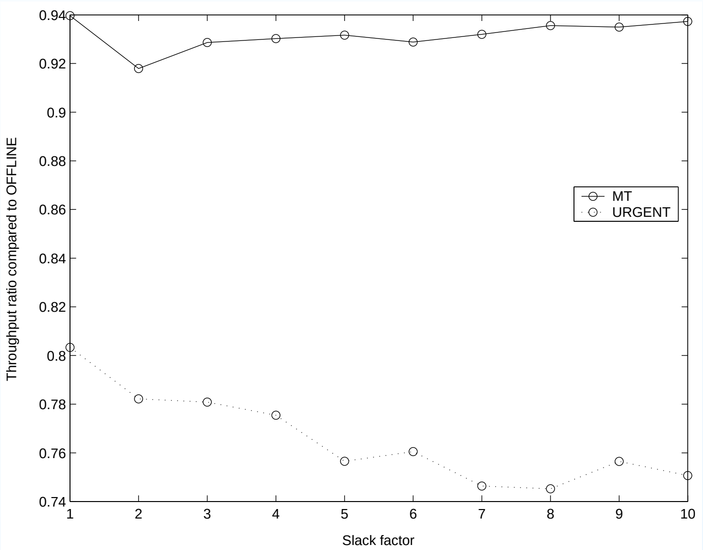
Experimental Results for A_{MT}

- We compared A_{MT} with a natural greedy algorithm A_{URGENT}, on randomly generated input.
- Principles of A_{URGENT}:
  - A packet is *urgent* at time t, if its residual-slack is 0.
  - Prefer the packet with least residual-slack (i.e., most 'urgent').
  - An urgent packet is never preempted.
  - Preempt only in favor of an urgent packet.

Experimental Results for A_{MT} (Cont.)

We evaluated the performance of both algorithms vis-à-vis the offline 2-approximation of [Adler *et al.* (1998)].





- For randomly generated input, A_{MT} performance is close to OFFLINE.
- A_{MT} outperforms the intuitive algorithm which prefers to schedule urgent packets first.

Maximum Network Utilization - Algorithm

(adapted from Garay *et al.*)

- Intuition: prefer packets with long paths.
- Given some wave c and a newly arrived packet p, S_p^c - the set of packets currently assigned to c, intersecting p on c.
- Algorithm A_{MNU}:

> Given packet p newly arrived,
> 1. If p has a free eligible wave c, assign p to c.
> 2. Otherwise, assign p instead of a set of packets S_p^c already assigned to some c eligible for p iff $|p| > \phi \max_{q \in S_p^c} |q|$ (ϕ - the golden ratio).

Theorem. A_{MNU} *is* $(2\phi + 1)$*-competitive.*

Arbitrary Weights - Algorithm

- Assume w.l.o.g. $\rho_{\min} = 1$.
- Define for every set of packets Y,
  - $U(Y) = \sum_{p \in Y} |p|$.
  - $w(Y) = \sum_{p \in Y} w_p$.
- Run A_{MNU}. Let A be the set of packets scheduled.
- Let O_{MNU} (O_{AW}) be some optimal schedule to maximum network utilization (arbitrary weights).
- For c the constant approximation guarantee of A_{MNU},

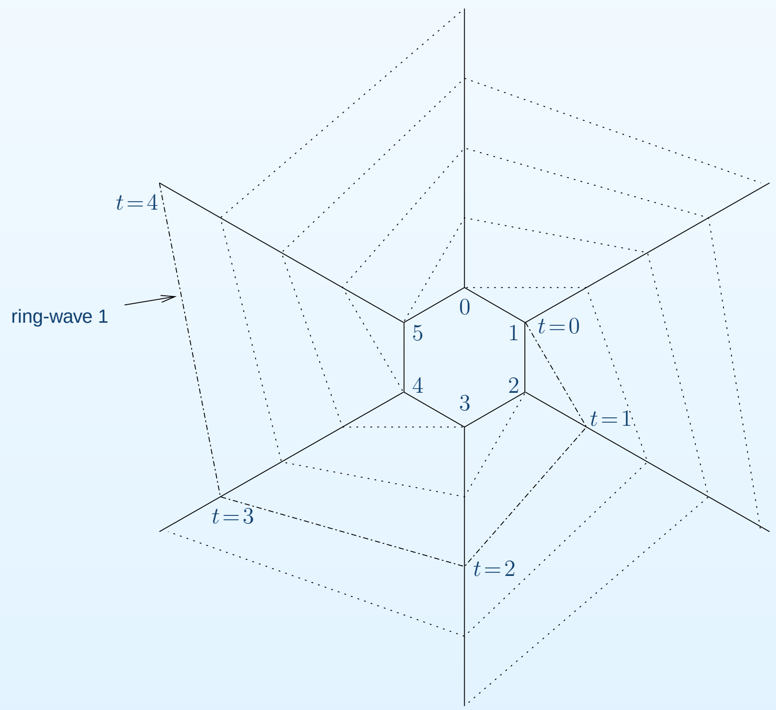
$$w(A) \geq U(A) \geq \frac{1}{c} U(O_{\mathrm{MNU}}) \geq \frac{1}{c\beta} w(O_{\mathrm{AW}}).$$

Theorem. A_{MNU} *is* $O(\beta)$*-competitive.*

Ring Topology

- Results extend to the ring topology.
- Follows from an adequate concept of waves.

	Length of each wave	Number of waves
Line	Finite	Unbounded
Ring	Unbounded	Finite



Conclusions and Future Work

Conclusions

- We give algorithms for online bufferless time-constrained scheduling.
- Our results apply to both linear and ring networks.
- We give analytical results independent of traffic pattern.
- We give experimental results on randomly generated input.

Conclusions and Future Work

Conclusions

- We give algorithms for online bufferless time-constrained scheduling.
- Our results apply to both linear and ring networks.
- We give analytical results independent of traffic pattern.
- We give experimental results on randomly generated input.

Future Work

- Closing the gap between the LB and the UB for the problem of Throughput Maximization.
- Can rescheduling help?