The Online Metric Matching Problem for Doubling Metrics

Anupam Gupta[1] **Kevin Lewi**[2]

[1]Carnegie Mellon University
[2]Stanford University

ICALP 2012

Outline

Outline

The Problem

Setting:

- k servers are placed in a metric space
- k requests appear anywhere on the metric space, and each must be matched to an unassigned server.

Goal:

- minimize the cost of the matching

Assignment Rules:

- the requests come one at a time, and we must match each request up as soon as they arrive
- as soon as a request is matched to some server, it can never be reassigned

An Example

● = requests

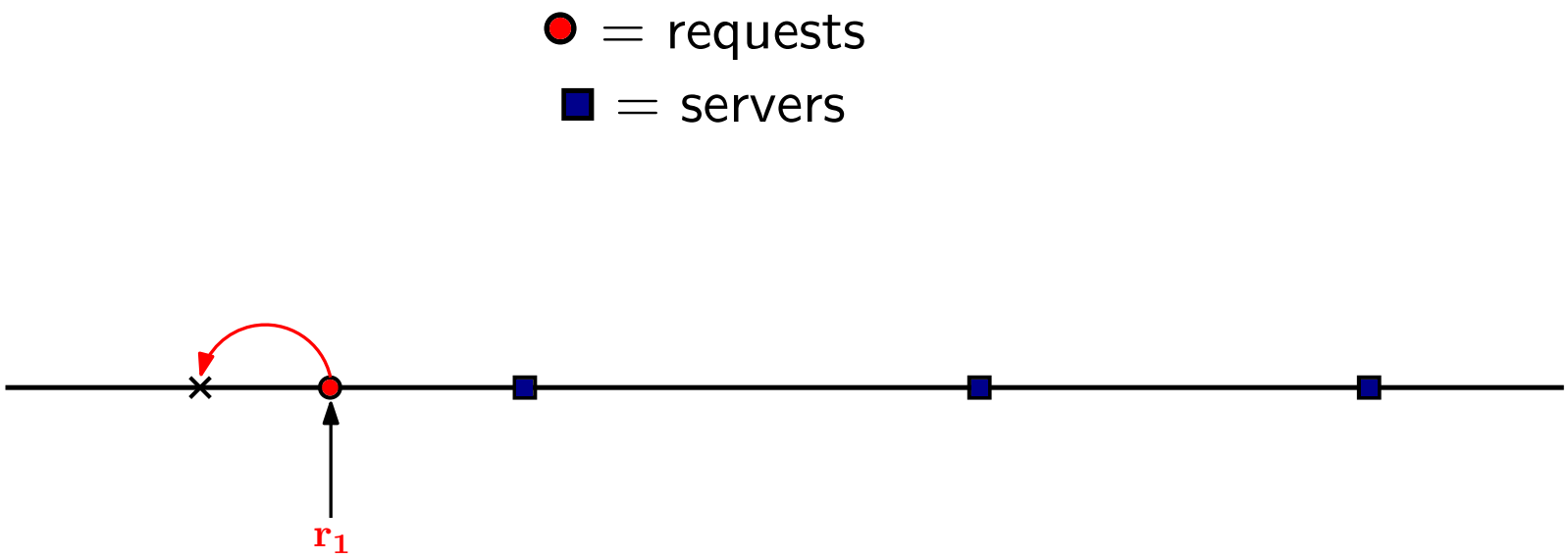
■ = servers

An Example

● = requests

■ = servers

An Example

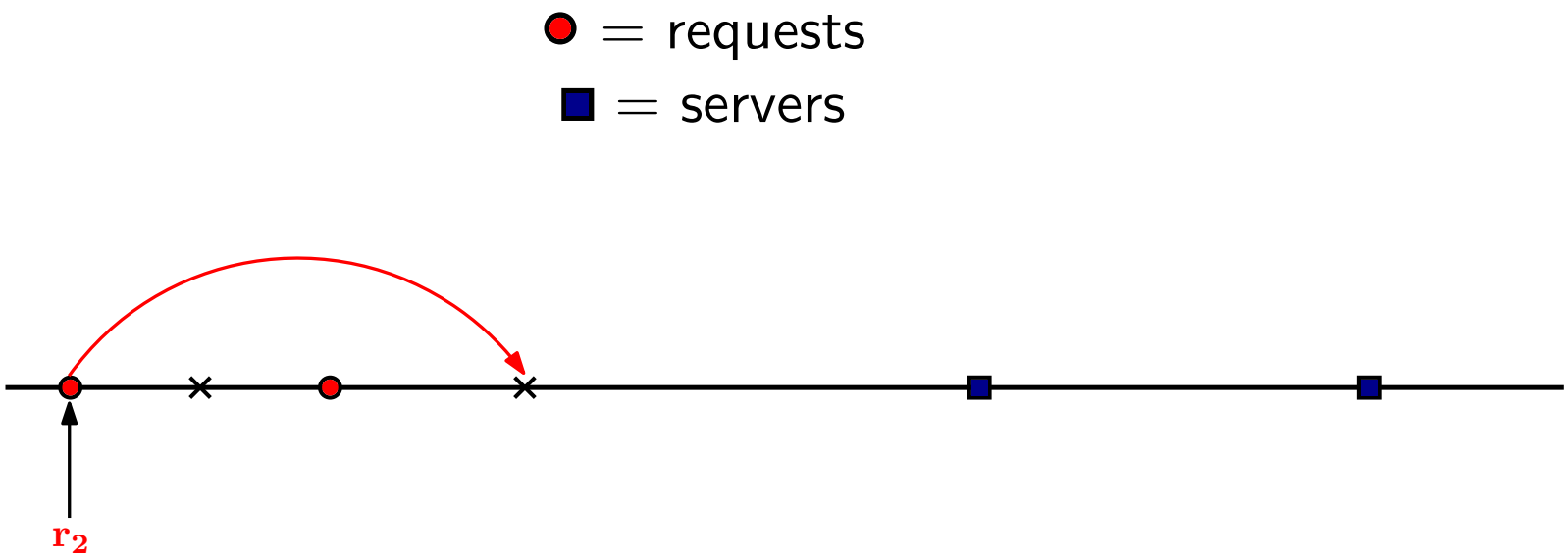


An Example

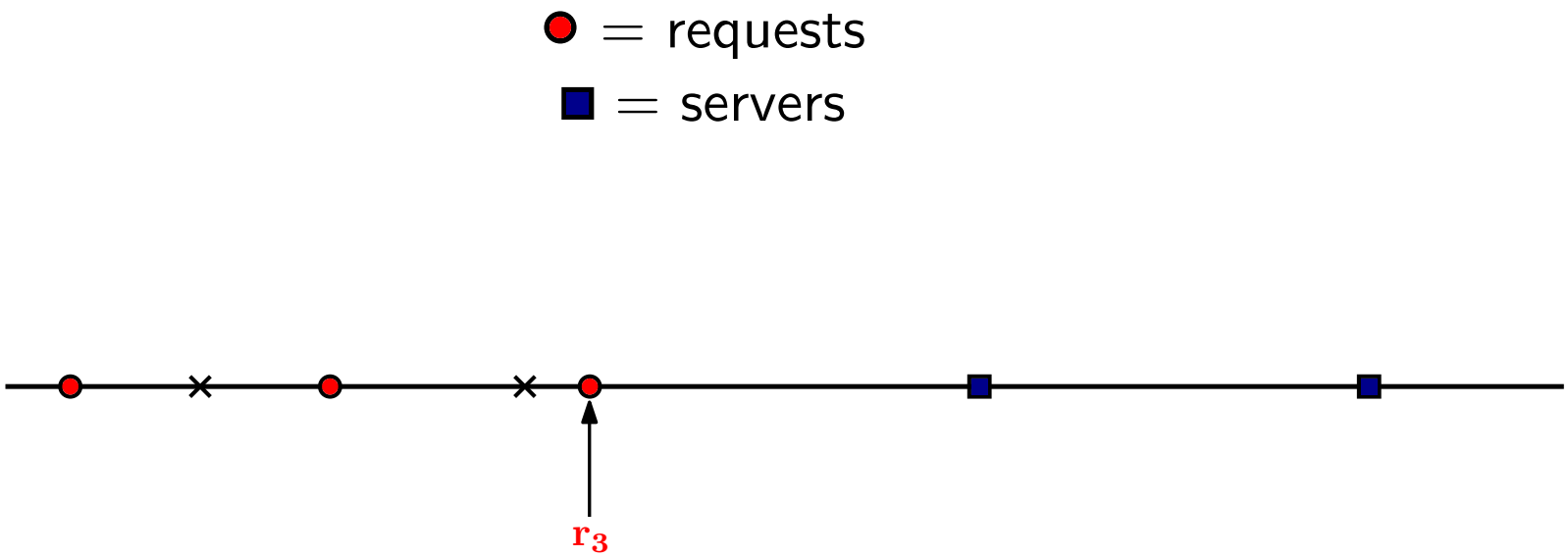
● = requests

■ = servers

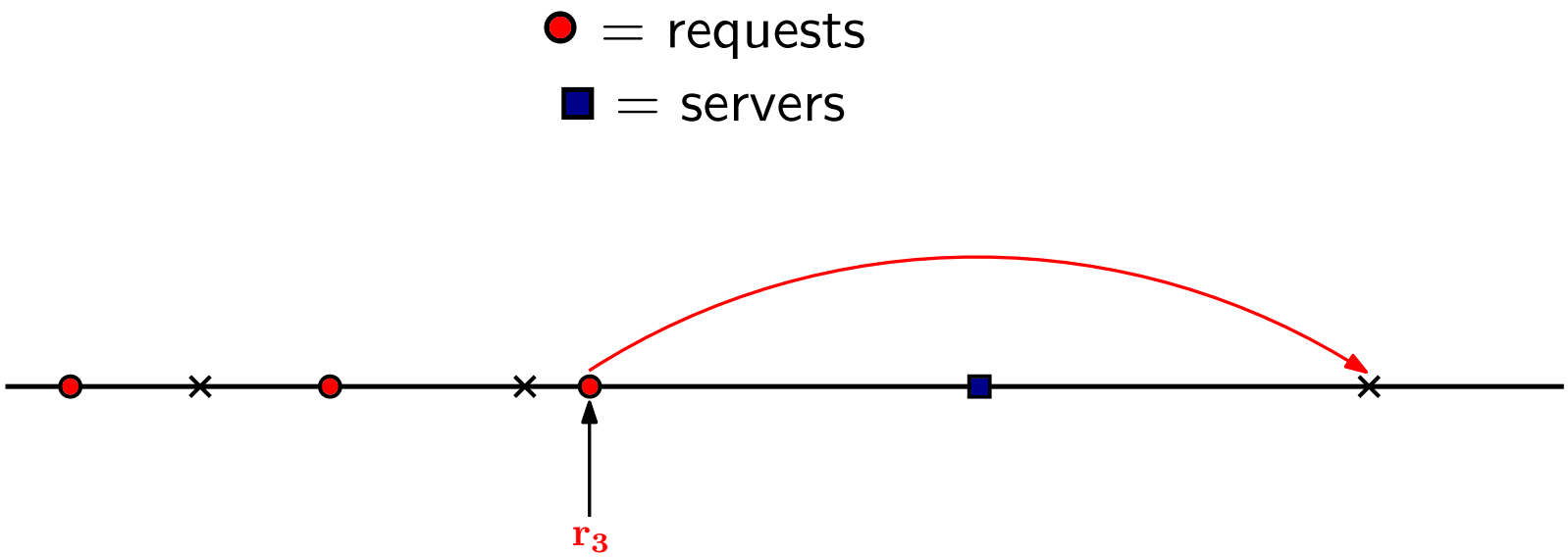
An Example



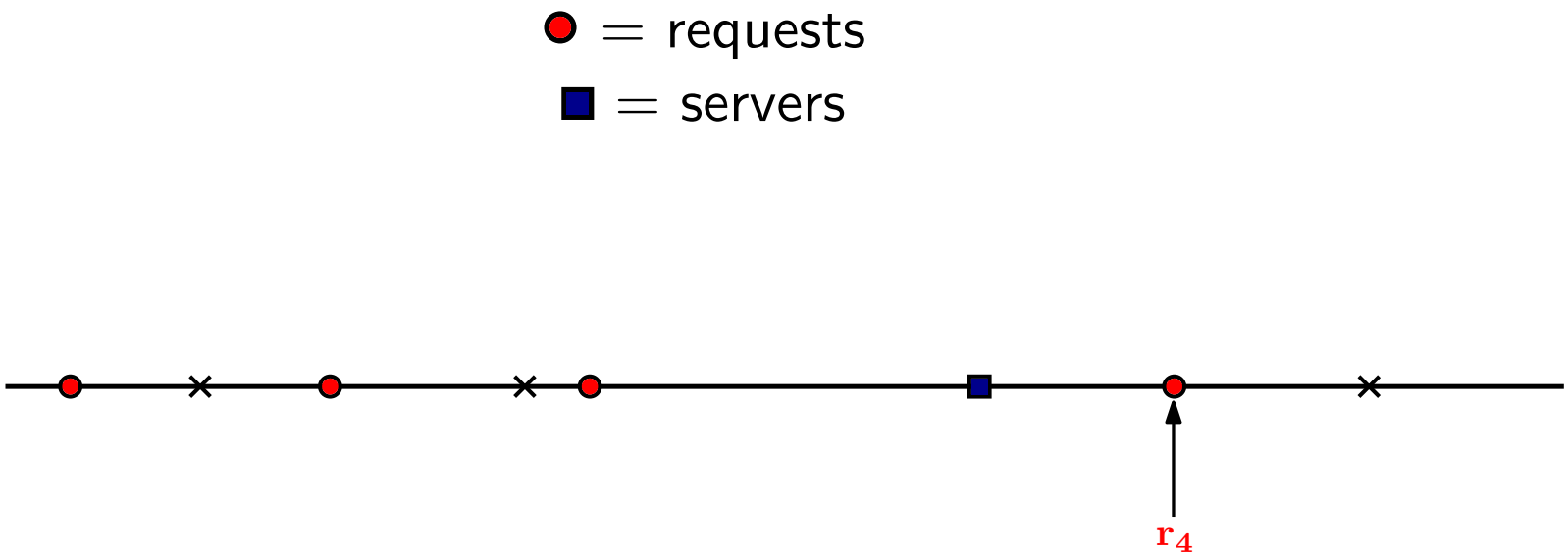
An Example



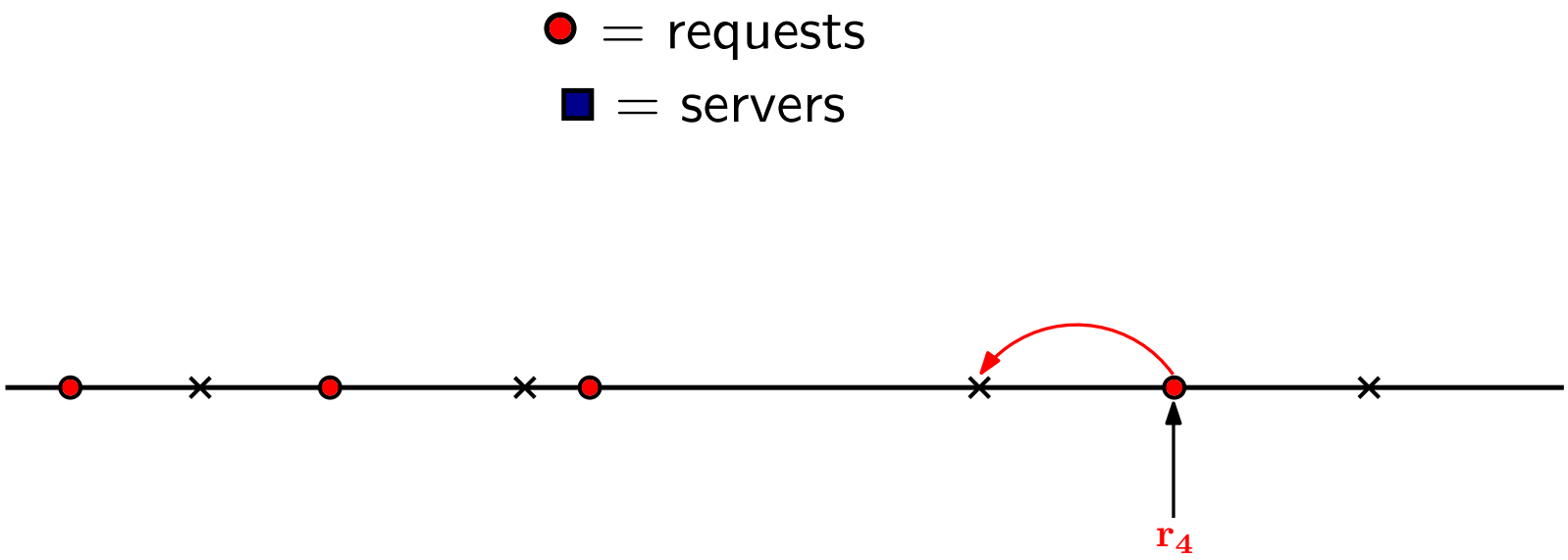
An Example



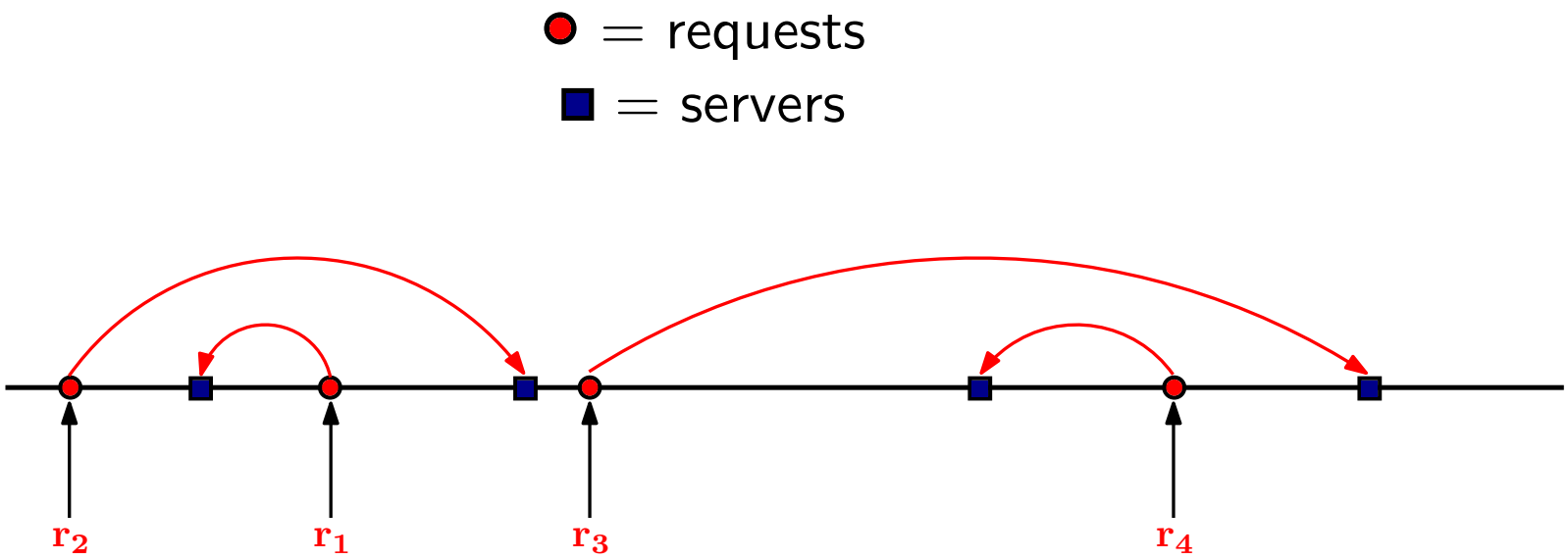
An Example



An Example

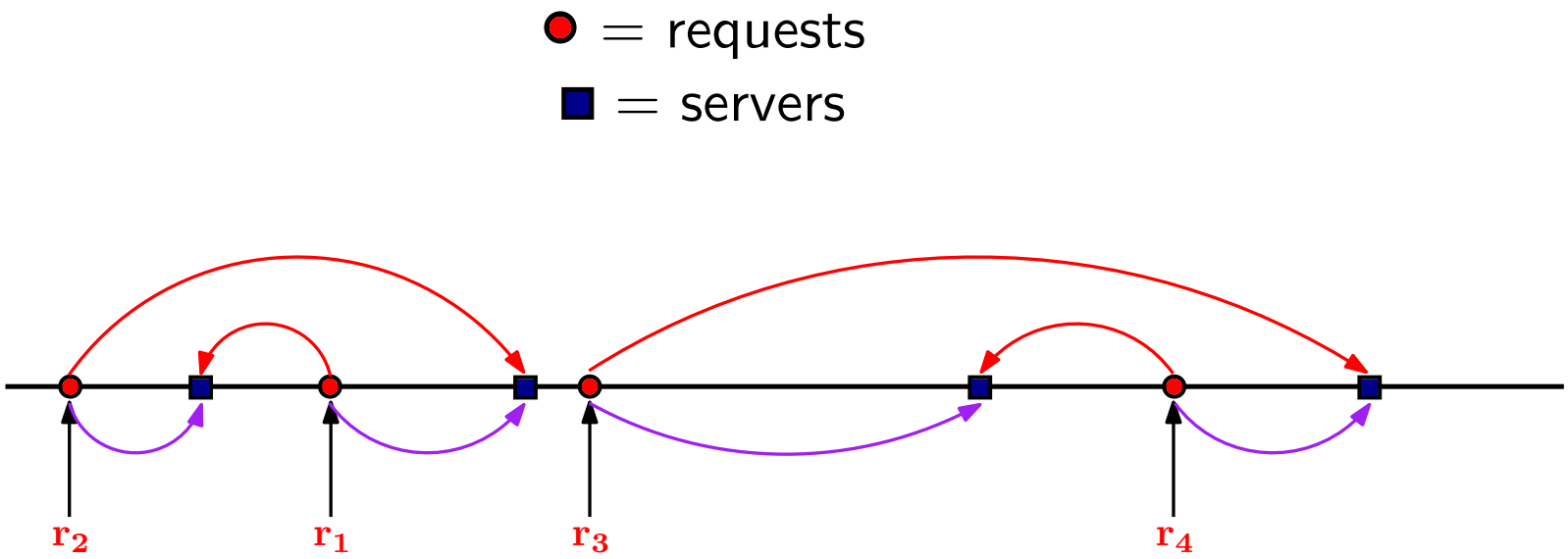


An Example



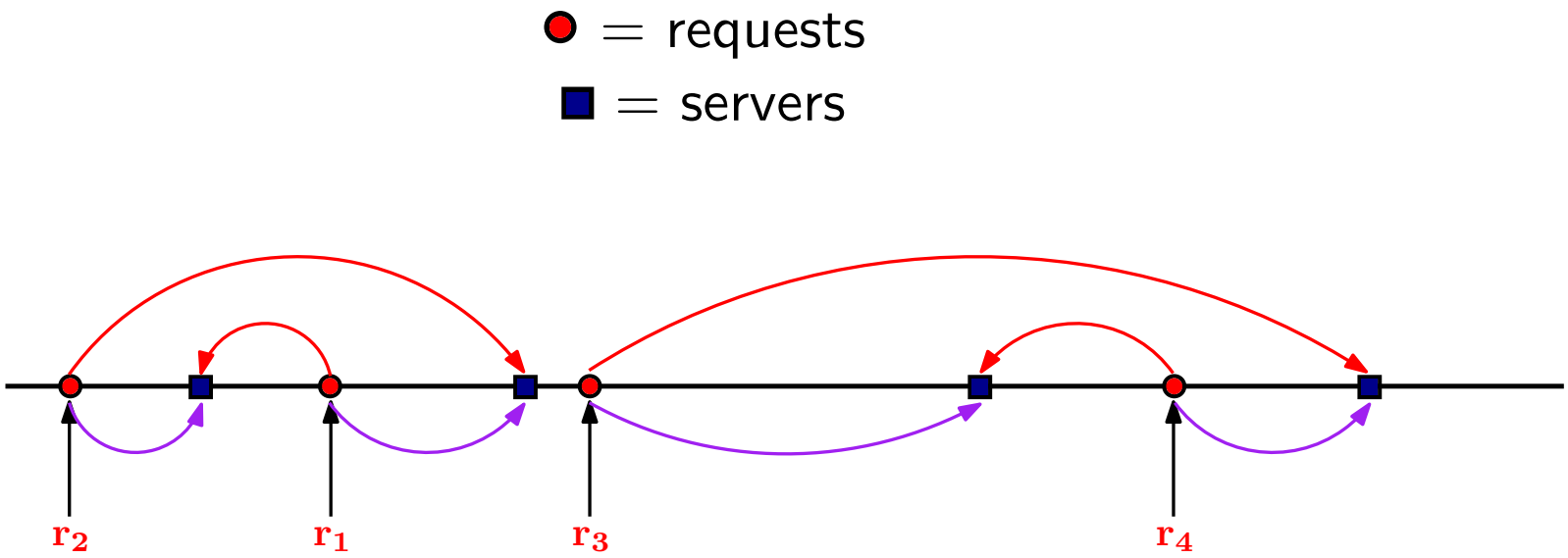
Total Cost: sum over all distances traveled in the matching

An Example



Total Cost: sum over all distances traveled in the matching

An Example



Competitive Ratio: how much we travel / how much OPT travels

The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

The Greedy Algorithm

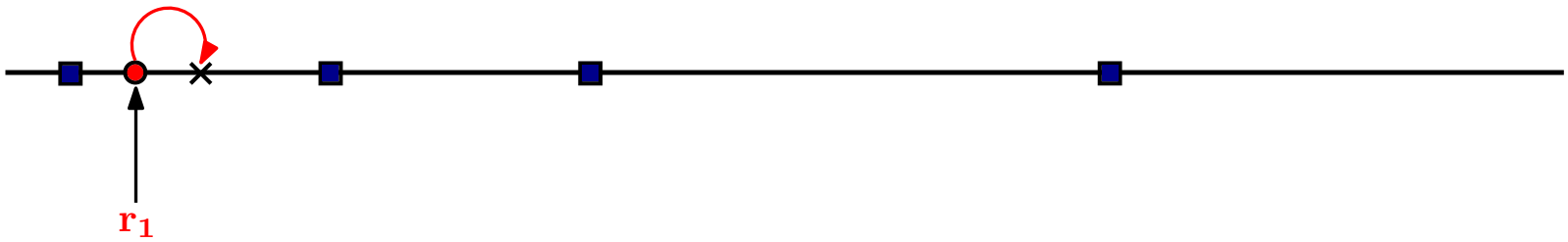
Greedy Approach: "match the request to its closest available server"

The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

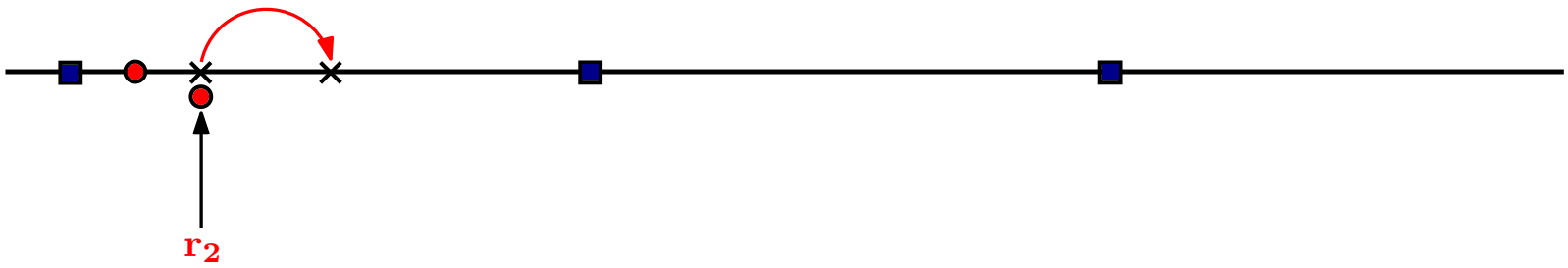


The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"



The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

The Greedy Algorithm

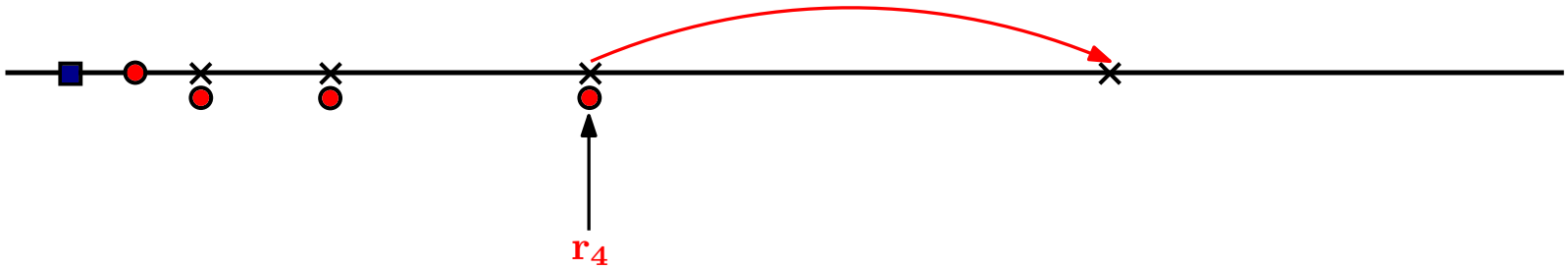
Greedy Approach: "match the request to its closest available server"

The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

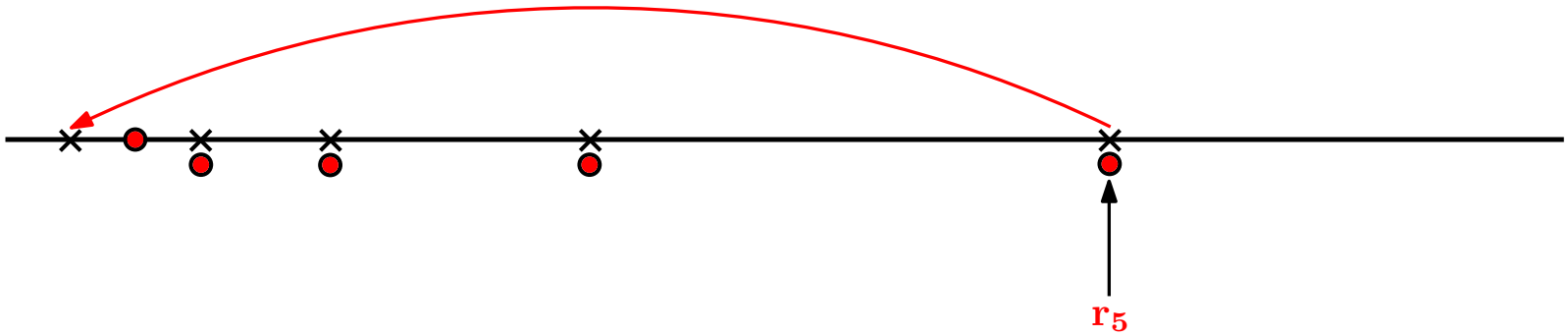


The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

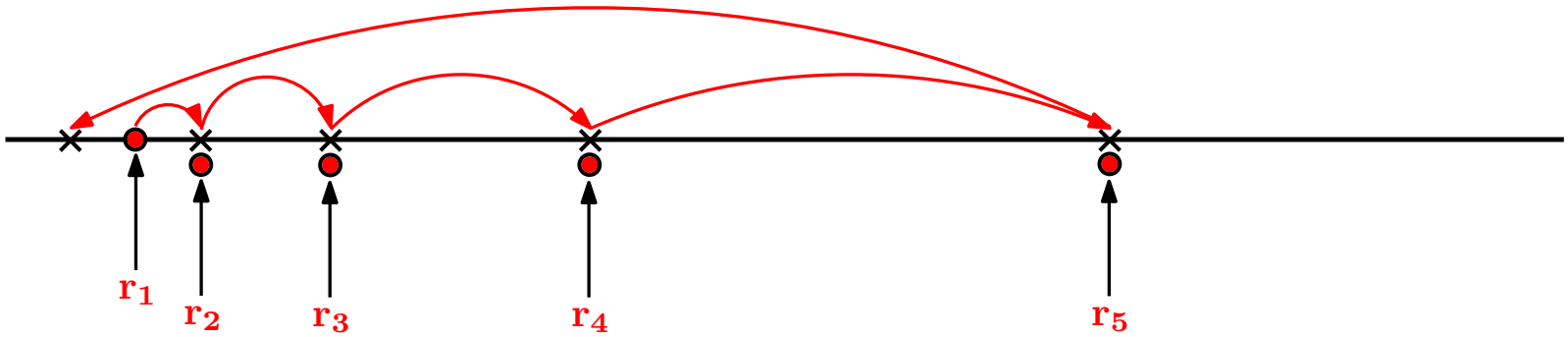
The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"



The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

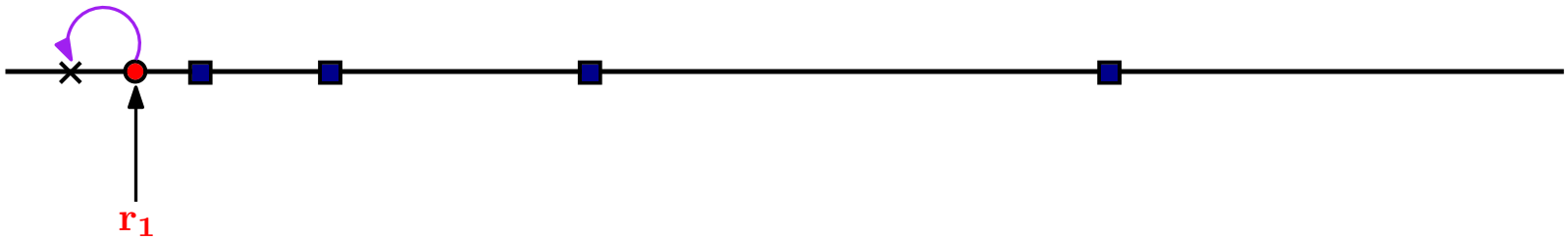


The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

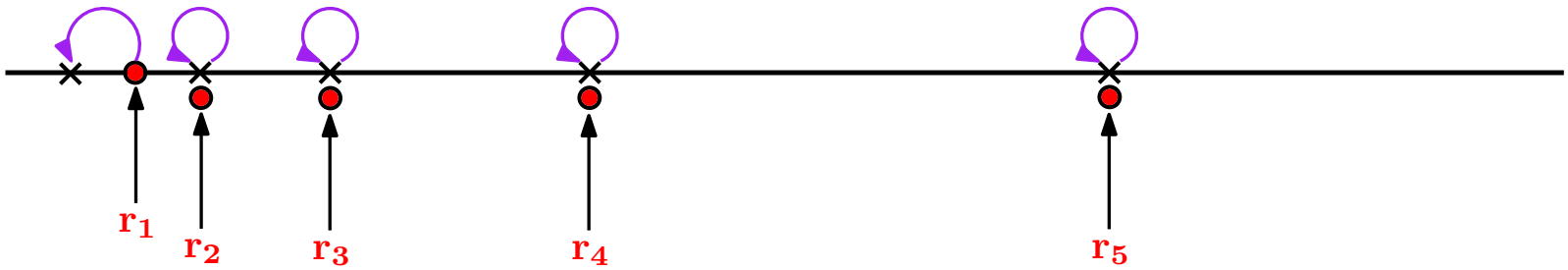
The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"



The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"



The Greedy Algorithm

Greedy Approach: "match the request to its closest available server"

Unfortunately, Greedy has competitive ratio $\Omega(2^k)$.

Outline

Bounds

The best deterministic upper bound for this problem is:

$$\text{competitive ratio} \leq 2k - 1$$

from [Kalyanasundaram, Pruhs 1993], and also [Khuller, Mitchell, V. Vazirani 1994]

Randomization

But the $2k - 1$ bound applies to deterministic algorithms. Can randomization help?

Tightest upper bound for randomized algorithms for this problem:

$$\text{competitive ratio} \leq O(\log^2 k)$$

by Bansal et al. in 2007.

Our Results

- For the line, we give a simple $O(\log k)$ competitive randomized algorithm.
- For metrics with constant doubling dimension (ex: $\mathbb{R}^2$, $\mathbb{R}^3$), we give an $O(\log k)$ competitive randomized algorithm.

Outline

The Harmonic Algorithm

The "Harmonic" Algorithm: As each request r arrives,

The Harmonic Algorithm

The "Harmonic" Algorithm: As each request r arrives,

- Let s_L be the closest server to the left of r

The Harmonic Algorithm

The "Harmonic" Algorithm: As each request r arrives,

- Let s_L be the closest server to the left of r
- Let s_R be the closest server to the right of r

The Harmonic Algorithm

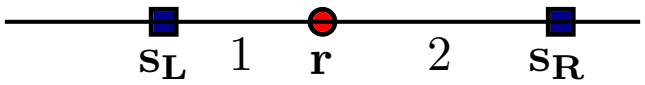
The "Harmonic" Algorithm: As each request r arrives,

- Let s_L be the closest server to the left of r
- Let s_R be the closest server to the right of r
- Assign to s_L with probability $\frac{d(r,s_R)}{d(s_L,s_R)}$ and assign to s_R otherwise.

The Harmonic Algorithm

The "Harmonic" Algorithm: As each request r arrives,

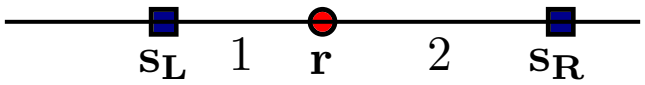
- Let s_L be the closest server to the left of r
- Let s_R be the closest server to the right of r
- Assign to s_L with probability $\frac{d(r,s_R)}{d(s_L,s_R)}$ and assign to s_R otherwise.



The Harmonic Algorithm

The "Harmonic" Algorithm: As each request r arrives,

- Let s_L be the closest server to the left of r
- Let s_R be the closest server to the right of r
- Assign to s_L with probability $\frac{d(r,s_R)}{d(s_L,s_R)}$ and assign to s_R otherwise.



In this example, assign to s_L with probability:

$$\frac{d(r, s_R)}{d(s_L, s_R)} = \frac{2}{3}$$

Performance of Harmonic

- Define the aspect ratio of a metric space to be

$$\text{aspect ratio} = \frac{d_{max}}{d_{min}} = \frac{\max_{s,s'} d(s,s')}{\min_{s,s'} d(s,s')}$$

- We can show that Harmonic achieves a competitive ratio of $O(\log(\text{aspect ratio}))$

- Then, we can show that there is a preprocessing technique (a "guess-and-double" procedure) which allows us to assume that the line always has aspect ratio at most $O(k^3)$.

Performance of Harmonic

- Define the aspect ratio of a metric space to be

$$\text{aspect ratio} = \frac{d_{max}}{d_{min}} = \frac{\max_{s,s'} d(s,s')}{\min_{s,s'} d(s,s')}$$

- We can show that Harmonic achieves a competitive ratio of $O(\log(\text{aspect ratio}))$

- Then, we can show that there is a preprocessing technique (a "guess-and-double" procedure) which allows us to assume that the line always has aspect ratio at most $O(k^3)$.

$\Rightarrow$ We have an $O(\log k)$ competitive algorithm for the line

Outline

Tree Metrics

- A tree metric is a metric space that can be "embedded" into a tree (all points lie at the leaves of the tree)
- Distances between points in the metric space correspond to the lengths of the paths between the points in the tree.
- There exists a way [FRT03] to transform any arbitrary metric space into a tree-based one, where distances are stretched by at most $O(\log k)$.

Our Techniques

- Idea [BBGN07, MNP06]: transform metric space into a tree, run an algorithm on the tree, map the solution back to the original metric space.

Our Techniques

- Idea [BBGN07, MNP06]: transform metric space into a tree, run an algorithm on the tree, map the solution back to the original metric space.
- From [MNP06]: transform metric into tree, then run $O(\log^2 k)$ competitive algorithm on tree

Our Techniques

- Idea [BBGN07, MNP06]: transform metric space into a tree, run an algorithm on the tree, map the solution back to the original metric space.
- From [MNP06]: transform metric into tree, then run $O(\log^2 k)$ competitive algorithm on tree
- From [BBGN07]: transform metric into tree, then run $O(\log k)$ competitive algorithm on tree

Our Techniques

- Idea [BBGN07, MNP06]: transform metric space into a tree, run an algorithm on the tree, map the solution back to the original metric space.
- From [MNP06]: transform metric into tree, then run $O(\log^2 k)$ competitive algorithm on tree
- From [BBGN07]: transform metric into tree, then run $O(\log k)$ competitive algorithm on tree
- Our techniques: transform metric into d-ary tree, then run $O(\log d)$ competitive algorithm on tree

Our Techniques

- This yields a competitive ratio of $O(\log k \log d)$ for arbitrary metric spaces.
- Metric spaces with doubling dimension d (like $\mathbb{R}^d$) can be converted into d-ary trees

Outline

Gaps and Open Questions

- Online Metric Matching for General Metrics:
  Lower Bound: $\Omega(\log k)$
  Upper Bound: $O(\log^2 k)$

Gaps and Open Questions

- Online Metric Matching for General Metrics:
  Lower Bound: $\Omega(\log k)$
  Upper Bound: $O(\log^2 k)$

- Online Metric Matching for the Line:
  Lower Bound: $\Omega(1)$
  Upper Bound: $O(\log k)$

Gaps and Open Questions

- Online Metric Matching for General Metrics:
  Lower Bound: $\Omega(\log k)$
  Upper Bound: $O(\log^2 k)$

- Online Metric Matching for the Line:
  Lower Bound: $\Omega(1)$
  Upper Bound: $O(\log k)$

- (Deterministic) Online Metric Matching for the Line:
  Lower Bound [BCR93]: 9
  Upper Bound [KP93]: $2k - 1$

Gaps and Open Questions

- Online Metric Matching for General Metrics:
  Lower Bound: $\Omega(\log k)$
  Upper Bound: $O(\log^2 k)$

- Online Metric Matching for the Line:
  Lower Bound: $\Omega(1)$
  Upper Bound: $O(\log k)$

- (Deterministic) Online Metric Matching for the Line:
  Lower Bound [F05]: $9 + \epsilon$
  Upper Bound [KP93]: $2k - 1$