

The Delayed k-Server Problem

Wolfgang Bein

Center for the Advanced Study of Algorithms
School of Computer Science
University of Nevada, Las Vegas

August 2005



with: Kazuo Iwama (Kyoto University)

Lawrence Larmore (University of Nevada)

John Noga (California State University, Northridge)

Grant support: NSF CCR-0312093

The k -Server Problem

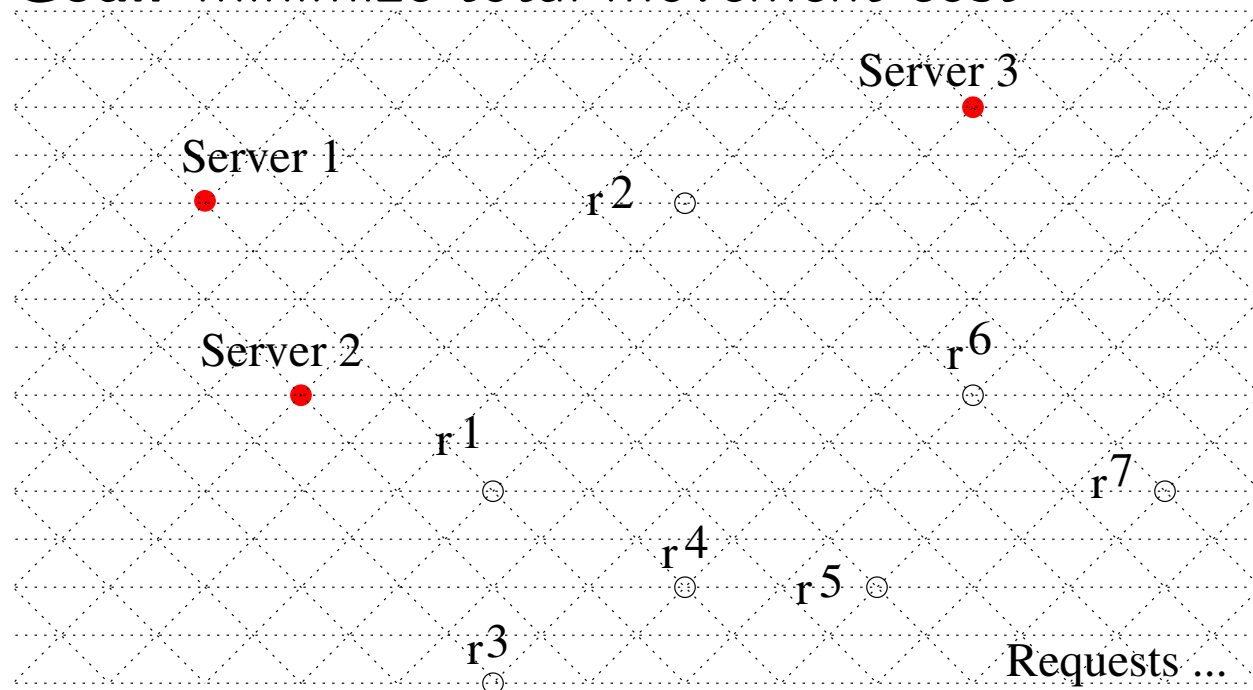
Schedule the movement of k servers in a metric space M in response to a sequence ρ of requests

requests $\rho = r^1 r^2 \dots r^n$, points in M

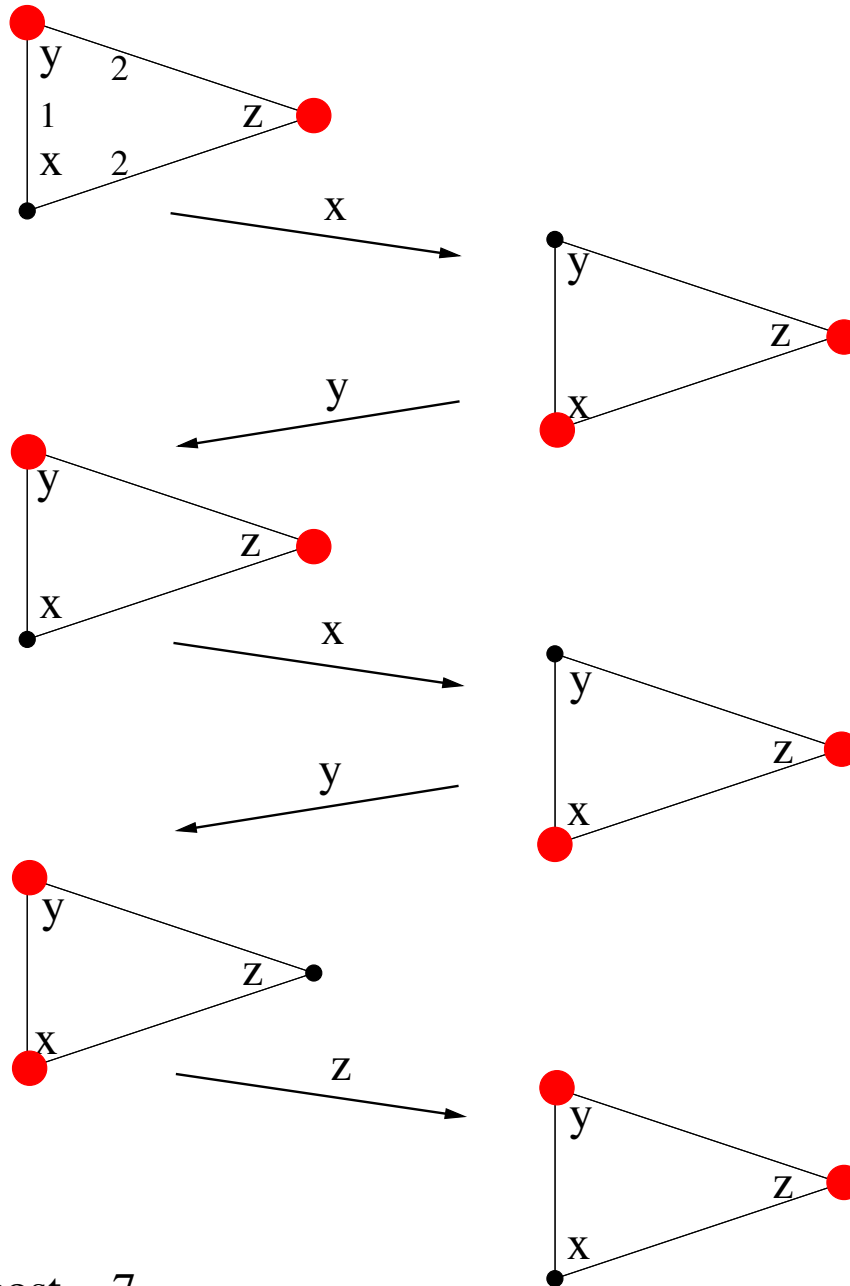
after r^i : move one server to r^i

online: decision must be made before r^{i+1} is revealed

Goal: minimize total movement cost

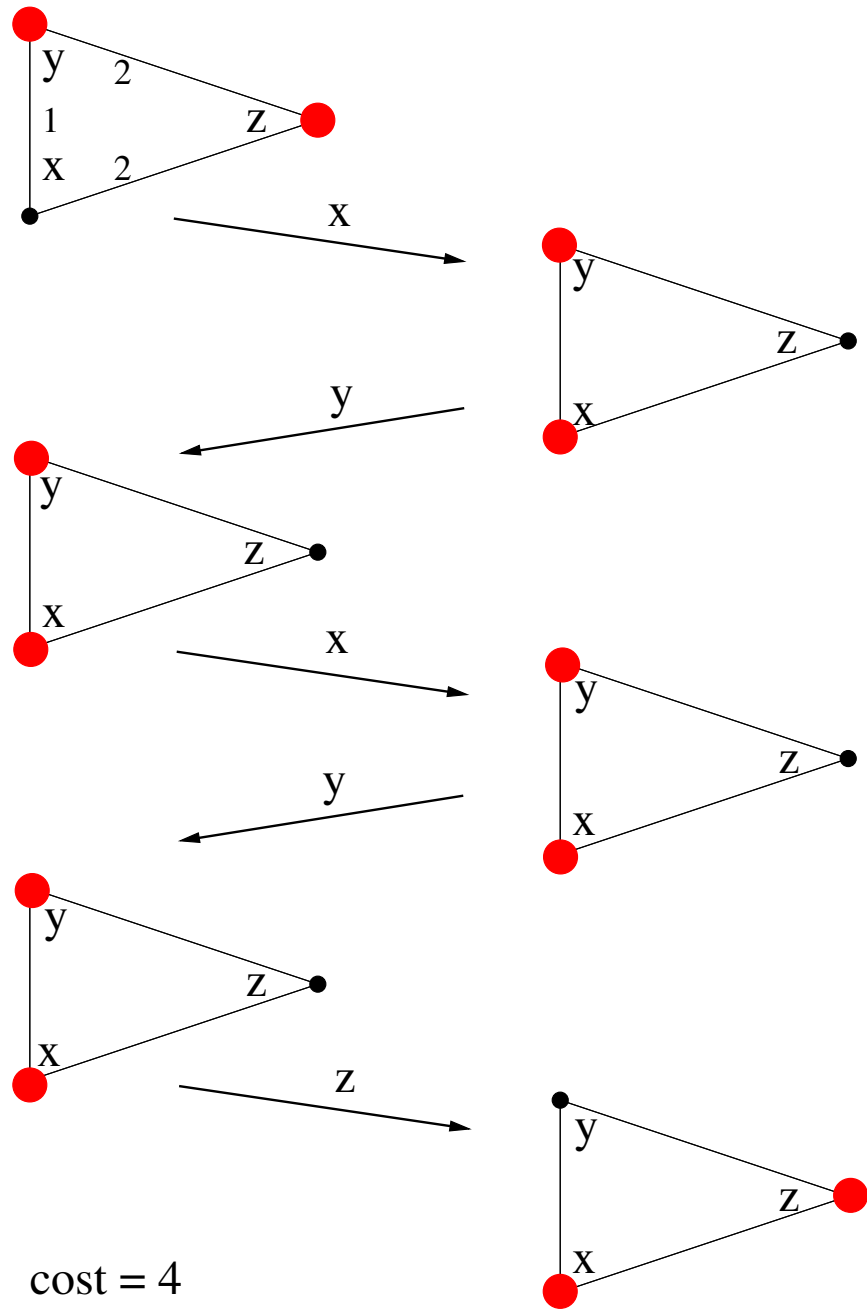


Example: $k = 2$ and $\rho = xyxyz$



cost = 7

Better schedule:



... optimal, as it turns out

Competitiveness

For request sequence $\varrho = r^1, r^2, \dots$ consider

$cost_{\mathcal{A}}(\varrho)$: the cost on ϱ achieved by the servers of our online algorithm \mathcal{A}

$cost_{opt}(\varrho)$: the cost on ϱ achieved by the server of an optimal offline algorithm

We say that our algorithm is *C-competitive* if for each sequence ϱ we have

$$cost_{\mathcal{A}}(\varrho) \leq C \cdot cost_{opt}(\varrho) + \lambda$$

λ depends only on the initial configuration

- The *competitive ratio* of \mathcal{A} is the smallest C for which \mathcal{A} is C -competitive
- The *competitiveness* of any online problem is then defined to be the smallest competitive ratio of any online algorithm for that problem

Manasse, McGeoch, Sleator, 1988:

- 2-competitive algorithm for $k = 2$
- k -competitive for $k + 1$ points
- Lower bound of k
- The k -Server Conjecture: upper bound = k

Chrobak, Larmore, 1991:

- Upper bound k for trees

Koutsoupias, Papadimitriou, 1994:

- Upper bound $2k - 1$
- Upper bound k for $k + 2$ points

Many other results in literature ...

Delayed Server Problem

If a server serves the request at time t , it must stay at that request point until time $t + 2$.

However, if $r^{t+1} = r^t$, the server may serve the request at time $t + 1$.

Version Issues

We could also consider "Version (b)"

If a server is used to serve the request at time t , it cannot be used to serve the request at time $t + 1$.

Theorem:

Competitiveness(Version (a)) \leq Competitiveness(Version (b))

A first (false) intuition

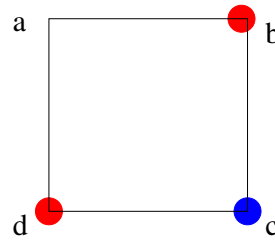
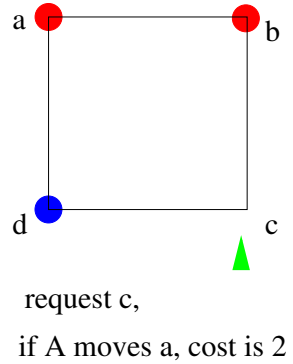
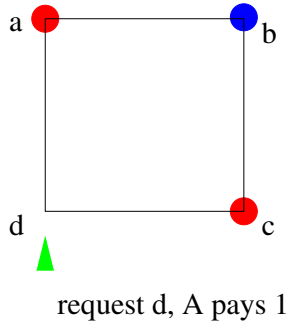
- Delayed k -Server Problem $\equiv (k - 1)$ -Server Problem

FALSE

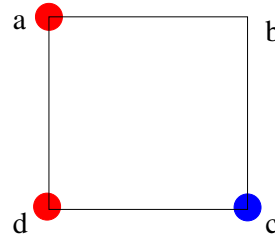
WHY?

- This would imply a competitiveness of 2 for the Delayed 3-Server Problem
- We show a lower bound of 3 for the Delayed 3-Server Problemon the next slide

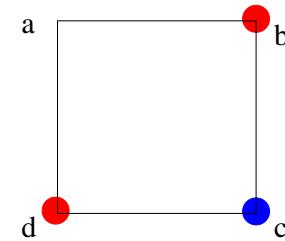
A Lower Bound of 3 for the Delayed 3-Server Problem



if A moves b



request (dbc) many times



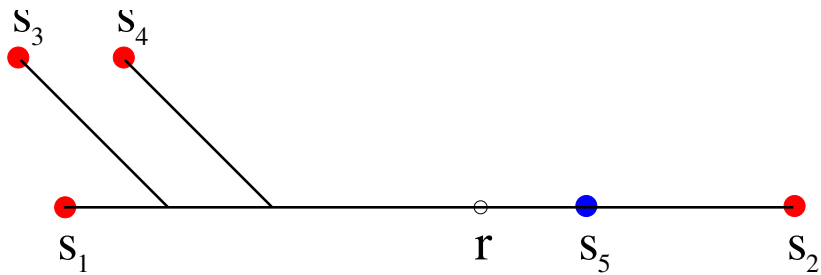
- \mathcal{A} pays 3 no matter what

- opt can serve at cost 1

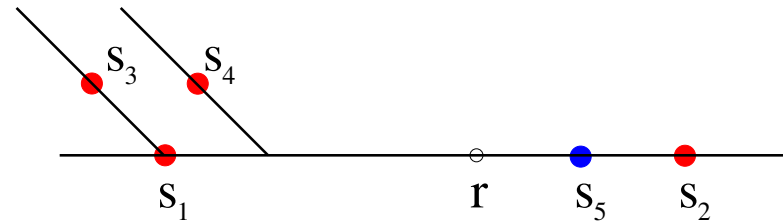
The Tree Algorithm

based on Chrobak, Larmore [1991]

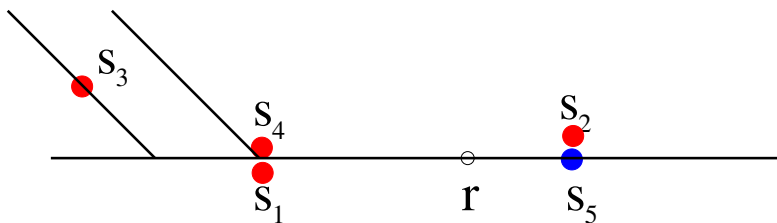
Slogan: "All fire trucks rush to the scene, unless they are blocked by another fire truck"



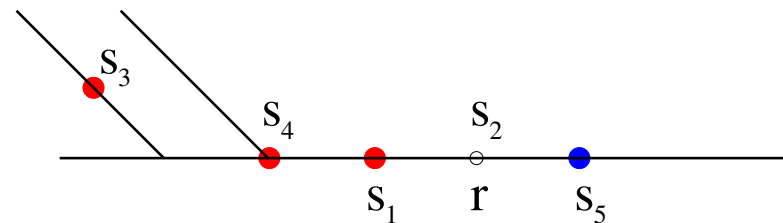
(a) Beginning of Step



(b) After first Phase



(c) After second Phase



(d) End of Step

Theorem: The tree algorithm is k -competitive for the delayed k -server problem in a tree T

Proof: Use the *Coppersmith-Doyle-Raghavan-Snir* potential

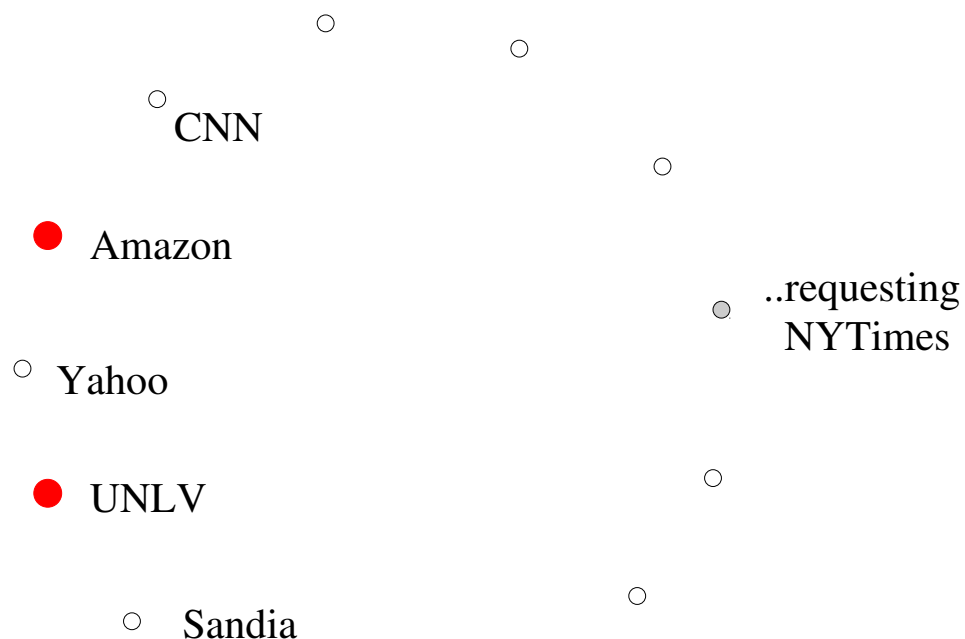
Theorem: If a metric space M can be embedded into a continuous tree T , then the delayed k -server problem is k -competitive in M .

Proof: Keep virtual locations

Caching

- Universe of pages in “slow” memory
- Fast memory can hold k pages
- Online requests for pages have to be served
- Hit (no cost) or Miss (cost 1 to bring in the new page, and evict the old one)

Reduction to the server problem in a uniform space:



Delayed Caching Problem

If a page in the cache is read at time t , it cannot be ejected at time $t + 1$.

Immediate (easy) results:

The algorithm LRU is $(k - 1)$ -competitive for the delayed k -cache problem

Delayed k -Cache Problem \equiv $(k - 1)$ -Cache Problem

We show (on the next few slides) that the

Delayed k -Cache Problem
reduces to the
 $(k - 1)$ -Cache Problem *

(... though the same reduction works in both directions)

* This means that we have to show that any algorithm for the regular cache problem can somehow be made to be an algorithm for the delayed problem...

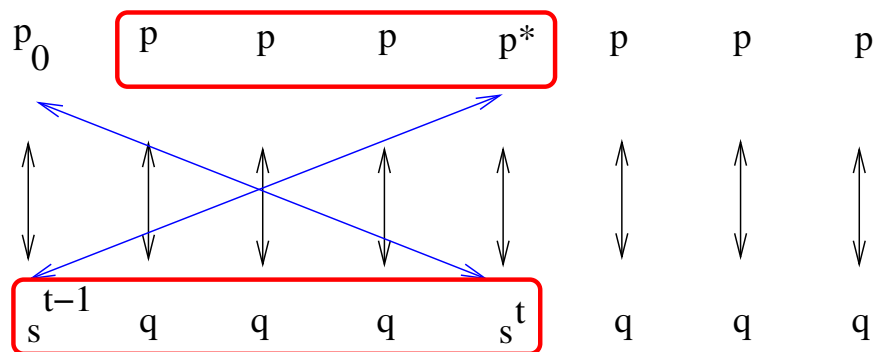
Our Reduction...

- q_0, q_1, \dots set of pages for the delayed k -cache problem
- p_1, p_2, \dots set of pages for the $(k - 1)$ -cache problem
- Add fictitious page p_0 ; never requested, fictitious location
- Map q 's to p 's in such a way as to "hide" the frozen server at p_0

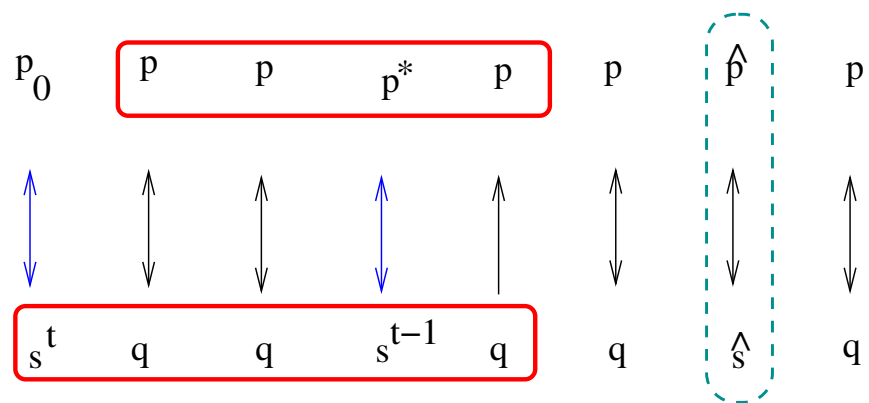
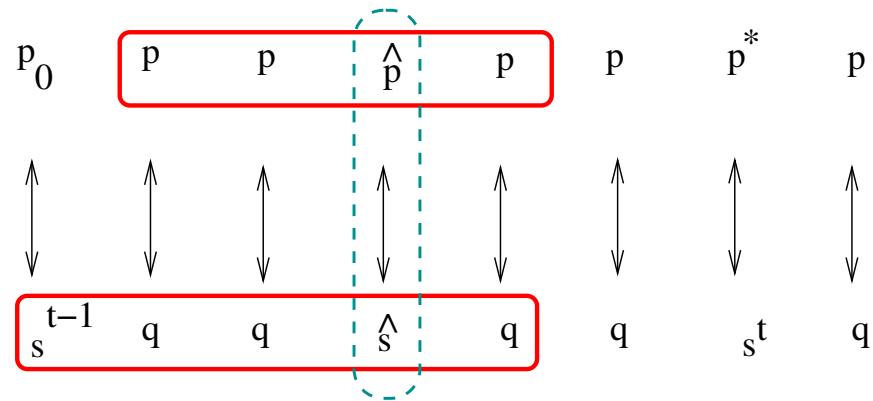
Remapping of Pages

- For request sequence $\varsigma = s^1 s^2 \dots s^n$,
construct a permutation $f^t(q_i)$ at every time step t
- $r^t = f^{t-1}(s^t)$
- Obtain $\varrho = r^1 r^2 \dots r^n$

Case 1: The requested page is in the cache ("Hit")

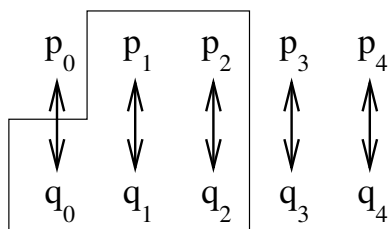


Case 2: The requested page is not in the cache ("Miss")

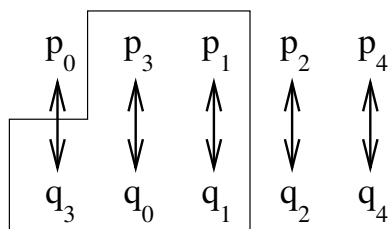


Example

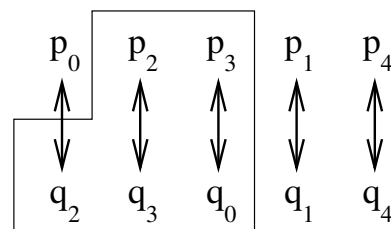
Suppose that $\varsigma = q_3q_2q_3q_1q_0$



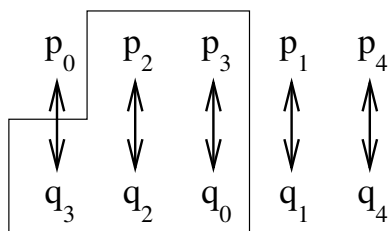
$t=0$



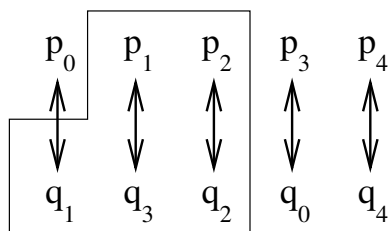
$t=1$



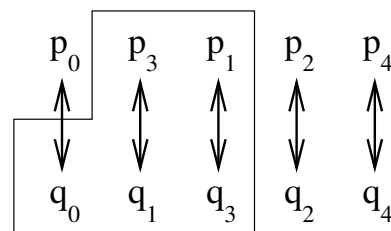
$t=2$



$t=3$



$t=4$



$t=5$

Then $\varrho = f(\varsigma) = p_3p_2p_2p_1p_3$

Open Problems: Delayed Server Problem

- Prove a lower bound of k for general metric spaces, trees?
- Give an upper bound for general metric spaces
- WFA?