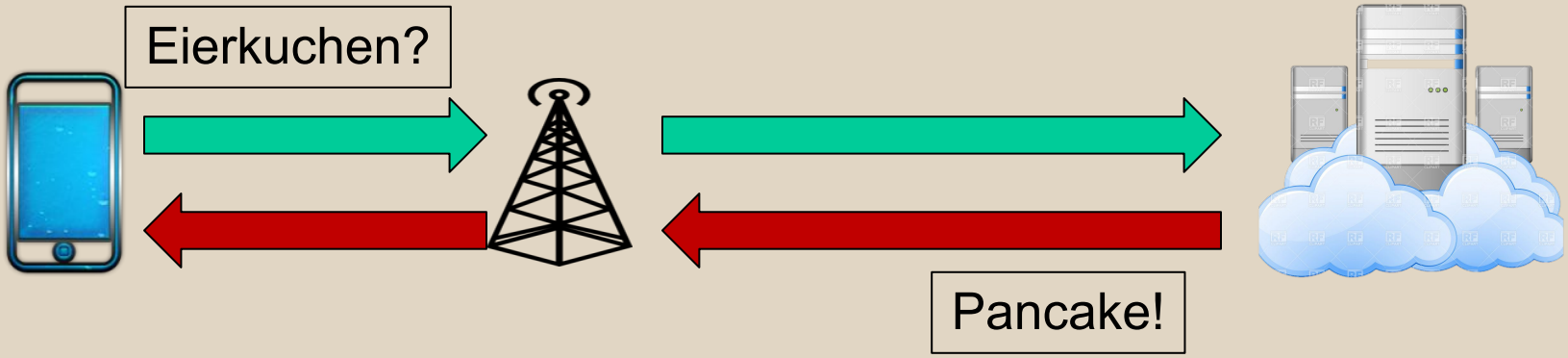
TEXAS A&M UNIVERSITY

Asymptotically Optimal Algorithm for Online Reconfiguration of Edge-Clouds

I-Hong Hou, Tao Zhao,
Shiqiang Wang, and Kevin Chan

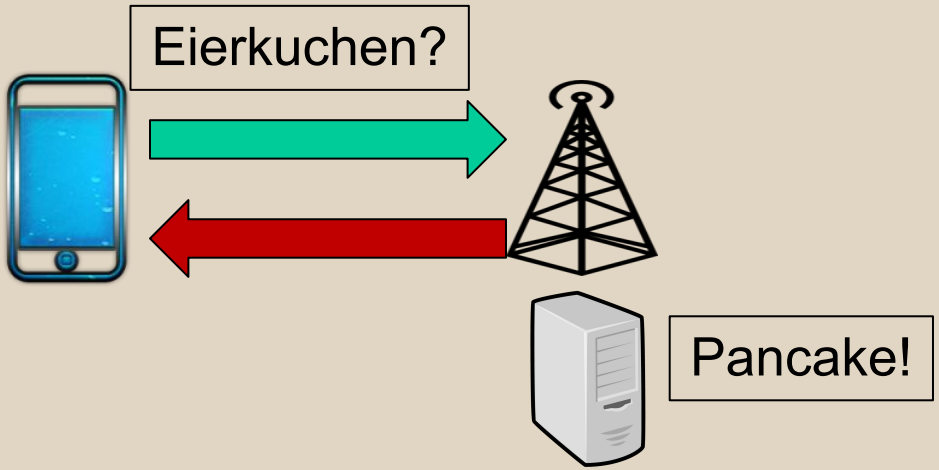
Background

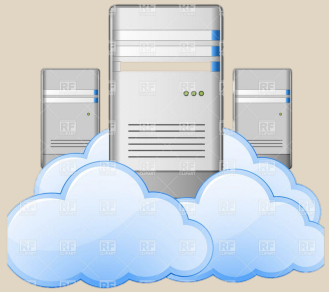
- Many emerging mobile applications rely on cloud computing techniques
- In a typical scenario, a mobile device forwards a request to the cloud for processing, and the cloud returns with a result



Background

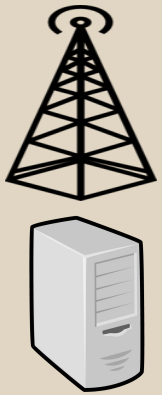
- The long distance between mobile users and data centers can cause high delay and traffic
- Edge-cloud: small-scale data centers located near cellular base stations
  - A.k.a. cloudlet, follow-me cloud, fog computing, etc.

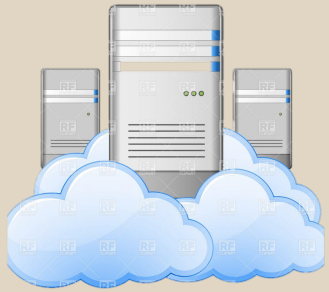




Challenges

- Edge-clouds can only host a small number of services
- Configurations of edge-clouds can have a significant impact on performance
- This paper studies online algorithms for reconfiguring edge-clouds





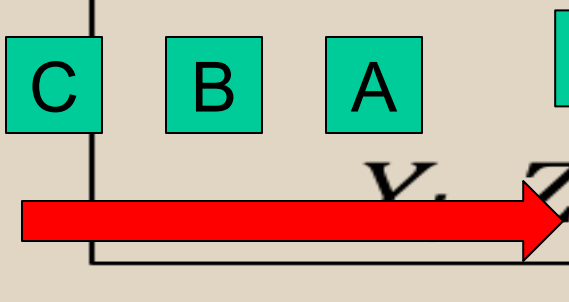
System Model

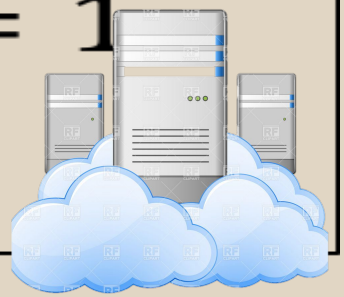
- Dual

$$Min \sum_i C_i Y_i + \sum_m Z_m$$

$$s.t. Y_i + Z_m \geq 1, \quad \text{if } K_{im} = 1$$

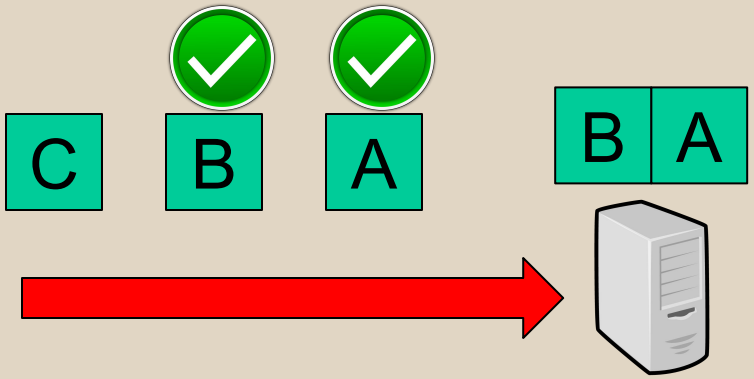
$$Y_i, Z_m \geq 0$$

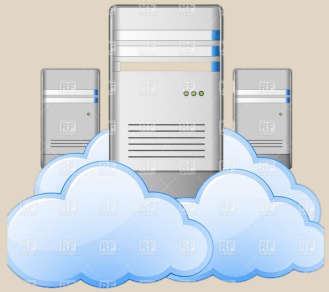




Cost of the System

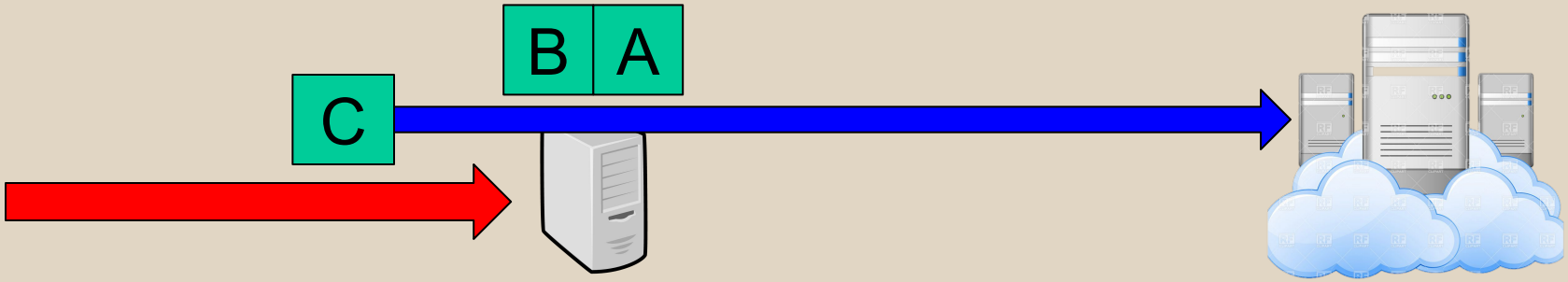
- If the edge-cloud hosts the service of a request, it can serve the request with 0 cost





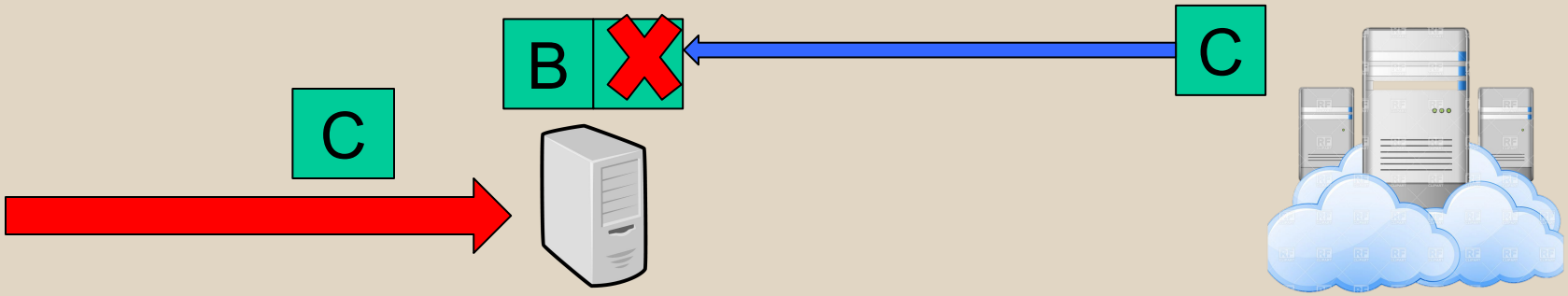
Cost of the System

- If the edge-cloud does not host the service of a request, it has two choices:
- 1. Forward the request to the backend cloud.
- This incurs a cost of 1



Cost of the System

- 2. Download the service to the edge-cloud, and evict one of current services
- This incurs a cost of M
- From now on, the edge-cloud can serve requests of this service for free



Goal of this Paper

- Goal: Minimize total cost

- Intuition:
- If a service will have a lot of requests, we should migrate the service to the edge-cloud
- Otherwise, we should forward these requests to the backend cloud

- Challenge:
- We have no information about future requests

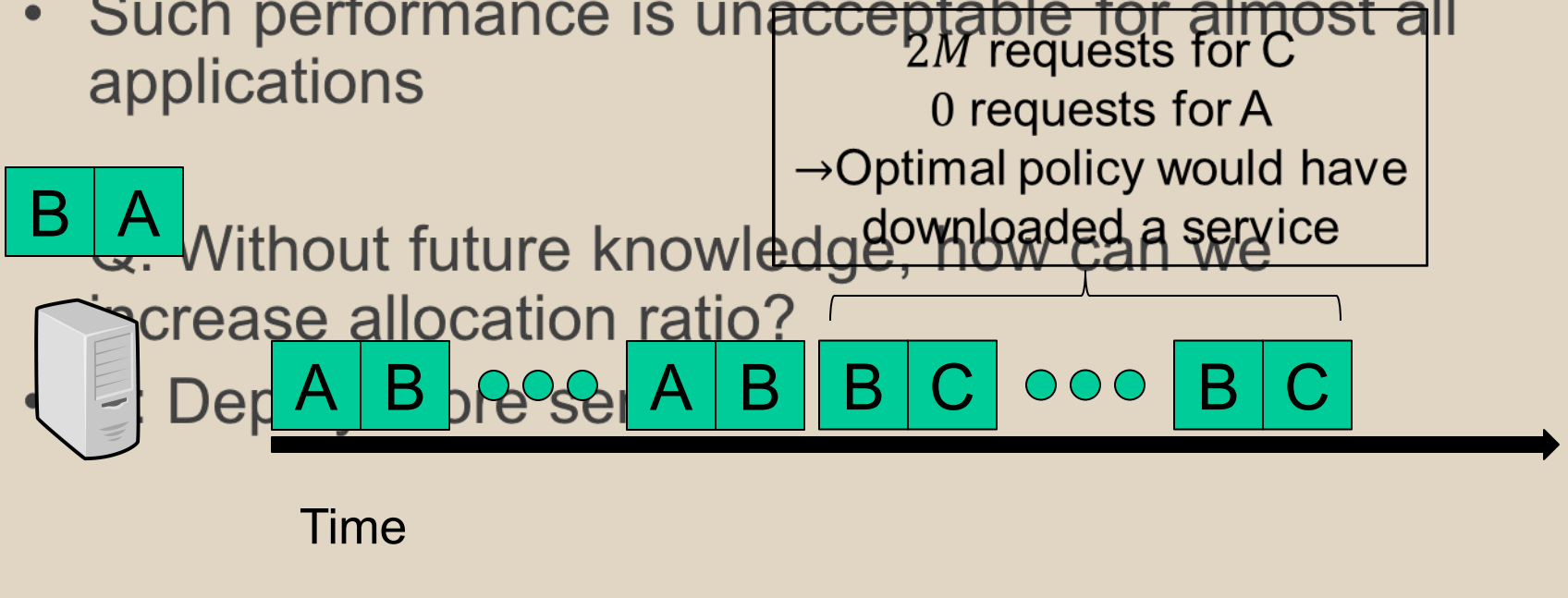
Online Policy and Competitive Ratio

TEXAS A&M UNIVERSITY

- Online policy: A policy that makes decisions without any knowledge about future arrivals
- Optimal offline policy: A policy that knows all future arrivals and achieves the smallest cost
- Competitive ratio of an online policy: Largest value for (cost of online policy)/(cost of optimal offline policy) over all arrival sequences

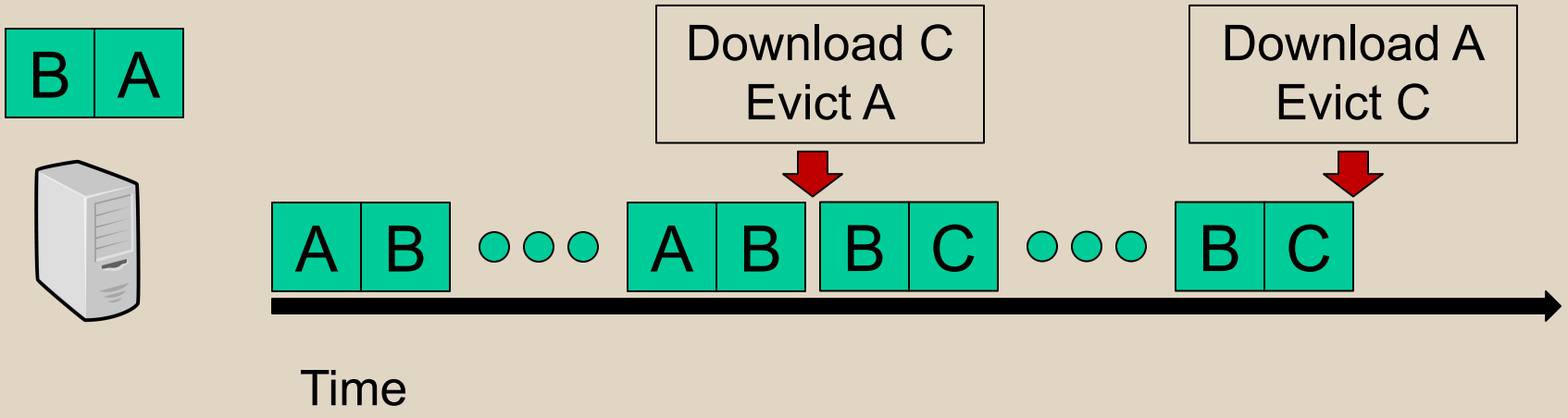
A Property of Optimal Policy

- Suppose offline policy can allocate 100% of jobs
- The online policy is guaranteed to allocate at least $1/(\frac{e}{e-1}) \approx 63\%$ of jobs
- The online policy can drop up to 37% of jobs
- Such performance is unacceptable for almost all applications



Proof by Contradiction

- If the optimal policy does not download anything in the duration, we can change the policy by
- 1. Downloading X at the beginning of the duration
- 2. Evicting X at the end of the duration
- This change does not increase cost



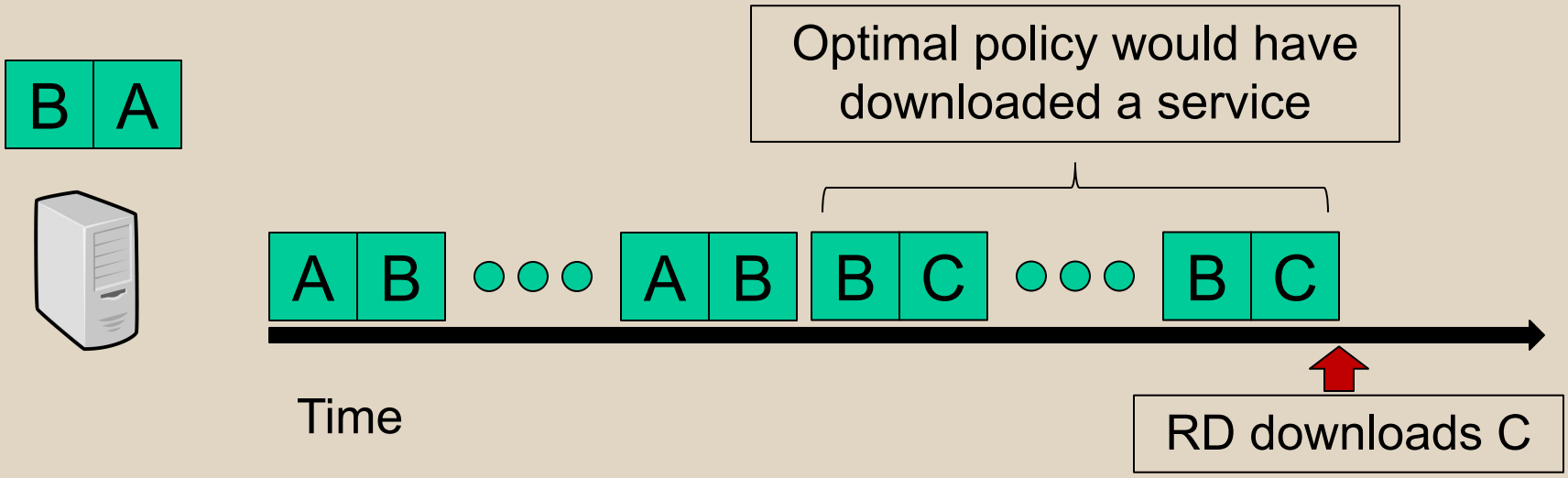
Our Policy: RL

- An online policy consists of two parts: Deciding whether to download a new service, and which service to evict

- For the first part, we propose retrospective download (RD)
- For the second part, we propose least recently used (LRU)

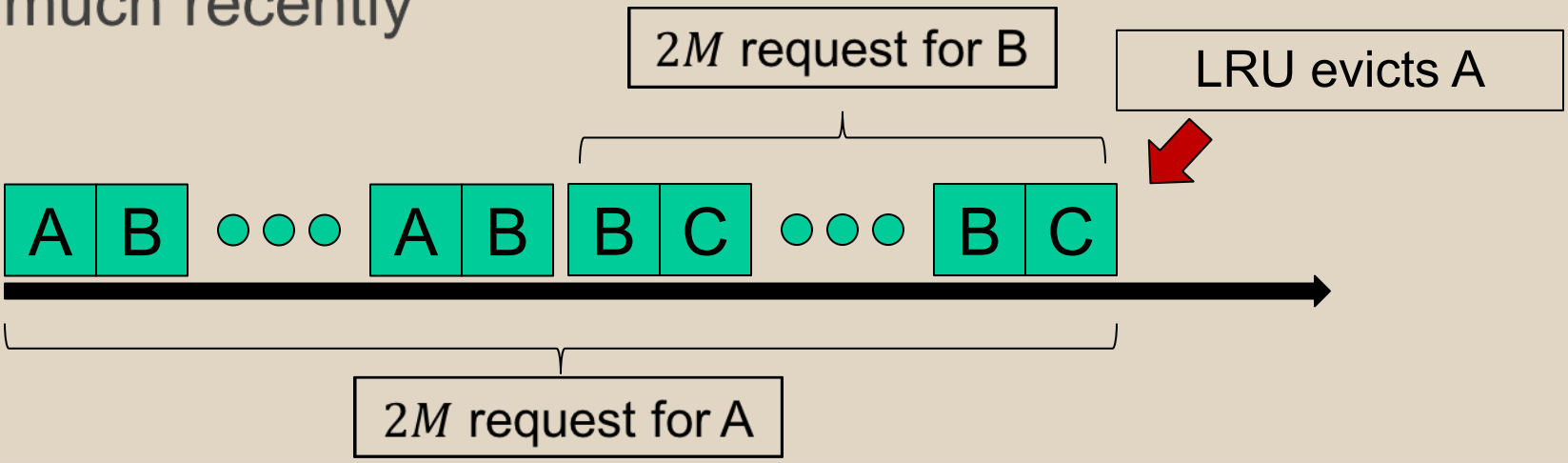
Retrospective Download (RD)

- If a service X has more than $2M$ more requests than any of the services at the edge-cloud in any past duration, RD downloads X
- Intuition: RD downloads a new service when it realizes the optimal policy would have made a download



Least Recently Used (LRU)

- For each service Y hosted by the online policy, find t_Y such that there are $2M$ request for Y in the last t_Y requests
- When downloading a new service, evict the service with the largest t_Y
- Intuition: Larger t_Y means the service is not used much recently



Performance Analysis

- A system with several servers
- Different servers can serve different kinds of jobs, and have different capacity
- C_i: The capacity of server i

Comparison with Existing Approach

- Most existing studies employ stochastic optimization or online learning
- These studies assume that request arrivals follow some unknown ergodic random process
- However, real-life request arrivals may not be ergodic
  - Ex. Viral videos come and go, without having a “steady-state” popularity
  - Safety-critical infrastructures, such as FirstNet, need to be resilient against natural disasters
- Competitive ratio analysis provides “worst-case” performance guarantees

Trace-Based Simulations

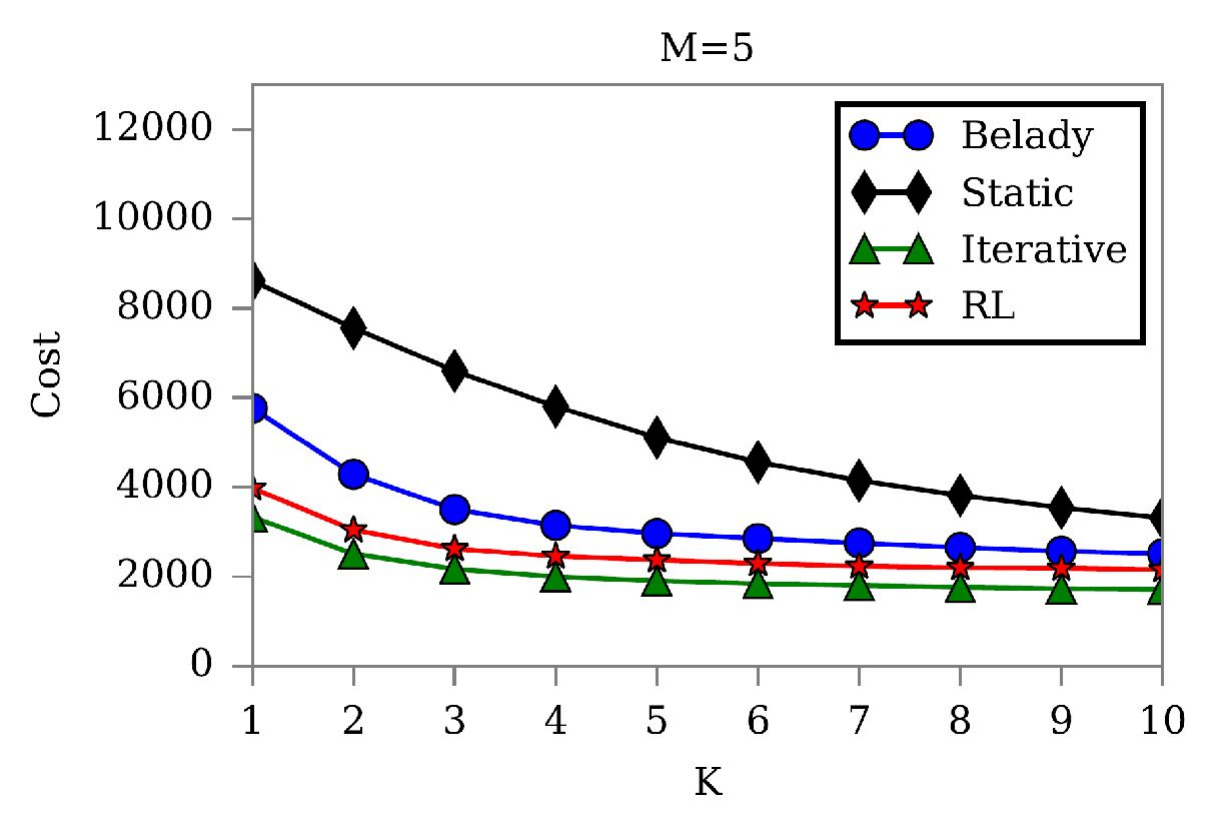
- We use a data set from Google cluster
- The data set registers request arrivals over 7 hours
- It contains more than three million requests, with 9218 different services
- We break the data set into 10 non-overlapping parts, and run simulations on each part
- Simulation results are the average of these 10 parts

Compared Policies

- We evaluate three other offline policies motivated by different approaches
  - They all require full knowledge about future arrivals
- Cache management: Belady's algorithm
  - Belady's algorithm minimizes number of misses
- Stochastic optimization/Online learning: A static policy that hosts the K most popular services
- Dynamic programming: An iterative policy to reduce complexity
  - Computing the optimal policy is infeasible even when K=5

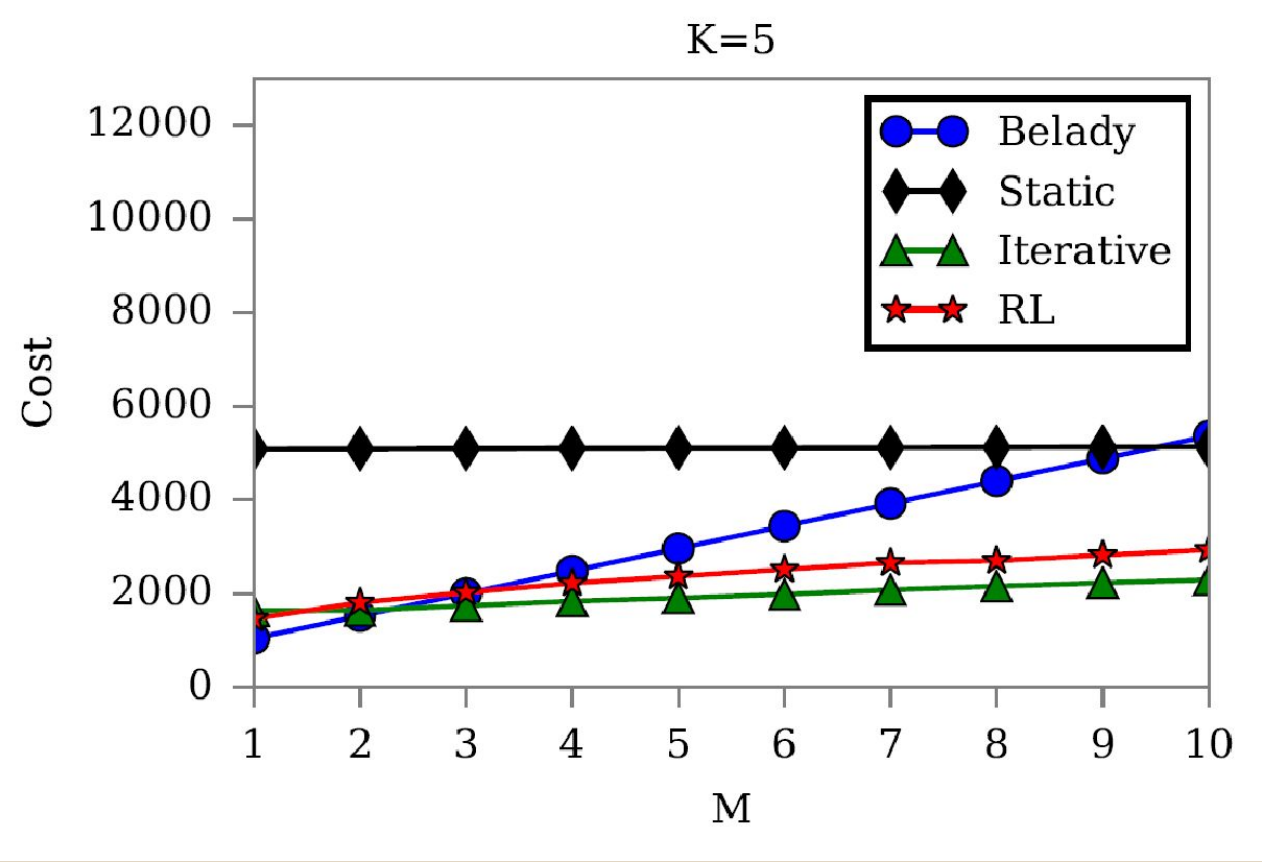
Simulation Results

- Fix the cost of downloading a new service (M)
- Vary the size of edge-cloud (K)



Simulation Results

- Vary the cost of downloading a new service (M)
- Fix the size of edge-cloud (K)



Observations and Implications

- Our policy is usually better than Belady and Static, and is close to Iterative

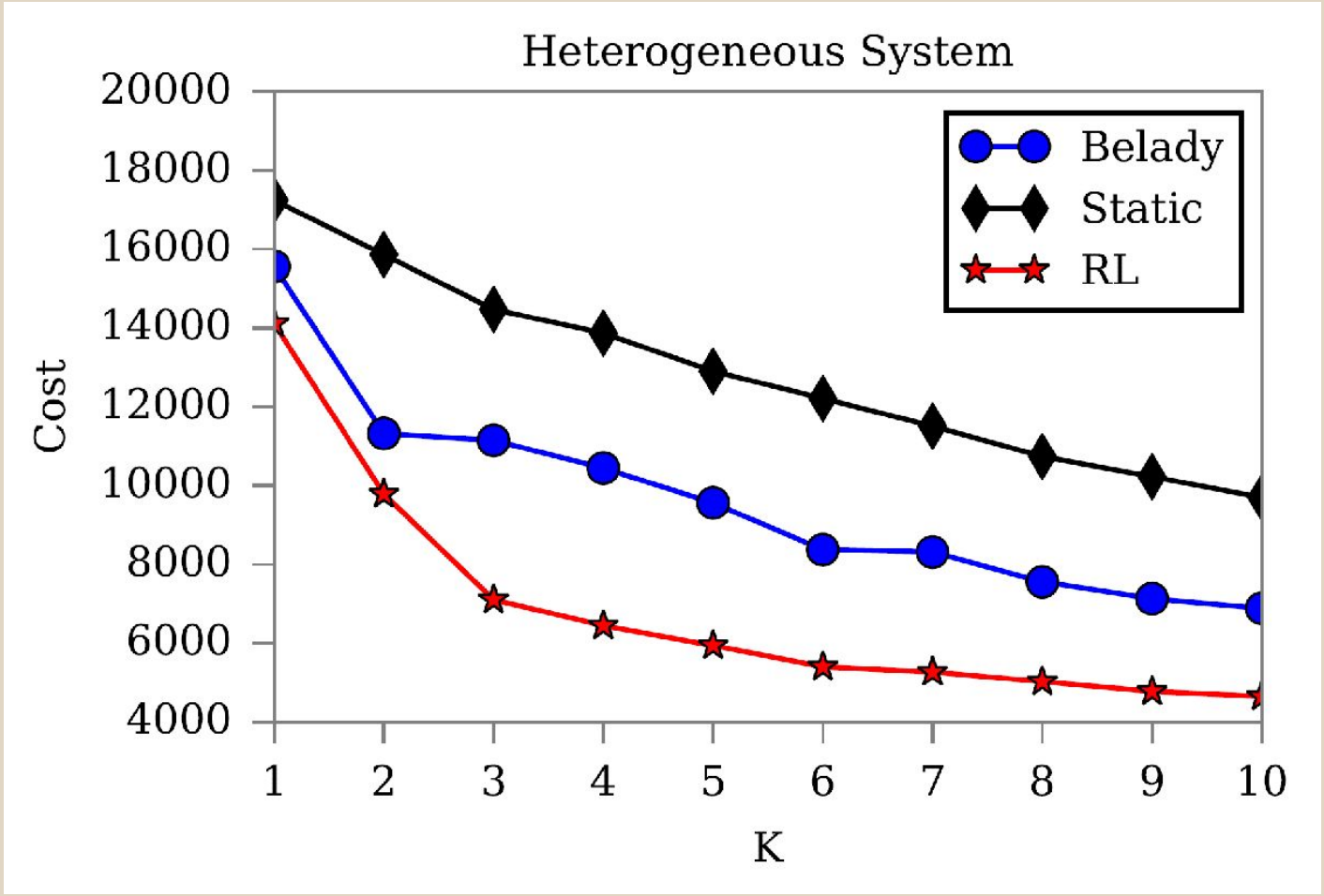
- Better than Belady: Edge-cloud reconfiguration is fundamentally different from cache management
- Better than Static: stochastic optimization/online learning may not work well because request arrivals are non-ergodic
- Close to Iterative: Our policy seems to be close to optimal

Heterogeneous Systems

- The cost to download service i is M_i
- Forwarding a request of i to the backend incurs a cost of F_i
- The size of service i is W_i
- To host a subset S of services, we need $\sum_{i \in S} W_i \leq K$

- Our policy (RL), Belady, and Static can be easily extended for heterogeneous systems

Simulation Results for Heterogeneous Systems



Conclusions

- We study online algorithms for dynamic edge-cloud reconfigurations
- We propose an online policy (RL) and proves that its competitive ratio is asymptotically optimal
- RL is compared against several offline policies that are motivated by different approaches
- Trace-based simulations show that RL is better than, or at least close to, these offline policies