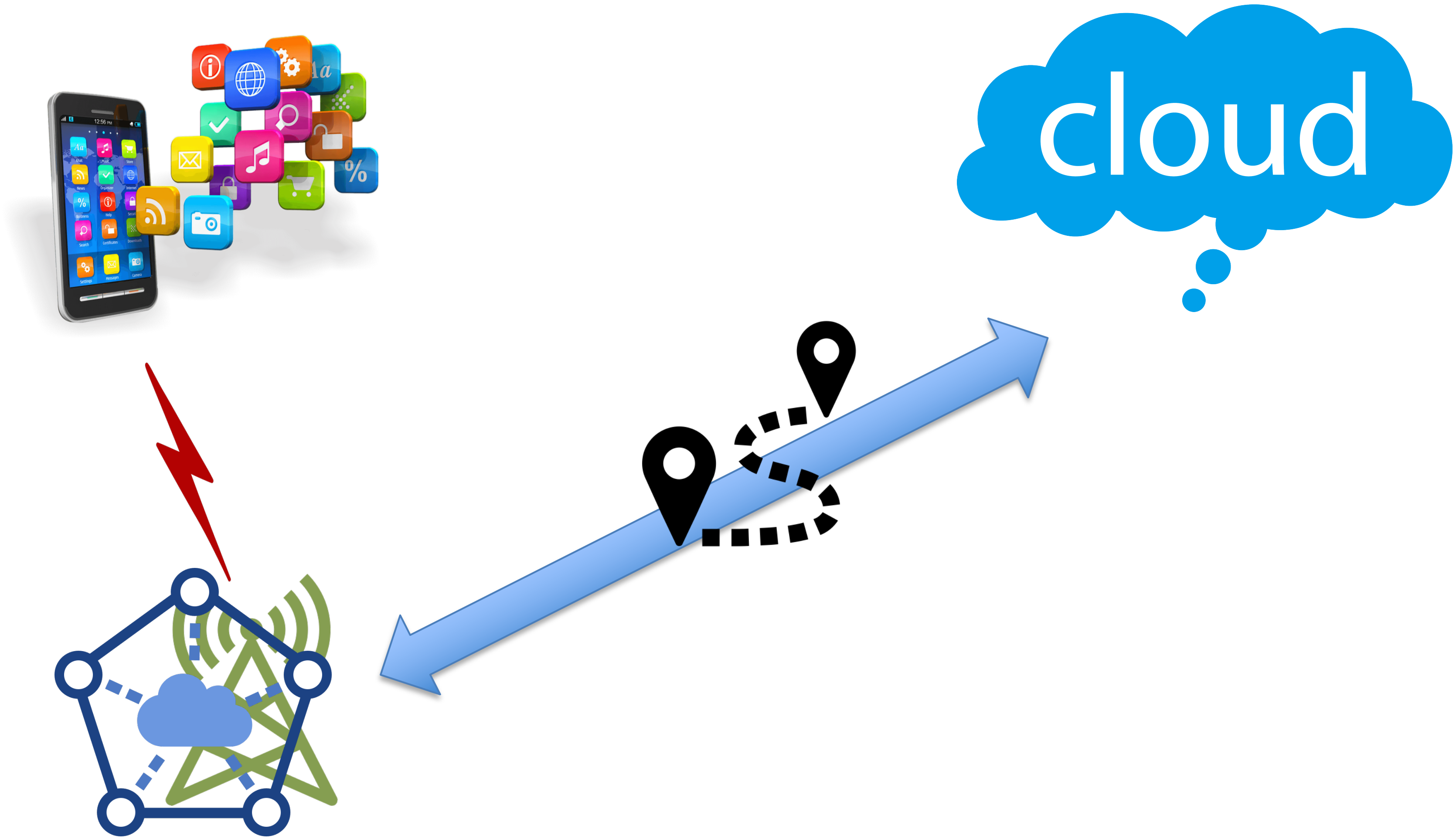
Mobile Task Offloading Under Unreliable Edge Performance

Md Rajib Hossen, Mohammad A. Islam

The University of Texas at Arlington

Edge Offloading

What and Why



Edge Offloading

Pros and Cons

Energy savings

Faster processing

- However, edge offloading incurs delay and consumes energy for data transmission
- We need to decide when and which tasks to offload

Edge Offloading

Current Studies

- Optimize latency and energy for offloaded tasks
- Consider single/multiple devices
- However, existing works assume offloaded task will always be processed at edge

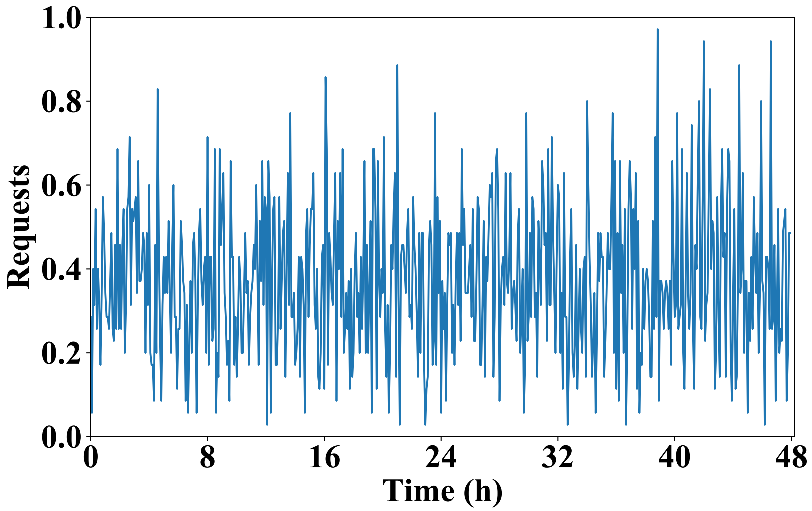
- K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, andY. Zhang. Deep learning empowered task offloading for mobile edge computing in urban informatics. IEEE Internet of Things Journal, 6(5):7635{7647, 2019.
- Ali Shakarami, Mostafa Ghobaei-Arani, and Ali Shahidinejad. A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective. Computer Networks, page 107496, 2020.

Edge Offloading

Challenges

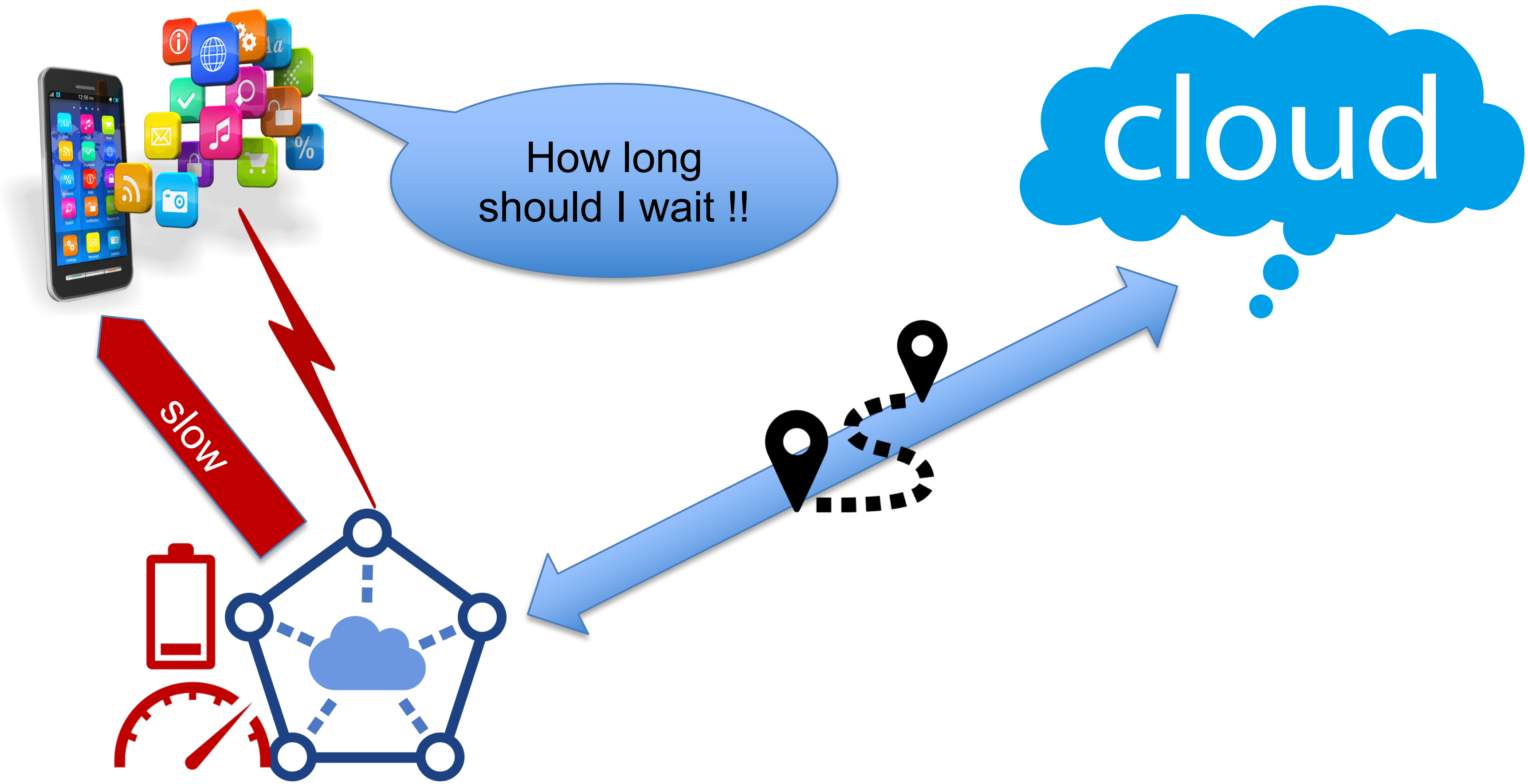
- Intermittent capacity

- Rapidly changing workload



Edge Offloading

Edge Offload to Cloud



Our Approach

- We optimize mobile energy and task completion time
- We consider unreliability in the edge
- We use “learning” to navigate the unknown environment

Objective

OPTO: $\min_{x_k} \sum_k (t_k + w \cdot e_k)$

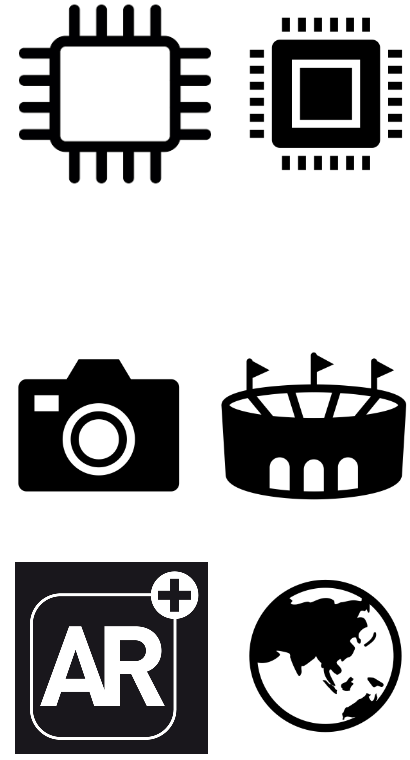
Where,

- t_k is task completion time
- e_k is energy consumption
- w is the weight variable
- x_k is the decision variable

Why Learning?

Model Free Solution

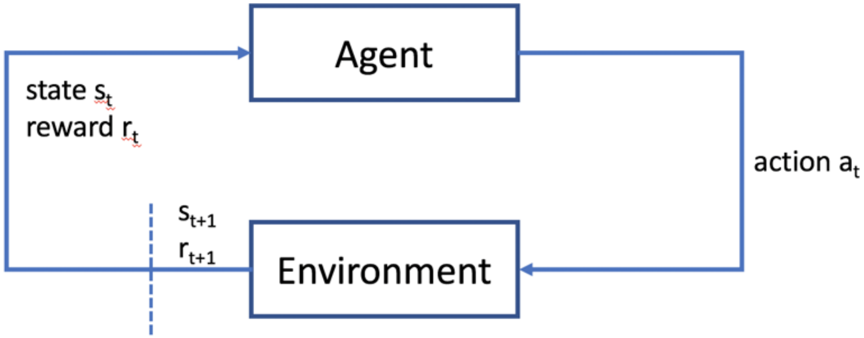
OPTO: $\min_{x_k} \sum_k (t_k + w \cdot e_k)$



- All offloading decisions are tied together
- t_k and e_k are hard to estimate
- Difficult to generalize energy & latency
- Different area may have different resource requirements
- We use MDP with Reinforcement Learning

Reinforcement Learning

Model Free DQN Solution



- No previous knowledge needed
- Q-learning, DQN, DDPG

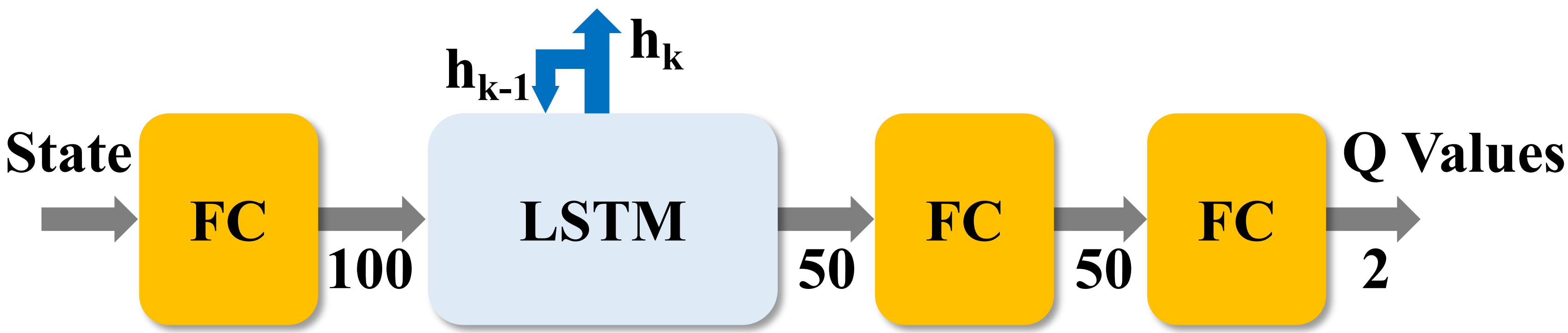
DeepTO

Markov Decision Process

- System State: $s_k = \left(a_k, e_k^b, \mu_k^m, \mu_k^e, u_k\right)$
    - $a_k = Task(data, cpu\ cycle), e_k^b = available\ energy$
    - $\mu_k^m = mobile\ capability,\ \ \mu_k^e = edge\ capability,$
    - $u_k = uplink\ rate$
- Action: $x_k \in \mathrm{A}(s_k) \in [0,1]$
- Reward: $r_k = R(s_k, x_k, s_{k+1})$
- Q-learning poor performance
    - DQN
- User request arrival is not Markovian
    - Long Short-Term Memory (LSTM) to estimate user request and state transition

DeepTO

Network Architecture



Experimental Settings

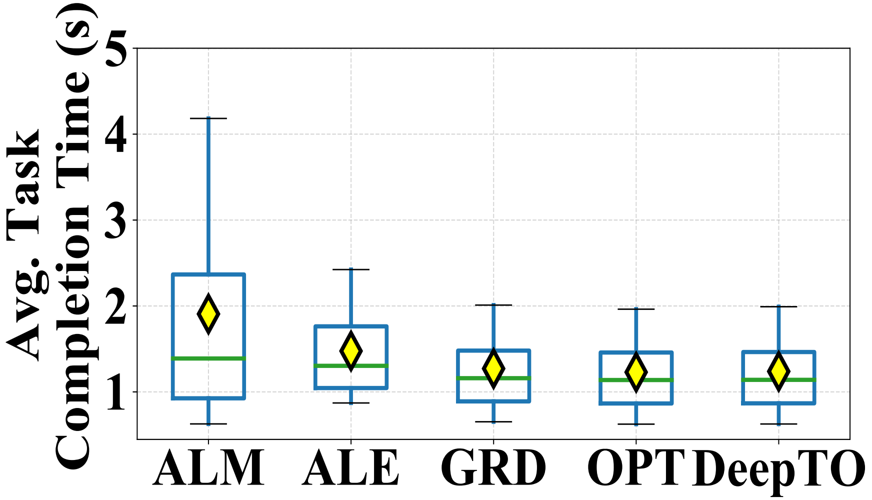
Real World & Synthetic

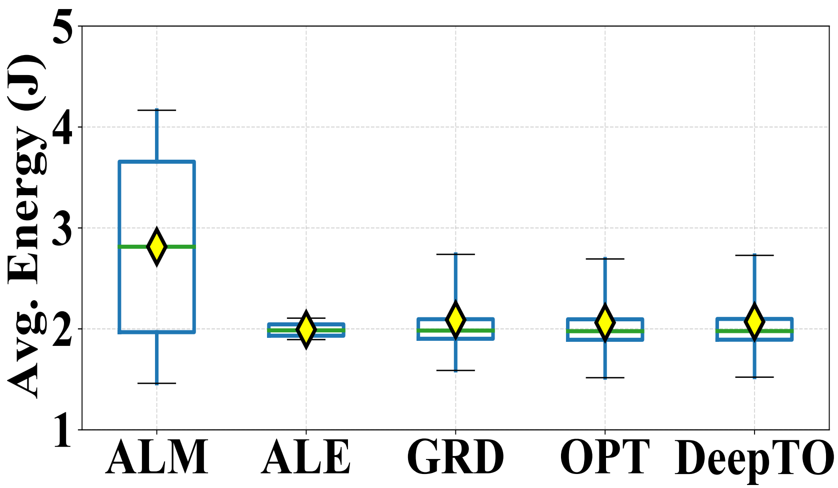
- Edge workload – uber trace
- Edge-to-cloud additional latency – cloud ping test
- Data sizes [500 KB-1 MB] randomly distributed
- CPU cycle 500 M – 2000 M
  - 500 M / 1.5 GHz = 0.33 sec
- Battery capacity 120 J
- Transmission power 500 mW
- Mobile capacity 1.5 GHz [20%-100%]
- Edge and Cloud 3.6 GHz

Results

Comparison of Algorithms

ALM – All Mobile
ALE – All Edge
GRD – Greedy
OPT – Optimum
DeepTo – Our approach

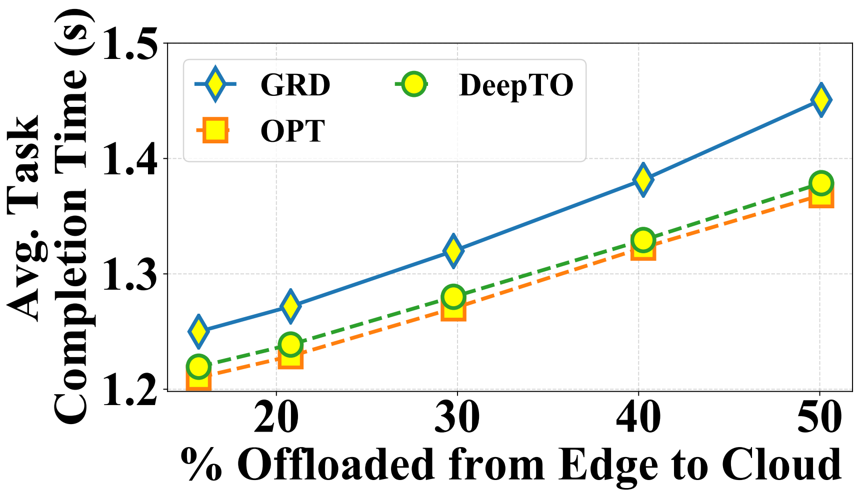


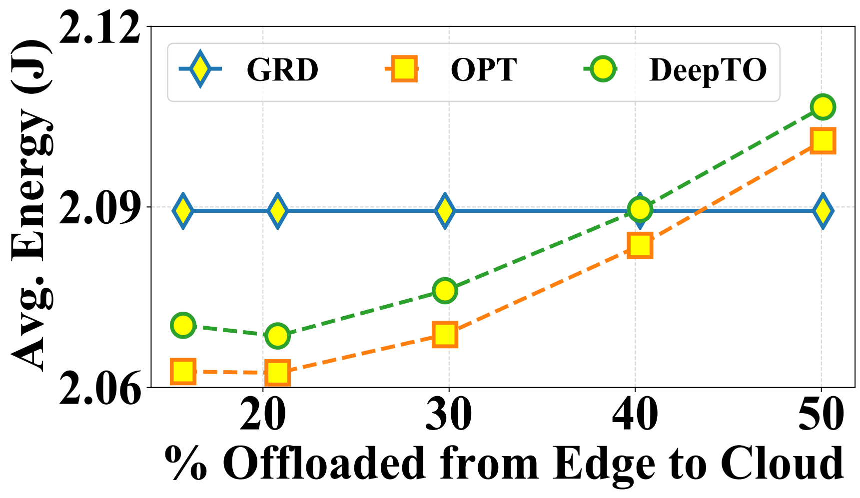


Results

Impact of Offloading from Edge

GRD – Greedy
OPT – Optimum
DeepTo – Our approach





Future Works

- Training time
  - Use federated learning
  - Use pre-trained model
- Use real world mobile user data

Thank you!

Questions?