Energy-efficient Task Offloading and Computing Scheme via Multi Agent Reinforcement Learning

Sinwoong Yun

DGIST

lion4656@dgist.ac.kr

28/Jun/2019

Group Meeting

Outline

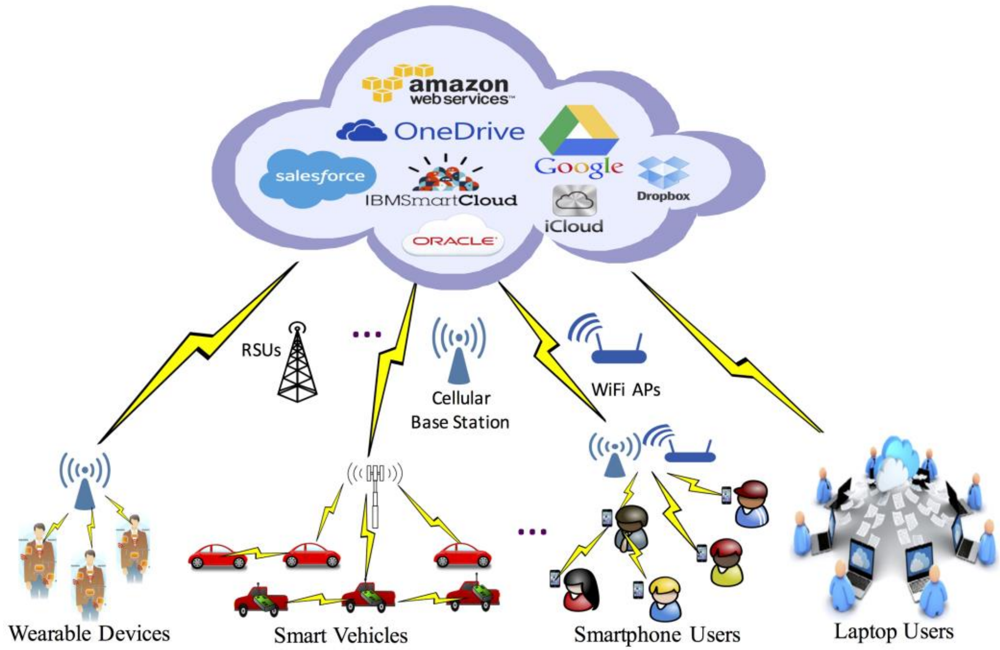
- **Overview**
  - Background
  - Prior Works & Limitations
  - Objectives & Expected Contributions
- **Ongoing Research**
  - System Model
  - Problem Formulation
  - RL Formulation
  - Simulation Results
    - Initial Result
    - Modified Interim Result
- **Research Plan**
  - Schedule

Part 1

OVERVIEW

Background : Cloud Computing

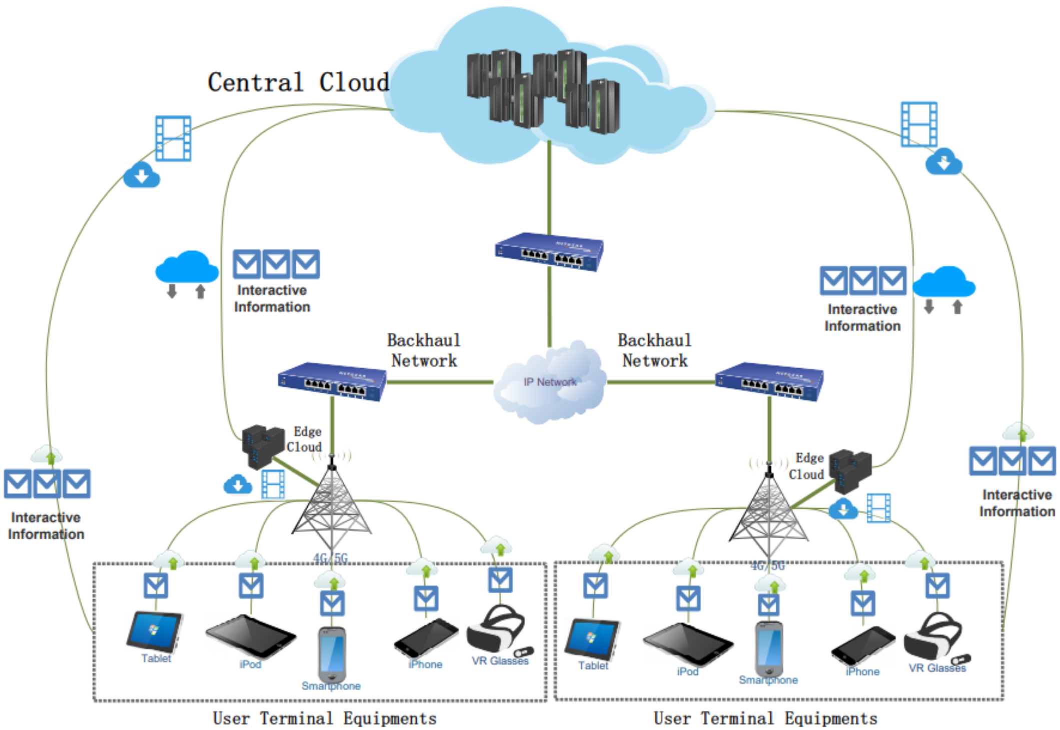
- **Cloud Computing (CC)**
  - Computing service at remote **cloud data center**
  - Long propagation distance (end user ↔ remote cloud center)
    - Excessively long latency for mobile applications
    - Not adequate for latency-critical applications



"Security and Privacy in Cloud Computing", *BBCR*

Background : Edge Computing

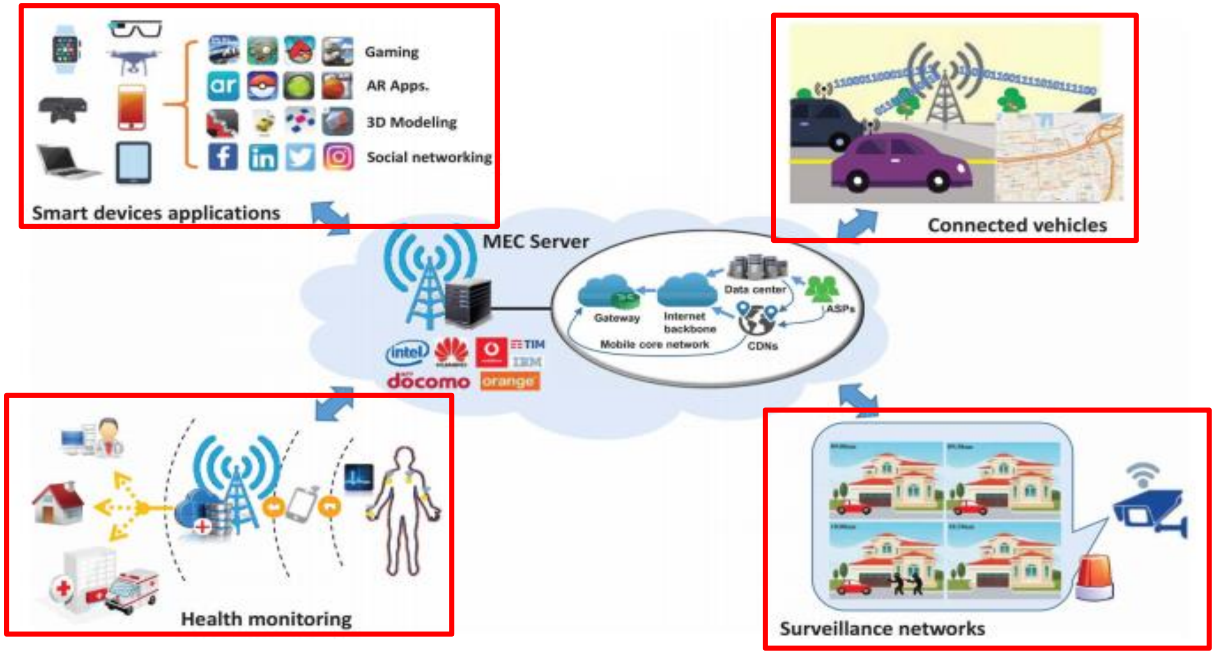
- **Edge Computing (EC)**
  - Address a drawback of CC with the proximate access
  - Harvest the idle computation power at the **network edges**
  - Perform computation-intensive & latency-critical tasks



[Cho:19] Chongwu Dong and Wushao Wen (2019). "Joint Optimization for Task Offloading in Edge Computing: An Evolutionary Game Approach". *Sensors 2019, 19(3), 740*

Background : Edge Computing

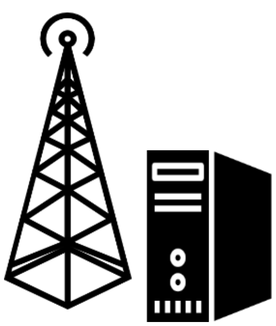
- **Applications of EC**
  - V2X (Vehicle to everything)
  - Smart phone applications (VR/AR, UHD video streaming, …)
  - IoT (Internet of things)

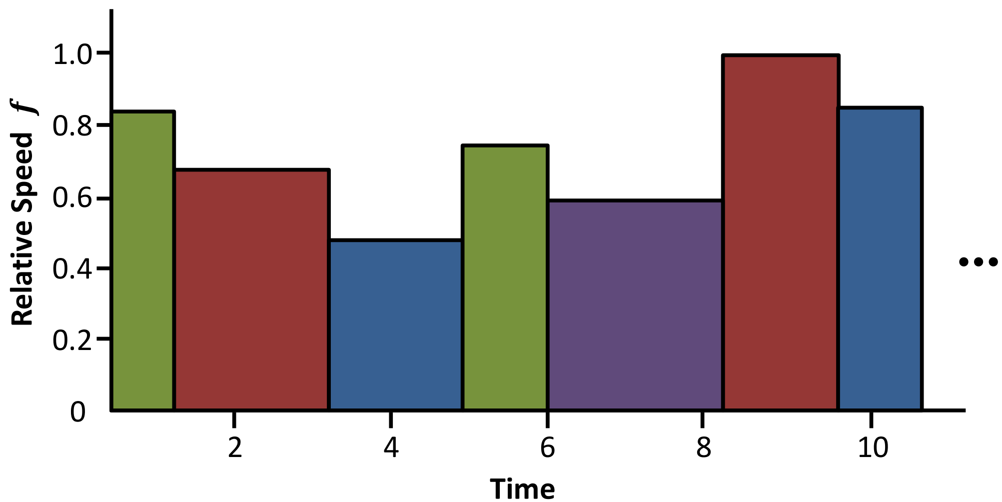


[Yuy:17] Yuyi Mao et al., (2019). "A Survey on Mobile Edge Computing: The Communication Perspective" . *IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 19, NO. 4, FOURTH QUARTER 2017*

Background : DVFS

- **Dynamic Voltage and Frequency Scaling (DVFS)**
  - CPU technology to reduce energy consumption
  - System energy consumption
    - Static energy : idle system power, cooler, …
    - Dynamic energy : CPU clock ($E_{dynamic} \propto f^2$)
- **DVFS in EC**
  - Energy-efficient computing by adjusting clock frequency
  - Task deadline : Optimize the frequency to maintain QoS

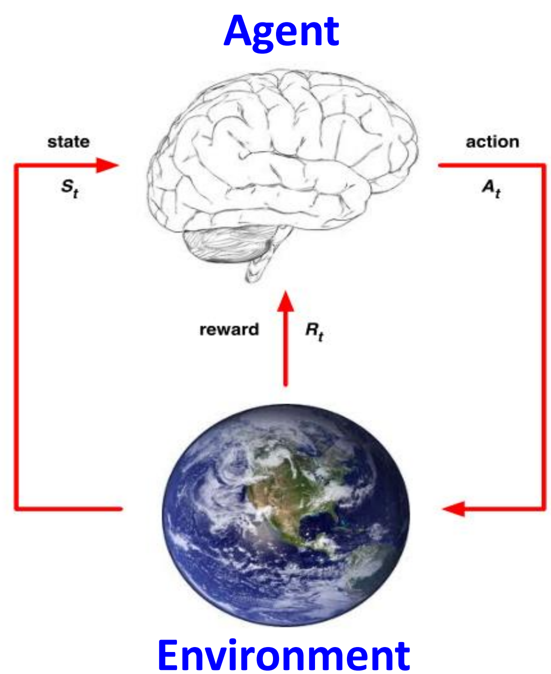




Background : Reinforcement Learning

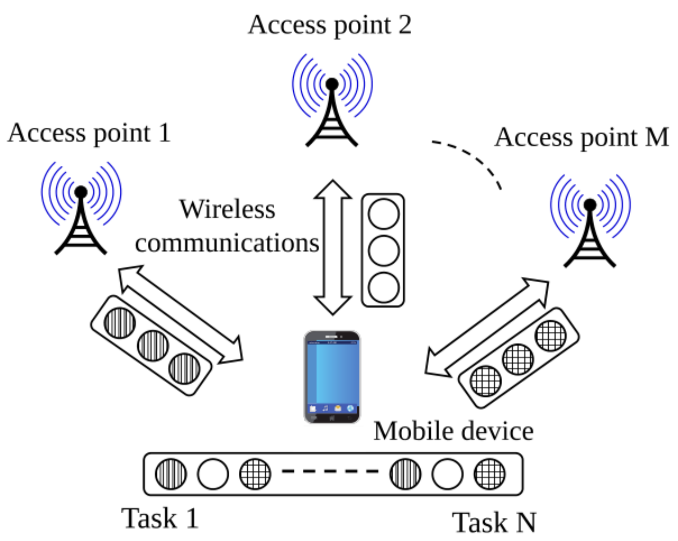
- **Reinforcement learning (RL)**
  - Agent learns how to behave in a environment
    - Agent receives state from the environment
    - Agent takes an action based on the state
    - Environment transitions to a new state
    - Agent receives reward from the environment

  - Try to learn optimal policy



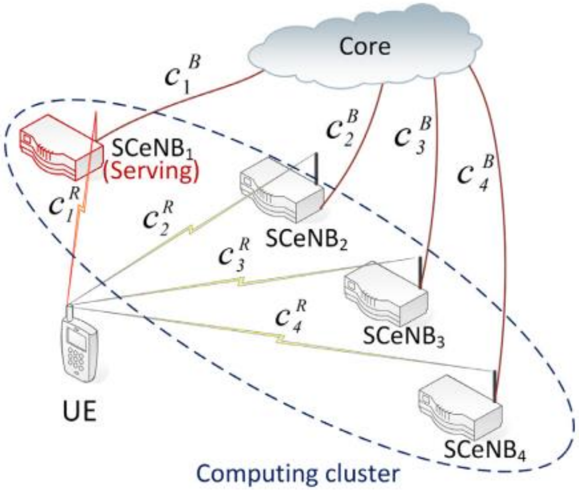
Prior Works & Limitations

- **Offloading in Mobile Edge Computing : Task Allocation and Computational Frequency Scaling**
  - *Thinh Quang Dinh, Jianhua Tang, Quang Duy La and Tony Q. S. Quek*
  - *2017 IEEE TCOM*
- **Contribution**
  - A mobile device (MD) can offload its tasks to multiple APs
  - Computational offloading with couples DVFS with task offloading
  - Minimize both MD's energy & latency
- **Limitation**
  - DVFS technique only for the MD
  - Single user ▶ do not know the effect for multiple users



Prior Works & Limitations

- **Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network**
  – *Jan Plachy, Zdenek Becvar, Pavel Mach*
  – *2016 ELSEVIER Computer Network*
- **Contribution**
  – Path selection algorithm finding the most suitable way for data delivery for small cell cloud (SCC) model
  – Estimate transmission delay and energy consumed by the transmission of offloaded data
- **Limitation**
  – Single user
  – Do not consider server energy & DVFS



Objectives

- **Energy efficient frequency scheduling**
  - CPU consumes energy with regard to its processing frequency
  - CPU frequency with high level can process task fast
  - However, this consumes more energy than low level frequency
  - **Find optimal frequency level**
- **Network backhaul (EC-cloud)**
  - BSs are connected with wire
    - Distribute a load among BSs to prevent one server from becoming congested
    - Wired transmission consumes a few time, energy
  - **Determine task distribution decision**

Minimize EC server energy consumption while meeting task deadline

Relative processor frequency | **Task distribution strategy**

Expected Contributions

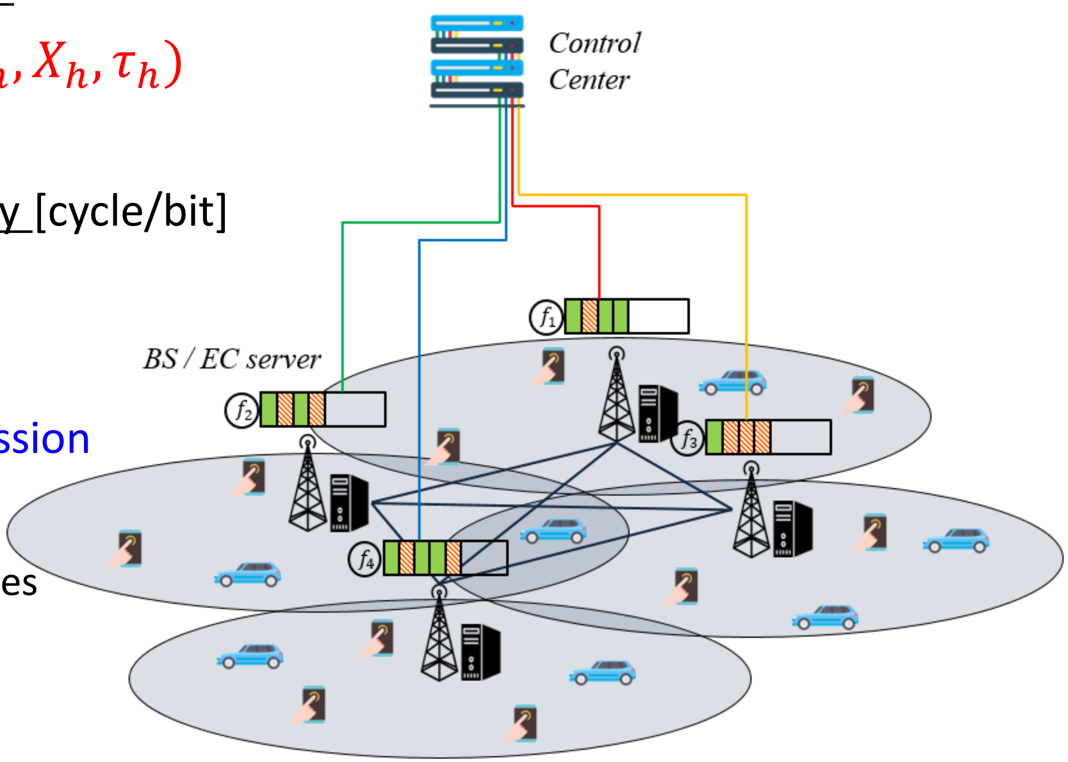
- **Joint optimization of computation and task distribution**
  - Task computation at **EC server** with DVFS technique
    - Reduce energy consumption while meeting QoS
  - Task distribution among the EC-cloud
    - **Control center** decides task distribution path
    - Tasks are distributed via wired backhaul
    - Alleviate load congestion for the **EC servers**
  - Inhomogeneous user distribution + time-variant task arrival rate
- **Reinforcement learning**
  - Separate the original problem into two problems
    - Task computation (**each EC server**) : low level
    - Task distribution (**control center**) : high level
  - Multi-agent RL network
    - **Each EC servers** learn independently : reduced state, action space
    - **Control center** assembles low level rewards : Maximize overall reward

Part 2

ONGOING RESEARCH

System Model

- **System model**
  - N-BSs are deployed in the area
    - BS index $i \in \mathcal{N} = \{1,2,\dots,N\}$
    - EC server equipped with each BS
    - f_i : Relative frequency for i
  - Request task type h : (D_h, X_h, τ_h)
    - D_h : input bit size [bit]
    - X_h : computation intensity [cycle/bit]
    - τ_h : deadline [sec]
  - **Control center**
    - Determine wired transmission
    - Re-distribute the loads
      - By observing server states



System Model

- **System model**
  - Episode
    - Discrete time slot $t \in \{1,2,\dots,t,\dots,t_{end}\}$
    - Each episode lasts for a long time
      - Each episode ends (t_{end}) when all tasks are processed (or time over)
  - Task request
    - $\lambda_{i,h}^{t}$: Average task request rate with task type h to i at time slot t
    - Different task request rate for BSs
    - Time-variant request rate
- **Task information in the server queue**
  - Request task type for task k : $h(k)$, task info : $(D_{h(k)}, X_{h(k)}, \tau_{h(k)})$
    - $D_{h(k)}$: left calculation bit size for k in the queue [bit]
    - $X_{h(k)}$: computation intensity for k [cycle/bit]
    - $\tau_{h(k)}$: left deadline for k[sec]

System Model

- **Overall phase**
  - **Network model : uplink transmission**
    - Nearest association
  - **Task offloading model : wire transmission via backhaul**
    - Tasks arriving at the BSs are re-distributed via backhaul
    - Control center decides where to offload the task based on the arriving task and the state of EC servers
    - Wire transmission uses a little time & energy
  - **Task computation model : frequency scaling**
    - EC server energy consumption
      - Determined by relative frequency, task input bit size and computation intensity
    - Task deadline
      - Total delay should not be longer than task deadline τ_h
      - Total latency = uplink transmission time + wire transmission time + queueing time + computation time

Problem Formulation

- **Original problem**

$$\min_{f_i^t, M_{ij}^t} \quad \sum_{t=1}^{t_{end}} E_{comp}^t + E_{wired}^t$$

Minimize BS/EC server energy consumption

$$\text{s.t.} \quad 0 \le f_i^t \le 1$$

Relative frequency

$$E_{comp}^t = \sum_{i=1}^{N} \left[c_{stat,i} + c_{dyna,i}(f_i^t)^2\right]$$

Computation energy consumption

$$E_{wired}^t = E_{wired} \sum_{i,j \in \mathcal{N}, i \ne j} M_{ij}^t$$

Backhaul energy consumption

$$M_{ij}^t \in \{0, 1\}$$

$$\sum_{i,j \in \mathcal{N}, i \ne j} M_{ij}^t \le 1$$

$$T_{ul,h}^k + T_{wire,h}^k + T_{queue,h}^k + T_{comp,h}^k \le \tau_h$$

Deadline constraint

- Objective : minimize BS/EC server energy consumption
- Constraint : Meet task deadline constraint

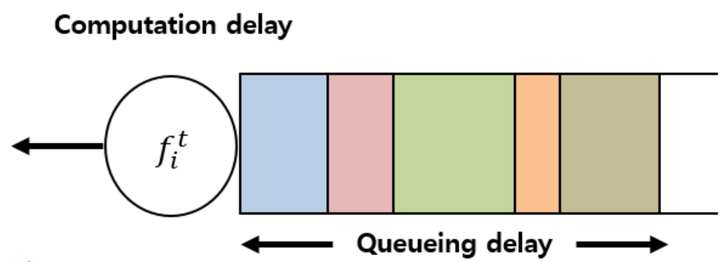
$c_{stat,i}$, $c_{dyna,i}$: computation energy constant
k : task index
M_{ij}^t : binary decision of wire transmission

Problem Formulation

- **Original problem**

  $T^{k}_{queue,h}$: Queueing delay

  $T^{k}_{comp,h}$: Computation delay



  - Relative frequency f_i^t is time-variant
  - Hard to calculate queueing delay/computation delay
  - In real world, future information (e.g., task input) is unknown
    - Strict to handle the whole problem

- **Reformulation**
  - Choose variable without knowing future information
    - Current information : **state**
    - Choose variable : **action**
  - **Reinforcement learning** can approximately optimize variables
    - Markov decision process (MDP)

RL formulation

- **Multi-agent reinforcement learning (MARL)**
  - Separate the problem into two problems
    - **Server** side : schedule CPU frequency
    - **Center** side : determine wire transmission strategy
  - Center controls BSs and each BS has its own small problem
- **CPU frequency scaling by DVFS (BS)**
  - Each server i processes the task with utilization f_i^t at time step t
  - Original : $0 \leq f_i^t \leq 1,\ \forall i, t$ ▶ RL : $f_i^t \in \{0.3, 0.4, \ldots, 0.9, 1\}$ (discrete)
  - Computation energy consumption at t : $E_{comp}^t = \sum_{i=1}^{N} \left[c_{stat,i} + c_{dyna,i}(f_i^t)^2\right]$
- **Task distribution strategy (control center)**
  - Assume time slot interval is short ▶ at most 1 task arrival to EC-cloud at each time
  - Choose task distribution path RL : $j \in \{1, 2, \ldots, N\}$ (discrete)
  - Consume transmission energy if the task is re-distributed via backhaul

RL formulation

- **Low level : CPU frequency scaling**

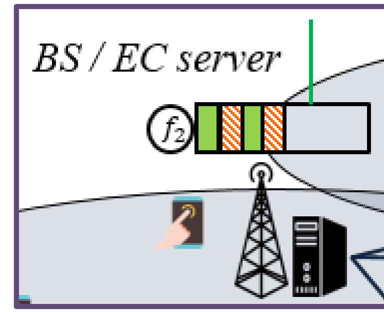
State : $[(D_{h(1)}X_{h(1)}, D_{h(2)}X_{h(2)}, D_{h(3)}X_{h(3)}),$
$(\tau_{h(1)}, \tau_{h(2)}, \tau_{h(3)}),\ \boldsymbol{backlog}]$

Action : $[f_i^t]$, discrete

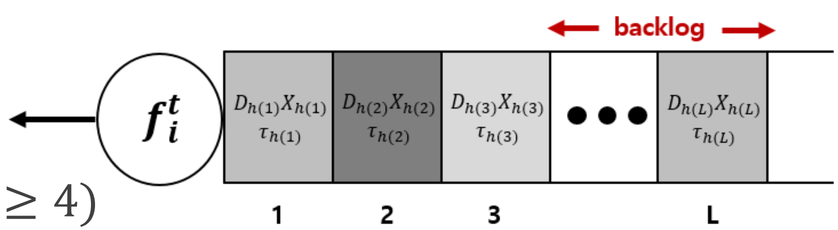
Reward : $r_i^{t_{end}} = \sum_{t=1}^{t_{end}} f(E_i^t) + \sum_{k \in tasks} p_{i,k}$

$$e.g.:\ f(E_i^t) = -E_i^t, \qquad E_i^t \propto (f_i^t)^2$$

$$e.g.:\ p_{i,k} = \begin{cases} -1\ (over\ deadline) \\ +0.1\ (meed\ deadline) \end{cases}$$



- **Agent** : each EC server
- ***backlog*** : left queue size [cycles]
  - Fixed state space for RL format
  - $\boldsymbol{backlog} = \begin{cases} \sum_{k=4}^{L} (D_{h(k)}X_{h(k)}),\ (k \geq 4) \\ 0,\ (k < 4) \end{cases}$



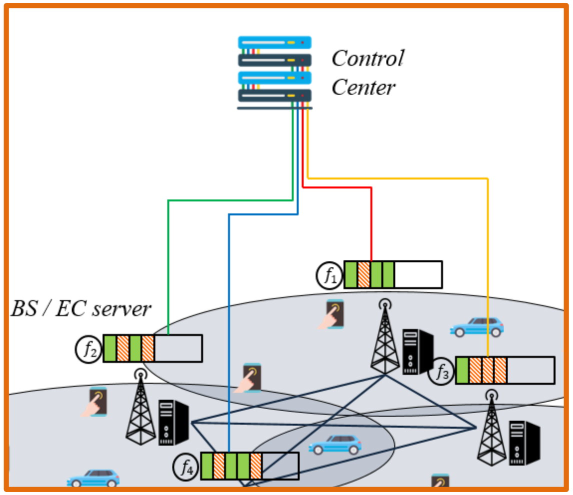
RL formulation

- **High level : task distribution**

State : $[i\ , h(k),\ \overrightarrow{\sum_{k=1}^{L}(D_{h(k)}X_{h(k)})}]$

Action : $[j]$, backhaul destination, discrete

Reward : $f\left(r_1^{t_{end}}, \ldots, r_N^{t_{end}}, \sum_{t=1}^{t=t_{end}} E_{wired}^{t}\right)$



  - **Agent** : control center
  - **State**
    - arrival task info $(i\ , h(k))$ + queue length [cycle]
  - **Reward**
    - function of low level rewards (e.g., sum of low level rewards)
    - function of backhaul energy consumption over one episode

RL formulation

- **Overall structure**

Contol center : task distribution

State : $[i\ , h(k),\ \overrightarrow{\sum_{k=1}^{L}(D_{h(k)}X_{h(k)})}]$

Action : $[j]$, backhaul destination, discrete

Reward : $f\left(r_1^{t_{end}}, \dots, r_N^{t_{end}}, \sum_{t=1}^{t=t_{end}} E_{wired}^{t}\right)$

Server 1: frequency scaling

State : $[(D_{h(1)}X_{h(1)}, D_{h(2)}X_{h(2)}, D_{h(3)}X_{h(3)}),$ $(\tau_{h(1)}, \tau_{h(2)}, \tau_{h(3)}),\ \boldsymbol{backlog}]$

Action : $[f_i^t]$, discrete

Reward : $r_i^{t_{end}} = \sum_{t=1}^{t_{end}} f(E_i^t) + \sum_{k \in tasks} p_{i,k}$

$$e.g.: f(E_i^t) = -E_i^t, \quad E_i^t \propto (f_i^t)^2$$

$$e.g.: p_{i,k} = \begin{cases} -1\ (over\ deadline) \\ +0.1\ (meed\ deadline) \end{cases}$$

● ● ●

Server N: frequency scaling

State : $[(D_{h(1)}X_{h(1)}, D_{h(2)}X_{h(2)}, D_{h(3)}X_{h(3)}),$ $(\tau_{h(1)}, \tau_{h(2)}, \tau_{h(3)}),\ \boldsymbol{backlog}]$

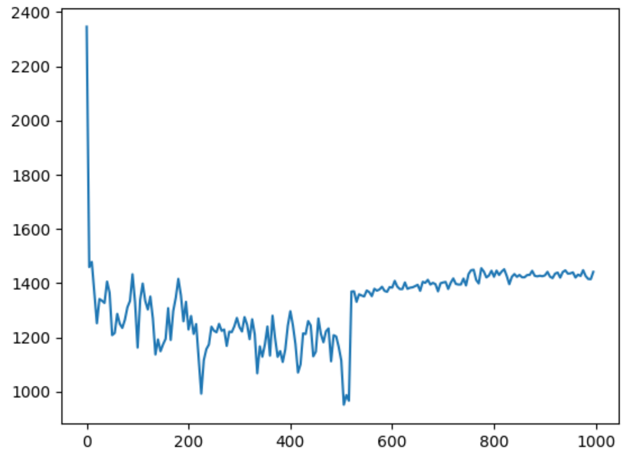
Action : $[f_i^t]$, discrete

Reward : $r_i^{t_{end}} = \sum_{t=1}^{t_{end}} f(E_i^t) + \sum_{k \in tasks} p_{i,k}$

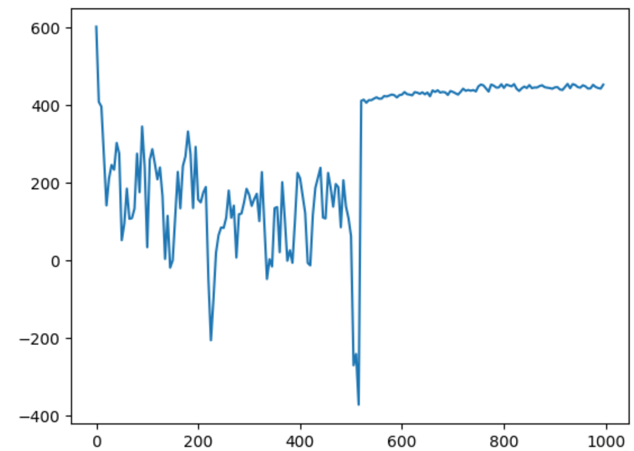
$$e.g.: f(E_i^t) = -E_i^t, \quad E_i^t \propto (f_i^t)^2$$

$$e.g.: p_{i,k} = \begin{cases} -1\ (over\ deadline) \\ +0.1\ (meed\ deadline) \end{cases}$$

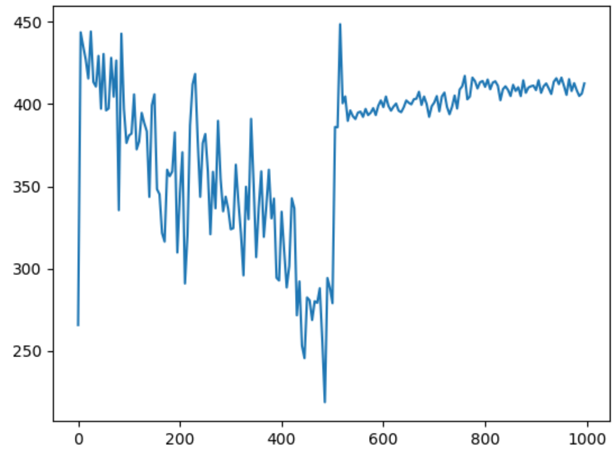
Initial Result



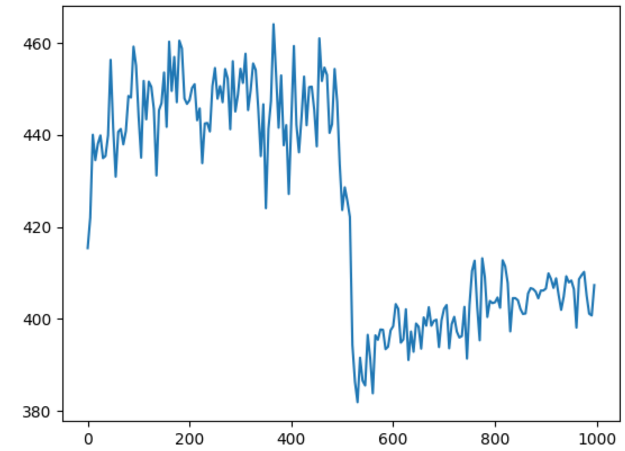
Control center reward



Server 1 reward



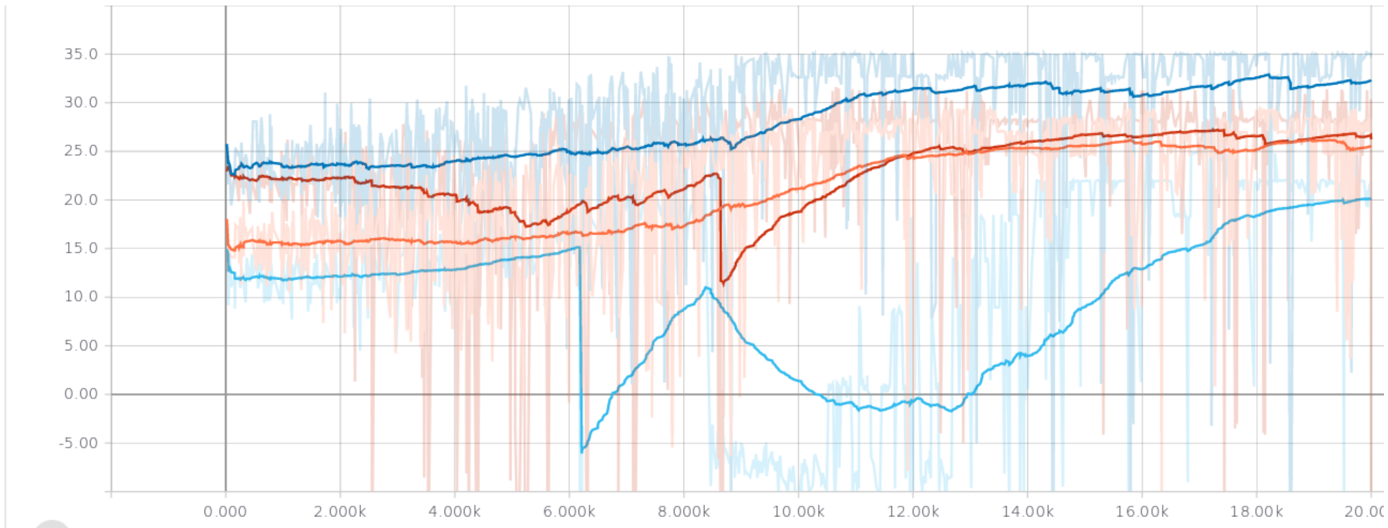
Server 2 reward



Server 3 reward

Modified Interim Result

- **Low level (frequency scaling) reward convergence**



  - Learn one EC server for low level
  - Reward increases after learning (with DQN algorithm)

    : small input size + deadline penalty

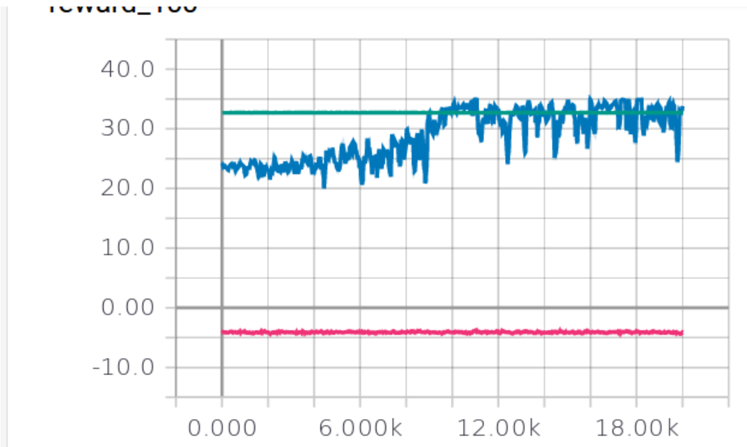
    : large input size + deadline penalty

    : small input size + no deadline

    : large input size + no deadline

Modified Interim Result

- **Low level (frequency scaling) reward convergence**



  – Compare a reward with benchmarks
    - Better than $f = 1$ and $f = 0.5$
    - However, not guarantee the result is better than any fixed frequency

: $f = 1$ for all time slot

: $f = 0.5$ for all time slot

Part 3

RESEARCH PLAN

Schedule

- **System model**
  - Survey & clarify the contribution point
    - DVFS in MEC
    - DVFS and task allocation via RL
    - Cloud MEC (backhaul among the BSs)
  - Time variant arrival rate, wireless backhaul
  - Reward design
    - Reward function affects performance significantly
    - Compare the performance for different reward functions
- **Write a journal/conference paper**
  - ICC 2020 (2020-06-07~11)
  - IEEE journal – Energy efficient Task Offloading and Computing Scheme via Multi-Agent Reinforcement Learning

Schedule

- **Simulation result**
  - Task offloading probability optimization
    - Proximal policy optimization (PPO)
    - Adjust hyperparameters
  - Multi-agent reinforcement learning
    - High level (task distribution) + low level (frequency)
    - Divide-and-conquer
    - Maximize high level reward
- **Search a new topic**
  - *RL with 5G network (IoT security, MEC, UAV, UDN, ..)*

Schedule

- Schedule (~ 2020 Jan)

Month / Progress	6	7	8	9	10	11	12	1
Study (Queueing theory, RL)								
Find contribution point & system model update								
Simulation result & performance analysis								
Paper writing (Journal & Conference)								
Search a new topic & system model								
Simulation for new topic								

Any Questions?

THANK YOU

Email : lion4656@dgist.ac.kr