

Computer Simulation Final Project Report

R09942082 楊雅婷

Introduction

This final project is mainly based on the paper from 2019 Globecom: [A SDN-Assisted Energy Saving Scheme for Cooperative Edge Computing Networks](#). This paper proposed an SDN-assisted cooperative edge computing scheme that has two main contributions:

1. Save energy and maintain the queueing delay at controlled levels by using the square-root staffing rule to determine the sets of active and sleeping edge devices.
2. Implement a load balancing scheme to equalize the load distribution among edge devices.

And the goal of this final project is trying to reproduce those results from this paper.

System Model and Problem Formulation

System Model

The system model is shown as Fig. 1 in the original paper.

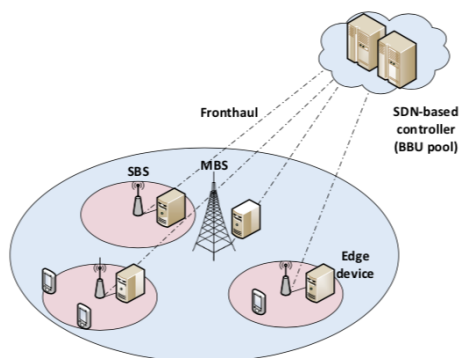
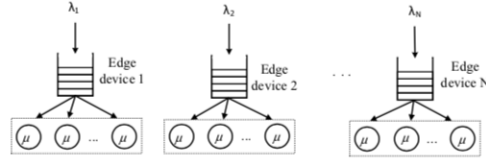


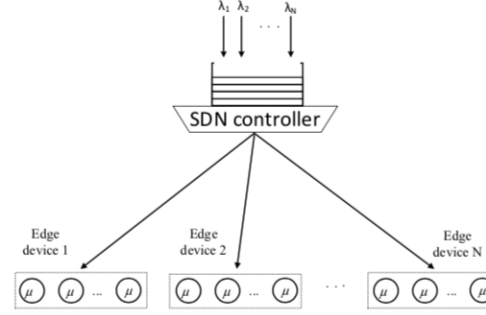
Fig. 1: System model.

The wireless links are provided by the small base stations (SBSs). They assume that all the computation tasks are processed by the edge devices, while the SDN-based controller can monitor and reschedule traffic among edge devices. And the interference is neglected.

The proposed SDN-assisted cooperative computing model is depicted in Fig. 2. They assumed that all servers (VMs) from all edge devices constitute a shared pool. Thus, the incoming traffic and edge resources are modeled as a single M/M/k queueing system.



(a) Traditional edge computing queue model.



(b) SDN-assisted queue model.

Fig. 2: Proposed edge computing layout.

Problem Formulation -- Energy Saving

The notations are listed as follows:

Notation	For edge device n	Notation	what for
k_n	number of virtual machines	p_u	user transmit power
λ_n	arrival rate	D_u	task data size
μ_n	service rate	s_u	wireless link speed
ρ_n	utilization = $\lambda_n / k_n \mu_n$	c_e	edge device processing speed

Performance metric: queueing delay

- Probability that a user will have to queue at edge device n

$$P_Q^n = \sum_{i=k_n}^{\infty} \pi_i = \pi_0 \frac{k_n^{k_n}}{k_n!} \frac{\rho_n^{k_n}}{1 - \rho_n}$$

- π_i : steady-state probability that i th server are occupied
- π_0 : steady-state probability that zero task exists in edge device n

$$\pi_0 = \left[\sum_{i=0}^{k_n-1} \frac{(k_n \rho_n)^i}{i!} + \frac{k_n^{k_n}}{k_n!} \frac{\rho_n^{k_n}}{1 - \rho_n} \right]^{-1}$$

- Mean queueing delay of the n th edge device

$$e_u^n = p_u \left[E[T_Q^n] + \frac{D_u}{s_u} + \frac{D_u}{c_e} \right]$$

Then defined notations for power consumption:

- p_e : power consumed by edge device
- p_T : total power consumption of edge devices = $\sum_{n=1}^N x_n p_e$
where $x_n \in \{0, 1\}$ is the on/off operator

Finally, the energy saving problem can be formulated as:

P1: Minimize: p_T
 x_n

Subject to: $C1: P_Q^n < \alpha, \quad \forall n \in \mathcal{N},$
 $C2: \rho_n < 1, \quad \forall n \in \mathcal{N},$
 $C3: x_n \in \{0, 1\}, \quad \forall n \in \mathcal{N}.$

With constraints C1 for queueing delay, C2 for stability maintenance, and C3 for the on/off operator. However, this is a binary integer non-linear programming that requires intensive computations to tackle. Not to mention that traffic offloaded from “off” edge devices need to be accommodated by “on” edge devices, which might affect the queueing probability at the host edge device.

Therefore, the paper modified the traditional model to the SDN-assisted one.

Modified as SDN-assisted model

- $k_T = \sum_{n=1}^N k_n$
- $\lambda_T = \sum_{n=1}^N \lambda_n$
- $\mu_T = \mu_n$
- $R_T = \lambda_T / \mu_T$

Minimum # of servers required to ensure a queueing probability $< \alpha$

$$k_T^* \approx R_T + c\sqrt{R_T}$$

- c is the solution to $\frac{c\Phi(c)}{\phi(c)} = \frac{1-\alpha}{\alpha}$
==> relation between c and α :

$$\alpha = \left[1 + \frac{c\Phi(c)}{\phi(c)} \right]^{-1}$$

The number of edge devices that can enter off mode = n_s

$$n_s = \left\lfloor \frac{Nk_n - k_T^*}{k_n} \right\rfloor$$

Problem Formulation -- Load Balancing

The load balancing scheme is designed to balance the long term probability of the utilization of each edge device.

P_{ij} : transitional probability

- $P_{ij} = \rho_i$ if $i = j$
- $P_{ij} = \frac{1-\rho_i}{N-1}$ if $i \neq j$

λ_n^* : re-scheduled arrival rate = $\lambda_T \pi_n^p$

- π_n^p : long term probability of the utilization of each edge device

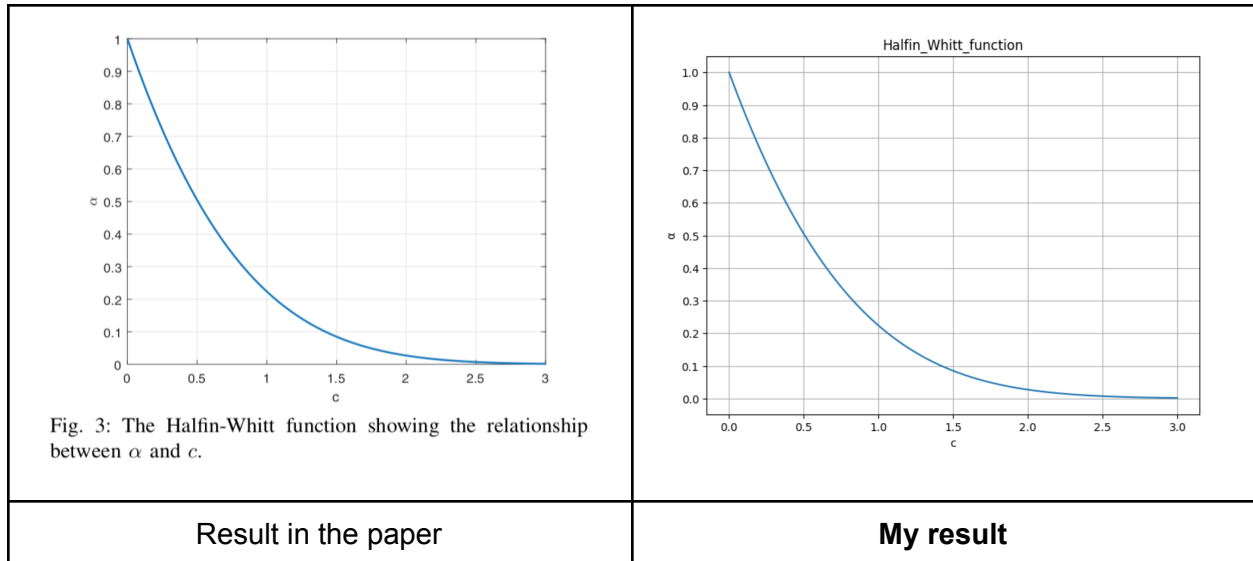
σ_p : std. of the utilization of edge devices

- $\sigma_p = \sqrt{\frac{1}{N} \sum_{n=1}^N |\rho_n - \bar{\rho}|^2}$, $\bar{\rho}$ = mean utilization of all edge devices

Implementation Steps and Results

Step 1

Implement the Halfin-Whitt function and show the relationship between α and c as Fig. 3 in the paper.



And I also get the pairs of α and c that might be useful later.

α	c
1.0	0.0
0.7	0.27

0.5	0.51
0.3	0.83
0.1	1.42

Step 2

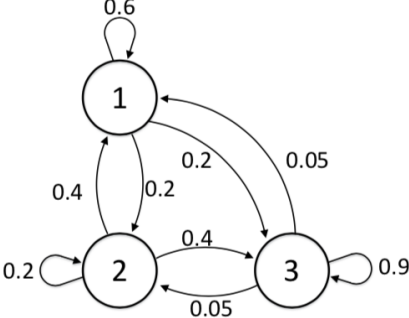
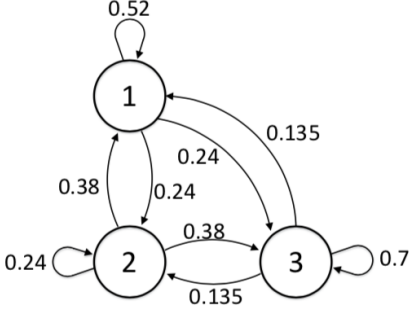
Implement Algorithm 1 (sleeping mechanism) to get the number of “on” and “off” devices as well as the rescheduled arrival rate after offloading.

<hr/> Algorithm 1: Proposed edge device sleeping mechanism <hr/> Define λ_{off} : arrival rate of tasks offloaded from sleeping edge devices; Find k_T^* according to (8); Find n_s according to (11); Set $x_n = 1, \forall n \in \mathcal{N}$; $n \leftarrow 1$; $\lambda_{off} \leftarrow 0$; while $n \leq n_s$ do Find $\min \{\lambda_n\}, \forall n \in \mathcal{N}$; $x_n \leftarrow 0$; $\mathcal{N} = \mathcal{N} \setminus n$; $\lambda_{off} \leftarrow \lambda_{off} + \lambda_n$; $n \leftarrow n + 1$; end Accommodating offloaded tasks: Sort \mathcal{N} in ascending order according to λ_n ; $n \leftarrow 1$; while $\lambda_{off} > 0$ do Define $\lambda_q \subset \lambda_{off}$ such that $\frac{\lambda_n + \lambda_q}{k_n \mu_n} < 1$ to satisfy C2; $\lambda_n \leftarrow \lambda_n + \lambda_q$; $\lambda_{off} \leftarrow \lambda_{off} - \lambda_q$; $i \leftarrow n + 1$; end <hr/>	<pre> For k_n= 10 mu_n= 1 α = 0.5 lambda_n_list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] k_T: 58.78226122841879 n_s: 4.0 ==== Before offload ==== lambda_n: [5, 6, 7, 8, 9, 10] n_on: [5, 6, 7, 8, 9, 10] x_n: [0, 0, 0, 0, 1, 1, 1, 1, 1, 1] lambda_off: 10 ==== After offload ==== lambda_n: [10, 10, 8, 8, 9, 10] n_on: [5, 6, 7, 8, 9, 10] x_n: [0, 0, 0, 0, 1, 1, 1, 1, 1, 1] lambda_off: 0 </pre> <hr/> <pre> For k_n= 10 mu_n= 1 α = 0.5 lambda_n_list: [6, 6, 6, 6, 6, 6, 6, 6, 6, 6] k_T: 63.95044301313157 n_s: 3.0 ==== Before offload ==== lambda_n: [6, 6, 6, 6, 6, 6, 6, 6] n_on: [4, 5, 6, 7, 8, 9, 10] x_n: [0, 0, 0, 1, 1, 1, 1, 1, 1, 1] lambda_off: 18 ==== After offload ==== lambda_n: [10, 10, 10, 10, 8, 6, 6] n_on: [4, 5, 6, 7, 8, 9, 10] x_n: [0, 0, 0, 1, 1, 1, 1, 1, 1, 1] lambda_off: 0 </pre>
<p style="text-align: center;">Algorithm 1.</p>	<p>My result: 2 examples show that we can turn n_s devices off due to k_T, and show the arrival rate of “on” devices before/ after offloading.</p>

Step 3

Implement Algorithm 2 (load balancing scheme) to find the balanced arrival rate at each edge device.

<hr/> Algorithm 2: Proposed load balancing algorithm <hr/> Calculate \mathbb{P} according to (13); Calculate $\mathbb{P}^* = \tau \mathbb{P} + \frac{1-\tau}{N}$; Calculate λ_n^* according to (14). <hr/>
--

 <p>(a) Cooperative edge computing without rescheduling. The long-term probabilities of edge device utilization is $\pi_1^p = 0.1818$, $\pi_2^p = 0.0909$, $\pi_3^p = 0.7273$.</p>	 <p>(b) Cooperative edge computing with rescheduling. The long-term probabilities of edge device utilization is $\pi_1^p = 0.2933$, $\pi_2^p = 0.1852$, $\pi_3^p = 0.5214$.</p>
Fig. 4(a) in the paper	Fig. 4(b) in the paper
<pre>[[0.6 0.2 0.2] [0.4 0.2 0.4] [0.05 0.05 0.9]]</pre> <p>Long-term probabilities of edge device utilization: [0.18181818 0.09090909 0.72727273]</p>	<pre>[[0.52 0.24 0.24] [0.38 0.24 0.38] [0.135 0.135 0.73]]</pre> <p>Long-term probabilities of edge device utilization: [0.29331046 0.18524871 0.52144082]</p>
My result for the long term prob.	My result for matrix P and long term prob.

We use parameters in Table I for the following implementations.

TABLE I: Simulation Parameters

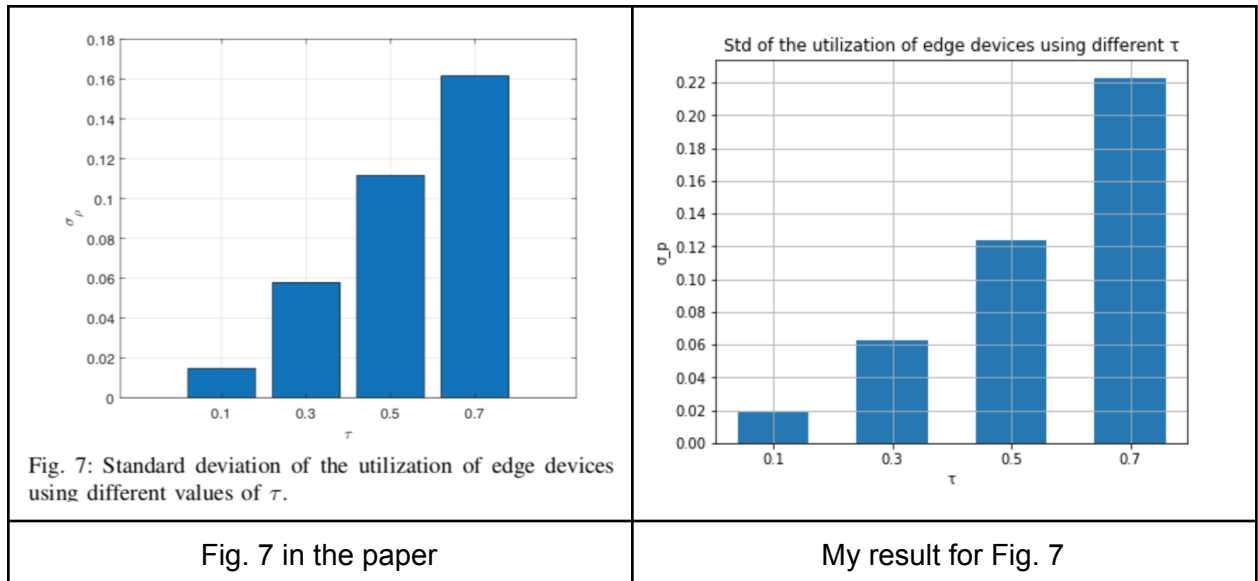
Description	Value
Number of edge devices (N)	10
Power consumption of an edge device (p_e)	50W
Arrival rate at each edge device (λ_n)	[1-10] user/sec
Departure rate at each edge devices (μ_n)	1 user/sec
Data size of tasks (D_u)	1 MB
User transmit power (p_u)	0.05W
Wireless link speed for each user (s_u)	1 Mbps
Edge device processing speed (c_e)	1 Gbps

So we test algorithm 2 for 10 users according to Table I.

<p>The original arrival rate at each device: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]</p> <p>P:</p> <pre>[[0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1] [0.089 0.2 0.089 0.089 0.089 0.089 0.089 0.089 0.089 0.089] [0.078 0.078 0.3 0.078 0.078 0.078 0.078 0.078 0.078 0.078] [0.067 0.067 0.067 0.4 0.067 0.067 0.067 0.067 0.067 0.067] [0.056 0.056 0.056 0.056 0.5 0.056 0.056 0.056 0.056 0.056] [0.044 0.044 0.044 0.044 0.044 0.6 0.044 0.044 0.044 0.044] [0.033 0.033 0.033 0.033 0.033 0.033 0.7 0.033 0.033 0.033] [0.022 0.022 0.022 0.022 0.022 0.022 0.022 0.8 0.022 0.022] [0.011 0.011 0.011 0.011 0.011 0.011 0.011 0.011 0.9 0.011] [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]</pre> <p>(before) Long-term probabilities of edge device utilization: [3.46054296e-15 3.91189467e-15 4.62172555e-15 5.39017599e-15 6.40259995e-15 8.47734302e-15 1.20290217e-14 1.53600991e-14 4.30575004e-14 1.00000000e+00]</p>	<p>P_star (tau = 0.5):</p> <pre>[[0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1] [0.0945 0.15 0.0945 0.0945 0.0945 0.0945 0.0945 0.0945 0.0945 0.0945] [0.089 0.089 0.2 0.089 0.089 0.089 0.089 0.089 0.089 0.089] [0.0835 0.0835 0.0835 0.25 0.0835 0.0835 0.0835 0.0835 0.0835 0.0835] [0.078 0.078 0.078 0.078 0.3 0.078 0.078 0.078 0.078 0.078] [0.072 0.072 0.072 0.072 0.072 0.35 0.072 0.072 0.072 0.072] [0.0665 0.0665 0.0665 0.0665 0.0665 0.0665 0.4 0.0665 0.0665 0.0665] [0.061 0.061 0.061 0.061 0.061 0.061 0.061 0.45 0.061 0.061] [0.0555 0.0555 0.0555 0.0555 0.0555 0.0555 0.0555 0.5 0.0555] [0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.55]]</pre> <p>(after tau) Long-term probabilities of edge device utilization: [0.0714746 0.07567454 0.08039888 0.08575239 0.0918697 0.09899524 0.10723867 0.11697967 0.12866712 0.14294919]</p> <p>The re-scheduled arrival rate at each device: [3.93110275 4.1620996 4.42193863 4.71638134 5.05283332 5.44473838 5.89812668 6.43388185 7.07669174 7.86220564]</p>
Original arrival rate and long term prob.	My result: New long term prob. and arrival rate after running algorithm 2

Step 4

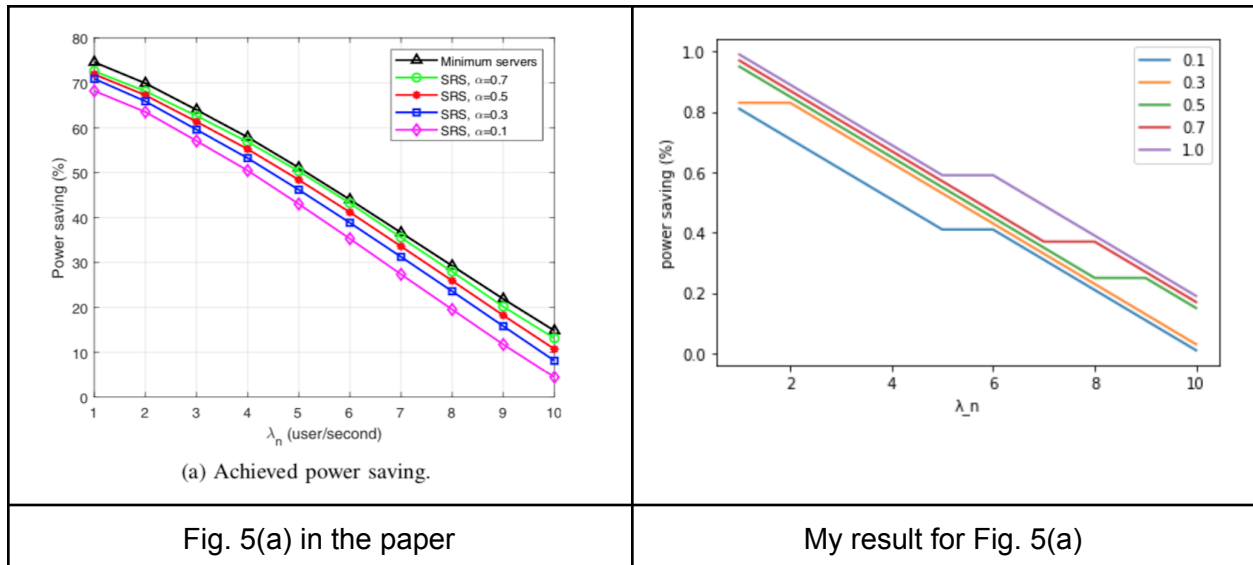
Try to plot Fig. 7 in the paper, that is, standard deviation of the utilization of edge devices using different values of τ .



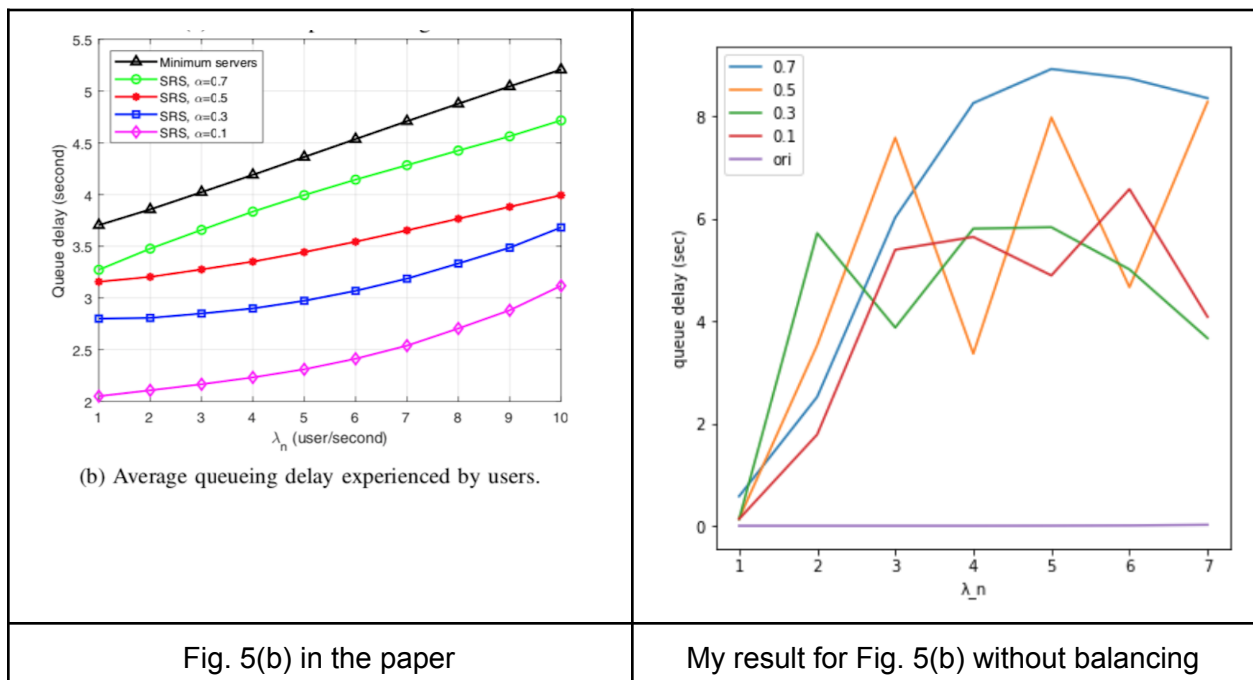
Step 5

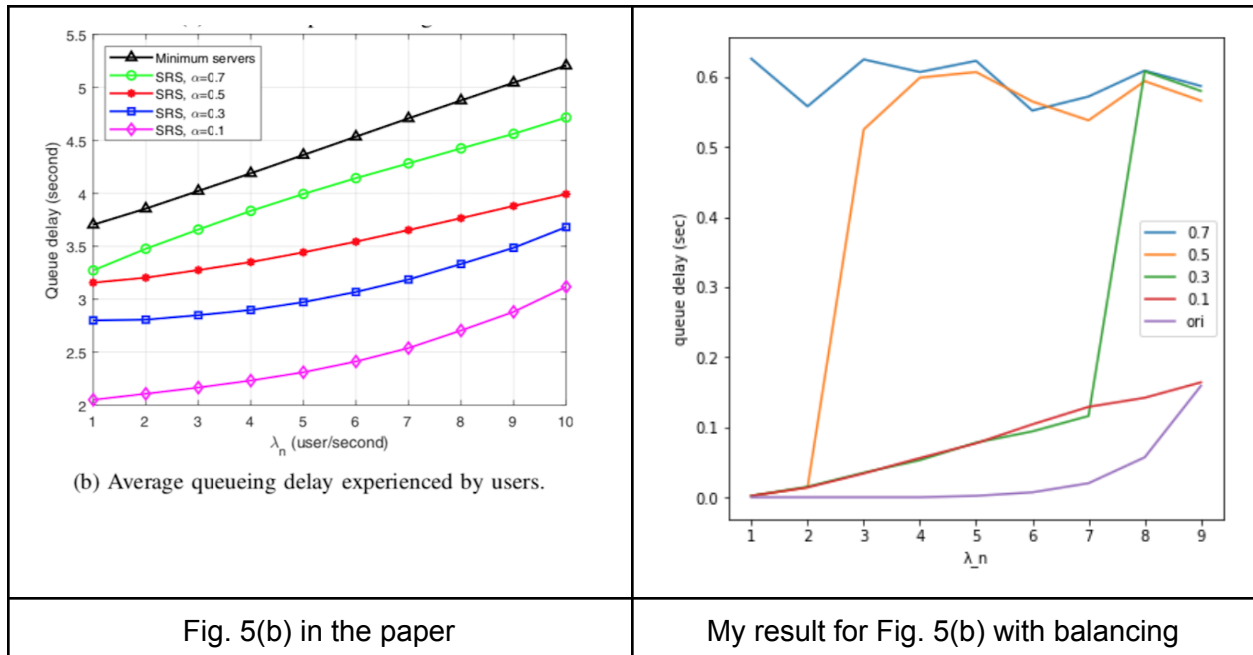
Finally, we try to plot Fig. 5, which are power saving and queueing delay as a function of the average arrival rate at edge devices under different values of α .

First, for Fig. 5(a), we directly implement Algorithm 1 and get the result by calculating how many edge devices can be turned off. The power saving ratio is calculated by the number of “off” devices / number of total devices. The values 0.1, 0.3, 0.5, 0.7, 1.0 of the lines mean different α , and lines that are very close should actually have the same value (since I add small values to each line to make it easier to be distinguished).



Then, for Fig. 5(b), we use Algorithm 1 to run different λ_n and different α to get results with a **number of “on” edge devices** and a **list of probability distributions for those “on” devices**. Collected these numbers and lists as the input of the C code in class. We slightly modify the code and make it be suitable for N stations, each station has k_n servers. However, we find out that the results for queue delay are NOT that similar to the original paper.





Conclusion and Discussion

The two algorithms mentioned in this paper are quite interesting and I've learned some new knowledge such as the square-root staffing rule and the DTMC-based algorithm similar to that carried out by web page ranking algorithms used by search engines during reading and implementing this paper. Before running the simulation, I thought those parameters in Table I are enough to reproduce a similar result as the original paper. However, there are still some parameters that the authors did not mention, such as k_n , total simulation time, as well as the setting for λ_n (the same for all devices? Or just a mean value of all the devices?). And the difficulty of implementing the simulation program is lower than I expected (just need to modify the code in class). That is, I should be more careful when choosing the paper I want to implement next time.

Demo

1. Run the file "term_project.py" can get all the results (all figures and numbers in this report since the simulation results are already provided)
2. If you want to change the setting and re-run the simulation for Fig. 5(b), copy the numbers and lists showed in terminal to "terms_old.in" (without balancing) or "terms.in" (with balancing), then compile and run the c code "term.c". The "term_project.py" will read "terms_old.out" and "terms_.out" to plot Fig. 5(b).