Project Progress Report: Survey of Mobile Edge Computing Networks

Apeksha Hada*, Dhruv Gupta*, Shailesh Kumar Jha*

* Department of Computer Science, Georgia State University

I. INTRODUCTION

With the advent of technologies like IoT, 5G and artificial intelligence we need better computing technology for faster processing. The need of the hour has become to not only have bigger pipes for data transfer but also address emerging concerns such as addressing network delays. New network architecture are being proposed to address some of these concerns.

Cloud computing has become the flavor of the season, but with increasing data volumes, latency and congestion has become a mammoth issue which is road block to new innovations on the application layer. One such step in this direction is the onset and division of some computation geared towards the mobile BS (Base Stations) or edge clouds. Edge clouds are resource constrained servers deployed close to the mobile base stations which can perform computation closer to the user instead of routing the request all the way to the remote server. This helps in reducing and helps addressing the issue of congestion in the long run. Edge cloud servers can be reached by nearby mobile users via wireless connections.

In this way, mobile devices can offload their tasks to edge clouds and receive the computing results with low network latency. However, because these specially deployed servers are of smaller scale than the remote ones, the resource and computation abilities of edge-clouds are relatively constrained and pose a scalability challenge that we look to explore further in this paper

II. MOTIVATION

Mobile Edge Computing (MEC) is like a cloud server which is running at the edge of a mobile network. Better performance of MEC is key to the future because growth in Mobile traffic will augment in the future as the Mobile devices becomes smarter and the end user is drawn to use the Mobile device for most of their daily tasks, which were earlier handled by Personal Computers.

Practical Applications of MEC:

1. MEC is the key player in implementation of 5G. MEC will reduce the latency for 5G networks.
2. Increased use of live streaming (YouTube, Netflix) on the mobile devices, can lead MEC to reduce the latency.
3. Augmented Reality.
4. Autonomous Vehicles can benefit from the MEC, as low latency is the key to operating autonomous vehicles.

III. DETAILED PROBLEM STATEMENT

The world has been moving from bigger to smaller devices. We now live in the age of the smart phones. The most important focus has been on mobility of such devices so that computation and data processing become very accessible.

With the onset of cloud computing many companies off shoulder their overhead of maintaining huge infrastructure which was seldom used at its full capacity. Instead, it is now managed by centralized service providers with end clients now focusing more on their central business and landscape without having to worry much about the maintenance or its background nuances. To reach these servers, mobile devices connect to their nearest base stations and their requests are routed to the nearest server. Latency is still an issue with the centralized data servers even though they are accessible from everywhere. The latency in data retrieval has always been a problem with this service and hence the concept of mobile edge computing came into existence.

New mobile applications with a low latency requirement such as self-driving cars are on the rise and end clouds are being proposed to meet this low latency requirement. A small-sized server cluster hosting a mobile micro-clouds (MMC) is directly connected to a network component at the network Edge providing cloud services. This poses the challenge of being able to process the request and return a response in a tight bound latency requirement. The following issues need to be addressed in such a requirement:

- Dispatching the request to a server which has low processing delay (d_{proc})
- Dispatching to an edge cloud vs. remote cloud (d_{trans})
- Scheduling requests at a server while keeping fairness (d_{queue})
- Service migration of servlets from base station to base station
- Placement of servlets in the edge cloud to optimally reduce latency

The main goal of this project is to study the newly proposed edge server architecture, existing models to optimize deployment, identify issues and provide solutions. To that end, the problem areas identified include optimization in scheduling and dispatching, implementation of fair queuing on edge servers without overwhelming them.

The existing models that are being studied currently are the On-Disc Model and the OREO model. The short comings identified include

- Lack of fair queuing in the scheduling model
- Requiring a dynamic programming model / Integer Linear Programming model for each edge server to be calculated on the dispatcher
- Using an online algorithm to address this issue, [4] suggests that online algorithms are not a good fit for this issue and thus comparing the implementations is an open problem

Another aspect of issue of mobility is the fact that mobile users move frequently and rather quickly. This requires the tasks sent to the cloud to move with them. This is achieved by porting the cloudlet serving the application to be sent to a new base station b' closer to the user. This model is formulated using a Markov decision process (MDP) and was not considered in the works on online algorithms so far. Studying the application of this process to the on Disc model is an open problem. Another aspect would be see the application of a 'transport factor´ like the one used in Page Rank in the residual density calculation used by On Disc in scheduling tasks.

It is also worth looking into other alternatives for the dispatching including the use of a load balanced edge cloud.

We propose to study these issues and survey existing solutions to find an optimal way of addressing the short comings in these implementations.

IV. RELATED WORK

In a recent paper related to mobile edge computing, they have proposed a novel idea of OREO (Online service caching for of Mobile Edge Computing) algorithm. They use this to perform a stochastic service caching without requiring future information. The OREO algorithm is based on Lyapunov optimization and a variation of Gibbs sampling.

In this, the time is divided into discrete time slots each of which has a duration matching the time at which service caching decision can be updated. Service is an abstraction of the application that is hosted by the BS and requested by the mobile users. The demand predictor with the help from machine learning algorithms like autoregression analysis can predict the instantaneous demand prior to the beginning of time slot.[2]

They have also considered the energy consumption for different service caching and task offloading. The CPU speed automatically adjusts based on the task workload. They convert the long-term optimization problem into per-slot optimization problem requiring only current slot information. Their algorithm helps the BS to decide which services should be cached and how much task should be offloaded at the edge or moved towards the cloud.

Their simulation on an area of 500m X 500m with 9 base stations was compared with three benchmarks – Non cooperative service caching, Centralized delay-optimal service caching, and Myopic service caching. Based on the results, they found that OREO achieves near-to-optimal delay performance while closely following the long-term energy constraint. OREO also slightly sacrifices the delay performance to satisfy the energy.

According to 'Mobile-Edge Computing – Introductory Technical White Paper', Traffic Offload Function (TOF) is a part of MEC application-platform services, which addresses only problem of prioritizing traffic and routing it.
Computation Offloading is one of the major topics of research in Edge/Cloud computing, with various research papers talking about different aspects of job offloading to Edge/Cloud servers.

This paper presents a mathematical model to calculate the computation offloading cost (time and energy consumption) of mobile cloud application models. This paper takes more detailed approach to address the problem of Computation Offloading and breaks it down to two separate issues: dispatching and scheduling problem. The above papers focus more on context-awareness of the jobs being sent by the mobile device. In this paper they consider that the jobs are arbitrary.
That is why here Latency Sensitivity is considered as deciding factor for scheduling jobs. The algorithm On-Disc is said to be scalable and distributed [1].

The two parts of the algorithm:

• **Scheduling Policy:**
Highest Residual Density First (HRDF) rule is used to schedule all unfinished jobs. In case of tie, job that arrive first has high priority.
Residual density is (weight of the job)/ (processing time). Jobs take the order in the queue based on their Residual density. If two jobs have same Residual density, the job which arrived first will be processed.

• **Dispatching Policy:**
According to this policy the jobs are dispatched to the server which brings least increase to Weighted Response Time (WRT), i.e., dispatch the jobs greedily to server with minimum total Weighted Response Time (WRT).

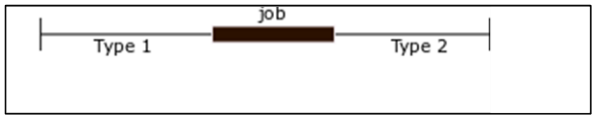

Fig. 1. Shows Type I and Type II jobs and 'job' is the job to be dispatched.

Weighted Response Time=
Time to process packets of higher Priority (Type I jobs) + Time to process job + Time to Process Low Priority jobs (Type II jobs)

In other literature, offline algorithms have been proposed with improved running time [9] which stands in contrast to the online models discussed earlier.

V. PLAN OF ATTACK

We are currently in the evaluation phase of the current work in the space of mobile edge computing and the caching and offloading of data requests to bring the optimal solution to latency and faster data processing.

Our division of work is streamlined as follows:

Shailesh Kumar Jha:

1) Analysis of current work: Identify how to simulate the OnDisc algorithm on a network

Apeksha Hada

2) Comparative study of various MEC (Mobile Edge Computing) algorithms.
3) *Review literature survey [7]*

Dhruv Gupta

4) Proposal of an alternate model
5) improvement on the current processing algorithms and its possible implementation

VI. MILESTONES

Our current approach is based on the following milestones:

Week 1: Analysis of the current work

Week 2: Comparative Analysis of the various approaches.

Week 3: Discussion on the possible improvement / new approach.

Week 4: Strategy on the path of the new possibility.

Week 5: Implementation feasibility discussion of the approach

Week 6: Implementation results if possible and their comparison.

VII. OUR IMPROVEMENT ON 'ON DISC'

We found issue with current approach related to the fairness in 'On Disc' scheduling algorithm. As per our analysis, we discovered that wait time low priority jobs, in case of constant incoming packets, will be very high.

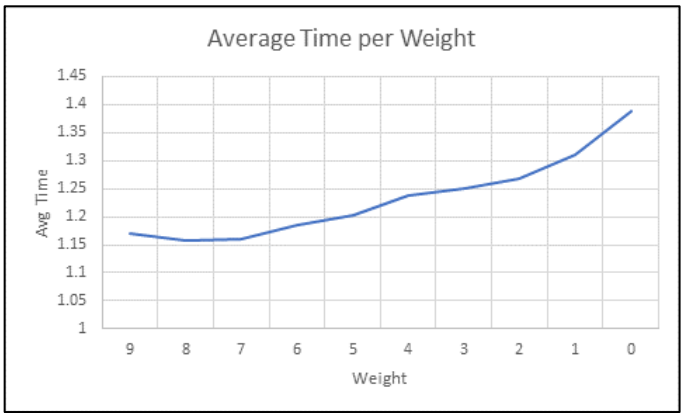

Fig. 2. Shows low priority packets very high processing times in the queue.

If high priority jobs keep coming in the low priority jobs may have to wait infinitely in the queue. In order to solve this problem, we suggest implementing 'Weighted Fair Queue (WFQ)' at the edge-servers. This can address the infinite waiting time of the low priority jobs.

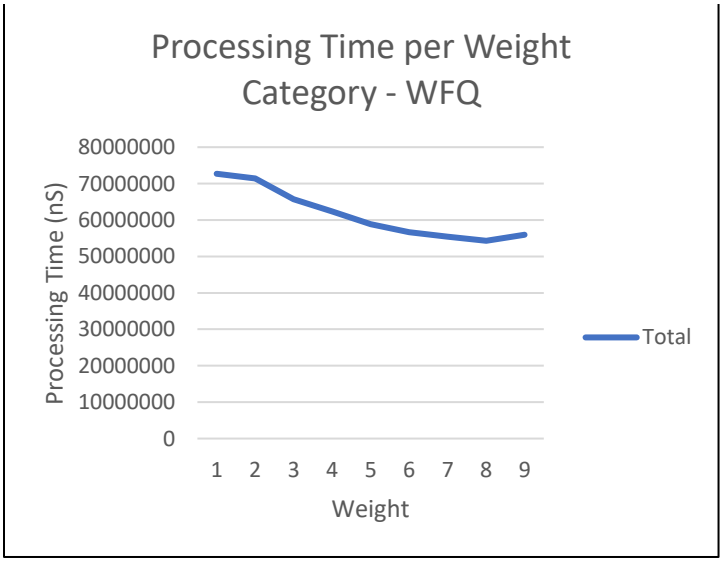

Fig 3. Graph of Processing times vs. Weight of the jobs

Row Labels	Count of duration
0	2047
1	1971
2	1999
3	2052
4	1989
5	2007
6	2082
7	1946
8	1930
9	1976
Grand Total	19999

Fig 4

Fig 3 & 4. Shows <<to be filled>>

VIII. Result Comparision

We were able to replicate 'On Disc' scheduling algorithm and compare it with our modification of it, using Weighted Fair Queue (WFQ). We see a lot of improvement in performance when compared to 'On Disc' scheduling algorithm.

Having different queues based on the weights of the jobs give low priority jobs to be processed without waiting for infinite times.

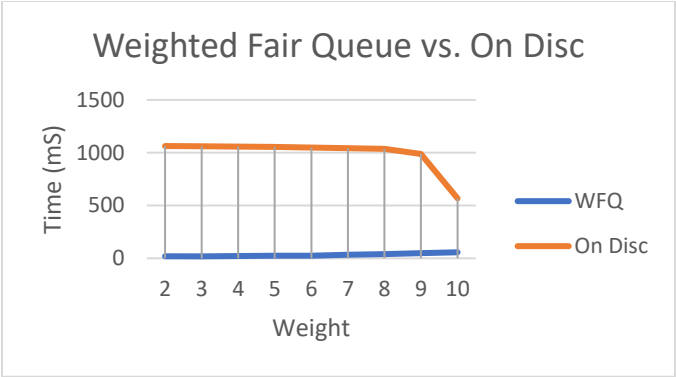

Fig 5. Shows a clear comparison graph between On Disc and Weighted Fair Queue algorithm.

As shown in Fig 5., the processing times for low priority jobs, with weights 2,3 or 4, in On Disc is very high. Whereas, in Weighted Fair Queue (WFQ), the processing times of low priority jobs is almost same as the high priority ones, with weights 7,8 or 9.

This improves the waiting times of the jobs in the queue and also improves fairness in the queue. While 'On Disc' scheduling policy suggests giving priority to high weight jobs, the low weight jobs will have almost infinite wait times in case of continuous incoming traffic. 'Weighted Fair Queue' solves this problem.

References

[1] Haisheng Tan, Zhenhua Han, Xiang-Yang Li, Francis C.M. Lau, Online job dispatching and scheduling in edge-clouds, IEEE INFOCOM 2017

[2] Jie Xu, Lixing Chen, Pan Zho, Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks, pp.207–215, IEEE 2018.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," arXiv preprint arXiv: 1701.01090, 2017.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[5] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590–3605, 2016.

[6] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5g ultra-dense cellular networks," IEEE Wireless Communications, vol. 23, no. 1, pp. 72–79, 2016.

[7] Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys & Tutorials*, *19*(3), 1628-1656.

[8] Wang, Shiqiang, et al. "Dynamic service placement for mobile micro-clouds with predicted future costs." *IEEE Transactions on Parallel and Distributed Systems* 28.4 (2017): 1002-1016.

[9] Urgaonkar, Rahul, et al. "Dynamic service migration and workload scheduling in edge-clouds." *Performance Evaluation*91 (2015): 205-228